Speeding Up Recovery from Concept Drifts

Silas Garrido Teixeira de Carvalho Santos[1], Paulo Mauricio Gonçalves Júnior[2], Geyson Daniel dos Santos Silva[1], and Roberto Souto Maior de Barros[1]

[1] Centro de Informática, Universidade Federal de Pernambuco, Brazil
{sgtcs,gdss,roberto}@cin.ufpe.br
[2] Instituto Federal de Educação, Ciência e Tecnologia de Pernambuco, Brazil
paulogoncalves@recife.ifpe.edu.br

Abstract. The extraction of knowledge from data streams is an activity that has progressively been receiving an increased demand. However, in this type of environment, changes in data distribution, or concept drift, can occur constantly and is a challenge. This paper proposes the *Adaptable Diversity-based Online Boosting (ADOB)*, a modified version of the online boosting, as proposed by Oza and Russell, which is aimed at speeding up the experts recovery after concept drifts. We performed experiments to compare the accuracy as well as the execution time and memory use of ADOB against a number of other methods using several artificial and real-world datasets, chosen from the most used ones in the area. Results suggest that, in many different situations, the proposed approach maintains a high accuracy, outperforming the other tested methods in regularity, with no significant change in the execution time and memory use. In particular, ADOB was specially efficient in situations where frequent and abrupt concept drifts occur.

Keywords: data stream, concept drift, ensemble classifier, online boosting, diversity.

1 Introduction

Nowadays, several applications need the use of mechanisms that enable the extraction of knowledge in real time. Examples of such applications include monitoring the purchase history of customers, the movement data from sensors, or water temperatures. Thus, the algorithms used for this purpose must be constantly updated, trying to adapt to new instances and taking into account the computational constraints.

When working in environments with a continuous flow of data, it is not possible to guarantee that the distribution of the data will remain stationary. On the contrary, several changes may occur over time, triggering situations commonly known as concept drift. The speed with which the changes occur may be classified as abrupt, when the transition from an old to a new concept occurs suddenly, or gradual, when such a transition is smooth [1].

There are several approaches proposed to detect changes in concepts. Some of these approaches provide for the adaptation of the internal structure of a

T. Calders et al. (Eds.): ECML PKDD 2014, Part III, LNCS 8726, pp. 179–194, 2014.

classifier to deal with the changes of concept. Others try to identify when a concept drift has occurred, and then drop the old classifier and create a new one for the most actual concept [1–6]. Other existing methods use ensemble classifiers with some weighting policy applied to their members as well as for dropping the worst classifiers and adding new ones [7–11]. Finally, there are also approaches that seek to increase the efficiency of change detection when dealing with concepts that have previously occurred [12–14].

In situations where many changes of concepts occur, learning algorithms constantly need to adapt to the new distribution. In such scenarios, it is common to observe a delay before the complete adaptation occurs, i.e., a period that is used for learning a satisfactory generalization. The longer this period, the greater the number of incorrect predictions.

Based on these observations and using an ensemble of classifiers, we decided to try modifying the online version of boosting, originally proposed by Oza and Russell [15], aiming at a more rapid recovery of the experts accuracy in environments with frequent changes of concepts. More specifically, we have changed the way diversity is distributed during training.

This paper describes the Adaptable Diversity-based Online Boosting (ADOB) and compares the performance of our proposal with some of the major existing ensemble methods for dealing with data streams and concept drifts using the Massive Online Analysis (MOA) framework [16]. The results indicate that ADOB maintains good accuracies in different situations, surpassing the other tested methods in regularity.

The rest of this paper is organized as follows: Section 2 surveys the related work; Section 3 introduces the datasets used in the experiments; Section 4 describes in detail the operation of our proposal; Section 5 compares the proposed method to other existing ensemble methods; and, finally, Section 6 presents our conclusions.

2 Related Work

A large number of methods have been proposed to learn from data streams containing concept drifts. Examples of older methods include [12, 17, 18]. Nowadays, several methods have been proposed using the concept of ensemble classifiers.

Bagging [19] and Boosting [20] are techniques that use a set of classifiers trained on the original data by aggregating the responses of each classifier to get a better prediction. They use different strategies both to manipulate the data during the training of experts and also to combine their predictions. Online Bagging and Boosting [15] are adapted versions of these techniques to data stream environments. They both make use of the Poisson distribution to simulate a behavior similar to their offline versions.

Adwin Bagging (ADWINBAG) [8] and Leveraging Bagging (LEVERAGING) [9] both make use of the online version of bagging, as it was defined by Oza and Russell [15], adding Adaptive Windowing (ADWIN) [3] as their concept drift detector. In addition, Leveraging Bagging makes two changes to the original

proposal: the first is to increase the value of diversity (λ) which, as a consequence, leads to an increase in the probability that an expert trains on the same instance. The second is to change the way the experts predict instances in order to increase diversity and reduce the correlation.

The Dynamic Weighted Majority (DWM) [7] extends the Weighted Majority Algorithm [21] and implements a weighted ensemble classifier specifically designed to identify concept drifts. This method adds and removes classifiers according to the ensemble global performance. If the ensemble commits an error, then a classifier is added. If one classifier commits an error, its weight is reduced. If after many examples a classifier continues with a low accuracy, indicated by a low weight, it is removed from the ensemble.

Diversity for Dealing with Drifts (DDD) [10] uses four ensemble classifiers with high and low diversity, before and after a concept drift is detected. A previous study [22] analyzed how these ensembles behaved in data sets suffering from abrupt and gradual concept drifts with several speeds of change, right after the drift and longer after. With the results obtained, Diversity for Dealing with Driftsd was proposed, trying to select the best ensemble (or weighted majority of ensembles) before and after drifts, detected by the use of a drift detection method.

The Accuracy Updated Ensemble (AUE2) [11], recently proposed, maintains a set of classifiers and its strategy is to, at every n instances (called chunks), remove the expert with the worst accuracy and replace it with a new one. The weight of the experts are also defined according to their accuracy, making the most accurate one have a greater influence on the prediction. The way the experts are updated makes the method sensitive to changes in concepts.

3 Datasets

This section describes the datasets that were selected for the experiments used to analyze the performance of ADOB against those of other recent methods. We chose both real-world and artificial datasets. In the artificial ones, it is possible to define the position of the concept drifts as well as its quantity and size. Thus, several situations can be simulated. In the real-world datasets, the unpredictability and volume of data makes their use interesting, complementing the scenarios provided by the artificial data. All the datasets used are available, most of them in the MOA website at the address `http://moa.cs.waikato.ac.nz/`.

3.1 Artificial Datasets

For the experiments described in Section 5, we chose four artificial datasets, two of them with gradual concept drifts and two with abrupt concept drifts. These are: LED [23, 24], RBF [8, 25], Sine [1, 2, 4], and Stagger [7, 17].

The LED dataset is composed of 24 categorical attributes, 17 of which are irrelevant, and one categorical class with ten possible values. It represents the problem of predicting the digit shown by a seven-segment LED display, where each attribute has 10% probability of being inverted (noise). We used a version of LED available at MOA that includes concept drifts to the data sets by simply

changing the attributes positions. This dataset was used in our experiments to test gradual concept drifts.

RBF (Radial Basis Function) creates complex concept drifts that are not straightforward to approximate with a decision tree model. It works as follows: a fixed number of random centroids are generated. Each center has a random position, a single standard deviation, a class label, and a weight. New examples are generated by selecting a center at random, taking weights into consideration so that centers with higher weight are more likely to be chosen. A random direction is chosen to offset the attribute values from the central point. The length of the displacement is randomly drawn from a Gaussian distribution with standard deviation determined by the chosen centroid. The chosen centroid also determines the class label of the example. This effectively creates a normally distributed 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. This dataset is composed of six classes and 20 attributes, and was also used to test gradual concept drifts.

Sine presents the problem of identifying the position of coordinates, represented by two attributes, in relation to the curve $y = sin(x)$. In the first context, points below the curve are classified as positive. After each concept drift, the classification is reversed. Each coordinate has values uniformly distributed in the [0,1] interval. It is possible to include two other attributes, filled with random data in the same interval, with no influence on the classification function (irrelevant data). Gama et al. [1] named these data sets as Sine1 and Sinirrel1, respectively. They also described Sine2, similar to Sine1 but using a different curve: $y < 0.5 + 0.3sin(3\pi x)$. Positive and negative examples are interchanged to ensure a stable learning environment. This dataset was used to test abrupt concept changes and is available at `https://sites.google.com/site/moaextensions/`.

In Stagger, each example consists of the following attributes: $color \in \{green, blue, red\}$, $shape \in \{triangle, circle, rectangle\}$, and $size \in \{small, medium, large\}$. According to the Stagger original paper [17], there are three kinds of different concepts: in concept 1, $color = red \wedge size = small$; in concept 2, $color = green \vee shape = circle$; and in concept 3, $size = medium \vee size = large$. This data set is usually used to simulate abrupt concept drifts and is fairly simple to learn – it has few attributes and concepts, and concepts 2 and 3 overlap.

3.2 Real Datasets

In addition to the artificial datasets, we chose three real-world datasets, from the most used ones in the area of data streams, all with very different number of instances and complexity. In these datasets, the number and position of concept drifts (if existent) are unknown.

The Electricity dataset [1, 2, 7, 8, 10, 25], composed of 45,312 instances and eight attributes, presents data collected from the Australian New South Wales Electricity Market. In that market the prices are not fixed, varying based on market demand and supply. The prices are set every five minutes and the class

label identifies the change of the price related to a moving average of the last 24 hours. The goal of the problem is to predict if the price will increase or decrease.

The Forest Covertype dataset [8, 15, 24, 25] contains the forest cover type for 30 x 30 meter cells obtained from US Forest Service (USFS) Region 2 Resource Information System (RIS) data. The goal is to predict the forest cover type from cartographic variables. It contains 581,012 instances and 54 attributes, including numeric and categoric ones.

The Poker Hand data set [8, 25] represents the problem of identifying the value of five cards in the Poker game. It is constituted of five categoric and five numeric attributes and one categoric class with 10 possible values informing the value of the hand; for example, one pair, two pairs, a sequence, a street flush, etc. "In the Poker hand data set, the cards are not ordered, i.e., a hand can be represented by any permutation, which makes it very hard for propositional learners, especially for linear ones" [25]. Even though a simpler modified version exists (where the cards are sorted by rank and suit, and duplicates are removed), we decided to use the original harder version in our experiments. The used data set contains 1,000,000 instances.

4 Adaptable Diversity-Based Online Boosting

The Adaptable Diversity-based Online Boosting (ADOB) is a variation of the Online Boosting [15] method, which proposes to distribute instances more efficiently among experts, aiming to more quickly adapt to the situation where concept drifts occur frequently, specially if they are abrupt. This distribution is performed by controlling the diversity (λ) through the accuracy of each expert.

When a new distribution starts, the accuracy of the experts can be used to reduce the initial error, increasing the focus on instances of difficult classification and accelerating the diversity. As the experts have a high degree of similarity in the beginning, due to the reduced number of known instances, the accuracy can be used to define their behaviors. However, because more instances of the same distribution are coming and experts are diversifying, the use of the experts accuracies will tend to have little influence in their behaviors.

Algorithm 1 shows the ADOB pseudo-code, which is our modified version of the Online AdaBoost algorithm [15]. Initially, the ensemble of classifiers (h) is sorted by accuracy in ascending order. Before, several variables are initialized, including $minPos$ and $maxPos$ with values that represent the classifier with the worst and best accuracy, respectively, as well as λ, λ^{sc}, and λ^{sw} (lines 1 to 4).

When an instance d arrives, initially the expert with less accuracy will be selected. If the instance is correctly classified, we assume that probably the other experts, which are more accurate, will also have good chances of correctly classifying it. However, the correct classification of the worst expert does not guarantee that the others will do it properly too, even if an error is unlikely. Accordingly, we will refer to the error of another expert with better accuracy as an unlikely error.

Algorithm 1. Adaptable Diversity-based Online Boosting

Input: ensemble size M, instance d, ensemble h

1 $minPos \leftarrow 1$; $maxPos \leftarrow M$;
2 $correct \leftarrow$ false;
3 $\lambda \leftarrow 1.0$; $\lambda^{sc} \leftarrow 0.0$; $\lambda^{sw} \leftarrow 0.0$;
4 **sort** h **by** accuracy **in** ascending order;
5 **for** $m \leftarrow 1$ **to** M **do**
6 **if** *correct* **then**
7 $pos \leftarrow maxPos$; $maxPos \leftarrow maxPos$ - 1;
8 **else**
9 $pos \leftarrow minPos$; $minPos \leftarrow minPos + 1$;
10 **end**
11 $K \leftarrow$ Poisson(λ);
12 **for** $k \leftarrow 1$ **to** K **do**
13 $h_{pos} \leftarrow$ Learning(h_{pos}, d);
14 **end**
15 **if** $h_{pos}(d)$ *was correctly classified* **then**
16 $\lambda_m^{sc} \leftarrow \lambda_m^{sc} + \lambda$;
17 $\lambda \leftarrow \lambda \left(\frac{N}{2\lambda_m^{sc}}\right)$;
18 $correct \leftarrow$ true;
19 **else**
20 $\lambda_m^{sw} \leftarrow \lambda_m^{sw} + \lambda$;
21 $\lambda \leftarrow \lambda \left(\frac{N}{2\lambda_m^{sw}}\right)$;
22 $correct \leftarrow$ false;
23 **end**
24 **end**
25 **return** h;

Looking into lines 15 to 23 of Algorithm 1, it is possible to observe that the value of λ will be reduced when the classification is done correctly and increased when it is incorrect. In this way, if an unlikely error occurs, the later it occurs, the smaller the influence on λ it will have. To minimize the consequences of an unlikely error, the next expert selected to do the classification will be the one with the best accuracy, followed by the second best, and so on (lines 6 to 7). Using this procedure, experts with the worst accuracies, and most likely to make mistakes, will only be selected at the end.

Another possible scenario is the case where the expert with the worst accuracy incorrectly classifies the instance. In this situation, we distribute the greatest possible λ for the next experts which are more likely to make mistakes in the classification. Therefore, the next experts will be selected according to their performances, from the worst to the best (lines 8 to 9). Assuming that experts with the lowest performances have higher probabilities of making mistakes in the classification, we force them to be selected earlier and maximize λ.

As previously mentioned, after more instances of the same distribution are presented, the lower the influence of this procedure in the experts accuracy will be. At this stage, the experts tend to have a low correlation, and the accuracy of the worst expert will have low importance to the others. Thus, ADOB will now have similar behavior to the original online boosting [15], except for the fact that experts begin to be selected unpredictably, varying for each instance. Thus, these changes are especially valid for situations in which the concept changes often and abruptly, as a consequence of their rapid recovery.

To help understanding how ADOB is used, Algorithm 2 presents a simplified version of MOA's singleClassifierDrift. The classification result is monitored by a concept drift detection method – we used ADWIN. To classify new instances, $\text{ADOB}_{classifier}$ (line 4) behaves the same as defined in [15] and the return is:

$h(x) = argmax_{y \in Y} \sum m : h_m(x) = y^{log \frac{1}{\beta_m}}$, where

$\beta_m = \frac{\epsilon_m}{1-\epsilon_m}$, $\epsilon_m = \left(\frac{\lambda_m^{sw}}{\lambda_m^{sc} + \lambda_m^{sw}} \right)$,

and $m \in [1..M]$ is limited by the number of experts. If ADWIN returns a warning, a new ensemble h_2 immediately starts to be trained using ADOB alongside the existing one (lines 6-7). When the drift is confirmed, the newly created ensemble is used and the old one, representing the last distribution, is removed (lines 8-9).

Algorithm 2. Simplified code of MOA's singleClassifierDrift with ADWIN

```
   Input: ensemble size M, data stream D, base learner b
 1 ADWIN ← new ADWIN method;
 2 h, h2 ← new ensemble using M times b;
 3 foreach instance d in D do
 4 |   ADWIN ← ADOB_classifier(M, d, h);
 5 |   switch ADWIN do
 6 |   |   case detect a warning level
 7 |   |   |   h2 ← ADOB(M, d, h2);
 8 |   |   case detect a drift
 9 |   |   |   h ← h2;   h2 ← reset ensemble;
10 |   endsw
11 |   h ← ADOB(M, d, h);
12 end
```

It is worth pointing out that, in the real code, h_2 is also reset when a warning is *not* confirmed. This and other less important details were omitted here.

5 Experiments Configuration and Results

This section describes the set up and results of the experiments used to evaluate ADOB against other implementations of online bagging and boosting, as well as other recent ensemble methods aimed at detecting concept drifts in data streams. The chosen methods are: ADWINBAG, LEVERAGING, DWM, DDD, and AUE2. The

original online version of AdaBoost (OzaBoost) [15] was included in the tests for comparative purposes, also using ADWIN to detect concept drifts. This method will be called Adwin Boosting (ADWINBOOST). All these methods were compared in terms of accuracy, execution time, and memory used.

The choice of methods was also based on the following additional criteria: ADWINBAG and ADWINBOOST were selected because they implement the original version of online bagging and boosting, respectively; LEVERAGING and DDD, because they use modified versions of the online bagging; whereas DWM and AUE2, because they set their own training strategies to detect changes.

To compute the precision, memory usage, and execution time of the methods, experiments were repeated 40 times in the artificial datasets. Average was computed alongside with a 95% confidence interval. In addition, each artificial dataset was composed of 10,000 instances.

Finally, all the tests were performed in a Core i3 350M processor, 2GB of main memory, running the Ubuntu 12.04 64 bits operating system.

5.1 Drift Configuration in the Artificial Datasets

In the configuration of the artificial datasets, we inserted abrupt and gradual concept drifts. Noise was also inserted in some datasets in order to check the behavior of the methods in these situations.

Two versions of the LED dataset were used, both with gradual concept drifts. In one dataset, drifts occur at instances 3,000 and 6,000. In the other, four gradual changes were inserted at instances 2,000, 4,000, 6,000, and 8,000. In both versions, every time a change occurs, 10% of noise was added.

Other gradual changes were tested, this time making use of the Random RBF dataset. The position of the concept drifts in the two versions of RBF were the same used in the LED data sets.

The two versions of both Sine and Stagger datasets have four and eight abrupt changes, respectively. In their first configurations, the changes were inserted at instances 2,000, 4,000, 6,000, and 8,000. In their second versions, eight changes occur at instances 2,000, 3,000, 4,000, ..., 8,000, and 9,000, respectively.

5.2 Ensemble Methods Configuration

To perform a fair comparison between the methods, common parameters were all set similarly: the base learner was a Hoeffding Tree [18] and the number of experts was set to ten. To set individual parameters, each method was executed with ten different configurations for each dataset used, to check if their default values were the ones which produced the best accuracy. In most cases this was indeed the case, but there were exceptions. In these few cases, we adopted a different parametrization – specific values are given below.

To detect concept drifts, ADWINBAG, ADWINBOOST, LEVERAGING, and ADOB all make use of ADWIN. In all of them, the δ parameter of ADWIN, that corresponds to the maximum global error, was set to 0.1. This value influences the hypothesis test used to check for any change in the distribution [3].

DDD originally uses the Early Drift Detection Method (EDDM) [2] to detect changes. The parameter values used for EDDM were their defaults, i.e., $n = 30$, $w = 0.95$, and $d = 0.99$. These represent, respectively, the number of instances before starting to detect changes, the confidence level to activate the warning level, and the confidence level to detect a change.

Regarding the parameters of the methods, LEVERAGING uses λ, which controls the weight of resampling. The higher the value of λ, the greater the probability that a given instance is repeated for each expert ensemble. This probability is defined according to the Poisson distribution [9]. In our tests, we used $\lambda = 6$.

DWM uses three parameters: p, which corresponds to the time needed to verify if any expert will be removed or added as well as to update their weights if any classifier incorrectly classifies the actual instance; β, the value that will be decremented by the expert every time it makes a mistake; and, finally, θ, which is the minimum value that an expert can have without being removed [7]. We used $p = 100$ (artificial datasets), $p = 250$ (real datasets), $\beta = 0.5$, and $\theta = 0.01$.

The parameters of DDD are: W, responsible for controlling the robustness of the method to false alarms; λ_l, useful to define the value that will represent an ensemble with low diversity; and λ_h, a parameter that will represent an ensemble with high diversity [10]. In our experiments we used $W = 1$ (except in the LED datasets), $W = 3$ (LED datasets), $\lambda_l = 1$, and $\lambda_h = 0.05$.

Finally, AUE2 has two parameters, which control the memory usage (m) and the chunk size (c). The first is responsible for limiting the maximum amount of memory that each component of the ensemble may have. The latter defines the number of instances needed to check the accuracy and memory usage of the ensemble members [11]. After preliminary tests, their values were defined as $m = 32$MB, $c = 50$ (except in the Forest Covertype and Poker Hand datasets), and $c = 500$ (Forest Covertype and Poker Hand datasets).

5.3 Accuracy Analysis

Table 1 presents the accuracies obtained for each method on the artificial and real-world datasets. Bold values identify the best results. The average rank is the average of the positions that each method achieved in different datasets.

In the LED dataset with two concepts drifts, the ADWINBAG method had the best accuracy, closely followed by leveraging. Following, there are DDD, ADOB, and AUE2. In general, these methods have similar performance, with differences ranging from 0.1% to 3%, approximately. On the other hand, ADWINBOOST and DWM obtained the worst results. As can be seen in the first graphic of Figure 1, the slow recovery of ADWINBOOST negatively affects its accuracy at every concept drift. In addition, DWM had the worst performance throughout the dataset.

With the addition of two more drifts in the LED dataset, as might be expected, the performance of all methods deteriorated. The most affected method was AUE2, with a drop of about 3%. The less impacted was DWM, with a worsening of only 0.74%. However, DWM's performance remained well below those of the others. In descending order of accuracy, the result is LEVERAGING, ADWINBAG, DDD, ADOB, AUE2, ADWINBOOST, and DWM.

Table 1. Average accuracy in percentage (%) with 95% confidence in artificial datasets

	ADOB	AdwinBag	AdwinBoost	Leveraging	DWM	DDD	AUE2
LED_2	59.30±0.14	**61.18±0.08**	55.52±0.53	61.08±0.09	45.48±0.78	60.62±0.16	58.18±0.23
LED_4	56.86±0.10	58.57±0.09	53.69±0.52	**59.01±0.06**	44.74±0.89	57.92±0.14	55.13±0.29
RBF_2	53.08±0.26	**54.59±0.15**	47.13±0.45	39.20±0.15	40.12±0.62	53.41±0.21	53.26±0.27
RBF_4	56.77±0.40	**58.29±0.12**	51.13±0.36	43.58±0.20	42.75±0.59	57.58±0.22	57.19±0.28
$Sine_4$	90.28±0.10	81.84±0.79	**90.68±0.13**	89.90±0.14	87.60±0.43	88.97±0.31	87.57±0.21
$Sine_8$	88.42±0.10	80.14±0.53	**88.91±0.12**	87.42±0.14	86.90±0.36	87.55±0.33	86.06±0.23
$Stagger_4$	**99.02±0.02**	92.73±0.21	96.00±1.21	97.80±0.05	96.45±0.32	97.47±0.15	94.27±0.17
$Stagger_8$	**98.74±0.02**	90.32±0.18	96.52±0.82	94.70±0.13	95.34±0.29	96.59±0.13	88.29±0.19
Elec	87.98	86.44	87.09	**89.71**	88.17	85.72	80.98
Cov	82.79	84.44	81.31	**88.16**	87.34	83.96	65.95
Poker	**53.74**	53.20	52.42	52.18	46.60	53.19	48.84
Rank	**2.82**	3.82	4.18	3.36	5.09	3.18	5.55

Analyzing the RBF dataset with two concept drifts, ADWINBAG again had the best accuracy, closely followed by DDD, AUE2, and ADOB. ADWINBOOST, DWM, and LEVERAGING had the worst results. Unlike the previous dataset, with the insertion of two more concept drifts, the performance of the methods improved. This can be explained by how the changes were defined. In RBF_2, the gradual changes take twice as long to fully occur than in RBF_4. Thus, methods spend more time to detect if there was a change and, consequently, take longer to recover. Although all methods increased their accuracies in this version, the order remained the same, except for the fact that DWM assumed the worst position, swapping places with LEVERAGING.

Up to this point, the presented datasets included gradual concept drifts. Making a general analysis of this type of change, the methods that had the best accuracies were: ADWINBAG, DDD, ADOB, and AUE2, with differences ranging from 0.77% to 2.21%. Right after, with a considerable distance, follows ADWINBOOST, LEVERAGING, and DWM occupying the last positions, respectively.

Differently from the other datasets, at Sine, ADWINBAG returned the worst results. Its differences to AUE2 are of 5.73% and 5.92%, in the datasets with four and eight drifts, respectively. These are significant differences, given that the differences from the first (ADWINBOOST) to the second to last (AUE2, in both cases) are of only 3.11% and 2.85%, respectively.

Finally, on Stagger the results were somewhat different from those on Sine. AUE2 and ADWINBAG were again the worst methods, but the order was different in the two versions used. The best accuracies were achieved by ADOB and DDD.

Analyzing the overall accuracy of the methods in all datasets with abrupt concept drifts, ADOB had the best performance, followed by ADWINBOOST, DDD, LEVERAGING, DWM, AUE2, and ADWINBAG. Comparing these results with the results of the datasets with gradual concept drifts, ADWINBAG had the highest drop in performance, from first to last. The methods that maintained a better balance in different situations were ADOB and DDD, respectively.

In the real datasets, LEVERAGING was the method with the best overall performance, followed by ADOB, ADWINBAG, DDD, ADWINBOOST, DWM, and AUE2, respectively. It is worth noting that, despite its poor overall performance in these

datasets, DWM was the second best in both Electricity and Forest Covertype. This is explained by its very bad performance in the Poker Hand dataset.

Finally, observe that, in the first two graphics of Figure 1, it is possible to visualize the gradual and abrupt changes, respectively. However, in the third, referring to the Electricity dataset, apparently no drastic change occurs.

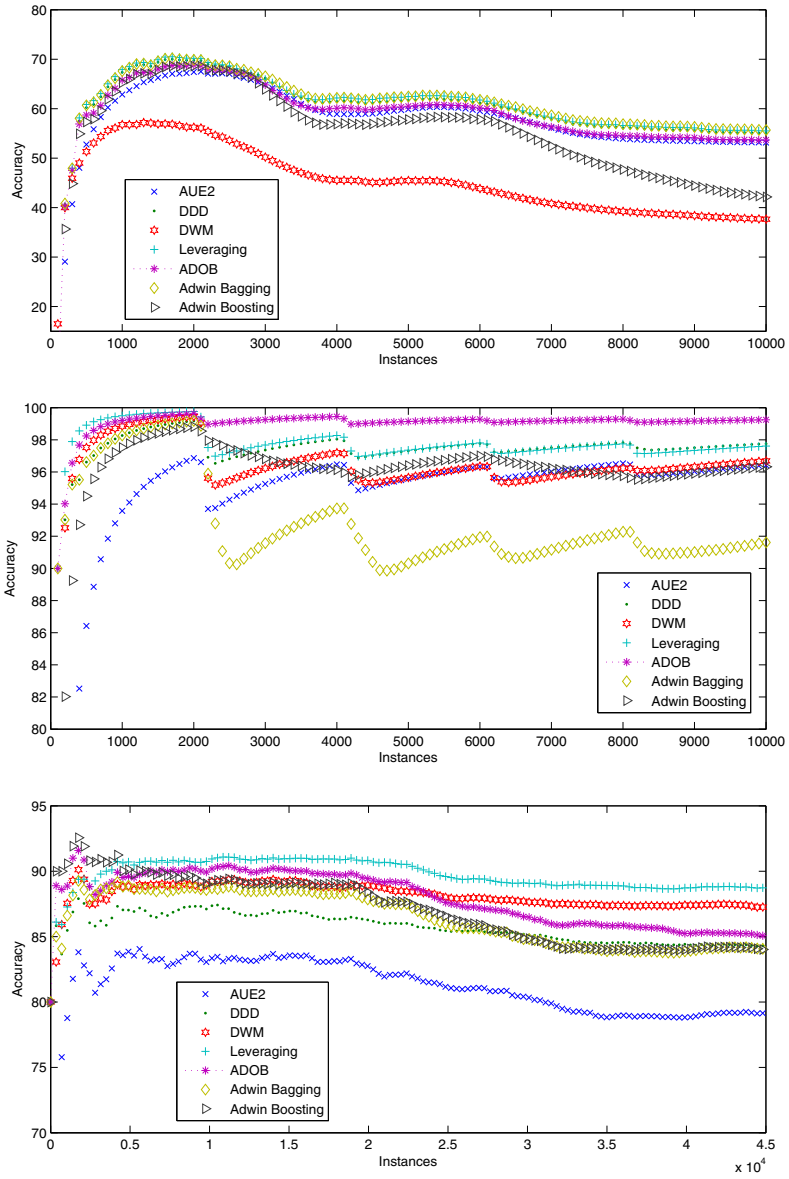

Fig. 1. Accuracy results in the LED_2, $Stagger_4$ and Electricity datasets

Complementing the accuracy analysis, a statistic based on the nonparametric Friedman test [26] was used. For this test, the null hypothesis states that all methods are statistically equal. If this hypothesis is rejected, the test indicates that there is a statistical difference in any of the methods, but it does not specify which method(s). For this task, the use of a post-test is required. In our case, we used the Bonferroni-Dunn [27] to compare ADOB with the other methods.

Initially, we compared the accuracies of the methods, using a 95% confidence interval: $F_{6;60} = 2.25$. Therefore, with $F_F = 2.79$ (bigger than $F_{6;60}$), the null hypothesis is rejected. Then, proceeding to the post-test, the critical difference (CD) was found to be 2.44. So, we can say that ADOB is statistically superior in accuracy to AUE2. Figure 2 graphically represents these results.

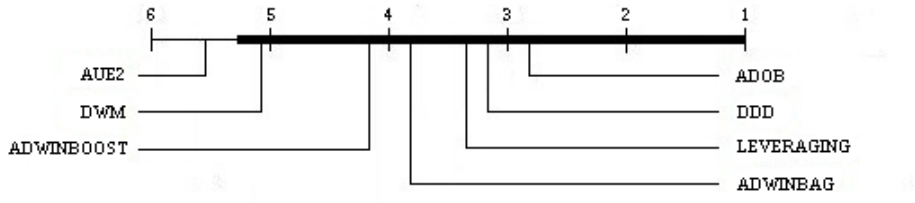

Fig. 2. Comparison results of ADOB against the other methods using the Bonferroni-Dunn test with a 95% confidence interval

5.4 Time Analysis

Table 2 displays the values in seconds that each method took to rank the different datasets. An important observation is that the times of the real-world datasets were significantly higher than those of the artificial datasets, except for RBF and Electricity, which presented similar times. Because the real-world datasets are much bigger, requiring more time to be processed is an expected behavior.

Table 2. Average runtime in seconds (s) with 95% confidence in artificial datasets

	ADOB	AdwinBag	AdwinBoost	Leveraging	DWM	DDD	AUE2
LED_2	3.34±0.04	3.05±0.03	2.86±0.04	3.74±0.02	**2.74±0.05**	4.87±0.14	4.09±0.02
LED_4	3.62±0.05	3.23±0.03	3.05±0.04	3.90±0.03	**2.72±0.04**	5.74±0.11	4.13±0.02
RBF_2	8.10±0.07	8.79±0.10	**7.76±0.10**	12.63±0.06	9.63±0.11	13.80±0.50	11.65±0.11
RBF_4	9.34±0.14	9.46±0.09	**7.79±0.09**	13.43±0.10	9.42±0.11	16.28±0.61	13.16±0.14
$Sine_4$	2.30±0.03	2.10±0.02	2.36±0.04	3.79±0.04	**1.48±0.04**	2.97±0.06	2.49±0.03
$Sine_8$	2.59±0.04	2.14±0.03	2.58±0.05	3.80±0.04	**1.58±0.03**	3.37±0.06	2.53±0.04
$Stagger_4$	1.67±0.03	1.66±0.03	1.68±0.03	2.09±0.02	**0.70±0.01**	1.59±0.02	1.99±0.02
$Stagger_8$	1.81±0.03	1.59±0.03	1.84±0.03	2.22±0.02	**0.69±0.01**	1.78±0.02	2.93±0.07
Elec	8.98	7.22	8.27	16.65	**4.73**	15.25	8.18
Cov	291.52	228.08	229.42	468.86	**165.49**	545.17	312.90
Poker	469.67	358.27	374.05	3574.45	**155.88**	358.56	152.66

Analyzing these results, the slowest artificial dataset was RBF, with an average of 11.27s and 10.34s in the versions with four and two concept drifts, respectively. All the others were much faster. The fastest was Stagger, with an average time of 1.84s and 1.63s in the versions with 8 and 4 drifts, respectively.

In the real-world datasets, the average times were somewhat proportional to the number of instances: Poker Hand was the one that took longer (777.65s), followed by Forest Covertype (320.21s), and Electricity (9.90s).

A fact that possibly influences the time is diversity. The higher the diversity, the higher the probability of the same instance be repeatedly distributed to a different expert and hence more time be used. ADOB and ADWINBOOST are methods that tend to increase their diversities in proportion to the error. Thus, the lower the accuracy of the method – which suggests more errors – the greater the diversity, causing a longer running time.

For example, observing the RBF and Stagger results, the ones with the lowest and highest accuracies, were the ones with the highest and lowest times in the artificial datasets, respectively. The same idea might be true in the real-world datasets, but these should be compared separately because they possess very different numbers of instances. The differences in execution times between ADOB and ADWINBOOST demonstrate the first attempt to maximize/minimize diversity, as discussed in Section 4, in favor of a better accuracy in different situations.

On the other hand, one of the characteristics of LEVERAGING is to maintain a higher diversity during the processing of the instances. As a result, it is the slowest method, followed by DDD. Although ADWINBAG and LEVERAGING use a static diversity, Minku et al. [22] show that different diversities in different situations of concept drifts contribute to improve the accuracy of the method.

5.5 Memory Analysis

We decided to monitor the memory usage to confirm that, despite using more memory than ADWINBOOST, ADOB is *not* memory intensive in absolute terms.

The evaluation was based on a metric that computes the amount of memory used by the methods per hour (in KB) and the results are presented in Table 3. Again, in the real-world datasets, the methods used much more memory than in the artificial datasets as a consequence of the greater number of instances.

The method with the lowest memory usage was DWM, followed by AUE2, ADWINBOOST, ADWINBAG, ADOB, DDD, and LEVERAGING. A possible explanation for the high memory usage of DDD is that it stores four times more classifiers than the value set by the user. But not all of them are used to classify each instance: the best ones are chosen within the ensemble according to each situation.

Memory usage by methods using both the original and the modified online bagging/boosting versions can be explained similarly to the explanation made to the execution time. The greater the diversity, the greater the likelihood of repetition of an instance among experts involving more training. As a direct consequence, it increases the used memory. The observation of the relationship with the precision, made earlier, apparently also applies to the memory usage. However, it is important to notice that there are exceptions in both cases.

Table 3. Average memory usage in (KB/h) with 95% confidence in artificial datasets

	ADOB	AdwinBag	AdwinBoost	Leveraging	DWM	DDD	AUE2
LED_2	0.34±0.00	0.21±0.00	0.29±0.00	0.26±0.00	**0.02±0.00**	1.29±0.05	0.26±0.00
LED_4	0.36±0.01	0.22±0.00	0.31±0.00	0.26±0.00	**0.02±0.00**	1.56±0.04	0.27±0.00
RBF_2	1.14±0.01	0.94±0.01	1.09±0.01	2.18±0.05	**0.10±0.00**	6.23±0.29	1.20±0.01
RBF_4	1.31±0.02	1.00±0.01	1.09±0.01	2.89±0.06	**0.10±0.00**	7.36±0.37	1.36±0.01
$Sine_4$	0.18±0.00	0.10±0.00	0.18±0.00	0.46±0.01	**0.01±0.00**	0.44±0.01	0.07±0.00
$Sine_8$	0.20±0.00	0.10±0.00	0.19±0.00	0.39±0.01	**0.01±0.00**	0.49±0.01	0.07±0.00
$Stagger_4$	0.10±0.00	0.07±0.00	0.10±0.00	0.11±0.00	**0.00±0.00**	0.18±0.00	0.04±0.00
$Stagger_8$	0.11±0.00	0.06±0.00	0.11±0.00	0.12±0.00	**0.00±0.00**	0.20±0.00	0.29±0.01
Elec	0.85	0.47	0.86	3.43	**0.05**	2.40	0.29
Cov	37.54	25.75	31.79	290.38	**3.76**	168.50	27.41
Poker	293.80	208.34	195.37	30816.18	**0.67**	266.08	6.17

6 Conclusion

Dealing with concept drifts in data streams is a challenging topic, given that such drifts can be abrupt or gradual, slow or fast, rare or frequent, cyclical or not, etc. Thus, the single classifier approach is unlikely to achieve good results in general and, so, the ensemble of classifiers methods are becoming more popular.

This paper presented ADOB, an ensemble algorithm based on the Online Boosting [15] method specially built to deal more efficiently with *frequent* and *abrupt* concept drifts on on-line learning environments. More specifically, ADOB proposes to distribute instances more efficiently among experts, by controlling the diversity (λ) through the accuracy of each expert, aiming at recovering faster from the situations where concept drifts occur frequently.

We run experiments to compare ADOB to six different online ensemble methods, including other variations of Online Bagging and Boosting [15], all of them using ADWIN [3] as their drift detector, namely Adwin Bagging [8], Leveraging Bagging [9], and ADWINBOOST, as well as other well known and/or recent ensembles such as DWM [7], DDD [10], and AUE2 [11].

To perform the comparison, we used two different versions of four selected artificial datasets (eight in total), with both abrupt and gradual concept drifts, as well as three real-world datasets, all of them chosen from the most used ones in the concept drift research area.

It is important emphasizing that our main subject of interest in these experiments was the performance evaluation of Algorithm 1 – more specifically we wanted to compare it to the ADWINBOOST version that inspired it. Mainly for this reason, ADOB, ADWINBOOST, ADWINBAG, and LEVERAGING all used very similar versions of Algorithm 2 – MOA's singleClassifierDrift using ADWIN as drift detection method, as well as the same parametrization. DDD also used a similar version of singleClassifierDrift but the selected drift detection method was EDDM, as in its original reference [10].

The tested ADOB configuration presented good precision in several situations and, in particular, it was specially efficient in the Stagger [17] and Sine [1] datasets, which had abrupt concept drifts. It is worth pointing out that ADOB presented the best overall accuracy considering all tested datasets. In addition,

according to a statistic based on the non-parametric Friedman test, ADOB presented statistically superior accuracy, when compared to AUE2, and comparable performance to the other methods in the tested data sets.

Even so, we believe the efficiency of ADOB can be further improved, both by optimizations in the algorithms and by using different drift detection methods in different types of datasets. These might be investigated in the near future. Another possible future work is a deeper investigation of the relationship between diversity and accuracy of the methods with the run time and memory usage.

Finally, it is worth pointing out that both ADOB and DDD were implemented as part of this work. They have been added to the MOA framework and are freely available at `https://sites.google.com/site/moamethods`. The implementation of ADWINBOOST was a mere parametrization of code previously available in MOA. DWM was already available at `https://sites.google.com/site/moaextensions`.

Acknowledgements. Silas Santos is supported by postgraduate grant number 0837-1.03/12 from FACEPE. We also thank the comments and suggestions from the anonymous referees which helped to improve this final version of the paper.

References

1. Gama, J., Medas, P., Castillo, G., Rodrigues, P.: Learning with drift detection. In: Bazzan, A.L.C., Labidi, S. (eds.) SBIA 2004. LNCS (LNAI), vol. 3171, pp. 286–295. Springer, Heidelberg (2004)
2. Baena-García, M., Del Campo-Ávila, J., Fidalgo, R., Bifet, A., Gavaldá, R., Morales-Bueno, R.: Early drift detection method. In: International Workshop on Knowledge Discovery from Data Streams, IWKDDS 2006, pp. 77–86 (2006)
3. Bifet, A.: Learning from time-changing data with adaptive windowing. In: Proceedings of the Seventh SIAM International Conference on Data Mining, SDM 2007, Lake Buena Vista, Florida, USA, pp. 443–448. SIAM (2007)
4. Ross, G.J., Adams, N.M., Tasoulis, D.K., Hand, D.J.: Exponentially weighted moving average charts for detecting concept drift. Pattern Recognition Letters 33(2), 191–198 (2012)
5. Nishida, K., Yamauchi, K.: Detecting concept drift using statistical testing. In: Corruble, V., Takeda, M., Suzuki, E. (eds.) DS 2007. LNCS (LNAI), vol. 4755, pp. 264–269. Springer, Heidelberg (2007)
6. Page, E.S.: Continuous inspection schemes. Biometrika 41(1/2), 100–115 (1954)
7. Kolter, J.Z., Maloof, M.A.: Dynamic weighted majority: An ensemble method for drifting concepts. Journal of Machine Learning Research 8, 2755–2790 (2007)
8. 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, pp. 139–148. ACM, New York (2009)
9. Bifet, A., Holmes, G., Pfahringer, B.: Leveraging bagging for evolving data streams. In: Balcázar, J.L., Bonchi, F., Gionis, A., Sebag, M. (eds.) ECML PKDD 2010, Part I. LNCS, vol. 6321, pp. 135–150. Springer, Heidelberg (2010)

10. Minku, L.L., Yao, X.: DDD: A new ensemble approach for dealing with concept drift. IEEE Transactions on Knowledge and Data Engineering 24(4), 619–633 (2012)
11. Brzezinski, D., Stefanowski, J.: Reacting to different types of concept drift: The accuracy updated ensemble algorithm. IEEE Transactions on Neural Networks and Learning Systems 25(1), 81–94 (2013)
12. Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. Machine Learning 23(1), 69–101 (1996)
13. Ramamurthy, S., Bhatnagar, R.: Tracking recurrent concept drift in streaming data using ensemble classifiers. In: Proceedings of the 6th International Conference on Machine Learning and Applications, ICMLA 2007, pp. 404–409. IEEE Computer Society, Los Alamitos (2007)
14. Gonçalves, Jr. P.M., Barros, R.S.M.: RCD: A recurring concept drift framework. Pattern Recognition Letters 34(9), 1018–1025 (2013)
15. Oza, N.C., Russell, S.: Online bagging and boosting. In: Artificial Intelligence and Statistics 2001, pp. 105–112. Morgan Kaufmann (2001)
16. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: MOA: Massive online analysis. Journal of Machine Learning Research 11, 1601–1604 (2010)
17. Schlimmer, J.C., Granger, R.H.: Incremental learning from noisy data. Machine Learning 1(3), 317–354 (1986)
18. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2001, pp. 97–106. ACM, New York (2001)
19. Breiman, L.: Bias, variance, and arcing classifiers. Technical report, Statistics Department, University of California, Berkeley, CA, USA (1996)
20. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: International Conference on Machine Learning, vol. 96, pp. 148–156 (1996)
21. Blum, A.: Empirical support for winnow and weighted-majority algorithms: Results on a calendar scheduling domain. Machine Learning 26(1), 5–23 (1997)
22. Minku, L.L., White, A.P., Yao, X.: The impact of diversity on online ensemble learning in the presence of concept drift. IEEE Transactions on Knowledge and Data Engineering 22(5), 730–742 (2010)
23. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. In: Wadsworth Statistics / Probability series. Wadsworth International Group, Belmont (1984)
24. Gama, J., Rocha, R., Medas, P.: Accurate decision trees for mining high-speed data streams. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2003, pp. 523–528. ACM Press, New York (2003)
25. Bifet, A., Holmes, G., Pfahringer, B., Frank, E.: Fast perceptron decision tree learning from evolving data streams. In: Zaki, M.J., Yu, J.X., Ravindran, B., Pudi, V. (eds.) PAKDD 2010. LNCS, vol. 6119, pp. 299–310. Springer, Heidelberg (2010)
26. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30 (2006)
27. Dunn, O.J.: Multiple comparisons among means. Journal of the American Statistical Association 56(293), 52–64 (1961)