Computer Engineering and Intelligent Systems
ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)
Vol 3, No.7, 2012

Mining Data Streams using Option Trees

B. Reshma Yusuf* Dr. P. Chenna Reddy
JNTUA College of Engineering, Pulivendula, Andhra Pradesh, INDIA
* E-mail of the corresponding author: resh.yusuf@gmail.com

Abstract

Many organizations today have more than very large databases. The databases also grow without limit at a rate of several million records per day. Data streams are ubiquitous and have become an important research topic in the last two decades. Mining these continuous data streams brings unique opportunities, but also new challenges. For their predictive nonparametric analysis, Hoeffding-based trees are often a method of choice, which offers a possibility of any-time predictions. Although one of their main problems is the delay in learning progress due to the presence of equally discriminative attributes. Options are a natural way to deal with this problem. In this paper, Option trees which build upon regular trees is presented by adding splitting options in the internal nodes to improve accuracy, stability and reduce ambiguity. Adaptive Hoeffding option tree algorithm is reviewed and results based on accuracy and processing speed of algorithm under various memory limits is presented. The accuracy of Hoeffding Option tree is compared with Hoeffding trees and adaptive Hoeffding option tree under circumstantial conditions**.**

Keywords: data stream, hoeffding trees, option trees, adaptive hoeffding option trees, large databases

1. Introduction

Data mining is the task of discovering interesting and hidden patterns from large amounts of data where the data can be stored in databases, data warehouses, OLAP (on line analytical process) or other repository information. At present the most efficient algorithms available are focused on making it possible to mine databases that do not fit in main memory by only requiring sequential scans of the disk. But these algorithms are tested only up to a few million examples. In many applications this is less than a day's worth of data. For example, every day, retail chains record millions of transactions, telecommunications companies connect millions of calls, large banks process millions of ATM and credit card operations, and popular Web sites log millions of hits. As the expansion of the Internet continues and ubiquitous computing becomes a reality, we can expect that such data volumes will become the rule rather than the exception. Current data mining systems are not suitable for such type of data.

A *data stream* [1] is an ordered sequence of instances with bounded main memory and data may be evolving over time at a higher rate than they can be mined. Even simply storing the examples for future use may be lost or corrupted and can become unusable when the relevant information is no longer available.

One of the tasks is to design a decision tree learner for extremely large (potentially infinite) datasets. Such a decision tree learner should require each example to be read at most once, and only a small constant time to process it. This will make it possible to directly mine online data sources (i.e., without ever storing the examples), and to build potentially very complex trees with acceptable computational cost. The problem of deciding exactly how many examples are necessary at each node is solved by using a statistical result known as the Hoeffding bound [2] (or additive Chernoff bound).

Consider a real-valued random variable r whose range is R (e.g., for a probability the range is one, and for an information gain the range is log c, where c is the number of classes). Suppose we have made n independent observations of this variable, and computed their mean r.

Hoeffding bound states that, with probability 1- δ, the true mean of the variable is at least r' – ε, where

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

Hoeffding-based tree learners have been recognized as the most efficient in terms of processing speed per example, although their learning might be slow, which results in lower any-time accuracy at the beginning. The Hoeffding trees tend to be less accurate in situations where several attributes appear to be equally discriminative. This problem is solved using option trees, which can include option nodes in addition to ordinary split nodes. The main motivation is that introducing option nodes removes the need for selecting the best splitting attribute. Main idea is to introduce options only when splitting decisions are ambiguous, which will avoid excessive and unnecessary tree growth and reduce memory consumption.

2. Related Work

Classification is one of the most familiar and most popular data mining techniques. It aims to find a function that can map each data item in dataset into one of several predefined classes. Many classification algorithms are available in literature but decision trees is most commonly used because of its ease of implementation and easier to understand compared to other classification algorithms.

Previous work on scaling up decision tree learning produced systems such as SLIQ [3], SPRINT [4]. These systems perform batch learning of decision trees from large data sources in limited memory by performing multiple passes over the data and using external storage. Such operations are not suitable for high speed stream processing.

Hoeffding tree an incremental decision tree algorithm proposed by Domingos and Hulten in the paper "Mining High Speed Data Streams". For streams made up of discrete types of data, Hoeffding bounds guarantee that the output model is asymptotically nearly identical to that of a conventional decision tree. The Hoeffding tree algorithm is the basic theoretical algorithm, while VFDT adopts several enhancement techniques for practical applications, such as grace period, pre-pruning, and tie breaking.

Option decision trees [5] are used to reduce the error of decision trees on real world problems by combining multiple options which is quite similar to that of voting algorithms that learn multiple models and combine the predictions. The main goal of the paper is to explore when option nodes are most useful and to control the growth of trees by which complexity of little utility is limited.

Option nodes are also used in the context of learning from data streams, as an extension of Hoeffding trees [6] for classification. Although Hoeffding trees are more stable than batch tree learners, decisions are still subject to limited lookahead. This is the main motivation of Pfahringer et al. Their approach is somewhat different from the proposed batch ones, mainly because option nodes are not introduced in the split selection process, but only after a node has been transformed into an internal (decision) node

Kirkby [7] proposed an Option Tree that allows each training example to update a set of option nodes rather than just a single leaf. Option nodes work like standard decision tree nodes with the difference that they can split the decision paths into several sub trees. Making a decision with an option tree involves combining the predictions of all applicable leaves into a single result.

3. Option Trees

Option trees are a single general structure making it possible to travel down multiple paths and arrive at multiple leaves. This is achieved by introducing the possibility of *option nodes* to the tree, alongside the standard decision nodes and leaf nodes.

An option node [4] splits the decision path several ways—when an option node is encountered several different sub trees are traversed, which could themselves contain more option nodes, thus the potential for reaching different leaves is multiplied by every option. Making a decision with an option tree involves combining the predictions of the applicable leaves into a final result. A potential benefit of option trees over a traditional ensemble is that the more flexible representation can save space.

Consider as an extreme example an ensemble of one hundred mostly identical large trees, where the only difference between each tree lies at a single leaf node, in the same position in each tree. The standard ensemble representation would require one hundred copies of the tree where only the leaf would differ. Efficiently represented as an option tree this would require almost a hundred times less space, where the varying leaf could be replaced by an option node splitting one hundred ways leading to the one hundred different leaf variants.

Adaptive Hoeffding option tree which is quite similar to Hoeffding option tree with several improvements was introduced later for accurate result calculation. Each leaf stores an estimation of the current classification error and the weight of each node in the major voting process which is proportional to the square of the inverse of the error.

4. Methdology

Massive Online Analysis (MOA) is a software environment for implementing algorithms and running experiments for online learning. It is designed to deal with the problems of scaling up the implementation of state of the art algorithms to real world dataset sizes and of making algorithms comparable in benchmark streaming settings. It is implemented in Java and contains a collection of data stream generators, online learning algorithms, and evaluation procedures.

4.1.Environments:

Three environments are simulated using memory limits, since memory limits cannot be ignored and can significantly limit capacity to learn from data streams. Potential practical deployment of data stream classification has been divided into scenarios of increasing memory utilization, from the restrictive sensor environment, to a typical consumer grade handheld PDA environment, to the least restrictive environment of a dedicated server.

4.1.1. Sensor Network

This environment represents the most restrictive case, learning in 100 kilobytes of memory. Because this limit is so restrictive, it is an interesting test case for algorithm efficiency.When memory limits are in the order of kilobytes, other applications requiring low memory usage also exist, such as specialized hardware in which memory is expensive.

4.1.2. Handheld Computer

In this case the algorithm is allowed 32 megabytes of memory. This simulates the capacity of lightweight consumer devices designed to be carried around by users and can easily fit into a shirt pocket. The ability to do analysis on site with a handheld device is desirable for certain applications.

4.1.3. Server

This environment simulates either a modern laptop/desktop computer or server dedicated for processing a data stream. The memory limit assigned in this environment is 400 megabytes. Considering that several algorithms have difficulty in fully utilizing this much working space, it seems sufficiently realistic to impose this limit.

4.2.Data generators:

MOA stream generators allow simulating potentially infinite sequence of data. The following are the generators used for generating data according to some pattern instead of reading data from file, database or any other data source.

4.2.1. Random Tree Generator

Random tree generator generates examples by assigning uniformly distributed random values to attributes which then determine the class label via the tree which choose attributes at random to split. The generator has parameters to control the number of classes, attributes and depth of tree. Two random trees were generated one is simple and the other complex.

The simple random tree (rts) has ten nominal attributes with five values each, ten numeric attributes, two classes, a tree depth of five, with leaves starting at level three and a 0.15 chance of leaves. The complex random tree (rtc) has 50 nominal attributes with five values each, 50 numeric attributes, two classes, a tree depth of ten, with leaves starting at level five and a 0.15 chance of leaves.

A degree of noise can be introduced to the examples after generation. The streams rtsn and rtcn are introduced by adding 10% noise to the respective random tree data streams.

4.2.2. Random RBF Generator

Random RBF generates examples from a fixed number of random centroids. Each time a centroid is selected at random, a data point around the centroid is drawn randomly. The centroid determines the class label of the example, and the data point determines the attributes of the example. Examples in Random Tree are generated by assigning random values to each attribute first and the class label is determined via a pre-constructed decision tree. Only numeric attributes are generated. Two random streams are produced one simple and other complex.

The simple RBF (RRBFS) generator has 100 centers and ten attributes where complex RBF (RRBFC) generator has 50 attributes and 1000 centers. Both have only two classes.

4.2.3. LED Generator

The generator actually generates stream predicting the digit displayed on a seven-segment LED display, where each attribute has a 10% chance of being inverted. It has an optimal Bayes classification rate of 74%. The particular configuration of the generator used for experiments produces 24 binary attributes, 17 of which are irrelevant.

4.2.4. Waveform Generator

This generator shares its origins with LED. The goal of the task is to differentiate between three different classes of waveform, each of which is generated from a combination of two or three base waves. There are two versions of the problem. Wave21 has 21 numeric attributes, all of which include noise. Wave40 introduces an additional 19 irrelevant attributes.

4.2.5. Function Generator

Function generator produces a stream containing nine attributes, six numeric and three categorical which was used as data source for work on scaling up decision tree learners. There are ten functions defined for generating binary class labels from the attributes. For the experiments the ten functions are used, with a perturbation factor of 5% (referred to as genF1-genF10). Perturbation shifts numeric attributes from their true value, adding an offset drawn randomly from a uniform distribution, the range of which is a specified percentage of the total value range.

5. Methdology

Wide variety of data sets is used for evaluation. The experiments are performed on a machine 2.40 GHz Intel Core 2 Duo processor, 2 GB RAM, running windows XP. The evaluation procedure of a learning algorithm determines which examples are used for training algorithm and which are used to test the model output by the algorithm. Prequential evaluation approach is used for the experiments. It provides a learning curve that monitors the evaluation of learning as a process. It is based on the idea that statistical methods should be assessed by means of validity of predictions that flow from them and that such assessments can usefully be extracted from a sequence of realized data values, by forming, at each intermediate time point, a forecast for next value, based on an analysis of earlier values.

The first, and baseline algorithm is a single Hoeffding tree, enhanced with majority class prediction(HTMC) and hoeffding tree with adaptive naïve Bayes leaf predictions(HTNBA) and next the Hoeffding Option Tree algorithm(HOT) for which maximum of five option paths are given, then Adaptive Hoeffding Option Tree(ADAHOT) is considered which is quite similar to HOT.

TABLE I

COMPARISON OF ACCURACY FOR HOEFFDING TREES WITH MAJORITY CLASS AND NAÏVE BAYES PREDICTION AT LEAVES

Method	**Htmc** memory limit			**htnba** memory limit		
Dataset	100KB	32MB	400MB	100KB	32MB	400MB
Rts	97.2	99.8	99.8	97.2	99.9	99.9
Rtsn	74.9	77.3	77.3	72.9	77.8	78.1
Rtc	77.8	68.1	68.1	63.5	69.2	69.2
Rtcn	56	58.9	58.9	56.0	59.3	59.3
Rrbfs	88	90	90	88	91.7	91.7
Rrbfc	92.1	95.3	95.3	92.1	97.3	97.3
wave21	81.9	83.1	83.1	81.5	86.7	86.7
wave40	82.1	83.1	83.1	82.1	86	86
Led	74.9	75.8	75.8	75.3	75.2	75.2
genF1	95.5	95.5	95.5	95.5	95.5	95.5
genF2	81.1	85.4	85.4	81.1	86.5	86.5
genF3	97.8	97.8	97.8	97.8	97.8	97.8
genF4	94	94.4	94.4	94.1	94.3	94.3
genF5	86.8	91.9	91.9	86.8	93.1	93.1
genF6	88.9	90.8	90.8	88.9	91.2	91.2
genF7	96	95.9	95.9	96.2	97.2	97.2
genF8	99.4	99.4	99.4	99.5	99.4	99.4
genF9	95.2	96.5	96.5	95.2	96.4	96.4
genF10	99.7	99.7	99.7	99.7	99.7	99.7

Result of accuracy of Hoeffding tree which uses majority class and naïve bayes at the leaves to predict the attribute are presented.

Hoeffding tree with majority class prediction gives more accurate values when compared with hoeffding tress with adaptive naïve bayes prediction at only restricted 100KB environments. In the remaining two environments HTNBA is comparatively more accurate than HTMC.

The 32MB memory limit gives approximately good accuracy results in both evaluations. The datasets with noise generated give less accuracy. When a random tree is complex it gives very low accuracy results mainly in 100KB environment.

TABLE II

COMPARISON OF ACCURACY FOR HOEFFDING TREES AND HOEFFDING OPTION TREES WITH FIVE OPTION PATHS

method	**htnba** memory limit			**hot5** memory limit		
dataset	100KB	32MB	400MB	100KB	32MB	400MB
Rts	97.2	99.9	99.9	98.5	100	100
Rtsn	72.9	77.8	78.1	71.3	77.7	78.1
Rtc	63.5	69.2	69.2	55.7	70.2	68.1
Rtcn	56.0	59.3	59.3	55	62	61
Rrbfs	88	91.7	91.7	88.6	92.1	92.1
Rrbfc	92.1	97.3	97.3	94.5	97.9	97.9
wave21	81.5	86.7	86.7	83.8	86.7	86.7
wave40	82.1	86	86	80.6	86.4	86.4
Led	75.3	75.2	75.2	75	75.2	75.2
genF1	95.5	95.5	95.5	95.5	95.5	95.5
genF2	81.1	86.5	86.5	86.5	86.5	86.5
genF3	97.8	97.8	97.8	97.8	97.8	97.8
genF4	94.1	94.3	94.3	94.2	94.4	94.4
genF5	86.8	93.1	93.1	86.4	92.4	92.4
genF6	88.9	91.2	91.2	90.1	91.2	93
genF7	96.2	97.2	97.2	96.4	97.1	97.1
genF8	99.5	99.4	99.4	99.1	99.5	99.5
genF9	95.2	96.4	96.4	95.7	96.5	96.5
genF10	99.7	99.7	99.7	99.7	99.7	99.7

The results given above of Hoeffding tree with two ways of predictions at leaves are compared with hoeffding option tree with option paths restricted to five.

But Hoeffding option tree has more accurate values than hoeffding trees in both cases with different prediction at leaves. HOT5 is more accurate in all environments than hoeffding trees.

The LED generator takes more time for evaluation when compared to other generators. The results in 32MB and 400MB are quite similar but at an average, accuracy of hoeffding option trees is greater than the hoeffding trees.

TABLE III
COMPARISON OF ACCURACY FOR HOEFFDING OPTION TREES AND ADAPTIVE HOEFFDING OPTION TREES

Method	**ADAHOT** memory limit			**HOT5** memory limit		
Dataset	100KB	32MB	400MB	100KB	32MB	400MB
Rts	98.5	100	100	98.5	100	100
Rtsn	74.9	77.6	77.9	71.3	77.7	78.1
Rtc	71.7	74.2	70.8	55.7	70.2	68.1
Rtcn	58.8	62.2	62	55	62	61
Rrbfs	88.6	92	92	88.6	92.1	92.1
Rrbfc	94.5	97.9	97.9	94.5	97.9	97.9
wave21	83.8	87	87	83.8	86.7	86.7
wave40	80.6	87.3	87.3	80.6	86.4	86.4
Led	75	75.2	75.2	75	75.2	75.2
genF1	95.2	95.5	95.5	95.5	95.5	95.5
genF2	80.3	86.5	86.5	86.5	86.5	86.5
genF3	97.8	97.8	97.8	97.8	97.8	97.8
genF4	94.2	94.4	94.4	94.2	94.4	94.4
genF5	86.4	93.2	93.2	86.4	92.4	92.4
genF6	90	91.1	91	90.1	91.2	93
genF7	96.4	96.9	96.9	96.4	97.1	97.1
genF8	99.1	99.5	99.5	99.1	99.5	99.5
genF9	95.7	96.3	96.3	95.7	96.5	96.5
genF10	99.7	99.7	99.7	99.7	99.7	99.7

From Table 3 it is observed that accuracy values in both algorithms is more for 32 MB memory limit for various data sets. All the function generators give good accuracy values when compared to led generator and waveform generator.

Data streams which are produced using noise parameter give less accurate values with more evaluation time for more leaves and nodes. When random tree generator is induced with noise for complex data gives less accuracy values when compared to various data generators.

MOA framework is used for evaluating the algorithms, in which parameter settings are made.

All **graphs** presented are using 32 MB memory limit which represents evaluation time and accuracy of various data sets for different number of examples.

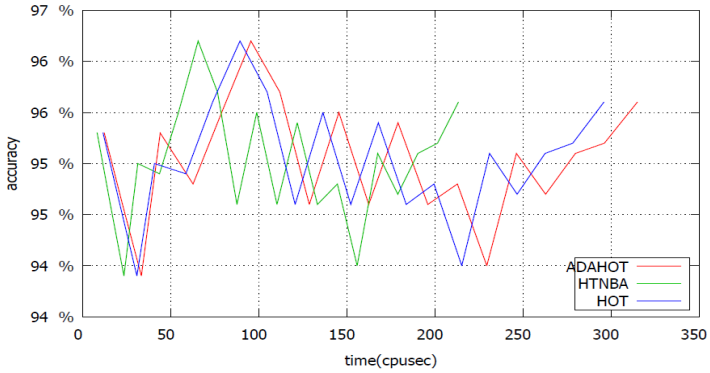

Figure 1: GEN F1 –sampled every 20 million instances

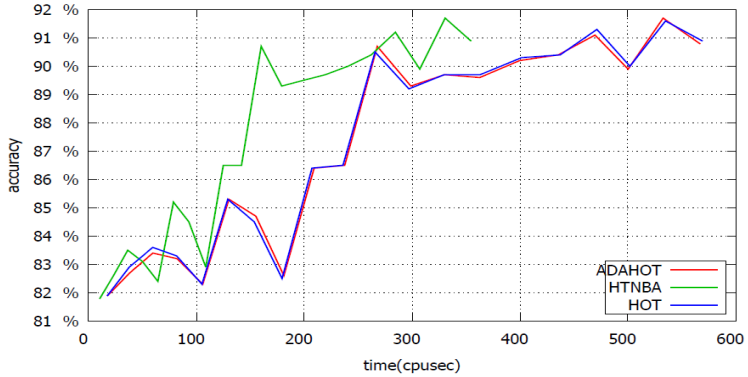

Figure 2: GEN F2-sampled every 20 million instances

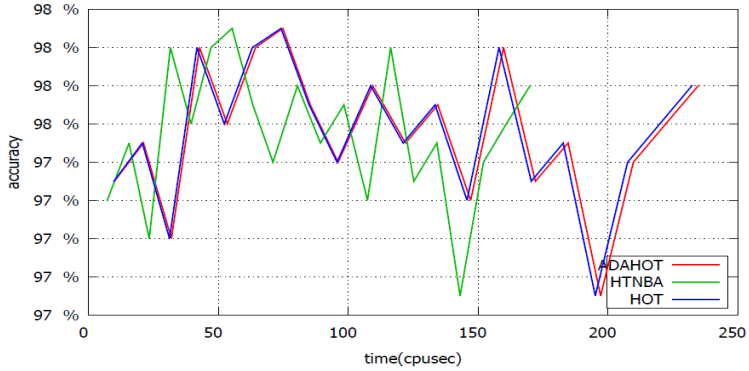

Figure 3: GEN F3-sampled every 20 million instances

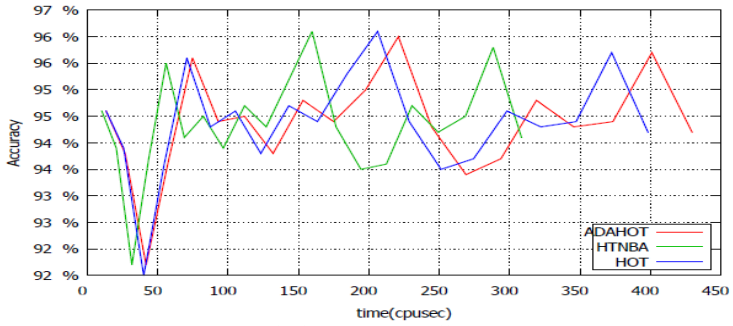

Figure 4: GEN F4-sampled every 20 million instances

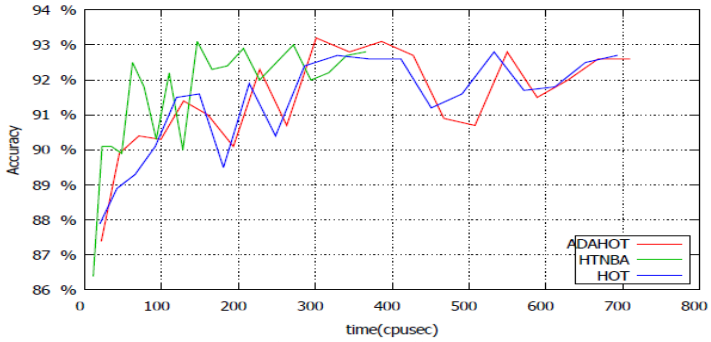

Figure 5: GEN F5-sampled every 20 million instances

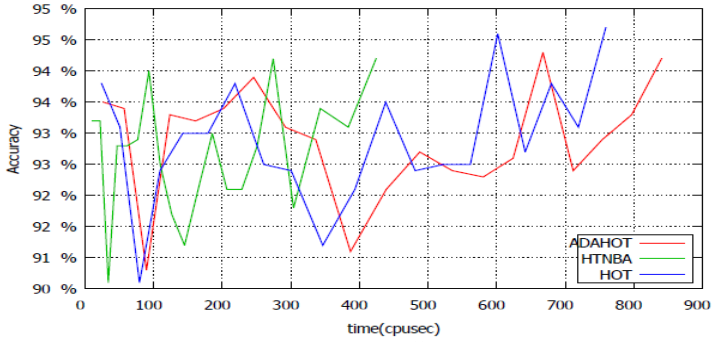

Figure 6: GEN F6-sampled every 20 million instances

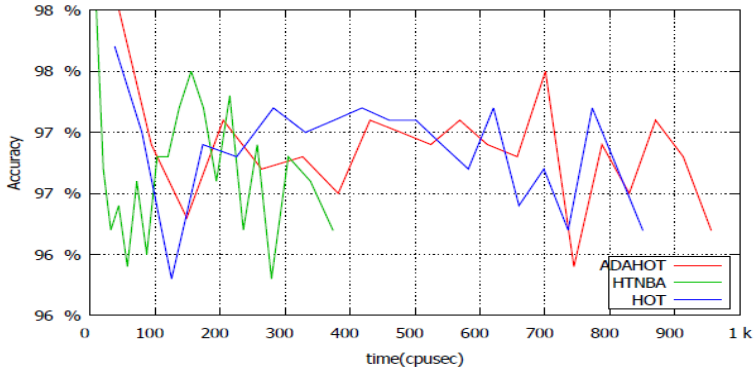

Figure 7: GEN F7-sampled every 20 million instances

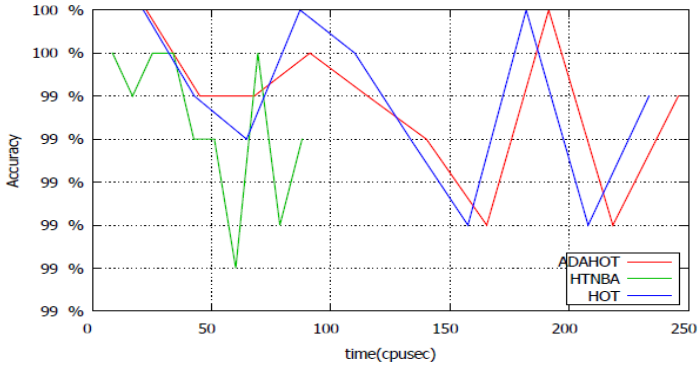

Figure 8: GEN F8-sampled every 10 million instances

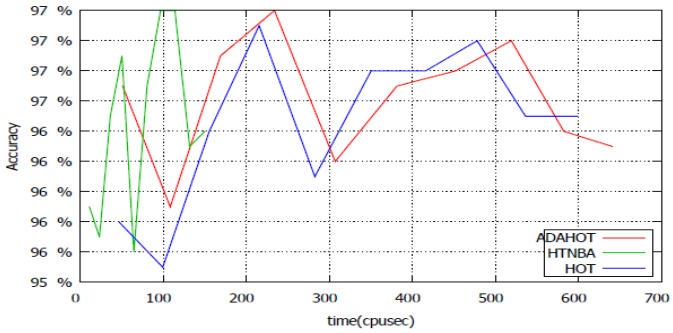

Figure 9: GEN F9-sampled every 10 million instances

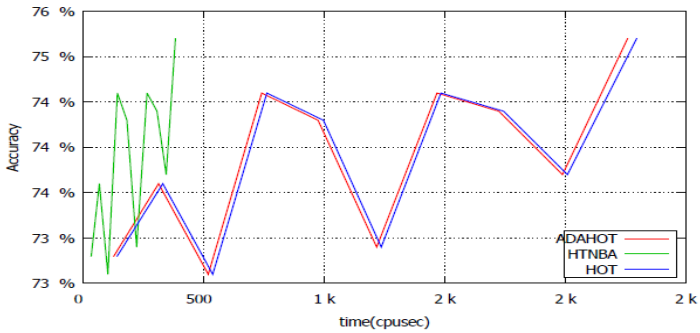

Figure 10: Led generator- sampled every 10 million instances

From Hoeffding option tree and Hoeffding tree, it is known that accuracy is quite similar for both algorithms but evaluation time varies more. Hoeffding tree has less evaluation time than Hoeffding option tree.

Based on the results all the values such as accuracy, tree size, number of leaves of adaptive hoeffding option tree and Hoeffding option tree are quite similar, but evaluation time of adaptive Hoeffding option tree is more than Hoeffding option tree.

The graphs are drawn using table 1, table 2, table 3 and the accuracy values of adaptive hoeffding tree, Hoeffding tree and Hoeffding Option Trees. All graphs indicate time in X-axis which is computed in cpuseconds and accuracy values in Y-axis which is represented in percentages.

6. Conclusion

Data streams are defined and the evaluation of the Massive Online Analysis framework as a software environment for research on learning from evolving data streams and evaluating algorithms is done. Prequential evaluation method is used to experimentally compare single classifier and ensemble data stream classifier. Use of option trees for classification in data streams is made and results are noted. Memory limits are set to define various environments where evaluation can be done.

Data generators which generate data sets based on some pattern, which are further used for evaluation and experiments are carried out on the given algorithms Hoeffding trees, Hoeffding option trees and adaptive Hoeffding option trees.

Majority class prediction and naïve Bayes prediction at the leaves is used for Hoeffding trees and they are evaluated using the framework and the results are noted and accuracy of the algorithms mainly Hoeffding trees and Hoeffding option trees is compared for different data sets under various memory limits. It is observed that option nodes enable faster growth without instability in splitting decisions and have improved lookahead strategy. Accuracy and speed of the algorithms based on Hoeffding option tree and adaptive Hoeffding option tree are observed and graphs are represented for different data sets on various memory limits.

As a future work, we plan to work on the evaluation method with other large number of algorithms to take a diversified look at the performance of the most recent stream mining techniques.

References

[1] P. Domingos and G. Hulten, “Mining High Speed Data Streams”, in Proceedings of the Association for Computing Machinery Sixth International Conference on Knowledge Discovery and Data Mining, 2000.

[2] P. Domingos and G. Hulten, “A General Framework for Mining Massive Data Streams”.

[3] Manish Mehta, Rakesh Agrawal, and Jorma Rissanen , ”SLIQ: A fast scalable classifier for data mining”, in Extending Database Technology, 1996.

[4] John Shafer, Rakesh Agrawal, and Manish Mehta., “SPRINT: A scalable parallel classifier for data mining”, in International Conference on Very Large Databases, 1996.

[5] Ron Kohavi and Clayton Kunz, “Option decision trees with majority votes”, in International Conference on Machine Learning, 1997.

[6] Bernhard Pfahringer, Geoffrey Holmes, and Richard Kirkby, “New options for hoeffding trees”, 2007.

[7] Richard kirkby, “Improving Hoeffding Trees”, University of Waikato, 2007.

Dr. P. Chenna Reddy did his B.Tech from S.V. University College of Engineering, Tirupati, M.Tech & Ph.D from JNTU, Hyderabad. He has 15 years of Teaching experience. His areas of interest are Computer Networks and related fields. He is currently working on Bio inspired networking. He is currently working as Associate Professor at JNTUA College of Engineering, Pulivendula. He has published several papers in reputed journals and conferences.

B. Reshma Yusuf received her Bachelor degree with Computer Science and Engineering in 2010. She joined Master of Technology degree with specialization in Computer Science and Engineering and a research scholar in JNTU, pulivendula. Her research interests include Data Mining and related fields.