Future Generation Computer Systems 95 (2019) 198–211

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Learning concept drift with ensembles of optimum-path forest-based classifiers

Adriana Sayuri Iwashita [a], Victor Hugo C. de Albuquerque [c,*], João Paulo Papa [b]

[a] Federal University of São Carlos, São Carlos, São Paulo, Brazil
[b] São Paulo State University, Bauru, São Paulo, Brazil
[c] Universidade de Fortaleza, Fortaleza, Ceará, Brazil

HIGHLIGHTS

- To introduce the Optimum-Path Forest (OPF) classifier to the context of drift detection.
- To evaluate ensembles of OPF classifier for novelty detection in data streams.
- To conduct an extensive experimental evaluation considering three distinct concept drift paradigms.

ARTICLE INFO

Article history:
Received 20 September 2018
Received in revised form 29 December 2018
Accepted 1 January 2019
Available online 9 January 2019

Keywords:
Optimum-path forest
Concept drift
Ensemble learning

ABSTRACT

Concept drift methods learn patterns in non-stationary environments. Although such behavior is usually not expected in traditional classification problems, in real-world scenarios one can face them very much easier. In such a context, classifiers can be fooled and their effectiveness affected as well. Some examples include theft detection in energy distribution systems, where the consumer's behavior may change suddenly or smoothly, or even churn prediction in mobile companies. In this paper, we introduce the Optimum-Path Forest (OPF) classifier in the context of concept drift, using decisions for concept drift handling based on a committee of OPF classifiers. We consider three distinct perspectives (three rounds of experiments with variations of streaming managements) over publics datasets, being the results compared to the ones obtained by standard OPF. We consider OPF ensemble suitable to work under these dynamic scenarios since its recognition rates were considerably better when compared to traditional OPF.

1. Introduction

Learning algorithms are usually designed to handle stationary environments only. However, real-world applications may have dynamic behaviors [1]. In this context, *Concept Drift* was addressed to a situation of non-stationary environments in which the target concept changes over time, i.e., when the training and application data mismatch [2]. Several domains can contain concept drift-related problems, including monitoring and control, management and strategic planning, as well as personal assistance, among others [3–7].

Methods for handling and detecting concept drift have been proposed in the literature. Some techniques deal with concept drift by adding or discarding samples according to some criterion. Klinkenberg and Joachims [8], for instance, employed Support Vector (SVM) to cope with the concept drift problem. Irrelevant data are discarded in order to minimize the generalization error with a sliding window with four data management approaches: "full memory", "no memory", "window of fixed size" and "window of adaptive size". ZareMoodi et al. [9] proposed the LOCE (LOcal Classifier Ensemble) algorithm in which each class has an ensemble of classifiers updated by their own metrics and by a pruning phase. Stanley [10] proposed a committee of hypotheses (Concept Drift Committee) that classifies instances with weighted voting in which a new member replaces an older one if its vote drops below a threshold. Kolter and Maloof [11] proposed a weighted ensemble method based on global performance that adds or delete weighted experts to deal with such changes and weight learners based on their performance.

Among drift detection techniques, we can cite those that have two thresholds for detecting concept drift: Gama et al. [1] proposed the DDM (Drift Detection Method), a drift detector that works with probability distribution using the online error-rate. It defines two levels: the warning and the drift level. If the error reaches respectively the warning level at instance i_w and the drift level at instance i_d, it is considered the occurrence of a drift, and the method starts the training process with data from i_w. According to the authors,

* Corresponding author.
E-mail addresses: sayuri.iwa@gmail.com (A.S. Iwashita), victor.albuquerque@unifor.br (V.H.C. de Albuquerque), papa@fc.unesp.br (J.P. Papa).

https://doi.org/10.1016/j.future.2019.01.005
0167-739X/

it can operate with online and incremental methods, and also as a wrapper to batch classifiers [1]. García et al. [12] presented EDDM (Early Drift Detection Method), a drift detector that works with distance-error-rate (estimated distribution of the distance among classification errors) rather than errors classifications. EDDM has two thresholds: the Warning level – exceeded this level, instances are kept; and the Drift level – the method considers concept has drifted and builds a new model with data from warning level. EDDM starts to search for a concept drift after 30 errors occurred. Otherwise, if the system has a rise of similarity after warning threshold, the instances are deleted and the system reverts to an "in-control level" [12]. Barros et al. [13] proposed the RDDM (Reactive Drift Detection Method) based on DDM that discards older instances of very long concepts aiming to detect drifts earlier to improve the final accuracy. Li et al. [14] proposed the EDTC (Ensemble Decision Trees for Concept drifting data streams), an incremental algorithm that defines cut-points in the growing tree with three different random feature selection approaches. When an instance arrives, each growing node split features at random to avoid producing unnecessary branches. EDTC employs two thresholds and uses local data distributions to detect drift as well. Ross et al. [15] presented the ECDD (EWMA for Concept Drift Detection), a drift detector mechanism based on the exponentially weighted moving average chart. The method employs classification error stream and according to authors does not require any data to be stored in memory.

Some drift detectors select instances, such as: Widmer and Kubat [16] proposed the FLORA framework (FLOating Rough Approximation), a family of algorithms that handle concept drift with groups of descriptors and a variable-sized instance window to select the descriptors sets. Liu et al. [17] proposed a drift detector in a sensor network domain based on angle optimized global embedding (AOGE) and Principal Component Analysis (PCA). PCA and AOGE analyze the projection variance and the projection angle, respectively, being further used to detect drift. Bifet and Gavalda [18] presented the ADWIN2 (ADaptive WINdowing) algorithm, an improved version of ADWIN algorithm. ADWIN2 has a variable sized window: it grows or shrinks when no change or concept drift is detected, respectively. This supervised method detects drifts using the average of the elements in the window [18]. Xu and Wang [19] proposed the DELM (Dynamic Extreme Learning Machine) algorithm that uses Extreme Learning Machine (ELM) to detect drifts. Roughly speaking, the main idea is to employ a double hidden layer structure to train and improve the performance: when an alert of drift is issued, additional hidden layer nodes are included in the neural network; once the drift is detected, a new classifier replaces an older classifier with lower performance. Lobo et al. [20] introduced the well-known Spiking Neural Networks in the context of online learning in data streams. The main idea of their work was to limit the size of the neuron repository, as well as to make use of data reduction techniques to take advantage of the compressed neuron learning ability.

Some drift detectors are ensemble learnings: Zhang et al. [21] proposed a three-layered drift detection technique in a text data stream domain, where each layer denotes, respectively, the layer of label space, the layer of feature space, and the layer of the mapping relationships between labels and features. The method can measure changes in each layer to detect different types of drift. Lobo et al. [22] proposed DRED, which is based on multi-objective optimization for labeling data that are generated synthetically. The authors highlighted the importance of using adaptable ensembles that can quickly deal with changes in the data stream as soon as they have been detected.

Other drift detection techniques can deal with imbalanced data, such as: Mirza et al. [23] proposed the ESOS-ELM (Ensemble of Subset Online Sequential Extreme Learning Machine), a drift detector that can deal with class imbalance. The main ensemble represents the short-term memory, in which each network is trained with a balanced subset of the original imbalanced data. Arabmakki and Kantardzic [24] proposed the RLS-SOM (Reduced labeled Samples-Self Organizing Map) framework for imbalanced data stream. An ensemble classifies with DWM (Dynamic Weighted Majority) and retrains a new model using partial labeled samples (the method also selects majority and minority class instances) when a drift is detected. If an individual model has a higher performance than the ensemble's performance, it is chosen instead of the others. Lobo et al. [25] proposed a probabilistic approach to deal with imbalanced data streams in the context of concept drift. In a nutshell, the authors suggested finding the most informative samples from the past learners, i.e., the ones trained over recent data, to build a new predictive model to classify incoming information from the data stream.

Finally, we can cite some drift detectors for unlabeled streams or with clustering technique: Sethi and Kantardzic [26] proposed the MD3 (Margin Density Drift Detection) to detect drift from an unlabeled stream. A number of instances in a classifier's region of uncertainty (margin) is used as a metric to detect drift. If a deviation in the margin density occurs, the classifier needs a set of labeled instances to be retrained. Silva et al. [27] proposed the FEAC-Stream (Fast Evolutionary Algorithm for Clustering data Streams) algorithm, which is based on k-means clustering with k automatically estimated from the stream. FEAC-Stream uses the Page-Hinkley Test to detect a decrease in quality of the clusters to start the re-estimation of k with an evolutionary algorithm.

Some years ago, Papa et al. [28–30] proposed the Optimum Path Forest (OPF), a pattern recognition framework that partition instances into a graph of optimum-path trees (OPTs). Each OPT is rooted at key samples (prototypes) that compete among themselves in order to conquer the remaining samples. The authors showed OPF can be so effective as SVM, but faster for training. However, as far as we are concerned, OPF-based classifiers have been poorly evaluated in the context of concept drift domains [31]. Therefore, the main contribution of this paper is to propose an ensemble-based approach composed of a committee of OPF classifiers to cope with the problem of concept drift handling. We considered three different perspectives with variations of streaming management and classifiers in public datasets, being the results compared to the ones obtained by standard OPF and OPF with simple concept drift handling (no-memory, full-memory, and windowed-based approach).

The remainder of this paper is organized as follows. Sections 2 and 3 present the main theoretical background regarding concept drift and OPF classifier. Section 4 describes datasets and methodology, Section 5 presents experimental analysis. Finally, Section 6 states conclusions and future works.

2. Concept drift

Concept drift occurs when a concept target changes over time. Consider two target concepts A and B, and an ordered sequence of instances $\mathcal{I} = \{i_1, i_2, \ldots, i_n\}$. Before instance i_d, the target concept does not change and is stable at A. After some instances Δx, the concept is stable in a different concept, named B. The concept is *drifting* between instances i_{d+1} and $i_d + \Delta x$, replacing concept target A for concept B [10]. Depending on the speed of the drift (Δx), a change may be declared as gradual or abrupt: in gradual drifts occurs a slow change between two concepts (i.e., concept shift), whereas in abrupt drifts occurs a sudden change (i.e., concept drift).

Some authors handle concept drift in one of the three following ways [32]: (a) window-based approaches, (b) weight-based approach and (c) ensemble of classifiers. Window-based approach selects instances within a fixed or dynamic sliding window; whereas

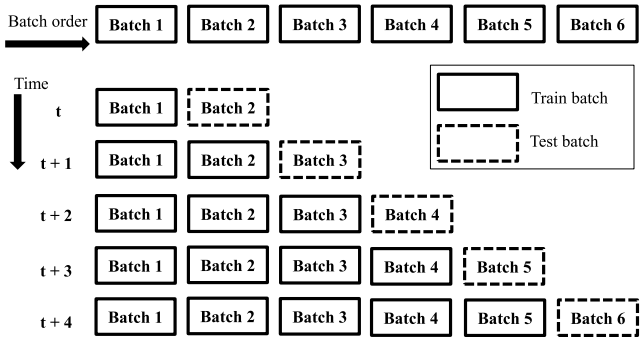

Fig. 1. *Full-memory* approach.

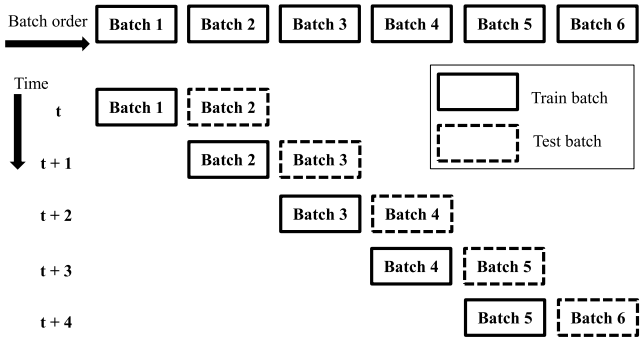

Fig. 2. *No-memory* approach.

a weight-based approach weights instances and discards older ones based on their weights. Yet, ensemble classifiers present several classifiers results and combine them to define a final classification [32].

The number of instances considered at training phase can characterize concept drift handling algorithms: the online approach updates the classifier after receiving an instance; whereas the batch approach waits to receive plenty of instances to start the learning [33]. The latter learning approach usually receives streams of data and split them into batches. Some methods to deal with a stream of batches are listed below [8]:

1. Full-memory: the learner uses all previously training instances (batches), i.e., it cannot "forget" old examples (Fig. 1).
2. No-memory: uses only the most recent batch for training purposes (Fig. 2).
3. Window-of-fixed size n: uses the n most recent batches, i.e., a sliding window of size 3 is used to observe samples (Fig. 3).

Žliobaitė [2] also surveyed applications where the concept drift problem is presented and divided them into four types: (i) monitoring and control, often employs unsupervised learning and abnormal behavior detection, includes detection of adversary activities on the web, telecommunications, computer networks, financial transactions; (ii) personal assistance and information applications, include recommendation systems, categorization, and organization of textual information, customer profiling for marketing; (iii) decision making, like diagnostics, evaluation of creditworthiness; and (iv) artificial intelligence, embraces a spectrum of moving and stationary systems that interact with non stationary environments, e.g., robots, mobile vehicles, and smart household appliances.

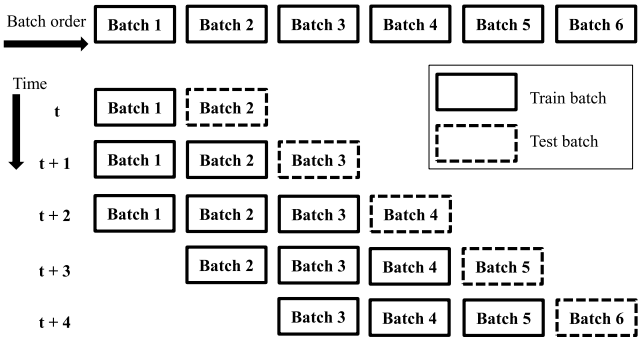

Fig. 3. *Window*-of-size 3 approach.

3. Optimum path forest

The OPF classifier comprises two learning procedures: a supervised and an unsupervised one. The supervised classifier has two approaches: one that uses a complete graph [28,29] and another that uses a k-nn graph [30,34,35], being the former the most used and widespread, and also used in this paper. Supervised OPF with complete graph has no parameters, whereas OPF with k-nn graph has a user-defined k_{max} parameter: the algorithm calculates a k value which k maximizes some criterion within the k_{max} value.

Regarding OPF with complete graph, let $\mathcal{Z} = \mathcal{Z}_1 \cup \mathcal{Z}_2$ be a labeled dataset, where $\mathcal{Z}_1$ and $\mathcal{Z}_2$ stand for the training and test sets, respectively. Let $\lambda(\mathbf{s})$ be the function that associates the correct label to any sample $\mathbf{s} \in \mathcal{Z}_1 \cup \mathcal{Z}_2$, such that $\lambda(\mathbf{s}) \in \{1, 2, \ldots, c\}$ and $\mathbf{s} \in \Re^n$. Let $\mathcal{S} \in \mathcal{Z}_1$ be the set of prototype samples (i.e., the most important samples that better represent the classes), and $d(\mathbf{s}, \mathbf{t})$ the distance between two samples $\mathbf{s}$ and $\mathbf{t}$. The problem consists in using S, d and $\mathcal{Z}_1$ to project an optimal classifier that can predict the correct label $\lambda(\mathbf{s})$ of any sample $\mathbf{s} \in \mathcal{Z}_2$.

Roughly speaking, the OPF classifier employs a graph partition task, where each node is encoded by a sample that is connected to others by means of a predefined adjacency relation. The partition process is ruled by a competition process among prototype samples, which try to conquer the remaining ones offering to them optimum-path costs. When a sample is conquered, it receives the label of its conqueror together with an updated cost. At the final of the process, we have a collection of optimum-path trees, which are rooted at each prototype.

Let $(\mathcal{Z}_1, \mathcal{A})$ be a complete graph in which nodes are samples in $\mathcal{Z}_1$, and any pair of samples defines an edge in $\mathcal{A}$ (i.e., a complete graph). Also, the edges are weighted by the distance d among their corresponding nodes (Fig. 4a). A path in $\mathcal{Z}_1$ is a sequence of samples $\pi = \langle \mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_k \rangle$, where $(\mathbf{s}_i, \mathbf{s}_{i+1}) \in A$ for $1 \leq i \leq k - 1$. Additionally, a path is said to be trivial when $\pi = \langle \mathbf{s}_1 \rangle$.

A given path-cost function $f(\cdot)$ is associated to each path π, being denoted by $f(\pi)$, and a given path π' is said to be optimum if $f(\pi') \leq f(\tau)$ for any path τ, where π' and τ end at the same sample $\mathbf{s}$, regardless of their origin. We also denote $\pi \cdot (\mathbf{s}, \mathbf{t})$ the concatenation of the path π ending at $\mathbf{s}$ and the edge $(\mathbf{s}, \mathbf{t}) \in \mathcal{A}$.

The OPF algorithm can be used with any smooth path-cost function, which can combine samples with similar properties [36]. Papa et al. [28] opted to use the f_{max} cost function due to its theoretical properties to estimate optimum prototypes, which can be defined as follows:

$$f_{max}(\langle \mathbf{s} \rangle) = \begin{cases} 0 & \text{if } \mathbf{s} \in \mathcal{S}, \\ +\infty & \text{otherwise} \end{cases}$$

$$f_{max}(\pi \cdot (\mathbf{s}, \mathbf{t})) = \max\{f_{max}(\pi), d(\mathbf{s}, \mathbf{t})\}. \quad (1)$$

Notice $f_{max}(\pi)$ computes the maximum distance among adjacent samples in π, when π is not a trivial path.

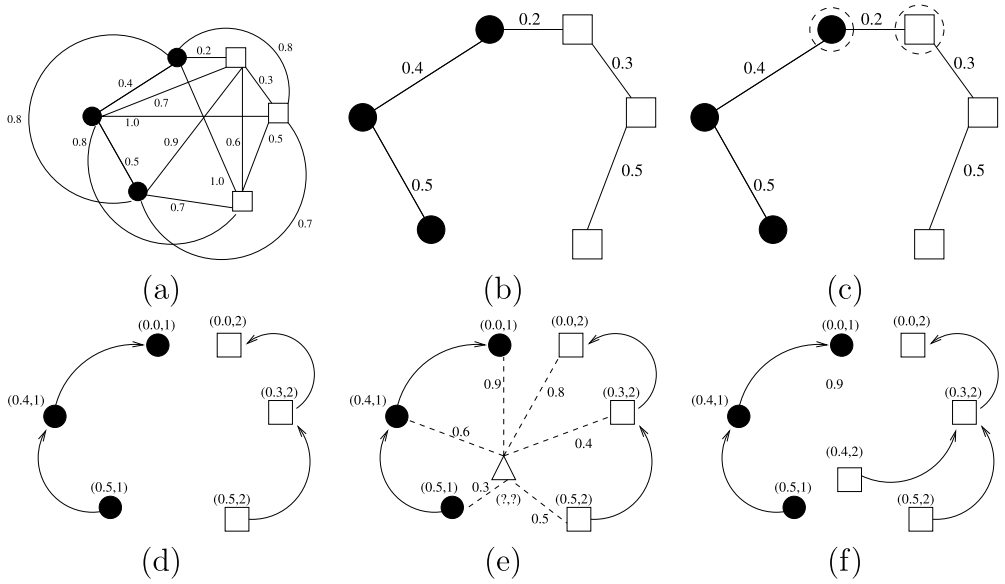

Fig. 4. OPF training step: (a) Complete and weighted graph representing the training set and its (b) MST, (c) prototypes highlighted and (d) resulting optimum-path forest. OPF classification step: (e) a testing sample (triangle) is added into the graph and connected to all training samples, and (f) the testing sample is conquered by the sample that offered the lowest cost.

The OPF comprises a training and a classification step, being the former in charge of building the optimum-path forest over the training set using f_{max} and $\mathcal{S}$, and the classification phase simply computes the training sample that will conquer that specific testing node (details below). Papa et al. [28] proposed to compose $\mathcal{S}$ with the nearest nodes from different classes in $\mathcal{Z}_1$, since such samples fall in the boundary of the classes, thus being informative enough to the learning process. In order to find them, one just needs to compute a minimum spanning tree in $\mathcal{Z}_1$ (Fig. 4b), and then select the connected samples with different labels as the prototype nodes (Fig. 4c).

Roughly speaking, the OPF training algorithm associates an optimal path $P^*(\mathbf{s})$ from $\mathcal{S}$ to all samples $\mathbf{s} \in \mathcal{Z}_1$, thus building an optimum path forest P (a function without cycles which associates to all $\mathbf{s} \in \mathcal{Z}_1$ its predecessor $P(\mathbf{s})$ in $P^*(\mathbf{s})$, or assigns *nil* when $\mathbf{s} \in \mathcal{S}$). Let $R(\mathbf{s}) \in \mathcal{S}$ be the root of $P^*(\mathbf{s})$ that can be achieved using $P(\mathbf{s})$. The OPF algorithm computes, for each $\mathbf{s} \in \mathcal{Z}_1$, the cost $C(\mathbf{s})$ of $P^*(\mathbf{s})$, the label $L(\mathbf{s}) = \lambda(R(\mathbf{s}))$, and its predecessor $P(\mathbf{s})$.

The classification step is straightforward, i.e., given a test sample $\mathbf{t} \in \mathcal{Z}_2$, we connect it to all training nodes (Fig. 4e) of the optimum-path forest generated in the training phase (Fig. 4d), and we evaluate which node $\mathbf{p} \in \mathcal{Z}_1$ minimizes the equation:

$$C(\mathbf{t}) = \min\{\max\{C(\mathbf{p}), d(\mathbf{p}, \mathbf{t})\}\}, \forall \mathbf{p} \in \mathcal{Z}_1. \quad (2)$$

Thus, the node $\mathbf{p} \in \mathcal{Z}_1$ that minimizes $C(\mathbf{t})$ will be the one that conquer $\mathbf{t}$ (Fig. 4f).

4. Proposed approach

In this section, we evaluated three different ensemble-based approaches in three different scenarios to address the problem of concept drift with the data management described in Section 2: "full memory" (OPF-fullmemory), "no memory" (OPF-nomemory), and "window-of-fixed-size three" (OPF-window3). Also, the experiments used real-world and synthetics datasets, being the latter of great importance, since they shape an environment in which we know that concept drift really occurs, as well as which kind of change is occurring (i.e., gradual or abrupt). Although ensembles of OPF-based classifiers have been evaluated before [37–39], they were not considered in the context of concept drift. Table 1 presents the main information concerning the synthetic datasets used in this work.

Table 1
Synthetic Datasets.

Dataset	# of samples	Drift time	# of features
Hyperplane	90,000	at every 10,000 samples	10
Usenet 1	1,500	at every 300 samples	99
Usenet 2	1,500	at every 300 samples	99
SEA	60,000	at every 15,000 samples	3

Table 2
Real-world Datasets.

Dataset	# of samples	# of features
Forest Covertype	581,012	54
Electricity	45,312	8
Poker Hand	829,201	10

In regard to the real-world dataset, we used the following (Table 2):

In order to evaluate the techniques, we divided each dataset into 30 batches (equally) aiming to simulate a stream of data. For the sake of explanation, consider SEA dataset that contains 60,000 samples, which means each batch comprises 2,000 samples. Since the concept drifts after every 15,000 samples (Table 1), we have a change in the stream behavior every 7.5 batches.

Therefore, the main idea of this paper is to evaluate whether an ensemble version of OPF is robust enough to handle such situations or not. Additionally, naïve OPF (i.e., without support to handle concept drift) is compared against OPF-nomemory, OPF-fullmemory, and OPF-window3 (of 3 batches). With respect to the OPF implementation, we employed LibOPF.[1]

Additionally, we considered the accuracy measure proposed by Papa et al. [28] that takes into account unbalanced datasets. The accuracy *Acc* is measured as:

$$Acc = \frac{2c - \sum_{i=1}^{c} E(i)}{2c} = 1 - \frac{\sum_{i=1}^{c} E(i)}{2c}, \quad (3)$$

[1] https://github.com/jppbsi/LibOPF.

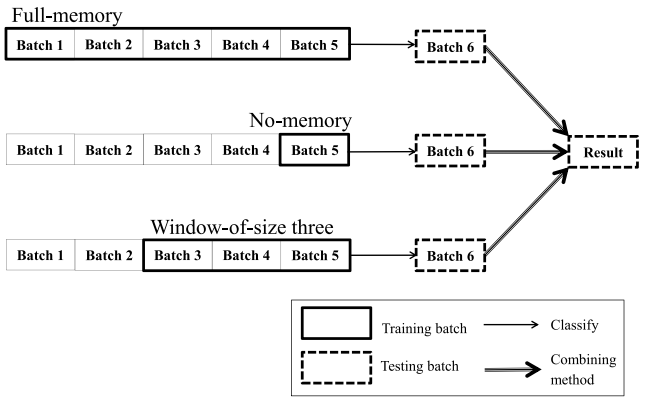

Fig. 5. Pictorial example of the approach named as "Experiment 1".

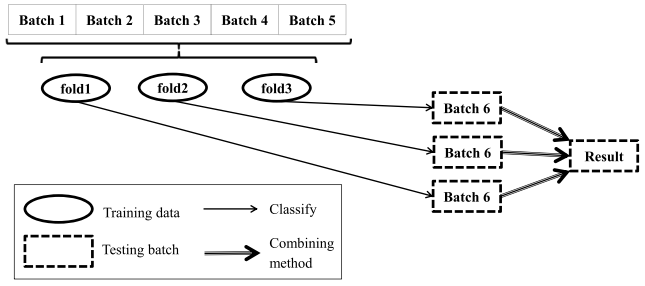

Fig. 6. Experiment 2 with the full memory approach.

being $N(i)$, $i = 1, 2, \ldots, c$, the number of samples in the dataset $\mathcal{Z}$ from class i, and $E(i)$ the partial sum error of class i.

The OPF ensemble employs three base classifiers and combines their results in three different ways (three voting mechanisms):

- Combined: the classification result is obtained by the most voted result among base classifier outputs, in which each learner has the same weight.
- Weighted: each base classifier receives a different weight for its voting relevance based on its previous accuracy classification.
- Major: the base classifier having the greatest accuracy in the previous batch receives an additional weight.

With the voting mechanisms described previously, we designed three rounds of experiments with variations of streaming managements, as described below:

- Experiment 1: it employs OPF-fullmemory, OPF-nomemory and OPF-window3 as base classifiers (Fig. 5), in which each approach handles the stream of batches as described in Section 2 with the OPF classifier. The results of each approach are further combined using the "combined", "weighted" and "major" approaches.
- Experiment 2: it partitions the stream of batches into three different portions of training data for each base classifier, hereinafter called "fold1", "fold2", and "fold3", considering the data management described in Section 2. In regard to the fullmemory (Fig. 6) and nomemory (Fig. 7) experiments, since one has one batch only, it is partitioned into three distinct subsets for further training one OPF classifier on each subset. With respect to the window3 (Fig. 8) approach, all three batches are merged into only one and further partitioned into three subsets. The results of fold1, fold2 and fold3 are further combined using the "combined", "weighted" and "major" approaches.

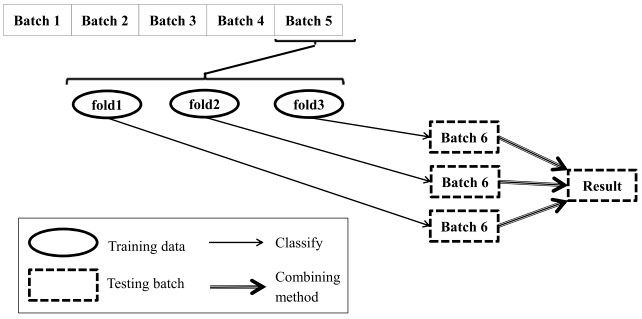

Fig. 7. Experiment 2 with the non-memory approach.

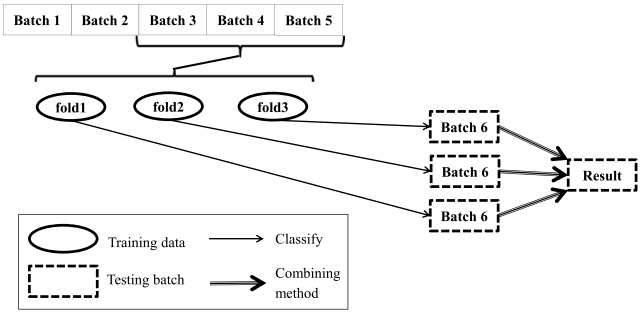

Fig. 8. Experiment 2 with the window-of-fixed-size three.

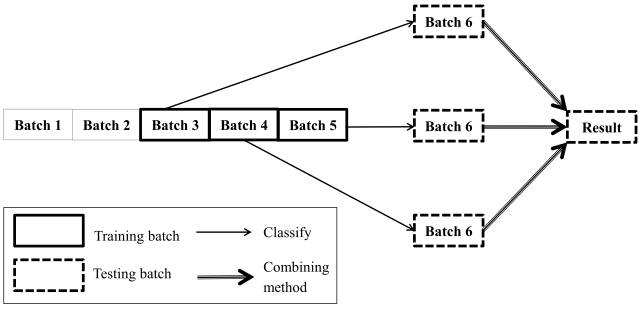

Fig. 9. Experiment 3.

- Experiment 3: one classifier is trained for each batch. The ensemble considers the last three models for each new batch classification procedure, similarly to the window-3 management (Fig. 9). The results are combined using the "combined" and "weighted" approaches.

With respect to the weight values, we adopted the following methodology[2]: concerning Experiments 1 and 2, the classifiers are weighted according to their accuracies, i.e., the approach with the highest accuracy receives the weight as of 0.8, followed by the second best one with weight as of 0.5, and the last one with weight as of 0.3. In regard to Experiment 3, the most recent batch received the highest weight, i.e., suppose we are classifying the batch i, then the training batch $i-1$ has a weight as of 0.4, batch $i-2$ has a weight as of 0.35, and finally batch $i-3$ has a weight as of 0.25. The experiments were performed on a computer with Intel®Core(TM) i5-4690 processor, Z97M-PLUS/BR motherboard, NVIDIA Corporation GM107 [GeForce GTX 750] graphics processing unit, and Corsair DDR3 8GB 1600MHz RAM.

[2] Notice that the weights were chosen empirically.

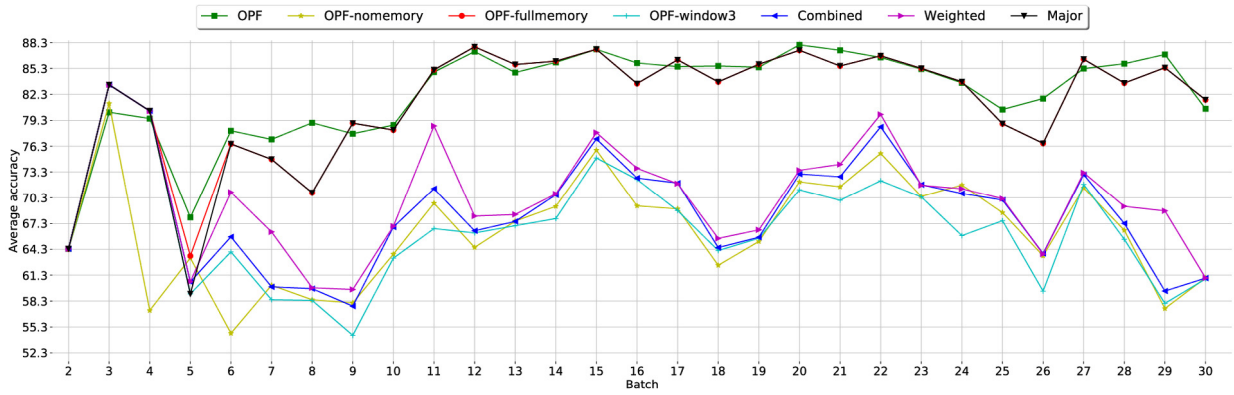

Fig. 10. Experimental results concerning Covtype dataset with respect to Experiment 1.

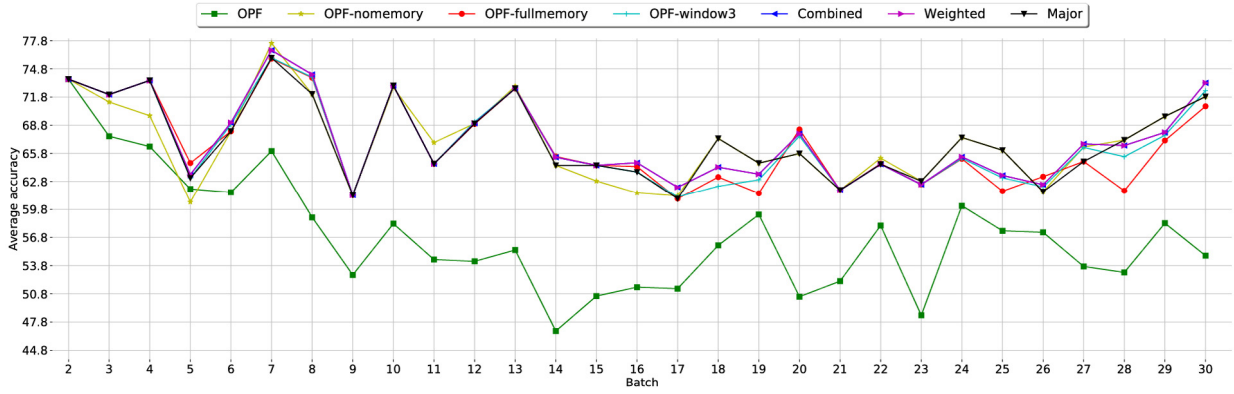

Fig. 11. Experimental results concerning Electricity dataset with respect to Experiment 1.

5. Experimental results

In this section, we evaluated the three experimental approaches described in Section 4.

5.1. Experiment 1

As aforementioned, the idea of this experiment is to consider all outputs given by full memory, non-memory and window-of-size three approaches. Fig. 10 depicts the results concerning Covtype dataset. Clearly, one can observe that standard OPF, OPF-fullmemory and the proposed OPF with majority voting ("Major") obtained the best results. Both OPF-fullmemory and OPF with majority voting were consistently similar to each other, which reflects the fact that OPF-fullmemory was also the best approach among the ones used to handle concept drift.

Fig. 11 depicts the results concerning Electricity dataset. OPF with concept drift handling versions (OPF-fullmemory, OPF-nomemory, and OPF-window3) and the ensemble-based versions obtained higher accuracy when compared to traditional OPF. Only in batch #5, the traditional OPF has been more accurate than the OPF-nomemory.

The combined and weighted versions obtained a similar behavior, whereas the major version has a better performance in some batches. With respect to the previous dataset shown in Fig. 10, one can now realize the importance in handling concept drift, since in that results standard OPF (i.e., with no concept drift handling) showed very much good results. The main idea in using such a dataset where standard OPF can outperform some approaches that can handle concept drift is just to show the proposed approach can obtain best results in any situation, i.e., where one does have or does not have a strong concept drift situation (usually called "concept shift").

Fig. 12 depicts the results concerning Hyperplane dataset. The combined and weighted version obtained similar performances. The combined version has a drop of performance in batches #11 and #21, whereas OPF-nomemory has a better performance in batches #6, #10, #16, #20, #22, #23, among others. The major version has a similar performance as OPF-nomemory, which figured out as the two best techniques concerning this dataset.

Fig. 13 depicts the results concerning Poker dataset. The weighted version has a better performance than combined version, whereas the major version has a similar performance as OPF-fullmemory. Once again, these last two approaches obtained the best results so far. The accuracies vary consistently, thus degrading the performance of standard OPF and OPF-nomemory. Some batch transitions can really show the problem of concept drift in this dataset. Consider the transition between batches #12 and #13: clearly, one can observe the accuracy decreased from 84.8% to nearly 60% considering standard OPF. Although all the other accuracies decreased either, the proposed OPF with majority voting reached nearly 72%.

Fig. 14 depicts the results concerning SEA dataset. The combined and weighted versions have similar behavior and also a higher accuracy when compared to traditional OPF and OPF with concept drift handling versions. This dataset showed the robustness of the proposed approaches, which obtained the best results so far, outperforming considerably standard OPF and OPF with concept drift handling with no ensemble-based learning.

Fig. 15 depicts the results concerning Usenet1 dataset. Once again, the combined and weighted versions obtained similar performance, whereas major version figured an oscillatory behavior when compared to other ensemble versions. The same behavior can be observed in Usenet2 dataset as well (Fig. 16).

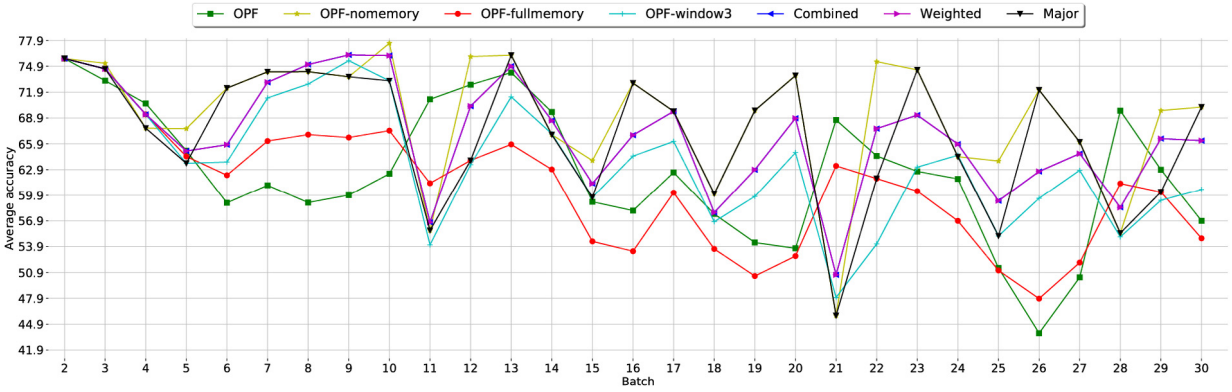

Fig. 12. Experimental results concerning Hyperplane dataset with respect to Experiment 1.

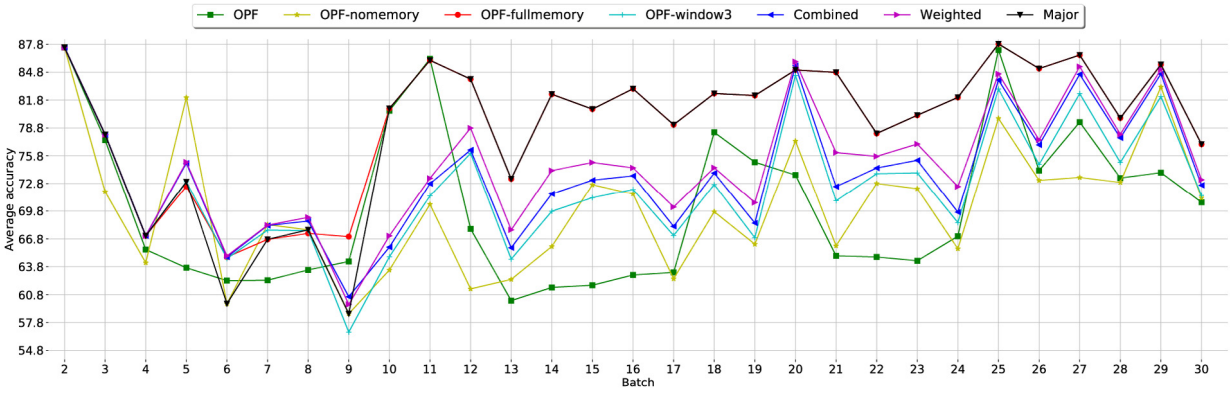

Fig. 13. Experimental results concerning Poker dataset with respect to Experiment 1.

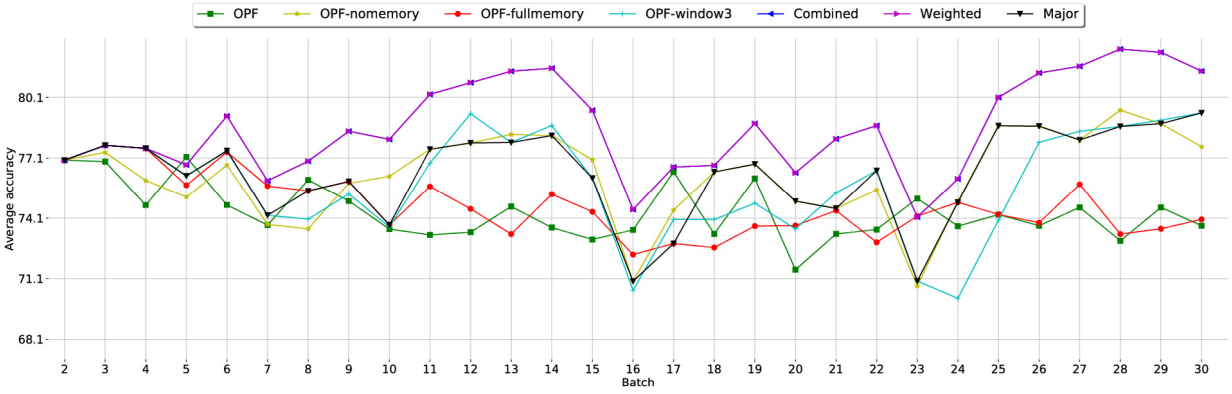

Fig. 14. Experimental results concerning SEA dataset with respect to Experiment 1.

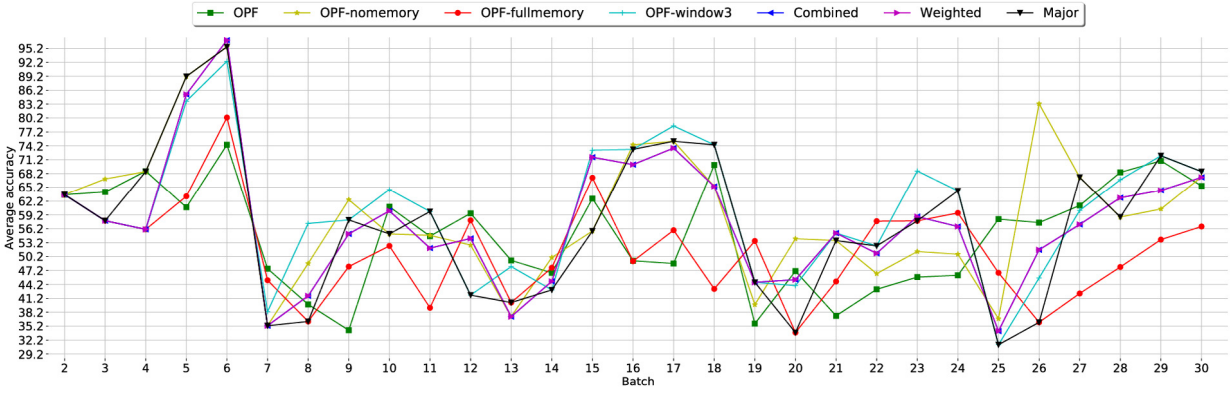

Fig. 15. Experimental results concerning Usenet1 dataset with respect to Experiment 1.

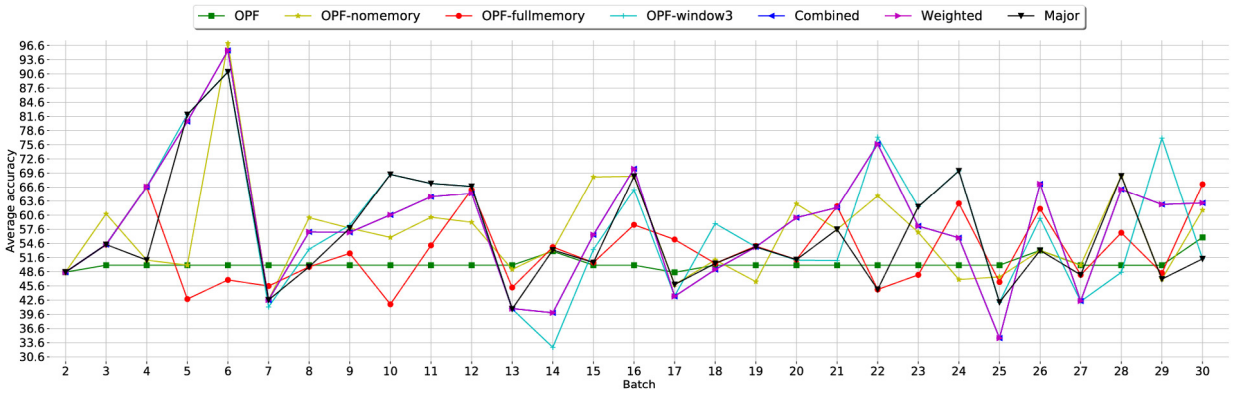

Fig. 16. Experimental results concerning Usenet2 dataset with respect to Experiment 1.

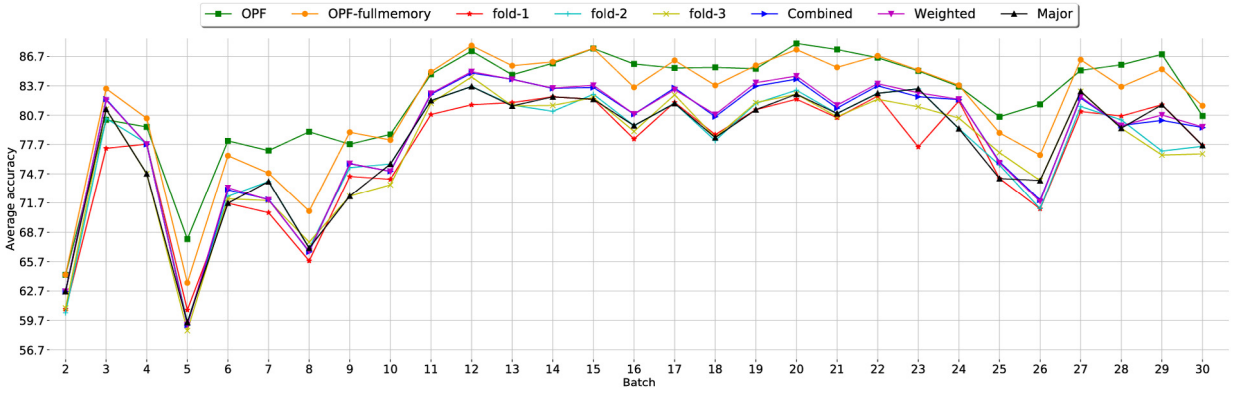

Fig. 17. Experimental results concerning Covtype dataset with respect to Experiment 2 in full-memory management.

5.2. Experiment 2

In this experiment, we decided to verify whether one can improve the concept drift handling using ensemble-based learning, but not combining different approaches, i.e., OPF-fullmemory, OPF-nomemory, and OPF-window3. Therefore, the idea was, for each aforementioned approach, to partition the training data into three subsets, and then apply ensemble learning over them. In this experiment, we are comparing standard OPF, OPF-nomemory, OPF-fullmemory, OPF-window3, the proposed ensemble-based learning with combined, weighted, and major approaches, as well as the performance of each individual fold (i.e., fold1, fold2, and fold3).

Fig. 17 depicts the results concerning Covtype dataset. The combined and weighted versions obtained similar behavior, being the weighted version better in some batches. The major version has a worse behavior among all. In no-memory and window-3 management, the traditional OPF obtained the higher accuracy among all. A similar behavior to the one obtained in Experiment 1 can be observed, i.e., with OPF and OPF-fullmemory obtaining the most accurate results.

In the Electricity dataset (Fig. 18), the combined and weighted versions of full-memory, no-memory, and window-3 obtained similar behaviors, being the majority voting the one with lower accuracies. One can also observe that standard OPF did not perform well, with accuracies much below the ones obtained by the others. We can observe in the no-memory management (Fig. 18) that OPF-nomemory and the combined and weighted versions figured an oscillatory performance.

In Hyperplane dataset (Fig. 19), the combined and weighted versions of full-memory, no-memory, and window-3 obtained similar performances, being the majority voting the one with smaller accuracies. The OPF with ensemble learning and traditional OPF figured an oscillatory performance in the full-memory approach, whereas traditional OPF in no-memory management has higher accuracy only on batches #4, #11, #21 and #28. Traditional OPF in window-3 management (Fig. 19) performed a little better when compared to the no-memory management. Interestingly, standard OPF performed better in a transition between batches #10 and #11, which seems to be a concept drift, but with the statistics of the testing data similar to the one standard OPF has been trained for (the beginning of the streaming).

In Poker dataset (Fig. 20), the combined, weighted and major versions of full-memory, no-memory, and window-3 obtained similar performances once more, with the weighted version the most accurate one followed by the combined one. In this dataset, one can clearly observe some situations where the standard OPF's accuracy drops considerably (e.g., transitions between batches #11 and #12, and between batches #20 and #21), since it is not prepared to handle concept drift.

In SEA dataset (Fig. 21), the combined and weighted versions of full-memory, no-memory, and window-3 obtained similar performances, performing better than traditional OPF and OPF with concept drift handling. The major version achieved the worst accuracy compared to the other ensemble versions.

In Usenet1 (Fig. 22) and Usenet 2 (Fig. 23) datasets, the combined and weighted versions of full-memory, no-memory, and window-3 management obtained similar performances either, with the ensemble learning with majority voting figuring an oscillatory behavior.

5.3. Experiment 3

In this experiment, we consider the last three batches to perform the ensemble learning step. Therefore, one classifier is trained

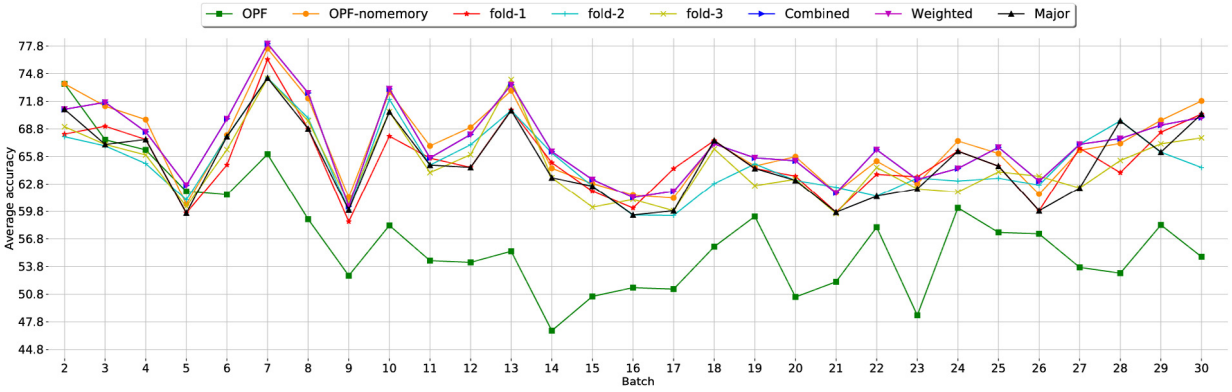

Fig. 18. Experimental results concerning Electricity dataset with respect to Experiment 2 in no-memory management.

Fig. 19. Experimental results concerning Hyperplane dataset with respect to Experiment 2 in window-3 management.

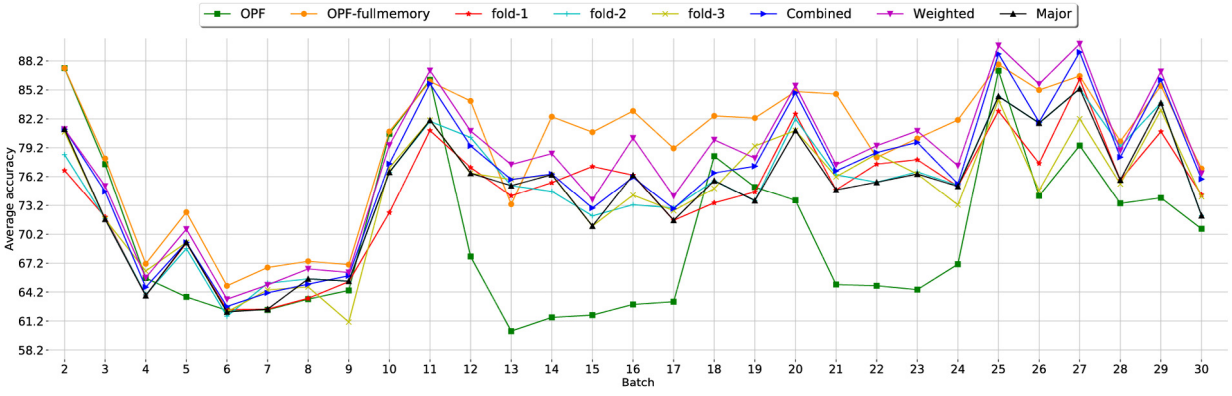

Fig. 20. Experimental results concerning Poker dataset with respect to Experiment 2 in full-memory management.

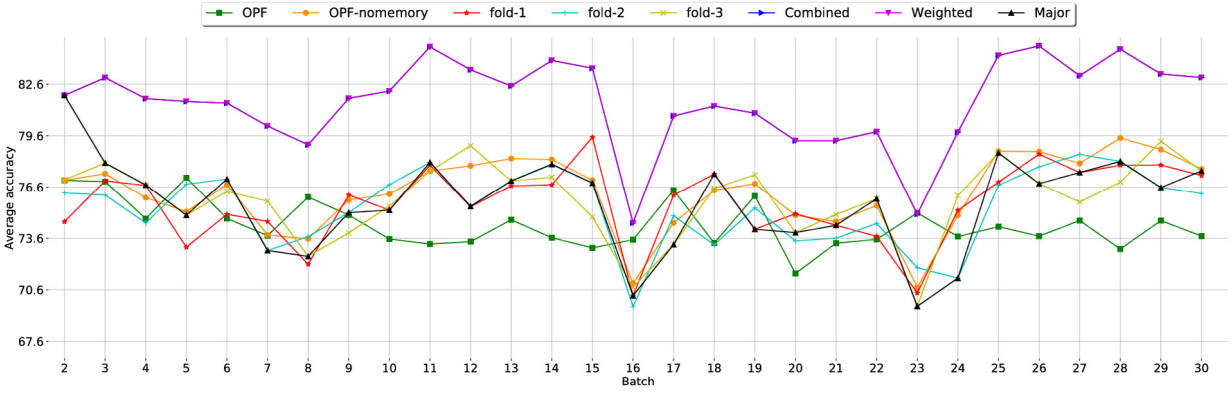

Fig. 21. Experimental results concerning SEA dataset with respect to Experiment 2 in no-memory management.

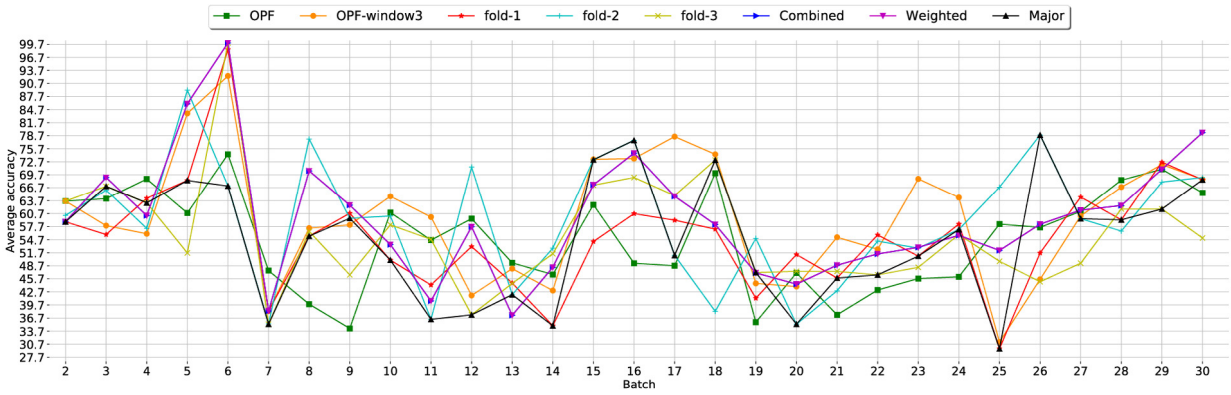

Fig. 22. Experimental results concerning Usenet1 dataset with respect to Experiment 2 in window-3 management.

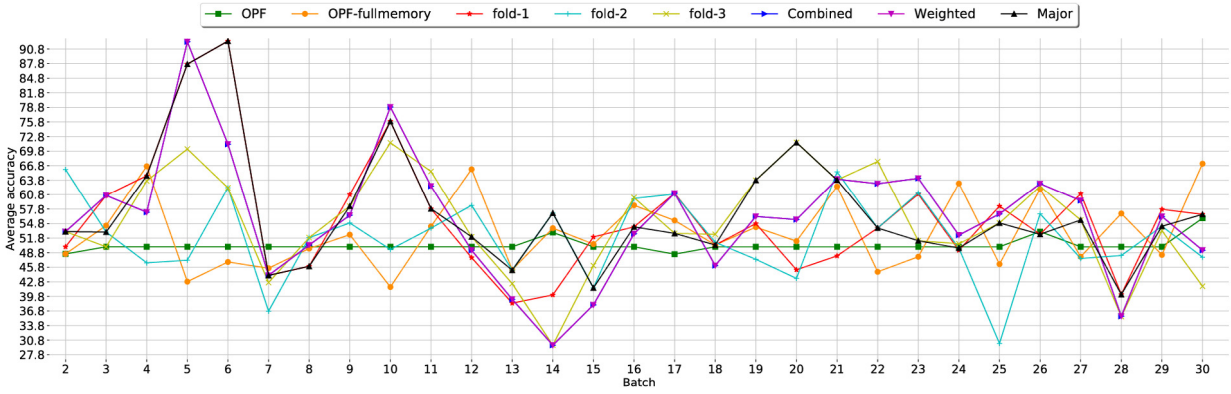

Fig. 23. Experimental results concerning Usenet2 dataset with respect to Experiment 2 in full-memory management.

only once on each batch for further combination of results. This experiment compared traditional OPF, OPF-nomemory, OPF-window3, as well as the ensemble-based approaches combined and weighted.

Fig. 24 presents the results over Covtype dataset. Once again, traditional OPF obtained the best recognition rates so far, being the weighted version similar to the combined approach, but being a little more accurate. In this case, the proposed approaches did not perform well, probably because the last three batches may have conflicting information, i.e., different statistics concerning the testing batch.

Fig. 25 depicts the results concerning Electricity dataset. The combined and weighted version obtained similar performances, as well as an oscillatory behavior compared to OPF with concept drift with no ensemble learning can be observed as well. In this dataset, the proposed approaches outperformed all the others, thus showing to be way robust to concept drift. As a matter of fact, only in a few situations OPF-nomemory obtained the highest accuracies.

Fig. 26 depicts the results concerning Hyperplane dataset. Once again, the proposed approaches (combined and weighted versions) obtained the best results and with similar performances, being also slightly more accurate than OPF-window3. For some batches, one can observe that OPF-nomemory obtained very much accurate results. OPF with window-of-size 3 did not perform well with respect to the other approaches, although it has outperformed OPF with no concept drift handling. Actually, the task of choosing proper window sizes is still an open issue in the context of concept drift. The point is that smaller windows may not contain enough information about the training data, whereas bigger-sized windows may have too old and outdated information about the new data that is coming through the stream.

Fig. 27 depicts the results concerning Poker dataset. The combined and weighted versions obtained similar performances, but the weighted version a little more accurate. In this situation, a window of size 3 to handle concept drift seems to be a good alternative, since it allowed the best recognition rates for some batches. Standard OPF did perform well in some situations too, mainly in batches where concept drift-driven approaches did not obtain satisfactory results.

Fig. 28 depicts the results concerning SEA dataset. The combined and weighted variants obtained similar performances. As one can observe, the aforementioned approaches obtained the better performance among all, except in batch #24. In this specific situation, OPF-nomemory was slightly more accurate.

Figs. 29 and 30 depict the results concerning Usenet1 and Usenet2 dataset, respectively. The combined and weighted version obtained similar performances in both datasets, as one can observe in the previous experiments concerning these datasets as well.

5.4. Discussion

In regard to Experiments 1 and 2, in most datasets they allowed us similar recognition rates, but we did not observe significant better performances with respect to the ensemble-based approaches and individual folds in Experiment 2, just in some cases. However, the proposed approaches outperformed individual folds in almost all datasets. Also, Experiment 1 showed us the proposed approaches are pretty much suitable to handle the concept drift, and using information from different learners can improve the recognition rates. For some datasets, we observed that combining information from different concept drift handling approaches (Experiment 1) is more beneficial than using information from the same handling approach. We can verify in Experiment 2 that training in each fold is faster than training in the non-split version (OPF-nomemory, OPF-fullmemory, and OPF-window3). Although

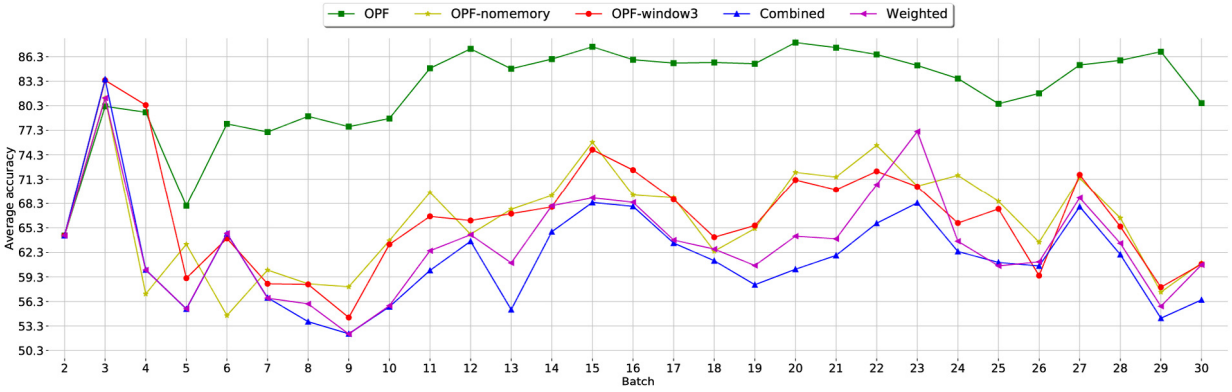

Fig. 24. Experimental results concerning Covtype dataset with respect to Experiment 3.

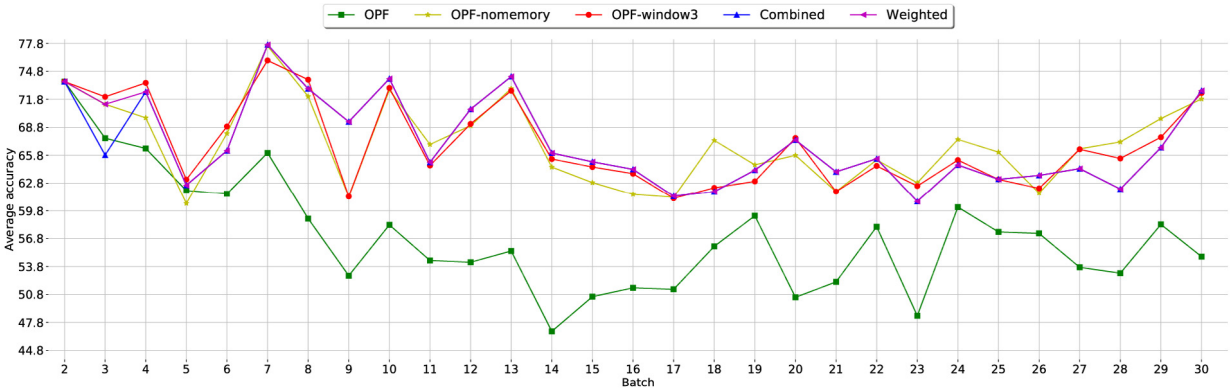

Fig. 25. Experimental results concerning Electricity dataset with respect to Experiment 3.

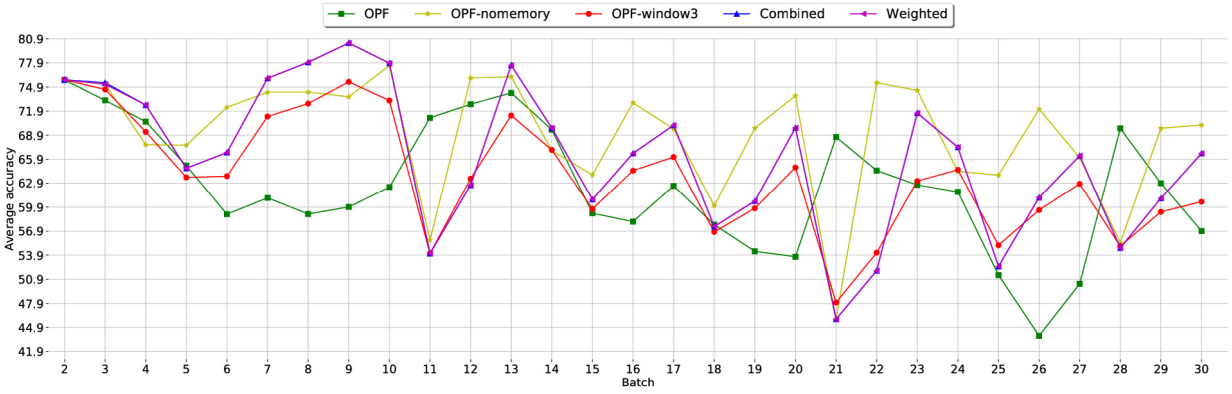

Fig. 26. Experimental results concerning Hyperplane dataset with respect to Experiment 3.

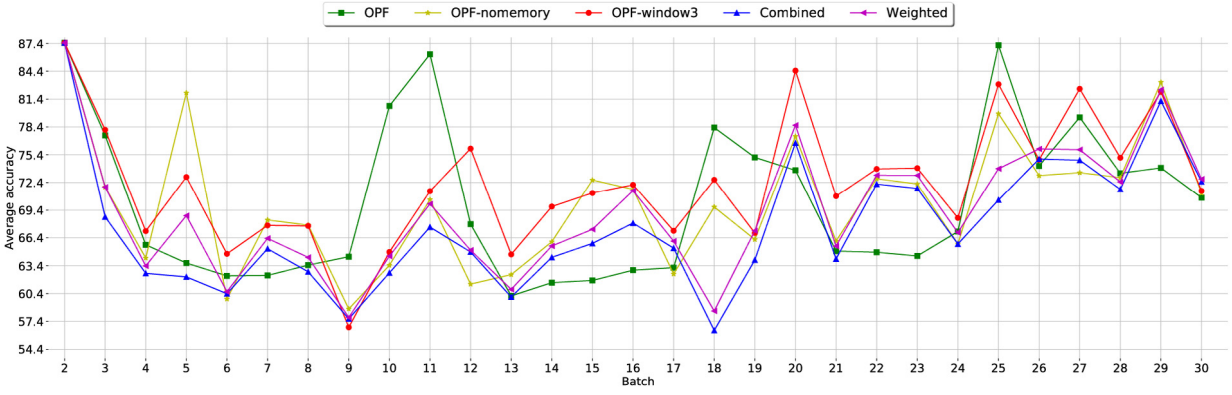

Fig. 27. Experimental results concerning Poker dataset with respect to Experiment 3.

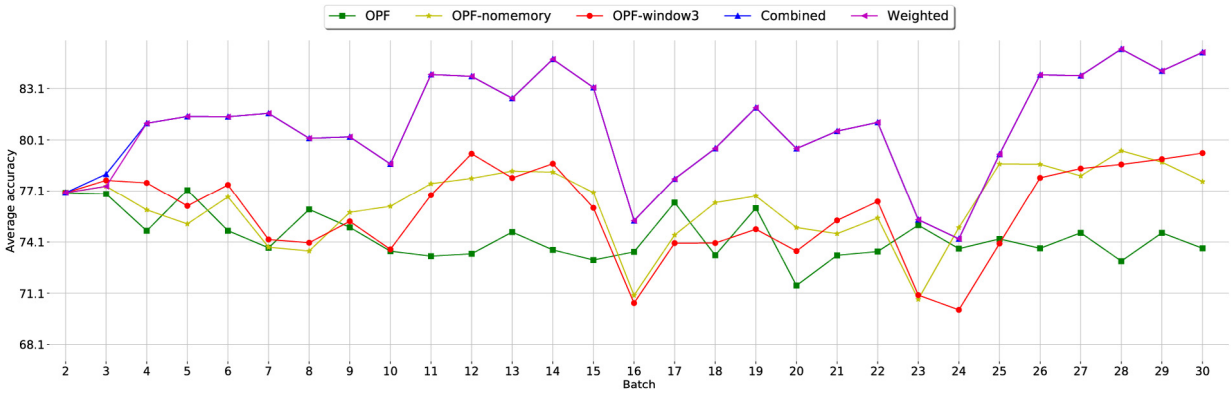

Fig. 28. Experimental results concerning SEA dataset with respect to Experiment 3.

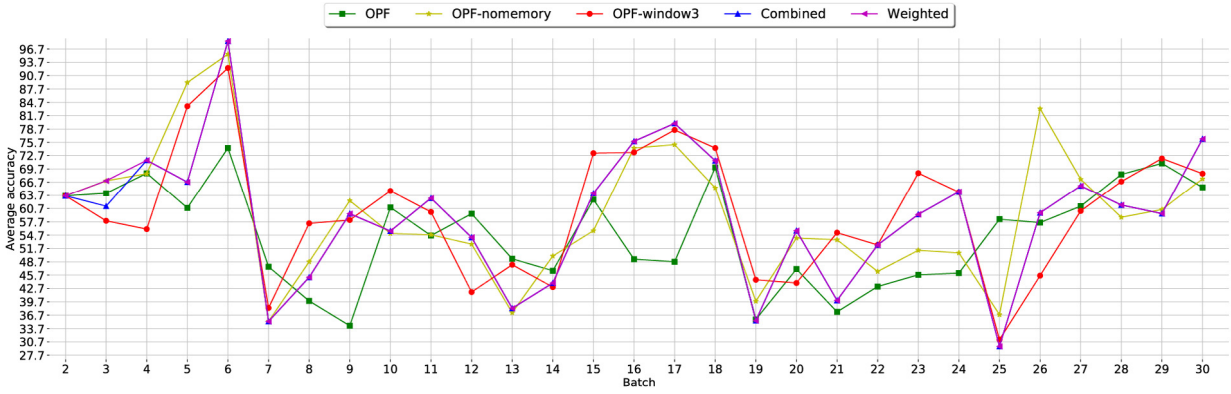

Fig. 29. Experimental results concerning Usenet1 dataset with respect to Experiment 3.

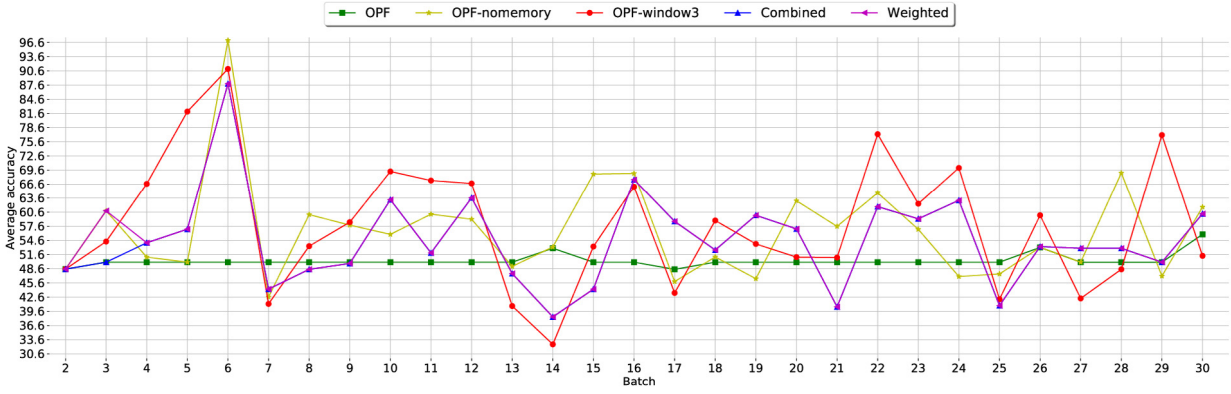

Fig. 30. Experimental results concerning Usenet2 dataset with respect to Experiment 3.

the size is pretty much the same, since OPF has a quadratic complexity for training, the learning process is faster when performed in smaller training sets [40].

Experiment 3 allowed us to combine the three last batches, and not mixing different concept drift approaches. We have observed that such methodology brings us results that are somehow close to the ones obtained with OPF with window-of-size 3 for concept drift handling, but being faster, since we are now training on three individual batches and each one only once.

6. Conclusions and future works

In this paper, we dealt with the problem of concept drift concerning the Optimum-Path Forest with ensemble learning. As far as we are concerned, no study aimed at considering the aforementioned method to handle changes in a data stream up to date.

The experiments over real and synthetic datasets compared standard OPF, OPF-fullmemory, OPF-nomemory, and OPF-window3 approaches, as well as and three proposed distinct ensemble-based versions to address concept drift. We have shown that OPF ensemble is suitable to work under these dynamic scenarios since its recognition rates were considerably better compared to traditional OPF and OPF with concept drift handling with no ensemble learning.

We have observed that, in Experiment 1, the combined ensemble approach obtained the highest accuracy on SEA dataset, considerably improving its performance compared to traditional OPF and the OPF with concept drift handling versions with no ensemble learning. We can also verify that combined methods have better

accuracy than the ones obtained on each fold individually in Experiment 2. Additionally, the main advantage concerning Experiment 3 over OPF-window3 is its training time, since training in smaller data is faster than a larger one, i.e., training three individual batches is faster than training three merged batches.

In regard to future works, we intend to dynamically learn and adjust the window size, as well as to implement OPF on data stream with incremental learning, i.e., where we can pose constraints in the training time. Therefore, the idea is not retraining the whole classifier when a new batch comes to play but just update the trained classifier.

Acknowledgments

The authors are grateful to FAPESP, Brazil grants #2013/07375-0, #2014/16250-9, #2014/12236-1, and #2016/19200-8, Capes and also CNPq, Brazil grants #306166/2014-3, #304315/2017-6, #307066/2017-7, and #430274/2018-1.

References

[1] J. Gama, P. Medas, G. Castillo, P. Rodrigues, Learning with drift detection, in: SBIA Brazilian Symposium on Artificial Intelligence, Springer Verlag, 2004, pp. 286–295.
[2] I. Žliobaitė, Learning under concept drift: An overview, in: Computing Research Repository by Cornell library, 2010, pp. 1–1.
[3] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation, ACM Comput. Surv. 46 (4) (2014) 44:1–44:37.
[4] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, V. Lempitsky, Domain-adversarial training of neural networks, J. Mach. Learn. Res. 17 (1) (2016) 1–35.
[5] S. Ben-David, J. Blitzer, K. Crammer, A.K.F. Pereira, V.J.W. Vaughan, A theory of learning from different domains, Mach. Learn. J. 79 (2010) 151–175.
[6] M. Wang, W. Deng, Deep visual domain adaptation: A survey, Neurocomputing 312 (2018) 135–153.
[7] F.I. Eyiokur, D. Yaman, H.K. Ekenel, Domain adaptation for ear recognition using deep convolutional neural networks, IET Biometrics 7 (3) (2018) 199–206.
[8] R. Klinkenberg, T. Joachims, Detecting concept drift with support vector machines, in: Proceedings of the Seventeenth International Conference on Machine Learning, Morgan Kaufmann, 2000, pp. 487–494.
[9] P. ZareMoodi, H. Beigy, S.K. Siahroudi, Novel class detection in data streams using local patterns and neighborhood graph, Neurocomputing 158 (2015) 234–245.
[10] K. Stanley, Learning Concept Drift with a Committee of Decision Trees, Tech. Rep. AI03-302, The University of Texas at Austin, Department of Computer Sciences, 2003.
[11] J. Kolter, M. Maloof, Dynamic weighted majority: An ensemble method for drifting concepts, J. Mach. Learn. Res. 8 (2007) 2755–2790.
[12] M. García, J. Ávila, R. Fidalgo, A. Bifet, R. Gavaldá, R. Bueno, Early drift detection method, in: Fourth International Workshop on Knowledge Discovery from Data Streams, 2006, pp. 77–86.
[13] R.S. Barros, D.R. Cabral, P.M. Gonąlves, S.G. Santos, RDDM: Reactive drift detection method, Expert Syst. Appl. 90 (2017) 344–355.
[14] P. Li, X. Wu, X. Hu, H. Wang, Learning concept-drifting data streams with random ensemble decision trees, Neurocomputing 166 (2015) 68–83.
[15] G.J. Ross, N.M. Adams, D.K. Tasoulis, D.J. Hand, Exponentially weighted moving average charts for detecting concept drift, Pattern Recognit. Lett. 33 (2) (2012) 191–198.
[16] G. Widmer, M. Kubat, Learning in the presence of concept drift and hidden contexts, Mach. Learn. 23 (1) (1996) 69–101.
[17] S. Liu, L. Feng, J. Wu, G. Hou, G. Han, Concept drift detection for data stream learning based on angle optimized global embedding and principal component analysis in sensor networks, Comput. Electr. Eng. 58 (2017) 327–336.
[18] A. Bifet, R. Gavaldà, Learning from time-changing data with adaptive windowing, in: SIAM International Conference on Data Mining, 2007.
[19] S. Xu, J. Wang, Dynamic extreme learning machine for data stream classification, Neurocomputing 238 (2017) 433–449.
[20] J.L. Lobo, I. Laña, J.D. Ser, M.N. Bilbao, N. Kasabov, Evolving spiking neural networks for online learning over drifting data streams, Neural Netw. 108 (2018) 1–19.
[21] Y. Zhang, G. Chu, P. Li, X. Hu, X. Wu, Three-layer concept drifting detection in text data streams, Neurocomputing 260 (2017) 393–403.
[22] J.L. Lobo, J.D. Ser, M.N. Bilbao, C. Perfecto, S. Salcedo-Sanz, DRED: An evolutionary diversity generation method for concept drift adaptation in online learning environments, Appl. Soft Comput. 68 (2018) 693–709.
[23] B. Mirza, Z. Lin, N. Liu, Ensemble of subset online sequential extreme learning machine for class imbalance and concept drift, Neurocomputing 149, Part A (2015) 316–329.
[24] E. Arabmakki, M. Kantardzic, SOM-based partial labeling of imbalanced data stream, Neurocomputing 262 (2017) 120–133.
[25] J.L. Lobo, J.D. Ser, M.N. Bilbao, I. Laña, S. Salcedo-Sanz, A probabilistic sample matchmaking strategy for imbalanced data streams with concept drift, in: Intelligent Distributed Computing X, Springer International Publishing, Cham, 2017, pp. 237–246.
[26] T.S. Sethi, M. Kantardzic, On the reliable detection of concept drift from streaming unlabeled data, Expert Syst. Appl. 82 (2017) 77–99.
[27] J. de A. Silva, E.R. Hruschka, J. Gama, An evolutionary algorithm for clustering data streams with a variable number of clusters, Expert Syst. Appl. 67 (2017) 228–238.
[28] J.P. Papa, A.X. Falcão, C.T.N. Suzuki, Supervised pattern classification based on optimum-path forest, Int. J. Imaging Syst. Technol. 19 (2) (2009) 120–131.
[29] J.P. Papa, A.X. Falcão, V.H.C. Albuquerque, J.M.R.S. Tavares, Efficient supervised optimum-path forest classification for large datasets, Pattern Recognit. 45 (1) (2012) 512–520.
[30] J.P. Papa, S.E.N. Fernandes, A.X. Falcão, Optimum-Path Forest based on k-connectivity: Theory and applications, Pattern Recognit. Lett. 87 (2017) 117–126.
[31] A. Iwashita, J. Papa, Learning concept drift with optimum-path forest, in: 23rd Iberoamerican Congress on Pattern Recognition, 2018.
[32] D. Farid, L. Zhang, A. Hossain, C. Rahman, R. Strachan, G. Sexton, K. Dahal, An adaptive ensemble classifier for mining concept drifting data streams, Expert Syst. Appl. 40 (15) (2013) 5895–5906.
[33] A. Tsymbal, The Problem of Concept Drift: Definitions and Related Work, Tech. Rep., Computer Science Department, Trinity College Dublin, Ireland, 2004.
[34] J.P. Papa, A.X. Falcão, A new variant of the optimum-path forest classifier, in: Advances in Visual Computing: 4th International Symposium, Springer, Berlin, Heidelberg, 2008, pp. 935–944.
[35] J.P. Papa, A.X. Falcão, A learning algorithm for the optimum-path forest classifier, in: A. Torsello, F. Escolano, L. Brun (Eds.), Graph-Based Representations in Pattern Recognition, in: Lecture Notes in Computer Science, 5534, Springer, Berlin, Heidelberg, 2009, pp. 195–204.
[36] A.X. Falcão, J. Stolfi, R.A. Lotufo, The image foresting transform: Theory, algorithms, and applications, IEEE Trans. Pattern Anal. Mach. Intell. 26 (1) (2004) 19–29.
[37] P.B. Ribeiro, J.P. Papa, R.A.F. Romero, An ensemble-based approach for breast mass classification in mammography images, in: SPIE Medical Imaging, 2017, 101342N-1–101342N-8.
[38] S.E.N. Fernandes, J.P. Papa, Pruning optimum-path forest ensembles using quaternion-based optimization, in: 2017 International Joint Conference on Neural Networks, IJCNN, 2017, pp. 984–991.
[39] S.E.N. Fernandes, A.N. Souza, D.S. Gastaldello, D.R. Pereira, J.P. Papa, Pruning optimum-path forest ensembles using metaheuristic optimization for land-cover classification, Int. J. Remote Sens. 38 (2017) 5736–5762.
[40] M.P. Ponti, J.P. Papa, Improving accuracy and speed of optimum-path forest classifier using combination of disjoint training subsets, in: C. Sansone, J. Kittler, F. Roli (Eds.), Multiple Classifier Systems, Springer, Berlin, Heidelberg, 2011, pp. 237–248.

Adriana Sayuri Iwashita received Bachelor degree in Computer Science in 2010 from São Paulo State University, SP, Brazil. In 2013, she received a Master Degree in Computer Science from São Paulo State University, SP, Brazil. Currently a Ph.D. student at the Federal University of São Carlos, SP, Brazil. Her research interests include machine learning, pattern recognition and concept drift detection.

Victor Hugo C. de Albuquerque has a Ph.D. in Mechanical Engineering with emphasis on Materials from the Federal University of Paraíba (UFPB, 2010), an M.Sc. in Teleinformatics Engineering from the Federal University of Ceará (UFC, 2007), and he graduated in Mechatronics Technology at the Federal Center of Technological Education of Ceará (CEFETCE, 2006). He is currently Assistant VI Professor of the Graduate Program in Applied Informatics, and coordinator of the Laboratory of Bioinformatics at the University of Fortaleza (UNIFOR). He has experience in Computer Systems, mainly in the research fields of: Applied Computing, Intelligent Systems, Visualization and Interaction, with specific interest in Pattern Recognition, Artificial Intelligence, Image Processing and Analysis, as well as Automation with respect to biological signal/image processing,

image segmentation, biomedical circuits and human/brain–machine interaction, including Augmented and Virtual Reality Simulation Modeling for animals and humans. Additionally, he has research at the microstructural characterization field through the combination of non-destructive techniques with signal/image processing and analysis and pattern recognition. Prof. Victor is the leader of the Computational Methods in Bioinformatics Research Group. He is an editorial board member of the IEEE Access, Computational Intelligence and Neuroscience, Journal of Nanomedicine and Nanotechnology Research, and Journal of Mechatronics Engineering, and he has been Lead Guest Editor of several high-reputed journals, and TPC member of many international conferences. He has authored or coauthored over 160 papers in refereed international journals, conferences, four book chapters, and four patents.

João P. Papa received his B.Sc. in Information Systems from the São Paulo State University, SP, Brazil. In 2005, he received his M.Sc. in Computer Science from the Federal University of São Carlos, SP, Brazil. In 2008, he received his Ph.D. in Computer Science from the University of Campinas, SP, Brazil. During 2008–2009, he had worked as a post-doctorate researcher at the same institute, and during 2014–2015 he worked as a visiting scholar at Harvard University. He has been a Professor at the Computer Science Department, São Paulo, State University, since 2009, and his research interests include machine learning, pattern recognition, and image processing. He is also the recipient of the Alexander von Humboldt fellowship.