Active Learning Classifier for Streaming Data

Michał Woźniak[1(✉)], Bogusław Cyganek[2,3], Andrzej Kasprzak[1], Paweł Ksieniewicz[1], and Krzysztof Walkowiak[1]

[1] Department of Systems and Computer Networks, Faculty of Electronics, Wrocław University of Technology, Wybrzeże Wyspiańskiego 27, 50-370 Wroclaw, Poland
{michal.wozniak,andrzej.kasprzak,pawel.ksieniewicz, krzysztof.walkowiak}@pwr.edu.pl
[2] ENGINE Center, Wrocław University of Technology, Wybrzeże Wyspiańskiego 27, 50-370 Wroclaw, Poland
cyganek@agh.edu.pl
[3] AGH University of Science and Technology, Al. Mickiewicza 30, 30-059 Krakow, Poland

Abstract. This work reports the research on active learning approach applied to the data stream classification. The chosen characteristics of the proposed frameworks were evaluated on the basis of the wide range of computer experiments carried out on the three benchmark data streams. Obtained results confirmed the usability of proposed method to the data stream classification with the presence of incremental concept drift.

Keywords: Pattern classification · Data stream classification · Active learning

1 Introduction and Related Works

Designing the appropriate models for data stream classification is nowadays focus of intense research, because canonical classifiers usually do not take into consideration that the statistical dependencies between the observations of incoming objects and their classifications may change during. Such phenomena is called *concept drift* [13] and we may meet it in many practical issues, as spam filtering, intrusion detection/prevention, or recognition of client behaviour to enumerate only a few. There are several taxonomies of concept drift. The first one is considering its rapidity, then we can distinguish sudden concept drift and smooth one. We can also distinguish two types of such an event, according to its influence on the probabilistic characteristics of the classification task [6]:

- virtual concept drift means that changes do not impact the decision boundaries (some say the *posterior* probabilities do not change, but it is disputable), but affect the probability density functions [12].
- real concept drift means that changes affect the *posterior* probabilities and may impact unconditional probability density function [13].

F. Martínez-Álvarez et al. (Eds.): HAIS 2016, LNAI 9648, pp. 186–197, 2016.
DOI: 10.1007/978-3-319-32034-2_16

From the classification point of view, the real concept drift is important because it can strongly affect the shape of the decision boundary. The virtual drift does not affect the decision rule, especially taking into consideration the Bayes decision theory [5]. Additionally, during the designing a data stream classifier we should take also into consideration the limitation of the resources as memory and computational power, as well that we are not able have access all labels of learning examples. In this work we will mostly focus on the problem related to cost of labelling. The most of the classifier devoted to data stream or drifted data streams use supervised learning algorithms, which could produce a classifier on the basis of labelled examples. Unfortunately, from the practical point of view it is hard to be granted that labels are always available, e.g.,

- Medical diagnosis - human expert should always verifies the diagnosis, i.e., we have to ensure the continuous access to the human expert.
- Credit application (the true label is available ca. 2 years after the decision);
- Spam filtering - user should confirm the decision if incoming mail is legitimate or not.

In this approach we should take into consideration the cost of data labelling, which is usually passed over. Let us notice that labels are usually assigned by human experts and therefore they can not label all new examples if they come too fast. Therefore methods of classifier design which could produce the recognition system on the basis of a partially labelled set of examples (especially if learner could point out the interesting example to be labelled) [7].

Changes are discovered by monitoring the unlabelled input data and discover novelties related to outlier detection, or by monitoring classification accuracy [9]. Constant update of a classifier is accomplished by using incremental learning methods that allow adding new training data during the exploitation of a classifier or by dataset windowing.

Therefore, developing methods which are able to effectively deal with this phenomena has become an increasing issue in the intense researches. In general, the following approaches can be considered to deal with the above problem.

- Rebuilding a classification model if new data becomes available. It is very expensive and impossible from a practical point of view, and especially for which the concept drift occurs rapidly.
- Detecting concept changes in new data, and if these changes are *sufficiently* significant then rebuilding the classifier.
- Adopting an incremental learning algorithm for the classification model.

Our work will focus on a hybrid aproach which is combination of online learning approach and sliding windows method with forgetting. The online learning relates to the family of algorithms that continuously update the classifier parameters, while processing the incoming data.Online learners have to meet some basic requirements [4]:

- Each learning example must be processed only once in the course of training.
- The system should consume only limited memory and processing time, irrespective of the execution time and amount of data processed.

– The training process can be paused at any time, and its accuracy should not be lower than that of a classifier trained on batch data collected up to the given time.

We should also mention algorithms that incorporate the forgetting mechanism. This approach is based on the assumption that the recently arrived data are the most relevant, because they contain characteristics of the current context. However, their relevance diminishes with the passage of time. Therefore, narrowing the range of data to those that were most recently read may help form a dataset that embodies the actual context. There are three possible strategies here:

– selecting the instances by means of a sliding window that cuts off older instances [13];
– weighting the data according to their relevance; and
– applying boosting-based algorithms that focus on misclassified instances [3].

When dealing with the sliding window the main question is how to adjust the window size. On the one hand, a shorter window allows focusing on the emerging context, though data may not be representative for a longer lasting context. On the other hand, a wider window may result in mixing the instances representing different contexts [10].

Therefore, certain advanced algorithms adjust the window size dynamically depending on the detected state (e.g., FLORA2 [13] and ADWIN2 [1]). In more sophisticated algorithms, multiple windows may even be used [11].

2 Active Learning Classifier for Data Stream

Let us propose the classifier learning framework which employs the active learning paradigm. This framework works as the block classifier, because it collects the data in the form of chunk, but for each chunk the online learner is used. The decision about the object labelling depends on two parameters:

– threshold - which is responsible for choosing the "interesting" examples, i.e., if support function related with the decision is lower than a given threshold the the object seems to be interesting and algorithm is asking for its label.
– The label will be assigned (i.e., algorithm will pay for it) only in the case if its budget related to a given chunk will allows to pay for it. For each chunk only limited percentage of the objects could be labelled (depends on parameter budget).

Because the data stream is collected in the form of chunk, but each of them is processing incrementally, therefore before its analysis the order of the collected objects is randomizing. The idea of the algorithm is presented in Algorithm 1.

Algorithm 1. Active learning classifier for data stream

Require: input data stream,
n - data chunk size,
incremental training procedure(),
classifier(),
budget - max. percent of labeled example in a chunk,
treshold

1: $i \leftarrow 0$
2: initialize classifier
3: **repeat**
4: collect new data chunk $DS_i = \{x_i^1, x_i^2, ..., x_i^n\}$
5: set random order of collected examples in DS_i
6: **for** $j = 0$ **to** n **do**
7: support $\leftarrow$ *max* (support function value returned by the classifier for x_i^j)
8: **if** *support* $<$ *treshold* **then**
9: **if** *budget* > 0 **then**
10: ask for the label of the jth example
11: classifier $\leftarrow$ incremental training procedure(x_j)
12: *budget* $\leftarrow$ *budget* $- 1$
13: **end if**
14: **end if**
15: **end for**
16: $i \leftarrow i + 1$
17: **until** end of the input data stream

3 Experiments

3.1 Goals

The goal of the experiment is to check dependencies between algorithm's parameters (chunk size, threshold and budged for labelling) and the accuracy and stability of the classifier. The experiments were carried out for the 3 artificially generated data streams available in MOA (WaveformGeneratorDrift, RandomTreeGenerator, LEDGeneratorDrift) and 3 online learners as minimal distance classifier (k-NN), Naïve Bayes and Perceptron.

Software Environment. All experiments were carried out in the programing language Java using MOA (Massive Online Analysis) experimental software environment[1]. MOA [2] is a very popular framework for data stream analysis. It includes tools for evaluation and a collection of machine learning algorithms. The MOA project is related to WEKA software [8] developed by the same team from University of Waikato, New Zealand.

[1] http://moa.cms.waikato.ac.nz/.

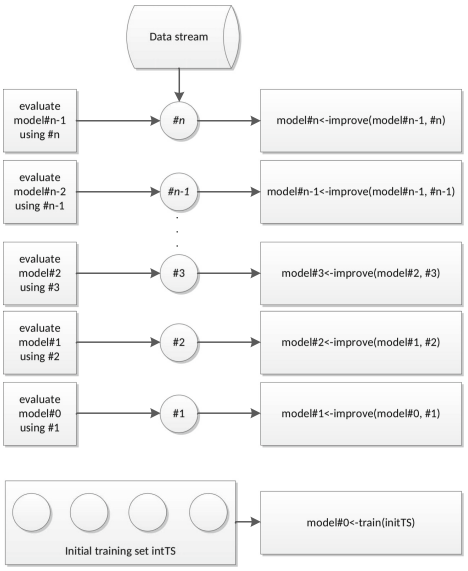

Fig. 1. Idea of test and train evaluation.

Table 1. Result of the experiments for k-NN classifier and LEDGeneratorDrift stream.

ch-s	tresh.	bud.	acc.	sd	ch-s	tresh.	bud.	acc.	sd
10	0.25	0.125	38.74	6.36	100	0.5	0.5	59.94	2.57
10	0.25	0.25	38.28	4.92	100	0.5	0.75	58.43	2.34
10	0.25	0.5	42.73	4.65	100	0.75	0.125	62.93	4.60
10	0.25	0.75	32.78	2.96	100	0.75	0.25	63.75	3.32
10	0.5	0.125	52.63	9.78	100	0.75	0.5	64.11	2.88
10	0.5	0.25	54.42	8.02	100	0.75	0.75	64.02	2.44
10	0.5	0.5	58.51	6.45	1000	0.25	0.125	38.43	2.27
10	0.5	0.75	58.93	6.57	1000	0.25	0.25	35.31	2.37
10	0.75	0.125	57.65	11.95	1000	0.25	0.5	41.77	2.09
10	0.75	0.25	60.01	9.29	1000	0.25	0.75	35.25	2.33
10	0.75	0.5	61.81	7.71	1000	0.5	0.125	56.65	1.92
10	0.75	0.75	62.26	7.20	1000	0.5	0.25	56.63	1.66
100	0.25	0.125	35.49	2.93	1000	0.5	0.5	60.94	1.75
100	0.25	0.25	38.5	2.99	1000	0.5	0.75	60.99	1.45
100	0.25	0.5	33.83	2.84	1000	0.75	0.125	64.82	1.38
100	0.25	0.75	32.71	3.43	1000	0.75	0.25	64.37	1.35
100	0.5	0.125	55.74	3.52	1000	0.75	0.5	63.87	1.58
100	0.5	0.25	55.45	2.79	1000	0.75	0.75	64.18	1.40

Table 2. Result of the experiments for k-NN classifier and RandomTreeGenerator stream.

ch-s	tresh.	bud.	acc.	sd	ch-s	tresh.	bud.	acc.	sd
10	0.25	0.125	58.54	3.36	100	0.5	0.5	57.93	3.17
10	0.25	0.25	58.54	3.36	100	0.5	0.75	60.75	3.15
10	0.25	0.5	41.45	3.44	100	0.75	0.125	75.44	2.49
10	0.25	0.75	58.54	3.36	100	0.75	0.25	75.63	2.59
10	0.5	0.125	58.54	3.36	100	0.75	0.5	76.04	2.26
10	0.5	0.25	59.31	3.26	100	0.75	0.75	75.72	2.32
10	0.5	0.5	60.94	3.00	1000	0.25	0.125	57.69	3.12
10	0.5	0.75	58.54	3.36	1000	0.25	0.25	42.31	3.12
10	0.75	0.125	73.40	4.78	1000	0.25	0.5	57.69	3.12
10	0.75	0.25	74.76	4.03	1000	0.25	0.75	57.69	3.12
10	0.75	0.5	74.79	3.97	1000	0.5	0.125	59.28	2.57
10	0.75	0.75	75.10	3.84	1000	0.5	0.25	64.16	2.38
100	0.25	0.125	57.93	3.17	1000	0.5	0.5	61.11	2.47
100	0.25	0.25	57.60	3.14	1000	0.5	0.75	57.69	3.12
100	0.25	0.5	57.93	3.17	1000	0.75	0.125	76.06	1.76
100	0.25	0.75	42.07	3.17	1000	0.75	0.25	75.96	1.64
100	0.5	0.125	57.93	3.17	1000	0.75	0.5	76.28	1.33
100	0.5	0.25	57.93	3.17	1000	0.75	0.75	75.79	1.86

Error Evaluation. The new model is trained on a given data chunk, but the error is estimated on the basis on the next (unseen) portion of data. Such a method is called "test and train" or "block evaluation method" [2] and its idea is depicted in Fig. 1.

3.2 Results

The results of the experiments are presented in Tables 1, 2, 3, 4, 5, 6, 7, 8 and 9. The abbreviations used in the column description: ch-s - chunk size, tresh.-treshold, bud. - budget, acc.-accuracy, sd - standard deviation.

3.3 Discussion

We realize that the scope of the experiments we carried out is limited and derived remarks are limited to the tested methods and three data stream generators only. In this case formulating general conclusions is very risky, but the preliminary results are quite promising, therefore we would like to continue the work on active learning methods applied to online learning in the future. Let's focus on some interesting observations:

Table 3. Result of the experiments for k-NN classifier and WaveformGeneratorDrift stream.

ch-s	tresh.	bud.	acc.	sd	ch-s	tresh.	bud.	acc.	sd
10	0.25	0.125	48.44	4.79	100	0.5	0.5	65.01	2.94
10	0.25	0.25	33.23	3.17	100	0.5	0.75	71.12	2.65
10	0.25	0.5	33.22	3.18	100	0.75	0.125	79.50	2.34
10	0.25	0.75	65.71	3.26	100	0.75	0.25	81.87	2.23
10	0.5	0.125	65.16	5.25	100	0.75	0.5	80.49	2.04
10	0.5	0.25	74.95	5.26	100	0.75	0.75	81.02	2.43
10	0.5	0.5	75.54	3.80	1000	0.25	0.125	57.70	2.76
10	0.5	0.75	69.54	3.27	1000	0.25	0.25	33.25	3.12
10	0.75	0.125	77.24	8.55	1000	0.25	0.5	74.18	2.46
10	0.75	0.25	77.85	6.83	1000	0.25	0.75	33.43	2.96
10	0.75	0.5	80.33	3.48	1000	0.5	0.125	72.09	2.35
10	0.75	0.75	80.96	3.56	1000	0.5	0.25	75.98	2.12
100	0.25	0.125	50.40	3.23	1000	0.5	0.5	73.13	2.24
100	0.25	0.25	61.23	3.17	1000	0.5	0.75	74.95	2.16
100	0.25	0.5	33.40	2.94	1000	0.75	0.125	80.57	1.79
100	0.25	0.75	33.40	2.94	1000	0.75	0.25	80.39	2.06
100	0.5	0.125	61.28	3.19	1000	0.75	0.5	81.48	1.56
100	0.5	0.25	72.38	2.91	1000	0.75	0.75	81.51	1.50

Table 4. Result of the experiments for Naive Bayes classifier and LEDGeneratorDrift stream.

ch-s	tresh.	bud.	acc.	sd	ch-s	tresh.	bud.	acc.	sd
10	0.25	0.125	35.65	4.05	100	0.5	0.5	62.13	3.16
10	0.25	0.25	42.38	4.05	100	0.5	0.75	66.70	3.08
10	0.25	0.5	35.14	5.18	100	0.75	0.125	73.61	3.75
10	0.25	0.75	36.79	3.08	100	0.75	0.25	73.81	3.39
10	0.5	0.125	58.04	10.34	100	0.75	0.5	73.65	2.94
10	0.5	0.25	61.97	8.14	100	0.75	0.75	74.05	2.87
10	0.5	0.5	62.46	5.37	1000	0.25	0.125	48.26	3.26
10	0.5	0.75	65.84	4.60	1000	0.25	0.25	50.70	3.32
10	0.75	0.125	68.87	12.05	1000	0.25	0.5	38.87	3.02
10	0.75	0.25	70.65	10.10	1000	0.25	0.75	45.90	3.14
10	0.75	0.5	72.20	6.70	1000	0.5	0.125	61.81	3.11

(*Continued*)

Table 4. (*Continued)*

ch-s	tresh.	bud.	acc.	sd	ch-s	tresh.	bud.	acc.	sd
10	0.75	0.75	72.28	7.03	1000	0.5	0.25	61.45	2.97
100	0.25	0.125	39.83	3.40	1000	0.5	0.5	68.35	3.14
100	0.25	0.25	42.64	11.25	1000	0.5	0.75	67.34	3.05
100	0.25	0.5	38.99	3.20	1000	0.75	0.125	73.85	2.85
100	0.25	0.75	32.5	2.97	1000	0.75	0.25	74.04	2.76
100	0.5	0.125	61.04	3.55	1000	0.75	0.5	73.95	2.83
100	0.5	0.25	67.01	3.26	1000	0.75	0.75	73.89	2.78

Table 5. Result of the experiments for k-NN classifier and RandomTreeGenerator stream.

ch-s	tresh.	bud.	acc.	sd	ch-s	tresh.	bud.	acc.	sd
10	0.25	0.125	41.47	3.36	100	0.5	0.5	57.93	3.17
10	0.25	0.25	56.72	3.15	100	0.5	0.75	61.36	6.07
10	0.25	0.5	58.49	3.28	100	0.75	0.125	73.57	3.00
10	0.25	0.75	41.47	3.36	100	0.75	0.25	73.68	2.92
10	0.5	0.125	58.36	3.21	100	0.75	0.5	73.54	2.92
10	0.5	0.25	58.48	3.32	100	0.75	0.75	73.58	2.96
10	0.5	0.5	60.60	3.07	1000	0.25	0.125	42.31	3.12
10	0.5	0.75	57.85	4.01	1000	0.25	0.25	42.31	3.13
10	0.75	0.125	72.63	3.84	1000	0.25	0.5	42.31	3.12
10	0.75	0.25	72.59	4.29	1000	0.25	0.75	57.69	3.12
10	0.75	0.5	72.83	3.67	1000	0.5	0.125	65.79	3.07
10	0.75	0.75	73.08	3.30	1000	0.5	0.25	59.30	3.14
100	0.25	0.125	42.07	3.17	1000	0.5	0.5	60.10	4.30
100	0.25	0.25	42.07	3.17	1000	0.5	0.75	57.48	3.20
100	0.25	0.5	57.93	3.17	1000	0.75	0.125	73.67	2.88
100	0.25	0.75	42.7	3.17	1000	0.75	0.25	73.64	2.79
100	0.5	0.125	58.25	3.24	1000	0.75	0.5	73.58	2.755818
100	0.5	0.25	57.31	3.17	1000	0.75	0.75	73.54	2.877642

– The experiments confirmed that proposed approach can adapt to changing concept returning a quite stable classifier, especially for a quite big data chunk.
– It's a strong dependency between threshold and the classifier accuracy. For the quite big data chunk (100 and 1000) the high threshold (0.75) guarantees the stable classifier. Usually, for the mentioned above treshold value the stability and accuracy do not depend on the budget.

Table 6. Result of the experiments for Naive Bayes classifier and WaveformGeneratorDrift stream.

ch-s	tresh.	bud.	acc.	sd	ch-s	tresh.	bud.	acc.	sd
10	0.25	0.125	33.49	3.04	100	0.5	0.5	72.45	2.82
10	0.25	0.25	42.90	3.13	100	0.5	0.75	74.00	2.84
10	0.25	0.5	43.53	2.93	100	0.75	0.125	80.29	2.70
10	0.25	0.75	33.49	3.044	100	0.75	0.25	80.04	2.67
10	0.5	0.125	70.78	5.19	100	0.75	0.5	81.30	2.56
10	0.5	0.25	70.28	4.29	100	0.75	0.75	81.54	2.57
10	0.5	0.5	66.31	2.95	1000	0.25	0.125	60.06	3.09
10	0.5	0.75	70.60	2.78	1000	0.25	0.25	58.47	3.18
10	0.75	0.125	79.48	6.44	1000	0.25	0.5	33.25	3.12
10	0.75	0.25	78.72	5.93	1000	0.25	0.75	33.25	3.12
10	0.75	0.5	80.83	3.08	1000	0.5	0.125	74.38	2.56
10	0.75	0.75	80.70	3.11	1000	0.5	0.25	69.73	2.869869
100	0.25	0.125	63.30	3.15	1000	0.5	0.5	61.24	2.96
100	0.25	0.25	75.04	2.87	1000	0.5	0.75	71.30	2.68
100	0.25	0.5	33.70	3.09	1000	0.75	0.125	80.25	2.34
100	0.25	0.75	46.13	3.23	1000	0.75	0.25	80.39	2.45
100	0.5	0.125	58.61	3.06	1000	0.75	0.5	80.89	2.28
100	0.5	0.25	62.32	3.29	1000	0.75	0.75	81.00	2.31

Table 7. Result of the experiments for Perceptron classifier and LEDGeneratorDrift stream.

ch-s	tresh.	bud.	acc.	sd	ch-s	tresh.	bud.	acc.	sd
10	0.25	0.125	36.57	7.36	100	0.5	0.5	58.35	3.95
10	0.25	0.25	38.14	5.92	100	0.5	0.75	66.00	3.46
10	0.25	0.5	44.07	5.70	100	0.75	0.125	71.18	7.61
10	0.25	0.75	39.44	4.48	100	0.75	0.25	72.19	5.70
10	0.5	0.125	51.30	11.62	100	0.75	0.5	72.84	4.53
10	0.5	0.25	54.56	10.44	100	0.75	0.75	73.27	4.10
10	0.5	0.5	61.42	10.45	1000	0.25	0.125	32.30	2.99
10	0.5	0.75	56.57	7.44	1000	0.25	0.25	33.46	3.61
10	0.75	0.125	62.14	15.26	1000	0.25	0.5	47.59	3.31
10	0.75	0.25	64.75	13.22	1000	0.25	0.75	32.83	3.19
10	0.75	0.5	67.84	11.61	1000	0.5	0.125	59.29	3.84

(*Continued*)

Table 7. (*Continued)*

ch-s	tresh.	bud.	acc.	sd	ch-s	tresh.	bud.	acc.	sd
10	0.75	0.75	68.38	11.06	1000	0.5	0.25	60.032	3.21
100	0.25	0.125	39.28	3.32	1000	0.5	0.5	64.45	3.23
100	0.25	0.25	37.73	3.12	1000	0.5	0.75	66.09	3.02
100	0.25	0.5	39.75	3.11	1000	0.75	0.125	73.51	3.02
100	0.25	0.75	42.17	3.19	1000	0.75	0.25	73.66	2.77
100	0.5	0.125	60.75	5.40	1000	0.75	0.5	73.66	2.62
100	0.5	0.25	60.23	4.24	1000	0.75	0.75	73.67	2.58

Table 8. Result of the experiments for Perceptron classifier and RandomTreeGenerator stream.

ch-s	tresh.	bud.	acc.	sd	ch-s	tresh.	bud.	acc.	sd
10	0.25	0.125	58.54	3.36	100	0.5	0.5	57.93	3.17
10	0.25	0.25	41.47	3.36	100	0.5	0.75	57.90	3.18
10	0.25	0.5	41.47	3.36	100	0.75	0.125	61.38	4.38
10	0.25	0.75	41.47	3.36	100	0.75	0.25	63.28	3.97
10	0.5	0.125	58.54	3.36	100	0.75	0.5	64.45	3.30
10	0.5	0.25	58.54	3.36	100	0.75	0.75	64.10	3.45
10	0.5	0.5	57.85	3.40	1000	0.25	0.125	57.69	3.12
10	0.5	0.75	58.54	3.36	1000	0.25	0.25	42.31	3.13
10	0.75	0.125	60.53	3.94	1000	0.25	0.5	42.31	3.12
10	0.75	0.25	62.08	4.72	1000	0.25	0.75	57.69	3.12
10	0.75	0.5	63.30	3.66	1000	0.5	0.125	57.54	3.11
10	0.75	0.75	63.52	3.76	1000	0.5	0.25	57.74	3.11
100	0.25	0.125	42.07	3.17	1000	0.5	0.5	58.10	3.06
100	0.25	0.25	57.93	3.17	1000	0.5	0.75	58.15	3.13
100	0.25	0.5	57.93	3.17	1000	0.75	0.125	59.17	3.78
100	0.25	0.75	42.07	3.17	1000	0.75	0.25	64.35	3.26
100	0.5	0.125	57.93	3.17	1000	0.75	0.5	64.66	3.03
100	0.5	0.25	57.64	3.178	1000	0.75	0.75	64.85	2.94

- For small chunk size the budget plays an important role. Its increasing causes that the model achieves better quality.
- It is obvious, but confirmed by the experiments that the accuracy and stability depend on chunk size, but for the biggest chunk size (1000) we observed the quality and stability detoration, what is probably causes by the fact that the chunks cover more than one model.

Table 9. Result of the experiments for Perceptron classifier and WaveformGenerator-Drift stream.

ch-s	tresh.	bud.	acc.	sd	ch-s	tresh.	bud.	acc.	sd
10	0.25	0.125	50.30	3.24	100	0.5	0.5	59.44	3.10
10	0.25	0.25	51.01	3.19	100	0.5	0.75	70.3	2.96
10	0.25	0.5	33.49	3.044	100	0.75	0.125	74.87	8.53
10	0.25	0.75	33.02	2.98	100	0.75	0.25	78.27	7.93
10	0.5	0.125	62.78	3.32	100	0.75	0.5	80.94	4.52
10	0.5	0.25	75.98	8.22	100	0.75	0.75	80.54	4.08
10	0.5	0.5	67.77	8.61	1000	0.25	0.125	33.39	2.82
10	0.5	0.75	53.10	5.31	1000	0.25	0.25	33.10	3.04
10	0.75	0.125	66.65	10.46	1000	0.25	0.5	33.10	3.04
10	0.75	0.25	69.94	12.26	1000	0.25	0.75	33.34	3.00
10	0.75	0.5	72.17	11.19	1000	0.5	0.125	62.25	3.22
10	0.75	0.75	76.28	7.41	1000	0.5	0.25	69.90	3.05
100	0.25	0.125	62.64	3.10	1000	0.5	0.5	60.36	3.22
100	0.25	0.25	33.40	2.94	1000	0.5	0.75	58.41	3.84
100	0.25	0.5	33.31	2.95	1000	0.75	0.125	81.15	3.15
100	0.25	0.75	61.79	3.16	1000	0.75	0.25	82.01	2.62
100	0.5	0.125	62.74	3.22	1000	0.75	0.5	80.64	3.69
100	0.5	0.25	61.89	5.07	1000	0.75	0.75	77.73	6.99

4 Conclusions

We realize that the scope of the experiments we carried out is limited. In this case formulating general conclusions is very risky, but the preliminary results are quite promising, therefore we would like to continue the work on active learning approach in the future.

In the near future we are going to:

- Carrying out experiments on the wider number of datasets, especially for the streams with and without concept drift.
- Applying the proposed approach to the classifier ensemble.
- Evaluating the proposed algorithm behavior for more type of concept drift and maybe employ concept drift detector to establish the chunk size dynamically.

Acknowledgement. This work was supported by the Polish National Science Centre under the grant no. DEC-2013/09/B/ST6/02264. This work was supported by EC under FP7, Coordination and Support Action, Grant Agreement Number 316097, ENGINE European Research Centre of Network Intelligence for Innovation Enhancement (http://engine.pwr.wroc.pl/). All computer experiments were carried out using computer equipment sponsored by ENGINE project.

References

1. Bifet, A., Gavalda R.: Learning from time-changing data with adaptive windowing. In: Proceedings of the Seventh SIAM International Conference on Data Mining. SIAM, Minneapolis, Minnesota, USA, 26–28 April 2007
2. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: Moa: massive online analysis. J. Mach. Learn. Res. **11**, 1601–1604 (2010)
3. Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., Gavaldà, R.: New ensemble methods for evolving data streams. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2009, pp. 139–148. ACM, New York (2009)
4. Domingos, P., Hulten, G.: A general framework for mining massive data streams. J. Comput. Graph. Stat. **12**, 945–949 (2003)
5. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. Wiley, New York (2001)
6. Gama, J., Zliobaite, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM Comput. Surv. (2013, in press)
7. Greiner, R., Grove, A.J., Roth, D.: Learning cost-sensitive active classifiers. Artif. Intell. **139**(2), 137–174 (2002)
8. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: an update. SIGKDD Explor. Newsl. **11**(1), 10–18 (2009)
9. Kuncheva, L.I.: Combining Pattern Classifiers: Methods and Algorithms. Wiley, New York (2004)
10. Kurlej, B., Wozniak, M.: Impact of window size in active learning of evolving data streams. In: Proceedings of the 45th International Conference on Modelling and Simulation of Systems, MOSIS 2011, pp. 56–62 (2011)
11. Lazarescu, M.M., Venkatesh, S., Bui, H.H.: Using multiple windows to track concept drift. Intell. Data Anal. **8**(1), 29–59 (2004)
12. Widmer, G., Kubat, M.: Effective learning in dynamic environments by explicit context tracking. In: Brazdil, P.B. (ed.) ECML 1993. LNCS, vol. 667, pp. 227–243. Springer, Heidelberg (1993)
13. Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. Mach. Learn. **23**(1), 69–101 (1996)