

Accuracy Updated Ensemble for Data Streams with Concept Drift

Dariusz Brzeziński and Jerzy Stefanowski

Institute of Computing Science, Poznań University of Technology,
ul. Piotrowo 2, 60–965 Poznań, Poland
{dariusz.brzezinski, jerzy.stefanowski}@cs.put.poznan.pl

Abstract. In this paper we study the problem of constructing accurate block-based ensemble classifiers from time evolving data streams. AWE is the best-known representative of these ensembles. We propose a new algorithm called Accuracy Updated Ensemble (AUE), which extends AWE by using online component classifiers and updating them according to the current distribution. Additional modifications of weighting functions solve problems with undesired classifier excluding seen in AWE. Experiments with several evolving data sets show that, while still requiring constant processing time and memory, AUE is more accurate than AWE.

1 Introduction

Ensembles have attracted many researchers as an approach for improving predictive accuracy. However, most of the research is devoted to static environments where the classification task is fixed and complete data is available for learning classifiers. On the other hand, a new type of problems is becoming more visible, one in which learning algorithms work in *dynamic environments* with data continuously generated in the form of a stream [1]. Processing *data streams* implies new requirements concerning limited amount of memory, small processing time, and one scan of incoming data. Moreover, the data distributions and definitions of target classes change over time. These changes are categorized into *sudden* or *gradual concept drift* depending on appearance of novel classes in a stream and the rate of changing definitions of classes [2]. Concept drifts directly influence algorithm classification abilities as classifiers generated prior to change have been trained on a different class distribution. As the reason of these changes is hidden and not known a priori, the task of learning classifiers becomes very difficult.

A classifier (individual or ensemble), if intended for such *non-stationary environments*, has to adapt to concept drifts. Several adaptation methods have been proposed including mainly: sliding window approaches, new online algorithms, special detection techniques, and adaptive ensembles. In the area of *adaptive ensembles*, component classifiers are generated from sequential blocks of training examples. When a new block arrives, classifiers are evaluated and later updated, removed, or modified according to the result of the evaluation. *Accuracy*

Weighted Ensemble (AWE) is the most popular method in this area [3]. However, defining an appropriate size of the data block can be problematic. Moreover, too many component classifiers can be excluded from the ensemble when they are not accurate enough. We have also noticed in preliminary experiments that AWE is not as accurate as other online classifiers [4].

Therefore, we decided to propose a new algorithm, called *Accuracy Updated Ensemble* (AUE), which would improve over AWE on classification accuracy, while still keeping good computational efficiency. The other aim of this paper is to evaluate the proposed algorithm on several changing data sets and compare it with AWE and other related ensembles available in the MOA framework [5].

2 Related Work

Similarly to most researchers we consider completely supervised learning, i.e. class labels of incoming examples in the stream are available and can be used for evaluating and updating a classifier. Below, we briefly discuss methods most related to our proposal and those used in experiments. For more comprehensive reviews, in particular concerning online incremental ensembles consult [1, 2]. Unlike ensembles that can be modified after reading single examples, we discuss streams divided into *blocks of examples* (non-overlapping and of the same size). In an adaptive ensemble, each component is learned using typical “batch” mode on the most recent block (also called a *chunk*) of data. Following discussion from [1], in time-changing streams, data is generated from a mixed distribution, which can be seen as a weighted combination of distributions characterizing the target concepts. This justifies multiple classifiers where each component classifier receives a *weight* reflecting its performance on the most recent block of data.

The SEA algorithm [6] was one of the first algorithms following this idea of building separate classifiers from sequential blocks. Component classification scores are evaluated on the newest block and the weakest classifier in the fixed size ensemble can be replaced by a newly trained one.

Another way of restructuring an ensemble was proposed by Wang et al. [3]. In their *Accuracy Weighted Ensemble* (AWE), the authors propose to train a new classifier on each incoming data block and use that block to evaluate all the existing classifiers in the ensemble. They formally proved that if component classifiers are weighted by their expected accuracy on the test data, the ensemble improves classification accuracy over a single classifier. To weight the members of an ensemble we need to know the actual function being learned, which is unavailable. That is why Wang et al. proposed to derive weights by estimating the error rate on the most recent data block x_i , as shown in Equations 1-2:

$$MSE_i = \frac{1}{|x_i|} \sum_{(x,c) \in x_i} (1 - f_c^i(x))^2, \quad MSE_r = \sum_c p(c)(1 - p(c))^2, \quad (1)$$

$$w_i = MSE_r - MSE_i, \quad (2)$$

where $f_c^j(x)$ denotes the probability given by classifier C_i that x is assigned to class c . The value of MSE_r is the mean square error of a randomly predicting

classifier and it is used as a threshold to zero the weights of classifiers, which are not accurate enough. For the first k data chunks AWE takes a set of all available classifiers, but when processing further chunks it selects only the k best components to form an ensemble. For a large data stream it is impossible to remember all the components and the number of stored components should be limited. The predictions of components are aggregated by a weighted voting rule. Experiments comparing AWE with different single classifiers, such as C4.5, RIPPER, and Naive Bayes showed that AWE improved both computational efficiency in learning and classification accuracy.

The performance of AWE and other block-built ensembles largely depends on the size of the data blocks. Bigger blocks can lead to more accurate classifiers, but can contain more than one concept drift. On the other hand, smaller blocks are better at separating changes, but usually lead to poorer classifiers.

In our approach we learn classifiers with *Very Fast Decision Trees* (also known as *Hoeffding Trees*). They were introduced in [7] to efficiently learn from massive data in an incremental way without the need for storing consecutive examples. The tree is learned by recursively replacing leaves with decision nodes. Each leaf stores sufficient statistics about attribute values, which are needed by an evaluation function that judges the merit of split-tests based on attribute values. The key idea is to show that a relatively small sample can be enough to choose the optimal split-test, based on collected statistics. The main innovation is the use of the *Hoeffding bound* to guarantee that the selected split is really the best one. The original VFDT algorithm was proposed for static data streams. However different adaptations to handle concept drift were later proposed; see their review in [1].

VFDT are also used in an ensemble called *Option Trees*. This generalization of a single tree includes *option nodes* where instead of selecting only the best split-test attribute, all promising attributes are kept. Later, for each of those attributes a decision subtree is constructed. Thus, making a final decision with an option tree involves weighted combining of the predictions of all applicable subtrees. In our experiments we use Kirkby’s [8] proposal of Hoeffding Option Trees (HOT) for streaming data (see section 10.4 in [1] for details).

3 Accuracy Updated Ensemble

Following the critical discussion of AWE in Section 1, we propose a new adaptive ensemble called *Accuracy Updated Ensemble* (AUE). The proposed algorithm is inspired by AWE and its weighting mechanism, but improves its flaws. AUE not only selects classifiers, but also *updates* them according to the current distribution. Let us remind that processing of data blocks allowed AWE to learn component classifiers by “traditional batch” algorithms (not special online ones) and later only adjust component weights according to the current distribution. However, this leads to problems with tuning the block size (in [3] it was estimated in many trails). In AUE we turn to online learning of component classifiers. This allows to update base classifiers rather than only adjust their weights. If no change

occurs between a series of blocks, the component classifier will improve just as if it was built on a bigger block (more appropriate for periods of stability). As a result, we can reduce the size of the block without the risk of creating less accurate components. Furthermore, we preserve the basics of AWE’s weighting mechanism and depreciate classifiers if sudden drift occurs. The combination of classifier selection and updating should make AUE better than AWE in times of stability or gradual drift, while being at least as accurate for sudden drift.

Another drawback of AWE is its weighting function. Because the algorithm is designed to perform well on cost-sensitive data [3], the MSE_r threshold in Equation 2 cuts-off “risky” classifiers. In rapidly changing environments with sudden concept drifts (as the Electricity data set) this threshold can “mute” all ensemble members causing no class to be predicted. To avoid this, in AUE we propose a simpler weighting function:

$$w_i = \frac{1}{(MSE_i + \epsilon)} \quad (3)$$

MSE_i is calculated just like in Equation 1 and ϵ is a very small constant value, which allows weight calculation in rare situations when $MSE_i = 0$.

We want to update component classifiers according to the current distribution, while still keeping their diversity. To achieve this, we update only selected classifiers. First of all, we consider only current ensemble members - the k top weighted classifiers. Then we use MSE_r as a threshold for allowing online updating of only “accurate enough” classifiers (line 12 of AUE psuedo-code). Therefore, inaccurate classifiers can enter the ensemble, but will not be updated. The full pseudo-code of the Accuracy Updated Ensemble is listed in Algorithm 1.

Algorithm 1 Accuracy Updated Ensemble

Input: \mathcal{S} : data stream of examples

k : number of ensemble members

Output: \mathcal{E} : ensemble of k online classifiers with updated weights

```

1:  $\mathcal{C} \leftarrow \emptyset$ ; //  $\mathcal{C}$ : set of stored classifiers
2: for all data chunks  $x_i \in \mathcal{S}$  do
3:   train classifier  $C'$  on  $x_i$ ;
4:   compute error  $MSE$  of  $C'$  via cross validation on  $x_i$ ;
5:   derive weight  $w'$  for  $C'$  using (3);
6:   for all classifiers  $C_i \in \mathcal{C}$  do
7:     apply  $C_i$  on  $x_i$  to derive  $MSE_i$ ;
8:     compute weight  $w_i$  based on (3);
9:    $\mathcal{E} \leftarrow k$  of the top weighted classifiers in  $\mathcal{C} \cup \{C'\}$ ;
10:   $\mathcal{C} \leftarrow \mathcal{C} \cup \{C'\}$ ;
11:  for all classifiers  $C_e \in \mathcal{E}$  do
12:    if  $w_e > \frac{1}{MSE_r}$  and  $C_e \neq C'$  then update classifier  $C_e$  with  $x_i$ ;

```

To sum up, AUE differs from AWE in the definition of the weight function, the use of online base classifiers, and updating components with incoming examples. Ensemble members are weighted, can be removed, and are not always updated, unlike in online bagging [1]. Compared to VFDT based ensembles, ASHT and HOT, we do not limit base classifier size, do not use any windows, and update members only if they are accurate enough according to the current distribution.

4 Experimental Evaluation

The aim of the experiments was to compare the newly proposed AUE with AWE and two other stream classifiers: the Hoeffding Option Tree (HOT) and a single Hoeffding Tree with a static window (HT+Win). We chose AWE as it is the classifier we tried to improve, HOT as a different ensemble that uses Hoeffding Trees, and HT+Win as a reference point to using a single classifier. In all the compared algorithms we construct component classifiers with Hoeffding Trees and compare basic characteristics on popular synthetic and real life data sets.

All of the tested algorithms were implemented in Java as part of the MOA framework [5]. We implemented the AWE and AUE algorithms, and a data chunk evaluation procedure, while all the other algorithms were already a part of MOA. The experiments were done on a machine equipped with an Intel Pentium Core 2 Duo P9300 @ 2.26 GHz processor and 3.00 GB of RAM. To make the comparison more meaningful, we set the same parameter values for all the algorithms. For ensemble methods we set the number of component classifiers to 15: AWE and AUE have 15 trees, HOT has 15 options. The size of data block, as suggested in [3], is equal $d_r = 500$ and $d_a = 1000$, for real and artificial data sets, respectively. We also set the static window size to $15 \times blockSize$ to make the number of examples seen by the windowed classifier similar to that seen by AWE and AUE. The parameters of the Hoeffding Tree used with a static window are the same as those of the option tree, and the base classifiers (also Hoeffding Trees) of the ensembles.

According to the main characteristics of data streams [5, 4], we evaluate performance of algorithms with respect to time efficiency, memory usage, and accuracy. Classification accuracy was calculated using the *data block evaluation method*, which works similarly to *test-then-train* paradigm. This method reads incoming examples without processing them, until they form a data block of size d . Each new data block is first used to test the existing classifier, then it updates the classifier. More details concerning this method can be found in [4].

4.1 Data Sets

Following literature on adaptive ensembles we selected three real and four synthetic data sets with concept drift, all of which are publicly available.

Real Data Sets. We selected: Electricity market data (Elec), Ozone level detection (Ozone), and Donation data (Don). Electricity [9] is a data set that consists of energy prices from the electricity market in the Australian state of

New South Wales. Ozone problem concerns local ozone peak prediction, that is based on eight hours measurement [10]. The true model behind the data evolves gradually over time. Another difficulty in mining this data set, is that many of the 72 features collected for each instance are irrelevant. Finally, Donation [11] is a data set used for the 2nd KDDCup. Donation is the largest real data set and contains examples of sudden drift.

Synthetic data sets. For testing algorithms on larger data sets we used four popular generators: LED (Led), Waveform (Wave) [1], Hyperplane (Hyp) [3], and SEA (Sea) [6]. LED consists of a stream of 24 binary attributes, 17 of which are irrelevant, that define the digit displayed on a seven-segment LED display. We generated 1,000,000 examples with sudden and gradual concept drift. Waveform consists of a stream with three decision classes where the instances are described by 40 attributes. With Waveform we generated 1,000,000 instances with gradual concept drift. Hyperplane is used to generate a stream of 5,000,000 examples representing gradual concept drift by rotating the decision boundary for each concept. Finally, we use SEA to generate a data stream of 20,000,000 instances with 10% of noise and sudden concept drift.

4.2 Results

Time analysis. Tables 1 and 2 present average train and test times for each data set. Training time for HOTA is usually the highest, in particular for larger artificial data. Of course the single tree (HT+Win) learns much faster than ensembles. We also analyzed graphical plots of time with respect to the number of processed examples¹, which showed that testing times remain close to constant throughout the whole processing of the data stream. This observation is true for all tested data sets. On the other hand, training time is not constant for all algorithms - HOTA shows clear linear growth of training time when no sudden drift occurs. Hoeffding trees become more complex as they see more examples. In periods of stability HOTA will successively grow bigger trees, thus consuming more memory. The windowed tree, AWE, and AUE are built from a limited number of data blocks, which naturally restricts memory.

Table 1. Average test and train times in ms for data blocks in real data sets.

	Elec		Ozone		Don	
	Train	Test	Train	Test	Train	Test
HOTA	101.11	6.64	62.40	1.00	2168.54	17.19
AWE	56.91	13.29	179.40	27.30	3290.93	1074.98
AUE	75.11	15.31	241.80	54.60	3292.64	1086.82
HT+Win	23.98	0.29	46.80	0.00	81.10	1.76

¹ Due to space limitations, in this paper we mostly present tabular summaries. For a complete set of plots see [4].

Table 2. Average test and train times in ms for data blocks in artificial data sets.

	Led		Wave		Hyp		Sea	
	Train	Test	Train	Test	Train	Test	Train	Test
HOT	563.68	9.93	2573.86	51.27	5170.84	14.18	4876.41	6.44
AWE	751.40	181.85	558.21	159.19	41.13	4.96	29.44	5.61
AUE	803.93	185.39	636.59	162.06	240.22	49.39	90.05	18.84
HT+Win	54.42	10.81	36.42	6.96	17.61	3.40	6.32	1.31

Memory Usage. According to Table 3, HT+Win used the least memory on most data sets. Additionally, the analysis of all memory plots showed that memory requirements are similar to training time requirements. HOT needs much more memory for larger data sets than HT+Win, AWE, and AUE, which processed the data streams using constant memory. An example of linear growth of HOT’s memory requirements is shown in Figure 1.

Table 3. Average size of the classifier for all data sets, measured in MB.

	Elec	Ozone	Don	Led	Wave	Hyp	Sea
HOT	0.41	0.08	13.97	2.76	12.27	18.49	24.45
AWE	0.23	0.28	5.52	0.58	0.33	0.17	0.18
AUE	0.36	0.36	5.64	0.88	0.92	0.86	0.46
HT+Win	0.22	0.30	1.15	0.73	0.42	0.17	0.07

Classification Accuracy. Table 4 presents average classification accuracies obtained by the tested algorithms on all the data sets.

Table 4. Average accuracy for all data sets in percent.

	Elec	Ozone	Don	Led	Wave	Hyp	Sea
HOT	74.37	91.60	94.35	70.68	82.57	85.07	89.81
AWE	71.22	67.59	94.35	71.16	79.63	70.38	78.52
AUE	74.92	76.56	94.35	71.41	82.26	84.72	88.61
HT+Win	42.43	91.60	94.35	60.22	75.46	79.08	87.64

One can notice that none of the tested classifiers is best for all the data sets. For larger data sets, with little and mostly gradual concept drift, HOT gives the best results with AUE being close second. For smaller data sets it is hard to choose a winner. However, in case of sudden changes (Elec, Led) AUE seems to be the most accurate. It is also important to notice that AUE is more accurate than AWE on all data sets (except for Don, where they are equal). The

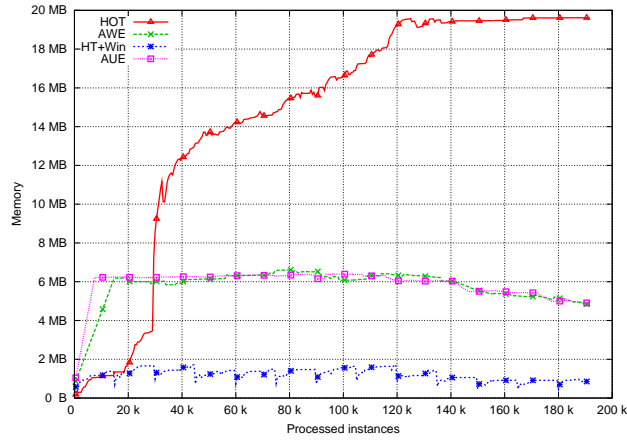


Fig. 1. Growth of HOT’s memory requirements (Donation data set).

difference in accuracy is especially visible for data sets with gradual drift (Hyp, Wave), where AWE falls far below AUE.

The most interesting concept drift was introduced in the generation of the LED data set. We joined two gradually evolving LED data sets with a sudden change. After half million examples we replaced one data source with another. The algorithms’ reactions to this type of change are presented in Figure 2. The plot shows that for this complex concept drift all algorithms have problems with adjusting to change. AUE seems to cope best with this situation. HOT, which performed well before the drift, falls down even below the level of AWE. In periods of stability, HOT grows accurate but complex structures, which are later difficult to rebuild. AWE and AUE are modular, allow quick substitution of components, and therefore quickly react to sudden drifts.

5 Conclusions and Future Work

In this paper we introduced a new stream classifier called Accuracy Updated Ensemble, inspired by an earlier proposed algorithm called Accuracy Weighted Ensemble. AUE was more accurate than AWE on all data sets (except Donation where it was equal) whilst still requiring constant processing time and memory.

Considering the updating technique of AUE we can suspect that in periods of longer distribution stability, when no concept drift occurs, the component classifiers can be trained on more examples and should become more accurate. However, updating many components with similar examples may reduce their diversity. Therefore, in on-going research we study this problem and possible modifications of AUE to ensure additional diversity of ensemble components.

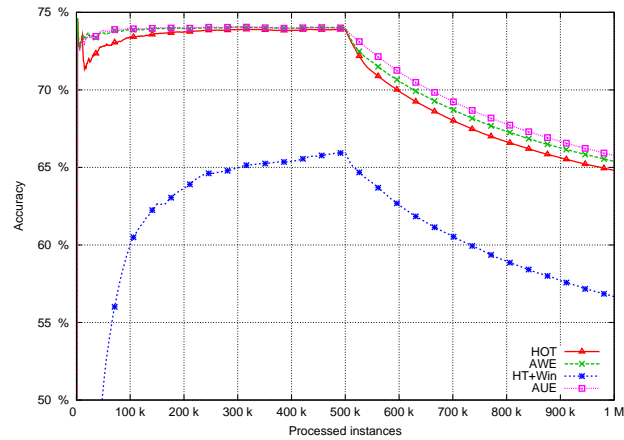


Fig. 2. Accuracy on the Waveform data set.

References

1. Gama, J.: Knowledge Discovery from Data Streams. CRC Press (2010)
2. Kuncheva, L.I.: Classifier ensembles for changing environments. In: Roli, F., Kittler, J., Windeatt, T. (eds.) Multiple Classifier Systems. LNCS, vol. 3077, pp. 1–15. Springer, Heidelberg (2004)
3. Wang, H., Fan, W., Yu, P.S., Han, J.: Mining concept-drifting data streams using ensemble classifiers. In: Getoor, L., Senator, T.E., Domingos, P., Faloutsos, C. (eds.) KDD, pp. 226–235. ACM Press (2003)
4. Brzezinski, D.: Mining data streams with concept drift. Master’s thesis, Poznan University of Technology, Poznan, Poland (2010) Available at: <http://www.cs.put.poznan.pl/dbrzezinski/publications/ConceptDrift.pdf>.
5. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: MOA: Massive online analysis. Journal of Machine Learning Research **11**, pp. 1601–1604 (2010)
6. Street, W.N., Kim, Y.: A streaming ensemble algorithm (SEA) for large-scale classification. In: KDD, pp. 377–382. ACM Press (2001)
7. Domingos, P., Hulten, G.: Mining high-speed data streams. In: KDD, pp. 71–80. (2000)
8. Kirkby, R.: Improving Hoeffding Trees. PhD thesis, Department of Computer Science, University of Waikato (2007)
9. Harries, M.: Splice-2 comparative evaluation: Electricity pricing. Technical report, The University of South Wales (1999)
10. Zhang, K., Fan, W., Yuan, X., Davidson, I., Li, X.: Forecasting skewed biased stochastic ozone days: Analyses and solutions. In: ICDM pp. 753–764. IEEE Computer Society (2006)
11. Fan, W.: Systematic data selection to mine concept-drifting data streams. In Kim, W., Kohavi, R., Gehrke, J., DuMouchel, W. (eds.) KDD, pp. 128–137. ACM Press (2004)