UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

GUSTAVO HENRIQUE FERREIRA DE MIRANDA OLIVEIRA

TACKLING VIRTUAL AND REAL CONCEPT DRIFTS VIA ADAPTIVE GAUSSIAN MIXTURE MODEL APPROACHES

Recife

2022

GUSTAVO HENRIQUE FERREIRA DE MIRANDA OLIVEIRA

TACKLING VIRTUAL AND REAL CONCEPT DRIFTS VIA ADAPTIVE GAUSSIAN MIXTURE MODEL APPROACHES

A Ph.D. Thesis presented to the Informatics Center of Federal University of Pernambuco in partial fulfillment of the requirements for the degree of Philosophy Doctor in Computer Science.

Computer Science

Advisor Adriano Lorena Inácio de Oliveira (Universidade Federal de Pernambuco, Brazil)

Co-advisor Leandro Lei Minku (University of Birmingham, UK)

Recife

2022

Catalogação na fonte
Bibliotecária Nataly Soares Leite Moro, CRB4-1722

O48t Oliveira, Gustavo Henrique Ferreira de Miranda
Tackling virtual and real concept drifts via adaptive gaussian mixture model approaches / Gustavo Henrique Ferreira de Miranda Oliveira. – 2022.
130 f.: il., fig., tab.

Orientador: Adriano Lorena Inácio de Oliveira.
Tese (Doutorado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2022.
Inclui referências.

1. Inteligência computacional. 2. Modelo de mistura gaussiana. 3. Fluxos de dados. 4. Mudança de conceito virtual. 5. Mudança de conceito real. I. Oliveira, Adriano Lorena Inácio de (orientador). II. Título

006.31 CDD (23. ed.) UFPE - CCEN 2022 – 35

Gustavo Henrique Ferreira de Miranda Oliveira

"TACKLING VIRTUAL AND REAL CONCEPT DRIFTS VIA ADAPTIVE GAUSSIAN MIXTURE MODEL APPROACHES"

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação. Área de Concentração: Inteligência Computacional

Aprovado em: 24/02/2022.

____

Orientador: Prof. Dr. Adriano Lorena Inacio de Oliveira

Eu dedico esse trabalho aos meus pais, Gilberto e Luci, que sempre me orientaram, encorajaram e incentivaram a sempre continuar estudando.

AGRADECIMENTOS

Eu gostaria de agradecer profundamente aos meus orientadores, Adriano Lorena e Leandro Minku. Sem a ajuda de vocês, esse documento não teria sido escrito. Sem as ideias de vocês, eu não teria conseguido publicar em conferências e revistas. Sem o auxílio de vocês, eu não teria experienciado minha primeira viagem internacional. Sem a paciência de vocês, eu não teria desenvolvido meu inglês. Muito obrigado por toda a infinita paciência, por todo o conhecimento ensinado e por todas as conversas amigáveis nas nossas reuniões.

A professor Adriano, meu muito obrigado pela parceria ao longo desses 6 anos. Obrigado por ter me aceitado como aluno de mestrado e doutorado e por ter acreditado que eu conseguiria. Obrigado por reconhecer as minhas limitações e por me guiar de modo superá-las. Obrigado por me apresentar a excelentes pesquisadores que tanto contribuiram para a minha formação.

A Leandro, meu muito obrigado pela parceria ao longo desses 5 anos, iniciando no mestrado e sendo meu coorientador no doutorado. Cada reunião com você foi uma aula. Muito obrigado por todas as suas ideias. Ideias tão boas que resultaram nos nossos três artigos. Obrigado por ter cedido um pouco do seu corrido tempo para me ajudar a melhorar no meu inglês. Obrigado por ter me ajudado tanto a conseguir passar no processo seletivo do doutorado sanduiche. Uma pena que o COVID não permitiu, mas o processo para isso valeu muito a pena!!!

Eu gostaria de agradecer ao centro de informática e aos professores George Darminton, Aluizio e Adriano por todo o conhecimento técnico ensinado nas disciplinas que pode ser aplicado na construção dessa tese.

Eu gostaria de agradecer aos meus amigos do centro de informática, especificamente para o grupo Trash: Leandro, Reinaldo, Walter, Eraylson, Flávio, Múcio, Diogo, Julio, Rafael e Vander. Os estudos no laboratório se tornaram prezerosos devido as nossas brincadeiras. Eu gostaria de fazer um agradecimento especial a Walter, Reinaldo e Flávio por terem me ajudado na parte profissional. A ajuda de vocês me ajudou a ser um professor durante o doutorado. Meu muito obrigado!

Eu gostaria de agradecer a minha família, Gilberto, Luci, Luciana, Augusto, Gabriela, Kariny e Francisco pelo suporte emocional. Agradeço a minha noiva Hellen Carolina, por toda a paciência, apoio e esperas ao longo desses anos. Você teve forte participação nisso. Agradeço aos meus amigos da residência das palmeiras, Emanuel, Northon, Wellington, Avishek e Renê por todas as conversas acadêmicas e pelas trocas de experiências ao longo desses anos em

Recife. Agradeço também aos meus tios Joseval, Josemir e Flávia, por todos os reencontros da família em Recife. Eles foram importantíssimos para me sentir amparado nessa cidade. Agradeço aos meus tios Rogério e Jailton por sempre serem prestativos comigo e por me ajudarem nas viagens para Recife quando eu ainda não possuia carro.

Novamente, a todos, o meu muito obrigado!!!

"O cavalo prepara-se para o dia da batalha, mas do Senhor vem a vitória." (Provérbios, 21:31)

RESUMO

As aplicações do mundo real têm lidado com uma grande quantidade de informações, que chegam de forma contínua e sequencialmente ao longo do tempo, caracterizadas como fluxos de dados. Esse tipo de dado desafia os algoritmos de aprendizado de máquina devido à mudança de conceito. A mudança de conceito é uma mudança na distribuição de probabilidade conjunta do problema e tem duas variações: a mudança virtual que afeta a distribuição de probabilidade incondicional $p(x)$; e a mudança real que afeta a distribuição de probabilidade condicional $p(y|x)$. Essas mudanças podem ocorrer separadamente e simultaneamente e ter impactos diferentes no desempenho do classificador. Os trabalhos da literatura geralmente não compreendem bem esses aspectos. Devido a isso, se concentram apenas nas mudanças reais, por que elas causam degradação direta no desempenho do classificador. No entanto, desvios virtuais também podem causar essa degradação de forma indireta. Novas observações podem chegar em uma região não treinada pelo classificador, forçando-o a confundir sua verdadeira classe, assim cometendo erros de classificação. O ideal seria ter classificadores que entendam que tipo de mudança ocorre em determinado momento para ativar estratégias apropriadas para lidar com este desafio. Este processo é chamado de entendimento da mudança. Como as abordagens da literatura não compreendem bem os diferentes impactos causados pelas mudanças virtuais e reais, o seu desempenho fica limitado. Motivados por isso, propomos três abordagens para entender o tipo da mudança e usar a estratégia correta para se adaptar, sendo elas: (i) Gaussian Mixture Model For Dealing With Virtual and Real Concept Drifts (GMM-VRD); (ii) On-line Gaussian Mixture Model With Noise Filter For Handling Virtual And Real Concept Drifts (OGMMF-VRD); e (iii) Gaussian Local Drift Detector for Drift Understanding (GLDD-DU). Essas abordagens atualizam e criam Gaussians *on-line* para lidar com mudanças virtuais, usam detectores de mudança para reinicializar o conhecimento do sistema para lidar com mudanças reais e recuperam modelos do pool para acelerar a adaptação a um novo conceito. Os principais resultados mostraram que todas as abordagens apresentam desempenho competitivo, mas o OGMMF-VRD foi mais consistente ao longo dos conjuntos de dados, apresentando melhor desempenho.

Palavras-chaves: modelo de mistura gaussiana; fluxos de dados; mudança de conceito virtual; mudança de conceito real; entendimento da mudança.

ABSTRACT

Real-world applications have been dealing with large amounts of data, which come continuously and sequentially over time, characterized as data streams. This type of data challenges machine learning algorithms due to concept drift. Concept drift is a change in the joint probability distribution of the problem and has two variations: virtual drift that affects the unconditional probability distribution $p(x)$; and real drift that affects the conditional probability distribution $p(y|x)$. These drifts can happen separately or simultaneously and can have different impacts on classifiers' suitability. Existing work focuses mainly on real drifts, typically dealing with them by adopting mechanisms to react to performance degradation, which may result from such drifts. However, virtual drifts can also cause such performance degradation. Adopting mechanisms to deal with real drifts when the performance degradation occurs as a result of virtual drifts may not be ideal, hindering classifiers' performances. Classifiers should ideally understand which type of drift occurs to activate appropriate strategies to deal with this challenge. This process is called drift understanding. Motivated by this, we propose three approaches to understand the drift type and use the correct strategy to adapt to it, namely: (i) GMM-VRD; (ii) OGMMF-VRD; and (iii) GLDD-DU. These approaches update and create Gaussians in an *on-line* manner to handle virtual drifts, use concept drift detection to reinitialize the system to handle real drifts, and retrieve models from a pool to speed up adaptation to a new concept. The main results show that these approaches are able to achieve competitive performance, but OGMMF-VRD was more consistent across different datasets. It thus performed better than the others.

Keywords: gaussian mixture model; data streams; virtual concept drift; real concept drift; drift understanding.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AIC	Akaike Information Criterion
AOS-ELM	Adaptive Online Sequential Extreme Learning Machine For Concept Drift Tackling
BIC	Bayesian Information Criterion
CDT	Change Detection Test
CSTH	Continuous Stirred Tank Heater
DCS	Dynamic Classifier Selection
Dynse	Dynamic Selection Based Drift Handler
ECDD	Exponentially Weighted Moving Average Charts
EM	Expectation-Maximization
FGMM	Finite Gaussian Mixture Model
GLDD-DU	Gaussian Local Drift Detector for Drift Understanding
GMM	Gaussian Mixture Model
GMM-VRD	Gaussian Mixture Model For Dealing With Virtual and Real Concept Drifts
ICA GMM	ICA Gaussian Mixture Model
ICI	Intersection of Confidence Intervals
IGMM-CD	Incremental Gaussian Mixture Model For Concept Drifts
JIT	Just in Time Classifier for Recurrent Concepts
KDN	K-Disagreeing Neighbors
KME	Knowledge-Maximized Ensemble
OGMMF-VRD	On-line Gaussian Mixture Model With Noise Filter For Handling Virtual And Real Concept Drifts
PCA	Principal Component Analysis
PSL	Partial Least Square
SPE	Squared Prediction Error

TEP	Tenesse Eastman Problem
WEA	Weight Estimation Algorithm

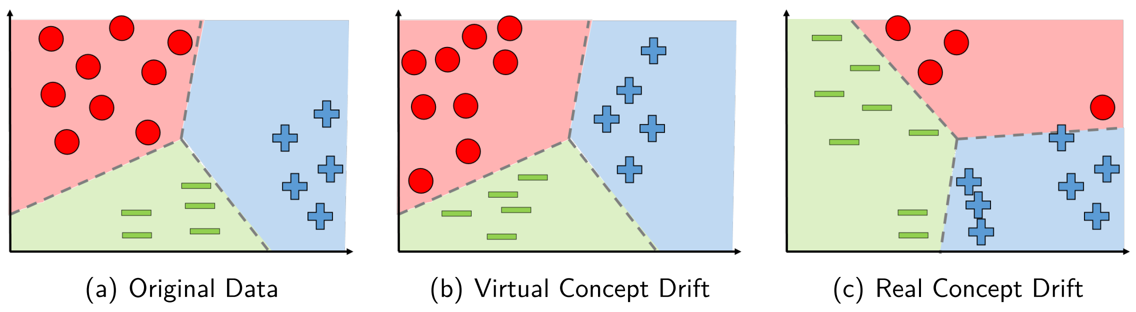
SUMÁRIO

1 INTRODUCTION

As technology advances, there has been an increase in the amount of data collected from different sources (RUANO-ORDAS; FDEZ-RIVEROLA; MENDEZ, 2018). These data are often generated as data streams, which means that they arrive continuously and sequentially over time and could evolve due to the dynamics of real-world activities (IDREES et al., 2020). Typical applications that need to tackle data streams include (i) Real-time classification of gaseous substances using chemical sensors, (ii) weather forecast in which data changes from season to season, (iii) forecast of rising or falling electricity prices, and (iv) credit evaluation task. These applications offer challenges for traditional machine learning algorithms due to their non-stationary nature (XIAO et al., 2019). That is because machine learning models are often trained with historical data from a stationary distribution. Since the assumption of stationarity is not true, drifts in data distribution happen and make modeling done in these concepts inefficient to deduce future behaviors. This problem is called a concept drift (GAMA et al., 2014), and typically impairs the performance of machine learning models, making them obsolete over time (CANO; GÓMEZ-OLMEDO; MORAL, 2019; MINKU; YAO, 2012).

Concept drift can be subdivided into two types: virtual drift and real drift (GAMA et al., 2014), both illustrated in Fig. 1. 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.

Figura 1 – Types of concept drift. The points presented are training observations, whereas the rectilinear gray lines are the *true* decision boundaries of the problem.


(a) Original Data (b) Virtual Concept Drift (c) Real Concept Drift

Source: Author.

Real drifts change the *true* decision boundaries of the problem, directly affecting the classifiers' performance, since their learned decision boundaries are no longer fully valid (GAMA et al., 2014). Real drifts can happen at different levels of severity in the spatial perspective of the data. If the drift is non-severe, it means that only a local region of the feature space has been affected, and part of the classifier's decision boundaries are still valid. However, if the drift is severe, a large portion of the feature space has been affected (MINKU; WHITE; YAO, 2009), requiring most of the old learned decision boundaries to be updated (SANTOS; BARROS; JR, 2019).

Virtual drifts, when not occurring concurrently with real drifts, do not change the *true* decision boundaries of the problem, but change the unconditional data distribution of the problem (GAMA et al., 2014), potentially affecting the suitability of the *learned* decision boundaries. Thus, if the classifier's decision boundaries are not adjusted to the new distribution, they will not represent the class boundaries well, potentially decreasing predictive performance over time (KHAMASSI et al., 2018). Nevertheless, as the *true* decision boundaries do not change, a significant part of the classifier's learning remains valid, meaning that virtual drifts can be typically less severe drifts (OLIVEIRA; MINKU; OLIVEIRA, 2019). So, if the classifier is reset to tackle this kind of drift, it loses all its learned decision boundaries, which were still partly useful and have the potential to accelerate the adaptation of the classifier to a new concept.

1.1 MOTIVATION

In general, 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 (KHAMASSI et al., 2018). The existing approaches for dealing with concept drift typically ignore that both drifts can happen simultaneously and do not address the effects caused by virtual concept drifts. 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 used for real drifts (KHAMASSI et al., 2018). A common strategy to adapt to a new (previously unseen) concept is to create a new classifier to learn it. This strategy is ideal for abrupt real concept drifts where the new concepts hardly do not share similarities with the old ones. 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 (GAMA et al., 2014), 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. This is not ideal because learn from scratch is associated to the use of an outdated model until sufficient data is available for retraining; the delay caused by the collection of new data forces the system to use an obsolete model which in turn degrades its predictive performance.

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.

The process of explicitly understanding “when", “how", and “where"the drift affected the feature space is called drift understanding (LIU et al., 2017). Drift understanding enables the choice of more appropriate strategies since we know the impacts caused by the concept drift in the classifier’s knowledge. For instance, in virtual drifts, the system may need to learn new areas of the feature space after its training phase, whereas useful past knowledge could potentially still be maintained. In real non-severe drifts, the system needs to update its learned decision boundaries, but some of its past knowledge may also still remain valid. Excluding such valid past knowledge could slow down adaptation to the new concept. In real severe drifts, the system may need to be reset quickly to keep up with the evolution of the new concept. Dealing with drifts properly implies maintaining useful knowledge and quickly removing obsolete knowledge, in this way speeding up the adaptation and consequently improving predictive performance.

A potential approach to deal with this type of problem is the Gaussian Mixture Model (GMM). GMM have been adopted in several pattern recognition problems, e.g., (LEE; LEWICKI; SEJNOWSKI, 2000), and non-stationary problems for a long time, e.g., (CHEN; ZHANG, 2010) (DITZLER; POLIKAR, 2011) (OLIVEIRA; BATISTA, 2015). It can also represent complex distributions and model both real (OLIVEIRA; BATISTA, 2015), and virtual drifts (CHOI; PARK; LEE, 2004). GMM has calculations of pertinence and classification that enable us to adopt different strategies for handling virtual and real drifts.

The pertinence calculations infer the degree of belonging of a new observation to a previously trained data distribution. If the observation has a low pertinence to this data, it may come from a new distribution. Then, depending on the degree size, we can decide when to create or update a Gaussian. Updating a Gaussian would be efficient when the observation still has some pertinence, as it may indicate that the observation is close to an existing distribution. On the other hand, creating a Gaussian would be helpful when the observation has zero pertinence, demonstrating that it came from an untrained distribution.

The classification calculations make it possible to generate metrics that evaluate the model's performance for supervised data. These metrics can be monitored by drift detectors that report when a real drift took place. At this point, we can reset the model using observations of the new concept.

1.2 PROBLEM DEFINITION

A data stream can be defined as a potentially infinite sequence of observations $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_t, y_t), \cdots\}$, where $\mathbf{x}_t \in \mathcal{X}$ is a d-dimensional vector of input attributes, $y_t \in \mathcal{Y}$ is a categorical output attribute, $\mathcal{X}$ is the input space and $\mathcal{Y}$ is the output space. Each observation $(\mathbf{x}_t, y_t)$ is drawn from a joint probability distribution $P_t(\mathbf{x}, y) = P_t(y|\mathbf{x})P_t(\mathbf{x})$, 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.

In a problem where it is possible to store data, on-line supervised learning in data streams consists of creating a model $f_t : \mathcal{X} \rightarrow \mathcal{Y}$ 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 online supervised learning algorithms 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 (WEBB et al., 2016). Drifts can happen when $P_t(\mathbf{x}) \neq P_{t+\Delta}(\mathbf{x})$, $P_t(y|\mathbf{x}) \neq P_{t+\Delta}(y|\mathbf{x})$, or both. Drifts affecting only $P(\mathbf{x})$ are categorised as virtual drifts, whereas drifts affecting $P(y|\mathbf{x})$ are categorised as real drifts (YAMAUCHI, 2010). Only real drifts affect the *true* decision boundaries of the problem.

Drifts in $P(\mathbf{x})$ and $P(y|\mathbf{x})$ are categorised as *non-severe* or *severe* according to the severity criterion proposed by (MINKU; WHITE; YAO, 2009). Severity is the percentage of the input space that had its class changed upon completion of the drift (MINKU; WHITE; YAO, 2009; GOLDENBERG; WEBB, 2019). It represents the difference between the old and new $P(y|\mathbf{x})$.

According to this definition, drifts affecting only $P(\mathbf{x})$ are always *non-severe* since they do not change the class of the instances, even though they locally affect the distribution of the input attributes ($\mathcal{X}$), which can affect the suitability of a learned classifier. Drifts in $P(y|\mathbf{x})$ could be *non-severe* or *severe*, because this kind of drift changes the relationship between the output space ($\mathcal{Y}$) and the input space ($\mathcal{X}$). The portion of the feature space that has its class changed defines the severity level.

Non-severe drifts require that parts of the classifier's decision boundaries be removed and updated according to the modified region. Retaining portions of the learned decision boundaries that remain useful allows the classifier to continue responding well to these areas, thus not reducing predictive performance. However, *severe* drifts force the classifier to forget all its decision boundaries since they cannot represent the new data distribution well. If this reset process is not fast, the system degrades quickly, as the classifier will make predictions based on obsolete knowledge.

1.3 AIMS

The main aim of this research is to provide the first in-depth analysis of the differences between the impact caused by virtual and real drifts on the suitability of Bayesian approaches. Based on such understanding, we aim to investigate and propose adaptive learning systems to better deal with problems where there may be both real and virtual drifts. Motivated to overcome the main problems described in Section 1.2, the proposed methods should be:

1. capable of handling drifts that occur in $P(\mathbf{x})$, since we are going to adjust the system knowledge *on-line* in order to track the new distribution.
2. capable of handling drifts that occur in $P(y|\mathbf{x})$, updating the system whenever its error degrades.
3. capable of being robust to noise.
4. transparent to the user, as we will report when drift occur.

The proposed approaches are dedicated to problems where an initial portion of observations can support the learning process. Drift detectors will monitor the relationship between input and output of attributes through the predictive model. Also, a noise filter will be used to separate observations without noise from observations with noise.

1.4 RESEARCH QUESTIONS AND HYPOTHESES

To achieve the goal of building adaptive learning systems capable to overcome the problems mentioned in Section 1, this research is guided by some research questions:

1. **RQ1) What is the difference between the impact of virtual and real drifts on the suitability of Bayesian 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 hypothesize 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.

2. **RQ2) How to efficiently deal with virtual drifts while preserving the ability to deal with real drifts when using GMM?** We hypothesize that combining the probabilistic inferences of the GMM can be a way to handle each type of drift appropriately. Probabilistic pertinence may help decide whether to update or create new Gaussians to tackle virtual drifts. Classification probabilities can generate errors that can be monitored by a detector and thus inform the system when to reset to handle real drifts.

3. **RQ3) How to deal with both virtual and real drifts while achieving robustness to noise?** 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. Besides that, we explore the potential of GMM to enable different strategies to be used to tackle these different types of drift. GMM has pertinence inferences 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.

4. **RQ4) 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 (ALMEIDA et al., 2018). 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.

5. **RQ5) How can we stimulate the generation of knowledge that is likely to help adaptation to new concepts when using GMMs?** The generation of diverse models increases the system's ability to have models with better capabilities to deal with new concepts. This is essential for a GMM-based system. The GMM is a statistical method in which different executions generate the same model; thus, it is difficult to generate models with different capacities to adapt to the new concept. To solve this, we can use subsamples with different training examples to train these models. These subsamples can be generated using an instance hardness metric considering various thresholds. Thus, using subsamples with different data perspectives, we can populate a pool with GMM models that may have different capabilities to adapt to concept drift. Therefore, during the training stage, we can select the best-suited GMM for the new concept, which is the best predictive performance model.

6. **RQ6) How do we use drift understanding to update only the portions of the learned decision boundaries that were affected by the concept drift?** We hypothesize that a way to find out what part of the characteristic space was affected by the drift would be to monitor the performance of each Gaussian in GMM separately using a drift detector. Each Gaussian represents the modeling of a different region in the feature space. Thus, if a Gaussian shows degradation over time, we will understand that its region has been affected, and in that way, we can replace it with a better one from the pool while not deleting knowledge that is still relevant from the model.

1.5 DOCUMENT STRUCTURE

To be able to implement the discussions mentioned, we organized the rest of this Document as follows:

In Chapter 2, we contextualize some concepts for the reader understand better our research problem and our proposed approaches. We organize this Chapter in the following concepts: 2.2: Types of concept drifts, both virtual and real, and their effects in the classifier performance; 2.3: Drift Shift Level, how they can be seen in the different types of concept drifts; 2.3.4: Drift Repeat Mode, the ways that drifts can reappear over time; 2.4: Drift Severity, how a drift spreads the data distribution in the feature space; and 2.1: The GMM Learning Processes, supervised and unsupervised respectively.

In Chapter 3, we present the main literature studies that relate to this work and their limitations in order to present how our proposal differs from them and fills the gaps encountered. We organize this Chapter in the following Subsections: 3.1: We discuss how GMM was explored in stationary data streams; 3.2; We discuss how GMM was explored in non-stationary data streams, addressing works that deal with only virtual and virtual and real concept drifts; 3.3: We discuss other literature works that proposed to deal separately with virtual and real concept drifts; and 3.4: We discuss the approaches that apply drift understanding to deal with concept drift.

In Chapter 4, we answer our RQ1, presenting an analysis of the impacts of virtual and real drifts on Bayesian classifiers' suitability. This Chapter provides the foundation for proposing novel approaches able to more efficiently and effectively deal with both types of drifts at the same time.

In Chapter 5, we present our proposed approaches, partly answering RQ2, RQ3, RQ4, RQ5 and RQ6. We organize this Chapter in three Subsections: 5.1: We present the GMM-VRD; 5.2: Its extension the OGMMF-VRD; and 5.3: We present the GLDD-DU.

In Chapter 6, we present the setup of our experiments and their results, completing the answer to RQ2, RQ3, RQ4, RQ5, and RQ6. Five experiments were realized: (i) comparison with literature works (Section 6.3); (ii) noise filter robustness (Section 6.4); (iii) impact of virtual and real drifts mechanisms (Section 6.5); (iv) impact of drift understanding mechanisms (Section 6.6); and (v) proposed methods' parameter sensitivity analysis (Section 6.7).

In Chapter 7, we summarize the main results and contributions obtained in this Document. To create this document we used part of the material published in our following papers:

1. **OLIVEIRA, GUSTAVO. H. F. M.; MINKU, LEANDRO L.; OLIVEIRA, ADRIANO L. I.** GMM-VRD: A Gaussian Mixture Model For Dealing With Virtual and Real Concept Drifts. In: IEEE. International Joint Conference on Neural Networks, 2019. DOI: 10.1109/IJCNN.2019.8852097.

2. **OLIVEIRA, GUSTAVO. H. F. M.; MINKU, LEANDRO L.; OLIVEIRA, ADRIANO L. I.** Tackling Virtual and Real Concept Drifts: An Adaptive Gaussian Mixture Model Approach. IEEE Transactions on Knowledge and Data Engineering, 2021. DOI: 10.1109/TKDE.2021.3099690.

3. **OLIVEIRA, GUSTAVO. H. F. M.; MINKU, LEANDRO L.; OLIVEIRA, ADRIANO L. I.** An Adaptive Gaussian Mixture Model Based on Drift Understanding to Tackle Virtual and Real Concept Drifts. Machine Learning, 2021. **(Submitted)**

2 FUNDAMENTALS

In this thesis, we investigate how GMM can be applied to deal with concept drift in classification problems. Therefore, in this Chapter, we present some fundamentals that serve to understand this topic. This chapter is organized as follows: Section 2.1 introduces GMM unsupervised and supervised learning; Section 2.2 introduces concept drift; Section 2.3 introduces the types of drift; and in Section 2.4 introduces drift severity.

2.1 GAUSSIAN MIXTURE MODEL

Since our approach is based on GMM, we present in this section some basic concepts of this approach to support the following sections. For this, we organize this Section in the following Subsections: 2.1.1: GMM unsupervised learning; and 2.1.2: GMM supervised learning.

2.1.1 Unsupervised Learning

The GMM can be applied as a clustering method, in which the Gaussians are created considering only the input data without their respective labels. Each generated distribution is seen as a cluster, also called a mixture component (OLIVEIRA; BATISTA, 2015). The idea of combining several mixture components assumes that not all real-world data can be modeled by a single Gaussian distribution, but if multiple components are combined, a new distribution can be modeled as follows:

$$P(x) = \sum_{i=1}^{K} P(x|C_i) \cdot w_i \tag{2.1}$$

where K is the number of Gaussians and x is a multivariate observation with d dimensions, formally represented by: $x^d = \{x_1, x_2, .., x_d\}$. Each constant w_i is a weight representing the a priori probability $P(C_i)$ that means the number of observations that constitute the mixture component i, $0 \leq w_i \leq 1$ and $\sum_{i=1}^{K} w_i = 1$. $P(x|C_i)$, represents the conditional probability of observation x to belong to the mixture component C_i. This probability is computed using the mean (μ_i) and the covariance (Σ_i) of the cluster C_i as follows:

$$P(x|C_i) = \frac{1}{(2\pi^{d/2}\sqrt{|\Sigma_i|})} exp(-\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i)) \tag{2.2}$$

The approach commonly used to make the adjustment of the GMM to the training data is called Expectation-Maximization (EM) (EM). This approach initialises each component C_i on a random subset from the training set, and then iteratively adjusts the means and covariance of the mixture components in order to maximize the probability of the data in the distribution created. This process was proposed by Moon (1996) and is accomplished by the following steps:

1. Start the parameters mean (μ_i), covariance (Σ_i) and weight (w_i) of each component C_i on a set of random data from the training set.

2. **Estep:** estimates the posterior probability $P(C_i|x_j)$ of all training observations $x_j = j = \{1, 2, .., m\}$ in relation to each mixture component C_i using the Equation 2.3.

$$P(C_i|x_j) = \frac{P(x_j|C_i) \cdot w_i}{\sum_{i=1}^{K} P(x_j|C_i) \cdot w_i} \tag{2.3}$$

3. **Mstep:** Maximize the PDF estimating the new parameters: weight (w_i) mean (μ_i) and covariance (Σ_i) using the Equations: 2.4, 2.5 and 2.6.

$$w_i = \frac{1}{m} \cdot \sum_{j=1}^{m} P(C_i|x_j) \tag{2.4}$$

$$\mu_i = \frac{\sum_{j=1}^{m} P(C_i|x_j) \cdot x_j}{w_i} \tag{2.5}$$

$$\Sigma_i = \frac{\sum_{j=1}^{m} P(C_i|x_j) \cdot (x_j - \mu_i)^T (x_j - \mu_i)}{w_i} \tag{2.6}$$

4. Repeat steps 2 and 3 until the stop criterion is satisfied. This criterion can be the number of training iterations.

As a disadvantage, this approach is sensitive to noise, which can cause the mixture components to fit too tightly to the data. An example of this situation is when noise appears too far away from the other observations; in this case, the EM algorithm will create a Gaussian only to learn the region of that observation.

2.1.2 Supervised Learning

The GMM can also be applied as a supervised learning approach (OLIVEIRA; BATISTA, 2015). in this case, there are two ways to train the GMM model. The first one is using only one Gaussian per class, where the mean (μ_i), covariance (Σ_i) and weight (w_i) are computed for each class (OLIVEIRA; BATISTA, 2015). The disadvantage of this approach is the assumption of normality, which in turn it cannot represent complex distributions, resulting in poor performance.

The second one is using one GMM per class (OLIVEIRA; BATISTA, 2015), where the unsupervised learning discussed in Section 2.1.1 is adopted to model each class separately. The disadvantage of this approach is the complexity to define the optimum number of Gaussians to have a good modeling for each class.

In both cases, the Gaussians represent the *learned* decision boundaries and the process to classify is based on the higher posterior probability given by the Equation 2.7, where K represents the number of Gaussians existing in the system.

$$\hat{y} = argmax_{i \in \{1,2,\cdots,K\}} P(C_i|x) \tag{2.7}$$

Therefore, the observation will be classified with the class of the Gaussian that obtained the highest probability for it, that is, if the chosen Gaussian learns class 1, then the given classification will also be 1 and so on.

2.2 CONCEPT DRIFTS

Traditional machine learning approaches are often applied to learn patterns over a concept. A concept is defined by the joint distribution $P(\mathbf{x}, y)$, determined by the prior class probabilities $P(y)$ and class conditional probabilities $P(\mathbf{x}|y)$ (WEBB et al., 2016). Thus, the trained model is used to estimate the class of new input data.

However, these traditional machine learning approaches are not effective when applied to data that arrives continuously and sequentially as data streams (DITZLER et al., 2015; GAMA et al., 2014; KRAWCZYK et al., 2017; GOMES et al., 2017a). The assumption of stationarity cannot be sustained due to the dynamism of the data streams (MINKU; WHITE; YAO, 2009). In data streams, drifts may occur between different instants of time t and u, changing the concept, as $P_t(\mathbf{x}|y) \neq P_u(\mathbf{x}|y)$ (WEBB et al., 2016). Drifts can happen for many different and possible reasons, an example would be the spam detection task (CAVALCANTE; OLIVEIRA, 2015). At some point, some emails may be considered important to the user, but at another time they may be considered as spam due to changes in user preference (ALMEIDA et al., 2018).

Drifts that occur over time are known in the literature as concept drift. Concept drift presents several challenges for machine learning approaches, as the concept learned by the model becomes inefficient to predict future behaviors (MINKU; YAO, 2012). Because of this, data formats such as data streams require algorithms capable of learning in *on-line* way, in what for each new observation the classifier is updated. Incorporating new data helps the classifier to keep up with the nature of the drifts and continuing give useful predictions for the users.

The literature divides the term concept drift into virtual and real. To understand this distinction, take Figure 1 as an example. The virtual concept drift, illustrated in Figure 2(b), happens when the distribution of the examples $P(x)$ changes, but the observation labels remain the same at different times (JR et al., 2014). This phenomenon is also known as sampling shift or feature change (GAMA et al., 2014).

The real concept drift, illustrated in Figure 2(c), is characterized by changes in $P(y|x)$, i.e, when a set of examples has class labels defined at a given time (Figure 2(a)) and at another time these labels change, also known as concept shift or conditional change (GAMA et al., 2014).

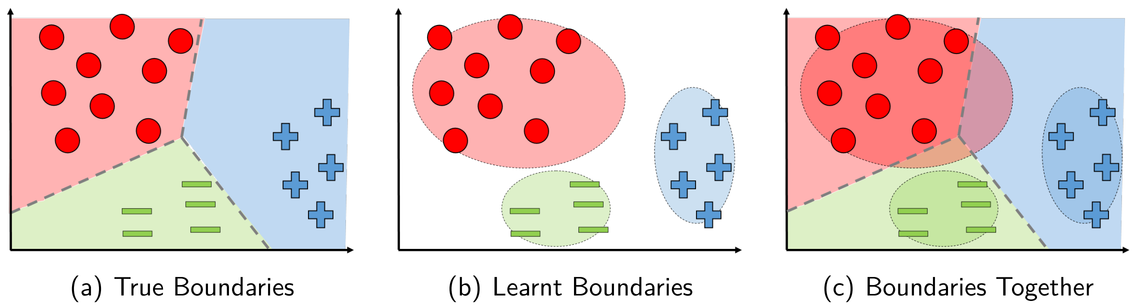
These two types of drifts may happen together, but it is necessary to understand that they have distinct characteristics and can affect the performance of machine learning algorithms in specific ways. So, in order to offer a background to the aforementioned in research question 1, we discuss these impacts in Subsections: 2.2.1: Virtual concept drifts and 2.2.2: Real concept drifts.

2.2.1 Virtual Drift

Virtual concept drifts refer to changes in the distribution of input data $P(x)$ (ALMEIDA et al., 2018). These drifts may be due to the changing the distribution or to incomplete or partial representation of the current data distribution (BUDIMAN; FANANY; BASARUDDIN, 2016). Partial representation means that a small portion of the data has arrived and is not sufficient to represent the complete data distribution. In this situation, if a model is trained on a partial representation of the true distribution, it means that new data may emerge in a region not trained by the model (BUDIMAN; FANANY; BASARUDDIN, 2016). Although *true* decision boundaries of the problem do not change, this kind of drift can result in *learned* decision boundaries being inadequate (KRAWCZYK et al., 2017).

To understand this problem, take as an example the representation of Figure 2. Figure 3(a) represents the *true* decision boundaries of the problem, represented by gray and rectilinear lines. Figure 3(b) represents the *learnt* decision boundaries by an algorithm at a given instant of time t. Finally, Figure 3(c) represents an overlap of these two decision boundaries together. Initially, in this example, it is observed that the *learnt* boundaries do not perfectly reflect the best separation between classes, because the model did not have access to all data, only to a portion of them.

Figura 2 – Types of decision boundaries. The points presented are training observations. The rectilinear gray lines represent the *true* decision boundaries of the problem and the Gaussians represents the *learnt* decision boundaries by an algorithm.

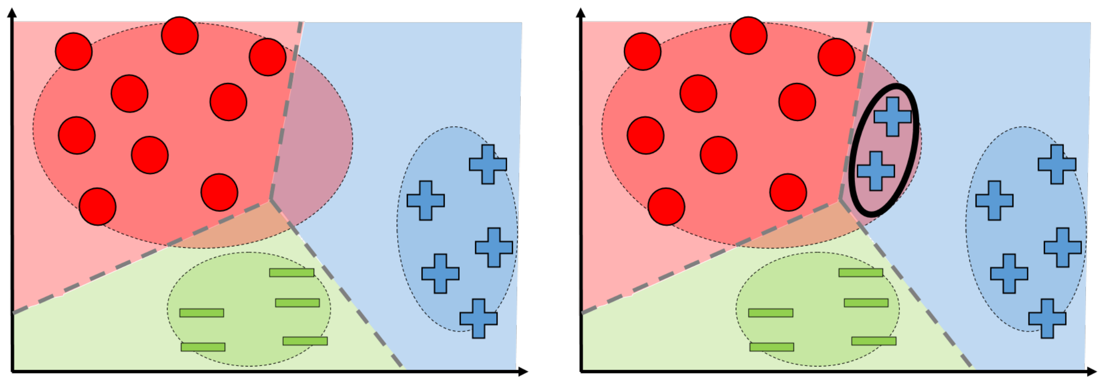


(a) True Boundaries (b) Learnt Boundaries (c) Boundaries Together

Source: Author.

In this kind of problem mentioned, some sort of errors can be generated due to the partial representation of the data. To understand this, take as an example Figure 3. Figure 4(a) represents the modeling at time t, for that moment the *learnt* boundaries are sufficient and do not generate classification errors, but at time $t + \Delta$ (Figure 3) it is observed that data from the blue class, highlighted with a black circle, arrived in an unknown region were allocated within a red Gaussian. This situation simulates one of the representations of a misclassification caused by a virtual concept drift.

Figura 3 – Example of misclassification caused by virtual concept drift. The points presented are training observations. Observations inside the black circle are going to be misclassified. The rectilinear gray lines represent the *true* decision boundaries of the problem and the Gaussians represents the *learnt* decision boundaries by an algorithm.



(a) Features distribution at time t (b) Features distribution at time t + Δ

Source: Author.

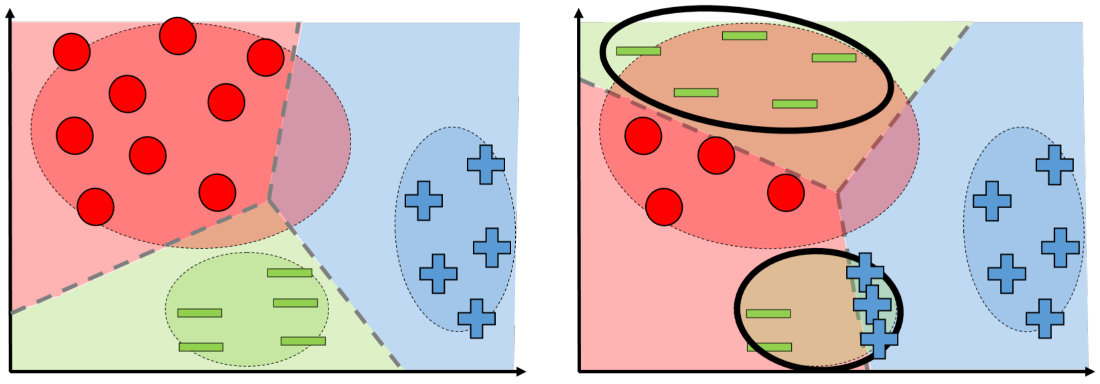
As the posterior probability $P(y|x)$ does not change, reconstructing the whole model is not ideal, because the decision boundaries are still the same (BUDIMAN; FANANY; BASARUDDIN, 2016). Techniques for dealing with real concept drifts can identify certain types of virtual drifts. The problem is that a misinterpretation may give a wrong decision about classifier retraining (KRAWCZYK et al., 2017). Retraining is often associated with gathering new observations, and this delay forces the system to use an outdated model which degrades the performance until there is enough data for retraining.

2.2.2 Real Drift

The real concept drift happens when the posterior probability $P(y|x)$ change over time. In this kind of concept drift, the relation between the target classes (γ) and the feature vectors (χ) change, i.e, in the spam e-mail filtering problem, where a "useful"e-mail at time t become to the class "spam"at a given time $t + \Delta$ (CAVALCANTE; OLIVEIRA, 2015; ALMEIDA et al., 2018).

To better illustrate this problem, take the Figure 4 as example. At time t (Figure 5(a)) the *learnt* decision boundaries are enough to classify all observations. However, at time $t + \Delta$ (Figure 4) the posterior probability $P(y|x)$ change, forcing an update in the classifier, since the *learnt* boundaries by it at time t become obsolete, which may cause strong accuracy drop.

Figura 4 – Example of misclassification caused by real concept drift. The points presented are training observations. Observations inside the black circle are going to be misclassified. The rectilinear gray lines represent the *true* decision boundaries of the problem and the Gaussians represents the *learnt* decision boundaries by an algorithm.



(a) Features distribution at time t (b) Features distribution at time t + Δ

Source: Author.

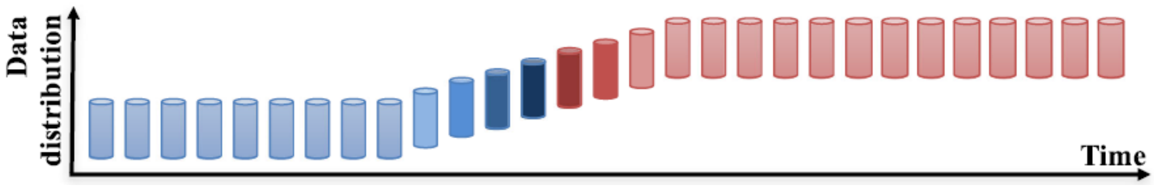
2.3 TYPES OF DRIFTS

Apart from differences in the cause and effects of the types of concept drifts, researchers distinguish between several ways of how such concept drift may occur. Webb et al. (2016) and (KHAMASSI et al., 2018) provide a quite comprehensive taxonomy of concept drift types. Drifts can fall into four categories: 2.3.1: Incremental drifts; 2.3.2 Gradual drifts; 2.3.3: Abrupt drifts; and 8: Recurrent drifts. The next Subsections are going to discuss the challenges involved in these types of drift levels.

2.3.1 Incremental Drift

According to Webb et al. (2016), incremental drifts consist of a steady progression from source distribution of a concept $P(x,y)_t$ toward concept $P(x,y)_{t+\Delta}$ such that at each time step the distance from the concept $P(x,y)_t$ increases and the distance to concept $P(x,y)_{t+\Delta}$ decreases. Other authors consider incremental drifts as a sequence (i.e, not severe) continuous shifts, following a certain trend (SOBOLEWSKI; WOŹNIAK, 2013), in which the changes are slow and take more time to be complete (KRAWCZYK et al., 2017). As presented in Fig. 5, this type of shifts are not so radical and are connected with old concepts (MINKU; WHITE; YAO, 2009). This type of drift take a long period of time to be noticed, even if each small change occurs suddenly (MINKU; WHITE; YAO, 2009).

Figura 5 – Example of incremental drift.



Source: Lu et al. (2018).

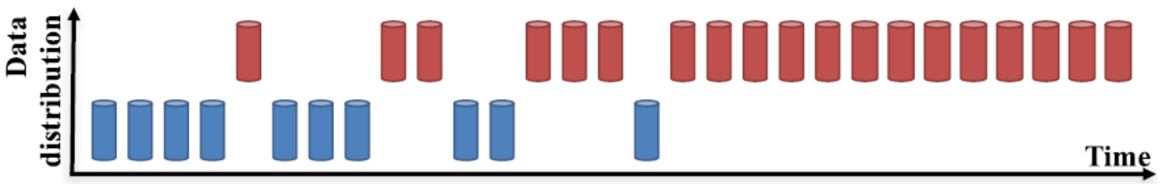
An example of incremental drift is where credit card fraud patterns change over time. Consider the introduction of a new technology in chips of credit cards. The new technology changes the types of fraud that can be committed. As more credit card customers get new cards with the new chips, the new types of fraud become more common until everyone has new cards, where the concept drift in the transaction stream would stop.

2.3.2 Gradual Drift

According to Liu et al. (2018), Gradual drifts refers to the transition phase where the probability of observations from the first distribution $P(x,y)_t$ decreases while the probability of getting observations from the next distribution $P(x,y)_{t+\Delta}$ increases (MINKU; WHITE; YAO, 2009). Another vision of this concept is that gradual drifts may recur alternately in the gradual stage for a certain period (KUNCHEVA, 2008). This kind of drift is harder to detect since it creates a period of uncertainty between stable states (KHAMASSI et al., 2018).

The main difference of this drift in relation to incremental (Subsection: 2.3.1) is that incremental is related to a constant and direct progression of the joint probability distribution to a new concept, whereas the gradual is related to a probabilistic progression in which at each moment of time the new concept has a specific probability of appearing. An example of gradual drift is presented in Figure 6 in which at each new instant of time there is a greater probability of the second concept appearing.

Figura 6 – Example of gradual drift.



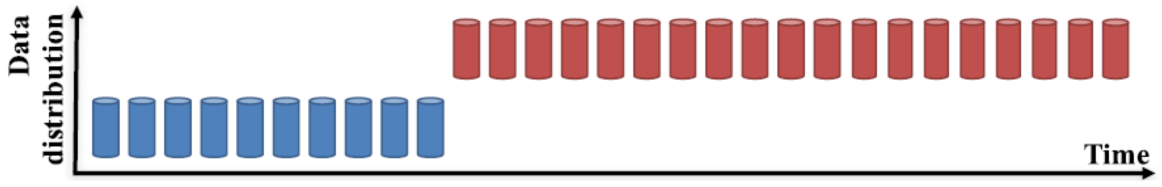
Source: Lu et al. (2018).

This drift offer to the classifier the challenge to choose the adequate time to learn the new concept. If the system is very sensitive to drifts it will be reset in a few steps, and it will be forced to relearn the previous concept again. Approaches that have the retraining based on a drift detector needs to have the care of not having a high number of false alarms because otherwise the system will be reset several times over time.

2.3.3 Abrupt Drift

Abrupt drift is also known by abrupt changes, sudden drifts or concept substitutions (KUNCHEVA, 2008). It occurs when at a moment in time t, the source distribution $P(x,y)_t$ is suddenly replaced by a different distribution in $P(x,y)_{t+1}$ (KRAWCZYK et al., 2017), as presented in Fig 7. This type of data may be easier to be detected than incremental drifts because of the quick degradation that it causes in the classifier. This kind of drift immediately deteriorates the learner performance, as the new concept quickly substitutes the old one.

Figura 7 – Example of abrupt drift.



Source: Lu et al. (2018).

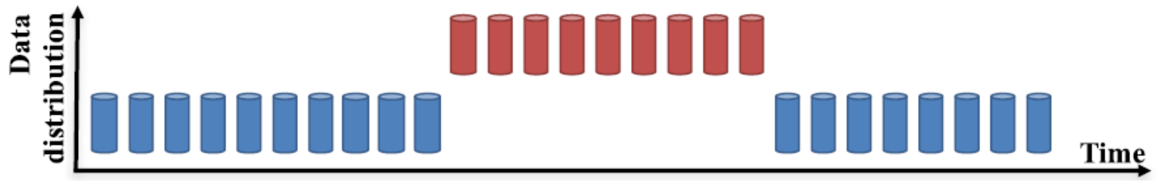
A real world example of abrupt drift could be a market crash. In a stock market stream, almost instantly, stock values will change and follow a pattern different to the previous one (WEBB et al., 2016).

2.3.4 Recurrent Drift

According to Webb et al. (2016), recurrent drift is the transition for a pre-existing concept. Recurrent drifts generally are associated with cycles, that is concepts which reoccur in a specific order (KRAWCZYK et al., 2017). It is a not-so-uncommon phenomenon observed in so-called cyclical environments (such as climate or any seasonal data) (ALIPPI; BORACCHI; ROVERI, 2013).

The main difference of this drift to the gradual (Subsection: 2.3.2) is that gradual drifts may occur forward to either a previous concept or to a new one, but recurrent is strictly oriented to a pre-existing concept. Figure 8 represents this type of drift.

Figura 8 – Example of recurrent drift.



Source: Lu et al. (2018).

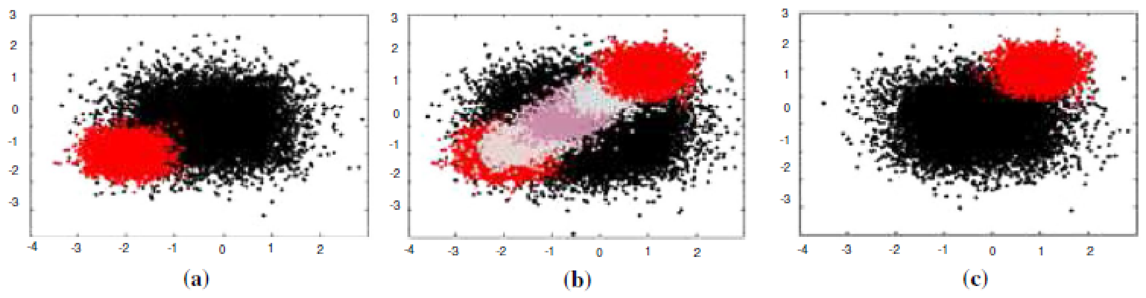
The great challenge behind this type of drift is if the discarded information later becomes relevant again, the system needs to relearn the previous concepts again (KRAWCZYK et al., 2017). In such cases, algorithms capable to recall previous knowledge will adapt to the new concept quickly as they have saved old information that will be useful at the time of change.

2.4 SEVERITY OF DRIFTS

The severity criterion proposed by (MINKU; WHITE; YAO, 2009) refers the percentage of the input space that had its class changed upon completion of the drift. Khamassi et al. (2018), organized the explanation of this classification drifts through two types: global and local, also known as severe and non-severe (MINKU; WHITE; YAO, 2009), respectively.

The severe drift affects much of the feature space, usually involving an evolution from one class to another region of space Khamassi et al. (2018). Fig. 9 demonstrates a severe shift from one class moving to another region of space. As this change significantly alters data distribution, it can be easily detected.

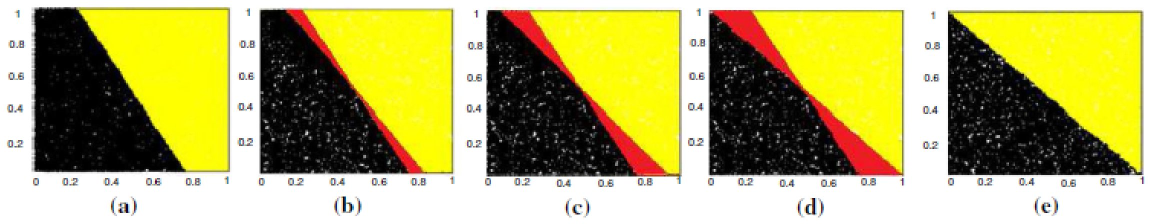
Figura 9 – Example of severe drift. Fig a) shows two classes, one in black and one in red. Fig b) shows the red class evolving to another region of the feature space, where the gray part shows the path taken. Fig c) shows the complete evolution of the red class.



Source: Khamassi et al. (2018).

The non-severe drift occurs near the already existing regions of the feature space (KHAMASSI et al., 2018). Fig. 10 demonstrates this kind of drift. Looking at space, we can see that only some parts of the data are affected by the drifts. It is essential to highlight that detecting a non-severe drift generally takes long, as it takes more time to identify the changes that have occurred. In some cases, the non-severe drift can be seen as noise, making the model to reset its learning if it is sensitive to noise. To overcome this problem, the model needs to (i) differentiate local drifts from noise and (ii) deal with the lack of instances in the region to update the classifier.

Figura 10 – Example of non-severe drift. Fig a) presents two classes, one represented in black and the other in yellow. Fig b), c) and d), show in red the region of the feature space that had its class changed. Fig e) shows the complete drift.



Source: Khamassi et al. (2018).

2.5 CHAPTER SUMMARY

In this Chapter, we have introduced basic concepts for understanding GMM supervised and unsupervised learning, concept drifts, the types of drift, and the levels of severity of concept drift.

Regarding Section 2.1: Gaussian Mixture Model (GMM), it was discussed that in unsupervised learning, the EM algorithm is often used to train the model, but it is highly sensitive to noisy observations. It was also argued that in supervised learning, training is associated with two main methodologies: (i) the first using one Gaussian per class, but as a weakness one has to assume that the data are normal; and (ii) the second is to create one GMM per class, but it becomes difficult to choose the best number of Gaussians to use for each class.

Section 2.2: Concept Drifts, has shown that: virtual drifts happen when the data distribution or prior probability $p(x)$ change but $p(y|x)$ do not; and real concept drifts occur when the *true* decision boundaries of the problem changes, i.e, when a posterior probability $p(y|x)$ changes. We also saw that both drifts cause misclassifications and need different strategies to address.

Regarding Section 2.3: Types of drifts, it was seen that: (i) incremental shifts have a steady progression of one concept to another and take more time to be completed; (ii) gradual drifts happen when the probability of an old concept slowly decreases over time while the probability of a new one increases; (iii) abrupt drift happen when the old concept is suddenly replaced by a new one; and (iv) recurring drifts happen in periodic times, i.e, cycles.

Finally, Section 2.4: Severity of drifts, has shown that: (i) severe drift involves a great change in the feature space, usually, an evolution of one class to another region of space; and (ii) non-severe drift occurs near the already existing regions of the feature space.

3 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 (DITZLER et al., 2015; GAMA et al., 2014). For ensemble approaches for non-stationary environments, we refer readers to (KRAWCZYK et al., 2017; GOMES et al., 2017a). For class imbalance learning in non-stationary environments, we refer readers to (WANG; MINKU; YAO, 2018). Knowing this, in this Chapter, we will focus only on literature studies that relate to this work.

Therefore, in order to fill this, we organize this Chapter in the following Subsections: 3.1: Gaussian mixture model in stationary data streams; 3.2: Gaussian mixture model in non-stationary data streams, addressing works that deal with only virtual and virtual and real drifts; 3.3: Approaches that deal with virtual and real drifts; and 3.4 Approaches that use drift understanding.

3.1 GAUSSIAN MIXTURE MODEL IN STATIONARY DATA STREAMS

Few studies in the literature have been found to deal with stationary data streams using the GMM method, but within this context, we can cite the work of Lee, Lewicki e Sejnowski (2000).

In Lee, Lewicki e Sejnowski (2000), an algorithm called ICA Gaussian Mixture Model (ICA GMM), was proposed. The ICA GMM was applied in an unsupervised classification task for the speaker identification problem. In the problem, two speakers keep a conversation, where when one speaks the other keeps silence. The difference between each speaker's lines is measured by the distance from each speaker to the recorder. The ICA GMM is performed by the following steps: (i) the algorithm estimates separately the density of each class (i.e, each speaker) using unsupervised learning. (ii) After the estimation, the model classifies the new sounds in an *on-line* way. The classification is made using the a posterior probability (Eq. 2.3), which verifies which is the most likely class for the arrival sound. After the end of the algorithm execution, the model receives the true classes for each arrival observation to verify its accuracy. Some experiments were performed to find out how many observations were needed to identify the true speaker. It was found that single observations were not discriminating. The best result was obtained using 2000 thousand observations, which means that the ICA GMM needs a lot of sounds to have a better classification. Although this work deals with an *on-line* problem, the investigated data stream does not have concept drift, characterizing this problem as a stationary data stream. When considering a non-stationary data stream this approach would not be efficient because it has no adaptation mechanism to handle concept drift.

3.2 GAUSSIAN MIXTURE MODEL IN NON-STATIONARY DATA STREAMS

The GMM method was also widely used in non-stationary data streams. Based on this, in this Section, we are going to discuss how these works were applied to solve problems in the following research areas: 3.2.1 Approaches that deal strictly with virtual drifts; and 3.2.2: Approaches that try to deal with both virtual and real concept drifts.

3.2.1 GMM Approaches That Deal With Virtual Concept Drifts

Regarding GMM used to deal with virtual concept drifts we can cite the works of Choi, Park e Lee (2004), Song e Wang (2005), Yu e Qin (2008), and Chen e Zhang (2010).

The work of Choi, Park e Lee (2004) aimed to detect abnormal failures in industrial processes in *on-line* way. The proposed approach combines the Principal Component Analysis (PCA) (PCA) and GMM. PCA is used to narrow the subspace of the problem and GMM is used to estimate the density of the new distribution. The training process of this approach consists in modeling normal failures of the process through unsupervised learning using the EM algorithm. The number of Gaussians used for each problem is known, so they are set manually. In *on-line* monitoring mode, abnormal failures are detected by GMM using the conditional probability (Eq. 2.2). A failure is identified when this probability get closer to zero for all created clusterings, indicating that it is not among training failures. As the distribution of failures evolves over time, this problem is characterized as the detection of a virtual concept drift. The study showed that the proposed approach detects failures more effectively than traditional PCA combined with the statistical tests as T^2 and Q. Two hypotheses test were used to verify whether the multivariate mean of a sample has changed or not, the T^2 is parametric and Q is non-parametric.

Song e Wang (2005) proposed an *on-line* clustering algorithm based on GMM. The proposed approach trains the GMM using the EM algorithm in historical data, and selects the number of Gaussians using the Bayesian Information Criterion (BIC) (BIC). After that, the approach deals with the data stream in an *on-line* way. In this processing, instead of storing historical data in memory, the approach employs a strategy to store only the parameters of the Gaussians, so as to keep only the cluster information. This strategy is divided into two steps, namely: (i) the merge and (ii) update steps. In the (i) merge step, a posterior probability (Eq. 2.3) is used to determine the Gaussian most likely to accommodate the arrival observation. After that, the Gaussian that was most similar to the observation incorporates it into his clustering. This incorporation is done using equations proposed in (SONG; WANG, 2005) to update the parameters mean, covariance, and the prior probability of the chosen Gaussian. As clusters iteratively incorporate the observation, there is an organizing mechanism to ensure that they do not increase their radius too much. This mechanism checks whether existing components have a statistical similarity. This similarity is measured by its mean and covariance parameters. The covariances are compared using the W test and the means using the T^2 test. If there are statistically equal Gaussians, then they are merged. Thus in the (ii) update step, for each merge pair from the previous step, a new Gaussian is created by combining the parameters of these two. This strategy aims to maintain the maximum number of clusterings of the problem. This type of approach adapts intuitively to virtual concept drifts, as data evolves the clustering remodels. Experiments were performed on noisy real signal datasets extracted from neuronal recordings and the results demonstrated that the proposed method is applicable to real-world *on-line* clustering problems.

Yu e Qin (2008) proposes the Finite Gaussian Mixture Model (FGMM) (FGMM), an approach to monitoring chemical processes. The FGMM combines a GMM approach with a Bayesian inference strategy using mahalanobis distance. The motivation for this approach assumes that methods such as PCA and Partial Least Square (PSL) (PLS) are inadequate for monitoring multivariate processes because they assume that the data have a normal distribution. Given this, the proposed approach procedure is divided into three steps: (i) FGMM is trained in an unsupervised way using data from the normal behavior of chemical processes. In the absence of a prior knowledge of the number of Gaussians needed, the F-J algorithm (Figueiredo-Jain) is adopted to estimate both the number of Gaussians and their parameters; (ii) After training, the model is applied to monitor the data in an *on-line* way. This monitoring uses a metric created by the author to calculate if the arrival example belongs to the Gaussians created in the training phase. This metric is called a BIP and is given by:

$$BIP = \sum_{i=1}^{K} P(C_i|x_t) \cdot P(x_t|C_i) \tag{3.1}$$

This metric combines a posteriori prob. $P(C_i|x_t)$ and conditional probability $P(x_t|C_i)$. The assumption discussed by the author is that this metric helps to avoid false detections caused by misclassification. (iii) In the *on-line* monitoring process, the arrival observation is said to be abnormal if it has near zero probability in the BIP metric. This type of *on-line* monitoring detects virtual concept drifts as data distribution evolves to a type not known in training phase. As the approach aims to identify changes in data distribution, it automatically identifies virtual concept drifts. The experiments were performed on three multimodal data sets including: an artificial multivariate system, a simulated Continuous Stirred Tank Heater (CSTH) (CSTH) and the Tenesse Eastman Problem (TEP) (TEP). Comparative results showed that FGMM detected failures with better efficiency than the approaches: PCA with T^2 and Squared Prediction Error (SPE) (SPE).

Finally, in Chen e Zhang (2010) applied the GMM method combined with the PCA method. In this case, PCA is used to decrease the dimensionality of the data and GMM is used to model them. This combination is applied to identify failures in semiconductor batch manufacturing process. Similar to the above mentioned approaches, an unsupervised learning process is done using EM algorithm to learn the normal functioning of the processes. The BIC model selection criterion is used to define the number of Gaussians. *On-line* monitoring checks through the a posteriori probability (Eq. 2.3) whether the incoming batches are failures or not. Failures are defined by probabilities close to zero for all Gaussians created. Therefore, this work can also be seen as an approach to the identification of virtual concept drifts, as the distribution of failures evolve over time. In the experiments performed, the proposed approach was compared with the T^2 and SPE statistical tests and their combination. The models were evaluated considering two metrics: the number of false alarms and the number of times that they missed identifying failures. It was found that the proposed approach identified all failures and obtained the lowest false alarm rate of the investigated methods.

Through discussions of the aforementioned works, we conclude that the GMM method can successfully monitor complex distributions and identify in an *on-line* way when they change or when an observation of a new distribution arrives. Discussions demonstrate how the use of statistical probabilities can be effective in making inferences about the degree of pertinence of an observation to particular clusterings. Yet, as disadvantages, all methods mentioned are strictly limited to use in unsupervised tasks.

3.2.2 GMM Approaches That Deal With Virtual and Real Concept Drifts

To deal with virtual and real concept drifts, we can cite the works of Ditzler e Polikar (2011) and Oliveira e Batista (2015), who developed approaches involving semi-supervised and supervised learning respectively.

Ditzler e Polikar (2011) proposed the Weight Estimation Algorithm (WEA) (WEA), an approach that combines GMM with an ensemble of classifiers. WEA is an incremental batch-based learning algorithm. Its *on-line* procedure starts with two main activities: (i) the supervised training of the ensemble of classifiers using the labeled data; and (ii) the unsupervised training of GMM using the unlabeled data. Thus, for each new unlabeled arrival observation, its distance is calculated for each cluster in the GMM. The purpose of this is to give to the new unlabeled arrival observation the class of the nearest cluster to it. The observations labeled with this process are called synthetic observations and they are used to verify if a concept drift occurred. If a new observation is too far from the trained observations it means that a change in the distribution happened. These synthetic observations also are used to estimate the accuracy of the ensemble in order to know how well the system is classifying new observations. Given this accuracy, the voting weights of the ensemble are updated. In theory, the WEA deals with virtual concept drifts training a new GMM for every new batch of unlabelled observations and assigns the labels to them, but this mechanism works well only in drifts that don't move so far from the initial set. For dealing with real concept drifts, the WEA trains a new classifier for each batch of information and updates the voting weights of the ensemble based on the accuracy of the synthetic observations. The problem with this approach is that, according to the author, it is strictly limited to incremental drifts, as the approach has very slow recycling. WEA was compared with the *Learn*$^{++}$.NSE (POLIKAR et al., 2001), an incremental learning algorithm for non-stationary environments. Results indicated that WEA performed similarly to *Learn*$^{++}$.NSE where the unlabeled data (test) has no similarity to labeled data (training). However, WEA showed significant improvement when there was a similarity between the labeled (training) and unlabeled (test) batch distributions. Another critical weakness of this approach is the dependence of the GMM for updating the weighting votes of the ensemble, which makes the WEA dependent on the similarity of the test data with the training data. It means that in the presence of abrupt real drifts the WEA will suffer a strong drop in performance because of the delay to update to the new concept.

Finally, Oliveira e Batista (2015) proposed an algorithm called Incremental Gaussian Mixture Model For Concept Drifts (IGMM-CD) (IGMM-CD). IGMM-CD uses an *on-line* learning which incorpores each new arriving observation using the GMM. Iteratively, the approach classifies new instances that arrive from the data stream. If the system classifies the instance correctly, then the nearest Gaussian of the observation is updated. If the system misclassified the observation and does not have a Gaussian with the minimum distance ($Cver$) from the observation, thus a new Gaussian with size ($sigma\_ini$) is created. The authors claim that the error presents the necessity to learn an emerging area, so claim that the creation of a new Gaussian can also be used to rapidly adapt the system to real concept drifts. In the same way, the update of the nearest Gaussian of the observation is used to adapt the system to virtual concept drifts. Besides that, the system has a mechanism to exclude obsolete Gaussians based on the parameter T, that counts the number of Gaussians per class. If that number is exceeded, the Gaussian with lower density is excluded. A key weakness of this approach is that, on the presence of abrupt drifts, the system can delay excluding obsolete Gaussians, which in turn may degrade the system performance. Moreover, virtual drifts resulting in misclassifications are treated in the same way as real drifts, potentially causing the unnecessary addition of new Gaussians, and leading to waste of knowledge when this results in other existing Gaussians to be excluded.

The above mentioned works tend to deal intuitively with virtual and real concept drifts. However, they can only perform well if the drifts are incremental because they can maintain the knowledge learned. On the other hand, if the changes are abrupt the slow forgetting of knowledge forces the system to degrade strongly. The reason for this is that both works do not have a mechanism to make possible to quickly recycle knowledge, which indicates that they deal with virtual and real drifts in the same way. If both had separately a mechanism to deal with the different types of drifts, they would readjust their knowledge quickly.

3.3 APPROACHES THAT DEAL WITH VIRTUAL AND REAL DRIFTS

In this section we are going to discuss some works in the literature that deal with virtual and real drifts and that are not based on GMMs. Among these works, the following stand out: Alippi, Boracchi e Roveri (2013), Budiman, Fanany e Basaruddin (2016), Ren et al. (2018) and Almeida et al. (2018).

Alippi, Boracchi e Roveri (2013) introduced the Just in Time Classifier for Recurrent Concepts (JIT) (JIT), an active learning approach capable of dealing with recurring concept drifts. The active approach means reacting only when a concept drift is detected, in which the classifier's knowledge is discarded and a new learning process is taken to maintain the accuracy of the approach high. JIT is a framework which can use any classifier. In the mentioned paper, the authors use two Change Detection Test (CDT) (CDT) based on the rule of Intersection of Confidence Intervals (ICI) (ICI) to identify concept drifts, the first to identify drifts in a prior probability $p(x)$ and the second to identify drifts in a posterior probability $p(y|x)$. The CDT for a prior probability $p(x)$ monitors the distribution of input data disregarding any label existence, while CDT for a posterior probability $p(y|x)$ checks whether the average classification error is constant or not. The adaptation strategy is activated when either detector reports a type of drift. This strategy consists of resetting the classifier, which in turn can be a problem since in virtual concept drift some of the classifier's knowledge is still important and the system needs to waste time to recover the knowledge enough for a retraining. As soon as a change is detected a new concept representation is created and compared to the stored representations, if there is a current equivalent representation, its supervised samples are used to retrain the classifier. JIT has been experimented with on a several datasets involving different types of concept drifts for both multivariate and scalar data. Key findings indicated that JIT can successfully exploit supervised information from the past and has been shown to be effective in dealing with abrupt shifts. This approach for monitoring a prior $p(x)$ and posterior probabilities $p(y|x)$ can identify virtual and real concept drifts, respectively.

Budiman, Fanany e Basaruddin (2016) proposed An Adaptive Online Sequential Extreme Learning Machine For Concept Drift Tackling (AOS-ELM) (AOS-ELM) that is capable of handling virtual and real concept drifts. The approach uses a unique classifier with a *on-line* learning, i.e, it updates with each new arrival observation. To deal with different types of concept drifts, two approaches are employed: (i) random assignment of input layer weights and (ii) creating new neurons in the output layer and joining them in the neural network. The author argues that the first approach can handle virtual drifts because according to the principle of ELM network learning theory, the input weights are independent of training observations and their updates may remove the bias of the previous training observations, thus preparing the classifier for the creation of new output weights and neurons. The second approach can handle real concept drifts as follows: new output neurons and their respective matrix weights are generated and with zero values. This new matrix is concatenated with the past weight matrix and its training is done analytically according to the common ELM network procedure. According to the authors, the assumption of this is that output layer training may get closer to any complex decision boundary, thus solving the problems that real concept drifts cause. Although this paper argues that it can handle both drifts, the strategy employed in the two cases is to reset the weights of the ELM network, which in turn is not always an appropriate strategy, as past knowledge is always lost regardless of the change. The experiments were performed on public data sets: SEA, STAGGER, MNIST, USPS and IDPS. For each data set studied, the type of drift was presented to show that AOS-ELM can reach good results. The experiments showed that the AOS-ELM obtained better classification performance than the traditional OS-ELM.

Ren et al. (2018) proposed the Knowledge-Maximized Ensemble (KME) (KME), an ensemble of classifiers with active learning for data streams. Active learning is the process to reset the classifier whenever a CDT informs a concept drift. In this algorithm, supervised and unsupervised indicators are used to detect concept drifts. Unsupervised indicator is obtained through two characteristics: the sample mean of the examples and the power-law transformation of sample variance. The supervised indicator is obtained through a threshold for the classification error. As both indicators are monitored in parallel, virtual and real concept drifts are not confused, as each type of indicator becomes responsible for one type of drift, a supervised for real concept drifts and an unsupervised for virtual concept drifts. This information is measured for a fixed size lot and is monitored for a confidence interval. In this way the KME can simultaneously monitor the a priori $p(x)$ and a posteriori $p(y|x)$ distributions. A concept drift is detected when any CDT reports that there has been a drift in its indicators. Upon detection, batches of observations are saved in memory for reuse to retrain a new ensemble. The classification for new observations is given by the average of the classifiers contained in the ensemble. Even though the KME has differents CDTs for each type of drift, this ability is little explored because in the end, for all drifts a new ensemble is trained from scratch, which makes this a limitation of this work. KME has been compared to 8 state-of-the-art classifiers across various synthetic and real-world data sets. The comparison showed that KME was very effective and reached good results in different types of concept drifts.

Finally, in the study by Almeida et al. (2018), it was proposed the Dynamic Selection Based Drift Handler (Dynse) (Dynse). Dynse is an approach based on Dynamic Classifier Selection (DCS) (DCS). DCS is the process to select a specific classifier/ensemble for each test instance according to its size neighborhood (k) in a validation set. The validation set (M) used on this method is represented by a sliding window which traverses the incoming data excluding the oldest observations. The authors claim that virtual concept drifts can be dealt with using a sliding window with a very large size, because it will contain many observations corresponding to the current concept. On the other hand, real drifts can be dealt with using a sliding window with a small size, since its observations can be excluded faster by speeding up the switch to new data. New classifiers are added into a pool (D) at each batch of m observations, once the batch has been filled. The size of the batch (m) and the sliding window (M) are user-defined. The disadvantage of this is that the approach can only deal well with one of the types of drift (virtual or real). The type of drift with which it deals well depends on the pre-defined value chosen for m and M. Moreover, the strategy used to learn a new concept (creating a new classifier from scratch and adding it to the pool) is not ideal for virtual concept drifts where part of the past knowledge remains valid and could be used to help better learning the new concept. Besides that, in the presence of abrupt drifts, the system will degrade its performance because it delays to collect new observations to train a new classifier.

Overall the approaches discussed have two main problems: (i) in some cases the approaches use specific indicators to detect the different types of drifts, but use the same strategy to deal with both; and (ii) other approaches for dealing with each type of drift needs to pre-define their parameters accordingly the type of drift that will be dealt with. The weakness of the first point is which using the same strategy, that is learn from scratch, the system cannot harness the useful past knowledge to expedite the learning of the new concept. The weakness of the second point is the dependence of prior knowledge to adjust the parameter to handle the drift. If another type of drift occurs, the unsuitable parameter value will lead to a degradation in performance. So, even though these approaches can have their parameters adjusted beforehand to deal with either drift in P($\mathbf{x}$) or drifts in P(y|$\mathbf{x}$), they cannot deal with data streams that contain both types of drift at the same time. In both cases, the approaches discussed will drop performance for not having specific strategies for the virtual and real concept drifts.

3.4 APPROACHES THAT USE DRIFT UNDERSTANDING

Once concept drift occurs, identifying its type (virtual or real) and understanding where and how it affects the classifier is important to support an efficient drift adaptation strategy. This process is called drift understanding (LIU et al., 2017).

Recently, Lu et al. (LU et al., 2018) have published a survey of approaches related to concept drift. Among the 130 publications reviewed, only 3 (three) approaches apply drift understanding (GAMA; CASTILLO, 2006; LU; ZHANG; LU, 2014; LIU et al., 2017). Among these three, only Gama et al. (GAMA; CASTILLO, 2006), and Liu et al. (LIU et al., 2017) developed a system to deal with drifts based on drift understanding, whereas Lu et al. (LIU et al., 2017) just proposed a drift detector capable of telling the type of drift. Therefore, we will discuss the former two approaches further in this section.

Gama et al. (GAMA; CASTILLO, 2006), applied an incremental decision tree to detect local drifts through each leaf node in the tree. Each leaf node represents a hyper-rectangle in the data feature space where a Bayesian Classifier is trained. Error-based drift detectors monitor each classifier, and classifiers with low performance are replaced by others trained from scratch. If regions not learned by the tree appear (e.g., emerging / novel areas), this method creates new branches to represent these regions. Despite identifying some local drifts, this method depends heavily on regions learned by the decision tree.

Liu et al.'s Density Synchronized Drift Adaptation algorithm (LDD-DSDA) (LIU et al., 2017), uses a local drift degree (LDD) measurement to identify what regions of the feature space were affected. The approach is based on two data windows with the same size (*m*). The first window is used to train the classifier, and the second window slides over each new observation. At each *m* time steps, both windows are compared, and observations affected by the drift are identified based on LDD and removed. Only observations from a new concept and those not affected by the drift are kept and used to retrain the classifier. The limitation of LDD-DSDA is the dependence on the similarities between two concepts. If an abrupt severe drift occurs, and the window size (m) has a high value, there is a long delay for the classifier's retraining, which impairs its performance. Small values for m store a few observations, which can train a classifier with little generalization.

3.5 DISCUSSION AND CHAPTER SUMMARY

In this Chapter, we present a brief literature review on how the GMM was applied in the data stream literature, both in stationary and non-stationary. Besides that, we present a review of the main works in the literature that sought to deal with virtual and real concept drifts. To summarize the main differences between the aforementioned works and the approaches proposed in this thesis, we present their main mechanisms in Tbl. 1.

Regarding Section: 3.1: GMM in stationary data streams, we identified that the highlighted work learns the classes using unsupervised learning. From this, the method uses this learning to perform classifications in an *on-line* way. This approach is restricted to stationary data only. In non-stationary data would not be efficient, since it has no adaptation mechanism to deal with the concept drifts.

Regarding Section: 3.2: GMM in non-stationary data streams, we found that works that deal with virtual drifts (Subsection 3.2.1) tend to estimate the distribution of data and *on-line* check whether the new observations belong to the estimated distribution. If the observations do not belong, a warning signal is given to the user. Works that deal with virtual and real drifts (Subsection 3.2.2) tend to adjust their knowledge interactively as they receive new information, so they can cope with virtual and real drifts. The problem with these approaches is that in the presence of abrupt real drifts the system takes a long time to forget its old knowledge, thus having a great degradation. Another point that favors this degradation is because these works do not use any drift detector to know when the system performance is decreasing so that to be reset.

Regarding Section: 3.3: Approaches that deal with virtual and real drifts, it was observed two main points: (i) they need prior knowledge of the problem to pre-define their parameters so that they can perform well with the type of drift; (ii) they use the same strategy for dealing with the different types of concept drifts, that is, reset the classifier, i.e, there is no strategy expert on the type of drift. Another point little discussed by them is the influences that different types of concept drifts have on classifier performance.

Regarding Section: 3.4: Approaches that use drift understanding, the approaches always have a drift detector that informs the user when the drift takes place and informs in which region of space the drift took place. This information is used to activate the most suitable strategy for dealing with the drift. Nevertheless, the studied approaches are dependent on the base classifier and only deal well with non-severe drifts.

Thus, this review concludes that (i) the GMM has not yet had its potential fully exploited in the data stream literature to deal with virtual and real concept drifts, leaving a large gap of possibilities for investigations and proposals to solve this problem; and (ii) although there are works that propose to deal with both drifts, they are limited to use the same strategy for both cases, and this leaves a large gap in how to deal with them in a more appropriate way, requiring different solutions so as not to lose knowledge or overload the approach with indiscriminate updates.

Tabela 1 – Comparison of the main mechanisms of the related works in relation to the proposed approaches of this Thesis. Each column represents a determined attribute for each work. The last and boldly row represents the summary of the mechanisms of our proposals.

Work	Task	Learning	Classifier	Key Weakness	Virtual Drifts	Real Drifts	Drift Understanding	Main Goal
(LEE; LEWICKI; SEJNOWSKI, 2000)	Unsupervised	On-line	GMM	Does not handle concept drift	-	-	-	Unsupervised Classification
(CHOI; PARK; LEE, 2004)	Unsupervised	Batch	GMM	Strictly limited to unsupervised tasks	Yes	No	No	Fault Detection
(SONG; WANG, 2005)	Unsupervised	On-line	GMM	Strictly limited to unsupervised tasks	Yes	No	No	On-line Clustering
(YU; QIN, 2008)	Unsupervised	Batch	GMM	Strictly limited to unsupervised tasks	Yes	No	No	Fault Detection
(CHEN; ZHANG, 2010)	Unsupervised	Batch	GMM	Strictly limited to unsupervised tasks	Yes	No	No	Fault Detection
(DITZLER; POLIKAR, 2011)	Supervised/ Unsupervised	Batch	Ensemble	Directed to incremental shifts; Slow forgetting of useless knowledge;	Yes	Yes	No	On-line Classification
(OLIVEIRA; BATISTA, 2015)	Supervised	On-line	GMM	Dependence on an adjustment of a parameter; Slow forgetting of useless knowledge.	Yes	Yes	No	On-line Classification
(ALIPPI; BORACCHI; ROVERI, 2013)	Supervised	Batch	Framework	Resets the classifier's knowledge in virtual drifts; Does not harness useful knowledge for adaptation;	Yes	Yes	No	On-line Classification
(BUDIMAN; FANANY; BASARUDDIN, 2016)	Supervised	On-line	ELM	Resets classifier knowledge for each observation;	Yes	Yes	No	On-line Classification
(REN et al., 2018)	Supervised	Batch	Ensemble	Resets the classifier's knowledge in virtual drifts; Does not harness useful knowledge for adaptation;	Yes	Yes	No	On-line Classification
(ALMEIDA et al., 2018)	Supervised	Batch	DCS/DES	Dependence on an adjustment of a parameter; Train new classifiers for handle with both drifts;	Yes	Yes	No	On-line Classification
(GAMA; CASTILLO, 2006)	Supervised	On-line	Decision Tree	Depends heavily on regions learned by the decision tree.	Yes	Yes	Yes	On-line Classification
(LIU et al., 2017)	Supervised	Batch	Framework	Sensible abrupt severe drift occurs	Yes	Yes	Yes	On-line Classification
This thesis	**Supervised/ Unsupervised**	**Batch/ On-line**	**GMM**	**Harnesses useful knowledge to accelerate adaptation; Updates knowledge with each observation; Uses a CDT to monitor the error; Erase useless knowledge quickly;**	**Yes**	**Yes**	**Yes**	**On-line Classification**

4 IMPACT OF VIRTUAL AND REAL DRIFTS ON CLASSIFIER SUITABILITY

This Chapter presents an analysis of the impacts of virtual and real drifts on Bayesian classifiers' suitability, answering RQ1. This analysis provides the foundation for proposing novel approaches able to more efficiently and effectively deal with both types of drift simultaneously. To this end, we present in Section 4.1 the datasets used in the experiments described in this Chapter as well as in the other Chapters of this thesis. Finally, in Section 4.3 we analyzed the impacts of virtual and real drifts on classifiers suitability, answering RQ1.

4.1 DATASETS

We used synthetic and real-world datasets, whose characteristics are presented in Tbl. 3. These datasets have only continuous attributes, because data clustering methods like the EM algorithm used in GMM are not applicable for datasets with categorical input attributes (GRIM, 2006). These datasets are available on Github[1].

4.1.1 Synthetic datasets

All synthetic datasets were generated using the Tornado framework proposed by (PESARANGHADER; VIKTOR; PAQUET, 2018) with exception of the virtual datasets. The virtual datasets were generated for this Thesis, to understand how virtual drifts can affect the approaches. Tbl. 2 presents the code example to generate the datasets and Tbl. 3 presents all datasets descriptions.

Tabela 2 – Code to generate the synthetic datasets with Tornado framework (Python) available on-line in https://github.com/alipsgh/tornado.

Datasets	Code to Generate
Circles	CIRCLES(concept_length=2000, noise_rate=0.1)
Sine1	SINE1(concept_length=2000, noise_rate=0.1)
Sine2	SINE2(concept_length=2000, noise_rate=0.1)
SEA	SEA(concept_length=2000, thresholds=[1, 9, 2, 6], noise_rate=0.1)
SEARec	SEA(concept_length=2000, thresholds=[1, 9, 2, 6, 1, 9, 2, 6], noise_rate=0.1)

The datasets were used to enable a detailed investigation of how each compared approach

[1] https://github.com/GustavoHFMO/OGMMF-VRD/tree/master/data_streams

behaves in the presence of known drifts. Incremental drifts consist of a steady progression from an old concept to a new one, that is, they can be seen as a sequence of abrupt drifts of low severity. Gradual drifts refer to the transition phase where the probability of observations from the old concept decreases while the probability of the new one increases (MINKU; WHITE; YAO, 2009). The severity calculation was proposed by (MINKU; WHITE; YAO, 2009), and represents the percentage of the input space which has its target class changed after the drift is complete. It was calculated by selecting 2000 instances randomly of each concept of the data stream and checking the percentage of such instances whose labels change from one concept to another. For virtual drifts, the severity is calculated when a region of the space that previously had no one instances ($P(\mathbf{x}) = 0$) receives observations of a class. Thus it is considered that this region had its target class changed, where the percentage of the input space is the same as the percentage of instances of the class that appeared. Each synthetic dataset has 10% of label noise, where the labels are changed randomly. Overlapping between classes is calculated for each data stream concept. In each concept, the data is separated by class. In each class, the K-Disagreeing Neighbors (KDN) (kDN) is calculated for each observation; in the end, we calculate the average of the kDN for the class. For the calculation, we define the number of neighbors used in the kDN equal to $k = 5$. So, for the overlapping, values close to 0 indicate low overlap, and values close to 1 indicate high. Finally, balancing represents to proportion of observations per class.

4.1.2 Real datasets

For real datasets, severity cannot be obtained, as no equation generates the data. For the overlapping calculation, as we do not know where the drifts occur, we define the calculation interval at every 5000 observations. Besides, only one modification was made in the ELEC dataset, where missing values were removed, resulting in the number of examples in Tbl. 3. These datasets are also available on-line[2], and they were discussed in (SOUZA et al., 2020).

[2] https://en.wikipedia.org/wiki/Concept_drift#Real

Tabela 3 – Dataset descriptions.

Type	Datasets	Attributes	#Drifts	Concept Size	#Examples	Balancing	Overlapping	Severity	Drift Type	
Synthetic	Virtual 5	2	5	2000	10000	34.3%, 35.5%, 32.2%	0.27, 0.12, 0.29	23.47, 34.1, 24.9, 34.95	Virtual	Abrupt
	Virtual 9	2	9	1000	10000	32.6%, 35.2%, 32.2%	0.24, 0.11, 0.23	28.23, 33.9, 28.32, 33.8, 27.23, 32.6, 23.39, 31.6	Virtual	Abrupt
	Circles	2	3	2000	8000	49.8%, 50.1%	0.28, 0.26	44.15, 37.55, 32.6	Virtual/Real	Incremental
	Sine1	2	4	2000	10000	49.8%, 50.2%	0.26, 0.26	89.7, 88.85, 89.45, 87.75	Real	Abrupt/Recurrent
	Sine2	2	4	2000	10000	49.5%, 50.6%	0.27, 0.26	89.8, 88.85, 89.2, 87.75	Real	Abrupt/Recurrent
	SEA	3	4	2000	8000	50.1%, 49%	0.24, 0.26	50.4, 24.15, 47.1	Real	Gradual
	SEAREC	3	8	2000	16000	49.9%, 50%	0.24, 0.25	49.4, 24.1, 45.55, 17.95, 47.8, 26.65, 46.2	Real	Gradual/Recurrent
Real	NOAA	9	2	-	18159	31.0%, 68%	0.54, 0.23	-	-	-
	ELEC	4	2	-	27549	41.5%, 58.4%	0.38, 0.26	-	-	-
	PAKDD	29	2	-	50000	80.2%, 19.7%	0.25, 0.97	-	-	-
	GasSensor	9	6	-	13910	13.1%, 18.4%, 21.1%, 11.7%, 13.9%, 21.6%	-	-	Real	-
	INS-Inc-Reo	128	6	-	79986	16.6%, 16.6%, 16.6%, 16.6%, 16.6%	-	-	Real	Incremental/Recurrent
	INS-Inc-Abt	33	6	-	79896	16.6%, 16.6%, 16.6%, 16.6%, 16.6%	-	-	Real	Incremental/Abrupt
	INS-Grad	33	6	-	24150	16.6%, 16.6%, 16.6%, 16.6%, 16.6%	-	-	Real	Gradual

4.2 EXPERIMENTAL SETUP

This Section describes our experiment designed to investigate our RQ1 that aims to know the difference between the impact of virtual and real drifts on the suitability of Bayesian classifiers' learned decision boundaries and also in their predictive performance. To this end, we selected two representative methods from the literature, IGMM-CD (OLIVEIRA; BATISTA, 2015), and Dynse (ALMEIDA et al., 2018) (see Chapter 3).

IGMM-CD and Dynse were applied on two synthetic datasets, Virtual 9 and Circles. The Virtual 9 dataset has virtual drifts that cause classes to evolve to different regions of the feature space. The Circles dataset presents real drifts with gradual transitions.

This experiment seeks to visually understand how the decision boundaries learned by the classifiers work when the concept drift takes place. For this, we collected prints of the model executions in two different moments, one before the concept drift and the other after the concept drift. The decision boundaries of the problem were generated by training a Gaussian Naive Bayes on all observations of the concept. The concept represents all observations of the same distribution before the drift occurs. Thus, we can visualize the intersection of learned decision boundaries versus problem decision boundaries.

4.3 RESULTS AND DISCUSSIONS

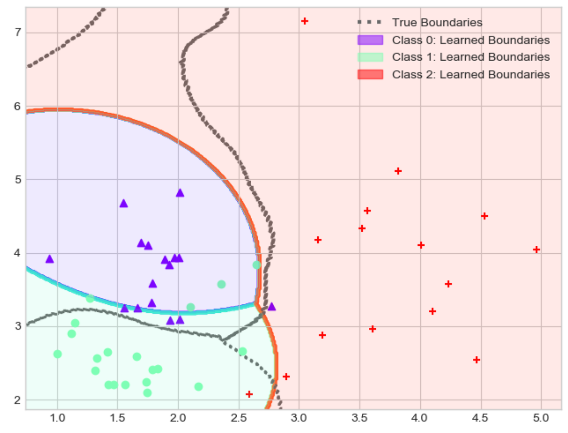
Figs. 11, and 12 illustrates representative plots of execution of Dynse and IGMM-CD, respectively, in the Virtual 9 and Circles datasets before and after a concept drift.

Virtual drifts had a different effect on the GMM-based (IGMM-CD) and Bayesian-based (Dynse) classifiers. In particular, we can see from Fig. 12(b) that the decision boundary of Class 2 was incorrectly learned by Dynse, even though a considerable portion of the previously acquired knowledge[3] 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

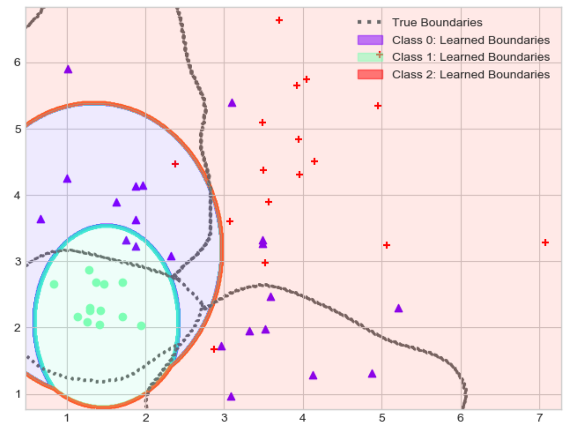
[3] We use the term "knowledge" here to refer to the portions of the space considered to belong to each class by the learned classifier.

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.

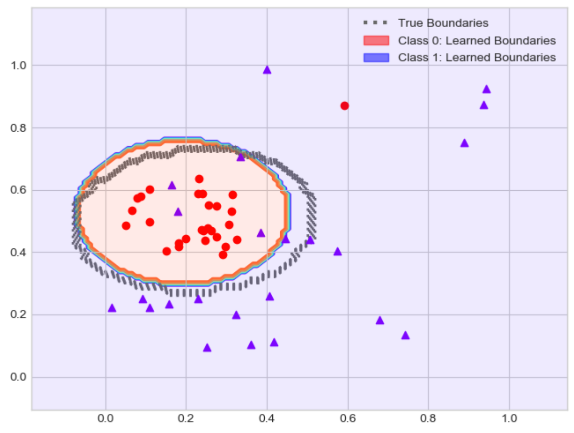
Figura 11 – Execution of Dynse (ALMEIDA et al., 2018) on Virtual 9 and Circles datasets (Tbl. 3). The points represent the first 50 before and after the concept drift. Dynse uses as base classifier the Gaussian Naive Bayes. Gray and dotted lines represent the *true* decision boundaries of the problem. The solid colored lines represent the decision boundaries *learned* by each approach. The text G0, ..Gn represents the Gaussian number.



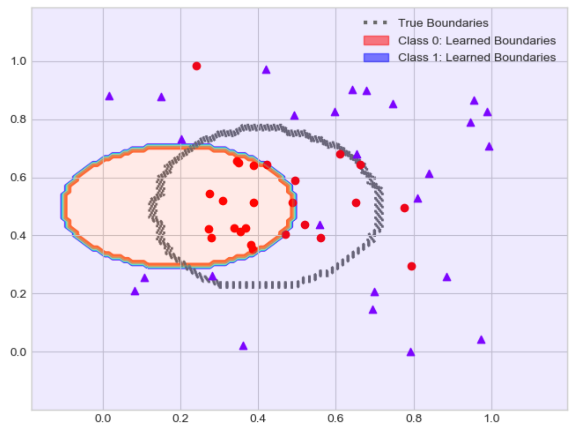
(a) Virtual 9 (Before)



(b) Virtual 9 (After)



(c) Circles (Before)

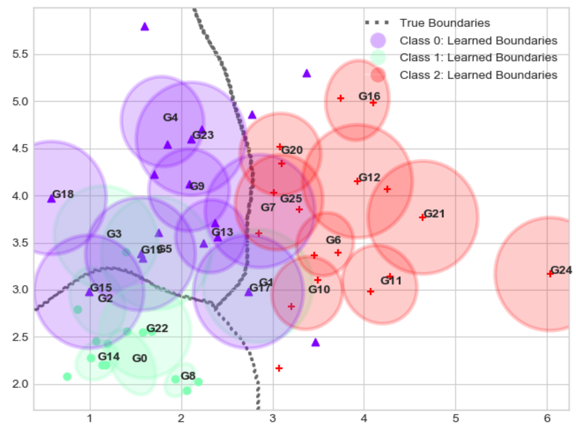


(d) Circles (After)

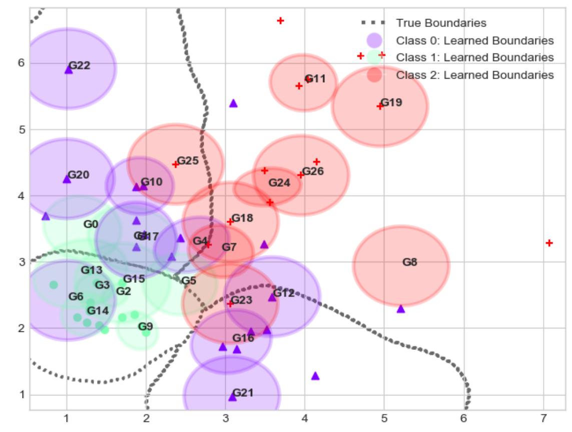
Source: Author.

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. 12(d) 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. 13(d)) 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.

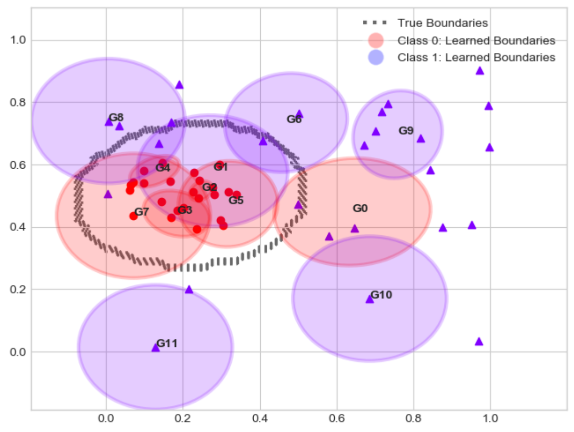
Figura 12 – Execution of IGMM-CD (OLIVEIRA; BATISTA, 2015) on Virtual 9 and Circles datasets (Tbl. 3). The points represent the first 50 before and after the concept drift. IGMM-CD uses as base classifier the GMM. Gray and dotted lines represent the *true* decision boundaries of the problem. The solid colored lines represent the decision boundaries *learned* by each approach. The text G0, ..Gn represents the Gaussian number.



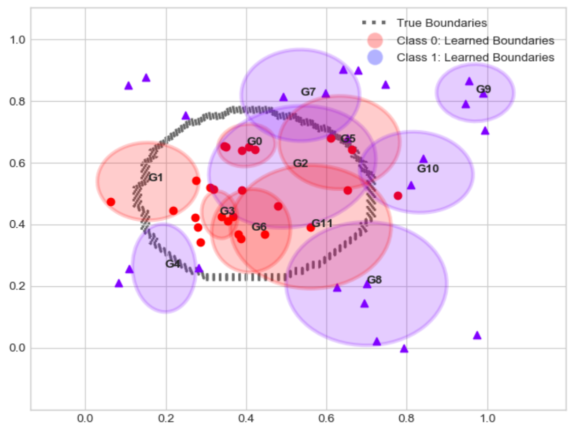
(a) Virtual 9 (Before)



(b) Virtual 9 (After)



(c) Circles (Before)



(d) Circles (After)

Source: Author.

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. Insufficient, in the sense that the classifier's decision boundaries cannot adequately represent the observations of the classes. 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.

4.4 DISCUSSION AND CHAPTER SUMMARY

This chapter presented an analysis of the impacts of virtual and real drifts on Bayesian classifiers' suitability, answering RQ1.

We observed that although virtual drifts do not change the decision boundaries of the problem, they impair the generalization ability of the predictive model. This is because the problem data evolves, causing the learned decision boundaries to no longer adequately represent the data, thus diminishing its ability to separate problem classes.

We also observed that although the real drifts strongly harm the predictive models, resetting all knowledge learned by the model is a drastic strategy. There are real non-severe drifts that change only a tiny part of the feature space. In this situation, updating the close boundaries of the drift would help the model adapt more quickly.

Therefore, we realized that understanding the type of drift is essential for building an efficient system. We understand that the system needs to have some on-line updates to keep up with non-severe drifts. Furthermore, it needs to have some mechanism to incorporate new areas of the feature space in case untrained data distribution arises. Finally, the system needs a mechanism to quickly erase all its knowledge to deal with severe drifts.

5 PROPOSED METHODS

This Chapter describes three proposed approaches for dealing with virtual and real drifts. In Section 5.1, we implement the RQ2 through the method GMM-VRD, our first attempt to deal simultaneously with virtual and real drift. In Section 5.2, we implement RQ3 and RQ4 through the method OGMMF-VRD, a noise-robust approach that can recall past models to handle both virtual and real drifts at the same time. Finally, in Section 5.3, we implement RQ5 and RQ6 through the method GLDD-DU, an approach that applies drift understanding to identify when and how the drift happened and, based on this, activates the most suitable strategy for dealing with feature space changes. The proposed methods will be validated through the experiments discussed in Chapter 6.

5.1 GMM-VRD: A GAUSSIAN MIXTURE APPROACH TO DEAL WITH VIRTUAL AND REAL CONCEPT DRIFTS

According to what we discussed in Section 3.3, literature approaches that deal with real and virtual drifts use the same strategy to deal with both or need to pre-define their parameters to deal appropriately with the type of drift. Given that, we propose our first approach, GMM-VRD, that aims to implement our RQ2) that asks how to efficiently deal with virtual drifts at the same time as preserving the ability to deal with real drifts when using GMM. This proposal was published in (OLIVEIRA; MINKU; OLIVEIRA, 2019), and its overall procedure is illustrated in Alg. 1.

GMM-VRD is a batch training method that updates itself on-line for each recent observation of the data stream. It also analyzes through probability calculations whether the new observations belong to the previously trained data distribution. If not, it creates new Gaussians to represent these regions. In addition, it uses a drift detector to monitor the GMM error and thus reset the entire system when degradation is reported. In general, GMM-VRD procedure is divided into four parts: (i) the procedure to train a new GMM model discussed in Subsection 5.1.1; (ii) the procedure to classify a new observation from the data stream, discussed on Subsection 5.1.2; (iii) the procedure to deal with virtual concept drifts, discussed in Subsection 5.1.3; and (iv) the procedure to deal with real concept drifts, discussed in Subsection 5.1.3.4.

Algorithm 1 GMM-VRD's overall procedure.

```
    {Part I: Train initial model}
 1: W ← $(x_i, y_i)_{i=1}^{m}$
 2: model ← trainGMM(W)                                  ▷ See Alg. 2
 3: detector ← detector(model, W)
    {Part II: On-line classification}
 4: for each instance $x_t$ do
 5:     $\hat{y}_t$ ← predict(model, $x_t$)
 6:     $y_t$ ← trueLabel($x_t$)
    {Part III: Update on virtual drifts}
 7:     if $y_t \neq \hat{y}_t$ then
 8:         model ← VirtualAdaptation($x_i, y_i$)        ▷ See Alg. 3
 9:     end if
    {Part IV: Update on real drifts}
10:     if detector triggers the drift signal then
11:         W ← storeData($x_t : x_{t+m}$)
12:         model ← trainGMM(W)                          ▷ See Alg. 2
13:         detector ← detector(model, W)
14:     end if
    {Part V: GMM maintenance}
15:     model ← removeGaussians(model)
16: end for
```

5.1.1 Model Creation

The model creation step has two substages: (i) the initialization of the classifier, discussed in Subsection 5.1.1.1; and (ii) the initialization of the drift detector, discussed in Subsection 5.1.1.2.

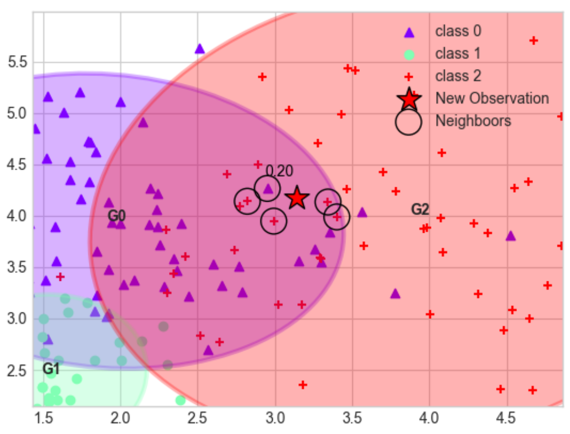
5.1.1.1 Classifier Initialization

We elaborate a training architecture for the GMM in order to overcome the problems with noise and to choose the optimal numbers of Gaussians discussed in Section 2.1. The architecture is illustrated in Alg. 2 and is composed of two stages: (i) the pre-processing stage and (ii) the model selection stage.

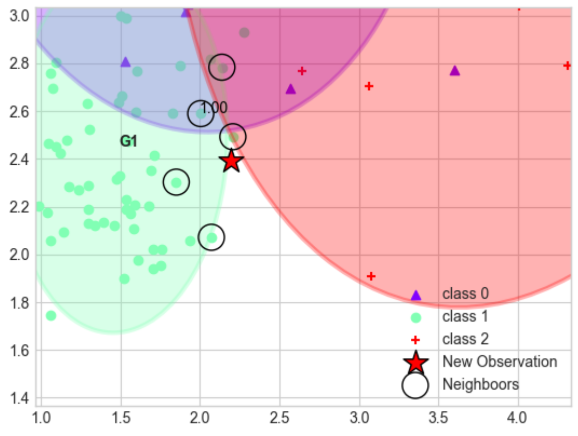
In the pre-processing stage, we use the approach KDN (kDN) discussed in (WALMSLEY et al., 2018) to define the hardness of each instance in the training dataset, as defined in Equation 5.1. The kDN represents the fraction of the k nearest neighbors of the query observation that do not share the same class. Thus, by definition, the kDN assumes values in the range of [0, 1], where values close to 0 indicate that the observation is easy to classify and 1 is difficult, as shown in Fig 13. For our approach, we have specified that observations with kDN greater than 0.8 tend to be noise and should thus be removed from the original training set because it means that 80% of neighboring observations have a different class. After, the observations are separated by class to train a GMM per class.

$$kDN(x) = \frac{|x'|x' \in kNN(x) \wedge label(x') \neq label(x)}{k} \tag{5.1}$$

Figura 13 – Illustration of a normal observation and a noisy observation according to KDN. The star-shaped point represents the incoming observation and the circles around the observations represent its neighbors according to KNN. The number above the star-shaped point represents the value obtained by KDN.



(a) A noise-free observation



(b) Noisy observation

Source: Author.

The model selection stage trains GMMs with numbers of Gaussians ranging from 1 to $Kmax$ using the EM (MOON, 1996) approach discussed in details in Subsection 2.1.1. The best resulting model is chosen using the higher value of the Akaike Information Criterion (AIC) (AIC) and is added to the final GMM model. The AIC criterion is defined as follows:

$$AIC = 2 \cdot p - 2 \cdot L \tag{5.2}$$

where p represents the number of model parameters. In a GMM with only one Gaussian, the parameters are the mean (μ_i), covariance (Σ_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, defined by the Equation 5.3:

$$L = \sum_{i=1}^{m} log \sum_{j=1}^{K} P(x_i|C_j) \cdot w_j \tag{5.3}$$

where m represents the number of observations used for training, K the number of Gaussians in the GMM model, and $P(x_i|C_j)$ the conditional probability defined by the Equation 2.2.

Algorithm 2 trainGMM()

```
    input: dataset (W),
    number max of gaussians ($K_{max}$),
    number of iterations for EM (i),
    {Part I: Pre-processing}
 1: H ← kDN(W)
 2: $W_{new} \leftarrow W[H < 0.8]$
    {Part II: Training process}
 3: gmmFinal ← ∅
 4: for each class in $W_{new}$ do
 5:     data ← $W_{new}[class]$
 6:     gmms ← $[1...K_{max}]$
 7:     for k ← 1 until $K_{max}$ do
 8:         gmms[k] ← trainEM(data, i, k)
 9:     end for
10:     bestGmm ← bestAIC(gmms)
11:     gmmFinal ← add(bestGmm)
12: end for
13: return: gmmFinal
```

Concluding, this training process has as time complexity $O(m + (skt))$, where m represents the initial number of observations used for training; s represents the not noisy observations; k represents the number of Gaussians tested $(Kmax)$; and t represents the number of iterations used in the EM algorithm.

5.1.1.2 Drift Detector Initialization

We use the Exponentially Weighted Moving Average Charts (ECDD) (ECDD) (ROSS et al., 2012) as the drift detection method in our approach. This drift detector was chosen due to its capability to monitor the error and trigger alarms at both warning (w) and drift (c) levels. Therefore, it enables different strategies to be used to cope with these two situations. In addition, ECDD obtained good results in existing works, e.g., (OLIVEIRA et al., 2017). Other drift detection methods could be investigated in future work.

Algorithm 3 VirtualAdaptation()

```
    Input: observation ($x_t$, $y_t$)
 1: gaussian, pertinence ← NearestGaussian($x_t$, $y_t$)          ▷ See Alg. 4
 2: if pertinence $< \theta$ then
 3:     UpdateGaussian(gaussian, $x_t$)
 4: else
 5:     CreateGaussian($x_t$, $y_t$)
 6: end if
```

5.1.2 On-line Classification

In the second part, as more incoming observations arrive, the trained model is used to predict their respective labels. The prediction is done using Equation 2.7. The predicted label can be used by users for decision-making.

5.1.3 Adaptation to Virtual Drifts

This component of the GMM-VRD approach performs the maintenance of useful knowledge. This mechanism is triggered whenever a misclassification has occurred. A misclassification can be a signal that the evolution of the data may be degrading the system's performance.

To activate the more appropriate strategy, for each incoming observation, we need to know where it is located in relation to existing Gaussians. For that, we use the routine NearestGaussian() described in Algorithm 4 and for activate the correct strategy we use the routine VirtualAdaptation() in Algorithm 3.

Algorithm 4 NearestGaussian()

```
    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($P(x_t|gaussian)$)                          ▷ Eq. 2.2
 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 given by Equation 2.2 for all existing

Gaussians (k) in the current GMM, thus having as time complexity $O(k)$. This process corresponds to lines 1 to 9 of the Algorithm 4. If there is a Gaussian near with the same class of the arrived observation (line 3 of the Algorithm 3) then the model is updated according to Subsection 5.1.3.1. If there is not, then another Gaussian is created (line 5 of the Algorithm 3)) according to Subsection 5.1.3.2.

5.1.3.1 Update a Gaussian

When the nearest Gaussian has $pertinence > \theta$ for the observation consulted, then it will be updated. The process of updating a Gaussian is done using modified equations of the EM approach proposed by Engel e Heinen (2010). These modifications are necessary because in the presence of data streams it is costly to store data groups to update these parameters. The difference between these equations and the old ones is that the Gaussian can be updated based on a single new observation x_t and the Gaussian's current parameters (mean μ_i^t, covariance Σ_i^t and weight w_i^t). Another parameter necessary is the variable sp_i^t that will store the accumulated posterior probability of each Gaussian. The illustration for this process is presented in Figure 14 and the Equations used to update the Gaussian are 5.4, 5.5, 5.6 and 5.7.

$$sp_i^t = sp_i^{t-1} + P(C_i|\boldsymbol{x}) \tag{5.4}$$

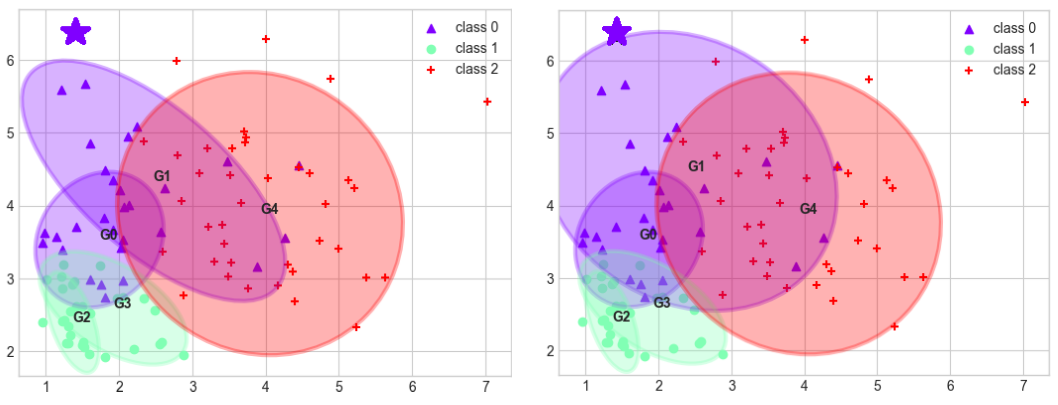
$$w_i^t = \frac{sp_i^t}{\sum_j^K sp_j^t} \tag{5.5}$$

$$\boldsymbol{\mu_i^t} = \boldsymbol{\mu_i^{t-1}} + \frac{P(C_i|\boldsymbol{x})}{sp_i^t} \cdot (\boldsymbol{x} - \boldsymbol{\mu_i^{t-1}}) \tag{5.6}$$

´1

$$\boldsymbol{\Sigma_i^t} = \boldsymbol{\Sigma_i^{t-1}} - (\boldsymbol{\mu_i^t} - \boldsymbol{\mu_i^{t-1}})^T(\boldsymbol{\mu_i^t} - \boldsymbol{\mu_i^{t-1}}) + \frac{P(C_i|\boldsymbol{x})}{sp_i^t} \cdot [\boldsymbol{\Sigma_i^{t-1}} - (\boldsymbol{x} - \boldsymbol{\mu_i^t})^T(\boldsymbol{x} - \boldsymbol{\mu_i^t})] \tag{5.7}$$

Figura 14 – Illustration of the process to update a Gaussian. On Figures 15(a), 15(b), the star-shaped observation represents the new test instance. The text G0, ..Gn represent the number of the Gaussian.



(a) Update (Before) (b) Update (After)

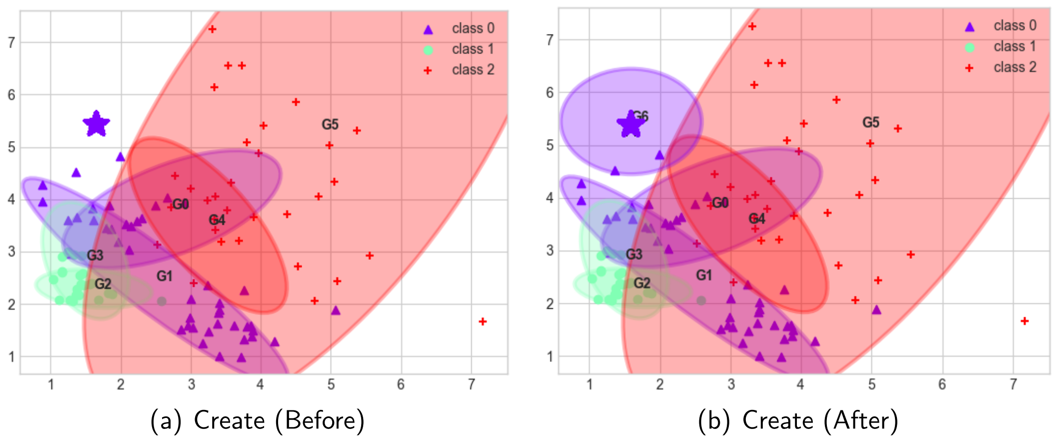
Source: Author.

5.1.3.2 *Create a Gaussian*

When no Gaussian has pertinence for the observation consulted, then it is necessary to create another Gaussian to represent the new region in the feature space. The new Gaussian is initialised using sp_i and $w_i = 1$, $\mu_i = x_i$ and $\Sigma_i = Cver \cdot I$, where I represents the identity matrix, which has the same number of dimensions as x_t, and $Cver$ represents the size of the Gaussian circumference. $Cver$ was configured equal to 0.5 for all experiments.

After the Initialisation of the new Gaussian's parameters, it is necessary to update the weights of all existing Gaussians to re-normalize the weights. This is done using Equation 5.5. An illustration of this process is depicted in Figure 15.

Figura 15 – Illustration of the process to create a Gaussian. On Figures 15(a), 15(b), the star-shaped observation represents the new test instance. The text G0, ..Gn represent the number of the Gaussian.



(a) Create (Before) (b) Create (After)

Source: Author.

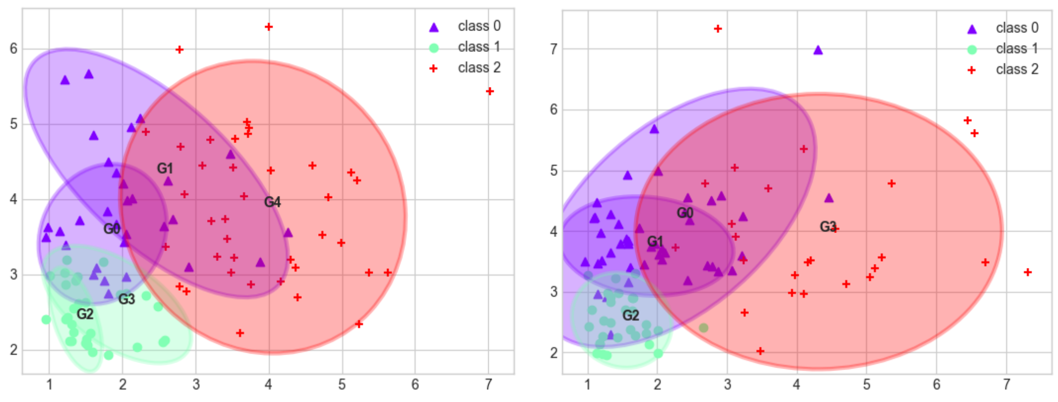
5.1.3.3 *Remove Gaussians*

With the data stream evolution over time, existing Gaussians may become obsolete, because the Gaussian no longer represents new observations in that region of space. Thus, with the normalization of the weights it tends to have weight next to zero. In that case, it is necessary to exclude these Gaussians. A Gaussian is deleted when it has its weight (ω_i) extremely small, less than 0.0001. This process is repeated whenever the adaptation virtual drifts is triggered.

5.1.3.4 *Adaptation to Real Drifts*

We consider that the real concept drifts requiring treatment will significantly degrade the system's predictive performance. Given the adoption of the mechanisms explained in 5.1.3, we also consider that virtual concept drifts will not degrade the system's predictive performance so much as real concept drifts. Therefore, the proposed approach detects the need for treating real concept drifts through a ECDD that monitors the error of the system over time. If ECDD detects a drift, the system collects new data to learn its knowledge from scratch. ECDD has as parameters the tolerance levels defined by c and w, which represent the drift and warning levels, respectively, and can be tuned so that ECDD will detect real 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 using these new observations in case of a concept drift; and (iii) for the drift level, m new observations are collected and added to the observations collected during the warning level to train a new model. This process is illustrated in Figures 17(a) and 17(b). In the second case (17(b)) the model had to be redefined because it was degrading.

Figura 16 – Illustration of the process to retrain a model. On Figures 15(a), 15(b), the star-shaped observation represents the new test instance. The text G0, ..Gn represent the number of the Gaussian.



(a) Initial Model (b) Model Redefined

Source: Author.

5.2 OGMMF-VRD: ON-LINE GAUSSIAN MIXTURE MODEL WITH NOISE FILTER FOR HANDLING VIRTUAL AND REAL CONCEPT DRIFTS

In our previous work GMM-VRD (OLIVEIRA; MINKU; OLIVEIRA, 2019), 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 GMM-VRD.

Given that, in this Section, we describe OGMMF-VRD which overcomes GMM-VRD problems. For this, in Section 5.2.2, we describe its the components to answer to RQ3, being the adaptation to virtual and real drifts and the noise filter. In Section 5.2.3, we describe the strategy of pool adaptation to answer RQ4.

OGMMF-VRD was published in (OLIVEIRA; MINKU; OLIVEIRA, 2020), and its overall procedure is presented in Fig. 5. In the first block, the main system mechanisms, such as GMM batch training, are initialized (Section 5.2.1). In the second block, we perform the on-line classification of the observations received from the data stream 5.1.2. In the third block, we update the batch-trained GMM on-line, i.e., using each new observation received (Section 5.2.2). In the fourth block, we introduce knowledge reset mechanisms for dealing with severe drifts (Section 5.2.3).

To clearly understand how OGMMF-VRD works, we provide the source code on GitHub[1], and we present some consecutive plots (Fig. 19) of its execution on Virtual 9 dataset in a video on youtube[2].

5.2.1 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, as described in Section 5.1.1.

In **drift detector initialization**, we start a CDT to monitor the system error and identify performance degradation. In an experiment we evaluate ECDD (ROSS et al., 2012), CUSUM (PAGE, 1954), FHDDM (PESARANGHADER; VIKTOR, 2016), DDM (GAMA et al., 2004), and EDDM (BAENA-GARCIA et al., 2006). EDDM reached the best results and so was chosen. After initialization, non-severe drifts are treated as in Section 5.2.2.1 and severe drifts as in Section 5.2.2.2.

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

In this Section we will discuss OGMM-VRD's strategies for dealing with non-severe virtual and real drifts (Section 5.2.2.1), dealing with severe real drifts (Section 5.2.2.2) and dealing with noise (Section 5.2.2.3).

[1] https://github.com/GustavoHFMO/OGMMF-VRD
[2] https://www.youtube.com/watch?v=IP-onPHSR0A

Algorithm 5 OGMMF-VRD's overall procedure

```
    {Part I: Train initial model}
 1: W ← $(x_i, y_i)_{i=1}^m$
 2: model ← trainGMM(W)                                         ▷ See Alg. 2
 3: pool ← createPool(model)
 4: detector ← detector(model, W)
    {Part II: On-line classification}
 5: for each instance $x_t$ do
 6:     $\hat{y}_t$ ← predict(model, $x_t$)
 7:     $y_t$ ← trueLabel($x_t$)
    {Part III: Update on virtual drifts}
 8:     model ← nonSevereDriftAdaptation($x_t$, $y_t$, V, k, th)     ▷ See Alg. 6
    {Part IV: Update on real drifts}
 9:     driftLevel ← modelDetector($\hat{y}_t$, $y_t$)            ▷ Monitoring the drift
10:     if driftLevel == "Drift" then
11:         WNew ← increment(($x_t$, $y_t$))
12:     end if
13:     if length(WNew) == $(m \div 30)$ then
14:         model ← reuseGMMs(model, pool, WNew)
15:     else if length(WNew) == m then
16:         model ← trainGMM(W)                                 ▷ See Alg. 2
17:         pool ← insertPool(model)
18:         detector ← detector(model, W)
19:     end if
20: end for
```

5.2.2.1 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. 9 presents the overall procedure.

Algorithm 6 NonSevereDriftAdaptation()

```
    Input: observation ($\boldsymbol{x_t}$, $y_t$), validation set ($V$), neighbors ($k$), noise threshold ($th$)
 1: if NoiseFilter($\boldsymbol{x_t}$, $y_t$, $V$, $k$, $th$) then
 2:     gaussian, pertinence ← GaussianClose($\boldsymbol{x_t}$, $y_t$)     ▷ See Alg. 4
 3:     UpdateGaussian(gaussian, $\boldsymbol{x_t}$)
 4:     if pertinence < $\theta$ then
 5:         CreateGaussian($\boldsymbol{x_t}$, $y_t$)
 6:         UpdateReach(pertinence)
 7:     end if
 8: 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. 4.

This routine computes the conditional probability (Eq. 2.2) 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 5.2.2.1.1 and 5.2.2.1.2).

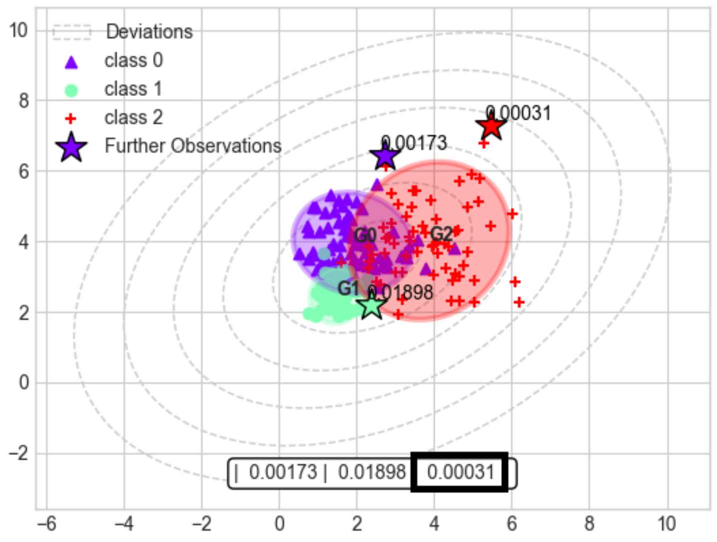
5.2.2.1.1 Adjusting Existing Gaussians

Line 2 (Gaussian Update) of Alg. 6, 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 on-line learning process described in Section 5.1.3.1.

5.2.2.1.2 Adding New Gaussians

Line 4 (Create Gaussian) of Alg. 6 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. 6), 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. 2.2) obtained from the observations in the training set (T). Points more distant than theta (θ) are out of GMM's reach. Fig. 17 illustrates how theta (θ) works.

Figura 17 – Illustration of the process to estimate the parameter theta (θ). The star-shaped observations represent further observations by class. The black square represents the value that will be used in theta (θ).



Source: Author.

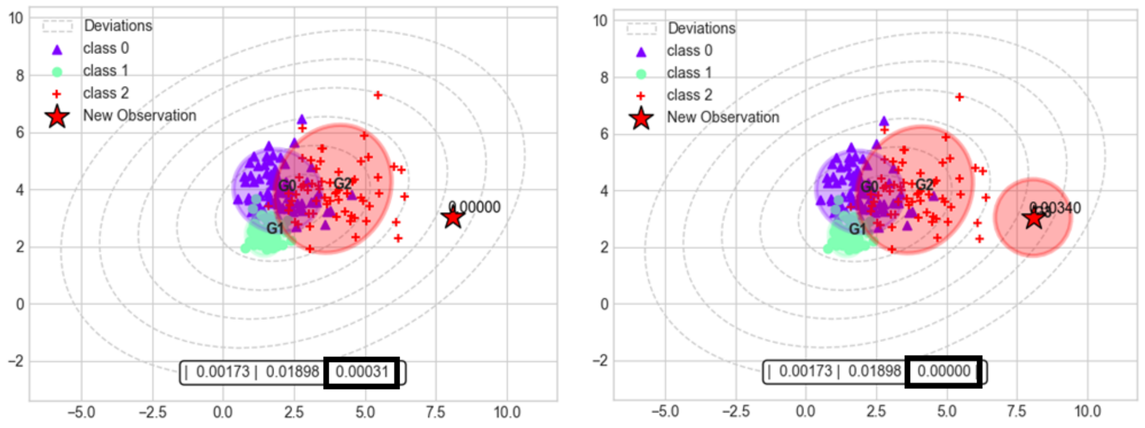
Thus, the new Gaussian is according the description of Section 5.1.3.2. The parameter *Cfc* used to initialize the Gaussian radius is defined using the Eq. 5.8.

$$\mathit{Cfc} = \frac{x_{max} - x_{min}}{20}, \tag{5.8}$$

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. 5). After the initialization of the new Gaussian's parameters, we re-normalize the weights of all existing Gaussians using Eq. 5.5.

Line 5 (Update Reach) represents the process of updating θ and Fig 18 illustrates how it works. For that, we use the pertinence of the incoming observation to substitute the older value (Line 1 of Alg 6). 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.

Figura 18 – Illustration of the process to extend the GMM reach. The star-shaped point represents the incoming observation, and the number above represents the pertinence of this observation to the GMM. Every dotted gray Gaussian represents a standard deviation from the center of GMM. The vector in the bottom of the figure shows the pertinence of the furthest observations by class, where each position represent a class (e.g. vector[0] represent the pertinence for the furthest observation for class 0).



(a) Arrival of a new observation. (b) Creation of a new Gaussian.

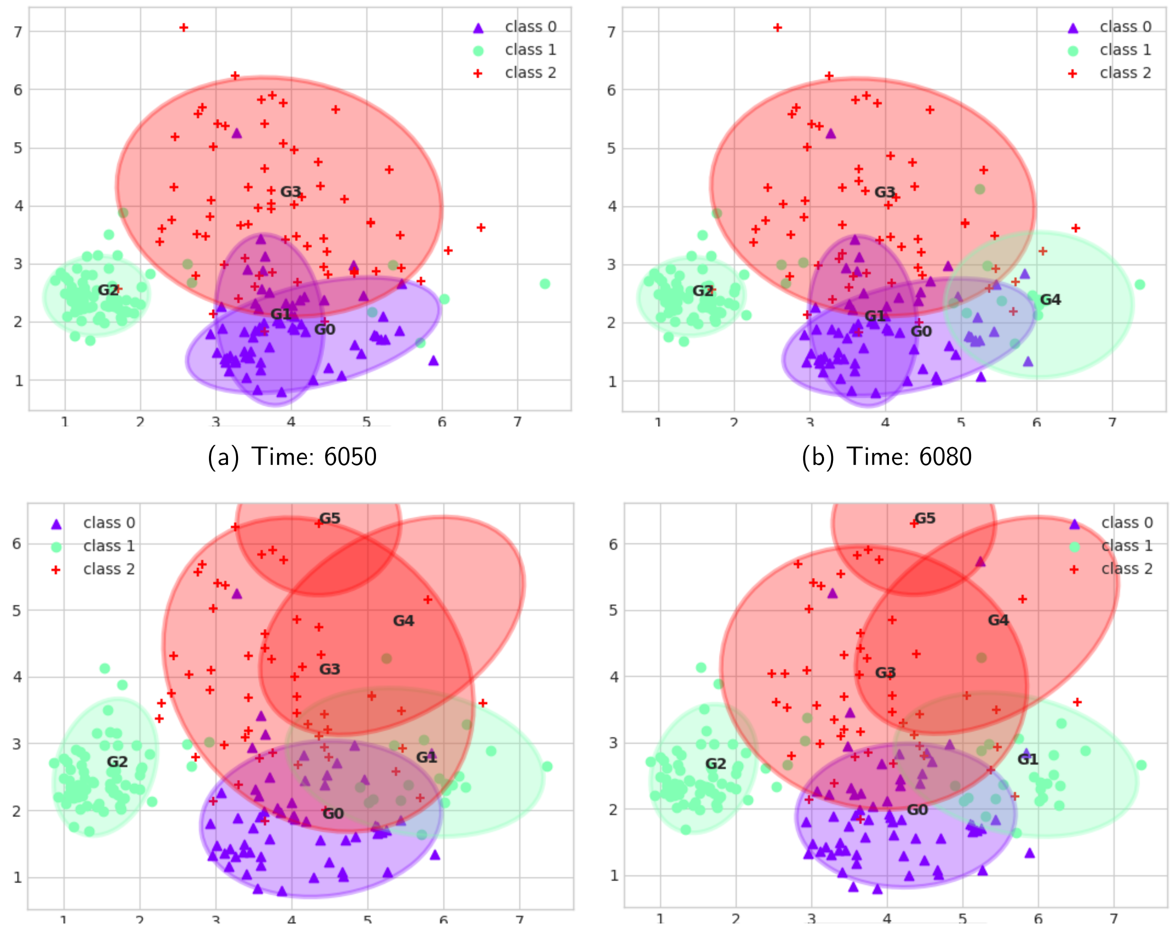
Source: Author.

5.2.2.2 *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 5.2.2.1, 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) (BAENA-GARCIA et al., 2006). 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.

Figura 19 – 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.



(a) Time: 6050

(b) Time: 6080

(c) Time: 6110

(d) Time: 6140

Source: Author.

5.2.2.3 *Noise Filter*

In this Section, we will discuss how to address the part of RQ3 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 Gausian. 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) (WALMSLEY et al., 2018), defined in Eq. 5.1, as a noise filter.

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 5.2.2.1). 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. 5), 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. 5) and the V set acts like a sliding window (Part 3 of Fig. 5), 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.

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

This Section presents our method proposed to answer RQ4, 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 (BAENA-GARCIA et al., 2006) 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 5.2.2.2), i.e. when 30% of m are collected. This is represented by block **Store GMM** in Fig. 5.

Classifiers are estimated from the pool when EDDM (BAENA-GARCIA et al., 2006) 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. 5.

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.

5.3 GLDD-DU: GAUSSIAN LOCAL DRIFT DETECTOR FOR DRIFT UNDERSTANDING

This section presents our proposed approach GLDD-DU. GLDD-DU is able to identify and update or erase only the parts of the decision boundary that have been affected by the concept drift. Therefore, it keeps the useful learned decision boundaries, only updating the parts of the decision boundary that have changed and when necessary. In this way, GLDD-DU is able to swiftly adapt to different types of concept drift, including virtual and real, severe or non-severe drifts.

We describe in Section 5.3.4 the components of the proposed method designed to answer to RQ4, including the process to stimulate the generation of knowledge to help adaptation to new concepts using GMM. In addition, in Section 5.3.5 the components to answer RQ6 are presented, namely the strategy to update only the portions of the learned decision boundaries that were affected by the concept drift. These two components are highlighted with blue lines in the GLDD-DU procedure in Figure 8.

GLDD-DU's general procedure is divided into four blocks discussed in detail in the next subsections. In Section 5.3.1, we describe the first and second blocks, which are the process to train the GMM, intialise the drift detector and perform the online classification of the observations received from the data stream. In Section 5.3.2, we describe the third block, which applies strategies to deal with virtual and non-severe drifts, updating the batch-trained GMM online, and creating new Gaussians for the observations received. Finally, in Section 5.3.3, we describe the fourth block, which applies strategies to deal with severe and non-severe real drifts, being responsible for the replacement of local and global learned decision boundaries.

Algorithm 7 SystemInitialization()

```
    Input: W ← $(x_i, y_i)_{i=1}^m$                          ▷ Initial Batch
 1: Train, Val ← SplitBatch(W)                    ▷ Training and Validation Sets
 2: GMMs ← TrainModels(Train)
 3: Pool ← CreatePool(GMMs)
 4: BestGMM ← BestModel(GMMs, Val)
 5: ModelDetector ← StartDetector(GMMs, W)
 6: GaussianDetectors ← StartDetector(GMMs, W)
 7: return W, Pool, BestGMM, ModelDetector, GaussianDetectors
```

Algorithm 8 GLDD-DU general procedure.

```
    {Part I: Initialization}
 1: BestGMM ← SystemInitialization(W)                              ▷ See Alg. 7
    {Part II: On-line Classification}
 2: for each $x_t$ in Data Stream: do
 3:     $\hat{y}_t$ ← BestGMM.predict($x_t$)
 4:     $y_t$ ← trueLabel($x_t$)
 5:     W ← slideWindow(W, ($x_t$, $y_t$))
    {Part III: Non-Severe Drifts}
 6:     BestGMM ← NonSevereDriftAdaptation($x_t$, $y_t$, V, k, th)       ▷ Alg. 9
    {Part IV: Severe Drifts}
 7:     driftLevel ← ModelDetector($\hat{y}_t$, $y_t$)              ▷ Monitoring the drift
 8:     if driftLevel == "Normal" then
 9:         if Drift in GaussianDetectors then
10:             BestGMM ← ReuseGaussians(BestGMM, Pool, W)
11:         end if
12:     else if driftLevel == "Warning" then
13:         WNew ← Increment(($x_t$, $y_t$))
14:     else if driftLevel == "Drift" then
15:         WNew ← Increment(($x_t$, $y_t$))
16:         if length(WNew) < m then
17:             if Drift in GaussianDetectors then
18:                 BestGMM ← ReuseGaussians(BestGMM, Pool, WNew)
19:             end if
20:         else if length(WNew) == m then
21:             BestGMM ← SystemInitialization(WNew)                 ▷ See Alg. 7
22:         end if
23:     end if
24: end for
```

5.3.1 Initialization

The GLDD-DU framework is initialised through three steps: (i) **GMM training**, (ii) **Drift Detector Initialization**, and (iii) **Define Classes Reach**. The last one is discussed in Section

5.3.2 since this component is used to create new Gaussians.

In GMM Training, we collect an initial portion (m observations) of the data stream as the training set (T) used to initialize a GMM, as described in Section 5.1.1.

In Drift Detector Initialization, using the trained GMM, we start a drift detector to monitor its error and identify performance degradations. The drift detector is just one component of the GLDD-DU, so that any error-based drift detector can be applied. In our preliminary experiment we evaluate ECDD (ROSS et al., 2012), CUSUM (PAGE, 1954), FHDDM (PESARANGHADER; VIKTOR, 2016), DDM (GAMA et al., 2004), and EDDM (BAENA-GARCIA et al., 2006), and it indicated that the Early Drift Detection Method (EDDM) (BAENA-GARCIA et al., 2006) obtained the best results for our approach. EDDM is thus used in Section 5.3.3 to activate the severe drift adaptation. An extensive comparison between different drift detection methods is out of the scope of this work, which focuses on drift adaptation strategies rather than drift detection mechanisms.

5.3.2 Non-severe Drifts Adaptation

This part of the GLDD-DU aims to maintain the useful knowledge of the system in the presence of virtual and non-severe real drifts. Alg. 9 presents the overall procedure.

Algorithm 9 NonSevereDriftAdaptation()

```
   Input: observation ($\boldsymbol{x_t}$, $y_t$), validation set ($V$), neighbors ($k$), noise threshold ($th$)
1: if NoiseFilter($\boldsymbol{x_t}$, $y_t$, $V$, $k$, $th$) then
2:     gaussian, pertinence ← NearestGaussian($\boldsymbol{x_t}$, $y_t$)          ▷ See Alg. 4
   {Real Non-severe Drifts Adaptation}
3:     UpdateGaussian(gaussian, $\boldsymbol{x_t}$)
   {Virtual Drifts Adaptation}
4:     if pertinence < $\boldsymbol{\theta}[y_t]$ then
5:         CreateGaussian($\boldsymbol{x_t}$, $y_t$)
6:         $\boldsymbol{\theta}[y_t]$ ← pertinence                    ▷ Update Class Reach
7:     end if
8: end if
```

Line 1 represents a mechanism to avoid noisy observations described in Section 5.2.2.3.

Line 2 of Alg. 9, 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 purpose, we use Alg. 4. It returns the nearest Gaussian with the same class of the incoming observation (Line 9 of Alg.

9), and the pertinence of the incoming observation to the nearest Gaussian (Line 10). With this information, we can trigger appropriate drift handling strategies: (i) Adjusting Existing Gaussians and (ii) Adding New Gaussians, both discussed below.

Line 3 (Gaussian Update) of Alg. 9, represents the process designed to adapt to non-severe real drifts, described in Section 5.1.3.1.

Line 5 (Create Gaussian) of Alg. 9 represents the process to adapt to virtual drifts. To this end, when the pertinence of the incoming observation ($\boldsymbol{x}_t$) to the nearest gaussian is less than $\boldsymbol{\theta}_i$ (Line 3 of Alg. 9), we create a new Gaussian. Theta ($\boldsymbol{\theta}$) is initialised in the block **Define Classes Reach** in block 1 in Fig. 8, and represents a vector with the number of dimensions equal to the number of classes of the problem. Each position of the vector stores the lowest pertinence (Eq. 2.2) obtained for each class in the training set (T). Using this, we can verify the reach in terms of the distance that the GMM modeling has arrived. If an instance is very distant from its class, then it got smaller pertinence than ($\boldsymbol{\theta}_i$), which means that it is in a region not learned. Thus, a new Gaussian is initialised according the descriptions of Section 5.2.2.1.2.

Line 6 of Alg. 9 represents the block 3 (**Update Class Reach**) in Fig. 5, where we update the values of $\boldsymbol{\theta}$. For that, we use the pertinence of the incoming observation ($\boldsymbol{x}_t$) to substitute the older value in $\boldsymbol{\theta}_i$ in the position of its class (y_t). This update indicates that, for the creation of new Gaussians, the incoming observation must have a lower pertinence than $\boldsymbol{\theta}_i$, indicating that it is farther away.

5.3.3 Severe Real Drifts Adaptation

This part of the GLDD-DU aims to reset unsuitable system knowledge with a global replacement. As explained in Section 5.3.1, we use EDDM to monitor the overall performance of the whole system. If EDDM detects a drift, the system collects new data to learn its knowledge from scratch. EDDM has as parameter drift level (c) which can be tuned so that it only detects severe drifts, which will degrade the system's performance more than non-severe drifts, given the strategy to deal with non-severe drifts presented in Section 5.3.2. Each tolerance level yields a different response: (i) for the normal error level the model remains untouched; (ii) for the drift level, new observations are collected and added into a batch, and when m new observations are collected, they represent a new training set (T) to initialise a new model as shown in the block 4.1) Global Replacement in Fig. 8.

5.3.4 Generation of GMM and Gaussian Pools

In this Section, we will discuss how to address part of RQ5, namely, the process to improve the GMM-based system predictive performance using a pool of GMMs. This procedure is shown in blue block **Pools Generation** in block 1 of Fig. 8. This process is datailed in Fig. 20 which represents the general procedure for generating the **pools of** (i) **GMMs and** (ii) **Gaussians**.

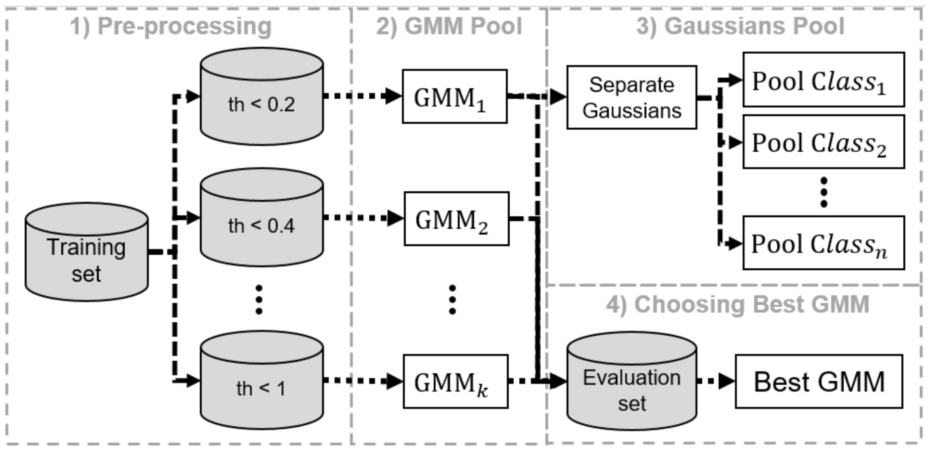
5.3.4.1 GMM Pool

Although GMM has a random initialization, it tends to create the same model because of the characteristics of the EM algorithm (BAILEY; ELKAN et al., 1994), and this makes it difficult to improve the GMM generalization. So, to solve this, we use kDN (Eq. 5.1) as a pre-processing method to generate different subsamples from the training set (T) to generate diverse GMMs. The process of generating different subsamples of the training set (T) consists of selecting the training examples that have kDN smaller than a pre-established value (th). For instance, if th = 0.2, we choose only those training examples with kDN < 0.2. Note that in this strategy, the kDN is varied only to generate the subsamples. The kDN value to remove noise online is fixed.

In Fig. 20, 1) **Pre-processing** part, the number of subsamples depends on the number of models (N) that will be trained. Then, the first subsample will have as threshold $th = (1/N)$, the second $th = (2/N)$, up to $th = (N/N) = 1$. We set the number of neighbors k for the kDN equal to 10. Note that as the threshold increases, so does the number of observations in the subsample as shown in Fig. 21. This strategy is important when a drift happens, because the generation of different subsamples of the data recently collected (training set (T)), can help the system to adapt more quickly to the distribution of the new concept. To understand this, suppose a GMM trained for a fixed value of th for kDN. Some observations that could be part of the new concept, can be confused with noise by the noisy filter, which could delay the adaptation of GMM on the new data. Therefore, once we generate GMMs with different capacities, there may be one that has already considered the observations of the new concept (while potentially still filtering out some appropriate level of noise) and will consequently perform better than the others.

In this way, to select the GMM with better predictive performance, before performing the pre-processing, we divide the training set (T) into 80% to generate the subsamples, and 20% to evaluate the models. The GMM used to perform the online classifications is the one with the best predictive performance for the 20% of T as shown in part **4) Choosing the best GMM** in Fig 20.

Figura 20 – Architecture to generate the pools and obtain the best GMM.

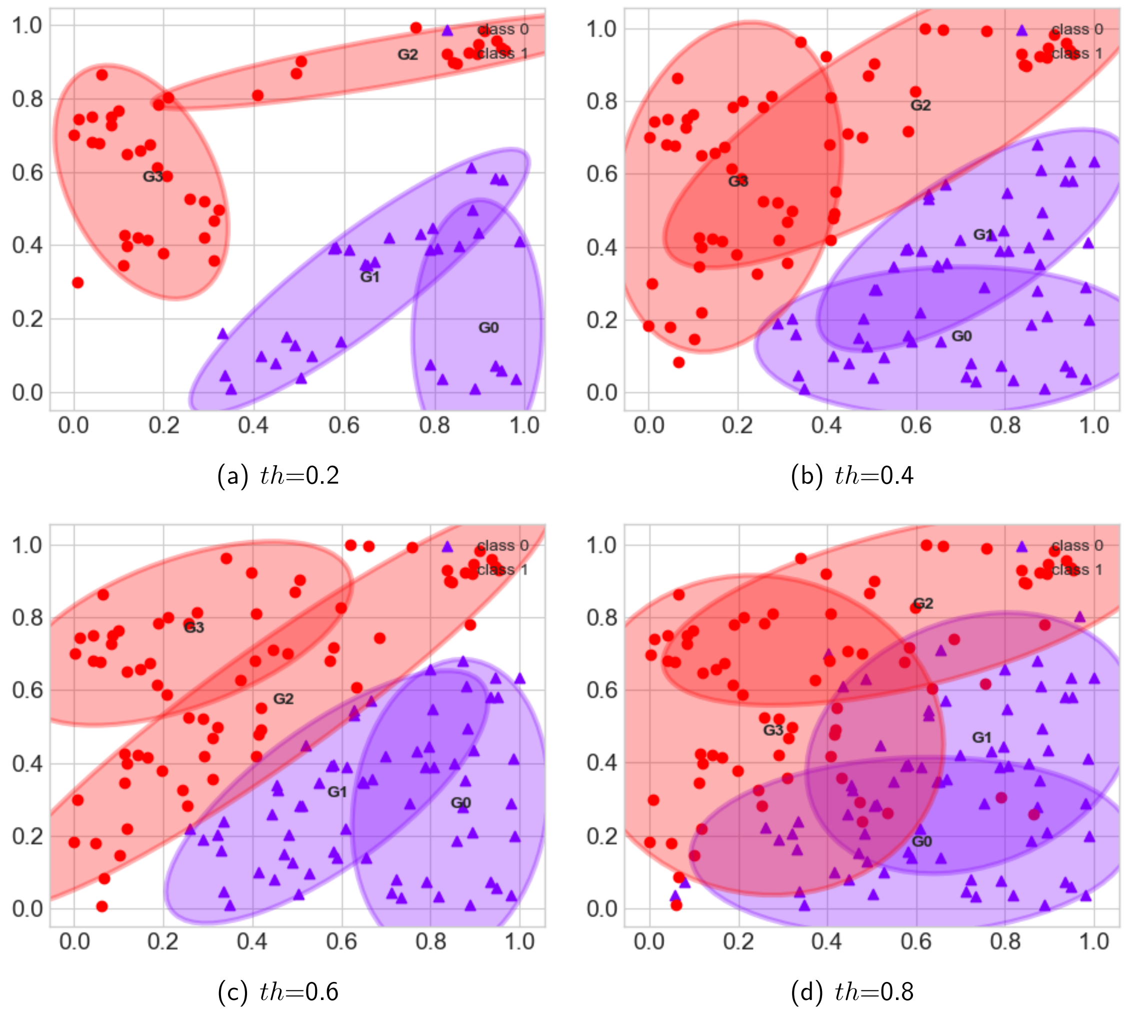


Source: Author.

5.3.4.2 *Gaussians Pool*

As presented in Fig. 20, the creation of the Gaussian Pools depends on the generation of the GMM Pool. For each GMM generated, the Gaussians are reserved according to their class. Thus, the number of Gaussian Pools depends on the number of classes existing in the training set (T). It is essential to say that the Gaussian Pools are a tool that will be used to update local decision boundaries, discussed in Section 5.3.5. Whenever new Gaussians are generated, they are added to the Gaussian pools. They are added in two moments: (i) when we retrain the model with the severe drift adaptation (Section 5.3.3); and (ii) when we create a Gaussian with the non-severe adaptation method (Section 5.3.3). So, we define as (P) the maximum size of models in Gaussian pools. If the pool is full, the new model is added, and the older one is removed.

Figura 21 – GMMs generated for different subsamples of the training set (Each subsample was generated from a different threshold (th) for kDN).



(a) th=0.2
(b) th=0.4
(c) th=0.6
(d) th=0.8

Source: Author.

5.3.5 Local Replacement of Decision Boundaries

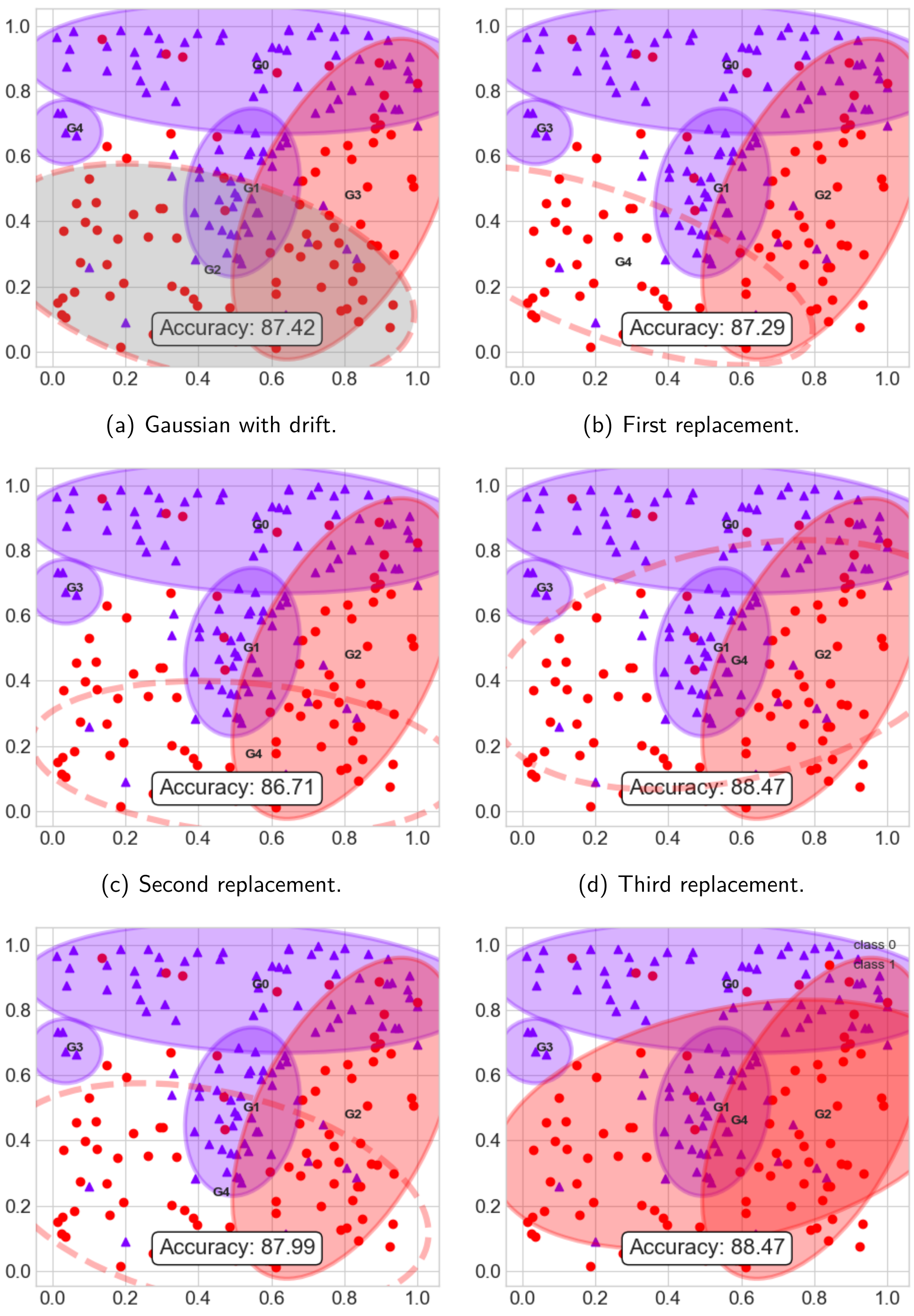
In this Section, we will discuss how to address part of RQ6 which seeks to update only a portion of the learned decision boundaries that were affected by the concept drift. The proposed procedure to achieve this goal is shown in blue block 4.2 of Fig. 8. To this end, we employ the current GMM used to make the online classifications.

For each Gaussian in the GMM, we assign an EDDM drift detector to monitor its performance separately. So, if we have 10 Gaussians in the GMM, we will have 10 respective EDDM models. The function *argmax* in Eq. 2.7 returns the index of the Gaussian that classified the observation, where this index ranges from 1 to K. This index is used to update the respective Gaussian detector. Note that EDDM monitors the Gaussian's error receiving true if the class of the example (y_t) matches with the class of the Gaussian returned $(\hat{y}_i)$, otherwise it is set to false.

All Gaussian detectors are initialised in block 1 (**Start Drift Detector**) of Fig 8. To allow the Gaussian detectors to be more sensitive than the GMM detector (Section 5.3.3), we decrease the number of observations to initiate the online detections to $(m/4)$, where m is the batch size used to train the GMM. When a Gaussian detector reports that the Gaussian is underperforming, we reset its drift detector and we reuse the Gaussians from the Pool. For each Gaussian suffering from drift, new Gaussians from the pool with the same class are tested to replace it in the GMM, and if one leads a better predictive performance, we replace. As these Gaussians were generated for different kDN configurations, some may have already started to deal with drift because they may have been trained with more data representative of the new concept. So, we evaluate these Gaussians using either accuracy or G-mean as a selection metric, and the observations used to calculate these metrics are discussed below, since they change according to the error obtained by the drift detector of the whole GMM. This procedure is illustrated in Fig. 22.

EDDM categorizes the GMM error as normal or drift levels (see Section 5.3.3). Normal level means that GMM is still classifying new observations well, so this may indicate that only a local drift occurred and not affected so much the GMM performance. For that reason, we evaluate the predictive performance of the Gaussians from the pool using the observations of the same concept (Validation set (V)) to improve the whole GMM predictive performance. In the drift level, we evaluate the Gaussians predictive performance using the observations recently inserted in the training set (T). In this case, we have enough indicators for a global drift because it affected so much the GMM performance. Thus, when we collect at least 20% of m (batch size), we firstly replace each Gaussian affected by the drift by another of the same class but with better predictive performance for these observations. This process allows the system to adapt to the drift quickly while waiting for more observations to arrive for retraining.

Figura 22 – Procedure to replace a Gaussian with drift in the dataset Sine 2. Figure 23(a) present a Gaussian with drift highlighted in gray. Figures 23(b) to 23(e) highlight without background colors the Gaussians from the pool used to replace the Gaussian affected by drift, leading to different accuracy values. Note that the third replacement 23(d) resulted in the best accuracy and will thus be chosen to help to adapt to the drift according to presented in the best model obtained in Figure 23(f).



(a) Gaussian with drift.

(b) First replacement.

(c) Second replacement.

(d) Third replacement.

(e) Fourth replacement.

(f) Best model.

Source: Author.

5.4 CHAPTER SUMMARY

In this Chapter we present three methods called the GMM-VRD (GMM-VRD) (OLIVEIRA; MINKU; OLIVEIRA, 2019), OGMMF-VRD (OGMMF-VRD) (OLIVEIRA; MINKU; OLIVEIRA, 2020), and GLDD-DU (GLDD-DU).

GMM-VRD implements the RQ2 and presents early mechanisms for dealing with virtual and real concept drifts at the same time. GMM-VRD is trained using a batch of observations to initialize the system before handling the data stream. It redefines its knowledge to deal with real drifts based on monitoring its performance through a drift detector. To deal with virtual drifts, it updates its knowledge using on-line learning for each observation received. Also, it creates new Gaussians to learn the regions of observations from untrained distributions.

OGMMF-VRD extends GMM-VRD and implements the RQ3 and RQ4, providing a series of solutions to its weaknesses. As solutions, OGMMF-VRD implements (i) a filter to differentiate noise-free observations to noisy observations; (ii) a mechanism to measure which are the new regions that must be learned to deal with virtual drifts; and (iii) to deal with real drifts, the recovery of GMMs obtained from a pool.

GLDD-DU implements the RQ5 and RQ6 and applies drift understanding to use the appropriate strategies according to the level of severity of the drift. GLDD-DU is able to identify and update or erase only the parts of the decision boundary that have been affected by the concept drift. Therefore, it keeps the useful learned decision boundaries, only updating the parts of the decision boundary that have changed and when necessary.

In Chapter 6, we evaluate how well these above approaches validated the research questions. Finally, we summarized in in the Tbl. 4 their main differences.

Tabela 4 – Differences between the GMM-VRD, OGMMF-VRD and GLDD-DU. Each row represents a specific difference between them.

Nº	Features	GMM-VRD (OLIVEIRA; MINKU; OLIVEIRA, 2019)	OGMMF-VRD (OLIVEIRA; MINKU; OLIVEIRA, 2020)	GLDD-DU (Submitted to Machine Learning)
1	Noise filter before creating or updating a Gaussian.	No	Yes	Yes
2	Mechanism to define the range of a Gaussian. Creates Gaussians for new regions quickly.	No	Limits of GMM.	Limits of each class.
3	Gaussians are sized according to the problem.	No	Yes	Yes
4	Incremental learning for all instances.	Only on model error.	Yes	Yes
5	Experiment with drift detectors. 5 detectors were evaluated.	ECDD	EDDM	EDDM
6	Models are saved in a pool for future use.	No	GMM is replaced.	Gaussians are replaced.
7	Models to expedite the adaptation of real drifts.	No	Yes	Yes
8	Drift detector for each GMM Gaussian. The Gaussians are reset to handle drifts.	No	No	Yes
9	A pool of GMMs is generated during training. The best model is used in online classification.	No	No	Yes

6 EXPERIMENTS

In this Chapter, we present the setup of our experiments and their results. Five experiments were realized: (i) comparison with literature works (Section 6.3) that answer RQ2; (ii) noise filter robustness (Section 6.4) that partially answer RQ3; (iii) impact of virtual and real drifts mechanisms (Section 6.5) that answer RQ3 and RQ4; (iv) impact of drift understanding mechanisms (Section 6.6) that answer RQ5 and RQ6; and (v) proposed methods' hyperparameter sensitivity analysis (Section 6.7). All experiments were evaluated using the datasets discussed in Section 4.1 and the metrics and statistical tests discussed in Section 6.1.

6.1 METRICS

We use different metrics to evaluate the predictive performance of the experimented approaches. In Subsection 6.1.1, we present Cross Validation For Data Streams; in Subsection 6.1.3, we present Accuracy Over Time; in Subsection 6.1.2, we present Average Accuracy and G-mean; and in Subsection 6.1.4, we present Runtime.

6.1.1 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 (SUN et al., 2016). In this version, the approaches first classify the arriving observations: first test and then the train. There is no validation set, as all observations are part of the test set. Folds are created by removing 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 second thirty elements and so on. For all datasets, 30 runs were executed, i.e. X=30. The element removed corresponds to the order of the execution.

6.1.2 Overall Accuracy and G-mean:

Accuracy and G-mean are used to evaluate the online predictions given by the system, according to Eq. 6.1 and Eq. 6.2, respectively.

Average Accuracy is a performance indicator to evaluate how the system behaves across the whole data stream, where values close to 1 represent better results. This metric was used in several works as (SUN et al., 2016; OLIVEIRA; BATISTA, 2015; ALMEIDA et al., 2018).

$$Accuracy = \frac{\text{Number of correct predictions}}{\text{Total number of predictions made}} \tag{6.1}$$

G-mean is an indicator that evaluates the model performance on databases with unequally sized classes. It is measured by the mean of the recall on each class, thus, being independent of the level of class imbalance in the data (WANG; MINKU; YAO, 2018).

$$G - mean = \sqrt{Recall \cdot Specificity} \tag{6.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{6.3}$$

$$Specifity = \frac{TN}{TN + FP} \tag{6.4}$$

In Eq. 6.3, and Eq. 6.4, TP (True Positive) is the number of positive cases correctly classified. FN (False Negative) is the number of positive cases incorrectly classified as negatives. FP (False Positive) is the number of negative cases that are incorrectly identified as positive cases. Finally, TN (True Negative) is the number of negative cases correctly classified.

6.1.3 Accuracy Over Time

Accuracy Over Time (AOT) is a time series that shows the system's accuracy over time, on which each plotted value represents the predictive performance of the system over a batch of past observations as shown in the Equation 6.5. Each batch is generated with each amount of information X, for example: if X = 250: $batch_1$ = [0 to 250], $batch_2$ = [251 to 500], $batch_3$ = [501 to 750] ... until the final dataset quantity. This metric enables us to investigate the evolution of the system's accuracy over time and was also used in several previous studies as (SUN et al., 2016; OLIVEIRA; BATISTA, 2015; ALMEIDA et al., 2018).

$$AOT = \{accuracy(batch_1), ..., accuracy(batch_n)\} \tag{6.5}$$

6.1.4 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.

6.2 STATISTICAL TESTS

To attest to a significant difference from our experiments we will use three statistical tests: (i) Friedman Test with Nemenyi Post-Hoc, discussed in Subsection 6.2.1; (ii) Wilcoxon For Average, discussed in Subsection 6.2.2; and (iii) Two-Way ANOVA, discussed in Subsection 6.2.3.

6.2.1 Friedman Test and Nemenyi Post-Hoc

Friedman is a non-parametric statistical hypothesis test that can be used to compare multiple approaches across multiple datasets (DEMŠAR, 2006). 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 test is used to check which of the approaches is significantly different from each other. Both tests were used in this work with a significance level $\alpha = 0.05$.

6.2.2 Wilcoxon For Average

This is a non-parametric statistical hypothesis test used in our experiments to evaluate how the average accuracy of a specific mechanism has improved over the average accuracy of the complete system. This test was used with a significance level $\alpha = 0.05$.

6.2.3 Two-Way ANOVA

We use this test to evaluate the parameters of the proposed approach due to the fact that ANOVA enables us to check the interaction between them whereas the Friedman test does not. Parameters are called factors and parameter levels are represented by the accuracy of each setup for a given dataset. The null hypothesis is that there is no difference in the interaction, while the alternative hypothesis is that there is a difference. This test was used with a significance level $\alpha = 0.05$. Also, we demonstrate through the effect Eta the percentage in which each interaction impacted the system. The main assumption done by split-plot ANOVA is the sphericity. If the sphericity assumption is violated, the split-plot ANOVA can get high type I error (reject the null hypothesis when it was true). Mauchly's test (MAUCHLY, 1940) can be used to detect violations of the sppherecity assumption. If violations are detected, corrections such as Greenhouse-Geisser (GREENHOUSE; GEISSER, 1959) can be applied to the ANOVA's p-value so that the type I error will not be increased.

6.3 COMPARISON EXISTING APPROACHES

This experiment aims to partially answer RQ2, which asks how to efficiently deal with virtual drifts at the same time as preserving the ability to deal with real drifts when using GMM. Besides that, it evaluates the performance of our proposed approaches, GMM-VRD, OGMMF-VRD, and GLDD-DU, in comparison with existing work. The discussions are divided into Section 6.3.1 (comparison against literature approaches with separate mechanisms to deal with virtual and real drifts) and Section 6.3.2 (comparison against other approaches from the concept drift literature).

For the first comparison, we compare GMM-VRD and OGMMF-VRD with methods that are not based on ensembles and that use single classifiers similar to GMM. This helps us to attribute improvements in performance to the proposed mechanisms, rather than to the base learners. This selection resulted in three approaches: LDD-DSDA (LIU et al., 2017), that use drift understanding, and IGMM-CD (OLIVEIRA; BATISTA, 2015), and Dynse (ALMEIDA et al., 2018) (see Section 3). LDD-DSDA does not consider virtual and severe real drifts, whereas IGMM-CD and Dynse do not simultaneously deal with virtual and real drifts. For all approaches, we performed a grid search on its most important parameters, considering the values shown in Tbl. 5.

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) (WANG et al., 2003), Adaptive Random Forest (ARF) (GOMES et al., 2017b), Leveraging Bagging ensemble classifier (LevBag) (BIFET; HOLMES; PFAHRINGER, 2010), and Oza Bagging Ensemble classifier (OzaAS) with and without ADWIN drift detector

(OzaAD) (OZA, 2005). Except for ARF, all used Gaussian Naive Bayes (GNB) as the base classifier. Methods based on drift detectors are: ADWIN (BIFET; GAVALDA, 2007), DDM (GAMA et al., 2004), and EDDM (BAENA-GARCIA et al., 2006), combined with Hoeffding Adaptive Tree classifier (HAT) (BIFET; GAVALDÀ, 2009). All of these approaches are available on-line in the library scikit-multiflow[1] and the chunk size was chosen as part of the hyperparameter tuning procedure based on a grid search presented in Tbl. 5; the other parameters were the default values provided by the library.

Tabela 5 – Best parameters found for the compared approaches for each type of dataset. For that, a grid search was executed as presented in the fourth column. The best parameter setting is the one that reached the best result considering the average accuracy across datasets.

Algorithm	Parameters	Parameter Description	Grid Search	Synthetic	Real
LDD-DSDA	m	Batch size	[50, 100, 200, 300, 400]	50	200
	BC	Base classifier	-	GNB	GNB
IGMM-CD	Sigma_ini	Gaussian size	[0.5, 1, 2, 5, 10]	0.05	10
	Cver	Gaussian distance	-	0.01	0.01
	T	Number of gaussians	[1, 5, 7, 9, 13]	13	13
Dynse	D	Pool size	-	25	25
	m	Batch size	[50, 100, 200, 300, 400]	50	50
	M	Window size	-	100	100
	k	Number of neighboors	-	5	5
	CE	DCS approach	-	A Priori	A Priori
	PE	Pruning approach	-	Age Based	Age Based
	BC	Base classifier	-	GNB	GNB
GMM-VRD	m	Batch size	[50, 100, 200, 300, 400]	50	200
	EM it.	EM iterations	-	10	10
	kmax	Number of gaussians	[2, 4, 6, 8]	2	2
	kDN	Noise level	-	5	5
	CDT	Drift detector	-	ECDD	ECDD
	c	Drift level	-	1	1
	w	Warning level	-	0.5	0.5
OGMMF-VRD	P	Pool size	-	20	20
GLDD-DU	P	Pool size	[5, 10, 15, 20]	10	10
	N	Number of models	[1, 3, 5, 7]	5	5
	k	Number of neighboors	-	10	10
	kDN	Noise level	-	0.5	0.5
	CDT	Drift detector	-	EDDM	EDDM
Ensembles	m	Batch size	[50, 100, 200, 300, 400]	50	200
HAT + CDT	m	Batch size	[50, 100, 200, 300, 400]	50	200

[1] https://scikit-multiflow.readthedocs.io/en/stable/index.html

Tabela 6 – Accuracy for approaches compared for all experimented datasets. The results highlighted with bold represent the best accuracy in comparison with the other approaches. Values in brackets are the standard deviations. The last line represents the algorithm position in the Friedman ranking.

Dataset	**IGMM-CD**	**Dynse**	**GMM-VRD**	**OGMMF-VRD (Filter)**	**OGMMF-VRD (No Filter)**	**GLDD-DU (Accuracy)**	**GLDD-DU (Gmean)**	**LDD-DSDA**	**HAT-DDM**	**HAT-EDDM**	**HAT-ADWIN**	**AWE**	**LevBag**	**OzaAD**	**OzaAS**	**ARF**
Circles	0.609 (0.033)	0.741 (0.033)	0.787 (0.007)	0.813 (0.005)	0.804 (0.005)	0.806 (0.005)	0.798 (0.004)	0.828 (0.003)	0.6189 (0.005)	0.4926 (0.003)	0.5592 (0.013)	**0.8489 (0.001)**	0.8122 (0.001)	0.7373 (0.008)	0.7247 (0.001)	0.815 (0.003)
Sine1	0.536 (0.023)	0.603 (0.090)	0.817 (0.004)	0.827 (0.005)	0.824 (0.004)	0.825 (0.004)	0.823 (0.003)	0.803 (0.003)	0.5653 (0.025)	0.4975 (0.003)	0.5094 (0.005)	**0.8286 (0.002)**	0.8082 (0.001)	0.6584 (0.017)	0.5658 (0.002)	0.8137 (0.003)
Sine2	0.538 (0.024)	0.584 (0.073)	0.722 (0.006)	0.750 (0.004)	0.739 (0.004)	0.765 (0.003)	0.761 (0.005)	0.743 (0.002)	0.7044 (0.003)	0.4978 (0.004)	0.6585 (0.013)	0.7532 (0.001)	0.7319 (0.002)	0.5893 (0.016)	0.5525 (0.001)	**0.7886 (0.003)**
Virtual5	0.755 (0.003)	0.792 (0.004)	0.815 (0.003)	0.827 (0.006)	0.824 (0.003)	0.841 (0.004)	0.838 (0.004)	0.831 (0.002)	0.6941 (0.01)	0.3987 (0.002)	0.5127 (0.01)	**0.8504 (0.001)**	0.7987 (0.001)	0.7702 (0.005)	0.7368 (0.002)	0.838 (0.004)
Virtual9	0.804 (0.007)	0.805 (0.007)	0.845 (0.005)	0.858 (0.006)	0.854 (0.005)	0.869 (0.005)	0.868 (0.003)	0.852 (0.002)	0.7082 (0.011)	0.4104 (0.002)	0.492 (0.014)	**0.8697 (0.001)**	0.8104 (0.004)	0.7658 (0.014)	0.7267 (0.001)	0.8451 (0.003)
SEA	0.641 (0.008)	0.725 (0.004)	0.779 (0.008)	0.841 (0.004)	0.832 (0.007)	0.844 (0.005)	0.839 (0.005)	0.832 (0.003)	0.6975 (0.004)	0.5046 (0.003)	0.5233 (0.004)	**0.8534 (0.001)**	0.8295 (0.002)	0.8057 (0.003)	0.8049 (0.001)	0.8335 (0.004)
SEARec	0.641 (0.006)	0.747 (0.003)	0.777 (0.004)	0.841 (0.004)	0.833 (0.002)	0.846 (0.003)	0.845 (0.004)	0.837 (0.002)	0.7146 (0.025)	0.5014 (0.002)	0.5279 (0.015)	**0.8558 (0.001)**	0.8304 (0.001)	0.8106 (0.002)	0.8094 (0)	0.8291 (0.003)
NOAA	0.420 (0.001)	0.462 (0.002)	0.708 (0.009)	0.743 (0.004)	0.723 (0.006)	0.740 (0.006)	0.736 (0.007)	0.694 (0.005)	0.7072 (0.004)	0.6862 (0.001)	0.6989 (0.007)	0.7416 (0.002)	0.7098 (0.003)	0.6634 (0.011)	0.6869 (0.015)	**0.7656 (0.004)**
ELEC	0.783 (0.001)	0.651 (0.001)	0.685 (0.008)	0.745 (0.004)	0.755 (0.005)	0.756 (0.003)	0.754 (0.005)	0.735 (0.002)	0.6966 (0.013)	0.6752 (0.005)	0.6967 (0.006)	0.7112 (0.001)	0.7606 (0.003)	0.7059 (0.004)	0.6785 (0.001)	**0.7935 (0.003)**
PAKDD	0.604 (0.007)	0.660 (0.002)	0.702 (0.016)	0.773 (0.001)	0.785 (0.001)	0.780 (0.009)	0.700 (0.010)	0.395 (0.007)	0.8002 (0.001)	0.8024 (0)	0.8001 (0.001)	0.6205 (0.041)	0.7109 (0.005)	0.3951 (0.026)	0.5484 (0.037)	**0.7955 (0.001)**
GasSensor	0.009 (0)	0.246 (0.001)	0.3333 (0.079)	0.5785 (0.022)	0.589 (0.019)	0.374 (0.012)	0.367 (0.011)	0.823 (0.004)	0.3729 (0.008)	0.3608 (0.003)	0.3619 (0.001)	0.3326 (0.003)	**0.8464 (0.002)**	0.565 (0.003)	0.5552 (0.002)	0.8053 (0.007)
INSECT-Inc-Rec-Bal	0.3951 (0.014)	0.656 (0.001)	0.4941 (0.032)	0.6555 (0.004)	0.651 (0.004)	0.497 (0.008)	0.439 (0.005)	0.686 (0.002)	0.4636 (0.005)	0.4872 (0.001)	0.4694 (0)	0.5788 (0.005)	0.6789 (0.007)	0.5007 (0.007)	0.4851 (0.001)	**0.7228 (0.005)**
INSECT-Inc-Abt-Bal	0.3938 (0.013)	0.636 (0.001)	0.4861 (0.031)	0.6464 (0.005)	0.645 (0.006)	0.502 (0.010)	0.435 (0.006)	0.667 (0.002)	0.4843 (0.003)	0.508 (0)	0.4919 (0)	0.5613 (0.006)	0.6708 (0.005)	0.6011 (0.004)	0.5865 (0.001)	**0.7024 (0.004)**
INSECT-Grad-Bal	0.473 (0.003)	0.646 (0.001)	0.4876 (0.007)	0.5866 (0.005)	0.586 (0.010)	0.403 (0.015)	0.360 (0.013)	0.683 (0.004)	0.3484 (0.01)	0.3841 (0.001)	0.3631 (0.001)	0.6013 (0.006)	0.6921 (0.003)	0.6232 (0.032)	0.5441 (0.001)	**0.703 (0.017)**
Friedman Ranking	12.14 (3.38)	9.81 (3.69)	8.69 (2.35)	4.21 (1.35)	-	5.12 (3.14)	6.68 (4.37)	6.66 (2.84)	10.87 (3.05)	12.31 (3.78)	11.62 (3.13)	4.58 (3.68)	5.19 (2.65)	8.98 (3.02)	10.10 (2.51)	**3.04 (2.17)**

Tabela 7 – G-mean for approaches compared for all experimented datasets. The results highlighted with bold represent the best accuracy in comparison with the other approaches. Values in brackets are the standard deviations. The last line represents the algorithm position in the Friedman ranking.

Dataset	**IGMM-CD**	**Dynse**	**GMM-VRD**	**OGMMF-VRD (Filter)**	**OGMMF-VRD (No Filter)**	**GLDD-DU (Accuracy)**	**GLDD-DU (Gmean)**	**LDD-DSDA**	**HAT-DDM**	**HAT-EDDM**	**HAT-ADWIN**	**AWE**	**LevBag**	**OzaAD**	**OzaAS**	**ARF**
Circles	0.609 (0.033)	0.741 (0.033)	0.787 (0.007)	0.813 (0.005)	0.804 (0.005)	0.806 (0.005)	0.798 (0.004)	0.828 (0.003)	0.6189 (0.005)	0.4926 (0.003)	0.5592 (0.013)	**0.8489 (0.001)**	0.8122 (0.001)	0.7373 (0.008)	0.7247 (0.001)	0.815 (0.003)
Sine1	0.536 (0.023)	0.603 (0.090)	0.817 (0.004)	0.827 (0.005)	0.824 (0.004)	0.825 (0.004)	0.823 (0.003)	0.803 (0.003)	0.5653 (0.025)	0.4975 (0.003)	0.5094 (0.005)	**0.8286 (0.002)**	0.8082 (0.001)	0.6584 (0.017)	0.5658 (0.002)	0.8137 (0.003)
Sine2	0.538 (0.024)	0.584 (0.073)	0.722 (0.006)	0.750 (0.004)	0.739 (0.004)	0.765 (0.003)	0.761 (0.005)	0.743 (0.002)	0.7044 (0.003)	0.4978 (0.004)	0.6585 (0.013)	0.7532 (0.001)	0.7319 (0.002)	0.5893 (0.016)	0.5525 (0.001)	**0.7886 (0.003)**
Virtual5	0.755 (0.003)	0.792 (0.004)	0.815 (0.003)	0.827 (0.006)	0.824 (0.003)	0.841 (0.004)	0.838 (0.004)	0.831 (0.002)	0.6941 (0.01)	0.3987 (0.002)	0.5127 (0.01)	**0.8504 (0.001)**	0.7987 (0.001)	0.7702 (0.005)	0.7368 (0.002)	0.838 (0.004)
Virtual9	0.804 (0.007)	0.805 (0.007)	0.845 (0.005)	0.858 (0.006)	0.854 (0.005)	0.869 (0.005)	0.868 (0.003)	0.852 (0.002)	0.7082 (0.011)	0.4104 (0.002)	0.492 (0.014)	**0.8697 (0.001)**	0.8104 (0.004)	0.7658 (0.014)	0.7267 (0.001)	0.8451 (0.003)
SEA	0.641 (0.008)	0.725 (0.004)	0.779 (0.008)	0.841 (0.004)	0.832 (0.007)	0.844 (0.005)	0.839 (0.005)	0.832 (0.003)	0.6975 (0.004)	0.5046 (0.003)	0.5233 (0.004)	**0.8534 (0.001)**	0.8295 (0.002)	0.8057 (0.003)	0.8049 (0.001)	0.8335 (0.004)
SEARec	0.641 (0.006)	0.747 (0.003)	0.777 (0.004)	0.841 (0.004)	0.833 (0.002)	0.846 (0.003)	0.845 (0.004)	0.837 (0.002)	0.7146 (0.025)	0.5014 (0.002)	0.5279 (0.015)	**0.8558 (0.001)**	0.8304 (0.001)	0.8106 (0.002)	0.8094 (0)	0.8291 (0.003)
NOAA	0.420 (0.001)	0.462 (0.002)	0.708 (0.009)	0.743 (0.004)	0.723 (0.006)	0.740 (0.006)	0.736 (0.007)	0.694 (0.005)	0.7072 (0.004)	0.6862 (0.001)	0.6989 (0.007)	0.7416 (0.002)	0.7098 (0.003)	0.6634 (0.011)	0.6869 (0.015)	**0.7656 (0.004)**
ELEC	0.783 (0.001)	0.651 (0.001)	0.685 (0.008)	0.745 (0.004)	0.755 (0.005)	0.756 (0.003)	0.754 (0.005)	0.735 (0.002)	0.6966 (0.013)	0.6752 (0.005)	0.6967 (0.006)	0.7112 (0.001)	0.7606 (0.003)	0.7059 (0.004)	0.6785 (0.001)	**0.7935 (0.003)**
PAKDD	0.604 (0.007)	0.660 (0.002)	0.702 (0.016)	0.773 (0.001)	0.785 (0.001)	0.780 (0.009)	0.700 (0.010)	0.395 (0.007)	0.8002 (0.001)	0.8024 (0)	0.8001 (0.001)	0.6205 (0.041)	0.7109 (0.005)	0.3951 (0.026)	0.5484 (0.037)	**0.7955 (0.001)**
GasSensor	0.009 (0)	0.246 (0.001)	0.3333 (0.079)	0.5785 (0.022)	0.589 (0.019)	0.374 (0.012)	0.367 (0.011)	0.823 (0.004)	0.3729 (0.008)	0.3608 (0.003)	0.3619 (0.001)	0.3326 (0.003)	**0.8464 (0.002)**	0.565 (0.003)	0.5552 (0.002)	0.8053 (0.007)
INSECT-Inc-Rec-Bal	0.3951 (0.014)	0.656 (0.001)	0.4941 (0.032)	0.6555 (0.004)	0.651 (0.004)	0.497 (0.008)	0.439 (0.005)	0.686 (0.002)	0.4636 (0.005)	0.4872 (0.001)	0.4694 (0)	0.5788 (0.005)	0.6789 (0.007)	0.5007 (0.007)	0.4851 (0.001)	**0.7228 (0.005)**
INSECT-Inc-Abt-Bal	0.3938 (0.013)	0.636 (0.001)	0.4861 (0.031)	0.6464 (0.005)	0.645 (0.006)	0.502 (0.010)	0.435 (0.006)	0.667 (0.002)	0.4843 (0.003)	0.508 (0)	0.4919 (0)	0.5613 (0.006)	0.6708 (0.005)	0.6011 (0.004)	0.5865 (0.001)	**0.7024 (0.004)**
INSECT-Grad-Bal	0.473 (0.003)	0.646 (0.001)	0.4876 (0.007)	0.5866 (0.005)	0.586 (0.010)	0.403 (0.015)	0.360 (0.013)	0.683 (0.004)	0.3484 (0.01)	0.3841 (0.001)	0.3631 (0.001)	0.6013 (0.006)	0.6921 (0.003)	0.6232 (0.032)	0.5441 (0.001)	**0.703 (0.017)**
Friedman Ranking	12.14 (3.38)	9.81 (3.69)	8.69 (2.35)	4.21 (1.35)	-	5.12 (3.14)	6.68 (4.37)	6.66 (2.84)	10.87 (3.05)	12.31 (3.78)	11.62 (3.13)	4.58 (3.68)	5.19 (2.65)	8.98 (3.02)	10.10 (2.51)	**3.04 (2.17)**

Tabela 8 – Time for approaches compared for all experimented datasets. The results highlighted with bold represent the best accuracy in comparison with the other approaches. Values in brackets are the standard deviations. The last line represents the algorithm position in the Friedman ranking.

Dataset	IGMM-CD	Dynse	GMM-VRD	OGMMF-VRD (Filter)	OGMMF-VRD (No Filter)	GLDD-DU (Accuracy)	GLDD-DU (Gmean)	LDD-DSDA	HAT-DDM	HAT-EDDM	HAT-ADWIN	AWE	LevBag	OzaAD	OzaAS	ARF
Circles	0.609 (0.033)	0.741 (0.033)	0.787 (0.007)	0.813 (0.005)	0.804 (0.005)	0.806 (0.005)	0.798 (0.004)	0.828 (0.003)	0.6189 (0.005)	0.4926 (0.003)	0.5592 (0.013)	**0.8489 (0.001)**	0.8122 (0.001)	0.7373 (0.008)	0.7247 (0.001)	0.815 (0.003)
Sine1	0.536 (0.023)	0.603 (0.090)	0.817 (0.004)	0.827 (0.005)	0.824 (0.004)	0.825 (0.004)	0.823 (0.003)	0.803 (0.003)	0.5653 (0.025)	0.4975 (0.003)	0.5094 (0.005)	**0.8286 (0.002)**	0.8082 (0.001)	0.6584 (0.017)	0.5658 (0.002)	0.8137 (0.003)
Sine2	0.538 (0.024)	0.584 (0.073)	0.722 (0.006)	0.750 (0.004)	0.739 (0.004)	0.765 (0.003)	0.761 (0.005)	0.743 (0.002)	0.7044 (0.003)	0.4978 (0.004)	0.6585 (0.013)	0.7532 (0.001)	0.7319 (0.002)	0.5893 (0.016)	0.5525 (0.001)	**0.7886 (0.003)**
Virtual5	0.755 (0.003)	0.792 (0.004)	0.815 (0.003)	0.827 (0.006)	0.824 (0.003)	0.841 (0.004)	0.838 (0.004)	0.831 (0.002)	0.6941 (0.01)	0.3987 (0.002)	0.5127 (0.01)	**0.8504 (0.001)**	0.7987 (0.001)	0.7702 (0.005)	0.7368 (0.002)	0.838 (0.004)
Virtual9	0.804 (0.007)	0.805 (0.007)	0.845 (0.005)	0.858 (0.006)	0.854 (0.005)	0.869 (0.005)	0.868 (0.003)	0.852 (0.002)	0.7082 (0.011)	0.4104 (0.002)	0.492 (0.014)	**0.8697 (0.001)**	0.8104 (0.004)	0.7658 (0.014)	0.7267 (0.001)	0.8451 (0.003)
SEA	0.641 (0.008)	0.725 (0.004)	0.779 (0.008)	0.841 (0.004)	0.832 (0.007)	0.844 (0.005)	0.839 (0.005)	0.832 (0.003)	0.6975 (0.004)	0.5046 (0.003)	0.5233 (0.004)	**0.8534 (0.001)**	0.8295 (0.002)	0.8057 (0.003)	0.8049 (0.001)	0.8335 (0.004)
SEARec	0.641 (0.006)	0.747 (0.003)	0.777 (0.004)	0.841 (0.004)	0.833 (0.002)	0.846 (0.003)	0.845 (0.004)	0.837 (0.002)	0.7146 (0.025)	0.5014 (0.002)	0.5279 (0.015)	**0.8558 (0.001)**	0.8304 (0.001)	0.8106 (0.002)	0.8094 (0)	0.8291 (0.003)
NOAA	0.420 (0.001)	0.462 (0.002)	0.708 (0.009)	0.743 (0.004)	0.723 (0.006)	0.740 (0.006)	0.736 (0.007)	0.694 (0.005)	0.7072 (0.004)	0.6862 (0.001)	0.6989 (0.007)	0.7416 (0.002)	0.7098 (0.003)	0.6634 (0.011)	0.6869 (0.015)	**0.7656 (0.004)**
ELEC	0.783 (0.001)	0.651 (0.001)	0.685 (0.008)	0.745 (0.004)	0.755 (0.005)	0.756 (0.003)	0.754 (0.005)	0.735 (0.002)	0.6966 (0.013)	0.6752 (0.005)	0.6967 (0.006)	0.7112 (0.001)	0.7606 (0.003)	0.7059 (0.004)	0.6785 (0.001)	**0.7935 (0.003)**
PAKDD	0.604 (0.007)	0.660 (0.002)	0.702 (0.016)	0.773 (0.001)	0.785 (0.001)	0.780 (0.009)	0.700 (0.010)	0.395 (0.007)	0.8002 (0.001)	0.8024 (0)	0.8001 (0.001)	0.6205 (0.041)	0.7109 (0.005)	0.3951 (0.026)	0.5484 (0.037)	**0.7955 (0.001)**
GasSensor	0.009 (0)	0.246 (0.001)	0.3333 (0.079)	0.5785 (0.022)	0.589 (0.019)	0.374 (0.012)	0.367 (0.011)	0.823 (0.004)	0.3729 (0.008)	0.3608 (0.003)	0.3619 (0.001)	0.3326 (0.003)	**0.8464 (0.002)**	0.565 (0.003)	0.5552 (0.002)	0.8053 (0.007)
INSECT-Inc-Rec-Bal	0.3951 (0.014)	0.656 (0.001)	0.4941 (0.032)	0.6555 (0.004)	0.651 (0.004)	0.497 (0.008)	0.439 (0.005)	0.686 (0.002)	0.4636 (0.005)	0.4872 (0.001)	0.4694 (0)	0.5788 (0.005)	0.6789 (0.007)	0.5007 (0.007)	0.4851 (0.001)	**0.7228 (0.005)**
INSECT-Inc-Abt-Bal	0.3938 (0.013)	0.636 (0.001)	0.4861 (0.031)	0.6464 (0.005)	0.645 (0.006)	0.502 (0.010)	0.435 (0.006)	0.667 (0.002)	0.4843 (0.003)	0.508 (0)	0.4919 (0)	0.5613 (0.006)	0.6708 (0.005)	0.6011 (0.004)	0.5865 (0.001)	**0.7024 (0.004)**
INSECT-Grad-Bal	0.473 (0.003)	0.646 (0.001)	0.4876 (0.007)	0.5866 (0.005)	0.586 (0.010)	0.403 (0.015)	0.360 (0.013)	0.683 (0.004)	0.3484 (0.01)	0.3841 (0.001)	0.3631 (0.001)	0.6013 (0.006)	0.6921 (0.003)	0.6232 (0.032)	0.5441 (0.001)	**0.703 (0.017)**
Friedman Ranking	12.14 (3.38)	9.81 (3.69)	8.69 (2.35)	4.21 (1.35)	-	5.12 (3.14)	6.68 (4.37)	6.66 (2.84)	10.87 (3.05)	12.31 (3.78)	11.62 (3.13)	4.58 (3.68)	5.19 (2.65)	8.98 (3.02)	10.10 (2.51)	**3.04 (2.17)**

6.3.1 Approaches With Separate Mechanisms to Deal with Virtual and Real Drifts

Tbls. 6, and 7 presents the Average Accuracy and G-mean for the compared approaches using both synthetic and real-world datasets. To attest the statistical difference of the results, we present in Fig. 23 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. 24(a)), and G-mean (Fig. 24(b)), OGMMF-VRD is significantly better than all these approaches (GMM-VRD, IGMM-CD, and Dynse). For runtime (Fig. 24(c)), OGMMF-VRD is only better than Dynse. To provide a more detailed understanding of these results, Figs. 24, and 25 presents plots of AOT for GMM-VRD, IGMM-CD, and Dynse.

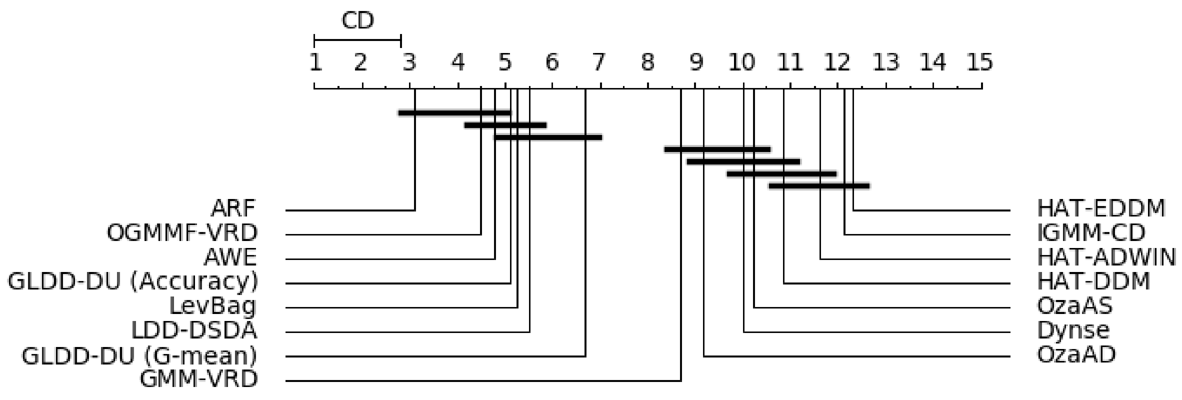
Looking at the dataset which has abrupt virtual drifts (Virtual 9 in Fig. 25(b)), we can see that IGMM-CD and GMM-VRD had good AOT. This is because these datasets have similarities between their concepts, and despite this approach potentially generating too many Gaussians over time can handle this type 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. 25(c)), which has virtual/real drifts, and Sine 2 (Fig. 25(e)), 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 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.

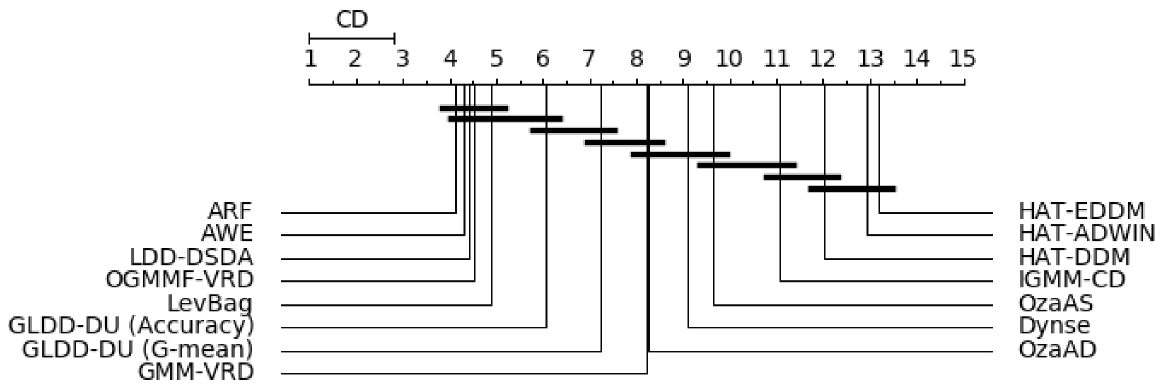
Now, we look at the real-world datasets: NOAA (Fig. 26(a)), and PAKDD (Fig. 26(c)). For the NOAA, it is observed that Dynse and IGMM-CD sometimes decline their accuracy, which demonstrates that these methods may not be robust to different types of drift. For PAKDD, GMM-VRD and OGMMF-VRD obtained very 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.

Therefore, answering RQ2, GMM-VRD, our first proposal to deal with virtual and real drifts at the same time, managed to perform well on most datasets, showing its stability while other methods sometimes perform well and sometimes poorly. It demonstrates that harnessing GMM benefits with different mechanisms for dealing with the concept drift makes the model more robust performance.

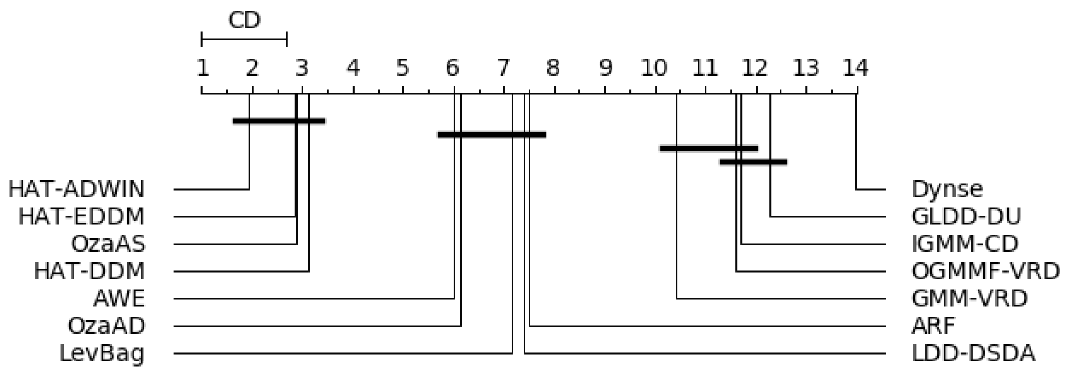
Figura 23 – Friedman ranking for the results obtained for all datasets used in Tbl. 3. Friedman's p-values were 1.02E-212, 0.00E+00 and 0.00E+00, respectively, indicating rejection of the null hypothesis at the level of significance of $\alpha = 0.05$. Any pair of approaches whose distance between them is larger than CD is considered to be different according to the Nemenyi posthoc tests.



(a) Accuracy



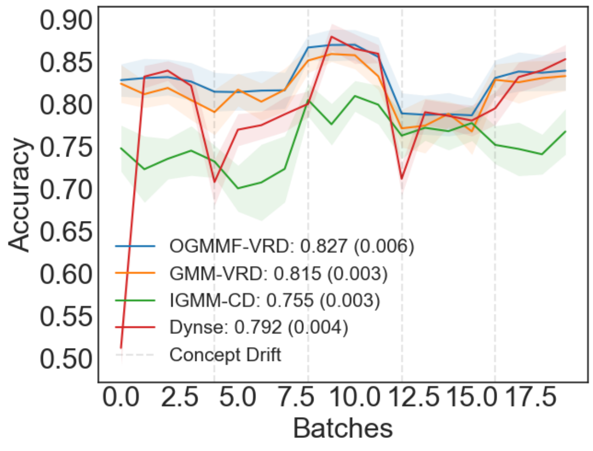
(b) G-mean



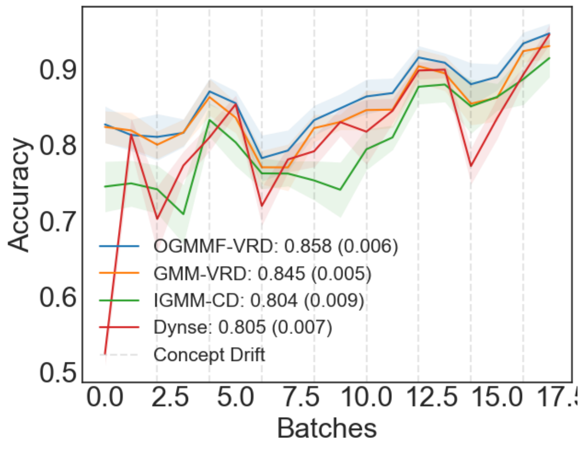
(c) Runtime

Source: Author.

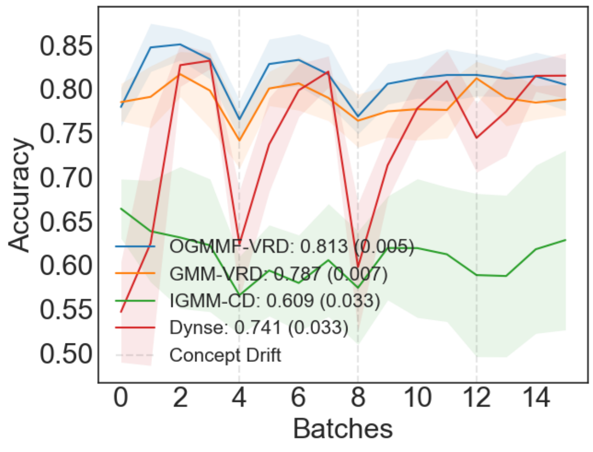
Figura 24 – Mean of accuracy over time for methods that handle virtual and real drifts on synthetic datasets. The standard deviation is represented by shadow lines of the same color. Each point represents the accuracy for a batch observations, where 500 was used.



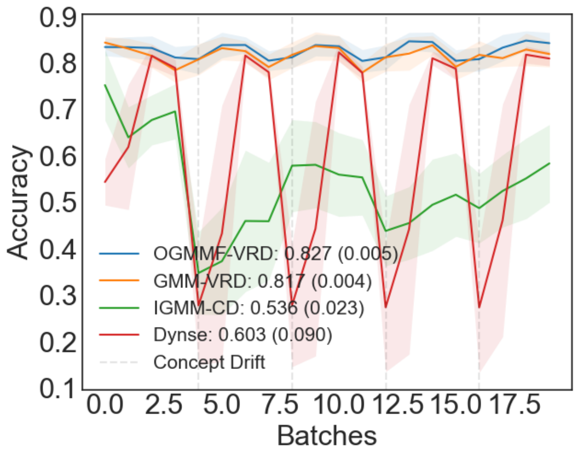
(a) Virtual 5 drifts



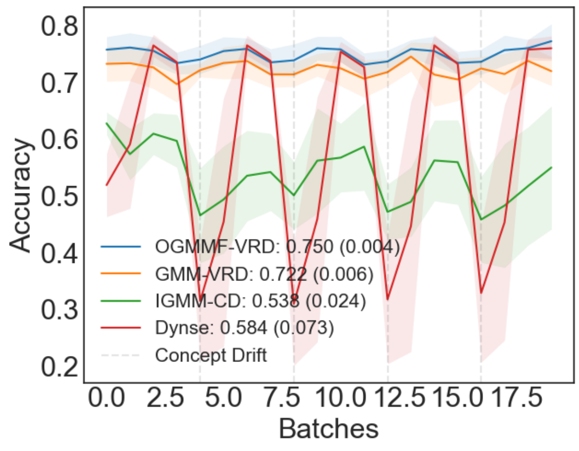
(b) Virtual 9 drifts



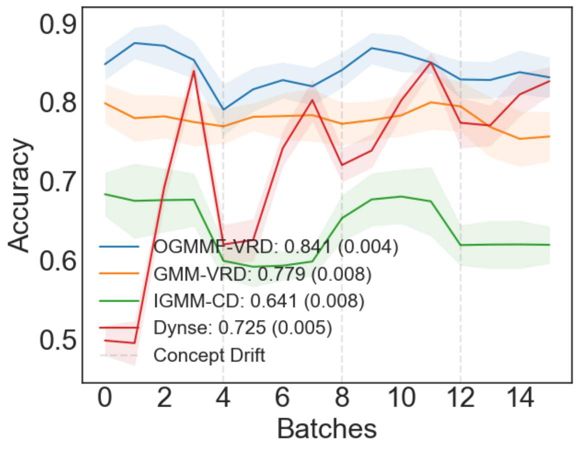
(c) Circles



(d) Sine1



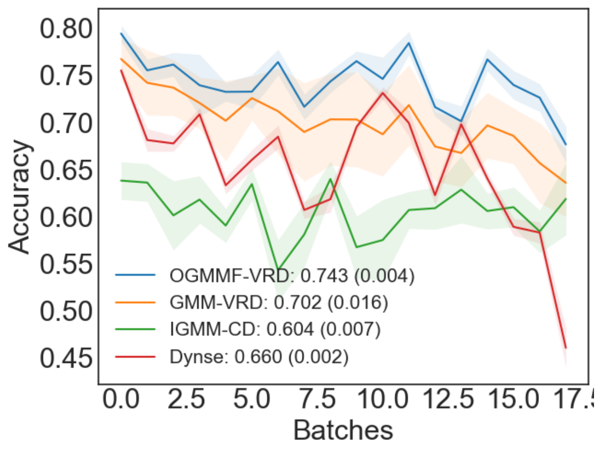
(e) Sine2



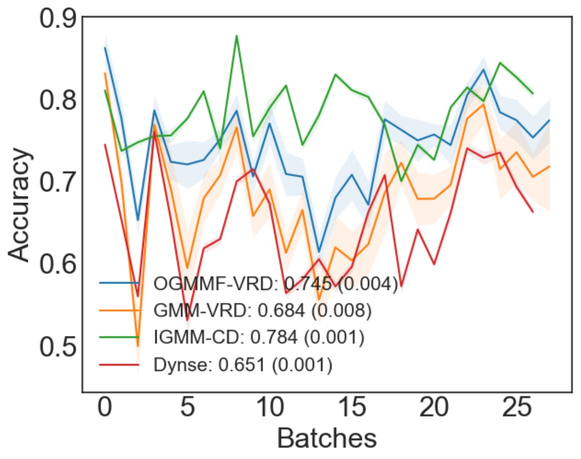
(f) SEA

Source: Author.

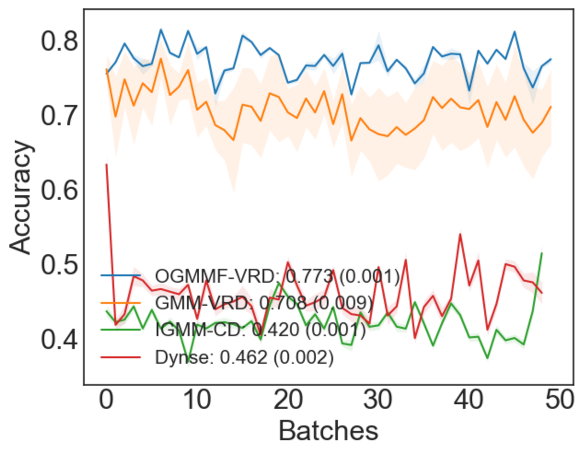
Figura 25 – Mean of accuracy over time for methods that handle virtual and real drifts on each dataset. The standard deviation is represented by shadow lines of the same color. Each point represents the accuracy for a batch observations, where 1000 was used.



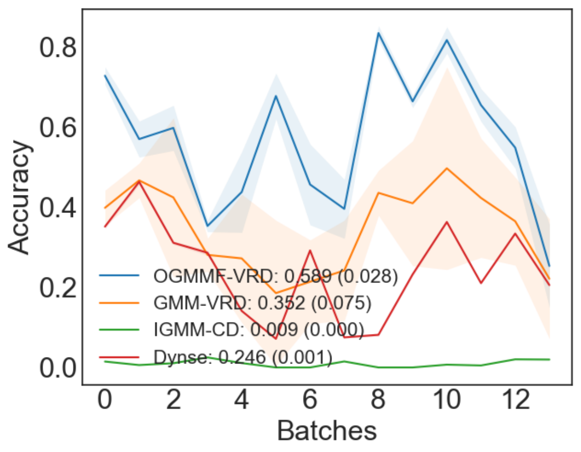
(a) NOAA



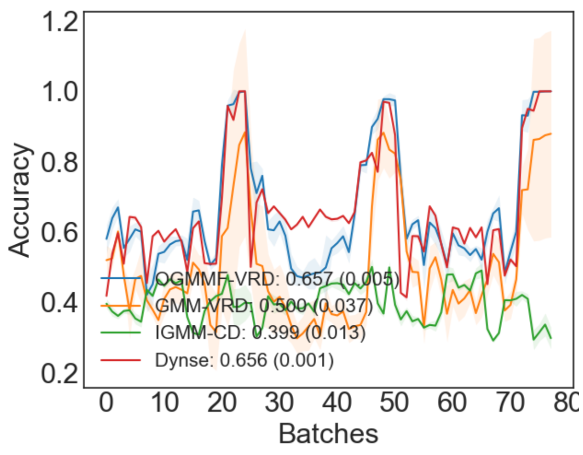
(b) ELEC



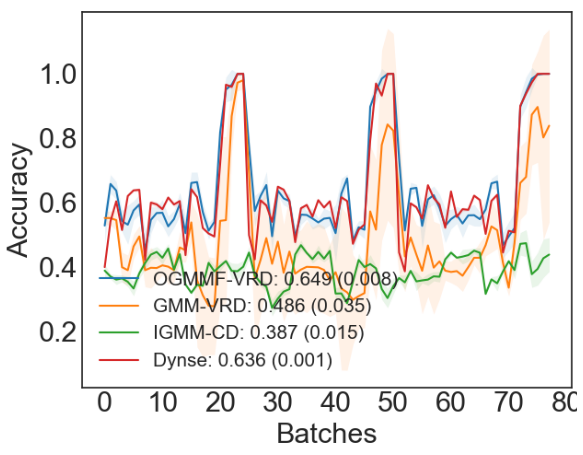
(c) PAKDD



(d) GasSensor



(e) INS-Inc-Reo



(f) INS-Inc-Abt

Source: Author.

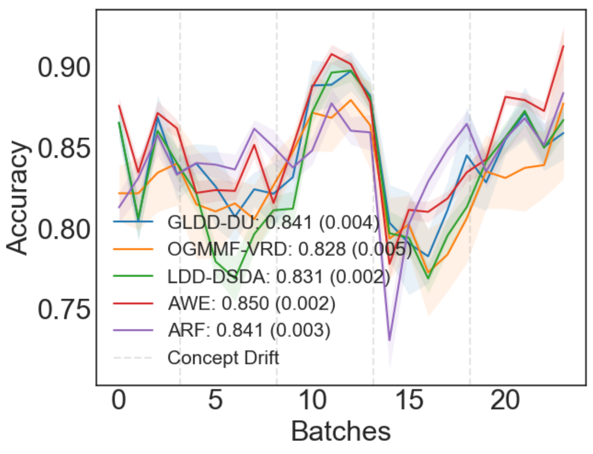
6.3.2 Other Approaches From the Concept Drift Literature

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. 24(a) and 24(b)), OGMMF-VRD was significantly better than all methods, except for AWE, ARF and GLDD-DU. As shown in the last line of Tbl. 6, these approaches reached the following rankings (and corresponding ranking standard deviations) for accuracy: OGMMF-VRD: 4.21 (1.35), ARF: 3.04 (2.17), AWE: 4.58 (3.68), and GLDD-DU: 5.12 (3.14). For G-mean, the rankings are: OGMMF-VRD: 4.30 (2.04), ARF: 4.06 (3.11), AWE: 4.11 (3.37), and GLDD-DU: 6.06 (3.67). OGMMF-VRD achieved rankings with smaller standard deviations (i.e., more stable rankings) than ARF, AWE and GLDD-DU across datasets. GLDD-DU obtained competitive stability in terms of Accuracy, but not in terms of G-mean. 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 and GLDD-DU achieved good accuracies but performed very poorly on datasets such as GasSensor (Tbl 6). Given the more stable rankings, OGMMF-VRD is more reliable for adoption in practice. Another point observed is that incremental learning methods (HAT) combined with detectors performed poorly, which indicates that only incremental learning with a reset is not enough to tackle all types of drifts effectively.

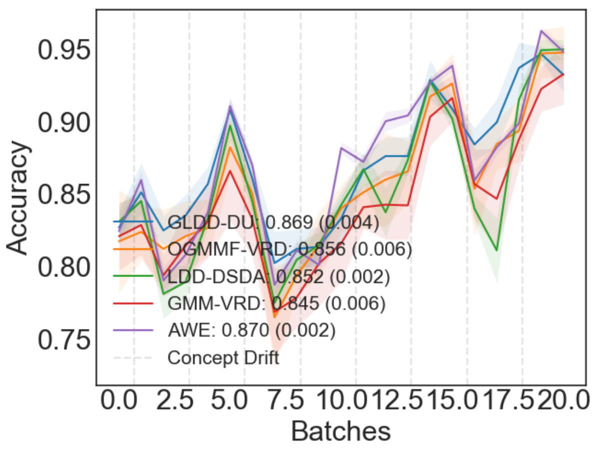
In terms of runtime (Fig. 24(c)), ours approaches OGMMF-VRD and GLDD-DU are costlier. This is partly because our code is not optimized like that of scikit-multiflow algorithms, though it may also be influenced by their mechanisms to deal with different types of drift.

The predictive performance of ensemble methods can be summarized in two main points based on the results obtained for the Sine 2 dataset (Fig. 27(e)). 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 and GLDD-DU 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.

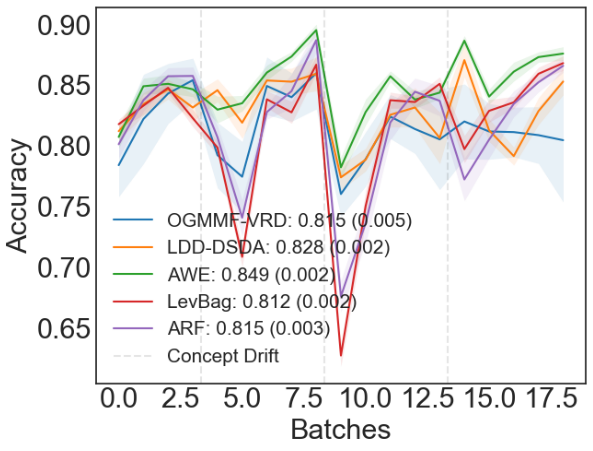
Figura 26 – Mean of accuracy over time for the five best methods in synthetic datasets. The standard deviation is represented by shadow lines of the same color. Each point represents the accuracy for a batch observations, where 500 was used.



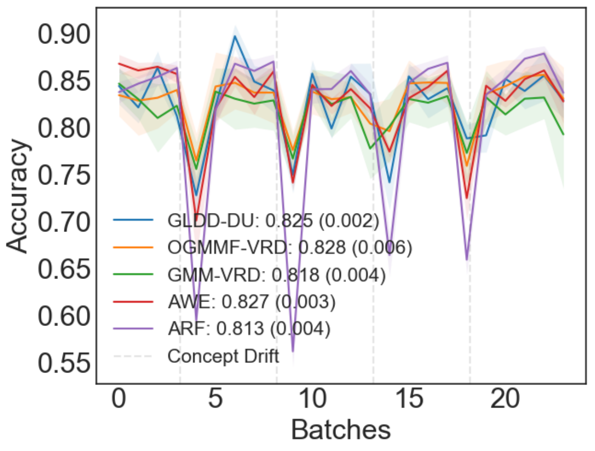
(a) Virtual 5 drifts



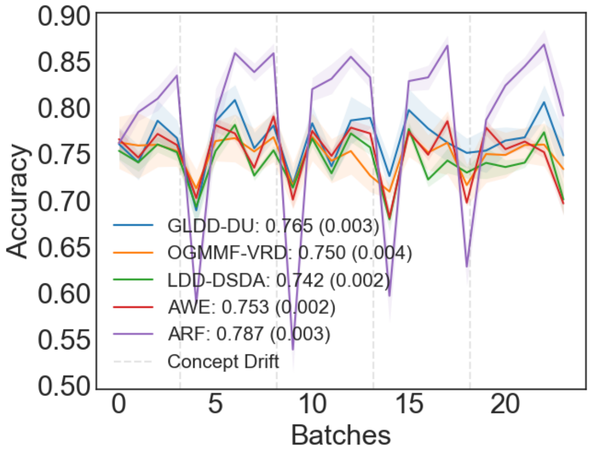
(b) Virtual 9 drifts



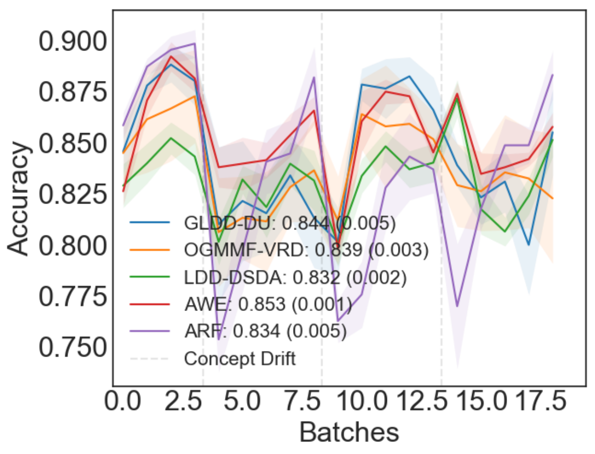
(c) Circles



(d) Sine1



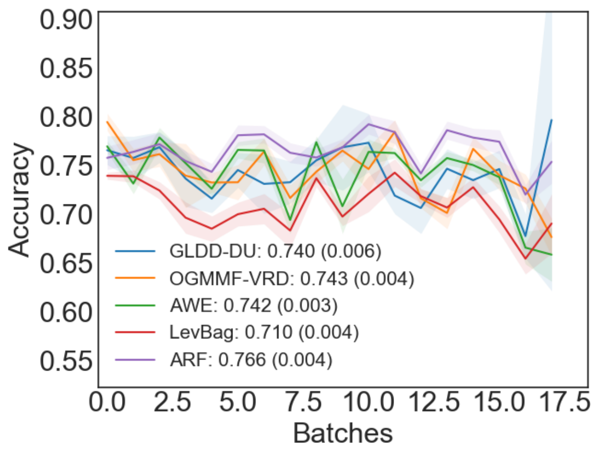
(e) Sine2



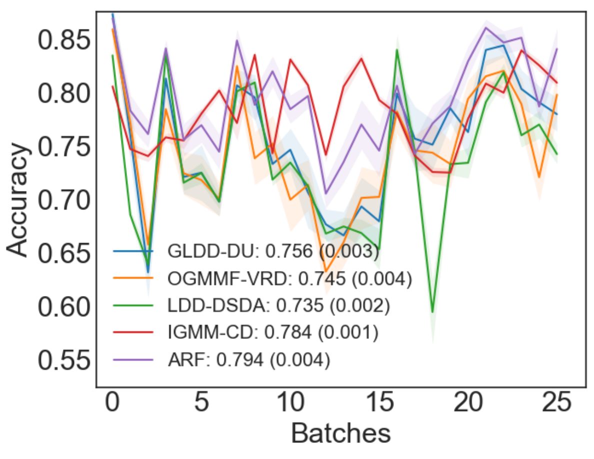
(f) SEA

Source: Author.

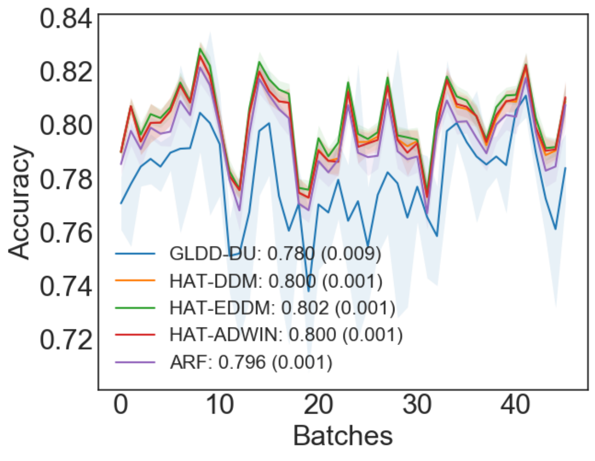
Figura 27 – Mean of accuracy over time for the five best methods in real datasets. The standard deviation is represented by shadow lines of the same color. Each point represents the accuracy for a batch observations, where 500 was used for synthetic datasets, and 1000 for real datasets.



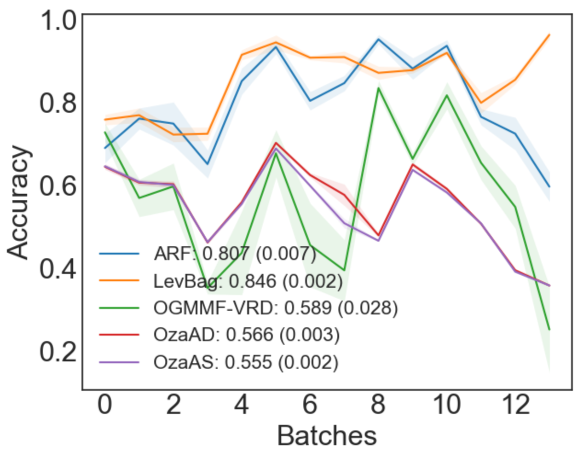
(a) NOAA



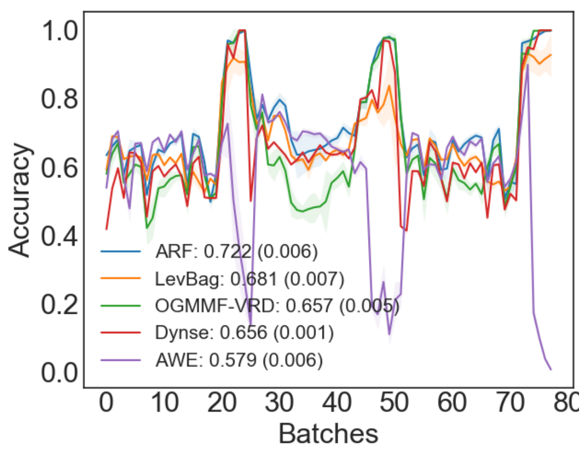
(b) ELEC



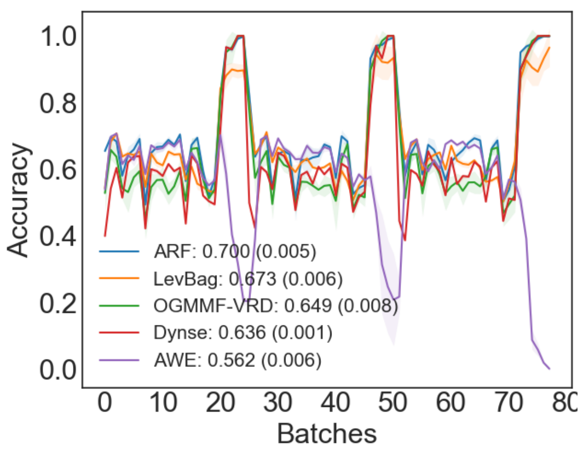
(c) PAKDD



(d) GasSensor



(e) INS-Inc-Reo



(f) INS-Inc-Abt

Source: Author.

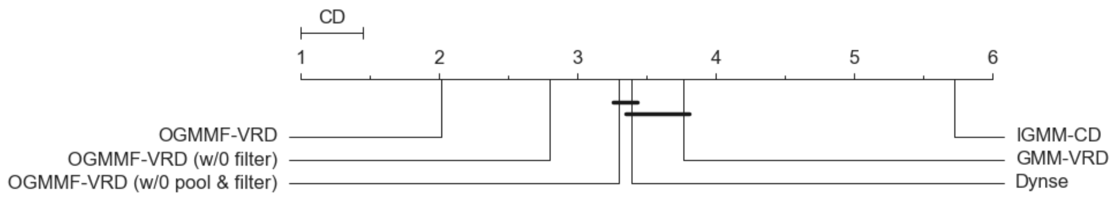
6.4 IMPACT OF NOISE ON CLASSIFIER PERFORMANCE

This experiment aims to determine how well the OGMMF-VRD answers RQ3 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. OGMMF-VRD uses a filter in a sliding window to differentiate noise-free observations to noisy observations and recovers GMMs from a pool to increase its performance. Both components were evaluated on the synthetic datasets discussed in Tbl 3 varying their noise level to [5%, 10%, 15%, 20%]. Noise is an observation chosen at random and having its class changed.

In this experiment, we compared with GMM-VRD, IGMM-CD, and Dynse, which are approaches with separate mechanisms to deal virtual and real drifts. GLDD-DU is not included because it has the same mechanism as OGMMF-VRD to deal with noisy observations. The parameters used are presented in Tbl. 5, 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, Tbl. 9 presents the approaches performances for accuracy, and Figs. 28, and 29 summarize the results. We observe in Fig. 29 that, regardless of the noise level, both mechanisms helped to improve the performance of OGMMF-VRD, being a significant improvement as shown in Fig. 28.

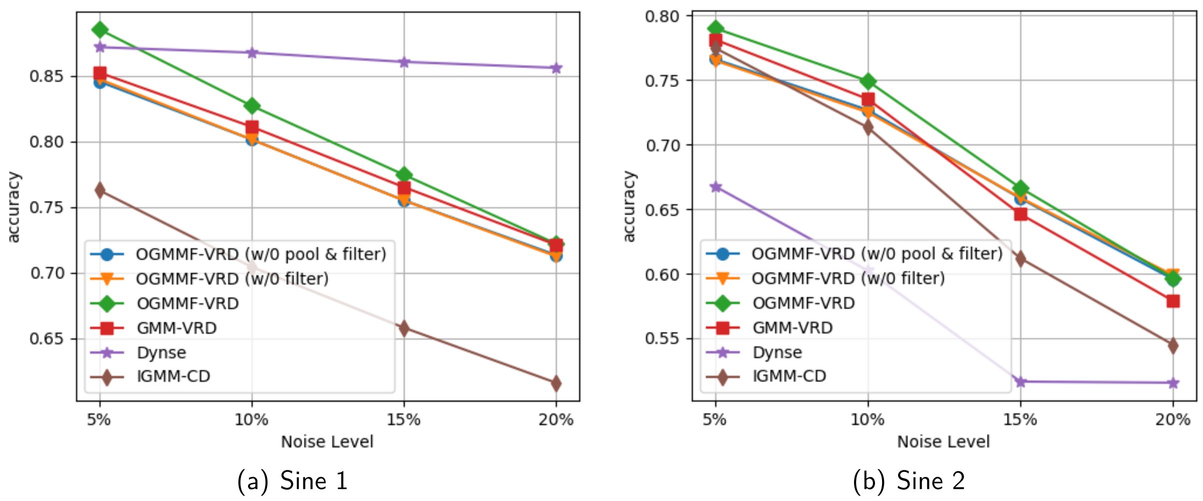
Figura 28 – Friedman ranking for the accuracy obtained in all synthetic datasets with all different noise levels. Friedman's p-value was 9.73E-25 and its ranking is shown from left (best) to right (worst). Any pair of approaches whose distance between them is larger than CD is considered to be different.



Source: Author.

As show in Tbl. 9, the filter becomes more effective when using batches with larger sizes because more observations help recognize when noise observations appear. 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 $m = 50$ 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.

Figura 29 – 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.



(a) Sine 1 (b) Sine 2

Source: Author.

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. 28. 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.

Tabela 9 – Average accuracy for synthetic datasets with different noise levels. w/o means the system without the mechanism. m is the chunk size used in the experiments. The results highlighted with bold represent the best accuracy in comparison with the other approaches.

Datasets	**OGMMF-VRD (w/o pool) m = 200**	**OGMMF-VRD (w/o filter and pool) m = 200**	**OGMMF-VRD m = 200**	**OGMMF-VRD (w/o filter) m = 50**	**OGMMF-VRD m = 50**	**GMM-VRD**	**Dynse**	**IGMM-CD**
Circles 5%	0.871 (0.003)	0.872 (0.003)	**0.885 (0.002)**	0.8657 (0.004)	0.8652 (0.003)	0.8583 (0.004)	0.8227 (0.013)	0.7923 (0.003)
Circles 10%	0.828 (0.003)	0.826 (0.003)	**0.828 (0.005)**	0.814 (0.004)	0.8143 (0.003)	0.798 (0.004)	0.78 (0.011)	0.7291 (0.002)
Circles 15%	**0.727 (0.005)**	0.725 (0.003)	0.725 (0.004)	0.7043 (0.004)	0.703 (0.004)	0.6854 (0.004)	0.6944 (0.014)	0.6276 (0.003)
Circles 20%	0.628 (0.005)	**0.630 (0.003)**	0.624 (0.005)	0.611 (0.004)	0.6118 (0.003)	0.5991 (0.006)	0.6124 (0.009)	0.5619 (0.003)
Sine1 5%	0.846 (0.004)	0.848 (0.005)	0.885 (0.006)	**0.8902 (0.003)**	0.889 (0.005)	0.8519 (0.002)	0.7791 (0.001)	0.7628 (0.003)
Sine1 10%	0.802 (0.003)	0.802 (0.002)	0.827 (0.005)	0.8295 (0.004)	**0.8313 (0.004)**	0.8121 (0.003)	0.7445 (0.001)	0.7042 (0.001)
Sine1 15%	0.755 (0.003)	0.755 (0.003)	0.775 (0.006)	0.8295 (0.004)	**0.8313 (0.004)**	0.8121 (0.003)	0.7445 (0.001)	0.7042 (0.001)
Sine1 20%	0.713 (0.001)	0.712 (0.003)	**0.722 (0.004)**	0.715 (0.004)	0.7157 (0.003)	0.7207 (0.003)	0.6738 (0.001)	0.6156 (0.002)
Sine2 5%	0.766 (0.005)	0.765 (0.004)	**0.790 (0.008)**	0.7888 (0.003)	0.7888 (0.003)	0.7821 (0.003)	0.4781 (0.026)	0.7741 (0.002)
Sine2 10%	0.727 (0.005)	0.725 (0.005)	**0.749 (0.003)**	0.7476 (0.003)	0.7475 (0.003)	0.7349 (0.003)	0.4819 (0.024)	0.7133 (0.002)
Sine2 15%	0.658 (0.004)	0.659 (0.003)	**0.666 (0.004)**	0.6601 (0.004)	0.658 (0.004)	0.6482 (0.003)	0.5602 (0.093)	0.612 (0.002)
Sine2 20%	0.596 (0.002)	**0.599 (0.004)**	0.597 (0.007)	0.587 (0.002)	0.5856 (0.003)	0.5788 (0.004)	0.6024 (0.002)	0.5449 (0.003)
Virtual5 5%	0.799 (0.003)	**0.809 (0.001)**	0.809 (0.003)	0.8014 (0.003)	0.8018 (0.004)	0.8076 (0.002)	0.7575 (0.001)	0.7176 (0.001)
Virtual5 10%	0.725 (0.005)	**0.738 (0.002)**	0.738 (0.003)	0.724 (0.004)	0.7243 (0.003)	0.7309 (0.006)	0.6894 (0.001)	0.6236 (0.001)
Virtual5 15%	0.622 (0.004)	0.647 (0.003)	**0.648 (0.003)**	0.6257 (0.003)	0.6226 (0.003)	0.6349 (0.006)	0.5992 (0.001)	0.5228 (0.002)
Virtual5 20%	0.521 (0.004)	0.544 (0.001)	**0.545 (0.004)**	0.5167 (0.004)	0.5189 (0.004)	0.5196 (0.005)	0.4904 (0.001)	0.4357 (0.001)
Virtual9 5%	0.825 (0.004)	0.831 (0.003)	**0.831 (0.002)**	0.806 (0.004)	0.8042 (0.003)	0.7879 (0.006)	0.7641 (0.026)	0.7568 (0.002)
Virtual9 10%	0.739 (0.005)	**0.753 (0.001)**	0.753 (0.003)	0.7143 (0.003)	0.7153 (0.004)	0.6843 (0.007)	0.6849 (0.022)	0.6413 (0.002)
Virtual9 15%	0.739 (0.004)	0.752 (0.003)	**0.753 (0.003)**	0.7145 (0.004)	0.7177 (0.004)	0.6836 (0.008)	0.6958 (0.004)	0.6449 (0.002)
Virtual9 20%	0.599 (0.006)	0.619 (0.004)	**0.620 (0.004)**	0.572 (0.004)	0.5752 (0.003)	0.5141 (0.012)	0.5689 (0.002)	0.5108 (0.001)
SEA 5%	0.900 (0.004)	0.899 (0.004)	**0.913 (0.003)**	0.9118 (0.002)	0.9133 (0.002)	0.9052 (0.002)	0.8718 (0.001)	0.6476 (0.001)
SEA 10%	0.854 (0.004)	0.853 (0.003)	**0.856 (0.005)**	0.8576 (0.002)	0.8555 (0.002)	0.8487 (0.005)	0.8221 (0.001)	0.6103 (0.001)
SEA 15%	0.801 (0.003)	**0.804 (0.003)**	0.800 (0.004)	0.7992 (0.004)	0.798 (0.003)	0.7911 (0.003)	0.7717 (0.001)	0.5885 (0.001)
SEA 20%	0.750 (0.003)	**0.752 (0.003)**	0.747 (0.002)	0.7474 (0.002)	0.7458 (0.004)	0.7475 (0.007)	0.7275 (0.001)	0.5687 (0.001)
SEARec 5%	0.900 (0.002)	0.899 (0.002)	**0.912 (0.001)**	0.9128 (0.001)	0.9115 (0.002)	0.9022 (0.004)	0.8671 (0)	0.635 (0.001)
SEARec 10%	0.849 (0.002)	0.850 (0.002)	**0.854 (0.002)**	0.8541 (0.001)	0.8519 (0.002)	0.8307 (0.005)	0.8197 (0.001)	0.6066 (0.002)
SEARec 15%	**0.802 (0.002)**	0.800 (0.002)	0.795 (0.004)	0.7943 (0.002)	0.7924 (0.003)	0.7766 (0.004)	0.7733 (0.001)	0.5859 (0.001)
SEARec 20%	**0.749 (0.002)**	0.747 (0.003)	0.743 (0.002)	0.7399 (0.002)	0.7405 (0.002)	0.7224 (0.003)	0.7289 (0.001)	0.5611 (0.001)

6.5 IMPACT OF HANDLING VIRTUAL AND REAL DRIFTS SIMULTANEOUSLY

This experiment analyzes the OGMMF-VRD mechanisms and aims to complement the answer to RQ3 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 OGMMF-VRD can harness past knowledge to accelerate adaptation to both virtual and real drifts, answering RQ4. Only the synthetic datasets were used for this analysis, because they enable a better understanding of the behavior of the OGMMF-VRD in relation to each type of drift, unlike the real-world datasets. This experiment was made considering the parameters in Tbl. 5 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. 31(a)), 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 drift 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.

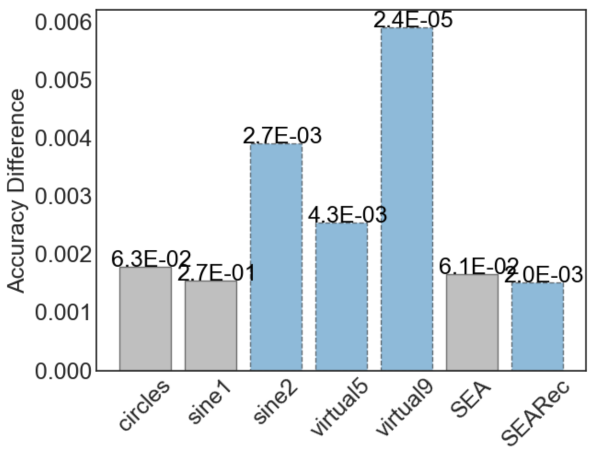
An interesting point to note is the gain from the Sine 2 dataset, which has abrupt real drifts, as well as the Virtual 9 which has abrupt virtual drifts. This indicates that besides having a good improvement in virtual datasets, other types of drift can sometimes also benefit from this mechanism.

To visibly analyzing these gains we present in the Fig. 31(b) 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 to create Gaussians can quickly prevent the system from losing performance. This complements the discussions of RQ3 in that the inclusion of a mechanism to deal with low severity drifts can help improving predictive performance in the presence of virtual drifts. These discussions confirm the part of the RQ3 which says that using (i) a noise filter, (ii) an *on-line* learning in all observations and (iii) creating Gaussians over time, helps the system handle non-severe drifts appropriately.

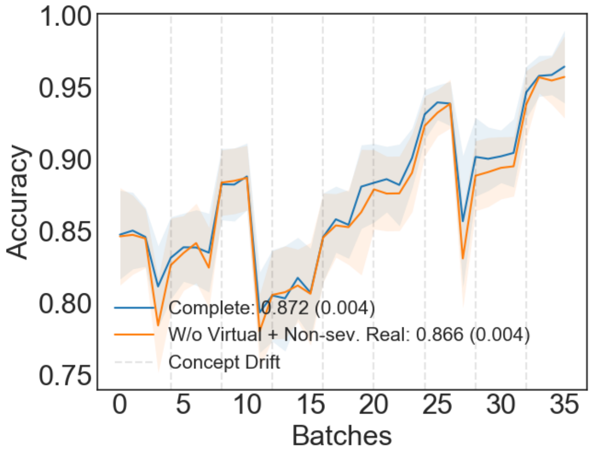
By analyzing the results for the severe real drift adaptation mechanism (Fig. 31(c)), we observe that this mechanism statistically significantly improved the performances in all cases, showing that this mechanism can be useful even for virtual drifts. This may be because CDTs can enable the system to 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. we observe that the improvements were larger for the more severe drifts and statistically significant for α=0.05 in all cases, thus, this mechanism is really adequate for severe drifts. We realize that monitoring the GMM's error significantly helps its performance as it is always reset in the presence of severe concept drifts, as we hypothesize for RQ4. In the best improvement that was for Virtual 9 (Fig. 31(d)), we note that not using a CDT can be quite derogatory for model performance since the AOTs compared are very far apart. This complements RQ3 to show that even virtual drifts can also benefit from this mechanism. This confirms the discussions of RQ1 that if real drifts is not addressed the system will have a great drop in performance.

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. 31(e)), 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. 31(f)), 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 RQ4, which states that harnessing the knowledge of past GMM's can accelerate the adaptation in both virtual and real drifts.

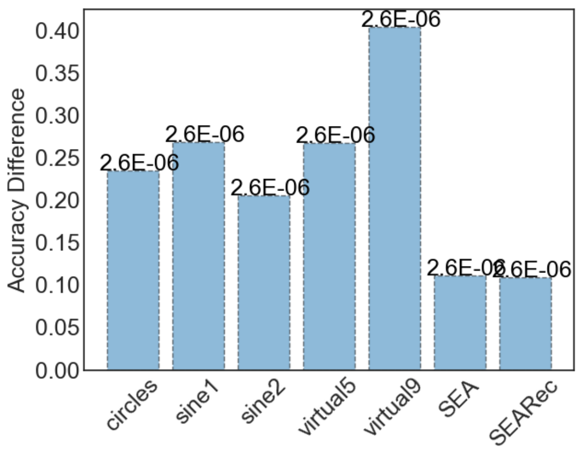
Figura 30 – Accuracy improvements obtained by each of OGMM-VRD's mechanisms. Each bar represents the subtraction of the average of the complete system from the average of the system without a given mechanism (e.g. w/o non-severe drift adaptation). The number above each bar represents the p-value obtained by Wilcoxon test over the comparison pair to pair. Bars with p-values smaller than $\alpha = 0.05$ are colored with blue with dashed lines. 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.



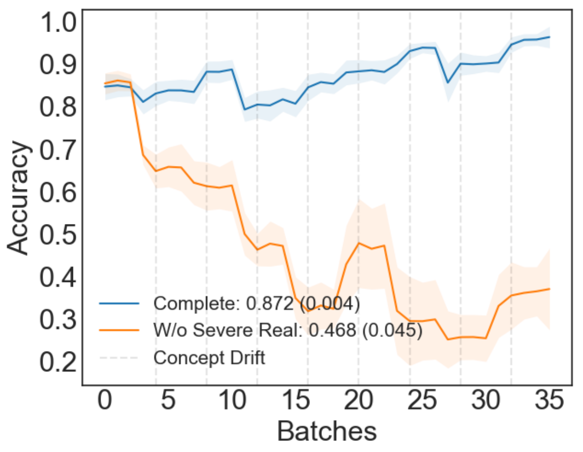
(a) Bar: Virtual + Ns. Real



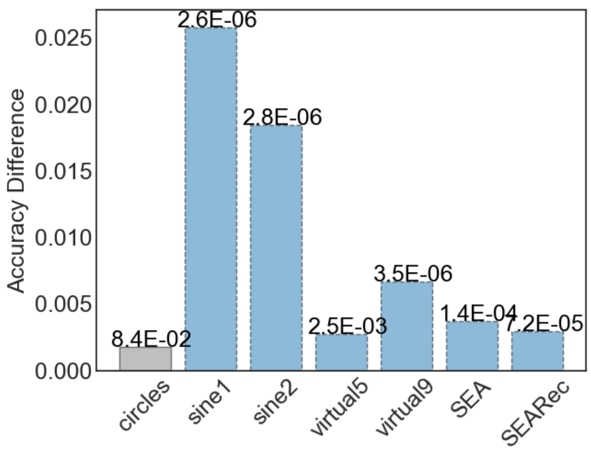
(b) Bar: Virtual + Ns. Real



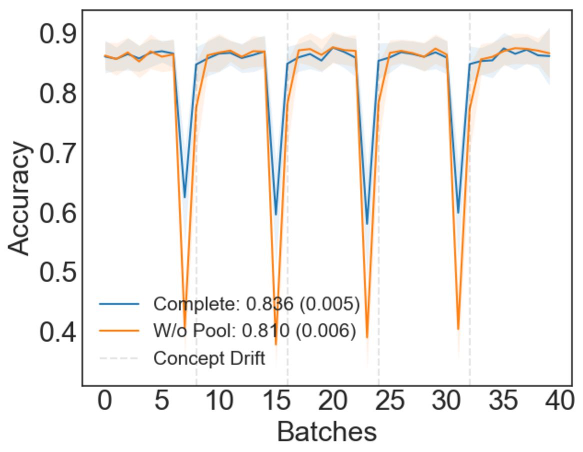
(c) Bar: Severe Real



(d) Best: Severe Real



(e) Bar: Pool



(f) Best: Pool

Source: Author.

6.6 IMPACT OF DRIFT UNDERSTANDING

This experiment analyzes the GLDD-DU mechanisms and aims to answer RQ5 and RQ6. RQ5 seeks to know how we could stimulate the generation of knowledge to help adapt to new concepts using GMM. RQ6 seeks to know how to update only a portion of the learned decision boundaries that were affected by the concept drift. In GLDD-DU, several different GMMs are generated during the training phase to increase the system's adaptability when dealing with new concepts. To update only the portion of the GLDD-DU affected by the drift, we use a drift detector for each Gaussian of the GMM, so when degradation is reported, we replace the bad Gaussian with a better performing one from the pool.

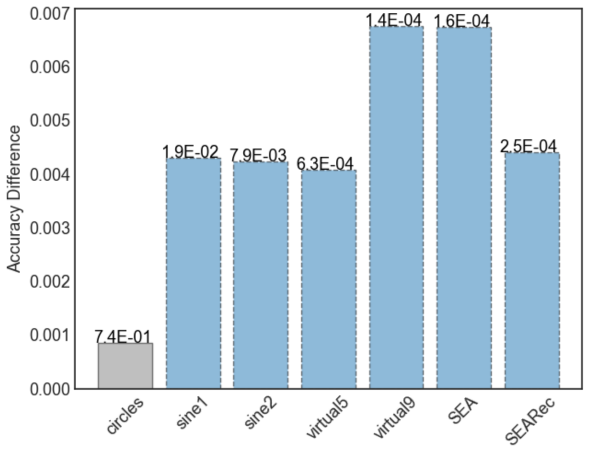
In this experiment, we investigate these mechanisms considering only the synthetic datasets. Synthetic datasets allow us to visualize and understand if the proposed mechanisms behave as expected in the presence of known drifts.

Thus, in Fig. 31, we made two types of comparisons, GLDD-DU with and without (i) Pool Training (Section 5.3.4 (RQ5)), and (ii) Gaussian Reuse (Section 5.3.5 (RQ6)). Pool Training means a comparison of (a) GLDD-DU using a single GMM, against (b) GLDD-DU using the pool of GMMs and choosing the best one. Gaussian Reuse means the comparison of (a) GLDD-DU choosing the best GMM from the pool, but without reusing Gaussians when local drifts happen, against (b) GLDD-DU choosing the best GMM from the pool and reusing Gaussians when local drifts happen. In this experiment, we adopted GLDD-DU with the parameter $m = 200$ to have a more robust model and better visualize the effects of the different mechanisms.

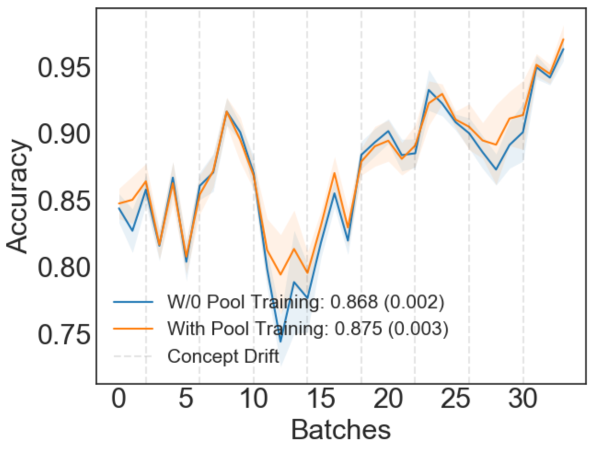
Regarding RQ5, we see in Fig. 32(a) that Pool Training has statistically improved the system results in 6 of the 7 synthetic data sets. This indicates that the choice of the best GMM model significantly improves the system predictive performance. Besides that, it shows that generating different subsamples of the training set using kDN is a good alternative to generate models with different capabilities to adapt to the data from the new concept. Observing Fig. 32(b), which shows the AOT for the best case (Virtual9), one can notice that during the periods of the same concept (between vertical dashed lines), there was a significant improvement in the performance of the system.

Regarding RQ6, we see in Fig. 32(c) that Gaussians Reuse has statistically improved the system results in 4 of the 7 synthetic datasets. We note that the datasets that have non-severe drifts were the most favored (Virtual5, SEA, and SEARec). This strategy to do local updates matches with these kind of drifts (non-severe) since they do not change the characteristics space entirely. Observing Fig. 32(d), which shows the AOT for the best case (SEARec), one can observe that there are several peaks of improvement over time. This indicates that updating local boundaries whenever they are performing poorly significantly improves results. This can also be seen in more detail in Fig. 32. Note that when drift happens not all Gaussians are affected at once, only a few. This shows that adaptation becomes faster for not having to erase all system's decision boundaries.

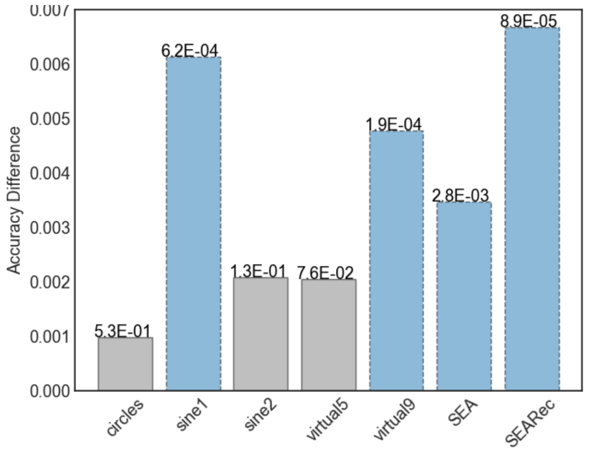
Figura 31 – Accuracy improvements obtained by each of GLDD-DU's mechanisms. Each bar represents the subtraction of the average of the complete system from the average of the system without a given mechanism (e.g. w/o Pool Training mechanism). The number above each bar represents the p-value obtained by Wilcoxon test over the comparison pair to pair. Bars with p-values smaller than $\alpha = 0.05$ are colored with blue with dashed lines. The AOT plots represent the dataset with the best improvement in each bar plot. The blue line represents the complete GLDD-DU system and the orange line represents the system without the respective mechanism.



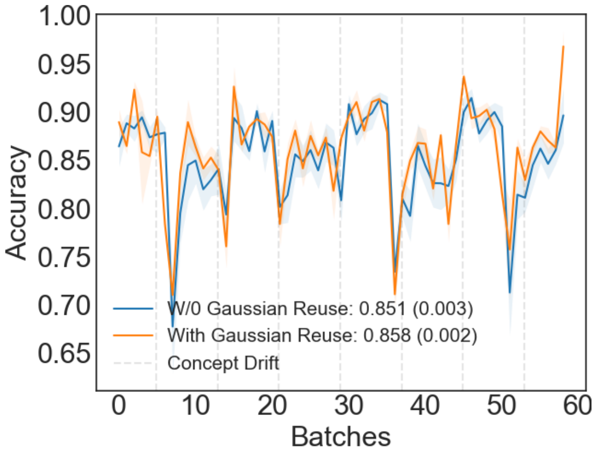
(a) Bar: Pool Training



(b) AOT: Virtual9



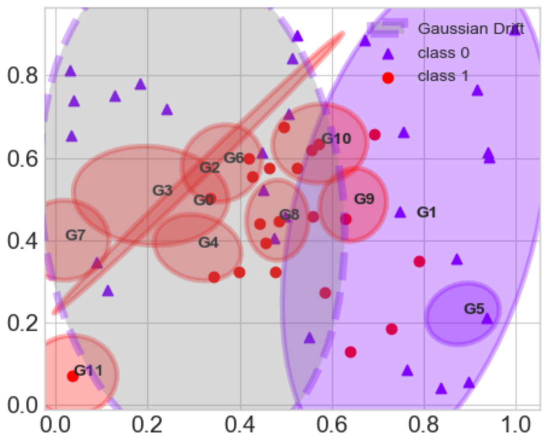
(c) Bar: Gaussians Reuse



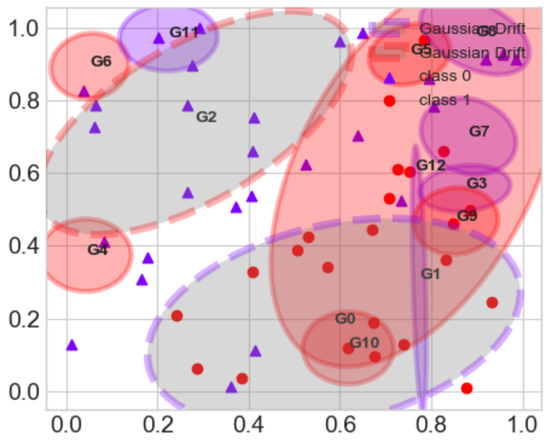
(d) AOT: SEARec

Source: Author.

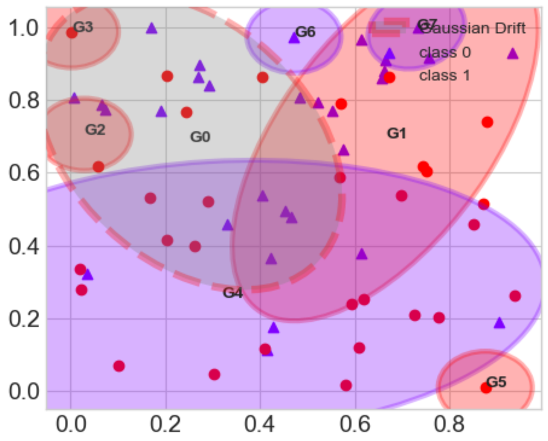
Figura 32 – Each plot shows one situation where concept drift took place in synthetic datasets. Gaussians that performed poorly are highlighted with dashed lines and gray background. Note that the most affected regions were close to the highlighted Gaussians.



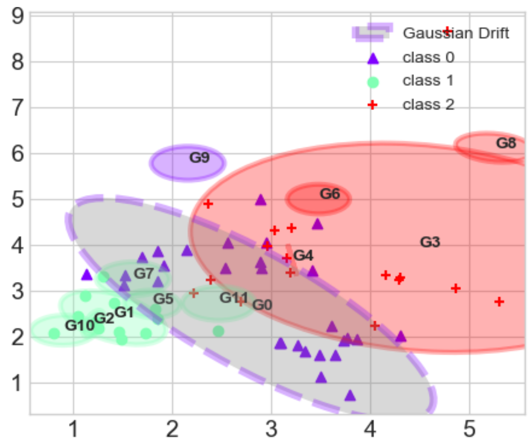
(a) Circles



(b) Sine1



(c) Sine2



(d) Virtual5

Source: Author.

6.7 HYPERPARAMETER SENSITIVITY ANALYSIS

In order to avoid results biased by the selection of the parameters for GMM-VRD, OGMMF-VRD, and GLDD-DU, we are going to analyze the results obtained for all tested parameters in the grid search shown in Tbl. 5. To measure the results we used only Average Accuracy and synthetic datasets. The discussions are divided in the following Subsections: (i) GMM-VRD's parameters (Section 6.7.1); (ii) OGMMF-VRD's parameters (Section 6.7.2); and (i) GLDD-DU's parameters (Section 6.7.3).

To analyze all the parameters at the same time, we used the Two-Way ANOVA in Average Accuracy according to Subsection 6.2.3. The parameters of the approaches and the datasets evaluated are called factors, and the values evaluated for each parameter are called factor levels. The interactions between 2 factors are represented by factor1*factor2. The other columns presents the type III of sum of squares (SS), mean squares (MS), degrees of freedom (DF), F statistics (F), confidence level (P-value) and Eta squared (Eta).

Statistical analysis of these factors is provided by three types of analyzes: (i) analysis of factor data, in which p-value is verified to determine whether the null hypothesis that the mean accuracy of the data is equal and the alternative hypothesis is that they are different; (ii) factor interaction analysis (factor1 * factor2), in which p-value is checked to determine whether the null hypothesis that there is no interaction between the factors, and alternative hypothesis that there is interaction; and (iii) analysis of the effect size of interaction, in which the value of eta in the factors and interactions are verified in order to know which is the configuration that has the greatest effect size on accuracy.

6.7.1 GMM-VRD's Hyperparameter Analysis

We evaluated two parameters of GMM-VRD: $Kmax$ the number of Gaussians generated per class during the training mode, and m the chunk size. Tbl. 10 presents the ANOVA test for these parameters. However, in this experiment, the interaction between them was not evaluated because we would like to only emphasize its individual effect. Given this, we first performed the experiment for the parameter $Kmax$ with $m = 50$, then using the best result obtained, that was $Kmax = 2$, we performed the experiment of the parameter m.

Tabela 10 – GMM-VRD's ANOVA Parameter Analysis. The p-value represents whether the group is statistically different. P-values with * mean that they are smaller than $\alpha = 0.05$. The eta represents in percentage what are the factors or interactions that most influence the results.

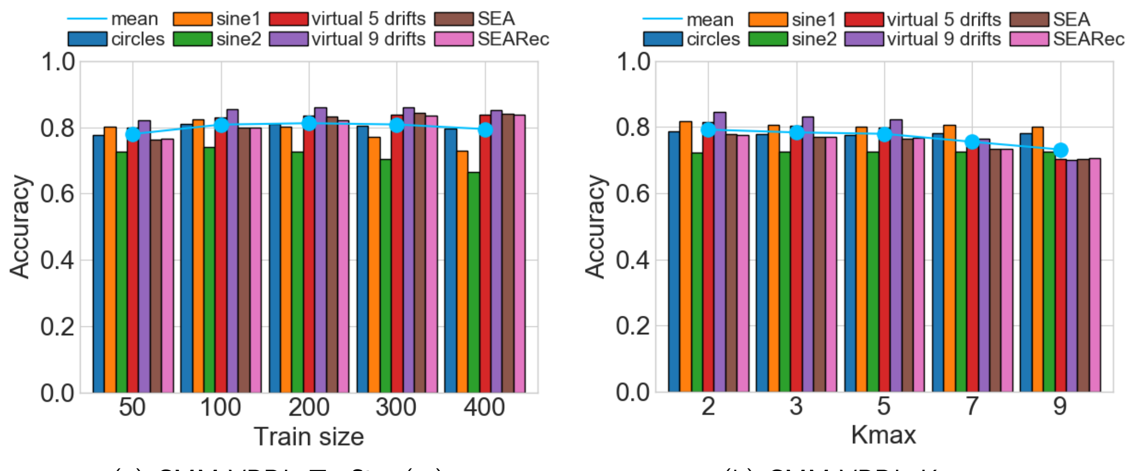
Approach	Factor/Int.	P-value	Eta
GMM-VRD	Kmax	7.7E-113*	0.065
	Datasets	2.9E-263*	0.731
	Kmax * Datasets	2.2E-162*	0.189
	m	3.9E-88*	0.154
	Datasets	6.00E-158*	0.687
	m * Datasets	1.5E-67*	0.121

Regarding (i) analysis of factor data, it is observed in Tbl 10 that all evaluated parameters have statistically different averages since all evaluated p-values are less than 0.05. Regarding

(ii) factor interaction analysis, it is observed that all evaluated p-value are less than 0.05, which indicates that there is interaction between the use of the parameters and the evaluated datasets, which means that one affect performance in the other. Regarding (iii) interaction effect size analysis, it is observed that the greatest effects are related to the datasets, which indicates that this approach is not tightly dependent of their hyperparameters, given that they have a very little variation over the datasets. In addition, we note that the m parameter has greater effect on the accuracy than $Kmax$ parameter, and the interaction of $m * Datasets$ has the third greater effect, which confirm that pre-adjustment of this parameter can improve the accuracy.

Finally, we present in Fig. 33 the average accuracy of GMM-VRD on each dataset, when varying their parameters. For GMM-VRD, the train size parameter (Figure 34(a)) tends to lead to better average accuracy when adopting larger values, reaching convergence in the value 200. This is understandable – more data for training results in a more stable system, thus leading to better accuracy. The Kmax (Fig. 34(b)) parameter leads to worse results with larger values, with exception for the sine2 dataset that has a complex distribution needing more Gaussians. This is because the criterion AIC tends to choose a number of Gaussians that overfits the data.

Figura 33 – Each bar represents the average accuracy of the 30 runs for a given data set. The bars were grouped by parameters. The blue line represents the mean across datasets.



(a) GMM-VRD's Tr. Size (m)

(b) GMM-VRD's Kmax

Source: Author.

6.7.2 OGMMF-VRD's Hyperparameter Analysis

For OGMMF-VRD we evaluated three parameters and its interactions: $Kmax$ the number of Gaussians generated per class during the training mode, m the chunk size, and $datasets$. Tbl. 11 presents the ANOVA test for these parameters.

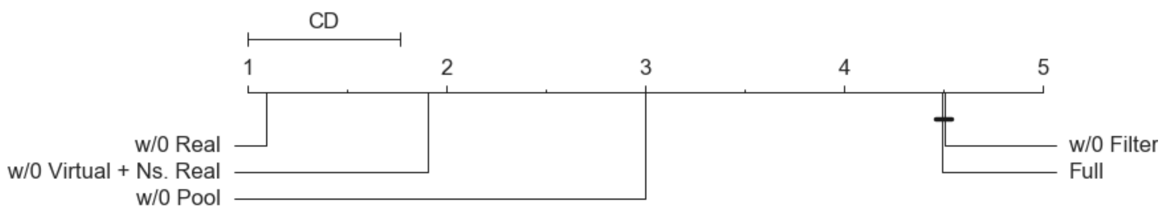
About OGMMF-VRD the same discussions made for GMM-VRD may apply. However, it is observed that for parameter m, the results vary much less compared to GMM-VRD. The reason for this is because the OGMMF-VRD can combine robust models (trained with many observations) and can be quickly replaced by intermediate models. These intermediate models are trained with few observations after the drift, and they are updated using *on-line* learning quickly, which gives the system less variability in performance.

Tabela 11 – OGMMF-VRD's ANOVA Parameter Analysis. The p-value represents whether the group is statistically different. P-values with * mean that they are smaller than $\alpha = 0.05$. The eta represents in percentage what are the factors or interactions that most influence the results.

Approach	Factor/Int.	P-value	Eta
OGMMF-VRD	Kmax	1.3E-113*	0.012
	m	0*	0.104
	Datasets	0*	0.797
	Kmax * m	1.4E-142*	0.017
	Kmax * Datasets	2.6E-42*	0.005
	m * Datasets	1.2E-240*	0.038
	Kmax * m * Datasets	1.3E-36*	0.006

In addition, we performed three more experiments. The first experiment is presented in Tbl. 12, presents a variety of values for the parameter Cfc that defines the size of the Gaussian that will be created in the model. We observed with the Friedman test that the variation of this parameter does not show any difference in the results, so we chose $Cfc = 20$ to fix this value. The second experiment is presented in Tbl. 13 and has the aim to evaluate the best parameter for the EDDM drift detector. The results have shown the same logic as the previous one, because of that we fixed $C = 2$. Finally, in the third experiment, we would like to understand better which components of our approaches are the most time consuming. For this, we evaluated (i) drift detector, (ii) virtual and non-severe real adaptation, and (iii) GMMs pool. Tbl 14 presents the numerical results and Fig. 34 the Friedman ranking. According to the results, the noise filter was the most expensive mechanism causing the system to get its current runtime.

Figura 34 – Friedman and Nemenyi tests for average time execution for each OGMMF-VRD's mechanisms in all synthetic datasets. Friedman's p-value was p=1.07E-49. Each approach represents OGMMF-VRD without a given mechanism (e.g. w/o Virtual + Ns. Real adaptation). The last column (Full) represents the complete OGMMF-VRD. Any pair of approaches whose distance between them is larger than CD is considered to be significantly different.



Source: Author.

Tabela 12 – Grid search for radius parameter of the OGMMF-VRD. The results highlighted with bold represent the best accuracy. According to the p-value 4.21E-01 obtained by the Friedman test. the H0 was not rejected. which means that this parameter did not have significant impact on the models' accuracy.

Datasets	**radius=10**	**radius=15**	**radius=20**	**radius=25**
Circles	0.8143 (0.003)	**0.8145 (0.003)**	0.813 (0.004)	0.8136 (0.003)
Sine1	0.8225 (0.003)	0.823 (0.004)	**0.8239 (0.003)**	0.8224 (0.003)
Sine2	0.7475 (0.003)	**0.7482 (0.004)**	0.7475 (0.003)	0.7481 (0.002)
Virtual5	**0.8299 (0.003)**	0.8295 (0.003)	0.8298 (0.003)	0.8292 (0.003)
Virtual9	0.8555 (0.003)	0.8543 (0.003)	**0.8569 (0.004)**	0.8558 (0.002)
SEA	0.8415 (0.003)	0.841 (0.005)	**0.8436 (0.004)**	0.8417 (0.003)

Tabela 13 – Grid search for EDDM drift level threshold used in OGMMF-VRD. The results highlighted with bold represent the best accuracy. According to the p-value 8.29E-01 obtained by the Friedman test, the H0 was not rejected, which means that this parameter did not have significant impact on the models' accuracy.

Datasets	**c=1**	**c=1.5**	**c=2**	**c=2.5**
Circles	**0.8138 (0.003)**	0.8133 (0.004)	0.813 (0.004)	0.8129 (0.004)
Sine1	0.8221 (0.003)	0.8219 (0.002)	**0.8227 (0.003)**	0.8212 (0.003)
Sine2	0.7478 (0.003)	0.7467 (0.003)	**0.7488 (0.003)**	0.7478 (0.003)
Virtual5	**0.8293 (0.004)**	0.8287 (0.003)	0.8288 (0.003)	0.8295 (0.003)
Virtual9	0.8563 (0.004)	**0.8573 (0.003)**	0.8564 (0.003)	0.8564 (0.004)
SEA	0.8416 (0.005)	**0.8416 (0.004)**	0.8408 (0.004)	**0.8416 (0.004)**

Tabela 14 – Time execution obtained by each of OGMMF-VRD's mechanisms. Each column represents OGMMF-VRD without a given mechanism (e.g. w/o non-severe drift adaptation). The results highlighted with bold represent the best time.

Datasets	w/o Virtual + Ns. Real	w/o Real	w/o Pool	w/o Filter	Full
Circles	35.4 (0.4)	**29.8 (1)**	39.4 (0.3)	45.8 (0.3)	46.1 (0.4)
Sine1	48.8 (0.5)	**43.8 (1.8)**	55.1 (0.2)	64.3 (0.3)	63.8 (0.5)
Sine2	52.6 (0.2)	**44.4 (1.1)**	57.3 (0.3)	67.9 (0.2)	67.8 (0.3)
Virtual5	52.1 (0.4)	**49 (1.3)**	56.4 (0.3)	68.1 (0.3)	68 (0.4)
Virtual9	40.5 (0.4)	**41.4 (2.2)**	47 (0.2)	55.4 (0.3)	55.5 (0.4)
SEA	35.2 (0.4)	**32.2 (0.5)**	40.1 (0.2)	47.4 (0.5)	47.5 (0.4)
SEARec	97.5 (0.3)	**86.1 (1)**	107.3 (0.3)	122.5 (0.3)	122.4 (0.6)

6.7.3 GLDD-DU's Hyperparameter Analysis

For GLDD-DU parameter analysis, we performed two ANOVA tests. The first checks the interaction of parameters N (Number of trained GMM models) with P (Pool size), since they depend on each other, and the second checks the impact of the parameter m (Batch size). We present the results of the ANOVA test in Tbl. 15 and the results of the parameters in Fig. 35.

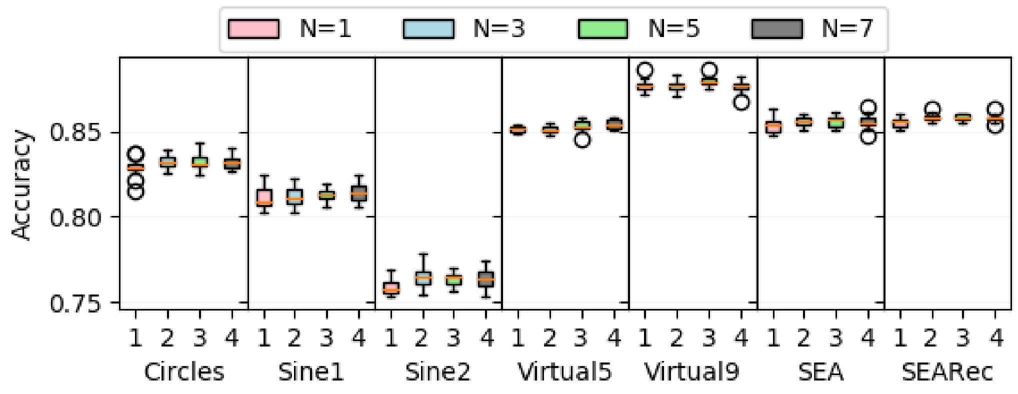
Regarding the parameters N and P, we see from the Eta value that the parameter N influences the results of more than P. The eta value represents the size of the effect (in percentage) that each parameter or interaction had in relation to all the factors measured. However, we found that even the interaction between them (N * P), does not affect the performance of GLDD-DU so much since its Eta was small (0.0002). The type of dataset is the one that most influences the results. The same discussions above can be applied to parameter m. However, we observed that the percentage Eta for m is larger than the other parameters, this indicates that it is the most influential parameter of GLDD-DU.

Tabela 15 – GLDD-DU's ANOVA Parameter Analysis. The p-value represents whether the group is statistically different. P-values with * mean that they are smaller than $\alpha = 0.05$. The eta represents in percentage what are the factors or interactions that most influence the results.

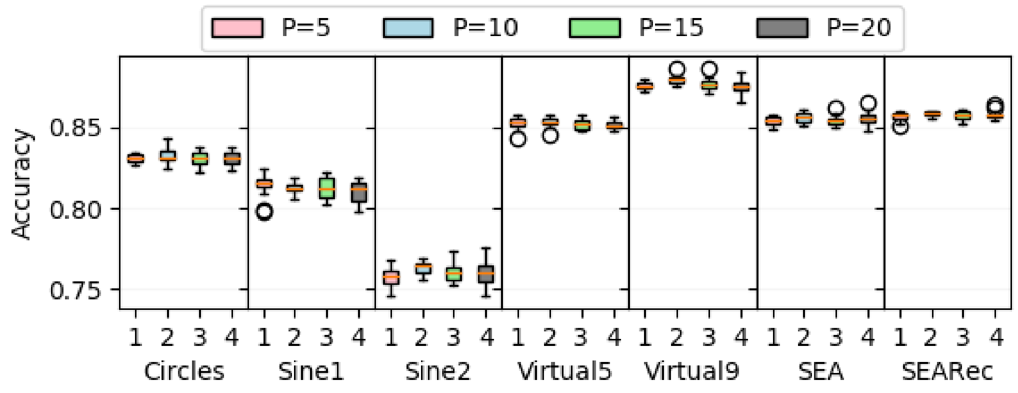
Approach	**ANOVA**	**Factor/Interaction**	**P-value**	**Eta**
GLDD-DU	N and P and Datasets	Datasets	1.78E-98*	0,98808
		N * Datasets	4.78E-46*	0,00126
		N	3.6E-47*	0,00103
		N * P * Datasets	0.22	0,00027
		P * Datasets	2.32E-04*	0,00021
		P	1.46E-09*	0,0002
		N * P	0.93	0,00002
	m and Datasets	Datasets	0.001*	0,9554
		m * Datasets	2,97E-87*	0,0185
		m	4,64E-80*	0,0138

These same discussions are illustrated in Fig 35. The parameter N (Fig. 36(a)), tends to have little variance over the datasets, only presenting a more significant variation in datasets like Sine1, Sine2, Virtual5 and Virtual9. This is because these datasets has a more complex distribution, so the more examples have been used to train the GMM models, the better the chances of having one with better performance. The parameter P (Fig. 36(b)), shows that values greater than 10 do not improve the results much. It shows that the Gaussians generated during the training period have good diversity. Finally, regarding parameter m, Fig. 36(c), shows that as the number of observations to train the GMM increases, the results tend to be better. This is because the greater the number of observations, the more representative the GMM modeling will be for the training set.

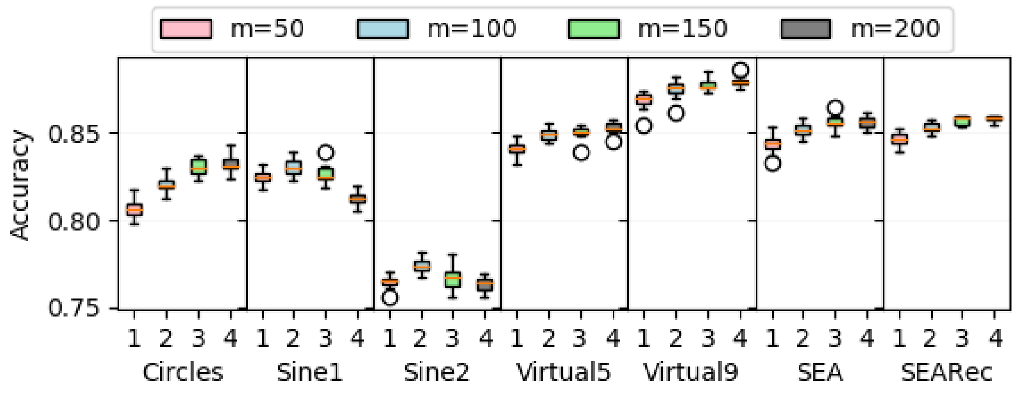
Figura 35 – Boxplot with the analysis of GLDD-DU's parameters for synthetic datasets. N represents the number of models generated during training. P the maximum number of models that can be saved in the pool. m represents the batch size of observations to train the model.



(a) Num Trained Models



(b) Pool Length



(c) Batch Size

Source: Author.

Concluding, we observed that the parameters of the GLDD-DU have a soft influence on its performance, and its most important parameter is the batch-size m. Even so, the results show that the type of dataset is the factor with a greater influence on the GLDD-DU performance, which indicates whether the GLDD-DU is more suitable or not.

6.8 CHAPTER SUMMARY

In this Chapter, we discussed five experiments: (i) comparison with literature works (Section 6.3) that answer RQ2; (ii) noise filter robustness (Section 6.4) that partially answer RQ3; (iii) impact of virtual and real drifts mechanisms (Section 6.5) that answer RQ3 and RQ4; (iv) impact of drift understanding mechanisms (Section 6.6) that answer RQ5 and RQ6; and (v) proposed methods' hyperparameter sensitivity analysis.

In the experiment of the comparison with the literature works, we found the proposed methods were competitive against the literature methods, especially the OGMMF-VRD. OGMMF-VRD presented more stable rankings regardless of the type of drifts and also obtained results equivalent to the ensembles methods, even using only a single one. This shows that handling different drifts simultaneously boosts system performance.

In the experiment of noise filter robustness, we observed that the noise filter significantly improved the performance of OGMMF-VRD. The results become even more significant when the number of observations in the training batch is increased. Finally, the use of the pool increased the system's robustness to false alarms caused by noisy observations since instead of resetting the system, it recovered a more accurate model.

In the experiment of the impact of virtual and real drifts mechanisms, we observed that the OGMMF-VRD's mechanisms to deal with virtual and real drifts significantly improved its results. This experiment demonstrated that both virtual and real drift degrade system performance, thus motivating the idea to handle both drifts with different strategies.

In the drift understanding mechanisms experiment, we observed that GLDD-DU's mechanism of generating a pool with different GMMs improved the system's adaptation to new concepts, thus significantly increasing its system performance. Furthermore, GLDD-DU's mechanism of replacing only the Gaussian with degradation was efficient, as it avoided a complete system reset and increased its performance.

Finally, in the proposed methods' hyperparameter sensitivity analysis experiment, we observed that the dataset type has more influence on the performance of the approaches than the parameters themselves. The most influential parameters are the number of observations used in the training batch and the number of Gaussians used in the system. In addition, we found that the noise filter is one of the most expensive proposed mechanisms in terms of runtime.

7 CONCLUSIONS

This Chapter summarizes the conclusions of the thesis and presents the directions of future work. The thesis focuses on understand the effects that virtual and real concept drift cause in the classifier. The main contributions are the answers to RQ1 to RQ6 outlined in Section 1 and summarized in Sections 7.1, 7.2, and 7.3.

7.1 IMPACT OF VIRTUAL AND READ DRIFTS ON CLASSIFIER SUITABILITY

No existing work provides an in depth understanding of the differences between the effect of virtual and real drifts on the suitability of classifiers. As a result, existing data stream learning approaches treat virtual drifts using the same strategies used for real drifts (KHAMASSI et al., 2018). In order to resolve this, Chapter 4 presented an analysis of the impacts of virtual and real drifts on Bayesian classifiers' suitability, answering RQ1 of the thesis:

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?

Overall, in terms of accuracy drop, we observed that both virtual and real drifts would result in an accuracy drop. The drop in accuracy for virtual drifts is likely to be smaller similar to what occurs in non-severe real drifts, as in both cases a good portion of the past knowledge will remain valid. 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.

7.2 IMPACT OF OF HANDLING VIRTUAL AND REAL DRIFTS SIMULTANEOUSLY

In general, most existing work on data stream learning focuses on real drifts, because the existing approaches typically ignore that virtual drifts can also affect the performance of the classifiers. As a result, existing data stream learning approaches treat virtual drifts resetting the classifier to learn it similar to the used for real drifts (KHAMASSI et al., 2018). However, such strategy is not the best because the knowledge acquired before the drift may remain valid after a virtual drift occurs.

Based on this, in Chapter 6, we discuss some experiments that validate the proposition of approaches that deal with virtual and real drifts simultaneously, answering the RQ2, RQ3, and RQ4 of the thesis:

RQ2) How to efficiently deal with virtual drifts at the same time as preserving the ability to deal with real drifts when using GMM?

To answer this question, we propose our first approach that deals simultaneously with virtual and real drifts, GMM-VRD. GMM-VRD handles virtual drifts using pertinence inferences to verify whether or not a new observation belongs to the trained distribution. If it belongs, it updates its knowledge with on-line learning. If it does not belong, it creates new Gaussians to represent these regions. GMM-VRD handles real drifts using classification inferences. These inferences are monitored using a drift detector, and whenever the detector reports a drift, the GMM-VRD is restarted. GMM-VRD performed with stability across datasets due to its different strategies, whereas other approaches that do not implement this sometimes performed very poorly.

RQ3) How to deal with both virtual and real drifts while achieving robustness to noise?

In the OGMMF-VRD method, we proposed two mechanisms to help deal with noisy observations: the noise filter and the pool of GMMs. The noise filter efficiently recognizes when a noisy observation appears in the data stream. This helps OGMMF-VRD to avoid on-line learning over these observations. The pool of GMMs has proven to be efficient in dealing with false alarms from the drift detector. Noisy observations cause the drift detector to confuse noisy observations with drifts, thus forcing a system reset. In this way, reusing the GMM models avoided a system reset. Overall, both mechanisms significantly improved OGMMF-VRD performance on datasets with varying noise levels.

RQ4) How to best harness knowledge gained from similar concepts to accelerate adaptation to both virtual and real drifts?

In OGMMF-VRD, we implemented an pool of GMMs that allowed the system to choose the best classifier in the presence of similar concepts regardless of the type of drift. The results showed that this mechanism led to statistically significant improvements even in datasets with no recurrent drifts.

7.3 IMPACT OF DRIFT UNDERSTANDING

Understanding how a drift affects the feature space and the classifier's performance helps choose the most appropriate strategies to deal with them. This process to know "when", "how"and "where"drifts occurred is called drift understanding and few approaches in the literature have implemented this. Knowing this, we propose GLDD-DU, an approach that implements drift understanding and we evaluate this approach in Chapter 6 to answer RQ5 and RQ6 of the thesis:

RQ5) How can we stimulate the generation of knowledge that is likely to help adaptation to new concepts when using GMMs?

In GLDD-DU, the mechanism to generate different subsamples of the training set using kDN was an excellent alternative to generate diverse GMM models with different capabilities to adapt to the data from the new concept. The choice of the best GMM model from the pool to make the *on-line* classification significantly improved the predictive system performance even in non-recurrent drifts.

RQ6) How do we use drift understanding to update only the portions of the learned decision boundaries that were affected by the concept drift?

In GLDD-DU, for each Gaussian in the GMM, we assign an EDDM drift detector to monitor its performance separately. For each Gaussian suffering from drift, a new Gaussians from the pool with the same class are tested to replace it in the GMM, and if one leads a better predictive performance, we replace. As these Gaussians were generated for different kDN configurations, some may have already started to deal with drift because they may have been trained with more data representative of the new concept. The experiments also showed that making updates in local decision boundaries improved the results considerably on real non-severe drifts.

7.4 FUTURE WORK

Although the results have shown a competitive performance of the proposed approaches, our proposals have the following limitations:

The proposed approaches are directly dependent on the use of GMM, indicating that other classifiers cannot be used and the proposed systems tend to suffer from the same weaknesses as

GMM. An example of this would be data with high overlap. Overlapping data causes different classes to be located in the same region of space. GMM can only learn well separated data. So in overlapping data, GMM tends to confuse overlapping classes causing classification errors. An alternative to this would be creating a classifier-independent framework that simultaneously handles virtual and real drifts. The fundamentals discussed of the impacts of each drift can serve to guide this framework.

The proposed approaches have limited performance because they are based on a single classifier. Approaches that use ensembles tend to perform best at times where there is no concept drift. Therefore, thinking of an ensemble method that can deal expertly with virtual and real drifts is also an alternative.

The proposed approaches are restricted to real-world problems that have a definite time step for the emergence of the labels, that is, they do not deal with the delay in receiving the true classes. So, investigating tasks with this problem would be an alternative, as the task of software defect detection. In this task, in addition to the delay in receiving the labels, the data has unbalanced.

REFERÊNCIAS

ALIPPI, C.; BORACCHI, G.; ROVERI, M. Just-in-time classifiers for recurrent concepts. *IEEE transactions on neural networks and learning systems*, IEEE, v. 24, n. 4, p. 620–634, 2013.

ALMEIDA, P. R.; OLIVEIRA, L. S.; JR, A. S. B.; SABOURIN, R. Adapting dynamic classifier selection for concept drift. *Expert Systems with Applications*, Elsevier, v. 104, p. 67–85, 2018.

BAENA-GARCIA, M.; CAMPO-ÁVILA, J. del; FIDALGO, R.; BIFET, A.; GAVALDA, R.; MORALES-BUENO, R. Early drift detection method. In: *KDD*. [S.l.: s.n.], 2006. v. 6, p. 77–86.

BAILEY, T. L.; ELKAN, C. et al. Fitting a mixture model by expectation maximization to discover motifs in bipolymers. Department of Computer Science and Engineering, University of California . . . , 1994.

BIFET, A.; GAVALDA, R. Learning from time-changing data with adaptive windowing. In: SIAM. *Proceedings of the 2007 SIAM international conference on data mining*. [S.l.], 2007. p. 443–448.

BIFET, A.; GAVALDÀ, R. Adaptive learning from evolving data streams. In: SPRINGER. *International Symposium on Intelligent Data Analysis*. [S.l.], 2009. p. 249–260.

BIFET, A.; HOLMES, G.; PFAHRINGER, B. Leveraging bagging for evolving data streams. In: SPRINGER. *Joint European conference on machine learning and knowledge discovery in databases*. [S.l.], 2010. p. 135–150.

BUDIMAN, A.; FANANY, M. I.; BASARUDDIN, C. Adaptive online sequential elm for concept drift tackling. *Hindawi CIN*, v. 2016, 2016.

CANO, A.; GÓMEZ-OLMEDO, M.; MORAL, S. A bayesian approach to abrupt concept drift. *Knowledge-Based Systems*, Elsevier, v. 185, p. 104909, 2019.

CAVALCANTE, R. C.; OLIVEIRA, A. L. An approach to handle concept drift in financial time series based on extreme learning machines and explicit drift detection. In: *IEEE IJCNN*. [S.l.: s.n.], 2015. p. 1–8.

CHEN, T.; ZHANG, J. On-line multivariate statistical monitoring of batch processes using gaussian mixture model. *Computers & chemical engineering*, Elsevier, v. 34, n. 4, p. 500–507, 2010.

CHOI, S. W.; PARK, J. H.; LEE, I.-B. Process monitoring using a gaussian mixture model via principal component analysis and discriminant analysis. *Computers & chemical engineering*, Elsevier, v. 28, n. 8, p. 1377–1387, 2004.

DEMŠAR, J. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, v. 7, n. Jan, p. 1–30, 2006.

DITZLER, G.; POLIKAR, R. Semi-supervised learning in nonstationary environments. In: IEEE. *IJCNN, The 2011 IJCNN on*. [S.l.], 2011. p. 2741–2748.

DITZLER, G.; ROVERI, M.; ALIPPI, C.; POLIKAR, R. Learning in nonstationary environments: A survey. *IEEE CIM*, v. 10, n. 4, p. 12–25, 2015.

ENGEL, P. M.; HEINEN, M. R. Incremental learning of multivariate gaussian mixture models. In: SPRINGER. *BRACIS*. [S.l.], 2010. p. 82–91.

GAMA, J.; CASTILLO, G. Learning with local drift detection. In: SPRINGER. *International Conference on Advanced Data Mining and Applications*. [S.l.], 2006. p. 42–55.

GAMA, J.; MEDAS, P.; CASTILLO, G.; RODRIGUES, P. Learning with drift detection. In: SPRINGER. *IEEE BRACIS*. [S.l.], 2004. p. 286–295.

GAMA, J.; ŽLIOBAITĖ, I.; BIFET, A.; PECHENIZKIY, M.; BOUCHACHIA, A. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, ACM, v. 46, n. 4, p. 44, 2014.

GOLDENBERG, I.; WEBB, G. I. Survey of distance measures for quantifying concept drift and shift in numeric data. *Knowledge and Information Systems*, Springer, p. 1–25, 2019.

GOMES, H. M.; BARDDAL, J. P.; ENEMBRECK, F.; BIFET, A. A survey on ensemble learning for data stream classification. *ACM CSUR*, v. 50, n. 2, p. 23, 2017.

GOMES, H. M.; BIFET, A.; READ, J.; BARDDAL, J. P.; ENEMBRECK, F.; PFHARINGER, B.; HOLMES, G.; ABDESSALEM, T. Adaptive random forests for evolving data stream classification. *Machine Learning*, Springer, v. 106, n. 9-10, p. 1469–1495, 2017.

GREENHOUSE, S. W.; GEISSER, S. On methods in the analysis of profile data. *Psychometrika*, Springer, v. 24, n. 2, p. 95–112, 1959.

GRIM, J. Em cluster analysis for categorical data. In: SPRINGER. *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*. [S.l.], 2006. p. 640–648.

IDREES, M. M.; MINKU, L. L.; STAHL, F.; BADII, A. A heterogeneous online learning ensemble for non-stationary environments. *Knowledge-Based Systems*, Elsevier, v. 188, p. 104983, 2020.

JR, P. M. G.; SANTOS, S. G. de C.; BARROS, R. S.; VIEIRA, D. C. A comparative study on concept drift detectors. *Expert Systems with Applications*, Elsevier, v. 41, n. 18, p. 8144–8156, 2014.

KHAMASSI, I.; SAYED-MOUCHAWEH, M.; HAMMAMI, M.; GHÉDIRA, K. Discussion and review on evolving data streams and concept drift adapting. *Evolving systems*, Springer, v. 9, n. 1, p. 1–23, 2018.

KRAWCZYK, B.; MINKU, L. L.; GAMA, J.; STEFANOWSKI, J.; WOŹNIAK, M. Ensemble learning for data stream analysis: A survey. *Elsevier IF*, v. 37, p. 132–156, 2017.

KUNCHEVA, L. I. Classifier ensembles for detecting concept change in streaming data: Overview and perspectives. In: *2nd Workshop SUEMA*. [S.l.: s.n.], 2008. v. 2008, p. 5–10.

LEE, T.-W.; LEWICKI, M. S.; SEJNOWSKI, T. J. Ica mixture models for unsupervised classification of non-gaussian classes and automatic context switching in blind signal separation. *IEEE TPAMI*, IEEE, v. 22, n. 10, p. 1078–1089, 2000.

LIU, A.; LU, J.; LIU, F.; ZHANG, G. Accumulating regional density dissimilarity for concept drift detection in data streams. *Pattern Recognition*, Elsevier, v. 76, p. 256–272, 2018.

LIU, A.; SONG, Y.; ZHANG, G.; LU, J. Regional concept drift detection and density synchronized drift adaptation. In: *IJCAI International Joint Conference on Artificial Intelligence*. [S.l.: s.n.], 2017.

LU, J.; LIU, A.; DONG, F.; GU, F.; GAMA, J.; ZHANG, G. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, IEEE, v. 31, n. 12, p. 2346–2363, 2018.

LU, N.; ZHANG, G.; LU, J. Concept drift detection via competence models. *Artificial Intelligence*, Elsevier, v. 209, p. 11–28, 2014.

MAUCHLY, J. W. Significance test for sphericity of a normal n-variate distribution. *The Annals of Mathematical Statistics*, JSTOR, v. 11, n. 2, p. 204–209, 1940.

MINKU, L. L.; WHITE, A. P.; YAO, X. The impact of diversity on online ensemble learning in the presence of concept drift. *IEEE TKDE*, v. 22, n. 5, p. 730–742, 2009.

MINKU, L. L.; YAO, X. Ddd: A new ensemble approach for dealing with concept drift. *IEEE TKDE*, v. 24, n. 4, p. 619–633, 2012.

MOON, T. K. The expectation-maximization algorithm. *IEEE Signal processing magazine*, IEEE, v. 13, n. 6, p. 47–60, 1996.

OLIVEIRA, G. H.; CAVALCANTE, R. C.; CABRAL, G. G.; MINKU, L. L.; OLIVEIRA, A. L. Time series forecasting in the presence of concept drift: A pso-based approach. In: IEEE. *ICTAI, 2017 IEEE 29th International Conference on*. [S.l.], 2017. p. 239–246.

OLIVEIRA, G. H.; MINKU, L. L.; OLIVEIRA, A. L. Gmm-vrd: A gaussian mixture model for dealing with virtual and real concept drifts. In: *IEEE IJCNN*. [S.l.: s.n.], 2019. p. 1–8.

OLIVEIRA, G. H.; MINKU, L. L.; OLIVEIRA, A. L. Tackling virtual and real concept drifts: An adaptive gaussian mixture model approach. *IEEE Transactions on Knowledge and Data Engineering*, IEEE, 2020.

OLIVEIRA, G. H. F. M.; MINKU, L. L.; OLIVEIRA, A. L. I. Gmm-vrd: A gaussian mixture model for dealing with virtual and real concept drifts. In: IEEE. *IJCNN, The 2019 IJCNN on*. [S.l.], 2019.

OLIVEIRA, L. S.; BATISTA, G. E. Igmm-cd: a gaussian mixture classification algorithm for data streams with concept drifts. In: IEEE. *BRACIS, 2015 Brazilian Conference on*. [S.l.], 2015. p. 55–61.

OZA, N. C. Online bagging and boosting. In: IEEE. *2005 IEEE international conference on systems, man and cybernetics*. [S.l.], 2005. v. 3, p. 2340–2345.

PAGE, E. S. Continuous inspection schemes. *Biometrika*, v. 41, n. 1/2, p. 100–115, 1954.

PESARANGHADER, A.; VIKTOR, H.; PAQUET, E. Reservoir of diverse adaptive learners and stacking fast hoeffding drift detection methods for evolving data streams. *Machine Learning*, Springer, v. 107, n. 11, p. 1711–1743, 2018.

PESARANGHADER, A.; VIKTOR, H. L. Fast hoeffding drift detection method for evolving data streams. In: *ECML PKDD*. [S.l.: s.n.], 2016. p. 96–111.

POLIKAR, R.; UPDA, L.; UPDA, S. S.; HONAVAR, V. Learn++: An incremental learning algorithm for supervised neural networks. *IEEE transactions on systems, man, and cybernetics, part C (applications and reviews)*, IEEE, v. 31, n. 4, p. 497–508, 2001.

REN, S.; LIAO, B.; ZHU, W.; LI, K. Knowledge-maximized ensemble algorithm for different types of concept drift. *Information Sciences*, v. 430, p. 261–281, 2018.

ROSS, G. J.; ADAMS, N. M.; TASOULIS, D. K.; HAND, D. J. Exponentially weighted moving average charts for detecting concept drift. *Pattern recognition letters*, Elsevier, v. 33, n. 2, p. 191–198, 2012.

RUANO-ORDAS, D.; FDEZ-RIVEROLA, F.; MENDEZ, J. R. Concept drift in e-mail datasets: An empirical study with practical implications. *Information Sciences*, Elsevier, v. 428, p. 120–135, 2018.

SANTOS, S. G.; BARROS, R. S.; JR, P. M. G. A differential evolution based method for tuning concept drift detectors in data streams. *Information Sciences*, Elsevier, v. 485, p. 376–393, 2019.

SOBOLEWSKI, P.; WOŹNIAK, M. Comparable study of statistical tests for virtual concept drift detection. In: *CORES*. [S.l.: s.n.], 2013. p. 329–337.

SONG, M.; WANG, H. Highly efficient incremental estimation of gaussian mixture models for online data stream clustering. In: INTERNATIONAL SOCIETY FOR OPTICS AND PHOTONICS. *Intelligent Computing: Theory and Applications III*. [S.l.], 2005. v. 5803, p. 174–184.

SOUZA, V. M.; REIS, D. M. dos; MALETZKE, A. G.; BATISTA, G. E. Challenges in benchmarking stream learning algorithms with real-world data. *Data Mining and Knowledge Discovery*, Springer, v. 34, n. 6, p. 1805–1858, 2020.

SUN, Y.; TANG, K.; MINKU, L. L.; WANG, S.; YAO, X. Online ensemble learning of data streams with gradually evolved classes. *IEEE TKDE*, IEEE, v. 28, n. 6, p. 1532–1545, 2016.

WALMSLEY, F. N.; CAVALCANTI, G. D.; OLIVEIRA, D. V.; CRUZ, R. M.; SABOURIN, R. An ensemble generation methodbased on instance hardness. *arXiv preprint arXiv:1804.07419*, 2018.

WANG, H.; FAN, W.; YU, P. S.; HAN, J. Mining concept-drifting data streams using ensemble classifiers. In: *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.: s.n.], 2003. p. 226–235.

WANG, S.; MINKU, L. L.; YAO, X. A systematic study of online class imbalance learning with concept drift. *IEEE TNNLS*, IEEE, 2018.

WEBB, G. I.; HYDE, R.; CAO, H.; NGUYEN, H. L.; PETITJEAN, F. Characterizing concept drift. *KDD*, v. 30, n. 4, p. 964–994, 2016.

XIAO, J.; XIAO, Z.; WANG, D.; BAI, J.; HAVYARIMANA, V.; ZENG, F. Short-term traffic volume prediction by ensemble learning in concept drifting environments. *Knowledge-Based Systems*, Elsevier, v. 164, p. 213–225, 2019.

YAMAUCHI, K. Incremental learning and model selection under virtual concept drifting environments. In: *IEEE IJCNN*. [S.l.: s.n.], 2010. p. 1–8.

YU, J.; QIN, S. J. Multimode process monitoring with bayesian inference-based finite gaussian mixture models. *AIChE Journal*, v. 54, n. 7, p. 1811–1829, 2008.