Performance analysis of Hoeffding trees in data streams by using massive online analysis framework

P.K. Srimani

R & D Division,
Bangalore University
Jnana Bharathi, Mysore Road,
Bangalore-560056, Karnataka, India
Email: profsrimanipk@gmail.com

Malini M. Patil*

Department of Information Science and Engineering,
J.S.S. Academy of Technical Education,
Uttaralli-Kengeri Main Road,
Mylasandra, Bangalore-560060, Karnataka, India
Email: patilmalini31@gmail.com
*Corresponding author

Abstract: Present work is mainly concerned with the understanding of the problem of classification from the data stream perspective on evolving streams using massive online analysis framework with regard to different Hoeffding trees. Advancement of the technology both in the area of hardware and software has led to the rapid storage of data in huge volumes. Such data is referred to as a data stream. Traditional data mining methods are not capable of handling data streams because of the ubiquitous nature of data streams. The challenging task is how to store, analyse and visualise such large volumes of data. Massive data mining is a solution for these challenges. In the present analysis five different Hoeffding trees are used on the available eight dataset generators of massive online analysis framework and the results predict that stagger generator happens to be the best performer for different classifiers.

Keywords: data mining; data streams; static streams; evolving streams; Hoeffding trees; classification; supervised learning; massive online analysis; MOA; framework; massive data mining; MDM; dataset generators.

Reference to this paper should be made as follows: Srimani, P.K. and Patil, M.M. (2015) 'Performance analysis of Hoeffding trees in data streams by using massive online analysis framework', *Int. J. Data Mining, Modelling and Management*, Vol. 7, No. 4, pp.293–313.

Biographical notes: P.K. Srimani is currently the Dean of R&D Division (Computer Science and Computational Fluid Dynamics), Bangalore University, Bangalore, Karnataka, India and has successfully produced many PHDs. She is actively engaged in research and doctoral programs. Her areas of interests are data mining, image processing, bio-informatics and CFD. She has several research publications in many prestigious international journals like proceedings of Royal Society of London, *IJKE* etc. She is a Fellow of National Academy of Sciences. She has authored many text books.

Malini M. Patil is presently working as Associate Professor in the Department of Information Science and Engineering at J.S.S. Academy of Technical Education, Bangalore, Karnataka, India. She is currently pursuing her research program from Bharthiyaar University, Coimbatore, Tamilnadu. Her research interests are data mining, data ware housing and data stream mining. She has published her research papers in many international journals. She has also participated in many workshops, faculty development programs, symposiums, national and international conferences in India and abroad. She has presented her research papers in many national and international conferences.

1 Introduction

Today's era of technology has resulted in the increased rate of data generation. This is mainly because of different mobile applications, sensor applications, measurements in network traffic monitoring and management, log records and web click streams in search engines, web logs, e-mails, blogs, twitter posts, etc. This kind of data generated can be considered as a streaming data since it is obtained for an interval of time. Aggarwal (2007) defines data stream as an ordered sequence of items that arrive in timely order. Data streams are different from data in traditional databases. They are continuous, unbounded, usually come in high speeds and have a data distribution which often changes with time (Gaber et al., 2007). Mining a stream data is referred as data stream mining or massive data mining (MDM). Important features of data streams can be summarised as:

1 data streams are huge in size

2 data streams are continuous in nature

3 data streams are fast changing and require fast response

4 random access of data is not possible

5 storage of data streams is limited, only the summary of the data can be stored.

Data stream environment has different requirements from the traditional batch learning setting. Ikonomovska et al. (2007) explained about the state of art in data stream mining. The main requirements in mining data streams are summarised as follows:

1 the example has to be processed at a time, and inspected only once

2 limited amount of memory can be used

3 work in a limited amount of time

4 any time prediction can be made.

Few important challenges of data streams are summarised as follows:

1 since the data collected is huge, multiple scans are not possible in data stream mining as compared with traditional data mining algorithms

2 the mining method of data streams should handle the change in data distribution

3 in case of online data streams mining methods should be more faster than the speed of incoming data

4 memory management issues related to data storage and CPU speed also matter more in data stream mining.

Data streams can be classified into two types. Guha et al. (2001) explained about different types of data streams viz., static (offline) streams and evolving (online) streams. Static streams are characterised by regular bulk arrivals. Web logs are considered as static data streams because most of the reports are generated in a certain period of time. Another best example of static data streams is queries on data warehouses. Evolving data streams are characterised by real time updated data that come one by one in time. Examples for evolving data streams are frequency estimation of internet packet streams, stock market data, and sensor data. Such data should be processed online. Another very important feature of online data streams is that they should be processed online with the rapid speed with which they arrive and should be discarded immediately after being processed. Bulk data processing is not possible in evolving data streams where as it is possible in static data streams.

Another type of learning algorithm is metric learning which provide useful distance functions for a variety of domains and recent work has shown good accuracy for problems where the learner can access all distance constraints at once. However, in many real applications, constraints are only available incrementally thus providing the methods that can perform online updates to the learned metric. Jain et al., (2008) present a new online metric learning algorithm that updates a learned Mahalanobis metric based on LogDet regularisation and gradient descent. They have developed online locality-sensitive hashing scheme which leads to efficient updates to data structures used for fast approximate similarity search.

Another important thing emphasised here is that among the online learning methods, online metric learning is not suitable for classification of data steams while it could be addressed to the problems associated with unsupervised settings such as clustering. In such cases generally algorithms like K-means, Mahalanobis, gradient descent, etc., are used. Therefore for the present investigation, online metric learning is not suitable.

The paper is organised as follows: Section 2 focuses mainly on the preliminaries of classification as a technique of data mining. Section 3 discusses about related work in the area of DM and MDM. Section 4 discusses about the methodology used for the present analysis, data streams used in the analysis, different algorithms used in for handling static streams and evolving streams. Section 5 presents the analysis and experimental results respectively. Future enhancement of the work and conclusions are briefed at the end of the paper.

2 Related work

Srimani and Patil (2011, 2012a, 2012b, 2012c) presented the techniques of data mining exhaustively. A technique called educational mining (Edu-mining) is proposed using classification technique to discover the knowledge from educational data (Edu-data).

Edu-data is large data repository consisting of data related to technical educational systems. Edu-data is evolved because of huge collection of data mainly from WWW, study material available in the internet, e-learning schemes, computerisation of education system, online registration schemes for admission process in the universities, student information system, examination evaluation systems, etc. Knowledge discovered helps the technical education system (TES) to take useful decisions for maintaining the quality of the education system. In an education system student, faculty and management are the three stake holders of the TES. Edu-mining is carried out on all the three stake holders of TES. The results of the exhaustive research work (Srimani et al., 2011, 2012a, 2012b, 2012c) are highly effective in optimal decisions at the managerial level. It has earned lot of scope in educational research. Literature survey reveals that (Romero and Ventura, 2010; Baker and Yacef, 2009; Knauf et al., 2008) have done good amount of work in data mining using different datasets related to education system and using different data mining techniques. It is clear that mining Edu-data is a novel approach. The framework used in Edu-mining is WEKA. Witten et al. (2011) contributed a lot on this framework which is mainly used for data mining. Weka is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from Java code. It can be used for data-processing, classification, regression, clustering, association rules and visualisation. It is also well suited for developing new machine learning schemes.

As discussed earlier massive collection of data from mobile applications, sensor applications, network monitoring, traffic management, etc., results in a new method of mining and such a type of data evolved is referred to as MDM. Traditional data mining is not applied as the nature of data streams is continuous, unbounded, very fast and with varying distributions. The complete survey on data stream mining is presented by Ikonomovska et al. (2007). Massive online analysis (MOA) is a framework proposed by Bifet and Kirkby (2009) and Bifet et al. (2009a, 2011), which is used to perform MDM. Srimani and Patil (2012d) have explored MOA framework with regard to MDM on data streams using classification algorithms. The authors have used four different classification evaluators available in MOA framework with a Bayesian approach for the analysis. Present work elaborates on the MOA framework for MDM using Hoeffding trees.

A number of recent techniques address the problem of metric learning by Davis et al. (2007), Frome et al. (2006), Globerson and Roweis (2005), Schultz and Joachims (2003), Weinberger et al. (2009) and Xing et al. (2002) in which the authors use a distance function between data objects and is learned based on given similarity constraints between examples. Such algorithms have been applied to a variety of real world learning tasks. In many applications the desired distance function may need to change gradually over time as additional information or constraints are received. In image search applications on the internet, online click through data that is continually collected may impact the desired distance function. Shwartz et al. (2004) discuss about metric learning algorithms which attempts to handle constraints that are received one at a time.

Weinberger et al. (2009) presents how to learn Mahanalobies metric for k-nearest neighbour (kNN) classification by semi definite programming. The method proposed by Globerson and Roweis (2005) presents an algorithm for learning a quadratic Gaussian

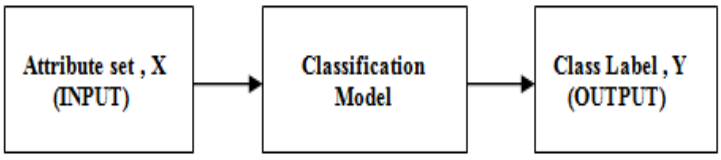
metric for use in classification tasks. Locality sensitive hashing scheme is proposed by (Charikar, 2002) which is basically derived from the hash function operating on a collection of objects. Such a scheme leads to compact representation of objects so that similarity of objects can be estimated from their compact sketches. Davis et al. (2007) present an information theoretic approach to learning mahalanobies distance function.

3 Preliminaries

Classification of large datasets is one of the important data mining technique used to solve statistics and machine learning problems in order to extract rules and patterns from data that are used for prediction. Han and Kamber (2007) explained the different techniques of classification, prediction and regression along with the state of art of algorithms. *Classification problem is a supervised learning problem*. Illustrative examples include classification problems which have binary outputs: Businesses are often interested in a simple *yes* or *no* response to a proposal. Doctors want to know whether a patient is *healthy* or *sick*. A bank loan officer needs to know which loan applicants are *reliable* and which are *non-reliable*. Different types of classification techniques include decision tree induction, rule-based classifier, nearest neighbour classifiers, statistical methods, and neural network approach. The objective in classification is to build a mapping function that assigns class labels to each new instance or to verify the appropriateness of class labels already assigned. Mathematically, classification is defined as follows:

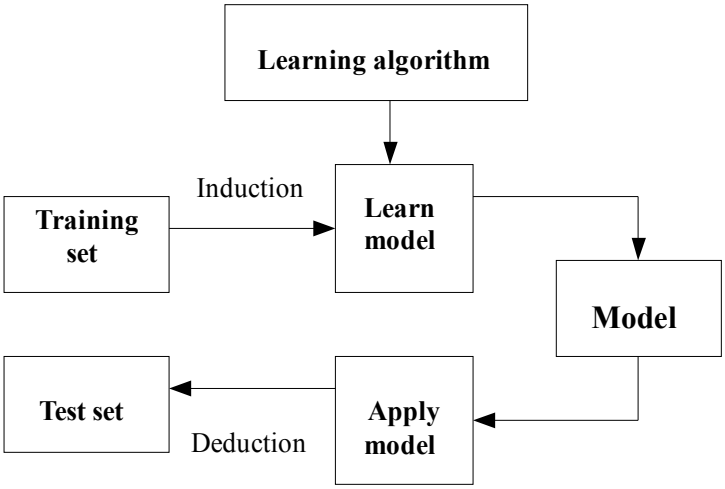
Given a database $X = \{x_1, x_2, x_3, .., x_n\}$ of tuples (items and records) and a set of classes $Y = \{y_1, y_2, y_3, …., y_m\}$. Classification is the task of learning a target function f: $X \rightarrow Y$ that maps each attribute set x to one of the predefined class labels y. Informally the target function is also known as a *classification model* and is illustrated in Figure 1. Thus, classification is the task of mapping an input attribute x into its class label y.

Figure 1 Classification problem as the task of mapping



The general approach for solving classification problems is shown in Figure 2. First, a training set consisting of records whose class labels are known must be provided. The training set is used to build a classification model, which is subsequently applied to the test set which consists of records with unknown class labels.

Figure 2 General approach for solving classification problems



4 Methodology

MDM is performed using MOA framework. The framework is designed by Bifet and Kirkby (2009) and Bifet et al. (2009a, 2011). It is a software environment for implementing algorithms and running experiments for online learning from evolving data streams. MOA is designed in such a way that it can handle the challenging problem of scaling up the implementation of state of the art algorithms to real world datasets. It consists of offline and online algorithms for classification and clustering. It also consists of tools for evaluation. Thus, MOA is an open source framework to handle massive, potentially infinite, evolving data streams. MOA mainly permits the evaluation of data stream learning algorithms on large streams under explicit memory limits. The method MDM mainly consists of the four steps. viz.,

Step 1 Select the task.

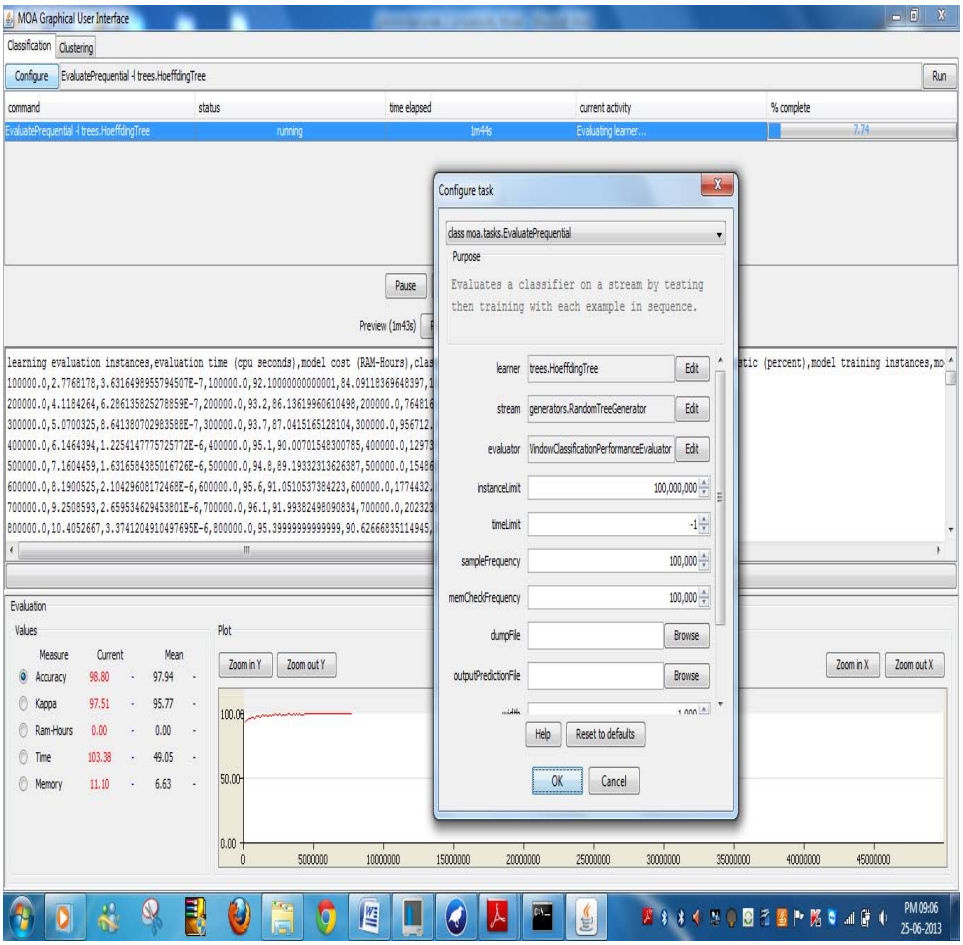
Step 2 Select the learner.

Step 3 Select the stream generator.

Step 4 Select the evaluator.

The model is configured with the above said steps. And results are noted. Sample GUI is provided in Figure 3. Presently, GUI comes with classification and clustering modules. The configuration of the modules is carried out using the above said steps.

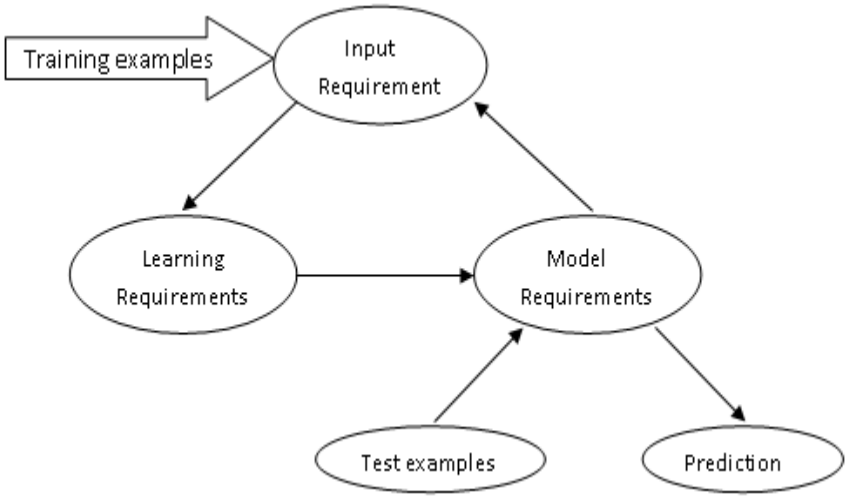
Figure 3 Configuration of MOA (see online version for colours)



4.1 Evaluation process in MOA

There are two options in the case of evaluation process in MOA. Viz. *Holdout and Prequential*. The first case is suitable when the division between train and test sets is predefined so that the results from different studies could be directly compared. In the second case each individual example can be used to test the model before it is used for training and accordingly the accuracy can be incrementally updated. A typical classification model using MOA is shown in Figure 4 and it is self explanatory.

Figure 4 A typical data stream classification model in MOA



4.2 Data streams used in the analysis

From the literature survey it is found that there is a scarcity of data sources. The present work is carried out on the data stream generators available in MOA framework. The following are the eight main data stream generators used in the present investigation. They are explained as follows:

4.2.1 RANDOMTREE-Generator (ST1)

Generates a stream based on a randomly generated tree contributed by Domingos and Hulten (2000). It produces concepts that in theory should favour decision tree learners. It constructs a decision tree by choosing attributes at random to split, and assigning a random class label to each leaf. Once the tree is built, new examples are generated by assigning uniformly distributed random values to attributes which then determine the class label via the tree. The generator has parameters to control the number of classes, attributes, nominal attribute labels, and the depth of the tree. A degree of noise can be introduced to the examples after generation. In the case of discrete attributes and the class label, a probability of noise parameter determines the chance that any particular value is switched to something other than the original value. For numeric attributes, a degree of random noise is added to all values, drawn from a random Gaussian distribution with standard deviation equal to the standard deviation of the original values multiplied by noise probability.

4.2.2 RANDOMRBF-Generator (ST2)

Generates a random radial basis function (RBF), introduced by Bifet et al. (2010). This generator was devised to offer an alternate complex concept type that is not straightforward to approximate with a decision tree model. The RBF generator works as follows: A fixed number of random centroids are generated. Each centre has a random

position, a single standard deviation, class label and weight. New examples are generated by selecting a centre at random, taking weights into consideration so that centres with higher weight are more likely to be chosen. A random direction is chosen to offset the attribute values from the central point. The length of the displacement is randomly drawn from a Gaussian distribution with standard deviation determined by the chosen centroid. The chosen centroid also determines the class label of the example. This effectively creates a normally distributed hyper sphere of examples surrounding each central point with varying densities. Only numeric attributes are generated.

4.2.3 SEA-Generator (ST3)

A streaming ensemble algorithm (SEA) is used for large-scale classification which mainly generates SEA concepts functions. This dataset contains abrupt concept drift, first introduced by Street and Kim (2001). It is generated using three attributes, where only the two first attributes are relevant. All three attributes have values between 0 and 10. The points of the dataset are divided into four blocks with different concepts. In each block, the classification is done using $f_1 + f_2 \leq \theta$, where f_1 and f_2 represent the first two attributes and θ is a threshold value. The most frequent values are 9, 8, 7 and 9.5 for the data blocks.

4.2.4 STAGGER-Generator (ST4)

Generates STAGGER concept functions. The function uses the incremental learning method from noisy data. They were introduced by Schlimmer and Granger (1986). The STAGGER concepts are Boolean functions of three attributes encoding objects: size (small, medium, and large), shape (circle, triangle, and rectangle), and colour (red, blue, and green). A concept description covering either green rectangles or red triangles is represented by (shape = rectangle and colour = green) or (shape = triangle and colour = red).

4.2.5 WAVEFORM-Generator (ST5)

Generates a problem of predicting one of three waveform types. It shares its origins with LED, and was also donated by Aha (2007) to the UCI repository. 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. The optimal Bayes classification rate is known to be 86%. There are two versions of the problem, wave21 which has 21 numeric attributes, all of which include noise, and wave40 which introduces an additional 19 irrelevant attributes.

4.2.6 AGARWAL-Generator (ST6)

Generates one of ten different pre-defined loan functions. It was a common source of data for early work on scaling up decision tree learners contributed by Agarwal et al. (1992). The generator produces a stream containing nine attributes, six numeric and three categorical. These attributes describe hypothetical loan applications. There are ten functions defined for generating binary class labels from the attributes. Presumably, these determine whether the loan should be approved.

4.2.7 HYPERPLANE-Generator (ST7)

Generates a problem of predicting class of a rotating hyperplane. It was introduced by Hulten et al. (2001). A hyperplane in *d*-dimensional space is the set of points *x* that satisfy

$$\sum_{i=1}^{d} W_i X_i = W_o = \sum_{i=1}^{d} W_i \tag{1}$$

where x_i , is the i^{th} coordinate of *x*. Hyperplanes are useful for simulating time-changing concepts, because we can change the orientation and position of the hyperplane in a smooth manner by changing the relative size of the weights. We introduce change to this dataset adding drift to each weight attribute $w_i = w_i + d\sigma$, where σ is the probability that the direction of change is reversed and *d* is the change applied to every example.

4.2.8 LED-Generator (ST8)

Generates a problem of predicting the digit displayed on a seven-segment LED display. This data source originates from the CART book. An implementation in C was donated to the UCI machine learning repository by David Aha. The main idea is contributed by Breiman et al. (1984). The goal is to predict 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 (LED) produces 24 binary attributes, 17 of which are irrelevant.

5 Algorithms used in the analysis

This section presents the five algorithms used in the performance analysis of Hoeffding trees viz., Hoeffding trees (HT), Hoeffding option tree (HOT), adaptive-size Hoeffding tree (ASHT), adaptive Hoeffding option tree (AHOT), Hoeffding adaptive tree (HAT).

5.1 Hoeffding trees (HT)

Hoeffding trees are first proposed by Hulten et al. (2001). One of the basic algorithm for stream data classification is Hoeffding tree algorithm. It is an incremental, anytime decision tree induction algorithm that is capable of learning from massive data streams assuming that the distribution generating examples does not change over time. It produces decision trees which are similar to traditional batch learning method. Hoeffding trees and decision trees are asymptotically related. HT algorithm is based on a simple idea that a small sample can be often sufficient to choose an optimal splitting attribute. The key point to be noted here is traditional batch learning methods and also generate decision trees based on splitting attributes. Mathematically, it is proved that HT algorithm uses Hoeffding bound. To understand the meaning of Hoeffding bound few assumptions are made. Suppose we make 'N' independent observations of a random variable 'r' with range 'R', where 'r' is an attribute selection measure. In case of Hoeffding trees 'r' is information gain and if we compute the mean value of r 'r_{mean}' of this sample the *Hoeffding bound* states that the true mean of 'r' is at least $1-\delta$ where δ is user specified and

$$\varepsilon = \sqrt{\frac{R^2 \ln\left(1/\delta\right)}{2N}} \tag{2}$$

Hoeffding bound is independent of probability distribution. Main advantages of HT algorithm are

1 it is incremental in nature

2 achieves high accuracy using small sample

3 multiple scans on same data are never performed.

But the main disadvantage is HT cannot handle concept drift because once the node is created it can never be changed. Wang et al. (2003) explained about handling concept drift using classifiers. The algorithm spends a great deal of time with attributes that have nearly identical splitting quality. In addition, the memory utilisation can be further optimised. The Hoeffding tree algorithm is presented in Figure 5.

Figure 5 Algorithm for Hoeffding tree

1: Let HT be a tree with a single leaf (the root)
2: **for all** training examples **do**
3: Sort example into leaf l using HT
4: Update sufficient statistics in l
5: Increment n_l, the number of examples seen at l
6: **if** $n_l \; mod \; n_{min} = 0$ **and** examples seen at l not all of same class **then**
7: Compute $\overline{G}_l(X_i)$ for each attribute
8: Let X_a be attribute with highest $\overline{G}_l$
9: Let X_b be attribute with second-highest $\overline{G}_l$
10: Compute Hoeffding bound $\epsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n_l}}$
11: **if** $X_a \neq X_\emptyset$ **and** $(\overline{G}_l(X_a) - \overline{G}_l(X_b) > \epsilon$ **or** $\epsilon < \tau)$ **then**
12: Replace l with an internal node that splits on X_a
13: **for all** branches of the split **do**
14: Add a new leaf with initialized sufficient statistics
15: **end for**
16: **end if**
17: **end if**
18: **end for**

Kirkby (2007) lists pseudo-code for inducing a Hoeffding tree from a data stream. Line 1 initialises the tree data structure, which starts out as a single root node. Lines 2–18 form a loop that is performed for every training example. Every example is filtered down the tree to an appropriate leaf, depending on the tests present in the decision tree built to that point (line 3). This leaf is then updated (line 4) – each leaf in the tree holds the sufficient statistics needed to make decisions about further growth. The sufficient statistics that are updated are those that make it possible to estimate the information gain of splitting on

each attribute. Line 5 simply points out that n_l is the example count at the leaf, and it too is updated. Technically n_l can be computed from the sufficient statistics. For efficiency reasons the code block from lines 6–17 is only performed periodically. The delayed evaluation controlled by n_{min}. Lines 7–11 perform the test described in the previous section, using the Hoeffding bound to decide when a particular attribute has won against all of the others. G is the splitting criterion function (information gain) and G is its estimated value. In line 11 the test for Xi, the null attribute, is used for pre-pruning. If an attribute has been selected as the best choice, lines 12–15 split the node, causing the tree to grow.

5.2 Hoeffding option trees (HOT)

Hoeffding option trees are regular Hoeffding trees containing additional nodes that allow several tests to be applied, leading to multiple Hoeffding trees as separate paths. HOTs are proposed by Pfahringer et al. (2007). They consist of a single structure that efficiently represents multiple trees. A particular example can travel down multiple paths of the tree, contributing in different ways to different options.

5.3 Adaptive-size Hoeffding tree (ASHT)

The adaptive-size Hoeffding tree (ASHT) is proposed by Bifet and Gavaldà (2009) and Bifet et al. (2009b, 2012) derived from the Hoeffding tree algorithm with the following differences:

- it has a maximum number of split nodes, or *size*
- after one node splits, if the number of split nodes of the ASHT tree is higher than the maximum value, then it deletes some nodes to reduce its size.

The intuition behind this method is as follows: smaller trees adapt more quickly to changes, and larger trees do better during periods with no or little change, simply because they were built on more data. Trees limited to size s will be reset about twice as often as trees with a size limit of $2s$. This creates a set of different reset-speeds for an ensemble of such trees, and therefore a subset of trees that are a good approximation for the current rate of change. It is important to note that resets will happen all the time, even for stationary datasets, but this behaviour should not have a negative impact on the ensemble's predictive performance. When the tree size exceeds the maximum size value, there are two different delete options:

- Delete the oldest node, the root, and all of its children except the one where the split has been made. After that, the root of the child which is not deleted becomes the new root
- Delete all the nodes of the tree, i.e., restart from a new root.

The maximum allowed size for the n^{th} ASHT tree is twice the maximum allowed size for the $(n-1)^{th}$ tree.

5.4 Adaptive Hoeffding option tree (AHOT)

Adaptive decision option tree for streaming data with adaptive naive Bayes classification at leaves. Bifet et al. (2010) summarised the a*daptive Hoeffding option tree* as a Hoeffding option tree with the following improvement: each leaf stores an estimation of the current error. It uses an EWMA estimator with $\alpha = 0.02$. The weight of each node in the voting process is proportional to the square of the inverse of the error.

5.5 Hoeffding adaptive tree (HAT)

Decision tree for streaming data with adaptive naive Bayes classification at leaves. Bifet et al. (2009b) and Bifet and Gavaldà (2009) proposed that this adaptive naive Bayes prediction method monitors the error rate of majority class and naive Bayes decision in every leaf and choose to employ naive Bayes decisions only where they have been more accurate in past cases.

6 Experiments and results

Experiments are carried out on the eight data streams mentioned in Section 4 by using five Hoeffding algorithms viz., Hoeffding trees (HT), Hoeffding option tree (HOT), adaptive-size Hoeffding tree (ASHT), adaptive Hoeffding option tree (AHOT), and Hoeffding adaptive tree (HAT) in the MOA framework. The results are presented in Tables 1 to 4 and Figures 6 to 11. Figures are self-explanatory.

Table 1 Results of time taken by each of the HTs for data streams ST1 to ST8

	HT-time	*ASHT-time*	*HOT-time*	*AHOT-time*	*HAT-time*
ST1	1,424.3	1,552.6	1,459.39	1,584.38	3,122.28
ST2	2,868.71	2,835.65	4,949.89	6,120.8	13,465.62
ST3	1,561.61	1,564.28	1,982.63	1,663.8	1,893.8
ST4	130.58	126.73	252.91	252.72	1,065.29
ST5	3,869.28	3,898.36	6,201.07	7,389.07	14,309.54
ST6	1,510.05	1,487.22	2,083.65	2,091.09	2,084.92
ST7	3,290.39	3,351.85	3,513.68	3,476.82	5,721.17
ST8	3,425.97	3,351	10,208.19	9,832.01	8,652.69

A glance at Table 1 reveals that the performance of the algorithms is very interesting. HT algorithm performs extremely well for ST1 (1,424.3 sec), ST3 (time = 1,561.61 sec), ST5 (time = 3,869.28 sec) and ST7 (time = 3,290.39 sec) since it takes minimal time when compared to other algorithms. ASHT algorithm performs extremely well for ST2 (time = 2,835.65 sec), ST4 (time = 126.73 sec), ST6 (time = 1,487.22 sec) and ST8 (time = 3,351 sec) since it takes minimal time when compared to other algorithms. HOT, AHOT and HAT no doubt perform well but the optimum result is obtained through HT and ASHT algorithms. The glance at Table 1 reveals that stream ST4 takes the minimum time while ST5 takes the maximum time when compared to the other streams.

Figure 6 Graph of data streams (8) vs time (for five Hoeffding tree algorithms) (see online version for colours)

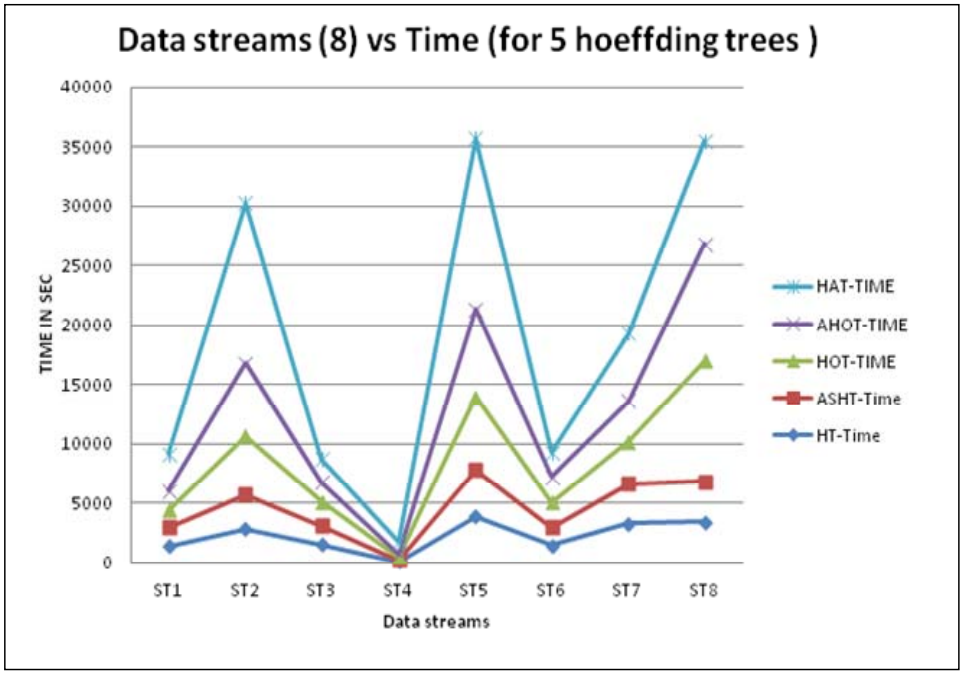


Figure 7 Graph of Hoeffding trees (5) vs time (for eight data streams) (see online version for colours)

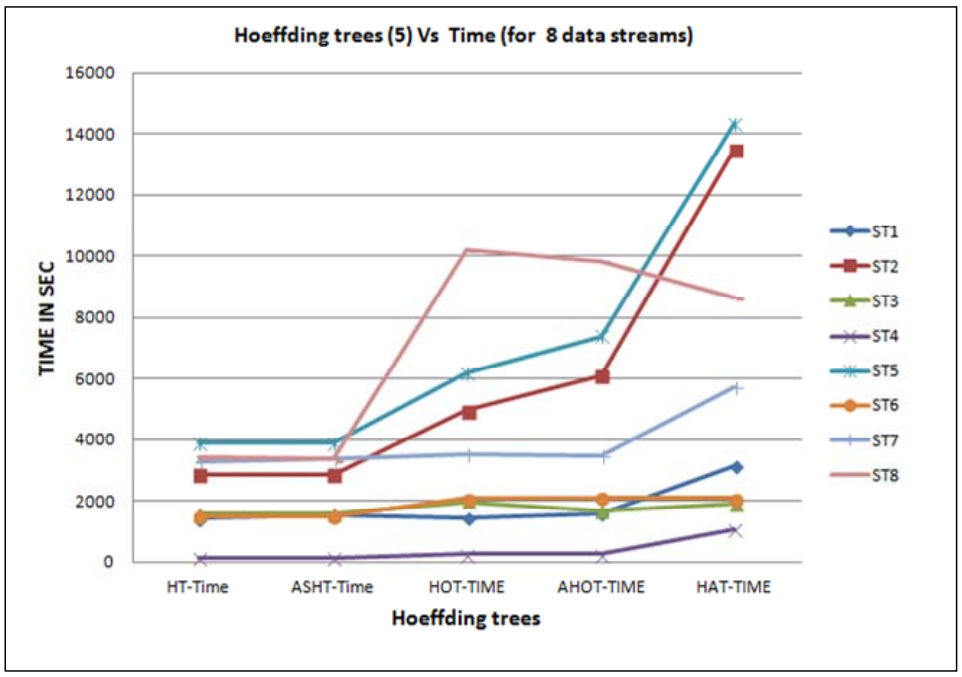


Table 2 Results of memory used by each of the HTs for data streams from ST1 to ST8

	HT-MEM	*ASHT-MEM*	*HOT-MEM*	*AHOT-MEM*	*HAT-MEM*
ST1	25.9	25.9	26.39	26.43	27.32
ST2	28.53	28.31	31.21	31.22	126.31
ST3	23.93	25.9	32.9	26.43	29.43
ST4	0	0	0.01	0.01	0.03
ST5	29.57	29.57	31.31	31.31	122.04
ST6	23.93	23.93	27.54	27.56	1.48
ST7	29.23	29.01	29.32	29.33	9.9
ST8	20.42	20.42	23.47	23.48	17.91

A glance at Table 2 reveals that the performance of the algorithms is very interesting in case of memory usage of the algorithms. HT algorithm performs extremely well for ST1 (25.9), ST4 (0), and ST5 (29.57), since it takes less ram hours when compared to other algorithms. ASHT algorithm performs extremely well for ST1 (25.9), ST2 (28.31), ST4 (0) and ST5 (29.57) since it takes minimal amount of memory when compared to other algorithms. HAT algorithm performs very well in case of memory usage for ST3 (29.43), ST6 (1.48) and ST8 (17.91). HOT and AHOT no doubt perform well but the optimum result is obtained through HT, ASHT and HAT algorithms. It is also observed that stream ST4 takes least memory while ST5 takes the maximum memory when compared to the other streams. But for the case of HAT stream ST2 has the maximum memory usage. Further. it is amazing to note that the memory usage for ST4 is almost nil in all the cases.

Figure 8 Graph of data streams (8) vs ram hours (for five Hoeffding tree algorithms) (see online version for colours)

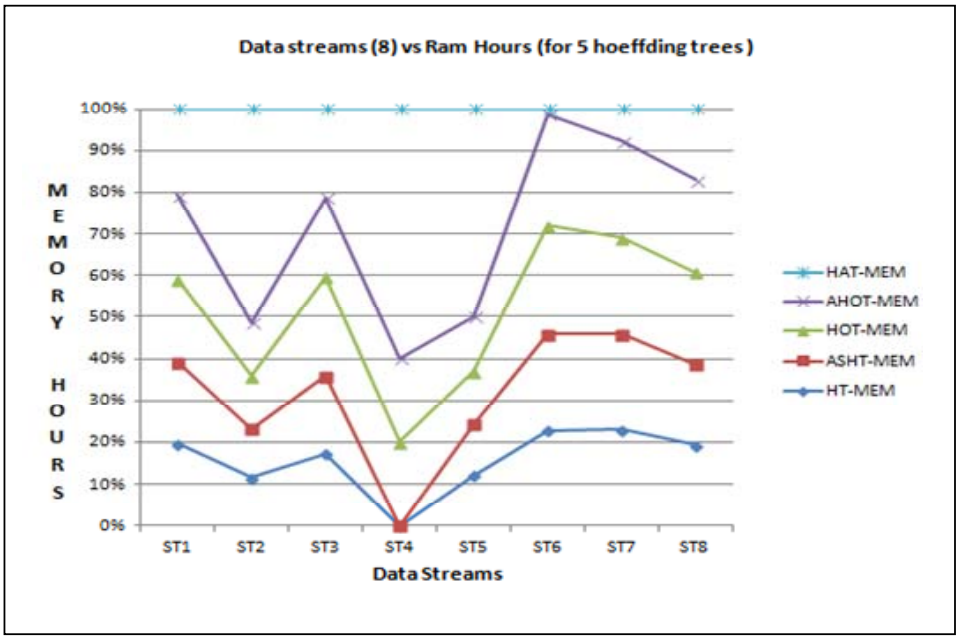
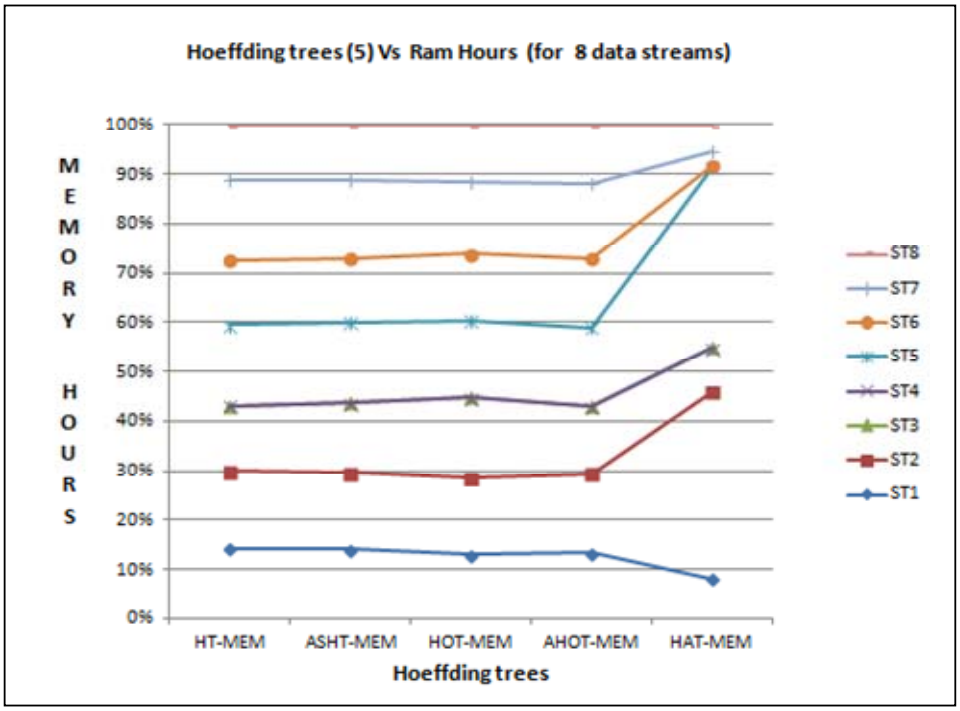


Figure 9 Graph of Hoeffding trees (5) vs ram hours (for eight data streams) (see online version for colours)



From Table 3 it is observed that algorithm HT performs well in the case of ST1 (kappa = 99.03%), ST4 (kappa = 100%) and ST6 (kappa = 88.77%). Algorithm ASHT performs well in the case of ST1 (kappa = 99.03%), ST3 (kappa = 99.03%), ST4 (kappa = 100%), and ST6 (kappa = 88.77%). Algorithm HOT performs well in the case of ST1 (kappa = 99.03%), ST2 (kappa = 90.54%), ST4 (kappa = 100%), and ST6 (kappa = 88.79%). Algorithm AHOT performs well in the case of ST1 (kappa = 99.03%), ST3 (kappa = 99.03%), ST4 (kappa = 100%), ST5 (kappa = 78.24%), ST6 (kappa = 88.78%), and ST7 (kappa = 88.25%). Algorithm HAT performs well in case of ST4 (kappa = 100%). Poor performance of algorithm is observed in the case of ST8. A glance at Table 3 reveals that in particular ST4 performs in an excellent manner and ST1 is also equally good.

Table 3 Results of kappa statistics for all HTs for data streams from ST1 to ST8

	HT-ACC	*ASHT-ACC*	*HOT-ACC*	*AHOT-ACC*	*HAT-ACC*
ST1	99.53	99.53	99.53	99.53	99.83
ST2	95.03	94.85	95.23	95.22	95.03
ST3	89.85	99.53	89.69	99.53	89.82
ST4	100	100	100	100	100
ST5	85.21	85.21	84.55	85.51	85.21
ST6	95.05	95.05	95.07	95.06	95.02
ST7	91.66	91.22	91.63	91.63	90.42
ST8	74.01	74.01	74.01	74.01	74.01

Figure 10 Graph of data streams (8) vs kappa statistics (for five Hoeffding tree algorithms) (see online version for colours)

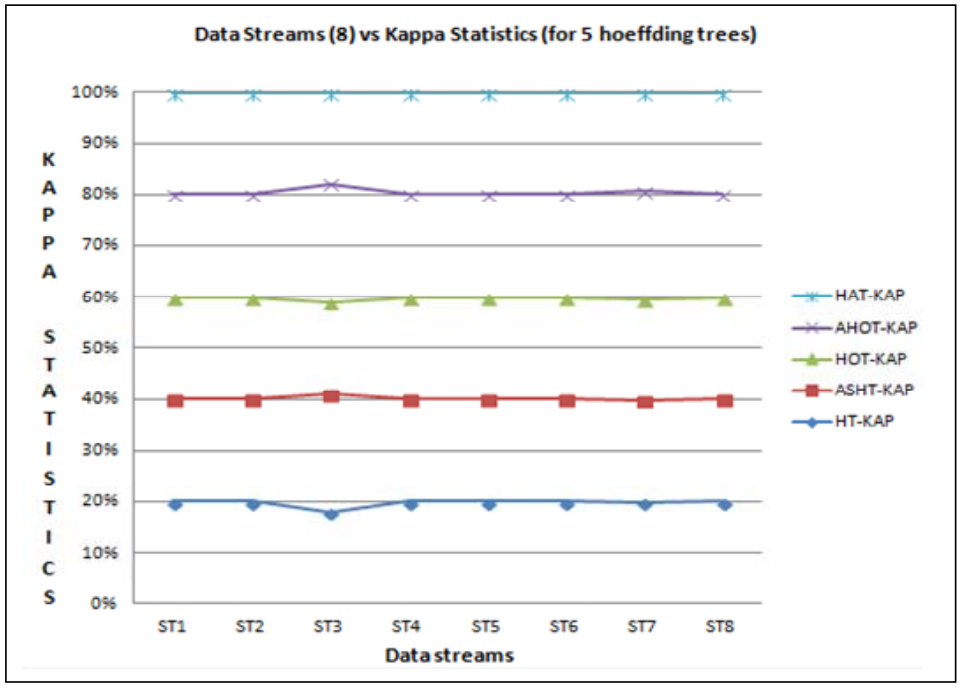
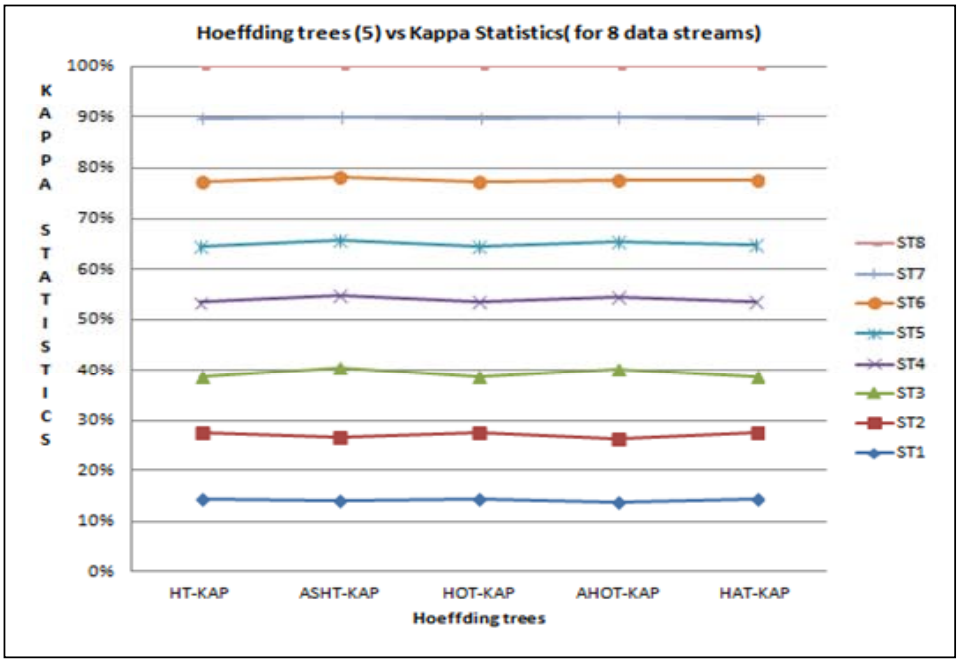


Figure 11 Graph of Hoeffding trees (5) vs kappa statistics (for eight data streams) (see online version for colours)



A glance at Table 4 reveals that the optimum accuracy is obtained by all the algorithms in the case of ST4 (acc = 100%). The algorithm HT performs well in the case of ST4 (acc = 100%), ST6 (acc = 95.05%) and ST7 (acc = 91.66%). Algorithm ASHT performs well in the case of ST3 (acc = 99.53%), ST4 (acc = 100%), and ST6 (acc = 95.05%). HOT performs well in the case of ST2 (acc = 95.23%), ST4 (acc = 100%) and ST6 (acc = 95.07%). Algorithm AHOT performs well in the case of ST2 (acc = 95.22%), ST3 (acc = 99.53%), ST4 (acc = 100%), and ST6 (acc = 95.06%). HAT performs well in the case of ST1 (acc = 99.83%) and ST4 (acc = 100%). Very poor performance is observed in the case of ST8 (acc = 78.01).

Table 4 Results of accuracy for HTs for data streams from ST1 to ST2

	HT-ACC	*ASHT-ACC*	*HOT-ACC*	*AHOT-ACC*	*HAT-ACC*
ST1	99.53	99.53	99.53	99.53	99.83
ST2	95.03	94.85	95.23	95.22	95.03
ST3	89.85	99.53	89.69	99.53	89.82
ST4	100	100	100	100	100
ST5	85.21	85.21	84.55	85.51	85.21
ST6	95.05	95.05	95.07	95.06	95.02
ST7	91.66	91.22	91.63	91.63	90.42
ST8	74.01	74.01	74.01	74.01	74.01

7 Conclusions

Advancement of the technology in both the areas of hardware and software has lead to the enormous storage of data which is referred to as a data stream(s). Quite often, these streams appear in the form of static or evolving streams and in such cases the traditional data mining methods are incapable of handling such data streams. Thus it is a challenging task to know about the storage, analysis and visualisation of such large volumes of data. In literature this is referred to as MDM. The present work is mainly concerned with the understanding of the problem of classification from the data stream perspective and for this purpose eight data streams and five algorithms are considered to carry out the analysis in the MOA framework. The results of the experiments performed are presented in Tables 1–4 and Figures 5–8 respectively. These clearly suggest that:

1 Stream-ST4 takes the minimum time while stream-ST5 takes the maximum time when compared to the other streams for all the five algorithms.

2 The stream ST4 takes least memory while stream-ST5 takes the maximum memory when compared to the other streams. But for the case of HAT, stream ST2 has the maximum memory usage. Further, it is amazing to note that the memory usage for ST4 is almost nil in all the five algorithms.

3 The performance of all the streams is equally good which is indicated by kappa statistics. In particular ST4 performs in an excellent manner and ST1 is also equally good.

4 The optimal accuracy of (100%) is achieved by the ST4 stream and accuracy of ST1 is 99.53% when compared to the other streams.

Finally, it is concluded that the performance of stream ST4 is excellent (100%) accuracy when compared to the other streams. In other words Stagger Generator (ST4) appears to be the best performer in the MOA frame work for different classifiers. The results of the present investigation are unique and provide an excellent platform for future investigation.

It is also concluded that among the online learning methods, online metric learning is not suitable for classification of data steams while it could be addressed to the problems associated with unsupervised settings such as clustering. In such cases generally algorithms like K-means, Mahalanobis, gradient descent, etc., are used. Therefore for the present investigation online metric learning is not suitable.

Acknowledgements

One of the authors Mrs. Malini M. Patil acknowledges JSSMVP's, J.S.S. Academy of Technical Education, Bangalore, Karnataka and Bharathiar University, Coimbatore, Tamilnadu, India for providing the facilities for carrying out the research work. Special acknowledgement is extended to Dr. Albert Bifet for his valuable and timely suggestions.

References

Agarwal, R., Ghosh, S.P., Imielinski, T., Iyer, B.R. and Swami, A.N. (1992) 'An interval classifier for database mining applications', *International Conference on Very Large Data Bases*, pp.560–573.

Aggarwal, C.C. (Ed.) (2007) 'Data streams: models and algorithms', *Series: Advances in Database Systems*, Vol. 31, No. 18, 354p, ebook, Springer, Berlin Heidelberg.

Aha, D. (2007) *UCI Machine Learning Repository* [online] http://www.archive.ics.uci.edu/.

Baker, R.J.D. and Yacef, K. (2009) 'The state of education data mining: a review and future visions', *Journal of Education Data Mining*, Vol. 1, No. 1, pp.3–17.

Bifet, A. and Gavaldà, R. (2009) 'Adaptive learning from evolving data streams', *Advances in Intelligent Data Analysis*, Vol. 8, pp.249–260, Springer Berlin Heidelberg.

Bifet, A. and Kirkby, R. (2009) *Data Stream Mining: A Practical Approach*, Technical report, The University of Waikato, New Zealand.

Bifet, A., Eibe, F., Holmes, G. and Pfahringer, B. (2012) 'Ensembles of restricted Hoeffding trees', *ACM Transactions on Intelligent Systems and Technology (TIST)*, Vol. 3, No. 2, pp.1–20.

Bifet, A., Frank, E., Holmes, G. and Pfahringer, B. (2010) 'Accurate ensembles for data streams combining restricted Hoeffding trees using stacking', *Journal of Machine Learning Research*, pp.225–240.

Bifet, A., Holmes, G., Kirby, R. and Pfahringer, B. (2011) 'MOA: massive online analysis', *Journal of Machine learning Research*, pp.1601–1604.

Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R. and Gavaldà, R. (2009b) 'New ensemble methods for evolving data streams', .*International Conference on Knowledge Discovery in Data Bases*, ACM, pp.139–148.

Bifet, A., Kirkby, R., Kranen, P. and Reutemann, P. (2009a) *Massive Online Analysis*, Technical Manual, University of Waikato, Hamilton, New Zealand.

Breiman, L., Friedman, J.H., Olshen, R.A. and Stone, C.J. (Eds) (1984) *Classification and Regression Trees*, Wadsworth International Group, Belmont, California.

Charikar, M.S. (2002) 'Similarity estimation techniques from rounding algorithms', *STOC '02 Proceedings of the Thirty-fourth annual ACM Symposium on Theory of Computing*, pp.380-388, ACM New York, NY, USA, doi:10.1145/509907.509965.

Davis, J., Kulis, B., Jain, P., Sra, S. and Dhillon, I. (2007) 'Information – theoretic metric learning', *International Conference in Machine Learning.*

Domingos, P. and Hulten, G. (2000) 'Mining time-changing data streams' in *KDD'00, Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp.71–80, ACM Press, NY, USA, doi:10.1145/347090.347107.

Frome, A., Singer, Y. and Malik, J. (2006) 'Image retrieval and classification using local distance function', *Neural Information Processing Systems*, pp417–434.

Gaber, M.M. Zaslavsky, A. and Krishnamurthy. S. (2007) 'Data streams: models and methods', Vol. 31, pp.39–59, Springer, Berlin Heidelberg.

Globerson and Roweis, S. (2005) 'Metric learning by collapsing classes', *Neural Information Processing Systems (NIPS'05)*, Vol. 18, pp.451–458, Vancouver, Canada.

Guha, S., Koudas, N.K. and Shim, K. (2001) 'Data streams and histograms', *ACM Symposium on Theory of Computing.*

Han, J. and Kamber, M. (Eds.) (2007) *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publishers, San Francisco, CA.

Hulten, G., Spencer, L. and Domingos, P. (2001) 'Mining time-changing data streams', in *KDD'01,* ACM Press, pp.97–106.

Ikonomovska, A., Suzana, L. and Gjorgjevik, D. (2007) 'A survey of stream data mining', in Mile J. Stankovski (Ed.): *Proceedings of 8th National Conference with International Participation, ETAI*, pp.I6-2.

Jain, P., Kulis, B., Dhillon, S.I. and Grauman, K. (2008) 'Online metric learning and fast similarity search', *Proceedings of the Neural Information Processing Systems Conference (NIPS)*, December, Vancouver, British Columbia, Canada.

Kirkby, R. (2007) *Improving Hoeffding Trees*, Ph.D. thesis, University of Waikato, Hamilton, New Zealand.

Knauf, R., Boeck, R., Sakurai, Y., Dohi, S. and Tsuruta, S. (2008), 'Knowledge mining for supporting learning processes', *IEEE International Conference on Systems, Man, and Cybernetics,* Singapore.

Pfahringer, B., Holmes, G. and Kirkby, R. (2007) 'New options for Hoeffding trees', *International Conference on Artificial Intelligence.* pp.90–99.

Romero, C. and Ventura, S. (2010) 'Education data mining: a review of the state of art', *IEEE Transactions on Systems, Man and Cybernetics,* Vol. 40, No. 6, pp.601–618.

Schlimmer, J.C. and Granger, R.H (1986) 'Incremental learning from noisy data', *International Conference on Machine Learning*, Vol. 1, No. 3, pp.317–354.

Schultz, M. and Joachims, T. (2003) 'Learning a distance metric from relative comparisons', *Proceedings of the Conference on Advance in Neural Information Processing Systems (NIPS)*, MIT Press, Vancouver and Whistler, British Columbia, Canada.

Shwartz, S.S., Singer, Y. and Ng, A.Y. (2004) 'Online and batch learning of pseudo-metrics', *ICML '04 Proceedings of the Twenty-first International Conference on Machine Learning*, ACM, NY, USA, p.94, doi:10.1145/1015330.1015376.

Shwartz, S.S. and Singer, Y. (2006) 'Online learning meets optimization in dual', in *Lecture Notes in Computer Science*. Vol. 4005, pp.423–437.

Srimani, P.K. and Patil, M.M. (2011) 'Edu-mining: a machine learning approach', *AIP Conference Proceedings 1414*, pp.61–66, doi: http://www.dx.doi.org/10.1063/1.3669932.

Srimani, P.K. and Patil, M.M. (2012a) 'A classification model for Edu-mining', *Proceedings of International Conference on Intelligent Computational Systems*, Dubai, UAE, pp.35–40.

Srimani, P.K. and Patil, M.M. (2012b) 'A comparative study of classifiers for student module in technical education system (TES)', *International Journal of Current Research*, Vol. 4, No. 1, pp.249–254.

Srimani, P.K. and Patil, M.M. (2012c) 'Performance evaluation of classifiers for Edu-data: an integrated approach', *International Journal of Current Research,* Vol. 4, No. 2, pp.183–190.

Srimani, P.K. and Patil, M.M. (2012d) 'Massive data mining on data streams using classification algorithms', *International Journal of Engineering Science and Technology*, Vol. 4, No. 6, pp.2839–2848.

Street, W.N. and Kim, Y. (2001) 'A streaming ensemble algorithm (SEA) for large-scale classification', in *KDD'01*, New York,USA, ACM Press, pp.377–382.

Wang, H., Fan, W., Yu, P.S. and Han, J. (2003) 'Mining concept-drifting data streams using ensemble classifiers', in *Proceedings of 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM Press, pp.226–235.

Weinberger, K., Blitzer, J. and Saul, K. (2009) 'Distance metric learning for large margin nearest neighbour classification', *Journal of Machine Learning Research (JMLR)*, Vol. 10, pp.207–244.

Witten, I.H., Frank, E. and Hall, M.A. (Eds.) (2011) *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann, San Francisco, CA.

Xing, E.P., Ng, A.Y., Jordan, M.I. and Russell, S. (2002) 'Distance metric learning, with application to clustering with side-information', in Becker et al. (Eds.): *Advances in Neural Information Processing Systems 16* (NIPS2002), MIT Press, pp.521–528.