Tackling Virtual and Real Concept Drifts: An Adaptive Gaussian Mixture Model Approach

Gustavo Oliveira, Leandro Minku, *Senior Member, IEEE,* and Adriano Oliveira, *Senior Member, IEEE,*

Abstract—Real-world applications have been dealing with large amounts of data that arrive over time and generally present changes in their underlying joint probability distribution, i.e., concept drift. Concept drift can be subdivided into two types: virtual drift, which affects the unconditional probability distribution $p(\boldsymbol{x})$, and real drift, which affects the conditional probability distribution $p(y|\boldsymbol{x})$. Existing works focuses on real drift. However, strategies to cope with real drift may not be the best suited for dealing with virtual drift, since the real class boundaries remain unchanged. We provide the first in depth analysis of the differences between the impact of virtual and real drifts on classifiers' suitability. We propose an approach to handle both drifts called On-line Gaussian Mixture Model With Noise Filter For Handling Virtual and Real Concept Drifts (OGMMF-VRD). Experiments with seven synthetics and seven real-world datasets show that OGMMF-VRD outperforms other approaches with separate mechanisms to deal with virtual and real drifts. It also has more stable rankings and smaller drops in performance during drifting periods than existing ensemble approaches, thus being more reliable for adoption in practice.

Index Terms—Data Streams, Virtual Concept Drift, Real Concept Drift, Gaussian Mixture Model.

1 Introduction

In recent years, real-world applications like credit card fraud detection, flight delay and weather forecasting have been dealing with tremendous growth in the amount of data, which typically arrive continuously and sequentially over time and evolve due to the underlying dynamics of real-world activities. Such sequences of data are known as data streams [2]. They are challenging for data modeling systems [3], requiring classifiers to adapt to changes over time. Changes in the underlying distributions of the problem are called concept drifts [4].

Concept drift can be subdivided into two types: virtual drift and real drift [3]. Virtual drift can be defined as a change in the unconditional probability distribution $P(\mathbf{x})$ and real drift can be defined as a change in the conditional probabilities $P(y|\mathbf{x})$. They may occur separately or simultaneously and may have different impacts on the classifier performance.

Most existing work on data stream learning focuses on real drifts, because such drifts change the *true* decision boundaries of the problem, directly degrading the performance of classifiers [5]. As virtual drifts do not change the *true* decision boundaries of the problem, they attracted much less attention from the research community. Nevertheless, they can also affect the classifier performance, because they may affect the suitability of the decision boundaries *learned* by the classifiers. For instance, the appearance of observations in regions of the space that were not covered by training examples may reveal insufficient or incorrectly *learned* decision boundaries, which need to be adjusted for the classifier to remain suitable.

No existing work provides an in depth understanding of the differences between the effect of these two types of drift on the suitability of classifiers. As a result, existing data stream learning approaches treat virtual drifts using the same strategies as for real drifts [5]. A common strategy to adapt to a new (previously unseen) concept is to create a new classifier to learn it. However, such strategy may not be the best for dealing with virtual drifts. This is because the knowledge acquired before the drift may remain valid after a virtual drift occurs [3], given that the *true* decision boundaries do not change. Learning a new classifier from scratch thus wastes potentially useful knowledge that could speed up adaptation to virtual drifts.

Moreover, the strategy of creating new models can be prone to noise, which could be very problematic in the presence of virtual drifts. For instance, approaches based on drift detectors could potentially be tuned to detect minor changes in the underlying distribution such as virtual drifts. This tuning may cause the system to confuse these drifts with noise, thus triggering the unnecessary creation of new models. This could potentially harm system predictive performance as a whole, given that new models require incoming observations to train to become accurate.

With that in mind, this paper provides the first in-depth analysis of the differences between the impact caused by virtual and real drifts on the suitability of Bayesian approaches, as they lend themselves to dealing with different types of drift, due to their pertinence inferences [1]. Besides that, we propose a new approach to handle both virtual and real drifts simultaneously while achieving more robustness to noise. This approach has been guided by the following Research Questions (RQs), which have not been considered by previous work:

RQ1) What is the difference between the impact of

A preliminary version of this research was published in [1].

- *G. Oliveira and A. Oliveira are with the Centro de Informática, Universidade Federal de Pernambuco, Recife, Pernambuco, Brazil.*
  E-mail: ghfmo@cin.ufpe.br and alio@cin.ufpe.br.
- *L. Minku is with the School of Computer Science, University of Birmingham, Birmingham, UK.*
  E-mail: L.L.Minku@cs.bham.ac.uk.

virtual and real drifts on the suitability of classifiers' *learned* decision boundaries and predictive performance over time? This RQ provides the foundation for proposing novel approaches able to more efficiently and effectively deal with both types of drift at the same time. We hypothesize that when virtual drifts occurs, the previously *learned* decision boundaries remain suitable, and the only thing that needs to be done is to learn the emerging region. We also hypothesized that when non-severe real drifts happen, only a small portion of the *learned* decision boundaries becomes unsuitable, whereas severe real drifts require a significant reset of the *learned* decision boundaries.

RQ2) How to deal with both virtual and real drifts while achieving robustness to noise? We explore the potential of GMM to enable different strategies to be used to tackle these different types of drift. GMM has pertinence inferences useful for virtual drifts, which enable us to verify whether or not a new observation belongs to the trained distribution. If new observations arrive in regions of the space that are not covered by existing Gaussians, a new Gaussian can be created for them. As incoming observations could be noise, it is essential for the work to handle it. Therefore, we hypothesize that creating a filter using techniques of instance hardness can help the system be less affected by noise. Also, a selection of models from the pool may increase robustness to false alarms in the drift detections, typically caused by noisy examples.

RQ3) How to best harness knowledge gained from past similar concepts to accelerate adaptation to both virtual and real drifts? The most widely used strategy to deal with concept drift is to learn previously unseen concepts from scratch. This forces the system to use obsolete models until sufficient new data have arrived for retraining, causing a large degradation in performance. Some studies have considered the storage of past models in a pool to accelerate adaptation to recurrent concepts [6]. We hypothesize that saving past GMMs in a pool can not only help the system to adapt to recurrent concepts, but also to accelerate adaptation to virtual and real drifts that lead to new concepts that share similarities to old concepts.

In our previous work [1], we preliminary demonstrated the potential benefit of GMMs for tackling virtual and real drifts. However, that preliminary work (i) did not provide a detailed understanding of how virtual and real drifts affect classifier performances; (ii) ignored the impact of noisy observations on virtual drift adaptation, which makes the system to create Gaussians in unwanted regions which can cause misclassification; (iii) delayed the learning of new regions through Gaussians because it was limited only to very distant areas; (iv) delayed the ability to track data evolution since the Gaussians were updated only when misclassification occurred; and (v) in real drift adaptation, new concepts were learned entirely from scratch, which in turn discard useful knowledge that could be useful in the drift adaptation. Together, these drawbacks limited the predictive performance of that framework. Therefore, the current paper proposes a new approach called On-line Gaussian Mixture Model With Noise Filter For Handling Virtual and Real Concept Drifts (OGMMF-VRD), which overcomes these problems.

This paper is further organized as follows. Section 2 explains related work. Section 3 presents our problem formulation. Section 4 presents the datasets used in our study. Section 5 presents our analysis to answer RQ1. Based on that, Section 6 proposes our approach OGMMF-VRD, partly answering RQ2 and RQ3. Section 7 presents our experimental analysis of the proposed approach, completing the answer to RQ2 and RQ3. Section 8 concludes the paper and gives directions for further research.

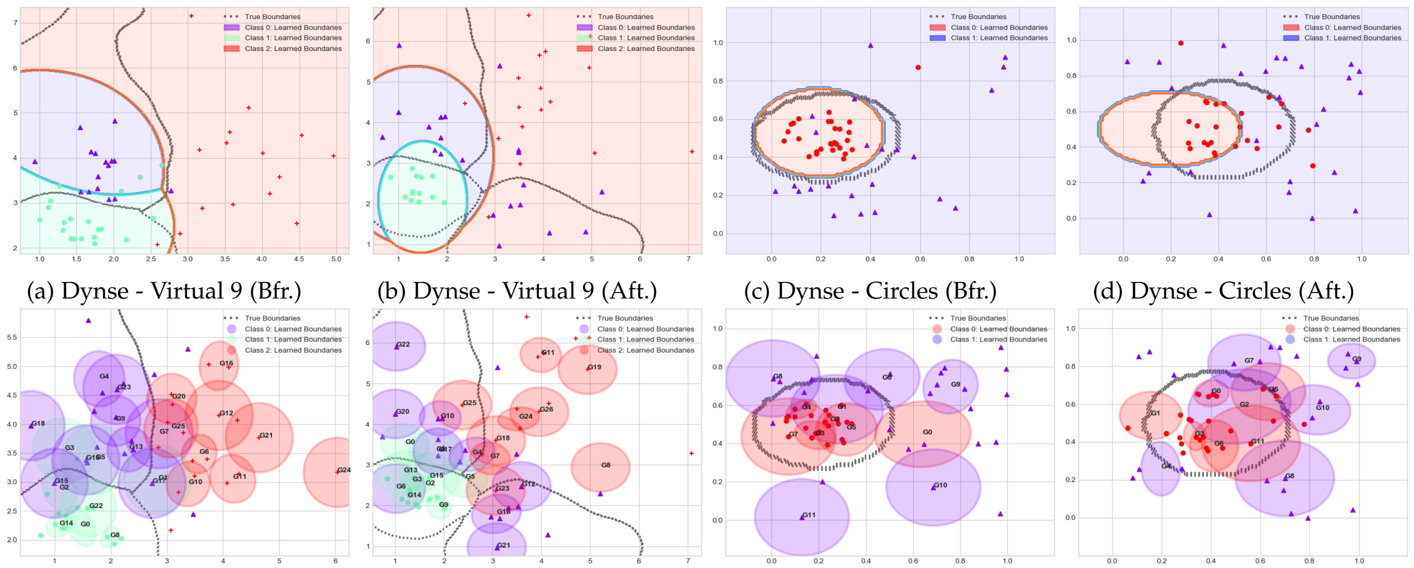
2 Related Work

In general, several existing methods have been proposed to deal with concept drift, and a few different surveys discuss these approaches. For general learning in non-stationary environments, we refer readers to [8, 3]. For ensemble approaches for non-stationary environments, we refer readers to [9, 10]. For class imbalance learning in non-stationary environments, we refer readers to [11]. Despite all these studies, few of the existing approaches differentiate between virtual and real drifts, and propose to handle both. In this sense, the following works stand out:

Oliveira et al.'s Incremental Gaussian Mixture Model for Concept Drift (IGMM-CD) [7] uses *on-line* learning to incorporate new incoming observations. If an incoming observation is classified correctly, the nearest Gaussian to it is updated in an attempt to handle virtual drifts. If the system misclassifies the incoming observation and does not have a Gaussian with the minimum distance ($Cver$) to it, a new Gaussian with size ($sigma\_ini$) is created in an attempt to handle real drifts. Besides that, the system excludes Gaussians with lower density if the predefined number of Gaussians per class (T) is exceeded. Despite that, virtual drifts resulting in misclassifications are treated in the same way as real drifts, causing the unnecessary addition of new Gaussians. In addition, this approach suffers from delays in excluding obsolete Gaussians in the presence of abrupt drifts, degrading performance.

Almeida et al.'s Dynamic Selection Based Drift Handler (Dynse) is an approach based on Dynamic Classifier Selection (DCS) [6]. DCS is the process of selecting a specific classifier for each test instance according to its neighborhood (k) in a validation set. The validation set (M) used by Dynse is represented by a sliding window which traverses the incoming data excluding the oldest observations. Virtual drifts can be dealt with by using a sliding window with very large size, because it will contain many observations corresponding to the current concept. Real drifts can be dealt with by using a sliding window with small size, since its observations can be excluded faster. New classifiers are added into a pool (D) at each batch of m observations to learn new concepts as they arrive. However, as Dynse relies on a pre-defined window size, it can only deal with one type of drift (virtual or real) at a given run. If the data stream presents both virtual and real drifts, or if the chosen window size does not match the type of drift presented in the data stream, Dynse is likely to have its predictive performance negatively affected.

Oliveira et al.'s Gaussian Mixture Model for dealing with Virtual and Real concept Drifts (GMM-VRD) [1] combines *batch* and *on-line* learning to handle both virtual and real drifts. A GMM is trained using a batch with m observations.



(a) Dynse - Virtual 9 (Bfr.) (b) Dynse - Virtual 9 (Aft.) (c) Dynse - Circles (Bfr.) (d) Dynse - Circles (Aft.)

(e) IGMM-CD - Virtual 9 (Bfr.) (f) IGMM-CD - Virtual 9 (Aft.) (g) IGMM-CD - Circles (Bfr.) (h) IGMM-CD - Circles (Aft.)

Fig. 1: Execution of Dynse [6] and IGMM-CD [7] on Virtual 9 and Circles datasets (Tbl. 1). The points represent the first 50 before (bfr.) and after (aft.) the concept drift. Dynse uses as base classifier the Gaussian Naive Bayes and IGMM-CD the GMM. Gray and dotted lines represent the *true* decision boundaries of the problem. The solid colored lines represent the decision boundaries *learned* by each approach. The text G0, ..Gn represents the Gaussian number.

The parameter ($Kmax$) is used to define the maximum number of Gaussians and the Akaike Information Criterion (AIC) is used to define the best number. If a misclassification occurs, the pertinence of the new incoming observation is used to decide whether to update or to create new Gaussians to handle virtual drifts. In parallel, a Concept Drift Test (CDT) is used to monitor the system's performance. If the performance is decreasing, the system is reset to cope with real drifts. The drawback of this approach is its sensitivity to noisy observations, which can cause creation and update of Gaussians on unwanted regions, degrading performance.

Our approach OGMMF-VRD is proposed to overcome the above mentioned problems of existing approaches, aiming at dealing with both virtual and real drifts concurrently.

3 Problem Definition

Consider a data stream as follows: $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_t, y_t), \cdots\}$, where $\mathbf{x}_t \in \chi$ is a d-dimensional vector of input attributes, $y_t \in \gamma$ is a categorical output attribute, χ is the input space and γ is the output space, in which each observation $(\mathbf{x}_t, y_t)$ comes from a joint probability distribution $P_t(\mathbf{x}, y)$.

On-line supervised learning from this kind of data consists of creating a model $f_t : \chi \rightarrow \gamma$ where at each new time step t, the previous model f_{t-1} is updated with the new incoming observation $(\mathbf{x}_t, y_t)$ to be able to generalize to unseen observations of $P_t(\mathbf{x}, y)$.

A challenge faced by on-line supervised learning is that observations produced at distinct time steps t and $t+\Delta$ may come from different joint probability distributions $P_t(\mathbf{x}, y) \neq P_{t+\Delta}(\mathbf{x}, y)$, i.e., the data stream may present concept drift [12]. The joint probability is formalized as $P_t(\mathbf{x}, y) = P_t(y|\mathbf{x})P_t(\mathbf{x})$, where $P_t(\mathbf{x})$ is the probability distribution of inputs and $P_t(y|\mathbf{x})$ is the conditional probability of the outputs given the inputs. The latter represents the *true* decision boundaries of the problem. So, drifts can happen if $P_t(\mathbf{x}) \neq P_{t+\Delta}(\mathbf{x})$, if $P_t(y|\mathbf{x}) \neq P_{t+\Delta}(y|\mathbf{x})$, or both. Drifts affecting $P(\mathbf{x})$ are categorized as virtual drifts and drifts affecting $P(y|\mathbf{x})$ are categorized as real drifts [13]. Only real drifts affect the *true* decision boundaries of the problem, whereas both types of drift may affect the suitability of the *learned* decision boundaries, depending on the learning algorithm and model being adopted.

4 Datasets

We used synthetic and real-world datasets, whose characteristics are presented in Tbl. 1. These datasets are available on Github[1]. All datasets described in Tbl. 1 have continuous attributes, as data clustering methods like the EM algorithm used in GMM are not applicable for datasets with categorical input attributes [14].

Synthetic datasets: Datasets were generated using the Tornado framework (Python) proposed in [15][2]. The descriptions of how to generate them are presented in the supplementary material due to space constraints. The virtual drift datasets have been proposed in [1].

About their characteristics, incremental drifts consist of a steady progression from an old concept to a new one, it can be seen as a sequence of abrupt drifts of low severity. Gradual drifts refers to the transition phase where the probability of observations from the old concept decreases while the probability of the new one increases [16]. Severity represents the percentage of the input space which has its target class changed after the drift is complete [16]. It was calculated

1. https://github.com/GustavoHFMO/OGMMF-VRD/tree/master/data_streams
2. https://github.com/alipsgh/tornado

TABLE 1: Dataset descriptions.

Type	Datasets	Attributes	#Classes	#Examples	Concept Size	#Drifts	Drift Type		Severity of Each Drift
Synthetic	Virtual 5	2	3	10000	2000	5	Virtual	Abrupt	[23.47, 34.1, 24.9, 34.95]
	Virtual 9	2	3	10000	1000	9	Virtual	Abrupt	[28.23, 33.9, 28.32, 33.8, 27.23, 32.6, 23.39, 31.6]
	Circles	2	2	8000	2000	4	Virtual/Real	Incremental	[44.15, 37.55, 32.6]
	Sine1	2	2	10000	2000	5	Real	Abrupt/Recurrent	[89.7, 88.85, 89.45, 87.75]
	Sine2	2	2	10000	2000	5	Real	Abrupt/Recurrent	[89.8, 88.85, 89.2, 87.75]
	SEA	3	3	8000	2000	4	Real	Gradual	[50.4, 24.15, 47.1]
	SEAREC	3	3	16000	2000	8	Real	Gradual/Recurrent	[49.4, 24.1, 45.55, 17.95, 47.8, 26.65, 46.2]
Real	PAKDD	29	2	50000	-	-	-	-	-
	ELEC	4	2	27549	-	-	-	-	-
	NOAA	9	2	18159	-	-	-	-	-
	GasSensor	128	6	13910	-		Real	-	-
	INS-Inc-Reo	33	6	79986	-		Real	Incremental/Recurrent	-
	INS-Inc-Abt	33	6	79896	-		Real	Incremental/Abrupt	-
	INS-Grad	33	6	24150	-		Real	Gradual	-

by generating 2000 random instances and checking the percentage of such instances whose labels change from one concept to another, considering 10% label noise. For virtual drifts, the severity is calculated when a region of the space that previously had $P(\mathbf{x}) = 0$ receives observations of a class. Thus it is considered that this region had its target class changed.

Real-world datasets: Only one modification was made on ELEC dataset, where missing values were removed, resulting in the number of examples in Tbl. 1. These datasets are also available on-line[3], and they were discussed in [17].

5 Impact of Virtual and Real Drifts on Classifier Suitability

This section analyses the impacts of virtual and real drifts on classifiers suitability, answering RQ1. Two representative methods from the literature, IGMM-CD [7] and Dynse [6] (see Section 2), are used. Fig. 1 illustrates some representative plots of their execution in the Virtual 9 and Circles datasets before and after a concept drift.

Virtual drifts had a different effect on the GMM-based (IGMM-CD) and Bayesian-based (Dynse) classifiers. In particular, we can see from Fig. 1b that the decision boundary of Class 2 was incorrectly learned by Dynse, even though a considerable portion of the previously acquired knowledge[4] remained valid after the drift. Therefore, part of the past knowledge acquired for Class 2 must be forgotten in order to modify the decision boundary, while most past knowledge about the Classes 0 and 1 should ideally be retained along with a good portion of the past knowledge of Class 2. Different from Dynse, for IGMM-CD, all past knowledge remained valid after the drift (except for some Gaussians that were incorrectly learned due to noise, which were never valid knowledge), even though such knowledge is insufficient once the virtual drift occurs. This type of model only needs to accommodate additional knowledge when a virtual drift occurs, rather than fixing incorrectly learned knowledge. Accommodating additional knowledge could be done for instance by expanding the decision boundaries of existing Gaussians or adding new Gaussians.

Real drifts always result in a need for forgetting at least part of the previously acquired knowledge. An example of this drift is presented in Fig. 1d when the model chosen by Dynse does not reflect the *true* decision boundaries well. DCS-based techniques, according to the selection rule, tend to choose the best classifier from the pool for current data. So, achieving good accuracy depends on good classifiers already trained on the new concept. If such classifiers are unavailable, they will have low accuracy. As this real drift is non-severe, an approach with *on-line* learning like IGMM-CD (Fig. 1h) was able to keep up faster. However, we can observe a blue Gaussian within the red class, which is a remnant of the old concept learning. Because IGMM-CD does not implement rapid forgetting, old knowledge can cause misclassifications.

Overall, in terms of accuracy drop, we can see that both virtual and real drifts will result in an accuracy drop, given that the *learned* decision boundaries become either incorrect or insufficient. This drop will be smaller/larger depending on the severity of the drift and on how fast the approach can adapt to the drift. However, it is likely that the drop in accuracy for virtual drifts is more in line with the drop in accuracy of non-severe real drifts, as in both cases a good portion of the past knowledge will remain valid. In such cases, adaptation mechanisms able to retain valid past knowledge would enable faster adaptation to the drifts. The drop in accuracy for severe real drifts is likely to be larger, given that a large portion of the previously acquired knowledge will become invalid. In such situation, resetting the system to speed up adaptation to the drift may be useful.

In terms of the need for forgetting past knowledge, for approaches not based on GMM, virtual drifts can have a similar effect to non-severe real drifts as they require forgetting *part* of the past knowledge. Severe real drifts would still have a considerably different effect from virtual drifts on these approaches, as they would require most knowledge to be forgotten, whereas in virtual drifts a good portion of the previously acquired knowledge remains valid. However, for approaches based on GMM, virtual drifts have a considerably different effect from both severe and non-severe drifts, as they do not result in the need for forgetting past knowledge. A good GMM-based approach should be able to benefit from that to adapt to virtual drifts faster.

6 Proposed Method

Considering the problems highlighted in Section 5, we describe in Sections 6.1 and 6.2, the components to answer to RQ2, being the adaptation to virtual and real drifts and the noise filter. Finally, in Section 6.3, the strategy of pool adaptation to answer RQ3.

We present in Fig. 2 the general procedure of the OGMMF-VRD. In the first block, the main system mecha-

3. https://en.wikipedia.org/wiki/Concept_drift#Real

4. We use the term "knowledge" here to refer to the portions of the space considered to belong to each class by the learned classifier.

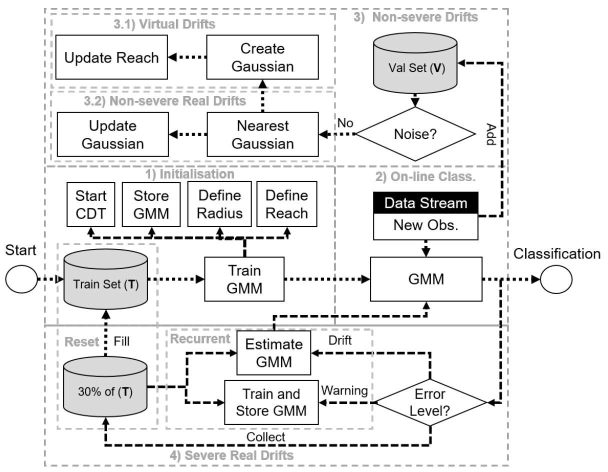


Fig. 2: OGMMF-VRD's overall procedure.

nisms, such as GMM batch training, are initialized. In the second block, we perform the on-line classification of the observations received from the data stream. In the third block, we update the batch-trained GMM on-line, i.e., using each new observation received. Finally, in the fourth block, we introduce knowledge reset mechanisms for dealing with severe drifts.

To clearly understand how OGMMF-VRD works, we provide the source code on GitHub[5], and we present some consecutive plots (Fig. 3) of its execution on Virtual 9 dataset in a video on youtube[6].

6.1 Initialization

The OGMMF-VRD framework is initialized through two steps: (i) **GMM training** and (ii) **drift detector initialization**.

In **GMM training**, we collect an initial portion (m observations) of the data stream as the training set (T) used to initialize a GMM for each class in T using the algorithm Expectation-Maximization (EM). This algorithm initializes each Gaussian C_i on a random subset from the training set (T), and then iteratively adjusts its mean ($\boldsymbol{\mu_i}$), covariance ($\boldsymbol{\Sigma_i}$) and weights (w_i) to maximize the probability of each Gaussian in the distribution modeled. The modeled distribution is given by $P(\mathbf{x}) = \sum_{i=1}^{K} P(\mathbf{x}|C_i) \cdot w_i$, where K is the number of Gaussians and $\mathbf{x}$ is a multivariate observation with d dimensions, formally represented by: $\mathbf{x}^d = \{x_1, x_2, .., x_d\}$. Each constant w_i is a weight representing the number of observations that constitute the Gaussian i, where $0 \leq w_i \leq 1$ and $\sum_{i=1}^{K} w_i = 1$. $P(\mathbf{x}|C_i)$, represents the conditional probability of observation $\boldsymbol{x}$ with respect to the Gaussian C_i. This probability is computed using the mean ($\boldsymbol{\mu_i}$) and the covariance ($\boldsymbol{\Sigma_i}$) of the Gaussian C_i as follows:

$$P(\mathbf{x}|C_i) = \frac{1}{(2\pi^{d/2}\sqrt{|\boldsymbol{\Sigma_i}|})} exp(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu_i})^T \boldsymbol{\Sigma_i^{-1}}(\boldsymbol{x} - \boldsymbol{\mu_i})) \tag{1}$$

5. https://github.com/GustavoHFMO/OGMMF-VRD
6. https://www.youtube.com/watch?v=lP-onPHSR0A

In order to define the optimum number of Gaussians, for each class, we train different GMMs ranging the number of Gaussians from 1 to $Kmax$. The best resulting GMM model is chosen using the higher value of AIC criterion and is added to the final GMM model. The AIC criterion is defined by $2 \cdot p - 2 \cdot L$. Here, p represents the number of model parameters. In a GMM with only one Gaussian, the parameters are the mean ($\boldsymbol{\mu_i}$), covariance ($\boldsymbol{\Sigma_i}$), and weight (ω_i). So, for each existing Gaussian in a GMM, the value of p is multiplied by three. The parameter L represents the maximum likelihood function of the GMM on a set of m observations, defined by Eq. 2:

$$L = \sum_{i=1}^{m} log \sum_{j=1}^{K} P(\boldsymbol{x_i}|C_j) \cdot w_j \tag{2}$$

The final GMM will be used to predict the labels of the incoming observations ($\boldsymbol{x_t}$) in Part 2 (*On-line* classification) of Fig. 2. The predicted label can be used by users for decision-making, using $\hat{y} = \text{argmax}_{i \in \{1,2,\cdots,K\}} P(C_i|\boldsymbol{x})$, where $\hat{y}$ represents the predicted label and $P(C_i|\boldsymbol{x})$ represents the posterior probability of a Gaussian C_i given the observation $\boldsymbol{x}$, as defined by Eq. 3. Thus, the incoming observation ($\boldsymbol{x_t}$) is classified with a class of the Gaussian that presented for it the higher posterior probability.

$$P(C_i|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|C_i) \cdot w_i}{\sum_{i=1}^{K} P(\boldsymbol{x}|C_i) \cdot w_i} \tag{3}$$

In **drift detector initialization**, we start a CDT to monitor the system error and identify performance degradation. In an experiment we evaluate ECDD [18], CUSUM [19], FHDDM [20], DDM [21], and EDDM [22]. EDDM reached the best results and so was chosen. After initialization, non-severe drifts are treated as in Section 6.2.1 and severe drifts as in Section 6.2.2.

6.2 Coping with Virtual and Real Drifts While Achieving Robustness to Noise

6.2.1 Dealing with Virtual Drifts and Non-Severe Real Drifts

This part of the OGMMF-VRD aims to maintain the useful knowledge of the system in the presence of virtual and non-severe real drifts. Alg. 1 presents the overall procedure.

Algorithm 1 NonSevereDriftAdaptation()

```
   Input: observation ($\boldsymbol{x_t}$, $y_t$)
1: gaussian, pertinence ← GaussianClose($\boldsymbol{x_t}$, $y_t$)
2: UpdateGaussian(gaussian, $\boldsymbol{x_t}$)
3: if pertinence > $\theta$ then
4:     CreateGaussian($\boldsymbol{x_t}$, $y_t$)
5:     UpdateReach(pertinence)
6: end if
```

In Line 1, Gaussian Close represents the mechanism used to determine when to create a new Gaussian, and which Gaussian will be updated. For each incoming observation ($\boldsymbol{x_t}$), we need to know where it is located in relation to existing Gaussians. For that, we use Alg. 2.

Algorithm 2 GaussianClose()

```
    Input: observation ($x_t$, $y_t$)
    Output: gaussian, pertinence
 1: aux ← ∅
 2: for each $gaussian$ in GMM do
 3:     if $gaussian \in y_t$ then
 4:         aux.append($P(\boldsymbol{x_t}|gaussian)$)          ▷ Eq. 1
 5:     else
 6:         aux.append(0)
 7:     end if
 8: end for
 9: ngaussian ← argmax(aux)                     ▷ Nearest gaussian
10: pertinence ← aux[$ngaussian$]               ▷ Pertinence of $x_t$
```

This routine computes the conditional probability (Eq. 1) of an incoming observation ($\boldsymbol{x_t}$) for all existing Gaussians with the same class as the incoming observation (y_t) (Lines 1 to 8). This algorithm returns the nearest Gaussian with the same class of the incoming observation (Line 9), and the pertinence of the incoming observation to the Gaussians (Line 10). With this information, we can trigger appropriate drift handling strategies (Sections 6.2.1.1 and 6.2.1.2).

6.2.1.1 **Adjusting Existing Gaussians**: Line 2 (Gaussian Update) of Alg. 1, represents the process to adapt to non-severe real drifts. The idea is that non-severe real drifts change little the decision boundaries of the problem, so a simple displacement of existing Gaussians can handle this. Therefore, we used the modifications to the equations of the EM algorithm proposed by Engel et al. [23]. These modifications can update a Gaussian based on a single new observation ($\boldsymbol{x_t}$) and its current parameters (mean $\boldsymbol{\mu_i^t}$, covariance $\boldsymbol{\Sigma_i^t}$ and weight w_i^t). Another parameter necessary is the variable sp_i^t, defined by $sp_i^t = sp_i^{t-1} + P(C_i|\boldsymbol{x})$, that will store the accumulated posterior probability (Eq. 3) of each Gaussian. Thus, the Equations used to update the Gaussian parameters are shown below [23]:

$$w_i^t = \frac{sp_i^t}{\sum_j^K sp_j^t} \tag{4}$$

$$\boldsymbol{\mu_i^t} = \boldsymbol{\mu_i^{t-1}} + \frac{P(C_i|\boldsymbol{x})}{sp_i^t} \cdot (\boldsymbol{x} - \boldsymbol{\mu_i^{t-1}}) \tag{5}$$

$$\begin{aligned}\boldsymbol{\Sigma_i^t} &= \boldsymbol{\Sigma_i^{t-1}} - (\boldsymbol{\mu_i^t} - \boldsymbol{\mu_i^{t-1}})^T(\boldsymbol{\mu_i^t} - \boldsymbol{\mu_i^{t-1}}) \\ &+ \frac{P(C_i|\boldsymbol{x})}{sp_i^t} \cdot [\boldsymbol{\Sigma_i^{t-1}} - (\boldsymbol{x} - \boldsymbol{\mu_i^t})^T(\boldsymbol{x} - \boldsymbol{\mu_i^t})]\end{aligned} \tag{6}$$

6.2.1.2 **Adding New Gaussians**: Line 4 (Create Gaussian) of Alg. 1 represents the process to adapt to virtual drifts. Since virtual drifts do not change the true decision boundaries of the problem, new Gaussians can be used to accommodate new data that is far from existing Gaussians. For this, when the pertinence of the incoming observation ($\boldsymbol{x_t}$) is less than θ (line 3 of Alg. 1), we consider that this observation is far away from existing Gaussians and it is necessary to create another Gaussian to represent the new region in the feature space. Theta (θ) is the lowest pertinence (Eq. 1) obtained from the observations in the training set (T). Points more distant than theta (θ) are out of GMM's reach.

Thus, the new Gaussian is initialized using $sp_i = w_i = 1$, $\boldsymbol{\mu_i} = \boldsymbol{x_i}$ and $\boldsymbol{\Sigma_i} = Cfc \cdot \boldsymbol{I}$, where $\boldsymbol{I}$ represents the identity matrix, which has the same number of dimensions as $\boldsymbol{x_t}$, and *Cfc* represents the size of the Gaussian circumference defined by $Cfc = (x_{max} - x_{min})/20$, where x_{max} and x_{min} represent the highest and lowest observed value for attributes in the initialization training set (T), and 20 was fixed for all datasets, but can be adjusted (block **Define Radius** in Fig. 2). After the initialization of the new Gaussian's parameters, we re-normalize the weights of all existing Gaussians using Eq. 4.

Line 5 (Update Reach) represents the process of updating θ. For that, we use the pertinence of the incoming observation to substitute the older value (Line 1 of Alg 1). This update indicates that, for the creation of new Gaussians, the incoming observation must have a lower pertinence than theta (θ), indicating that it is farther away.

6.2.2 Dealing With Severe Real Drifts

This part of the OGMMF-VRD aims to reset useless system knowledge. Given the adoption of the mechanisms explained in 6.2.1, non-severe drifts will not degrade the system's predictive performance so much as severe real drifts. Therefore, we consider that severe real drifts will significantly degrade the system's performance, and such degradation can be used to detect such drifts. Thus, to identify these degradations, we will use a CDT, which monitors the classification performance of the system using the error obtained for each new observation. If the system error rises above a threshold, the CDT reports a concept drift, indicating that system knowledge is obsolete. At that time new observations of the data streams are stored and used to retrain the whole system.

OGMMF-VRD thus detects drifts by using the Early Drift Detection Method (EDDM) [22]. If EDDM detects a drift, the system collects new data to learn its knowledge from scratch. EDDM has as parameters the tolerance levels defined by warning (w) and drift (c) levels, and can be tuned so that it only detects severe drifts, which will degrade the system's performance more than non-severe drifts. Each tolerance level yields a different response: (i) for the normal error level the model remains untouched; (ii) for the warning error level, the system begins to collect incoming observations to retrain the model; and (iii) for the drift level, new observations are collected and added to the observations collected during the warning level to fill a batch with m new observations to compose a new training set T to initialize a new model.

6.2.3 Noise Filter

In this Section, we will discuss how to address the part of RQ2 which seeks to know how to achieve robustness to noise in virtual and real drifts. Noisy observations cause two main problems to GMM models: (i) if an observation is too far from its class, the system tends to create a new Gaussian. If this observation was noisy, it means that a Gaussian of an undesired class will be created in the region belonging to another class; (ii) if a Gaussian is updated on a noisy observation, it will move to a region of the space that does not correspond to its true boundary. These points may impair the performance of the GMM over time. To overcome them, we use k-Disagreeing Neighbors (kDN) [24], defined in Eq. 7, as a noise filter.

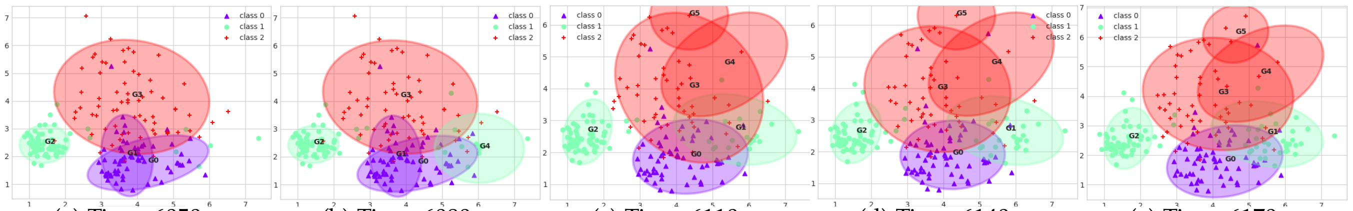


Fig. 3: Execution of OGMMF-VRD on dataset Virtual 9. Each image presented illustrates the *learned* decision boundaries over a batch with 200 observations for different time periods. The text G0, ..Gn represent the number of the Gaussian.

$$kDN(\boldsymbol{x}) = \frac{|\forall \boldsymbol{x'}|\boldsymbol{x'} \in \boldsymbol{kNN(x)} \wedge \boldsymbol{label(x')}|}{k} \tag{7}$$

Consider that we need to determine if an observation $(\boldsymbol{x}, y)$ is noise. The kDN represents the fraction of the k nearest neighbors of this observation that do not share the same class y. Values close to 0 indicate that $(\boldsymbol{x}, y)$ is easy to classify and unlikely to be noise, and close to 1 indicate that it $(\boldsymbol{x}, y)$ is difficult to classify and may be noise.

In OGMMF-VRD, we use the noise filter in two parts: (i) during the GMM initialization and (ii) before the non-severe drift adaptation (Section 6.2.1). In the GMM initialization, we use kDN as a *pre-processing* to remove all noisy observations of the initialization training set (T).

Before the non-severe drift adaptation (Block Noise on Part 3 of Fig. 2), we use kDN to avoid updates using noisy observations. To do this *on-line*, we use a validation set (V). This set has the same size as the training set (T) used to initialize the GMMs. The difference between them is that the T set is only used for initializing the GMMs (Part 1 of Fig. 2) and the V set acts like a sliding window (Part 3 of Fig. 2), on which for each new observation inserted an older one is removed.

Note that the validation set (V) helps to distinguish between noise-free observations, noisy observations and gradual drifts. A noise-free observation has many neighbors with the same class. A noisy observation is an observation that appears in a region with a different class from its own class only once. Gradual drifts are a set of observations that over time are more likely to appear in a new region. Since we store all data in the validation set (V), we will be able to verify the growth of observations in a new region, thus avoiding confusion between gradual drift and noise.

For our approach, we have specified the neighborhood to $k = 5$, and observations with kDN greater than 0.8 should be avoided because they have 80% of their neighborhood with a different class which can strongly indicate that it is a noise, and can hinder the generalization of the system.

6.3 Harnessing Knowledge From Past Similar Concepts by Using a Pool Adaptation Strategy

This section presents our method proposed to answer RQ3, which seeks to harness past knowledge to accelerate adaptation to both virtual and real drifts. For this, we use a pool (P) to store previously trained GMMs. GMMs are re-initialized at two occasions: (i) when EDDM [22] reaches drift level (c) indicating that the system must be reset, i.e. after collecting m observations from the data stream to use as a new training set (T); and (ii) when EDDM reaches warning level (w) indicating that a severe drift may be occurring (see Section 6.2.2), i.e. when 30% of m are collected. This is represented by block **Store GMM** in Fig. 2.

Classifiers are estimated from the pool when EDDM [22] reaches the drift level (c). So to avoid waiting for all m observations used for retraining, when we get 30% of m, we select the classifier from the pool with the best accuracy for this data and use it to replace the current model. This is represented by block **Estimate GMM** in Fig. 2.

If the maximum number of models in P is reached, the oldest GMM is removed from P. Some preliminary experiments were done and the maximum pool size of 20 obtained the best cost-benefit between accuracy and runtime.

7 Experiments and Discussions

Three experiments were realized in this work: (i) comparison with literature works (Section 7.2); (ii) noise filter robustness (Section 7.3); and (iii) proposed method mechanisms analysis (Section 7.4). All experiments were evaluated using the datasets discussed in Section 4 and the metrics discussed in Section 7.1.

7.1 Metrics and Statistical Tests

Cross Validation for Data Streams: traditional cross-validation cannot be applied to data streams because it splits data randomly, making it impossible to see concepts in the order that they should be viewed. Therefore, we use a modified version proposed in [25]. It consists of leaving out an observation every period X to construct the stream. For example, we remove the first element within the first thirty, then remove the first element from the thirty second and so on. For all datasets, 30 runs were executed, i.e. X=30. The element removed corresponds to the order of the execution.

Overall accuracy and G-mean: both are calculated based on the on-line predictions given by the system. Accuracy has been used in several data stream learning studies, such as [25, 7, 6]. G-mean is the geometric mean of the recall on each class, and is a metric independent of the level of class imbalance in the data [11].

Accuracy Over Time (AOT): this is the time series showing the system's accuracy over time [25, 7, 6], where each value represents the accuracy over a batch. For example, if the batch size is X = 250: $batch_1$ = [0 to 250), $batch_2$ = [251 to 500), $batch_3$ = [501 to 750), etc, until the end of the data stream. In order to increase the discriminative power

TABLE 2: Parameters used for the compared approaches. A grid search was executed for the most important parameters and the best was chosen considering the average accuracy across datasets.

Algorithm	Parameters	Grid Search	Synthetic	Real
IGMM-CD	Sigma_ini	[0.5, 1, 2, 5, 10]	0.05	10
	Cver	-	0.01	=
	T	[1, 5, 7, 9, 13]	13	=
Dynse	D	-	25	=
	m (Chunk Size)	[50, 100, 200, 300, 400]	50	=
	M	-	100	=
	k	-	5	=
	CE	-	A Priori	=
	PE	-	Age Based	=
	BC	-	Gaussian Naive Bayes	=
GMM-VRD	m (Chunk Size)	[50, 100, 200, 300, 400]	50	200
	EM it.	-	10	=
	kmax	[2, 4, 6, 8]	2	=
	kDN	-	5	=
	Detector	-	ECDD	=
	c	-	1	=
	w	-	0.5	=
OGMMF-VRD	P	-	20	=
	radius	[10, 15, 20, 25]	20	=
	c	[1, 1.5, 2, 2.5]	1	=
Ensemble Methods	Chunk Size	[50, 100, 200, 300, 400]	50	200
HAT+Drift Detectors	Chunk Size	[50, 100, 200, 300, 400]	50	200

of the metric, the standard deviation between the several accuracies is also reported.

Runtime: the execution time of the approaches was measured in seconds considering the difference between the end time and the start time on a machine with 8GB of ram and processor Intel Xeon E3-1220 v5.

Friedman and Nemenyi Tests: Friedman is a non-parametric statistical hypothesis test that can be used to compare multiple approaches across multiple datasets [26]. It ranks the algorithms and checks whether the null hypothesis that they are all equal can be rejected. If the null hypothesis is rejected, then the Nemenyi post-hoc is used to check which of the approaches is significantly different from each other. Both tests were used in this work for a significance level of α = 0.05.

Wilcoxon Test: this is a non-parametric statistical hypothesis test used in our experiments to evaluate how a specific mechanism has improved over the accuracy of the complete system. This test was used with a significance level of α = 0.05.

7.2 Comparison With Existing Approaches

This experiment aims to validate the performance of the OGMMF-VRD in comparison with literature works. The discussions are divided in Section 7.2.1, literature approaches with separate mechanisms to deal with virtual and real drifts, and Section 7.2.2, other approaches from the concept drift literature.

For the first comparison, we selected: IGMM-CD [7], Dynse [6] and GMM-VRD [1] (see Section 2), as they have explicit separate mechanisms to deal with virtual and real drifts. The parameters used for the approaches are shown in Tbl. 2. An analysis of sensitivity of OGMMF-VRD to the radius (Gaussian Circumference) and c (drift level for EDDM) parameters is presented in the supplementary material, due to space constraints, and shows that these parameters do not significantly affect OGMMF-VRD's accuracy. Its other parameters were the same as GMM-VRD, for fair comparison.

For the second comparison, we selected two groups of approaches (i) ensemble methods and (ii) drift detectors with an incremental algorithm. Methods based on ensembles are: Accuracy Weighted Ensemble (AWE) [27], Adaptive Random Forest (ARF) [28], Leveraging Bagging ensemble classifier (LevBag) [29], and Oza Bagging Ensemble classifier (OzaAS) with and without ADWIN drift detector (OzaAD) [30]. Execept of ARF, all used Gaussian Naive Bayes as the base classifier. Methods based on drift detectors are: ADWIN [31], DDM [21], and EDDM [22], combined with Hoeffding Adaptive Tree classifier (HAT) [32]. All of these approaches are available on-line in the library scikit-multiflow[7] and the chunk size was chosen as part of the hyperparameter tuning procedure based on a grid search presented in Tbl. 2; the other parameters were the default values provided by the library.

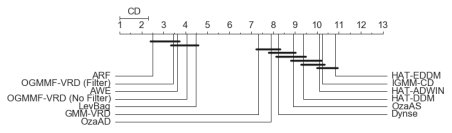
7.2.1 *Approaches With Separate Mechanisms to Deal with Virtual and Real Drifts*

A heat map is shown in Fig. 6 for the compared approaches using both synthetic and real-world datasets. The numerical table with standard deviation is in the supplementary material. To attest the statistical difference of the results, we present in Fig. 4 the rank of Friedman with the Nemenyi post-hoc for all metrics evaluated. For all performance metrics, Friedman rejected the null hypothesis that the algorithms have equal performance at the level of significance of $\alpha = 0.05$. According to the Nemenyi post-hoc tests for average accuracy (Fig. 4a), and G-mean (Fig. 4b), OGMMF-VRD is significantly better than all these approaches (GMM-VRD, IGMM-CD, and Dynse). For runtime (Fig. 4c), OGMMF-VRD is only better than Dynse. To provide a more detailed understanding of these results, Fig. 5 presents in the first row of figures some plots of AOT for GMM-VRD, IGMM-CD, and Dynse. AOTs for the rest of the datasets are presented in the supplementary material due to space constraints.

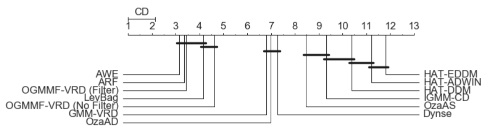
Looking at the dataset which has abrupt virtual drifts (Virtual 9 in Fig. 5a), we can see that IGMM-CD and GMM-VRD had good AOT. This is because these datasets have similarities between their concepts, and despite this approach potentially generating too many Gaussians over time can handle this type of drift properly. Regarding Dynse, its AOT has high peaks indicating that the choice of good classifiers from the pool improves the results. However, in the presence of drifts, its performance declines. OGMMF-VRD presents high accuracy almost all the time. One reason for this is its ability to quickly find out which new regions of space need to be learned.

We now look at the datasets with real drifts: Circles (Fig. 5b), which has virtual/real drifts, and Sine 2 (Fig. 5c), which has abrupt/recurrent shifts, we see that Dynse has the biggest drop in accuracy compared to other approaches. This is explained by two points: (i) in the presence of a new concept, Dynse's accuracy only rises when several classifiers are trained on the new concept; (ii) if its pool is small, over time older classifiers are deleted when a new one is added, so if a similar concept is slow to appear the pool may no longer have a suitable model. Regarding IGMM-CD, it is observed that its accuracy drops dramatically and hardly goes back up. This is because IGMM-CD does not have a

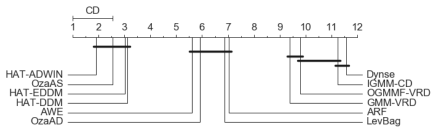
7. https://scikit-multiflow.readthedocs.io/en/stable/index.html



(a) Accuracy

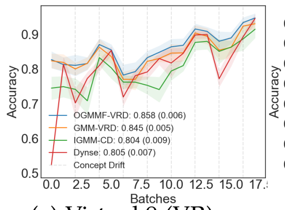


(b) G-mean

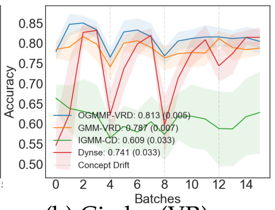


(c) Runtime

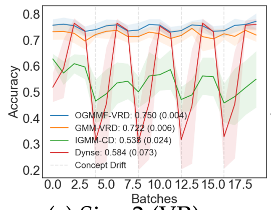
Fig. 4: Friedman ranking for the results obtained for all datasets used in Tbl. 1. Friedman's p-values were 2.07E-311, 2.53E-290 and 4.92E-243, respectively, indicating rejection of the null hypothesis at the level of significance of $\alpha = 0.05$. Any pair of approaches whose distance between them is larger than CD is considered to be different according to the Nemenyi posthoc tests.



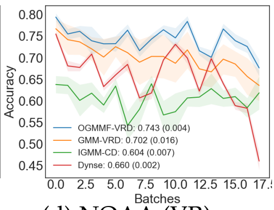
(a) Virtual 9 (VR)



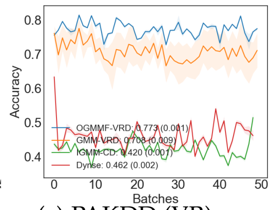
(b) Circles (VR)



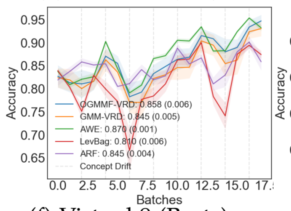
(c) Sine 2 (VR)



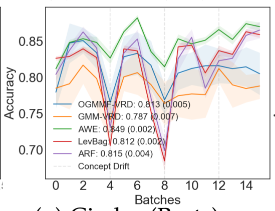
(d) NOAA (VR)



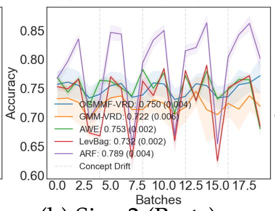
(e) PAKDD (VR)



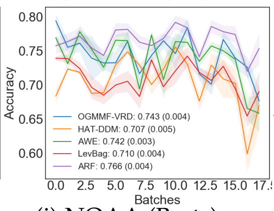
(f) Virtual 9 (Bests)



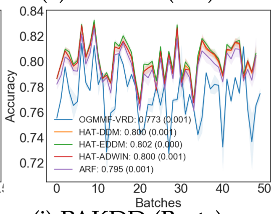
(g) Circles (Bests)



(h) Sine 2 (Bests)



(i) NOAA (Bests)



(j) PAKDD (Bests)

Fig. 5: Average accuracy over time for all methods on each dataset. The first row of figures (Figs. 5a, 5b, 5c, 5d, 5e) shows only the methods that deal with virtual and real drifts (VR). The second row of figures (Figs. 5f, 5g, 5h, 5i, 5j) shows only the five approaches with the best performances for each dataset (Bests). The standard deviation is represented by shadow lines of the same color. Each point represents the accuracy for a batch observations, where 500 was used for synthetic datasets, and 1000 for real datasets.

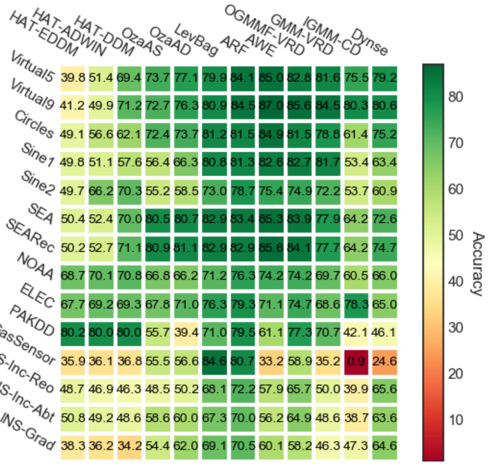
fast reset mechanism, and it forces the system to spend a lot of time with obsolete Gaussians. Regarding GMM-VRD and OGMMF-VRD, it is observed that in the presence of drifts their accuracy does not decrease so much in relation to other approaches, because both have a CDT that informs quickly when their performance is deteriorating, allowing it to be reset to fit the new concept.

Now, we look at the real-world datasets: NOAA (Fig. 5d), and PAKDD (Fig. 5e). For the NOAA, it is observed that Dynse and IGMM-CD sometimes decline their accuracy, which demonstrates that these methods may not be robust to different types of drift. For PAKDD, GMM-VRD and OGMMF-VRD obtained very good AOT, away from that of the other approaches. One reason for this is that both of these methods have a model selection mechanism targeted at improving accuracy in the initialization phase.

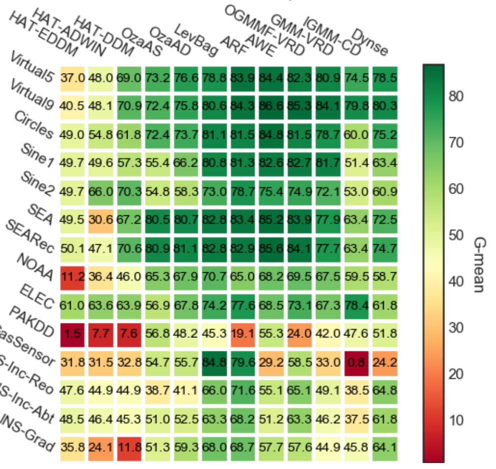
7.2.2 Other Approaches From the Concept Drift Literature

When comparing OGMMF-VRD to other approaches that do not explicitly attempt to deal with real and/or virtual drifts and may rely on different (non-Bayesian) types of base learners, for accuracy and G-mean (Fig. 4a and 4b), OGMMF-VRD was significantly better than all methods, except for AWE, ARF and LevBag. As shown in the supplementary material, these approaches reached the following rankings (and corresponding ranking standard deviations) for accuracy: OGMMF-VRD (Filter): 3.42 (1.59), ARF: 2.48 (1.60), LevBag: 4.48 (1.85), AWE: 3.58 (2.87). For G-mean, the rankings are: OGMMF-VRD (Filter): 3.44 (1.84), LevBag: 4.16 (1.86), AWE: 3.14 (2.42), ARF: 3.36 (2.47). OGMMF-VRD achieved rankings with smaller standard deviations (i.e., more stable rankings) than ARF, AWE and LevBag across datasets. LevBag obtained competitive stability in terms of G-mean, but not in terms of accuracy. ARF obtained competitive stability in terms of accuracy, but not in terms of G-mean. In practice, a high ranking standard deviation means that the model is not consistent with its results, sometimes ranking very well and sometimes ranking very poorly. For instance, AWE achieved good accuracies but performed very poorly on datasets such as GasSensor (Fig 6a). Given the more stable rankings, our approach is more reliable for adoption in practice. Another point observed is that incremental learning methods combined with detectors performed poorly, which indicates that only incremental learning with a reset is not enough to tackle all types of drifts effectively.

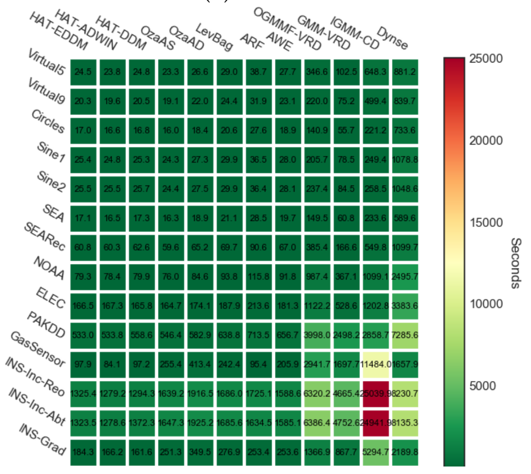
In terms of execution time (Fig. 4c), our approach OGMMF-VRD is costlier. This is partly because our code is not optimized like that of scikit-multiflow algorithms, though it may also be influenced by OGMMF-VRD's mechanisms to deal with different types of drift. To understand better which components of our approach are the most time consuming, our supplementary material reports on experiments that reveal that the most costly mechanisms



(a) Accuracy



(b) G-mean



(c) Runtime

Fig. 6: Overall Performance Average. Cells closer to green/red represent better/worse results.

in descending order were (i) drift detector, (ii) virtual and non-severe real adaptation, and (iii) GMMs pool.

The predictive performance of ensemble methods can be summarized in two main points based on the results obtained for the Sine 2 dataset (Fig. 5h). The first point is that ensemble-based methods have a hard drop in their performance when drifts happen. This is because these methods were not proposed to understand the drift in order to apply a corresponding appropriate strategy, they just reset their knowledge. Although OGMMF-VRD also has a drop in its predictive performance, its recovery is much faster, which shows that the strategies for dealing with virtual and real drifts are efficient. The second point is related to the performance of the ensembles when there is no drift. We observed that the combination of several models allows better performance during periods of stability. In OGMMF-VRD, we use only a single classifier that is inferior in periods of stability compared to a combination of models. Therefore, the combination of models can be an alternative in future work to further improve OGMMF-VRD's predictive performance.

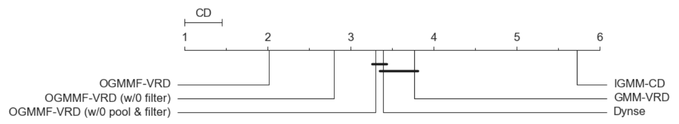


Fig. 7: Friedman ranking for the accuracy obtained in all synthetic datasets with all different noise levels. Friedman's p-value was 9.73E-25 and its ranking is shown from left (best) to right (worst). Any pair of approaches whose distance between them is larger than CD is considered to be different.

7.3 Impact of Noise on Classifier Performance

This experiment aims to determine how well the proposed approach answers RQ2 through its ability to deal with both real and virtual drifts while being robust to noise. For this, we evaluated two components of OGMMF-VRD that help it deal with noise; (i) the noise filter; and (ii) the GMMs pool on the synthetic datasets discussed in Tbl 1 varying their noise level to [5%, 10%, 15%, 20%]. In this experiment, we compared with GMM-VRD, IGMM-CD, and Dynse, which are approaches with separate mechanisms to deal virtual and real drifts. The parameters used are presented in Tbl. 2, but for OGMMF-VRD the batch size used was $m = 200$, because the more observations in the batch, the easier it is to recognize what are the noisy observations.

For the first analysis, Figs. 7, and 8 summarize the results. We observe in Fig. 8 that, regardless of the noise level, both mechanisms helped to improve the performance of OGMMF-VRD, being a significant improvement as shown in Fig. 7. The line graphs for the other datasets and numerical values for this analysis are supplementary material.

The filter becomes more effective when using batches with larger sizes because more observations help recognize when noise observations appear. To show this, we also added in the supplementary material this same experiment with batch size $m = 50$, and we observed that batches larger like $m = 200$ enable the filtering mechanism to achieve significantly better predictive performance than the approach without filtering, whereas a smaller batch of 50 does not improve the results in terms of accuracy. This strategy provides advantages over approaches such as IGMM-CD, which are very sensitive to noise due to updating its models with single examples.

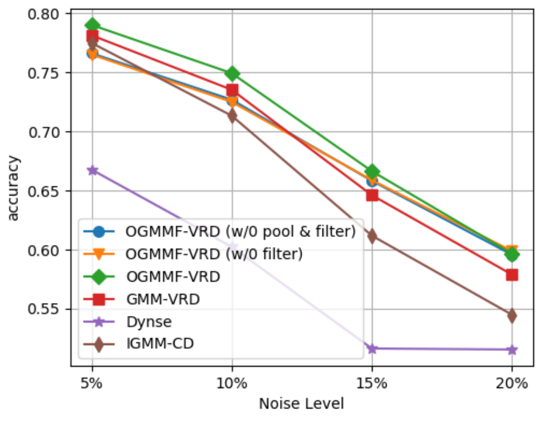


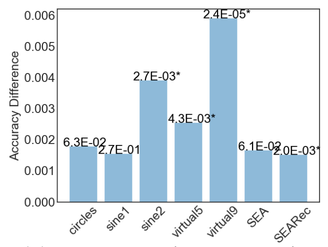
Fig. 8: Line graph with the accuracy of the approaches for different noise levels for the Sine 2 dataset. w/o means the OGMMF-VRD without some mechanism.

The pool is also an important mechanism to increase robustness to false alarms in the drift detections, which are typically caused by noisy examples. It significantly improves OGMMF-VRD's predictive performance, as shown in Fig. 7 and in Fig. 5 of the supplementary material. It leads to increased robustness compared to approaches such as GMM-VRD, which are very sensitive due to the system reset that is undesirably performed upon false alarms, requiring full retraining of the model.

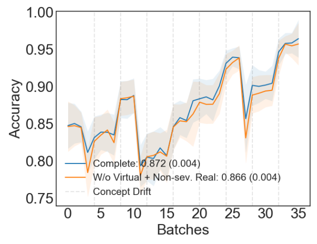
7.4 Impact of Proposed Approach's Mechanisms

This experiment aims to complement the answer to RQ2 by checking whether it is really necessary to have the distinct virtual and real drift mechanisms. It also checks how well the pool used by the proposed approach can harness past knowledge to accelerate adaptation to both virtual and real drifts, answering RQ3. Only the synthetic datasets were used for this analysis, because they enable a better understanding of the behavior of the approaches in relation to each type of drift, unlike the real-world datasets. This experiment was made considering the parameters in Tbl. 2 with $m = 200$, because the more observations for training, the more the system as a whole is favored to obtain a robust model, thus allowing a better discussion of each mechanism's impact on the accuracy.

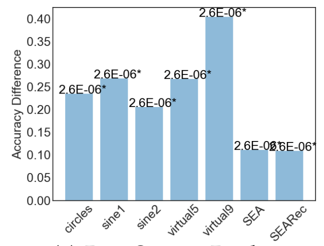
By analyzing the results for the virtual and non-severe real drift adaptation mechanism (Fig. 9a), we observe that there was a statistically significant slight performance gain in 4 out of 7 datasets. Datasets with virtual drifts were the most favored. The SEAREc with the severity of 17.95-49% also improved due to its gradual shift allowing the mechanism to track the drift before it becomes complete, despite this improvement being smaller than that of the datasets involving virtual drifts. Finally, Sine 2 also showed relatively large gains, indicating that this mechanism can also sometimes benefit the system in the presence of high severity real drifts. To visibly analyzing these gains we present in the Fig. 9b the best gain obtained, which was for Virtual 9. In this dataset, changes occur in one class at a time, with observations appearing in another region of the space. This indicates that the proposed mechanism



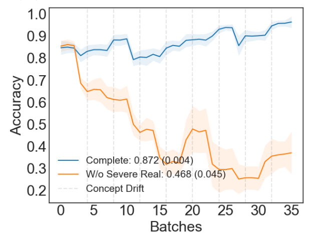
(a) Bar: Virtual + Ns. Real



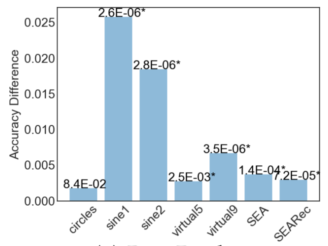
(b) Bar: Virtual + Ns. Real



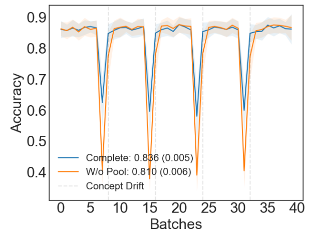
(c) Bar: Severe Real



(d) Best: Severe Real



(e) Bar: Pool



(f) Best: Pool

Fig. 9: Accuracy improvements obtained by each of OGMMF-VRD's mechanisms. Each bar represents the subtraction of the average of the complete system from the average of the system without a given mechanism (e.g. w/o non-severe drift adaptation). The number above each bar represents the p-value obtained by Wilcoxon test over the comparison pair to pair. P-values representing statistically significant difference at the level of $\alpha = 0.05$ are marked by *. The AOT plots represent the dataset with the best improvement in each bar plot. The blue line represents the complete OGMMF-VRD system and the orange line represents the system without the respective mechanism.

to create Gaussians can quickly prevent the system from losing performance. This complements the discussions of RQ2 in that the inclusion of a mechanism to deal with low severity drifts can help improving predictive performance in the presence of virtual drifts.

By analyzing the results for the severe real drift adaptation mechanism (Fig. 9c), we observe that this mechanism statistically significantly improved the performances in all cases, showing that this mechanism can be useful even for virtual drifts. This may be because CDTs can enable the system to quickly react to drifts as soon as they are detected. If the new concept is not too difficult to learn from scratch, enabling this quick reaction can potentially lead to faster adaptation than updating existing Gaussians depending on the adaptation parameters being used. This is unless the drifts have very low severity, in which case the mechanism for virtual and non-severe real drifts would likely still be the most helpful. In the best improvement that was for Virtual 9 (Fig. 9d), we note that not using a CDT can be quite derogatory for model performance since the AOTs compared are very far apart.

Finally, we compared (i) GMM with EDDM without Pool against (ii) GMM with EDDM with Pool. Therefore, the analyzes concern how the use of the Pool improved the predictive performance of the system. On the results for estimating GMMs from Pool in the presence of a concept drift (Fig. 9e), we observed that 6 out of 7 cases significantly improved predictive performance. In SEARec, despite the gain is significant, the improvement was low due to the small pool size. The recurring drifts happen after 4 concepts, by which time the pool has been entirely renewed. The gains were mainly significant in datasets that have drifts that are both real, severe, abrupt and recurrent such as Sine 1 and Sine 2. Looking at the best case improvement, which was Sine 1 (Fig. 9f), we see that in the presence of concept drifts, when estimating new models, system performance degrades less than when having to wait for a lot of data for re-initialize the classifier. These discussions support our RQ3, which states that harnessing the knowledge of past GMM's can accelerate the adaptation in both virtual and real drifts.

8 Conclusion

This paper provides (i) a detailed understanding of the effects that virtual and real drifts have on classifiers' suitability; (ii) the OGMMF-VRD approach, a system for dealing with virtual and real drift simultaneously in classification data streams; and (iii) an unsupervised/supervised methodology with noise filter to train the GMM and achieve better robustness to noise. The main results showed that the proposed approach outperforms approaches with separate mechanisms to deal with virtual and real drifts, has more stable rankings and fewer drops performance in drifts than existing ensemble approaches, thus being more reliable for adoption in practice. With these results, we answer the research questions of this work as follows:

RQ1) What is the difference between the impact of real and virtual drifts on the suitability of classifiers' *learned* decision boundaries and predictive performance over time? (i) Due to the partial representation of the data, some types of classifiers learn incorrect decision boundaries while others based on GMM learn insufficient decision boundaries for the problem. For this reason, when a new observation from the trained class appears in non-trained regions, the classifiers make mistakes. (ii) Dealing with virtual drifts using the same strategy for real drifts wastes useful knowledge that could be used to expedite the classifier adaptation to the new concept especially in the case of Bayesian-based systems. (iii) In the experiments of mechanisms analysis, we saw that not incorporating a strategy to handle virtual drifts significantly drops the system's performance. (iv) Real drifts change the *true* decision boundaries of the problem causing a significant drop in the classifier performance, but if the drift is non-severe, part of the knowledge can also be harnessed.

RQ2) How to deal with both virtual and real drifts while achieving robustness to noise? (i) Using a noise filter allowed us to avoid adapting to observations that would cause problems in our system. The experiments showed that this mechanism with larger batches improved significantly the results. (ii) Using pertinence threshold allowed us to create Gaussians quickly to properly deal with virtual drifts. (iii) Using *on-line* learning has enabled us to maintain existing Gaussians by modeling them as new distributions arrived. (iv) Using the pool enabled us to be robust to false alarms in drift detections caused by noisy examples. (v) Experiments have shown that combinations of these techniques statistically improved system performance on both virtual and real drifts.

RQ3) How to best harness knowledge gained from similar concepts to accelerate adaptation to both virtual and real drifts? (i) Saving older GMM's in a pool allowed the system to be able to choose the best classifier in the presence of similar concepts regardless the type of drift. (ii) The results showed that this mechanism led to statistically significant improvements on the system performance.

Although the results have shown the competitive performance of OGMMF-VRD, our proposals have some limitations, which should be addressed in future work: (i) it does not perform well on datasets with challenging class imbalanced distributions; (ii) It has limited performance because it is based on a single classifier; (iii) it discards useful knowledge when resetting the entire system; and (iv) it does not tackle verification latency.

Acknowledgment

The authors would like to thank CNPq, FACEPE and CAPES (Grant No. 88887.588731/2020-00), for their financial support. Leandro Minku was supported by EPSRC Grant No. EP/R006660/2.

References

[1] G. H. Oliveira, L. L. Minku, and A. L. Oliveira, "Gmm-vrd: A gaussian mixture model for dealing with virtual and real concept drifts," in *IEEE IJCNN*, 2019, pp. 1–8.

[2] Y. Song, J. Lu, H. Lu, and G. Zhang, "Fuzzy clustering-based adaptive regression for drifting data streams," *IEEE Transactions on Fuzzy Systems*, vol. 28, no. 3, pp. 544–557, 2019.

[3] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM CSUR*, vol. 46, no. 4, p. 44, 2014.

[4] L. L. Minku and X. Yao, "Ddd: A new ensemble approach for dealing with concept drift," *IEEE TKDE*, vol. 24, no. 4, pp. 619–633, 2012.

[5] I. Khamassi, M. Sayed-Mouchaweh, M. Hammami, and K. Ghédira, "Discussion and review on evolving data streams and concept drift adapting," *Evolving systems*, vol. 9, no. 1, pp. 1–23, 2018.

[6] P. R. Almeida, L. S. Oliveira, A. S. Britto Jr, and R. Sabourin, "Adapting dynamic classifier selection for concept drift," *EXPERT SYST APPL*, vol. 104, pp. 67–85, 2018.

[7] L. S. Oliveira and G. E. Batista, "Igmm-cd: a gaussian mixture classification algorithm for data streams with concept drifts," in *IEEE BRACIS*, 2015, pp. 55–61.

[8] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning in nonstationary environments: A survey," *IEEE CIM*, vol. 10, no. 4, pp. 12–25, 2015.

[9] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Woźniak, "Ensemble learning for data stream analysis: A survey," *Elsevier IF*, vol. 37, pp. 132–156, 2017.

[10] H. M. Gomes, J. P. Barddal, F. Enembreck, and A. Bifet, "A survey on ensemble learning for data stream classification," *ACM CSUR*, vol. 50, no. 2, p. 23, 2017.

[11] S. Wang, L. L. Minku, and X. Yao, "A systematic study of online class imbalance learning with concept drift," *IEEE TNNL*, vol. 29, no. 10, pp. 4802–4821, 2018.

[12] G. I. Webb, R. Hyde, H. Cao, H. L. Nguyen, and F. Petitjean, "Characterizing concept drift," *KDD*, vol. 30, no. 4, pp. 964–994, 2016.

[13] K. Yamauchi, "Incremental learning and model selection under virtual concept drifting environments," in *IEEE IJCNN*, 2010, pp. 1–8.

[14] J. Grim, "Em cluster analysis for categorical data," in *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*. Springer, 2006, pp. 640–648.

[15] A. Pesaranghader, H. Viktor, and E. Paquet, "Reservoir of diverse adaptive learners and stacking fast hoeffding drift detection methods for evolving data streams," *Machine Learning*, vol. 107, no. 11, pp. 1711–1743, 2018.

[16] L. L. Minku, A. P. White, and X. Yao, "The impact of diversity on online ensemble learning in the presence of concept drift," *IEEE TKDE*, vol. 22, no. 5, pp. 730–742, 2009.

[17] V. M. Souza, D. M. dos Reis, A. G. Maletzke, and G. E. Batista, "Challenges in benchmarking stream learning algorithms with real-world data," *Data Mining and Knowledge Discovery*, vol. 34, no. 6, pp. 1805–1858, 2020.

[18] G. J. Ross, N. M. Adams, D. K. Tasoulis, and D. J. Hand, "Exponentially weighted moving average charts for detecting concept drift," *Elsevier PRL*, vol. 33, no. 2, pp. 191–198, 2012.

[19] E. S. Page, "Continuous inspection schemes," *Biometrika*, vol. 41, no. 1/2, pp. 100–115, 1954.

[20] A. Pesaranghader and H. L. Viktor, "Fast hoeffding drift detection method for evolving data streams," in *ECML PKDD*, 2016, pp. 96–111.

[21] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Brazilian symposium on artificial intelligence*. Springer, 2004, pp. 286–295.

[22] M. Baena-Garcıa, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavalda, and R. Morales-Bueno, "Early drift detection method," in *KDD*, vol. 6, 2006, pp. 77–86.

[23] P. M. Engel and M. R. Heinen, "Incremental learning of multivariate gaussian mixture models," in *IEEE BRACIS*, 2010, pp. 82–91.

[24] F. N. Walmsley, G. D. Cavalcanti, D. V. Oliveira, R. M. Cruz, and R. Sabourin, "An ensemble generation methodbased on instance hardness," *arXiv preprint arXiv:1804.07419*, 2018.

[25] Y. Sun, K. Tang, L. L. Minku, S. Wang, and X. Yao, "Online ensemble learning of data streams with gradually evolved classes," *IEEE TKDE*, vol. 28, no. 6, pp. 1532–1545, 2016.

[26] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *JMLR*, vol. 7, no. Jan, pp. 1–30, 2006.

[27] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003, pp. 226–235.

[28] H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Enembreck, B. Pfharinger, G. Holmes, and T. Abdessalem, "Adaptive random forests for evolving data stream classification," *Machine Learning*, vol. 106, no. 9-10, pp. 1469–1495, 2017.

[29] A. Bifet, G. Holmes, and B. Pfahringer, "Leveraging bagging for evolving data streams," in *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 2010, pp. 135–150.

[30] N. C. Oza, "Online bagging and boosting," in *2005 IEEE international conference on systems, man and cybernetics*, vol. 3. Ieee, 2005, pp. 2340–2345.

[31] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing," in *Proceedings of the 2007 SIAM international conference on data mining*. SIAM, 2007, pp. 443–448.

[32] A. Bifet and R. Gavaldà, "Adaptive learning from evolving data streams," in *International Symposium on Intelligent Data Analysis*. Springer, 2009, pp. 249–260.

Gustavo H. F. M. Oliveira Ph.D. student, and M.Sc in Computer Science at the Federal University of Pernambuco (UFPE). Degree in Computing from the University of Pernambuco (UPE). Member of the subcommittee of Conference Activities and Communications of the IEEE Computational Intelligence Society.

Leandro L. Minku is a Senior Lecturer at the School of Computer Science, University of Birmingham, UK. Prior to that, he was a Lecturer at the University of Leicester, UK. He received the PhD degree in Computer Science from the University of Birmingham, UK, in 2010. Dr. Minku's main research interests are machine learning for non-stationary environments / data stream mining, class imbalance learning, ensembles of learning machines and computational intelligence for software engineering. Among other roles, Dr. Minku is Associate Editor-in-Chief for Neurocomputing and Associate Editor for IEEE Transactions on Neural Networks and Learning Systems.

Adriano L. I. Oliveira is an Associate Professor at the Center for Informatics of Federal University of Pernambuco, Brazil, since 2011. He was a Visiting Professor at Ecole de Technologie Superieure (ETS, Université du Québec, Montréal, Canada) from 2018 to 2019. He received his Ph.D. degree in Computer Science from the Federal University of Pernambuco in 2004. He is a Senior Member of the IEEE and has published over 140 articles in scientific journals and conferences and one book.