A New Adaptive Ensemble Boosting Classifier for Concept Drifting Stream Data

Kapil K. Wankhade and Snehlata S. Dongre, *Members, IACSIT*

***Abstract*—With the emergence of large-volume and high speed streaming data, mining of stream data has become a focus on increasing interests in real applications including credit card fraud detection, target marketing, network intrusion detection, etc. The major new challenges in stream data mining are, (a) since streaming data may flow in and out indefinitely and in fast speed, it is usually expected that a stream data mining process can only scan a data once and (b) since the characteristics of the data may evolve over time, it is desirable to incorporate evolving features of streaming data. This paper introduced new adaptive ensemble boosting approach for the classification of streaming data with concept drift. This adaptive ensemble boosting method uses adaptive sliding window and Hoeffding Tree with naïve bayes adaptive as base learner. The result shows that the proposed algorithm works well in changing environment as compared with other ensemble classifiers.**

***Index Terms*—Concept drift, ensemble approach, hoeffding tree, sliding window, stream data**

I. Introduction

The last twenty years or so have witnessed large progress in machine learning and in its capability to handle real-world applications. Nevertheless, machine learning so far has mostly centered on one-shot data analysis from homogeneous and stationary data, and on centralized algorithms. Most of machine learning and data mining approaches assume that examples are independent, identically distributed and generated from a stationary distribution. A large number of learning algorithms assume that computational resources are unlimited, e.g., data fits in main memory. In that context, standard data mining techniques use finite training sets and generate static models. Nowadays we are faced with tremendous amount of distributed data that could be generated from the ever increasing number of smart devices. In most cases, this data is transient, and may not even be stored permanently.

Our ability to collect data is changing dramatically. Nowadays, computers and small devices send data to other computers. We are faced with the presence of distributed sources of detailed data. Data continuously flow, eventually at high-speed, generated from non-stationary processes. Examples of data mining applications that are faced with this scenario include sensor networks, social networks, user modeling, radio frequency identification, web mining, scientific data, financial data, etc.

Manuscript received May 25, 2012; revised June 25, 2012.

K. K. Wankhade is with the Department of Information Technolgy, G. H. Raisoni College of Engineering, Nagpur, Maharashtra, INDIA 440019 (kaps.wankhade@gmail.com).

S. S. Dongre is with the Department of Computer Science and Engineering, G. H. Raisoni College of Engineering, Nagpur, Maharashtra, INDIA 440019 (dongre.sneha@gmail.com).

Most recent learning algorithms [1]-[12] maintain a decision model that continuously evolve over time, taking into account that the environment is non-stationary and computational resources are limited.

The desirable properties of learning systems for efficient mining continuous, high-volume, open-ended data streams:

- Require small constant time per data example; use fix amount of main memory, irrespective to the total number of examples.
- Built a decision model using a single scan over the training data.
- Generate an anytime model independent from the order of the examples.
- Ability to deal with concept drift. For stationary data, ability to produce decision models that are nearly identical to the ones we would obtain using a batch learner.

Ensemble method is advantageous over single classifier methods. Ensemble methods are combinations of several models whose individual predictions are combined in some manner to form a final prediction. Ensemble learning classifiers often have better accuracy and they are easier to scale and parallelize than single classifier methods.

The paper organized as, types of concept drift are discussed in Section II. Section III explains proposed method with boosting, adaptive sliding window, hoeffding tree. Experiments and results are included in Section IV with concluding conclusion in Section V.

II. Types of Concept Drift

Change may come as a result of the changing environment of the problem e.g., floating probability distributions, migrating clusters of data, loss of old and appearance of new classes and/or features, class label swaps, etc.

Fig. 1 shows four examples of simple changes that may occur in a single variable along time. The first plot (Noise) shows changes that are deemed non-significant and are perceived as noise. The classifier should not respond to minor fluctuations, and can use the noisy data to improve its robustness for the underlying stationary distribution. The second plot (Blip) represents a ‘rare event’. Rare events can be regarded as outliers in a static distribution. Examples of such events include anomalies in landfill gas emission, fraudulent card transactions, network intrusion and rare medical conditions. Finding a rare event in streaming data can signify the onset of a concept drift. Hence the methods for on-line detection of rare events can be a component of the novelty detection paradigm. The last two plots in Fig. 1

(Abrupt) and (Gradual) show typical examples of the two major types of concept drift represented in a single dimension.

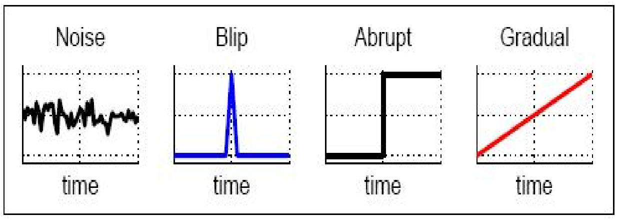


Fig. 1. Types of concept change in streaming data.

III. Proposed Method

This adaptive ensemble method uses boosting, adaptive sliding window and Hoeffding tree for improvement of performance.

A. *Boosting*

Boosting is a machine learning meta-algorithm for performing supervised learning. While boosting is not algorithmically constrained, most boosting algorithms consist of iteratively learning weak classifiers with respect to a distribution and adding them to a final strong classifier. When they are added, they are typically weighted in some way that is usually related to the weak learners' accuracy. After a weak learner is added, the data is reweighted; examples that are misclassified gain weight and examples that are classified correctly lose weight. Boosting focuses on the misclassified tuples, it risks overfitting the resulting composite model to such data. Therefore, sometimes the resulting "boosted" model may be less accurate than a single model derived from the same data. Bagging is less susceptible to model overfitting. While both can significantly improve accuracy in comparison to a single model, boosting tends to achieve greater accuracy. There is reason for the improvement in performance that it generates a hypothesis whose error on training set is small by combining many hypotheses whose error may be large. The effect of boosting has to do with variance reduction. However unlike bagging, boosting may also reduce the bias of the learning algorithm.

B. *Adaptive Sliding Window*

In data streams environment data comes infinitely and huge in amount. So it is impossible to stores and processes such data fast. To overcome these problems window technique comes forward. Window strategies have been used in conjunction with mining algorithms in two ways: one, externally to the learning algorithm; the window system is used to monitor the error rate of the current model, which under stable distributions should keep decreasing or at most stabilize; when instead this rate grows significantly, change is declared and the base learning algorithm is invoked to revise or rebuild the model with fresh data. A window is maintained that keeps the most recent examples and according to some set of rules from window older examples are dropped.

The idea behind sliding window [13] is that, whenever two "large enough" sub-windows of W exhibit "distinct enough" averages, one can conclude that the corresponding expected values are different, and the older portion of the window is dropped. In other words, W is kept as long as possible while the null hypothesis "μ_t has remained constant in W" is sustainable up to confidence δ. "Large enough" and "distinct enough" above are made precise by choosing an appropriate statistical test for distribution change, which in general involves the value of δ, the lengths of the sub-windows and their contents. At every step, outputs the value of $\hat{\mu}_W$ as an approximation to μ_W.

C. *Hoeffding Tree*

The Hoeffding Tree algorithm is a decision tree learning method for stream data classification. It typically runs in sublinear time and produces a nearly identical decision tree to that of traditional batch learners. It uses Hoeffding Trees, which exploit the idea that a small sample can be enough to choose an optimal splitting attribute. This idea is supported mathematically by the Hoeffding bound.

Suppose we make N independent observations of a randomvariable r with range R, where r is an attribute selection measure. In the case of Hoeffding trees, r is information gain. If we compute the mean, $\overline{r}$ of this sample, the Hoeffding bound states that the true mean of r is at least $\overline{r}-\varepsilon$, with probability 1-δ, where δ is user-specified and

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

D. *Adaptive Ensemble Boosting Classifier*

The designed algorithm uses boosting as ensemble method, sliding window and Hoeffding tree for to improve ensemble performance. Figure 1 shows algorithm for the data stream classification. In this algorithm there are m base models, D, a data used for training and initially assigns weight w_k to each base model equal to 1. The learning algorithm is provided with a series of training datasets $\{x_i^t \in X: y_i^t \in Y\}$, i= 1,..., m_t where t is an index on the changing data, hence x_i^t is the ith instance obtained from the ith dataset. The algorithm has two inputs, a supervised classification algorithm, Base classifier to train individual classifiers and the training data D^t drawn from the current distribution $P^t(x, y)$ at time t. When new data comes at time t, the data is divided into number of sliding windows $W_1,\ldots,W_n$. The sliding window is used for change detection, time and memory management. In this algorithm sliding window is parameter and assumption free in the sense that it automatically detects and adapts to the current rate of change. Window is not maintained explicitly but compressed using a variant of the exponential histogram technique [14]. The expected values $\hat{\mu}_W$ = total / width of respective windows are calculated.

If change detect, it raises change alarm. With using expected values algorithm decides which sub-window has dropped or not. Then algorithm generated one new classifier h_t, which is then combined with all previous classifiers to create the composite hypothesis H_t. The decision of H_t serves as the ensemble decision. The error of the composite hypothesis, E^t, is computed on new dataset, which is then used to update the distribution weights. Once the distribution is updated, the algorithm calls the Base classifier & asks it to create the t^{th} classifier h_t using drawn from the current training dataset D^t i.e. $h_t : X \rightarrow Y$. All classifiers are generated for h_k, k=1,…,t are evaluated on the current dataset by computing ε_k^t, the error of the k^{th} classifier h_k at the t^{th} time step.

$$\varepsilon_i^t = \sum_{i=1}^{m^t} D^t(i).[h_k(x_i) \neq y_i] \qquad (2)$$

for k = 1,2,…,t

At current time step t, we have t error measures one for each classifier generated. The error is calculated using equation (2) and then it is used for to update weight. Then the weight is dynamically updated using equation (3),

$$w_k^{'} = (1 - \varepsilon^t_k) / \varepsilon^t_k \qquad (3)$$

Using this updated weight, classifier is constructed. The performance evaluation of classifier is important in concept drifting environment. If the performance of the classifier has going down then this algorithm drops the classifier and add next classifier in ensemble for maintaining ensemble size.

This classifier assigns dynamic sample weight. It keeps the window of length W using only O (log W) memory & O (log W) processing time per item, rather than the O (W) one expects from a na ïve implementation. It is used as *change detector* since it shrinks window if and only if there has been significant change in recent examples, and *estimator* for the current average of the sequence it is reading since, with high probability, older parts of the window with a significantly different average are automatically dropped. The proposed classifier uses Hoeffding tree as a base learner. Because of this algorithm woks faster and increases performance. Basically algorithm is light weight means that it uses less memory. Windowing technique does not store whole window explicitly but instead of this it only stores statistics required for further computation. In this proposed algorithm the data structure maintains an approximation of the number of 1's in a sliding window of length W with logarithmic memory and update time. The data structure is adaptive in a way that can provide this approximation simultaneously for about O(logW) sub windows whose lengths follow a geometric law, with no memory overhead with respect to keeping the count for a single window. Keeping exact counts for a fixed-window size is impossible in sub linear memory. This problem can be tackled by shrinking or enlarging the window strategically so that what would otherwise be an approximate count happens to be exact. More precisely, to design the algorithm a parameter M is chosen which controls the amount of memory used to O(MlogW/M) words.

IV. Experiments and Results

The proposed method has tested on both real and synthetic datasets. The proposed algorithm has compared with OzaBag [15], OzaBoost [15] and OCBoost [16], OzaBagADWIN [17] and OzaBagASHT [17]. The experiments were performed on a 2.59 GHz Intel Dual Core processor with 3 GB RAM, running on UBUNTU 8.04. For testing and comparison MOA [18] tool and Interleaved Test-Then-Train method has used.

```
Algorithm: AEC : an ensemble set of classifiers { C1 , C2 ,… Cm} m <= M
Input :  D, a set of class labeled training tuples
         Base learning algorithm
Output : a composite model
Method:
1.  for base model i=1 to m
2.    initialize window W as an empty list of buckets
3.    initialize width, variance and total
4.    for each t > 0
5.      do setInput (x_t, W)
6.        output μ_w and change alarm
7.        for every window k
8.          compute error rate on every window ε_k^t
9.          if change detected
10.           set new sample weight w_k' = (1- ε_k^t)/ ε_k^t
11.         else
12.           w_k' = 1
13.         end if
14.         update weight
15.         learn classifier
16.       end for
17.   end for
18.   if m=M
19.     add next classifier C_m+1
20.     drop C1
21.   end if
22. end for
```

Fig. 2. Proposed agorithm.

A. Experiment using Synthetic Data

The synthetic data sets are generated with drifting concept concepts based on random radial basis function (RBF) and SEA.

Random Radial Basis Function Generator- The RBF generator works as follows - A fixed number of random centroids are generated. Each center has a random position, a single standard deviation, class label and weight. New examples are generated by selecting a center displacement is randomly drawn from a Gaussian distribution with standard deviation determined by the chosen centroid also determines the class label of the example. This effectively creates a normally distributed hypersphere of examples surrounding each central point with varying densities. Only numeric attributes are generated. Drift is introduced by moving the centroids with constant speed. This speed is initialized by a drift parameter.

SEA Generator- This artificial dataset contains abrupt concept drift, first introduced in [13]. 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 4 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 8, 9, 7 and 9.5 for the data blocks. 10% of class noise was then introduced into each block of data by randomly changing the class value of 10% of instances.

LED Generator- The dataset generated 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.

For all the above datasets the ensemble size has set to 10 and the dataset size in instances has been selected as 1 million. All the experiments are carried out using same parameter settings and prequential method.

Prequential method- In Prequential method the error of a model is computed from the sequence of examples. For each example in the stream, the actual model makes a prediction based only on the example attribute-values. The prequential-error is computed based on an accumulated sum of a loss function between the prediction and observed values.

Comparison in terms of time (in sec) and memory (in MB) has tabulated in Table I and II respectively. The learning curves using RBF, SEA and LED datasets has depicted in Fig. 3, Fig. 4 and Fig. 5.

TABLE I: COMPARISON IN TERMS OF TIME (IN SEC) USING SYNTHETIC DATASET

	RBF-drift=0.001	SEA w=100000	LED driftng attribute=3
OzaBag	132.07	58.21	227.29
OzaBagASHT	**134.64**	**48.48**	309.36
OzaBagADWIN	156.61	85.06	421.41
OzaBoost	163.29	68.58	**239.06**
OCBoost	190.91	90.12	364.54
AEBC	177.57	65.31	528.96

TABLE II: COMPARISON IN TERMS OF MEMORY(IN Mb) USING SYNTHETIC DATASET

	RBF-drift=0.001	SEA w=100000	LED driftng attribute=3
OzaBag	0.926	0.331	0.610
OzaBagASHT	0.163	0.069	0.289
OzaBagADWIN	0.003	0.331	0.149
OzaBoost	0.918	0.411	0.687
OCBoost	0.873	0.222	0.637
AEBC	**0.0005**	0.077	**0.131**

B. *Experiment using Real dataset*

In this subsection, the proposed method has tested on real data sets from UCI machine learning repository- http://archive.ics.edu/ml/datasets.html. The proposed classifier is compared with Ozabag, OzaBagASHT, OzaBagADWIN, OzaBoost and OCBoost. The result is calculated in terms of time, accuracy and memory. The comparison in terms of time, accuracy and memory has tabulated in Table III, Table IV and Table V.

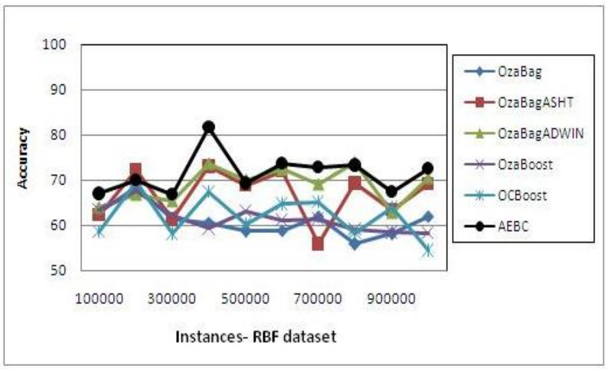


Fig. 3. Learning curves using RBF dataset.

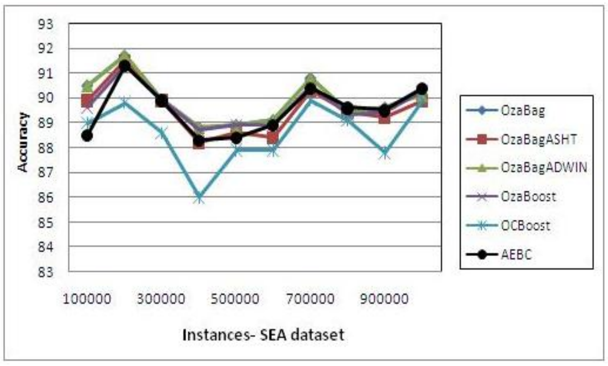


Fig. 4. Learning curves using SEA dataset.

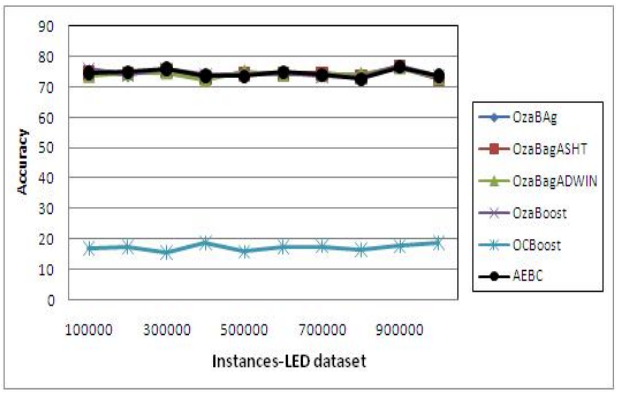


Fig. 5. Learning curves using LED dataset.

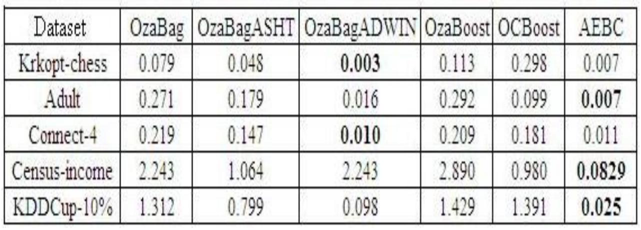
TABLE III: COMPARISON IN TERMS OF TIME(IN SEC) USING REAL DATASET

Dataset	OzaBag	OzaBagASHT	OzaBagADWIN	OzaBoost	OCBoost	AEBC
Krkopt-chess	4.62	4.86	3.30	6.64	13.8	**2.22**
Adult	16.38	13.72	13.34	18.12	14.52	**9.03**
Connect-4	30.26	27.98	**21.16**	33.80	35.35	27.40
Census-income	454.66	265.11	441.18	555.59	203.42	**89.75**
KDDCup-10%	898.15	757.71	**462.46**	1085.79	1341.88	529.00

TABLE IV: COMPARISON IN TERMS OF ACCURACY(IN %) USING REAL DATASET

Dataset	OzaBag	OzaBagASHT	OzaBagADWIN	OzaBoost	OCBoost	AEBC
Krkopt-chess	66.50	73.50	97.10	74.36	10.07	**99.00**
Adult	27.20	78.60	83.60	61.40	45.40	**84.60**
Connect-4	72.90	73.10	69.20	73.70	60.90	**76.80**
Census-income	**94.30**	93.60	**94.30**	93.80	92.00	94.10
KDDCup-10%	98.60	98.99	98.96	98.78	16.29	**99.60**

TABLE V: COMPARISON IN TERMS OF MEMORY(IN Mb) USING REAL DATASET



Dataset	OzaBag	OzaBagASHT	OzaBagADWIN	OzaBoost	OCBoost	AEBC
Krkopt-chess	0.079	0.048	**0.003**	0.113	0.298	0.007
Adult	0.271	0.179	0.016	0.292	0.099	**0.007**
Connect-4	0.219	0.147	**0.010**	0.209	0.181	0.011
Census-income	2.243	1.064	2.243	2.890	0.980	**0.0829**
KDDCup-10%	1.312	0.799	0.098	1.429	1.391	**0.025**

V. CONCLUSION

In this paper we have investigated the major issues in classifying large volume, high speed and dynamically changing streaming data and proposed a novel approach, adaptive ensemble boosting classifier, which uses adaptive sliding window and hoeffding tree for improving performance.

Comparing with other classifier, The adaptive ensemble boosting classifier method achieves distinct features as; it is dynamically adaptive, uses less memory and processes data fastly.

Thus, adaptive ensemble boosting classifier represents a new methodology for effective classification of dynamic, fast-growing and large volume data streams.

REFERENCES

[1] G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Journal on Machine Learning*, vol. 23, no. 1, pp. 69-101, 1996.

[2] Y. Freund and R. Schapire, "Experiments with a new boosting algorithm," in *the Proceeding of International Conference of Machine Learning*, 1996, pp 148-156.

[3] P. Domingos and G. Hulten, "Mining high-speed data streams," in *KDD'01 : Proceedings of the sixth ACM SIDKDD international conference on Knowledge discovery and dta mining, New York, NY, USA, ACM Press,* 2000, pp. 71-80.

[4] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Proc. ACM SIGKDD, San Francisco, CA, USA*, 2001, pp. 97-106.

[5] W. N. Street and Y. Kim, "A streaming ensemble algorithm (sea) for large-scale classification," in *KDD'01 : Proceedings of the seventh ACM SIDKDD international conference on Knowledge discovery and dta mining, New York, NY, USA, ACM Press,* 2001, pp. 377-382.

[6] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept drifting data streams using ensemble classifiers," in *KDD'03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining. New York, NY, USA, ACM Press*, 2003, pp 226-235.

[7] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: a new ensemble method for tracking concept drift," in *3rd International Conference on Data Mining, ICDM'03, IEEE CS Press*, 2003, pp. 123-130.

[8] B. Babcock, M. Datar, R. Motwani, and L. O'Callaghan, "Maintaining variance and k-medians over data stream windows," in *Proceeding of the 22nd Symposium on Principles of Database Systems*, 2003, pp. 234-243.

[9] J. Gama, R. Rocha, and P. Medas, "Accurate decision trees for mining high-speed data streams," in *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* 2003, pp. 523-528.

[10] J. Gama, Pedro Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Advances in Artificial Intelligence - SBIA 2004, 17th Brazilian Symposium on Artificial Intelligence, volume 3171 of Lecture Notes in Computer Science, Springer Verlag*, 2004, pp. 286–295.

[11] J. Z. Kolter and M. A. Maloof, "Using additive expert ensembles to cope with concept drift," in *Proc. International conference on Machine learning (ICML), Bonn, Germany*, 2005, pp. 449-456.

[12] G. Cormode, S. Muthukrishnan, and W. Zhuang, "Conquering the divide: Continuous clustering of distribututed data streams," in *ICDE*, 2007, pp. 1036-1045.

[13] Albert Bieft and Ricard Gavald`a, "Learning from time changing data with adaptive windowing," in *SIAM International Conference on Data Mining*, 2007, pp. 443-449.

[14] M. Datar, A. Gionis, P. Indyk, and R. Motwani, "Maintaining stream statistics over sliding windows," *SIAM Journal on Computing*, vol. 14, no. 1, pp 27-45, 2002.

[15] N. Oza and S. Russell, "Online bagging and boosting," in *Artificial Intelligence and Statistics, Morgan Kaufmann*, 2001, pp. 105-112.

[16] R. Pelossof, M. Jones, I. Vovsha, and C. Rudin., "Online coordinate boosting, http://arxiv.org/abs/0810.4553," 2008.

[17] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavalda, "New ensemble methods for evloving data streams," in *KDD'09, ACM, Paris*, 2009, pp. 139-148.

[18] Goeffrey Holmes, Richard Kirkby and Bernhard Pfahringer. "Moa: Massive [Online]. Analysis, http://sourceforge.net/projects/moa-datasream," 2007.

K. K. Wankhade He received B. E. degree in Information Technology from Swami Ramanand Teerth Marathwada University, Nanded, India in 2007and M. Tech. degree in Computer Engineering from University of Pune, Pune, India in 2010.

He is currently working as Assistant Professor the Department of Information Technology at G. H. Raisoni College of Engineering, Nagpur, India. Number of publications is in reputed Journal and International conferences like Springer and IEEE. His research is on Data Stream Mining, Machine Learning, Decision Support System, Artificial Neural Network and Embedded System. His book has published on titled Data Streams Mining: Classification and Application, LAP Publication House, Germany, 2010.

Mr. Kapil K. Wankhade is a member of IACSIT and IEEE organizations. He is currently working as a reviewer for Springer's Evolving System Journal and IJCEE journal.

S. S. Dongre She received B. E. degree in Computer Science and Engineering from Pt. Ravishankar Shukla University, Raipur, India in 2007 and M. Tech. degree in Computer Engineering from University of Pune, Pune, India in 2010.

She is currently working as Assistant Professor the Department of Computer Science and Engineering at G. H. Raisoni College of Engineering, Nagpur, India. Number of publications is in reputed International conferences like IEEE and Journals. Her research is on Data Stream Mining, Machine Learning, Decision Support System, ANN and Embedded System. Her book has published on titled Data Streams Mining: Classification and Application, LAP Publication House, Germany, 2010.

Ms. Snehlata S. Dongre is a member of IACSIT, IEEE and ISTE organizations.