ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 2, Issue 3, September 2012

A Comparative study of Stream Data mining Algorithms

Tusharkumar Trambadiya, Praveen Bhanodia

Abstract*— The problem to extract knowledge from large raw data has emerged as a new data structure. Data stream is a new era in data mining. Numerous algorithms are used for processing & classifying data streams. Traditional algorithms are not appropriate to process data stream which in cause generate problems regarding classification. A model which is developed from stream data for classification must update incrementally after the fresh arrival of new data. Data stream classification performance can be measured by various factors such as accuracy, computational speed, memory and time taken for processing. Data stream classification algorithm must have to meet certain requirements and measures to handle continuous flow of stream data. These algorithms get less time span to inspect data and build model, may be only once with less amount of resource, time and prediction. So to study the concepts of classification algorithms will lead towards development of better approaches for stream data mining. In this paper, we make a comparative study between Hoeffding tree, VFDT (Very Fast Decision Tree) and CVFDT (Concept Adaptinf Very Fast Decision Tree) from various algorithms which are used for stream data classification.

***Index Terms*— Concept Drift, CVFDTGrow, Decision trees, Hoeffding Bounds, Incremental Learning.**

I. INTRODUCTION

Recently, the increasing applications of data streams has led to the study of stream data, which is an important technique that is essential to a wide range of emerging applications, such as website log, e-business and stock market analysis and sensor networks. Various algorithms for mining a stream data do not fit in primary memory due to lack of resources. Traditional algorithms were only tested on few million examples but in today's scenario everyday millions of data are generated like transactions, millions of ATM, credit card operations and popular Web sites logs. This creates a massive amount of data. So for this type of large data current data mining systems are not sufficient and equipped to deal with them. In most cases, massive amount of data can be mined for frequent and relevant pattern, which are used in various applications. When the amount of data is very large, it leads to a numerous computational and mining challenges due to shortage of hardware limitations. First, because of large volume of the data, it is not easy to process the data efficiently by applying multiple passes of one algorithm. Sometimes, the algorithm gets only one chance to process the data. This leads to number of problematic situations for traditional algorithms. Therefore, algorithms which are specifically design for stream data mining must follow certain instructions so that they work with one pass of data. Second, in stream data mining, data may evolve over time so temporal factor of data must be taken into consideration in the design of algorithm. This nature and behaviour of data streams is referred to as *temporal locality*. So one-pass mining algorithms may not be an effective solution for this type of data and task to generate knowledge. There are number of application characteristics for stream data such as large amount of data (in terabytes) and where data arrive at a rapid rate. For such type of data, there is a need for near-real time analysis of data feeds. So Stream mining algorithms need to be carefully designed with a vision on the evolution of the underlying data. "A data stream is a real-time, continuous, ordered (implicitly by arrival time or explicitly by timestamp) sequence of items. It is not possible to control the order in which data item arrive, nor is it feasible to locally store a stream in its entirety."

A. THEORETICAL ISSUES IN STREAM DATA MINING

Stream data mining faces an issue which is lack or limited amount of computation resources to generate frequent data sets or patterns. Number of times, the computation power and primary memory can't handle massive amount of data in the input stream. For example, everyday, more than 130 million people use Google, Amazon serviced 20 million book searches and transactions, and Vodafone generated 5 million call records. Traditional data mining algorithms work on the assumption that they will have sufficient resources to process particular data. This assumption does not have any chance in data stream mining due to continuous evolvement of new data. Every Stream data mining algorithms shall take less time to learn underlying data with few amount of memory. Data is no longer static, but rather a continuous temporal stream. For effective generation of knowledge, stream mining must be adaptive to change for new data. For example, when user surf various products on online shopping site, according to user's different purchasing patterns, the data is generated and future marketing strategies are decided in order to satisfy customer needs.

B. Data Streams Mining Problems

To store continuous data stream is a great challenge for storage devices. To generate pattern or knowledge from stream data, algorithms with different techniques are needed. We don't have enough amount of space to store stream data and problem occurs between accuracy of data pattern and storage. So we can classify into five categories as shown in table 2. [4]

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 2, Issue 3, September 2012

Table I .Problems in Data Stream Mining

Traditional data mining	Stream data mining
• Whole data set require to generate frequent pattern • Due to static data multiple passes are allowed • More time to access particular data	• Impossible to store whole data • Due to continuous arrival of new data multiple passes are not possible • Less time to access data sometime only once

Table II. Classification of Challenges via Category

No.	Issues	Challenges	Solution approach for these issues
1	Memory management	Data arrival rate and variant data arrival rate over time are irregular and fluctuated	Summarization techniques
2	Data Pre-processing	Quality of mining result and automation of pre-processing techniques	Light-weight pre-processing techniques
3	Data Structure	Limited memory size and large volume of data stream	Incremental maintaining of data structure, novel indexing, storage and querying techniques
4	Resource	Limited resource like storage and computation capabilities	AOG
5	Visualization of results	Problem in data analysis and quick decision making by user	Still is a research issue(one of the proposed approach is: intelligent monitoring)

II. CLASSIFICATION ALGORITHMS

Various algorithms are available for data stream classification. Based on data mining tasks some algorithms and approaches for classification of data stream are defined below.

- Hoeffding tree algorithm that is based on decision tree.
- GEMM and FOCUS algorithm and that mining task is decision tree and frequent item set.
- OLIN algorithm uses info-fuzzy techniques for building a tree-like classification model.
- VFDT (Very Fast Decision Tree) and CVFDT (Concept-Adapting Very Fast Decision Tree) algorithm works on decision tree.
- On-demand stream classification, using cluster ideas that each cluster ids associated with specific class label which defines the class label of the points in it.

So in this paper we discuss the Hoeffding algorithm in next section IV. Very fast decision tree algorithm (VFDT) in section V and Concept adapting very fast decision tree algorithm (CVFDT) in VI with their advantages and disadvantages.

III. HOEFFDING TREE

In Hoeffding algorithm, classification problem must be defined. Classification problem is a set of training examples of the form (a, b), where a is a vector of d attributes and b is a discrete class label. Our goal is to produce a model b=f(a)such that it provides and predicts the classes y for future examples x with high accuracy. Decision tree learning is considered one of the most effective classification methods. By recursively replacing leaf node with test nodes, starting at the root we can learn a Decision trees. In decision tree each node has a test on the attributes and each branch gives possible outcome of the test and each leaf contain a class prediction. Before processing starts, data is first stored into main memory. After starting learning process for complex trees it is expensive to repeatedly read data from secondary memory so our aim is to design decision tree learners than read each example at most once, and use a small amount time to process it. First key role is to find the best attribute at a node and for that consider only some training examples that pass through that nodes. Second choose the root and then expensive examples are checked down to the corresponding leaves and used to choose the attribute there, and so on how many examples are required at each node is decided by Hoeffding bound after continuous use. Take a random variable a and its range is R. We have n observation of a. Now find mean of a ($\bar{r}$), so Hoeffding tree bound states that with probability 1-δ, the true mean of a is at least $\bar{r}$- ε. where Hoeffding bound ε

$$\varepsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n}}$$

A. Hoeffding Tree Algorithm

Algorithm:

1. HT is a tree with a root node (the root)
2. for all training data do
3. Sort example into leaf l using HT
4. Update sufficient statistics in l
5. Increment nl, the number of examples seen at l
6. if nl mod Nmin = 0 and examples seen at l not all of same Class then
7. Calculate $\bar{G}_l$ (C_l) for each attribute factor
8. Let C_a be attribute with highest $\bar{G}_l$
9. Let C_b be attribute with second-highest $\bar{G}_l$
10. Compute Hoeffding bound $\varepsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n}}$

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 2, Issue 3, September 2012

11. if $C_a \neq C_\emptyset$; and ($\bar{G}_1$ (C_a) - $\bar{G}_1$ (C_b) > ε or ε < τ) then
12. Replace *l* with an internal node that splits on C_a
13. for all branches do
14. Add a new leaf with initialized sufficient statistics
15. End for
16. End if
17. End if
18. End for

In first line we have to initializes the tree data structure with a single root node. When Lines 2 to 18 performed on every training data example will create a loop. Each training data is filtered down inclementally tree to an appropriate leaf; this will depend on the different tests present in the decision tree which is used to build that point at line. This leaf is then updated in line 4. In tree each leaf node holds sufficient statistics needed to make decisions about further scope. To estimate the information gain when any attribute is split is done by sufficient statistics that are updated. Line 5 shows that that nl is the example count at the leaf, and it is updated. Logically nl can be calculated from the sufficient statistics. In the previous section we describe a test, it is performed in Lines 7-11, and whether a particular attribute has produce better result than other attributes is decided using the Hoeffding bound. To split statistics attribute G is the splitting criterion function and $\bar{G}$ is its estimated value. To test $C_\emptyset$ pre-pruning technique is required with using null attribute in Line 11. To check that there is any tie occurs is find out by test involving τ. After applied various tests, if an attribute has been provide better result than all other nodes than split the node, for growth of tree. [1]

B. Pros of Hoeffding tree algorithm

Hoeffding tree algorithm scales different attributes better than other traditional algorithms. It consumes less primary memory and provides better utilization with sampling of statistics. Second it provides incremental approach in the sense new examples are added as they come and also they work in parallel.

C. Cons of Hoeffding tree algorithm

Hoeffding tree algorithm waste computational speed due to spend lot of time in checking that ties are occurs or not. More Memory required with the growth of tree expansion.

IV. VERY FAST DECISION TREE (VFDT) ALGORITHM

VFDT algorithm use basic fundamental of a decision-tree learning which is based on the Hoeffding tree algorithm. In VFDT algorithm, threshold value is specified by user. Best statistics attribute split, if the value of split attribute is less than a user-specified threshold. Because it's decided identical attribute is useful or not. So simply compute G and use split to find best attribute periodically. After comparing statistical analysis there may be chance to drop less useful leaves when needed as well as it rescan old data when time available. To keep counts for all leave nodes memory is required, this have a higher priority than VFDT's memory use. If j is the number of attributes, n is the maximum number of values per attribute, and c is the number of classes, VFDT requires O(jnc) memory to store the counts at each stage of leaf nodes. If l is the number of leaves in the tree, the total memory required is O (ljnc). This is neither dependent on how many number of examples we have seen nor training data set size.[2] Pruning play key role in Hoeffding as well as in VFDT algorithm. Pruning is classified into three categories: 1.no-pruning: - In this option VFDT refines tree indefinitely 2.pre-pruning:- In this option VFDT consider one more parameter 'τ', whenever all others 3.post pruning, VFDT supports all three kind of pruning option. τ', and stops growing any leaf where the difference between the best split candidate all others is less than G (.) and G (.) < τ'. At certain point if all of the leaves in the decision tree are pre-pruned, than VFDT algorithm terminates. VFDT is work on incremental approach, whenever a new example come it merged with old data. So after applying this algorithm on few examples a usable model is available after the first few examples and then progressively defined.

A. Very Fast Decision Tree (VFDT) Algorithm

Algorithm:

Input: δ desired probability level.

Output: τ a Decision Tree.

In it: τ ← Empty Leaf (Root)

1. While (TRUE)
2. Read next example
3. Propagate Example through the tree from the root till a leaf
4. Update sufficient statistics at leaf
5. If leaf (number of examples) > N_{min}
6. Evaluate the merit of each attribute
7. Let A_1 the best attribute and A_2 the second best
8. Let $\varepsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n}}$
9. If G (A_1)-G (A_2) > ε
10. Install a splitting test based on A_1
11. Expand the tree with two descendant leaves

Some changes are done in this algorithm that is defined below:

Sometimes more than one attributes have similar attribute values in the sense of G's than choose best attribute is quite critical. We have to decide between them with high confidence so VFDT can decide solve this problem that there is effectively a tie and split on the current best attribute if difference between the best split candidate all others is less than G(.) and G(.) < τ. where τ is a user-defined threshold.[3]

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 2, Issue 3, September 2012

G computation: inefficient to calculate G for every new data set, because at specific point it is hard to split best attributes. Users specify minimum amount of new data examples N_{min} that must be calculated at a leaf before G is computed. This mechanism incrementally takes less amount of global time which was spent on G computations. VFDT making learning work as fast as classify data sets. [3] VFDT is better than Hoeffding tree in terms of time, memory and accuracy. In this example memory limit is same for VFDT and C4.5 is 40MB. VFDT setting=10^{-7}, N_{min}=200 and τ=5%, Domain:2 classes and 100 binary attributes, fifteen synthetic trees 2.2k-500k leaves. So running time for c4.5 is takes 35 seconds to read and process 100k examples and VFDT takes 47 seconds. In second experiment VFDT takes 6377 seconds for 20 million examples. So in VFDT take advantage after 100k to greatly improve accuracy. Concept of drift is not handles in VFDT. Means as time goes by, Novice literature becomes experts and students become engineer so after some minimum permanence time span data is shift time to time.

V. CONCEPT ADAPTINF VERY FAST DECISION TREE (CVFDT) ALGORITHM

VFDT is included in knowledge data discovery assume training data is a sample drawn from stationary distribution. Data stream not considered this assumption because of concept drift. So to continuously change data stream is our only goal. CVFDT is an extended version of VFDT which provides same speed and accuracy advantages but if any changes occur in example generating process provide the ability to detect and respond. Various systems with this capability (Widmer and Kubat, 1996, Ganti et al., 2000), CVFDT uses sliding window of various dataset to keep its model consistent. In Most of systems, it needs to learn a new model from scratch after arrival of new data. Instead, CVFDT continuous monitors the quality of new data and adjusts those that are no longer correct. Whenever new data arrives, CVFDT incrementing counts for new data and decrements counts for oldest data in the window. The concept is stationary than there is no statically effect. If the concept is changing, however, some splits examples that will no longer appear best because new data provides more gain than previous one. Whenever this thing occurs, CVFDT create alternative sub-tree to find best attribute at root. Each time new best tree replaces old sub tree and it is more accurate on new data.

A. CVFDT Algorithm

1. Alternate trees for each node in HT staret as empty.
2. Process Examples from the stream indefinitely.
3. For Each Example (x, y),
4. Pass (x, y) down to a set of leaves using HT And all alternate trees of the nodes (x, y) pass Through.
5. Add(x, y) To the sliding window of examples.
6. Remove and forget the effect of the oldest Examples, if the sliding window overflows.
7. CVFDTGrow
8. Check SplitValidity if f examples seen since Last checking of alternate trees.

9. Return HT.

It is generally occurs in VFDT algorithm, but CVFDT continuously monitors the validity of its old decisions, by maintaining more than sufficient statistics at every node in Decision tree. As decision tree has grown Forgetting an old example is slightly complicated by the fact that DT may have grown or changed since the example was initially incorporated. To avoid forgetting an example from a node that has never seen it, nodes are assigned a unique, monotonically increasing ID as they are created. After addition of an example to W, the maximum ID of the leaves it reaches in DT and all alternate trees is recorded with it. An example's effects are forgotten if the example whose ID is less than or equal to stored ID by decrementing the counts in the sufficient statistics. CVFDTGrow: In CVFDTGrow, for each node reached by the example in Hoeffding Tree Increment the corresponding statistics at the node. If sufficient examples seen at the leaf in HT which the example reaches, Choose the attribute that has the highest average value of the attribute evaluation measure (information gain or gini index). If the best attribute is not the "null" attribute, create a node for each possible value of this attribute.[11] Forget Old Example: Maintain the sufficient statistics at every node in Hoeffding tree to monitor the validity of its previous decisions. VFDT only maintain such statistics at leaves. Than HT might have grown or changed since the example was initially incorporated. It will assign unique increasing ID as they are created. After each node reached by the old example with node ID no larger than the max leave ID the example reaches, Decrement the corresponding statistics at the node and For each alternate tree Talt of the node, forget (Talt, example, maxID).[11]

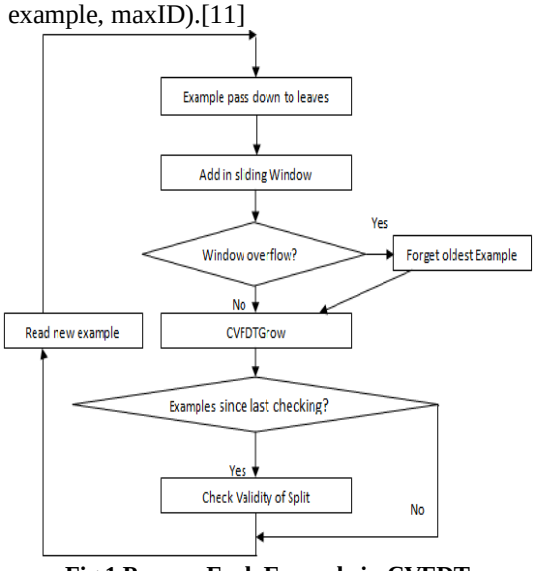

Fig.1 Process Each Example in CVFDT

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 2, Issue 3, September 2012

Check Split Validity: Validity is periodically checks the internal nodes of HT in split. When a new best attribute is found split generate new alternate tree. We assumes two approaches to comparing CVFDT and VFDT, First Set limit for number of alternate trees generation and second is to increase criteria to avoid tree generation. CVFDT is perform outrageous than any other algorithms which are used for data stream classification, we can prove this by performing number of experiments. Varying levels of drift and ability to scale the continuous data stream is evaluated by CVFDT and VFDT. If rotating plane is continuously change than synthetic data is generated and this data is used in these experiments.

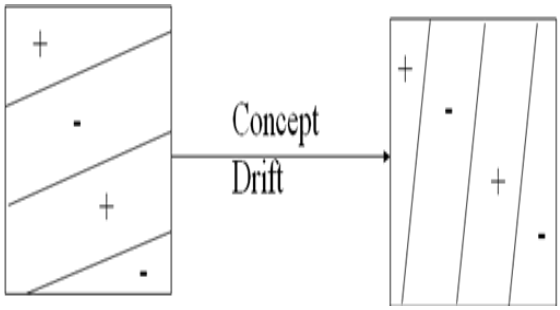

Fig.2 Concept Drift

In experiment 5 million training examples, drift inserted by periodically rotating hyper planes, about 8% of test points change label each drift, 100,000 examples in window,5% noise and results sampled every 10k examples throughout the run and averaged. Figure 3 compares the accuracy of the algorithms as a function of d, the dimensionality of the space. The reported values are obtained by testing the accuracy of the learned models every 10,000 examples throughout the run and averaging these results. Drift level, reported on the minor axis, is the average percentage of the test set that changes label at each point the concept changes. CVFDT is substantially more accurate than VFDT, by approximately 10% on average, and CVFDT's performance improves slightly with increasing d (from approximately 13.5% when d = 10 to approximately 10% when d = 150). [6]

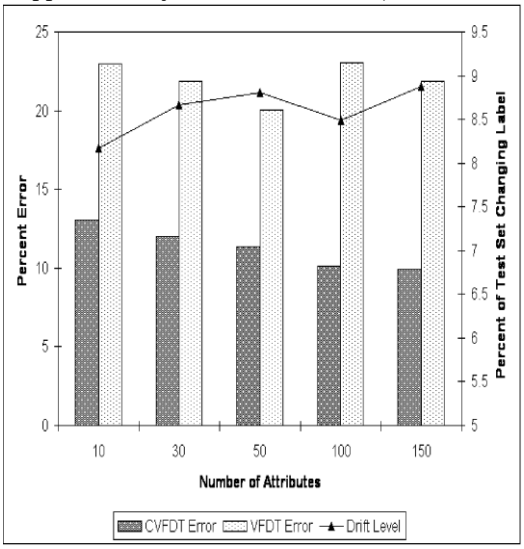

Fig.3 Error Rate vs. number of Attributes

Figure 4 compares the average size of the models induced during the run shown in Figure 11 (the reported values are generated by averaging after every 10,000 examples, as before). CVFDT's trees are substantially smaller than VFDT's, and the advantage is consistent across all the values of d we tried. This simultaneous accuracy and size advantage derives from the fact that CVFDT's tree is built on the 100,000 most relevant examples, while VFDT's is built on millions of outdated examples.

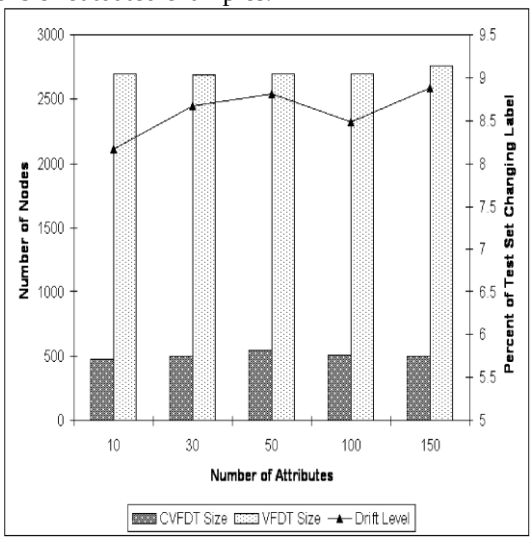

Fig.4 Tree size vs. number of Attributes

VI. CONCLUSION

In this paper we discuss about theoretical and practical problems which are often occurs in stream data mining classification algorithm. In these classification algorithms, Hoeffding trees spend small amount of time for learning. Hoeffding tree do not show any similarity with batch trees. In real world scenario we have a limited amount of hardware resources, despite of this it analyze and generate results with high result. In data mining Systems VFDT based on Hoeffding trees. Time-variant data is used to extend VFDT to develop the CVFDT system. Flooding is used to keep trees up-to-date with time variant data streams.

REFERENCES

[1] Albert Bifet, Geoff Holmes, Richard Kirkby and Bernhard Pfahringer (May 2011) Data Stream Mining a Practical Approach.

[2] Alexey T Symbal, (2004), The problem of concept drift: definitions and related work, Department of Computer Science, Trinity College Dublin, Ireland.

[3] Dariusz Brzezinski (2010), Mining Data Streams With Concept Drift, Poznan University of Technology.

[4] Aggarwal, C., Han, J., Wang, J., and Yu, P.S., (2004): On Demand Classification of Data Streams. In Proceedings of 2004 International Conference On Knowledge Discovery and Data Mining (KDD '04). Seattle, WA.

[5] Agrawal, C.C. (2007). Data Streams: Models and Algorithms. Springer.

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 2, Issue 3, September 2012

[6] Domingo's, P. and Hulten, G., (2000): Mining High-Speed Data Streams. In Proceedings of the Association for Computing Machinery Sixth International Conference on Knowledge Discovery and Data Mining.

[7] Babcock, B., Babu, S., Deter, M., Motwani, R., and Widom, J., (2002): Models and issues in data stream systems. In Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS). Madison, Wisconsin, pp. 1-16.

[8] R. Agrawal and G. Psaila. Active data mining. In Proceedings of the First International Conference on Knowledge Discovery and Data Mining, pages 3{8, Montr_eal, Canada, 1995. AAAI Press.

[9] Khaled Alsabti, Sanjay Ranka, and Vineet Singh. CLOUDS: A decision tree classifier for large datasets. In Knowledge Discovery and Data Mining, pages 2{8, 1998.

[10] P. L. Bartlett, S. Ben-David, and S. R. Kulkarni. Learning changing concepts by exploiting the structure of change. Machine Learning, 41:153{174, 2000.

[11] Kirankumar Patel: Review on Data Stream classification. In Proceedings of the International conference on Computing and Information Technology, pages 35{13, tirupati, India, 2012.

[12] A. Arasu, B. Babcock. S. Babu, M. Datar, K. Ito, I. Nishizawa, J. Rosenstein, and J. Widom. STREAM: The Stanford Stream Data Manager Demonstration description - short overview of system status and plans; in Proc. of the ACM Intl Conf. on Management of Data, June 2003.

[13] B. Babcock, M. Datar, and R. Motwani. Load Shedding Techniques for Data Stream Systems (short paper) In Proc. of the 2003 Workshop on Management and Processing of Data Streams, June 2003.

Author's Profiles

Tusharkumar J. Trambadiya was born in Gujarat, India in August, 1989. He received his bachelors of Engineering in Information Technology from North Maharashtra University, Jalgaon, Maharashtra (India) in august 2010 and is currently pursuing his Master of Technology in Computer Science Engineering from Patel Institute of Technology, Rajiv Gandhi Proudyogiki Vishwavidyalaya, Madhya Pradesh (India).His research interests include Stream Data mining and Wireless Networks.

Mr. Praveen Bhanodia is a Head of Computer Science Engineering at Patel College Of Engineering & Technology at Indore, Madhya Pradesh, India. He received his Bachelors from Shri Govindram Seksariya Institute of Science and Technology, Indore and completed His master from Rajiv Gandhi Technical University, Bhopal, India.