Online Batch Weighted Ensemble for Mining Data Streams with Concept Drift

Magdalena Deckert

Institute of Computing Science, Poznań University of Technology,
60-965 Poznań, Poland
magdalena.deckert@cs.put.poznan.pl

Abstract. This paper presents a new framework for dealing with two main types of concept drift (sudden and gradual) in labeled data with decision attribute. The learning examples are processed instance by instance. This new framework, called Online Batch Weighted Ensemble, introduces element of incremental processing into a block-based ensemble of classifiers. Its performance was evaluated experimentally on data sets with different types of concept drift and compared with the performance of Accuracy Weighted Ensemble and Batch Weighted Ensemble. The results show that OBWE improves value of the total accuracy.

1 Introduction

Mining streaming data is one of the recent challenges in data mining. Data streams are characterized by a large amount of data arriving at rapid rate and require efficient processing [8]. Moreover, the data may come from non-stationary sources, where underlying data distribution changes over time. It causes modifications in the target concept definition, which is known as *concept drift* [7]. The main types of changes are usually divided into sudden or gradual concept drifts depending on the rate of changes [11].

Classical static classifiers are incapable of adapting to concept drifts, because they were learned on the out-of-date examples. This is the reason why their predictions become less accurate with time. Some methods have already been proposed to deal with the concept drift problem [7]. They can be divided into two main groups: trigger based and evolving [13]. *Trigger*-based methods use a change detector to identify the occurrence of a change. If the change is detected, then the online classifier, connected with the detector, is re-trained [9]. One of the most popular detectors is DDM described in [6]. On the other hand, *evolving methods* attempt to update their knowledge without explicit information whether the change occurred. An example of such methods is an adaptive ensemble. This paper focuses mainly on block-based ensembles, which component classifiers are constructed on blocks (chunks) of training data. In general, a block-based approach operates in a way that when a new block is available, it is used for evaluation of already existing component and for creation of a new classifier. The new component usually replaces the worst one in the ensemble. For review of those methods see e.g. [7].

The best representative of evolving methods is Accuracy Weighted Ensemble (AWE) proposed in [12]. According to [3, 12], AWE is sensitive to the defined size of a block. Moreover, AWE is very demanding with respect to the memory and time cost, because it builds a new classifier for every incoming block of data. These were motivations for introducing Batch Weighted Ensemble (BWE) in [4]. It incorporates the Batch Drift Detection Method (BDDM) into the AWE inspired structure of the ensemble. Conducted experimental analysis [5] showed that BWE decreases the performance costs, while the total accuracy of classification is held on satisfying level. However, the reaction to the sudden drift, which appears inside the data block is not sufficient—it is delayed until the end of processed block. This was the reason for developing a new environment called Online Batch Weighted Ensemble (OBWE).

The main aim of this paper is to present the new framework for dealing with two main types of concept drift: sudden and gradual drift. This new framework introduces incremental processing of learning instances into the BWE block-based classifier. Its performance was evaluated experimentally on data sets with different types of concept drift and compared with performance of the AWE and standard BWE classifier. Evaluation criteria, on which ensembles will be compared, are: accuracy of classification, use of memory and processing time.

This paper is organized as follows. The next section presents related works on detecting concept drift and block ensembles. Section 3 describes the framework Online Batch Weighted Ensemble. Section 4 is devoted to the experimental evaluation of classifiers for various types of changes. Section 5 concludes this paper.

2 Related Works

This section concentrates on methods that are most related to the presented research study. For reviews of other approaches see [7–9, 11, 13].

First, the Drift Detection Method (DDM) [6] is presented, because it inspired BWE solution. This detector is used in combination with an online classifier. The main idea of DDM is to monitor the error-rate produced by the classifier. For each incoming example the classifier predicts a class label, which is compared with the original (true) one. Classification errors are modeled with a Binomial distribution. When the error increases, it signifies that the data distribution has changed. DDM signals two levels of change: warning and drift according to the sigma rule. When a warning level was exceeded, learning examples are stored in a special buffer. They are collected until drift level is reached. If the alarm level is reached, the previously taught classifier is removed and a new classifier is built from buffer examples. It is possible to observe warning level, followed by a decrease in error rate. This situation is treated as a *false alarm* and the model is not refined. DDM is independent from the learning algorithm and can be easily extended to processing data in blocks.

Evolving methods do not use explicit drift detection. An example of such methods are ensembles of classifiers. They have a natural ability to process data that arrive in blocks. The main idea of the block-based approach is to build a new classifier for each incoming block of data and then to use this block to evaluate performance of all components existing in the ensemble. Remaining base classifiers receive a weight reflecting their performance. In case when the number of base classifiers is restricted to k, the less accurate one is replaced by the new one. The final answer of the ensemble is constructed by combining single components' votes using weighted majority voting. In AWE classifier, proposed by Wang et al. in [12], each weight w_i of a component is estimated with the formula $w_i = MSE_r - MSE_i$, where MSE_r is mean square error of a random classifier and MSE_i is mean square error of the ith classifier. MSE_r can be calculated as $MSE_r = \sum_c p(c) * (1 - p(c))^2$, where $p(c)$ is the estimation of probability of observing class c in the last block of data. The MSE_i can be expressed by $MSE_i = \frac{1}{|S_n|} \sum_{(x,c) \in S_n} (1 - f_c^i(x))^2$, where S_n is the last block of data and $f_c^i(x)$ is the probability obtained from the classifier i that example x is an instance of class c. In each iteration, k best base classifiers are chosen to form the final ensemble.

According to [12], AWE is sensitive to the chosen block's size. Moreover, due to creating a new classifier in every iteration, AWE has high memory and time requirements. These were motivations for inventing Batch Weighted Ensemble (BWE) in [4]. Next, BWE was improved and fully examined in [5]. The main idea of BWE environment is incorporation of Batch Drift Detection Method (BDDM) into the AWE inspired block-based ensemble. In contrary to typical drift detectors instead of processing instance by instance, BDDM operates on blocks of data. For each example in the block, an accuracy of classification and a standard deviation are calculated incrementally. Next, on the obtained table of accuracy values, a linear regression model is found. It is used to estimate the trend in the data, without finding an ideal adjustment. In the next step, the slope a of the regression model is tested. Value less than 0 means that some change occurred. BDDM distinguishes between two levels of change: warning and drift. If the value of the slope a is less than 0, then default change level is warning. In the end, it is checked whether drift level was obtained. The threshold for the drift was inspired by the DDM [6] and is established using the sigma rule in standardized normal distribution. For more details on BDDM see [4, 5]. Batch Drift Detection Method is incorporated into an ensemble called Batch Weighted Ensemble. BWE operates as follows. In case when the ensemble is empty, a defined number of components is build on bootstrap samples created from the actual block of the data. Alternatively, BWE uses BDDM to check whether the change appeared. If BDDM signals warning or drift level, the weight of every base classifier is computed using appropriate formula. If the maximum size of an ensemble is exceeded, then the base classifier with the lowest weight is removed. If the detector signals drift level, this means that the ensemble must undergo major changes—must be pruned. That is why base classifiers, whose classification accuracy is lower than random guessing, are removed. When all of the ensemble's components are removed, half of them with the highest weights are restored. This is done in order to preserve some of the past knowledge and to avoid learning from scratch. In the end, for every change level, BWE builds a new classifier on the current block, calculate its weight and adds it to the ensemble. For details on BWE environment see [4, 5]. Experimental results of BWE environment showed the usefulness of incorporating a drift detector inside the block-based adaptive ensembles. Proposed integration reduces computational costs, while the total accuracy of classification is held on satisfying level.

3 Online Batch Weighted Ensemble

Researches on block-based ensembles of classifiers showed that they may insufficiently well react to concept drift appearing inside the data block [10]. They may delay its reaction to the occurring change till the end of actually processed block. This was the reason of developing a new environment called Online Batch Weighted Ensemble (OBWE). The main idea of proposed framework is to introduce an element of incremental processing into the BWE block-based approach. OBWE environment consists of explicit change detector called Online Batch Drift Detection Method and the OBWE ensemble of classifiers.

Algorithm 1: Online Batch Drift Detection Method

```
Input  : $C$: an ensemble of classifiers; $w$: weights for current ensemble of
         classifiers; $e$: a learning example; $r$: a regression window size; $b$: size of
         the data block
Output: $signal$: flag indicating type of discovered signal

calculate incremental accuracy of classification for example $e$ using ensemble $C$
with weights $w$;
calculate standard deviation incrementally;
update table containing previous classification accuracies;
if (regression window size $r$ was exceeded) then
    create regression function on incremental accuracy table;
    if ($a<0$) then  $\Longleftarrow$ test the slope $a$ of the regression model
        if ($currentAvgAccuracy - currentStdDev < maxAccuracy - 3 * maxStdDev$)
        then
            $signal$ = drift;
        else
            $signal$ = warning;
    if ($signal$ = drift) then
        reset important fields;
if (block size $b$ was exceeded) then
    store statistics only for the last examples;
Return $signal$
```

OBDDM modifies the standard BDDM detector in a way, that it processes every single learning example separately. First, the value of accuracy of classification is incrementally updated according to the result of prediction of the ensemble C with weights w for the learning example e. Obtained value is added to the table with classification accuracies. After every r learning examples, where r is the size of regression window, a linear regression model is found on the whole table with collected accuracies. This shows the tendencies present in the data. The slope a of the regression model less than 0 means that some change exists in the data. If the change was detected, current average value of accuracy is calculated as a mean value from the accuracies stored in the table. Current standard deviation is also obtained from the accuracies table. OBDDM uses the same thresholds for warning and drift levels as the base BDDM detector. After the drift, all important fields and statistics (e.g. accuracy table, current accuracy of classification, maximum values of accuracy and standard deviation) are cleared. In case when the size of a block was exceeded, the statistics calculated for the recent learning examples are stored in order to preserve the information about the trend between two subsequent blocks of data. The pseudo-code of OBDDM is given as Algorithm 1.

OBBDM is incorporated with the OBWE ensemble. Instead of processing streaming data in blocks, it allows reaction after every learning example. However, it maintains a fixed number of recent learning examples as sliding window. The maximum size of the sliding window is restricted to the block size. OBWE operates as follows. When a new learning example is available, it is added to the sliding window. OBWE procedure is executed, if the number of stored learning instances equals the block's size. First, the actual number of component classifiers in the ensemble is checked. If the ensemble is empty, the number of components is constructed on the bootstrap samples achieved from the sliding window. Otherwise, OBDDM is launched in order to check whether the change is present. If OBDDM signals warning level, the weight of every base classifier is computed using formula $w_i = 0.5 * (1 - \frac{e^{6*(x-0.5)} - e^{-6*(x-0.5)}}{e^{6*(x-0.5)} + e^{-6*(x-0.5)}})$. The reason for multiplication by 0.5 is that the codomain of this function is in range $(0; 1)$. Thanks to this all of the existing base classifiers have the same impact in the final answer of the ensemble. Moreover, proposed function w_i, for small values of x, decreases slower. The rate of change is controlled by the multiplication by 6. This value was established empirically by observing the variation of the function w_i—the higher the value is, then the decrease of the function w_i is slower at the beginning and very rapid near the inflection point. For $x = 0.5$ is the inflection point, from which the function w_i decreases faster. The function w_i is symmetrical about the inflection point. The reason for such characteristic is that for warning lever there is no necessity to decrease weights of component classifiers so severely. After weights' update, OBWE checks if the maximum size of an ensemble is exceeded. If so, then the base classifier with the lowest weight is removed. Next, OBWE builds a new classifier on the current window of learning examples, calculate its weight as $w' = maxEnsembleSize - \sum w_j$ and adds it to the ensemble. The newly created classifier obtains such a high weight in order to raise its importance in the final answer of the ensemble. The reason for such assistance is that the new component has the most current knowledge induced from the recent block of data. If the detector signals drift, weights of the existing components are altered with formula $w_i = 0.5 * (1 - \frac{e^{4*(x-0.25)} - e^{-4*(x-0.25)}}{e^{4*(x-0.25)} + e^{-4*(x-0.25)}})$.

Algorithm 2: Online Batch Weighted Ensemble

```
Input : $S$: a data stream of examples; $b$: size of the data block; $r$: size of the
        regression window; $bsC$: number of bootstrap classifiers; $maxC$:
        maximum number of classifiers in ensemble; $C$: a set of previously
        trained classifiers
Output: $C$: an updated set of classifiers; $w$: an updated weights for current
        ensemble of classifiers

foreach (learning example $s_i \in S$) do
    add example $s_i$ to the learning window;
    if (size of learning window = size of a block b) then
        if (size of ensemble = 0) then
            foreach (j = 1 .. bsC) do
                train classifier $C_j$ on $bootstrapSample(window)$;
                $C \leftarrow C \cup C_j$;
                $w_j = maxC/bsC$;
        else
            OBDDM ($C$, $w$, $s_i$, $r$, $b$); {build Online Batch Drift Detection Method}
            if (signal=warning) then
                foreach (classifier $C_j \in C$) do
                    compute $w_j$ using a formula:
                    $w_j = 0.5 * (1 - \frac{e^{6*(x-0.5)} - e^{-6*(x-0.5)}}{e^{6*(x-0.5)} + e^{-6*(x-0.5)}})$;
                if (memory buffer is full) then
                    remove classifier with the lowest weight;
                train classifier $C'$ on current learning window;
                $C \leftarrow C \cup C'$;
                compute weight $w'$ of classifier $C'$ as:
                $w' = maxEnsembleSize - \sum w_j$;
            else if (signal=drift) then
                foreach (classifier $C_i \in C$) do
                    compute $w_j$ using a formula:
                    $w_j = 0.5 * (1 - \frac{e^{4*(x-0.25)} - e^{-4*(x-0.25)}}{e^{4*(x-0.25)} + e^{-4*(x-0.25)}})$;
                if (memory buffer is full) then
                    remove classifier with the lowest weight;
                foreach (classifier $C_j \in C$) do
                    if ($w_j <= \frac{1}{classes_{no}}$) then
                        remove classifier $C_j$;
                if (size of ensemble = 0) then
                    restore half of the best classifiers with their weights;
                train classifier $C'$ on current learning window;
                $C \leftarrow C \cup C'$;
                compute weight $w'$ of classifier $C'$ as:
                $w' = maxEnsembleSize - \sum w_j$;
        remove the oldest example from learning window;
Return $C$
```

All of the parameters' values were also established empirically by observing the variation of the function w_i. Proposed function for recalculating weights has the inflection point for $x = 0.25$. Additionally, thanks to multiplication by 4, it decreases faster than the one for warning. The reason for this behavior is when the sudden drift is detected the components must be punished more quickly and severely for their mistakes. Next, OBWE prunes the ensemble—the base classifiers, whose accuracy of classification is less than $\frac{1}{|classes|}$, are removed. In case when all of the ensemble's components are removed, half of them with the highest weights are restored. Then, OBWE builds a new classifier on the current window of learning examples, calculate its weight as $w' = maxEnsembleSize - \sum w_j$ and adds it to the ensemble. In the end, OBWE removes only one—the oldest learning example from the stored sliding window. The pseudo-code of OBWE is given as Algorithm 2.

4 Experimental Evaluation

Three different classifiers were chosen for experimental comparison: two block-based approaches AWE, BWE with BDDM and OBWE environment. All classifiers were implemented in Java and were embedded into the Massive Online Analysis framework for mining streaming data. More about the MOA project can be found in [1] and at the website [1]. All base classifiers were constructed using the C4.5 decision tree algorithm (WEKA's J48)—to be consistent with the related works [5, 12]. Unpruned version of the tree was used in order to obtain a more precise description of the current block. Thanks to this the component classifiers will reflect only knowledge obtained from one block of data, so they will be more specialized for different knowledge regions.

Only one block size equals 1000 was tested. However, three different sizes of regression window were checked: 10, 100 and 1000. To estimate classification performance the EvaluateInterleavedTestThanTrain method from MOA was used. It first uses each example in the stream to assess the classification accuracy and than this example is used to re-train (update) the classifier. Evaluation measures were recorded after every 100 examples. Besides the total classification accuracy also values of accumulated processing time (from the beginning of the learning phase) and the size of current model (expressed by memory size) were logged.

All experiments were carried out on datasets with different types of changes, such as gradual drifts, sudden changes, complex (mixed) changes, blips (representing rare events, which should not be treated as real drifts) and no drifts (for which a classifier should not be updated). Nine different datasets were used: three real datasets, which are often considered by other researchers and six artificial datasets obtained using MOA generators, which precise descriptions can be found in MOA Manual [2]. Detailed characteristics of the datasets are given in Table 1.

Due to the page limits only the most representative results are presented. All compared algorithms were evaluated in terms of classification accuracy, memory

[1] see: http://moa.cs.waikato.ac.nz/

Table 1. Characteristics of datasets

Dataset	Examples	Attributes	Classes	Change type	Parameters
CovType	581012	54	7	unknown	N/A
Electricity	45312	8	2	unknown	N/A
Poker	829201	11	10	unknown	N/A
Hyperplane	100000	10	4	gradual	slow change: t=0.001
RBFGradual	100000	20	4	gradual	p=5001, a=45, w=1000
STAGGER	100000	3	2	sudden	p=3001, a=90, w=1
RBFSudden	100000	20	4	sudden	p=5001, a=90, w=1
RBFBlips	100000	20	4	blips	p=24990, a=80, w=200
RBFNoDrift	100000	10	2	N/A	default

usage and total processing time. The accuracy values and memory were averaged over recorded time points. The values of all measures on datasets are presented in Tables: 2, 3 and 4. They will be interpreted in the next section. For better insight into dynamics of learning figures after processing every learning example were plotted. Again, due to the space limits only the representative figures are presented—see Figures: 1, 2 and 3.

Dataset	AWE	BWE	OBWE-R10	OBWE-R100	OBWE-R1000
CovType	81,52	82,60	**85,54**	83,06	82,13
Electricity	73,53	71,41	74,20	**75,67**	71,77
Poker	78,32	75,49	81,18	**82,11**	77,12
Hyperplane	70,91	77,11	78,13	**81,90**	77,87
RBFGradual	75,25	74,49	82,68	**84,16**	78,31
STAGGER	**78,30**	**78,30**	77,65	74,87	77,36
RBFSudden	75,37	74,40	**83,68**	82,67	78,18
RBFBlips	88,41	85,55	87,86	**89,12**	85,90
RBFNoDrift	88,01	87,41	86,27	**88,22**	87,31

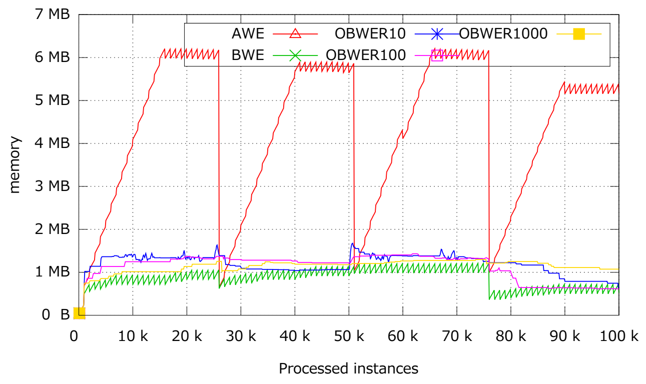
Table 2. Average values of classification accuracy [%]

Dataset	AWE	BWE	OBWE-R10	OBWE-R100	OBWE-R1000
CovType	5,49	0,79	1,12	1,19	1,13
Electricity	0,76	0,58	0,82	0,79	0,68
Poker	1,48	1,21	1,47	1,45	1,31
Hyperplane	0,63	1,06	1,28	1,25	1,21
RBFGradual	1,40	0,42	1,17	1,20	0,70
STAGGER	0,50	0,07	0,18	0,17	0,15
RBFSudden	1,40	0,43	1,11	1,13	0,68
RBFBlips	4,13	0,82	1,13	1,08	1,09
RBFNoDrift	4,02	0,79	1,02	0,99	0,87

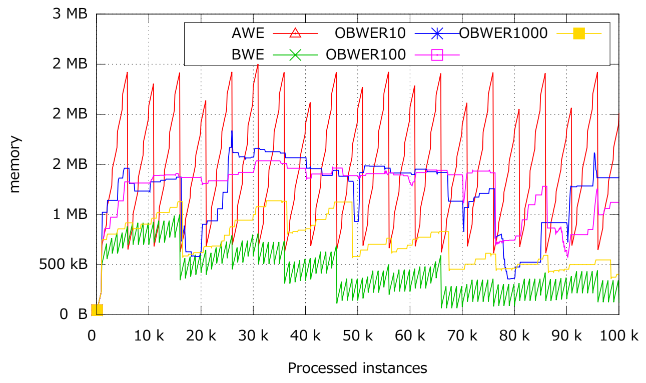
Table 3. Average amounts of used memory [MB]

Dataset	AWE	BWE	OBWE-R10	OBWE-R100	OBWE-R1000
CovType	897,01	338,30	837,51	258,87	163,35
Electricity	20,83	11,23	30,61	15,12	11,09
Poker	629,35	287,56	617,83	380,71	290,04
Hyperplane	35,74	37,27	201,52	54,05	37,67
RBFGradual	68,00	20,34	92,34	48,36	23,51
STAGGER	33,09	3,96	18,55	7,74	6,02
RBFSudden	68,50	20,69	118,84	46,82	23,79
RBFBlips	188,64	31,54	215,64	49,03	33,48
RBFNoDrift	228,14	28,25	201,55	39,25	27,21

Table 4. Total processing time [s]



(a) RBFBlips



(b) RBFGradual

Fig. 1. Memory usage for selected datasets

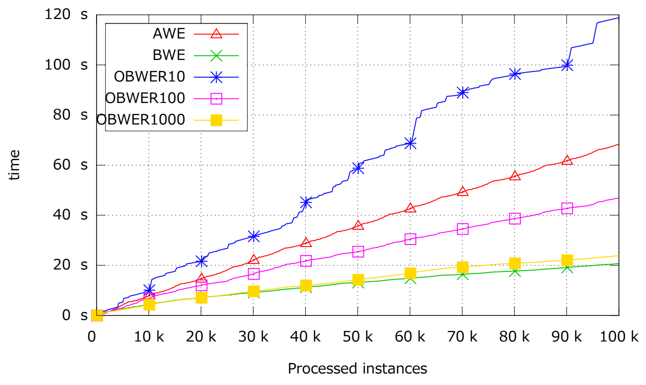
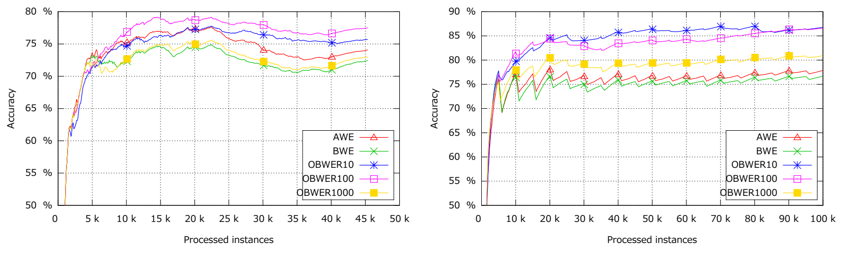
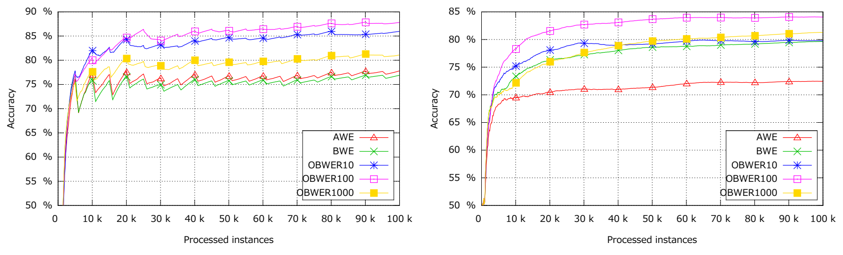


Fig. 2. Processing time for selected dataset



(a) Electricity (b) RBFSudden



(c) RBFGradual (d) Hyperplane

Fig. 3. Classification accuracy for selected datasets

5 Discussion of Results and Final Remarks

In this paper, a new framework for dealing with two main types of concept drift: sudden and gradual drift was presented. This framework, called Online Batch Weighted Ensemble, introduces incremental processing of learning instances into the BWE block-based environment.

After comparing results of total accuracy of classification, one can notice that OBWE obtains the highest value of this measure. The reason of this behavior is that, thanks to incremental processing, OBWE can react to the change more quickly. Instead of waiting until the end of the block, it can re-train immediately after the change was detected. Moreover, in most of the cases, OBWE with the size of regression window equals 100 is the best with respect to the total accuracy of classification. The second position belongs to OBWE with the size of regression window equals 10. AWE achieves the average value of classification accuracy. The worst, with respect to the accuracy of classification, are OBWE with the size of regression window equals 1000 and the standard BWE environment.

Results obtained on memory showed that AWE uses the most amount of memory. The standard BWE environment have the lowest memory requirements. OBWE, in comparison to the standard BWE, needs more memory, however it is still smaller amount than AWE demands. Moreover, in most of the cases, the lower the size of regression window is, the higher size of memory is needed.

Comparison of processing time showed that the standard BWE environment is the fastest classifier. OBWE with regression window size 1000 works for similar period of time as BWE. OBWE with regression size 100 operates longer than the standard BWE environment but is much faster than AWE and OBWE with regression size 10. Those two classifiers are the slowest ones. In more than a half of the cases, OBWE with regression window size 10 operates the longest and less time needs AWE classifier. In case when AWE is the slowest one, then OBWE with regression window size 10 works only a little bit faster than AWE.

In majority of cases, the type of change existing in the dataset does not influence the obtained results. However, for dataset with blips and when there does not exist any change, the advantage of BWE and OBWE over AWE is visible in memory usage—AWE is 4 to 5 times more demanding.

To sum up, the experimental evaluation on nine data sets with different types of drift showed that OBWE improves reaction to the drift, which results in higher classification accuracy. On the other hand, incremental processing is more demanding with respect to the evaluation measures like memory usage and processing time. Moreover, experiments showed that it is unprofitable to highly decrease the size of regression window. The performance requirements rise with decreasing size of regression window but the gain on accuracy of classification is not so significant comparing to the average regression window size 100.

The future research may consider integration of the proposed Online Batch Weighted Ensemble environment with an incremental learning algorithm e.g. Very Fast Decision Trees (VFDT).

References

1. Bifet A., Holmes G., Kirkby R., Pfahringer B.: MOA: Massive Online Analysis., Journal of Machine Learning Research (JMLR), vol. 11, pp. 1601-1604, 2010.
2. Bifet A., Kirkby R.: Massive Online Analysis Manual., COSI, 2009.
3. Brzezinski D., Stefanowski J: Accuracy updated ensemble for data streams with concept drift. Proceedings of HAIS Part II, LNAI, vol. 6679, pp. 155-163, 2011.
4. Deckert M.: Batch Weighted Ensemble for Mining Data Streams with Concept Drift., Proceedings of the 19th International Conference on Foundations of Intelligent Systems (ISMIS 2011), Warsaw, Poland, LNCS, vol. 6804, pp. 290-299, 2011.
5. Deckert M., Stefanowski J.: Comparing Block Ensembles for Data Streams with Concept Drift, New Trends in Databases and Information Systems, Advances in Intelligent Systems and Computing, vol. 185, pp. 69-78, 2012.
6. Gama J., Medas P., Castillo G. and Rodrigues P.: Learning with Drift Detection., In SBIA Brazilian Symposium on Artificial Intelligence, LNAI, vol. 3171, pp. 286-295, 2004.
7. Gama J.: Knowledge Discovery from Data Streams., CRC Publishers 2010.
8. Kuncheva L.I.: Classifier ensembles for changing environments., In Proceedings of the 5th International Workshop on Multiple Classifier Systems (MCS2004), Italy, LNCS, vol. 3077, pp. 1-15, 2004.
9. Kuncheva L.I.: Classifier ensembles for detecting concept change in streaming data: Overview and perspectives., In Proceedings of the 2nd Workshop SUEMA 2008 (ECAI 2008), Greece, pp. 5-10, 2008.
10. Nishida K., Yamauchi K., Omori T.: ACE: Adaptive Classifiers-Ensemble System for Concept-Drifting Environments., Multiple Classifier Systems, LNCS, vol. 3541, pp. 176-185, 2005.
11. Tsymbal A.: The problem of concept drift: Definitions and related work. Technical Report, Trinity College, Dublin, Ireland, 2004.
12. Wang H., Fan W., Yu P.S., Han J.: Mining concept-drifting data streams using ensemble classifiers., In Proceedings of the ACM SIGKDD, pp. 226-235, 2003.
13. Zliobaite I.: Learning under Concept Drift: an Overview. Technical Report, Vilnius University, Lithuania, 2009.