Comparative Study of Fast Stacking Ensembles Families Algorithms

Laura Maria Palomino Mariño[1](✉), Agustín Alejandro Ortiz-Díaz[2], and Germano Crispim Vasconcelos[1]

[1] Centro de Informática, Universidade Federal de Pernambuco, Recife, Brazil
{lmpm,gcv}@cin.ufpe.br

[2] Centro de Ciências Tecnológicas, Universidade do Estado de Santa Catarina, Joinville, Brazil
agaldior@gmail.com

Abstract. One of the main challenges in Machine Learning and Data Mining fields is the treatment of large Data Streams in the presence of Concept Drifts. This paper presents two families of ensemble algorithms designed to adapt to abrupt and gradual concept drifts. The families Fast Stacking of Ensembles boosting the Old (FASEO) and Fast Stacking of Ensembles boosting the Best (FASEB) are adaptations of the Fast Adaptive Stacking of Ensembles (FASE) algorithm to improve run-time, without presenting a significant decrease in terms of accuracy when compared to the original FASE. In order to achieve a more efficient model, adjustments were made in the update strategy and voting procedure of the ensemble. To evaluate the methods, Naïve Bayes (NB) and Hoeffding Tree (HT) are used, as learners, to compare the performance of the algorithms on artificial and real-world data-sets. An experimental investigation with a total of 32 experiments and the application of Friedman and Bonferroni-Dunn statistical tests showed the families FASEO and FASEB are more efficient than FASE with respect to execution time in many experiments, also some methods achieving better accuracy results.

Keywords: Concept drift · Data stream · Ensemble methods

1 Introduction

In recent years, data generated by different sources such as cell phones, sensors, networks, and satellites has increased significantly. Part of these data can be viewed as a sequence of examples that arrive at high rates and can often be read-only once using a small amount of processing time [1]. In the literature, such data are known as data-streams. According to [2], in the streaming scenario,

Supported by Coord. de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) and Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco - FACEPE (IBPG-0820-1.03/19).

R. Cerri and R. C. Prati (Eds.): BRACIS 2020, LNAI 12320, pp. 456–470, 2020.
https://doi.org/10.1007/978-3-030-61380-8_31

two primary issues have to be dealt with in the construction of training models: One-pass Constraint, and Concept Drift.

The first aspect, One-pass Constraint, is when a model needs to analyze each of the data only once. This feature is very important in online learning. Second, a concept drift can be described from the change between two different concepts: initial concept (P_I) and final concept (P_F) [3]. Attending to the time it takes to change from P_I to P_F (t_{ch}), this change can be abrupt (sudden) ($t_{ch} \sim 0$) or gradual ($t_{ch} > 0$).

Depending on the taxonomy of the involved aspects in the distribution change, different types of Concept Drift can be analyzed: virtual concept drift (the distribution of instances change but the underlying concept does not), real concept drift (there exist a change in the class boundary), recurrence of the concepts (when previously active concept reappears after some time) [4], and some other types.

The presence of Concept Drift affects the performance of the classification algorithms because models become stale over time. Therefore, it is crucial to adjust the model in an incremental way in order for the algorithm to achieve high accuracy over current unknown instances.

Many of the works published so far have focused mainly on accuracy as a fundamental parameter to establish comparisons between learning algorithms. However, in many real-life scenarios other parameters like run-time should also be taken into account due to their importance. Motivated by previous works [5] that analyzed the performance of several algorithms, it was observed that Fast Adaptive Stacking of Ensembles (FASE) [7] (one of the compared methods) presents good accuracy results, but its run-times indicate there is room for improvement.

The present work aims to introduce the families of methods Fast Stacking of Ensembles boosting the Best (FASEB) and Fast Stacking of Ensembles boosting the Old (FASEO) obtained from the algorithm FASE. All methods present, from both families, are designed to adapt to concept drifts, whether abrupt or gradual and to increase the "efficiency" of their base model. The word "efficiency" has a particular meaning here: it is a "suitable" balance among accuracy and run-time highlighting. Furthermore, "suitable" is associated with the degree to which these aspects have relevance in a given context [6]. In order to obtain the variants of FASE, it was experimentally investigated two main modifications, regarding (i) the update strategy and (ii) the voting procedure of the ensemble. This work is an extension of paper [12].

The paper is organized as follows: Sect. 2 describes related work; Sect. 3 explains the families methods and the explored strategies. Sect. 4 presents the data-set characteristics and provides the experimental results analyzing the main findings. Section 5 provides the performance evaluation of the FASE‘s family on Sensor data-stream, identify which of the methods had the best performance in the experimental evaluation. Finally, Sect. 6 concludes.

2 Related Works

In this section, a bibliographic study of the Fast Adaptive Stacking of Ensembles (FASE) [7] and Hoeffding-based Drift Detection Methods (HDDM) [8] will be carried out aiming to highlight the methods related.

2.1 FASE

Fast Adaptive Stacking of Ensembles (FASE) [7] is based on the Online Bagging algorithm [9], and uses HDDM$_A$ as a drift detection mechanism to estimate the error. It has a set of adaptive learners to handle Concept Drift explicitly by detecting the changes and updating the model if a Concept Drift is detected. The adaptive learners estimate error rates (by the corresponding change detectors) with a predictive sequential approach (test-then-train). FASE uses weighted voting to combine the predictions of the main and alternative models. It uses a meta-classifier too, combining the predictions of the adaptive learners. For that, it generates a training meta-instance $M = (\widehat{y}_1, \ldots, \widehat{y}_j, \ldots, \widehat{y}_k; y)$ where each $\widehat{y}_j$ is an attribute value and y is its corresponding class label. Each attribute value $\widehat{y}_j$ of the meta-instance M corresponds to the prediction from classifier h_j for the example $\mathbf{z}$. The class label of the meta-instance M is the same label of the original training example [7].

2.2 HDDM

Hoeffding-based Drift Detection Methods (HDDM) authors [8] propose to monitor the performance of the base learner by applying "some probability inequalities that assume only independent, univariate and bounded random variables to obtain theoretical guarantees for the detection of such distributional changes". HDDM$_A$ "involves moving averages and is more suitable to detect abrupt changes" and the second HDDM$_W$ "follows a widespread intuitive idea to deal with gradual changes using weighted moving averages". For both cases, the Hoeffding inequality [10] is used to set an upper bound to the level of difference between averages.

3 FASEO and FASEB Families Methods

This section introduces two families of classifier ensemble methods derived from FASE: FASEO and FASEB. These algorithm families are originated using different change adaptation strategies and methods to combine the predictions of the classifiers that make up the ensemble.

3.1 Overview of the Methods

According to [11], when designing ensemble of classifiers two main points must be considered: (i) how the base classifiers in the ensemble are updated and (ii) how they are combined to make a joint prediction. Taking into account the above assumption, variations were introduced in FASE to search for a better balance between accuracy and other necessary resources (run-time) for its operation.

The original algorithm is composed of a set of adaptive classifiers. Each one is formed by a base classifier and an alternative classifier (both classifiers include a drift detection mechanism) that is generated each time the base classifier issues a *warning* state. Both classifiers, the main and the alternative, process each instance and by weighted voting determines its class. This strategy is followed in the adaptive classifiers that form the ensemble, but also in the level of the meta-classifier that receives as input the predictions of each classifier in the form of a meta-instance [7].

Considering this scenario, the derived methods aim to handle resources more efficiently while keeping accuracy at similar levels. Thus, the main proposed modifications made on FASE are (i) the update strategy and (ii) the voting procedure of the ensemble. As a result of these modifications, two families of ensemble algorithms derived from FASE were devised: FASEO and FASEB.

In general, one of the main modifications made to the algorithm variants derived from FASE was to eliminate the use of alternative adaptive classifiers and create, in the structure of the general model, a parallel ensemble. In this alternative ensemble, a classifier is activated and begins to train once one of the classifiers of the main ensemble reaches the *warning* level. On the other hand, when a concept-drift level is detected, then one of two variants is followed, (i) the oldest classifier, the one that stayed longer in the alternative classifier ensemble is promoted, or (ii) the classifier with the best accuracy is promoted. Based on these strategies, the families of algorithms FASEO and FASEB were created respectively.

Each family of algorithms is based on classifiers that integrate detection mechanisms. So, the associated detector triggers each of the three different drift signals manipulated in the model. The first two methods, (FASEO and FASEB), maintained the meta-classifier proposed in FASE in order to perform class voting while the others combine weighted voting for final decision. To determine the whole weight of each classifier, accuracy, entropy degree and class probabilities are combined in different ways. Moreover, they are also considered two-class voting strategies. The first variant uses combined voting using a meta-classifier, like the FASE algorithm. The second one uses combined voting using weighted majority voting in different ways. The description of each algorithm follows below: The description of each algorithm follows below:

- FASEO: To update the main ensemble, the classifier with more training time in the set of alternative classifiers is promoted. To determine the final class, a meta-classifier is used with its inputs being the meta-instances formed by the predictions of each classifier in the main ensemble.
- FASEO$_{wv1}$: As in FASEO, to update the main ensemble, the oldest classifier in the set of alternative classifiers is promoted. To vote the final class, the weight of each classifier is computed taking into account accuracy, entropy degree, and the class probability vector.

- FASEO$_{wv2}$: As in FASEO and FASEO$_{wv1}$, to update the main ensemble the classifier with more training time in the set of alternative classifiers is promoted. To vote the final class, the weight of each classifier is computed taking into account only accuracy and the class probability vector.
- FASEO$_{wv3}$: As in the three former cases, to update the main ensemble, the classifier with more training time in the set of alternative classifiers is promoted. To vote the final class, the weight of each classifier is computed taking into account only accuracy and entropy degree.
- FASEB [12]: To update the main ensemble, the classifier with the best accuracy among the alternative classifiers is promoted. The decision on the final class is given by a meta-classifier, whose inputs are the meta-instances formed by the predictions from each classifier in the main ensemble.
- FASEB$_{wv1}$: As in FASEB, to update the main ensemble, the classifier with the best accuracy among the alternative classifiers is promoted. To vote the final class, the weight of each classifier is computed taking into account accuracy, entropy degree, and the class probability vector.
- FASEB$_{wv2}$: As in FASEB and FASEB$_{wv1}$, to update the main ensemble, the classifier with the best accuracy among the alternative classifiers is promoted. To vote the final class, the weight of each classifier is computed taking into account only accuracy and the class probability vector.
- FASEB$_{wv3}$ [12]: As in the three last cases, to update the main ensemble, the classifier with the best accuracy is promoted. To vote the final class, the weight of each classifier is computed taking into account only accuracy and entropy degree.

3.2 The Update Strategy

This section provides a description of update strategy used in the FASEO and FASEB families. In a general way, the derived methods are updated once one of the learners (classifier with change detection mechanism) that compose the main ensemble, experiment any of the following change of states:

(i) A classifier initially *in-control*, triggered a *warning* (through its detection mechanism)
(ii) A classifier suddenly reaches the *drift* level from *in-control* state
(iii) A classifier reaches the *drift* level from state of *warning*
(iv) A classifier, currently in *warning*, return to the (by-default state) *in-control*

In (i), an alternative classifier is activated and placed in a parallel set (ensemble of alternative classifiers). When no *drift* is detected, the learning process is carried out by the learners of the main set.

When one of the classifiers of the main set reached an out-of-control signal (drift), (case (ii) or (iii)) the main set is updated, firstly the drifted classifier is deleted and then is promoted to the main ensemble a) the alternative classifier with the greatest accuracy (FASEB methods) or b) the oldest alternative classifier (FASEO methods). Once the alternative classifier is promoted, it is deleted from the parallel ensemble. Then, in (ii), the model activates a new alternative classifier, since, due to a sudden change, it was not created and, therefore, it was taken "borrowed" from a classifier that triggered warning, therefore, should be "returned to it".

In order to handle the final states there is an arrangement of states: initially, it is assumed that each value corresponding to the status of each classifier that is part of the main ensemble is *in-control* and therefore takes value 0; whenever a classifier of the ensemble enters *warning*, its status has value 1. When a classifier of the main ensemble detects a drift, either by going from *in-control* to *drift* or from *warning* to *drift*, the algorithm quickly updates the ensemble and the state is again *in-control*. Therefore, once the algorithm removes the classifier from the array of alternatives, it updates the state corresponding to the new classifier that became part of the main ensemble to *in-control*.

3.3 Class Voting Strategies

An ensemble of classifiers $H(\mathbf{x})$ are models learned from a set of classifiers $h_1(\mathbf{x}), ..., h_j(\mathbf{x}), ..., h_k(\mathbf{x})$. During the training process, it receives as an input a labeled instance $(\mathbf{x}_i, y_i)$, and the model $H(\mathbf{x})$ aims to predict the class $\widehat{y}_i$ of each instance in unlabeled data-set. To achieve this task, there is different ways of ensemble combination methods like stacking, voting schemes, unweighted voting schemes. Thus, in these methods the classification is done using the stacking and the weighted-voting strategies. The particular way in which these strategies are applied is explained below.

Meta-classifier Strategy: The goal of a meta-learning process is to train a meta-classifier (meta-learner), which will combine the ensemble members' predictions into a single prediction. Thus, the input of the meta-learner are the outputs of the ensemble-member classifiers. In this process, both the ensemble members and the meta-classifier need to be trained. The meta-classifier is the ensemble's combiner, thus it is responsible for producing the final prediction [13]. Similar to the ensemble-members, the meta-classifier of these methods is a one class classifier; it learns a classification model from meta-instances, whose attributes are nominal. As FASE, both methods uses a Prequential methodology to generate meta-instances; the idea of this methodology is to use each instance first to test the model, and then to train the model. Thus, for each original training instance $\mathbf{z} = (\mathbf{x}, y)$ it is generated a training meta-instance $M = (\widehat{y}_1, \ldots, \widehat{y}_j, \ldots, \widehat{y}_k; y)$, where each attribute value $\widehat{y}_j$ of the meta-instance M corresponds to the prediction from the base classifier j in the main ensemble for the original example $\mathbf{z}$. The class label of the meta-instance M is the same label of the original training example.

Weighted Voting Strategies: A weighted voting is a system in which not all learners have the same amount of influence over the outcome because their votes have a different weight. In the classification task, an ensemble classifier can combines the decision of a set of classifiers by weighted voting to classify unknown examples. The weighting methods are best suited for problems where individual classifiers perform the same task [14]. Therefore, that is the reason why in this work was used the weighted majority vote to obtain the final prediction of the class label.

To determine the weight of each classifier, the accuracy (a component of the weight derived from historical performance) and the degree of entropy in its classification (the component of the weight coming from the current behavior of the classifier) are taken into account. A similar idea was previously proposed in [15].

These component are combined in different ways originating three weighting approaches: (1) it is used the classifier accuracy weight, class probability vector and entropy weight; (2) it is used the classifier accuracy weight and class probability vector; and (3) it is used the classifier accuracy weight and entropy weight.

4 Experimental Results and Analysis

4.1 Data-Sets

Table 1 summarizes the main characteristics of the data-sets used in the experiments. A total of 4 synthetic generators and 8 real data-sets were considered. The synthetic data-sets were built with two different sizes: 10000 (10k) and 50000 (50k) instances. Abrupt and gradual controlled concept drifts were introduced. MOA framework allows us to simulate the different types of concept drift using a sigmoid function. Real data-sets employed are available on the MOA website. These data-sets have very diverse characteristics regarding the number of instances, the number of classes, and the presence or absence of different types of concept drift. This diversity allows us to better describe the real problem situations that algorithms may face.

4.2 Experimental Results and Analysis

This section compares the performance between the families of methods and FASE using the synthetic and real data-sets. Both in the synthetic (8) and real (8) data-sets each algorithm is tested and trained using the classifiers HT and NB. In summary, 32 experiments were carried out to evaluate the performance of each method according to the two metrics considered.

Tables 2 and 3 present accuracy rates and run-times achieved by the algorithms in synthetics and real data-sets using both NB and HT as base learners.

In order to improve the visualization, the methods name was exposed as in parentheses: FASEO (FO), FASEO$_{wv1}$ (FOwv1), FASEO$_{wv2}$ (FOwv2) and FASEO$_{wv3}$ (FOwv3). FASEB (FB), FASEB$_{wv1}$ (FBwv1), FASEB$_{wv2}$ (FBwv2), and FASEB$_{wv3}$ (FBwv3).

The first values appearing in each table refer to respective base-learner used, NB or HT. The first rows of each table show the results obtained over the

Table 1. Main characteristics of synthetic and real datasets

Type	Data-sets	Size	# Atributes	# Class
Synthetic	LED	10k & 50k	24	10
	Sine	10k & 50k	2	2
	Waveform	10k & 50k	40	3
	Random-RBF	10k & 50k	40	6
Real	Connect-4	67557	42	3
	Covertype-Sorted	581,012	54	7
	Lung-cancer	32	56	3
	NslKdd99	125,973	41	2
	Pokerhand-1M	1,000,000	10	10
	WineRed	1599	11	9
	Usenet-2	1500	100	2
	Sensor	2,219,803	5	54

synthetic data sets and the last rows show the results over real data sets. Each cell in the tables presents the values reached by the methods. The result indicating improvements with respect to FASE are in highlighted bold (the winner) and italics. Note that higher values in accuracy indicate better performance whereas, for the run-time, the lower values are the better.

Regarding accuracy, the FASE's families methods performs better using NB as base classifier. With this base classifier, FASEB outperformed FASE in 4 out 8 synthetics data-sets. FASE outperformed each one out of all derived methods in 4 data-sets. FASEO, FASEO$_{wv}$ and FASEB$_{wv}$ methods outperformed FASE in 3 data-sets. On the other hand, using HT, FASEB outperformed FASE in 3 out 8 synthetics data-sets. FASE outperformed each one out of all derived methods in 3 data-sets. FASEO$_{wv}$ and FASEB$_{wv}$ methods outperformed FASE in 2 data-sets and FASEO outperformed FASE in 3 data-sets.

In general, FASEB, FASE, FASEB$_{wv2}$ and FASEO presented the best average results. With synthetic data, FASEB had better performance than the other variants followed by FASE, FASEO and FASEB$_{wv2}$. The data-set on which the developed variants performed better were those obtained from the Led generator, where the methods that use weighted voting to perform classification reached better behavior, especially FASEO$_{wv2}$ and FASEB$_{wv2}$. Concerning real data-sets, FASEO$_{wv2}$ and FASEB$_{wv2}$ performed equally or better than FASE in 5 out of 8 real data-sets using NB as base classifier. Similarly, FASEB improved or tied FASE in 5 out of the 8 real data-sets when HT was employed. In general, FASEO$_{wv2}$ and FASEB$_{wv2}$ are the best-ranked methods.

Considering run-time, the implemented variants of FASE had better performance than FASE, in almost all data-sets. In general, FASEB$_{wv3}$, FASEB$_{wv2}$, FASEB$_{wv1}$ and FASEO$_{wv3}$ presented the best average results. In particular, the same result was obtained in synthetic data-sets. FASEB performed slower than

FASE more frequently in real data-sets. FASEO$_{wv3}$, FASEB$_{wv3}$, FASEB$_{wv1}$ and FASEB$_{wv2}$ are the best-ranked methods. In conclusion, the variant of FASE with a meta-classifier demanded more run-time to perform the classification than the others with weighted voting. Particularly, FASEB$_{wv3}$ is the fastest among all proposed methods.

Table 2. Mean accuracies in percentage (%) with 95% confidence intervals in scenarios of abrupt and gradual concept drifts with artificial data-sets and real data-sets using NB and HT.

BC	DATA-SET	FO	FOwv1	FOwv2	FOwv3	FB	FBwv1	FBwv2	FBwv3	FASE
NB	LED-10k-gra	67,00	***67,77***	***67,79***	***67,51***	***67,21***	***67,79***	**67,81**	***67,51***	67,10
	Sine-10k-gra	***81,78***	81,52	81,52	81,35	**81,86**	81,60	81,59	81,35	81,71
	Waveform-10k-gra	77,62	77,56	77,53	77,36	77,62	77,58	77,55	77,36	**78,19**
	Random-10k-gra	30,90	30,81	30,79	30,92	30,94	30,84	30,82	30,85	**31,61**
	LED-10k-abr	***69,13***	***69,68***	**69,72**	***69,44***	***69,13***	***69,66***	***69,70***	***69,45***	68,64
	Sine-10k-abr	86,29	86,22	86,23	86,12	86,32	86,25	86,25	86,15	**86,42**
	Waveform-10k-abr	***78,63***	***78,53***	***78,50***	***78,38***	***79,00***	**78,65**	***78,53***	***78,50***	78,38
	Random-10k-abr	30,96	30,84	30,82	30,95	30,87	30,85	30,83	30,95	**31,58**
HT	LED-50k-gra	72,18	***72,52***	***72,57***	***72,49***	72,20	***72,52***	**72,58**	***72,50***	72,41
	Sine-50k-gra	90,80	90,20	90,21	89,84	90,84	90,21	90,22	89,84	**90,91**
	Waveform-50k-gra	81,45	80,54	80,70	80,31	81,42	80,61	80,79	80,38	**81,46**
	Random-50k-gra	***33,54***	32,55	32,64	32,74	**33,57**	32,53	32,67	32,71	33,38
	LED-50k-abr	72,52	***72,79***	***72,85***	***72,75***	72,52	***72,80***	**72,87**	***72,75***	72,73
	Sine-50k-abr	***91,98***	91,35	91,39	91,00	**92,01**	91,36	91,38	91,01	91,98
	Waveform-50k-abr	81,48	80,81	80,97	80,63	81,53	80,82	80,97	80,64	**81,59**
	Random-50k-abr	**33,64**	32,59	32,75	32,71	***33,60***	32,52	32,72	32,67	33,40
NB	Connect-4	74,47	**75,10**	**75,10**	***74,66***	74,48	***74,92***	***74,92***	74,64	74,66
	Covertype-Sorted	68,10	68,61	***69,55***	***69,15***	68,27	68,67	**69,74**	***69,35***	69,06
	Lung-cancer	**77,97**	65,57	66,82	65,57	**77,97**	65,57	66,82	65,57	**77,97**
	NslKdd99	**89,83**	89,79	89,79	89,75	**89,83**	89,79	89,79	89,75	89,81
	Pokerhand-1M	***50,10***	***50,10***	**50,11**	**50,11**	***50,10***	***50,09***	**50,11**	**50,11**	49,87
	WineRed	48,74	48,38	***52,86***	**54,09**	48,74	48,38	***52,86***	**54,09**	50,60
	Usenet-2	70,23	67,31	67,31	66,84	70,11	67,10	67,10	66,27	**72,86**
	Sensor	***86,41***	***89,36***	***88,06***	86,01	***87,15***	**89,86**	***88,30***	***87,00***	86,23
HT	Connect-4	***75,01***	74,89	**75,13**	74,50	***75,06***	74,78	***74,96***	74,44	74,94
	Covertype-Sorted	**74,08**	***72,13***	***72,71***	71,73	***73,70***	***72,46***	***73,28***	71,95	72,04
	Lung-cancer	**74,74**	67,29	66,46	67,29	**74,74**	67,29	66,46	67,29	**74,74**
	NslKdd99	98,62	98,35	98,47	98,41	98,62	98,35	98,47	98,41	**98,67**
	Pokerhand-1M	**54,40**	50,36	52,69	52,81	***54,30***	50,36	52,69	52,81	53,32
	WineRed	48,57	50,13	**54,41**	***54,25***	49,06	50,66	***54,24***	53,79	54,02
	Usenet-2	66,53	***68,07***	67,79	68,10	66,56	***68,07***	67,79	**68,10**	67,95
	Sensor	85,59	***89,39***	***88,43***	***86,71***	***86,06***	**89,55**	***88,84***	***87,71***	86,03

In order to conduct a statistical analysis of the derived methods regarding a control method (the original FASE), the *Friedman* test [16,17] and the Bonferroni-Dunn test [17,18] were applied. The tests were used with a significance level of 5%. The total of the experiment taken into consideration was 32, corresponding to the test performed by the base classifier (NB or HT) in synthetic and real data-sets.

Table 3. Mean of time in percentage (%) with 95% confidence intervals in scenarios of abrupt and gradual concept drifts with artificial data-sets and real data-sets using NB and HT.

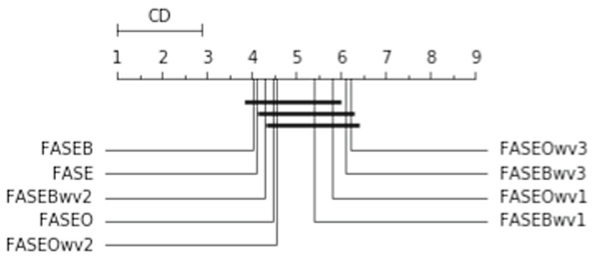
BC	DATA-SET	FO	FOwv1	FOwv2	FOwv3	FB	FBwv1	FBwv2	FBwv3	FASE
NB	LED-10k-gra	***2,67***	***2,47***	***2,34***	***2,45***	***2,65***	***2,30***	***2,33***	**2,27**	3,03
	Sine-10k-gra	***1,20***	***1,13***	***1,14***	***1,16***	***1,24***	***1,13***	***1,14***	**1,12**	1,42
	Waveform-10k-gra	***1,88***	***1,74***	***1,69***	***1,68***	***1,96***	**1,62**	***1,65***	***1,70***	2,25
	Random-10k-gra	***3,74***	***3,06***	***3,20***	***3,11***	***3,87***	***3,09***	***3,01***	**3,00**	4,40
	LED-10k-abr	***2,76***	***2,44***	***2,38***	***2,37***	***2,66***	***2,31***	***2,32***	**2,30**	2,92
	Sine-10k-abr	***1,23***	**1,12**	***1,15***	***1,14***	***1,26***	**1,12**	**1,12**	**1,12**	1,39
	Waveform-10k-abr	1,91	1,70	1,70	1,71	2,19	1,94	**1,63**	***1,66***	1,70
	Random-10k-abr	***3,81***	***3,09***	***3,26***	***3,21***	***3,87***	***3,16***	***3,03***	**2,98**	4,43
HT	LED-50k-gra	***16,39***	***15,97***	***15,81***	***14,87***	***17,32***	**14,41**	***15,14***	***15,16***	17,75
	Sine-50k-gra	***13,29***	***12,47***	***11,97***	***11,55***	***12,81***	***11,64***	***11,37***	**11,12**	13,58
	Waveform-50k-gra	***18,43***	***16,77***	***16,53***	***17,42***	***18,93***	**16,51**	***17,03***	***17,04***	19,16
	Random-50k-gra	***22,79***	***20,07***	***21,05***	***19,92***	***22,85***	***19,61***	**18,93**	***19,19***	24,94
	LED-50k-abr	***16,48***	***15,51***	***15,78***	***15,17***	***17,19***	**14,37**	***14,50***	***15,02***	17,65
	Sine-50k-abr	13,90	***12,64***	***12,45***	***11,77***	***12,88***	**11,54**	***11,67***	***11,59***	13,52
	Waveform-50k-abr	***17,52***	***16,12***	***16,24***	***16,80***	***18,32***	***16,48***	**16,05**	***16,12***	19,32
	Random-50k-abr	***22,86***	***20,10***	***19,90***	***19,68***	***22,28***	***19,78***	***19,32***	**18,56**	24,37
NB	Connect-4	***11,90***	**11,81**	***13,03***	***11,99***	***12,55***	***12,45***	15,88	***12,69***	13,58
	Covertype-Sorted	156,96	***139,68***	***132,32***	***128,72***	***140,94***	***124,66***	***122,60***	**120,41**	149,64
	Lung-cancer	**0,05**	0,06	0,06	**0,05**	0,06	0,06	0,06	**0,05**	0,06
	NslKdd99	***24,30***	***24,62***	30,45	***24,33***	***26,85***	***25,42***	***21,94***	**21,60**	27,96
	Pokerhand-1M	183,64	***129,78***	***125,88***	***126,00***	***163,13***	**123,44**	***126,20***	***130,43***	181,06
	WineRed	***0,56***	***0,53***	**0,45**	***0,51***	***0,56***	***0,52***	***0,54***	***0,49***	0,70
	Usenet-2	***0,89***	0,92	***0,91***	***0,86***	***0,86***	***0,85***	***0,90***	**0,82**	0,92
	Sensor	***1028,25***	***693,5***	***678,79***	***695,9***	***1042,65***	**635,06**	***644,64***	***654,65***	1153,36
HT	Connect-4	**21,87**	***22,60***	***23,61***	***22,21***	27,94	33,10	***26,91***	***23,55***	27,08
	Covertype-Sorted	372,31	269,34	278,12	267,18	333,73	262,40	268,59	275,65	245,78
	Lung-cancer	0,11	0,11	0,14	0,12	0,14	0,17	**0,10**	0,12	0,11
	NslKdd99	1256,79	1218,50	1107,60	1137,01	***1052,11***	**1013,71**	1155,79	1109,51	1069,80
	Pokerhand-1M	605,85	***472,22***	***427,66***	***435,17***	559,44	**421,31**	***433,76***	445,22	438,03
	WineRed	***0,93***	**0,88**	***0,92***	***0,90***	***1,01***	***0,94***	***0,98***	***1,00***	1,09
	Usenet-2	***1,51***	**1,22**	***1,37***	***1,38***	1,70	***1,56***	***1,35***	***1,40***	1,70
	Sensor	4697,27	***1085,96***	**1059,98**	***1076,79***	4547,49	***1093,21***	***1063,68***	***1078,59***	1384,75

Regarding accuracy, the best-ranked method was FASEB. FASEB and FASE are significantly better than FASEB$_{wv3}$ and FASEO$_{wv3}$ tacking into consideration all data-sets. With respect to the other methods the observed differences were not statistically significant. The same happens in synthetic data-sets. On the other hand, FASEO$_{wv2}$ is the best ranked in real data-set, but the methods do not present significant differences.

Concerning run-time, the best ranked algorithm was FASEB$_{wv3}$. Tacking into consideration all data-sets, FASE, FASEB and FASEO are the worst ranked and significantly less fast with respect to the others methods. Regarding synthetic data-sets FASE and FASEB are more time consuming methods, significantly less fast with respect to all methods (except FASEO). On the other hand, FASEO$_{wv3}$ and FASEB$_{wv3}$ are the best ranked and only they presents statistical differences respect to the FASEB and FASE in real data-sets.

Figure 1 showed the best measurements located in the first positions. Particularly, Figs. 1(a) show a comparison of the methods accuracies using the Bonferroni-Dunn test in synthetic and real data-sets. Figure 1(b) show a comparison of the methods run-time using the same test tests in all data-sets.

(a) Ranks_Accuracy



(b) Ranks_Time

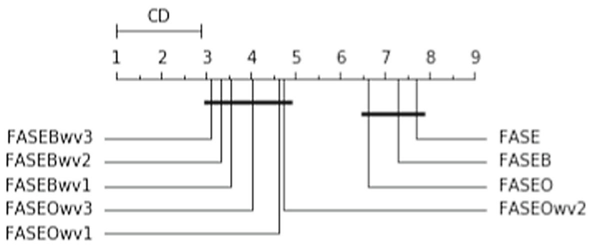


Fig. 1. Comparison of methods accuracies and time using the Bonferroni-Dunn tests with a 5% of significance level.

5 Application of the FASE Family Methods on the Sensor Data-Stream

A sensor, is a device that detects some physical stimulus (such as heat, light, sound, pressure, magnetism, or a particular motion) and responds usually with a transmitted signal resulting of impulse (as for measurement or operating a control) [6]. Normally, it is used to record that something is present or that there are changes in something [19]. Hence the importance of validating the behavior of these methods in data stream from sensors, because it represents a high complexity problem likely to presenting concept drift.

Particularly, the present research compared the performance of the FASEB, FASEO families and FASE method on the Sensor data-set. Sensor Stream [20] contains information collected from 54 sensors deployed in the Intel Berkeley Research Lab (temperature, humidity, light, and sensor voltage). It contains consecutive information recorded over a 2 months period, with one reading every 1–3 minutes. The sensor ID is the target attribute, which must be identified based on the sensor data and the corresponding recording time. This data-set is constituted of 2,219,803 instances, 5 attributes, and 54 classes. This is an interesting and intriguing data-set because, in addition to being much larger than the others, produces considerable variations in the accuracy performance of the methods.

Tables 2 and 3 presented the performance of the evaluated methods on the Sensor data-set among others. In addition, Fig.2 shows the results achievement on Sensor data-set using NB and HT.

(a) Accuracy (b) Run-time

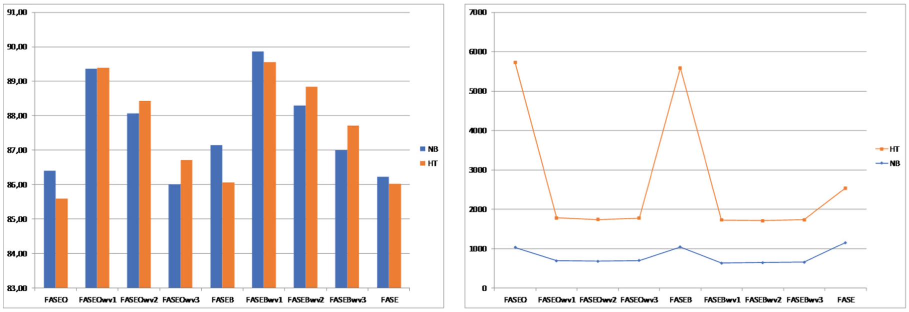


Fig. 2. Comparison of methods accuracies and run-time using NB and HT on Sensor data-stream.

As shown in the Fig. 2, the method that had the best performance on Sensor data-stream using NB and HT was FASEB$_{wv1}$ followed by FASEO$_{wv1}$ regarding the accuracy achievement. The worst result was by FASEO with HT. In general, this method and FASE had a worse rank. On the other hand, FASEB$_{wv2}$ was the best-ranked method regarding run-time. In particular, with NB the best result was achieved by FASEB$_{wv1}$, and FASEO$_{wv2}$ using HT. In general, FASE, FASEO and FASEB were the least fast.

Figure 3 show the performance of FASE derived algorithms with respect to the original algorithm using NB, when processing a fragment of 100,000 instances of the Sensor data-set. As it is possible to see, in all the methods the performance over time during the processing of the instances is more stable both in the FASEB and FASEO families regarding FASE, except in the FASEO$_{wv3}$ method in NB. Frequently, it can be seen that the accuracy values in all cases in the FASE method fall down more than in the methods derived from it.

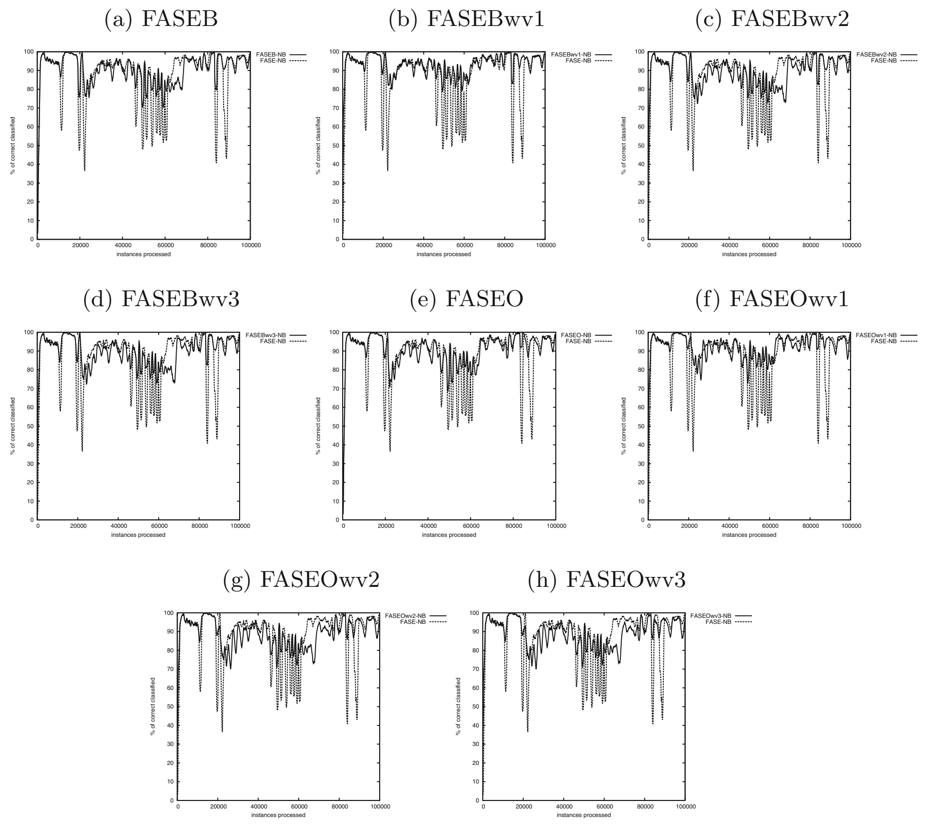


Fig. 3. Performance evaluation of the methods with respect to FASE regarding the accuracy over time on the Sensor data-stream using NB.

6 Conclusions

This work proposed FASEB and FASEO families of algorithms, a total of eight ensemble methods for operation in concept drift scenarios with full access to labeled classes. The algorithms are variants of the FASE ensemble. The main difference of FASEB and FASEO families as compared to FASE is the update strategy employed by the algorithms. While FASE uses adaptive classifiers to keep the ensemble updated, the implemented algorithms have in common a parallel ensemble formed by alternative classifiers, activated and set to be trained when one of the classifiers in the main ensemble issues a warning.

When a Concept Drift is detected, algorithms in the FASEB family boosts the alternative classifier with the greatest accuracy. Algorithms in the FASEO family, instead, promote the oldest active alternative classifier. The proposed variants were compared to FASE through similar parametrization and same testing conditions in order to accordingly evaluate their performance, using HT and NB as base learners. In terms of accuracy, FASEB obtained the best results in most of the tested data-sets using HT and NB, but it was noticed a very close approximation of FASEB$_{wv2}$ as compared to those of FASEB.

In addition, the performance of the methods studied in the Sensor data-stream was analyzed because it is a data-set of high complexity, large size, and representative of a real problem prone to presenting concept drift. FASEB$_{wv1}$ achieved the best result regarding accuracy. FASEB$_{wv2}$ was among the first 3 results in accuracy and, at the same time, it was the most rapid.

The statistical significance of the results provided by the experiments were evaluated using the non-parametric Friedman test together with the Bonferroni-Dunn test. Those tests confirmed the proposed algorithms were often significantly better than FASE with respect to run-time. In particular, versions FASEB$_{wv2}$ and FASEO$_{wv2}$, while not showing significant accuracy losses, were noticeably faster than the original FASE algorithm. This can be very useful in contexts that require quick access to partial information, a high level of accuracy is still needed, but a very fast decision has to be made.

References

1. Gama, J., Gaber, M.: Learning From Data Streams: Processing Techniques in Sensor Network. Springer, Heidelberg (2007). https://doi.org/10.1007/3-540-73679-4
2. Aggarwal, C.: Data Classification: Algorithms and Applications (2014)
3. Frías-Blanco, I.: Nuevos métodos para el aprendizaje en flujos de datos no estacionarios. Granada University (2014)
4. Ortiz-Diaz, A., et al.: Fast adapting ensemble: a new algorithm for mining data streams with concept drift. Sci. World J. **2015**, 1–14 (2015)
5. Barros, R., Silas, G.: An overview and comprehensive comparison of ensembles for concept drift. Inf. Fusion Number C **52**, 213–244 (2019)
6. Stevenson, A.: Oxford Dictionary of English. Oxford University Press, Oxford (2010)
7. Frías-Blanco, I., Verdecia-Cabrera, A., Ortiz-Diaz, A., Carvalho, A.: Fast adaptive stacking of ensembles. In: Proceedings of the 31st Annual ACM Symposium on Applied Computing, pp. 929–934. ACM (2016)
8. Frías-Blanco, I., et al.: Online and non-parametric drift detection methods based on Hoeffding's bounds. Trans. Knowl. Data Eng. **27**(3), 810–823 (2015)
9. Oza, N., Russell, S.: Online bagging and boosting. Artif. Intell. Stat. 105–112 (2001). https://doi.org/10.1109/ICSMC.2005.1571498
10. Hoeffding, W.: Probability inequalities for sums of bounded random variables. J. Am. Stat. Assoc. **58**, 13–30 (1968)
11. Ortiz-Díaz, A.: Algoritmo multiclasificador con aprendizaje incremental al que manipula cambios de conceptos. Universidad de Granada, España (2014)
12. Mariño, L., Hidalgo, J., Barros, R., Vasconcelos, G.: Improving fast adaptive stacking of ensembles. In: International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE (2019)
13. Menahem, E., Rokach, L., Elovici, Y.: Combining one-class classifiers via meta learning. In: Proceedings of the 22nd ACM International Conference on Information & Knowledge Management, pp. 2435–2440. ACM (2013)
14. Shen, H., Lin, Y., Tian, Q., Xu, K., Jiao, J.: A comparison of multiple classifier combinations using different voting-weights for remote sensing image classification. Int. J. Rem. Sens. **39**(11), 3705–3722 (2018)

15. Song, G., Ye, Y., Zhang, H., Xu, X., Lau, R., Liu, F.: Dynamic clustering forest: an ensemble framework to efficiently classify textual data stream with concept drift. Inf. Sci. **357**, 125–143 (2016)
16. Friedman, M.: The use of ranks to avoid the assumption of normality implicit in the analysis of variance. J. Am. Stat. Assoc. **32**(200), 675–701 (1937)
17. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**, 1–30 (2006). JMLR.org
18. Dunn, O.: Multiple comparisons among means. J. Am. Stat. Assoc. **56**(293), 52–64 (1961)
19. Dictionary Cambridge: Cambridge advanced learner's dictionary. PONS-Worterbucher, Klett Ernst Verlag GmbH (2008)
20. Qun, Z., Xuegang, H., Yuhong, Z., Peipei, L., Xindong, W.: A double-window-based classification algorithm for concept drifting data streams, pp. 639–644. IEEE (2010)