GMM-VRD: A Gaussian Mixture Model for Dealing With Virtual and Real Concept Drifts

Gustavo H. F. M. Oliveira
Centro de Informática
Universidade Federal de Pernambuco
Recife, Pernambuco, Brazil
ghfmo@cin.ufpe.br

Leandro L. Minku
School of Computer Science
University of Birmingham
Birmingham, UK
L.L.Minku@cs.bham.ac.uk

Adriano L. I. Oliveira
Centro de Informática
Universidade Federal de Pernambuco
Recife, Pernambuco, Brazil
alio@cin.ufpe.br

***Abstract*—Concept drift is a change in the joint probability distribution of the problem. This term can be subdivided into two types: real drifts that affect the conditional probabilities $p(y|x)$ or virtual drifts that affect the unconditional probability distribution $p(x)$. Most existing work focuses on dealing with real concept drifts. However, virtual drifts can also cause degradation in predictive performance, requiring mechanisms to be tackled. Moreover, as virtual drifts frequently mean that part of the old knowledge remains useful, they require different strategies from real drifts to be effectively tackled. Motivated on this, we propose an approach called Gaussian Mixture Model for Dealing With Virtual and Real Concept Drifts (GMM-VRD), which updates and creates Gaussians to tackle virtual drifts and resets the system to deal with real drifts. The main results show that the proposed approach obtained the best results, in terms of average accuracy, in relation to the literature methods, which propose to solve that same problem. In terms of accuracy over time, the proposed approach showed lower degradation on concept drifts, which indicates that the proposed approach was efficient.**

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

I. INTRODUCTION

In recent years, real-world applications have dealt with tremendous growth in the amount of information, where data arrive continuously and sequentially over time and tend to evolve due to the dynamics of real-world activities. This type of process is known as data streams [1]. Data streams present several challenges for data modeling systems [2], because, with drifts over time, the information used for modeling is no longer useful for defining or deducing future behaviors. This phenomenon is called a concept drift [3].

Concept drift can be subdivided into two types: real concept drift and virtual concept drift [4]. The real concept drift can be defined as a change in the conditional probabilities $p(y|x)$. Virtual concept drift can be defined as a change in the unconditional probability distribution $p(x)$. In practice, both real and virtual concept drifts can occur simultaneously.

Most existing work on tackling concept drift focuses on real concept drifts, which directly affect the performance of the classifiers through changes in the true decision boundaries of the problem [5]. Although virtual concept drift has attracted much less attention from the research community, it can also cause degradation of the classifier's predictive performance. This is because even though virtual drifts do not affect the true decision boundaries of the problem, the appearance of observations in regions of the search space that the classifier does not know about can affect the suitability of the *learned* decision boundaries. Therefore, it becomes important and necessary to readjust the margins of the learned classifier to avoid misclassification.

In general, existing approaches deal with concept drift by learning new concepts from scratch [5]. This strategy is ideal for real drifts where the new concepts do not share similarities with the old ones. But, if the drift is virtual, part of the past knowledge remains useful [2]. In that case, learning the new concept entirely from scratch is not ideal. It results in the system wasting potentially useful knowledge and spending a lot of time to collect new data to recover from the drift, hindering predictive performance. Therefore, different strategies are better suited to deal with each of these two types of drift.

An approach with the potential to deal with this type of problem is the Gaussian Mixture Model (GMM). GMM has probabilistic calculations of pertinence and classification, which have been adopted in several pattern recognition problems, e.g., [6]. It can also represent complex distributions and model both real [7] and virtual drifts [8], potentially enabling us to adopt different strategies for each of these types of drift. Besides that, it has been applied to non-stationary problems for a long time [9] [10] [7].

Given the need to deal with both types of concept drift and the potential of GMM, we aim to answer the following research question: how to efficiently deal with virtual drifts at the same time as 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 in an appropriate way. Our hypothesis results in a novel approach called Gaussian Mixture Model for Dealing With Virtual and Real Concept Drifts (GMM-VRD), which uses probabilistic pertinence to decide whether to update and create new Gaussians to tackle virtual drifts, and a drift detection method to decide whether to reset the system to deal with real drifts. We evaluate our proposal with approaches of the literature that claim to deal with the same problem, using metrics such as average accuracy and accuracy over time in both artificial and real datasets. Our

978-1-7281-2009-6/$31.00 ©2019 IEEE paper N-19437.pdf

experiments show that GMM-VRD obtained the best overall accuracy across datasets because of the different strategies used for virtual and real drifts.

This paper is further organized as follows. Section II presents related work that claims to deal with both virtual and real concept drifts. Section III presents background necessary to understand our proposed approach. Section IV presents the proposed approach in detail. Section V describes the experiments. In Section VI discuss the results. Section VII presents our conclusions and gives directions for further research.

II. RELATED WORK

The following two studies claim to deal with both virtual and real concept drift: Oliveira et al.'s Incremental Gaussian Mixture Model for Concept Drift (IGMM-CD) [7] and Almeida et al.'s Dynamic Selection Based Drift Handler (Dynse) [11].

IGMM-CD is an incremental approach based on 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 creation of a new Gaussian can be used to rapidly adapt the system to real concept drift. In the same way, the update of the nearest Gaussian of the observation is used to adapt the system to virtual drifts. Besides that, the system has a mechanism to exclude obsolete Gaussians based on the parameter T, that counts the number of Gaussians existing per class. If that number exceeds, 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.

Dynse is an approach based on Dynamic Classifier Selection (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 drifts can be dealt with using a 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 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 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 above can have their performance degraded over time because they do not use strategies that are always specific to the needs of each type of drift. Based on this, we elaborate on our proposal different strategies to deal with each type of drift and avoid degradation over time.

III. GAUSSIAN MIXTURE MODEL

In sections III-A and III-B we are going to discuss how the GMM can be used to recognize patterns in unsupervised and supervised problems, respectively.

A. 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 [7]. 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:

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

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 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:

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

The approach commonly used to make the adjustment of the GMM over the data is called Expectation-Maximization (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. Details on this iterative procedure can be found at [12].

As a disadvantage, this approach is sensitive to noise, which can cause the components to fit too tightly to the data, as an overfitting.

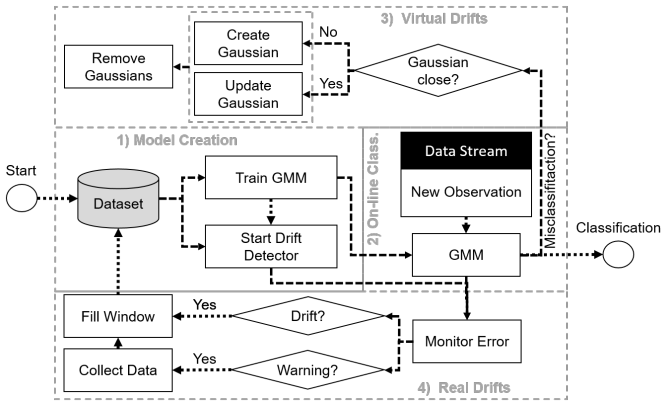


Fig. 1: GMM-VRD's Overall Procedure.

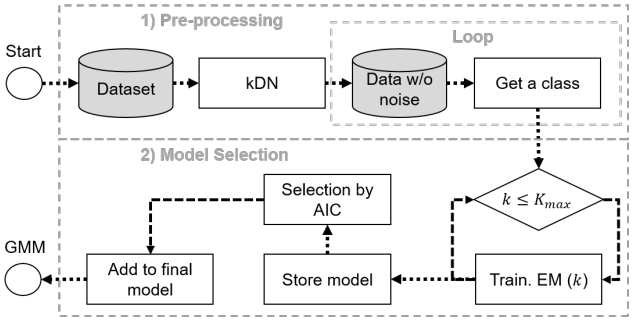


Fig. 2: Training architecture proposed for GMM.

B. Supervised learning

The GMM can also be applied as a supervised learning approach [7]. 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 estimated for each class [7]. The disadvantage of this approach it that it cannot represent complex distributions, which in turn results in poor performance. The second one is using one GMM per class [7], where the unsupervised learning discussed in Section III-A 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 of each class. In both cases, the Gaussians represent the decision boundaries and the process to classify is based on the higher posterior probability (Eq. 3) given by the Equation 4, where K represents the number of Gaussians existing in the system.

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

$$\hat{y} = argmax_{i \in \{1,2,\cdots,K\}} P(C_i|x) \tag{4}$$

IV. PROPOSED METHOD

Based on the necessity to deal with different types of drift, we propose the GMM-VRD. The overall procedure of the proposed approach is illustrated in Figure 1.

The proposed approach is divided into four parts: (i) the procedure to train a new GMM model discussed in section IV-A; (ii) the procedure to classify a new observation from the data stream, discussed on section IV-B; (iii) the procedure to deal with virtual drifts, discussed in section IV-C; and (iv) the procedure to deal with real drifts, discussed in section IV-D.

A. Model Creation

The model creation step has two substages: IV-A1: the initialization of the classifier and; IV-A2: the initialization of the drift detector.

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 III. The architecture is illustrated in Figure 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 k-Disagreeing Neighbors (kDN) discussed in [13] to define the hardness of each instance in the training dataset, as defined in Equation 5. 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. For our approach, we have specified that observations with kDN greater than 0.75 tend to be noise and should thus be removed from the original training set. After, the observations are separated by class to train a GMM per class.

$$kDN(x) = \frac{|x'|x' \in kNN(x) \wedge label(x') \neq label(x)}{k} \tag{5}$$

The model selection stage trains GMMs with numbers of Gaussians ranging from 1 to $Kmax$ using the EM approach [12]. The best resulting model is chosen using the higher value of the AIC criterion and is added to the final GMM model. The AIC criterion is defined as follows:

$$AIC = 2 \cdot p - 2 \cdot L \tag{6}$$

where p represents the number of model parameters. In a GMM with only one Gaussian, the parameters are the mean, covariance, and weight. 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 7:

$$L = \sum_{i=1}^{m} log \sum_{j=1}^{K} P(x_i|C_j) \cdot w_j \tag{7}$$

where m represents the number of observations used for training and K the number of Gaussians in the GMM model.

Algorithm 1 GaussianClose()

Input: observation (x_t, y_t)
1: aux = $\varnothing$
2: **for** each gaussian in GMM **do**
3: **if** gaussian $\in y_t$ **then**
4: aux $\leftarrow P(x_t|gaussian)$ ▷ Eq. 2
5: **else**
6: aux $\leftarrow 0$
7: **end if**
8: **end for**
9: gaussian = $argmax$(aux)
10: **if** $aux[gaussian] > 0$ **then**
11: UpdateGaussian(gaussian, x_t)
12: **else**
13: CreateGaussian(x_t, y_t)
14: **end if**

2) Drift Detector Initialization: We use Exponentially Weighted Moving Average Charts (ECDD) [14] 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., [15]. Other drift detection methods could be investigated in future work.

B. On-line Classification

In the second part, as more incoming instances arrive, the trained model is used to predict their respective labels. The prediction is done using Equation 4. The predicted label can be used by users for decision-making.

C. Adaptation to Virtual Drifts

This component of the proposed 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 GaussianClose() described in Algorithm 1.

This routine computes the conditional probability given by Equation 2 for all Gaussians existing in the current GMM, corresponding to lines 1 to 9. If there is a Gaussian near, then the model is updated according to line 11 and to the subsection IV-C1. If there is not, then another Gaussian is created, according to line 13 and as explained in subsection IV-C2. Both situations are illustrated in Figures 3c, 3d, 3e and 3f.

1) Update a Gaussian: When the nearest Gaussian has pertinence greater than zero 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 [16]. 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 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 Equations to update the Gaussian are shown below:

$$sp_i^t = sp_i^{t-1} + P(C_i|x) \tag{8}$$

$$w_i^t = \frac{sp_i^t}{\sum_j^K sp_j^t} \tag{9}$$

$$\mu_i^t = \mu_i^{t-1} + \frac{P(C_i|x)}{sp_i^t} + (x - \mu_i^{t-1}) \tag{10}$$

$$\begin{aligned}\Sigma_i^t = \Sigma_i^{t-1} - (\mu_i^t - \mu_i^{t-1})^T(\mu_i^t - \mu_i^{t-1}) \\ + \frac{P(C_i|x)}{sp_i^t} \cdot [\Sigma_i^{t-1} - (x - \mu_i^t)^T(x - \mu_i^t)]\end{aligned} \tag{11}$$

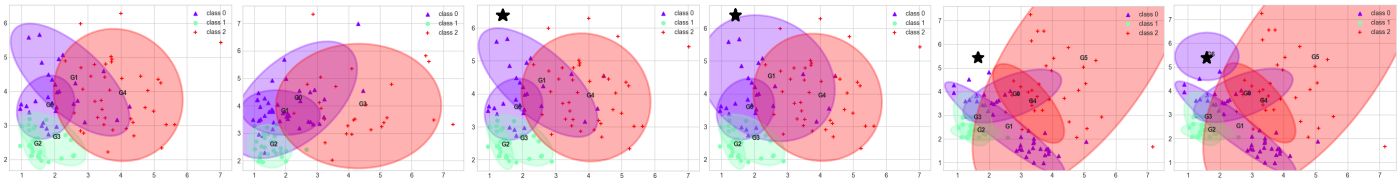
2) Create a Gaussian: When no Gaussian has pertinence for the observation consulted, then it is necessary to create another Gaussian to represent the new region in the feature space. The new Gaussian is initialised using sp_i and $w_i = 1$, $\mu_i = x_i$ and $\Sigma_i = 0.5 \cdot I$, where I represents the identity matrix, which has the same number of dimensions as x. The value 0.5 represents the size of the Gaussian circumference and can be defined according to the problem.

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 9.

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 extremely small, next to zero. This process is repeated whenever the adaptation virtual drifts is triggered.

D. Adaptation to Real Drifts

We consider that the real drifts requiring treatment will significantly degrade the system's predictive performance. Given the adoption of the mechanisms explained in IV-C, we also consider that virtual drifts will not degrade the system's predictive performance so much as real drifts. Therefore, the proposed approach detects the need for treating real drifts through a drift detection method that monitors the error of the system over time. In this work, ECDD [14] was used as the drift detection method. 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



(a) Initial Model (b) Model redefined (c) Update (Before) (d) Update (After) (e) Create (Before) (f) Create (After)

Fig. 3: Illustration of the adaptation to real and virtual drifts. On Figures 3c, 3d, 3e, 3f the star-shaped observation represents the new test instance. The text G0, ..Gn represent the number of the Gaussian.

TABLE I: Dataset descriptions.

Type	Datasets	Classes	Attributes	Examples	Drift Type	Drifts	Concept Size
Synthetic	Circles	2	2	8000	Gradual	4	2000
	Sine1	2	2	10000	Abrupt	5	2000
	Sine2	2	2	10000	Abrupt	5	2000
	Virtual 5 Drifts	3	2	10000	Virtual	5	2000
	Virtual 9 Drifts	3	2	10000	Virtual	9	1000
	SEA	2	3	8000	Gradual	4	2000
	SEAREC	2	3	16000	Recurrent (Repeat SEA)	8	2000
Real	PAKDD	2	29	50000	-	-	-
	ELEC	2	4	27549	-	-	-
	NOAA	2	9	18159	-	-	-

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 3a and 3b. In the second case (3b) the model had to be redefined because it was degrading.

V. EXPERIMENTAL SETUP

Experiments were performed with the objective of evaluating the proposed approach against related work in terms of predictive performance and sensitivity to parameters. The datasets used in the experiments are described in section V-A. The performance metrics are presented in section V-B. The setup for comparing the approaches in terms of predictive performance is presented in section V-C. The setup for analyzing their sensitivity to parameters is explained in section V-D.

A. Datasets

Both synthetic and real world datasets were used. The datasets are summarised in Table I.

1) Synthetic datasets: We used 7 synthetic datasets with varied types of drift to investigate how each compared approach behaves. With exception of the virtual datasets, all of them were generated using the Tornado framework proposed by [17] and available on GitHub[1]. The virtual datasets were generated for this paper, to understand how virtual drifts can affect the approaches. An example of virtual drift dataset is shown in Figure 4.

2) Real datasets: The real datasets used on this paper are available in [18]. Only one modification was made on ELEC dataset, that had missing values. The missing values were removed resulting in the number of examples in Table I.

[1]https://github.com/alipsgh/tornado

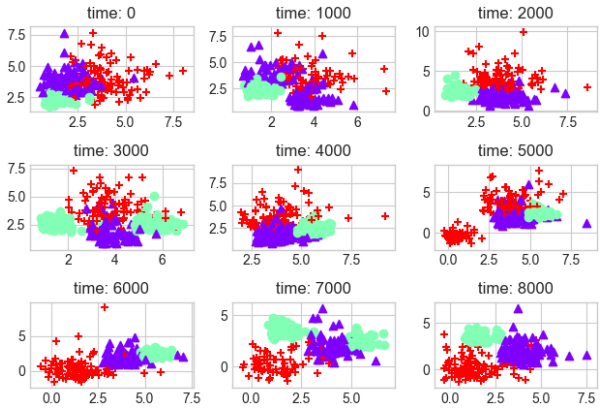


Fig. 4: Illustration of the virtual 9 drifts dataset.

B. Metrics

To evaluate the predictive performance of the approaches, we used two metrics: average accuracy and accuracy over time. To attest the significance of the results we used Friedman and Nemenyi statistical tests.

1) Average Accuracy: is a performance indicator to evaluate how the system behaved across the data stream, according to Equation 12. In order to increase the discriminative power of the metric, the standard deviation between the several accuracies is also reported. This metric was used in several works as [19, 7, 11].

$$accuracy = \frac{\text{Number of correct predictions}}{\text{Total number of predictions made}} \quad (12)$$

2) 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 13. This metric enables us to investigate the evolution of the system's accuracy over time, and was also used in several previous studies as [19, 7, 11].

$$AOT = \{accuracy(batch_1), ..., accuracy(batch_n)\} \quad (13)$$

3) Friedman and Nemenyi tests: Friedman [20] is a non-parametric statistical test that can be used to compare multiple approaches accross multiple datasets. If the null hypothesis is rejected, then the Nemenyi post-hoc is used to check which

of the approaches is different from each other. Both tests are used with a significance level of α = 0.05.

C. Comparison in Terms of Predictive Performance

To evaluate the performance of the proposed approach, we compare it with the two existing approaches that propose to solve the same problem of this paper, i.e., to deal with both virtual and real concept drifts. The existing approaches are Dynse [11] and IGMM-CD [7], both discussed in section II. The parameters of all approaches were chosen based on experiments with several different combinations using a grid search. The best parameters were considered as the ones that reached the best average accuracy across datasets, and are shown in Table II.

TABLE II: Best parameters found for the compared approaches.

Algorithm	Parameters	Grid Search	Synthetic	Real
IGMM-CD	Sigma_ini	[0.5, 1, 2, 5, 10]	0.05	10
	Cver	-	0.01	=
	T	[1, 5, 7, 9, 13]	13	=
Dynse	D	-	25	=
	m	[50, 100, 200, 300, 400]	50	=
	M	-	100	=
	k	-	5	=
	CE	-	A Priori	=
	PE	-	Age Based	=
	BC	-	Gaussian Naive Bayes	=
GMM-VRD	m	[50, 100, 200, 300, 400]	50	200
	EM it.	-	10	=
	kmax	[2, 3, 5, 7, 9]	2	=
	kDN	-	5	=
	Detector	-	ECDD	=
	c	-	1	=
	w	-	0.5	=

For the IGMM-CD, the parameter $Cver$ does not have an important impact on GMMs according to [7]. For the Dynse, due to the large number of parameters only the most important parameters have been experimented, the others were fixed to the default configurations defined by [11]. The A priori method was used as classification engine (CE), since it was the best technique of Dynamic Classifier Selection (DCS) described in [11]. As the pruning technique (PE) was used the default Age-Based method, which excludes the oldest classifier from the pool. For comparison purposes, we use as Base Classifier (BC) a similar classifier to our approach, that is Gaussian Naive Bayes.

For the synthetic datasets, 30 different data streams were generated to evaluate the predictive performance of the approaches. For real datasets, a modified version of cross-validation for data streams [19] was used. It consists in leaving out an observation from every 30 observations to construct the training stream. That is, in both cases 30 runs for each approach were executed.

D. Parameters Sensitivity Analysis

To analyse the impact of parameters on the predictive performance of the compared approaches we varied its most important parameters for each synthetic dataset using a grid search according the Table II. The synthetic datasets were chosen for this analysis because they enable a better understanding about the behavior of the approaches since it is possible to observe each type of drift existing in the data stream, unlike the real datasets.

TABLE III: Average accuracy for approaches compared.

Datasets	GMM-VRD	IGMM-CD	Dynse-priori
Circles	**0.787 (0.007)**	0.609 (0.033)	0.741 (0.033)
Sine1	**0.817 (0.004)**	0.536 (0.023)	0.603 (0.090)
Sine2	**0.722 (0.006)**	0.538 (0.024)	0.584 (0.073)
Virtual 5 changes	**0.815 (0.003)**	0.755 (0.003)	0.792 (0.004)
Virtual 9 changes	**0.845 (0.005)**	0.804 (0.007)	0.805 (0.007)
SEA	0.779 (0.008)	0.667 (0.008)	**0.795 (0.004)**
SEARec	0.777 (0.004)	0.674 (0.004)	**0.801 (0.003)**
PAKDD	**0.708 (0.009)**	0.420 (0.001)	0.462 (0.002)
ELEC	0.685 (0.008)	**0.783 (0.001)**	0.651 (0.001)
NOAA	**0.702 (0.016)**	0.604 (0.007)	0.660 (0.002)

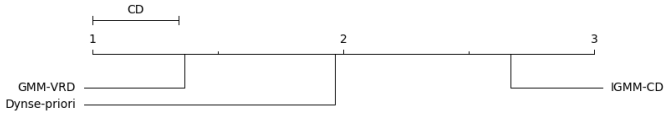


Fig. 5: Friedman ranking from left to right. Any pair of approaches whose distance between them is larger than CD are considered to be different.

VI. RESULTS AND DISCUSSION

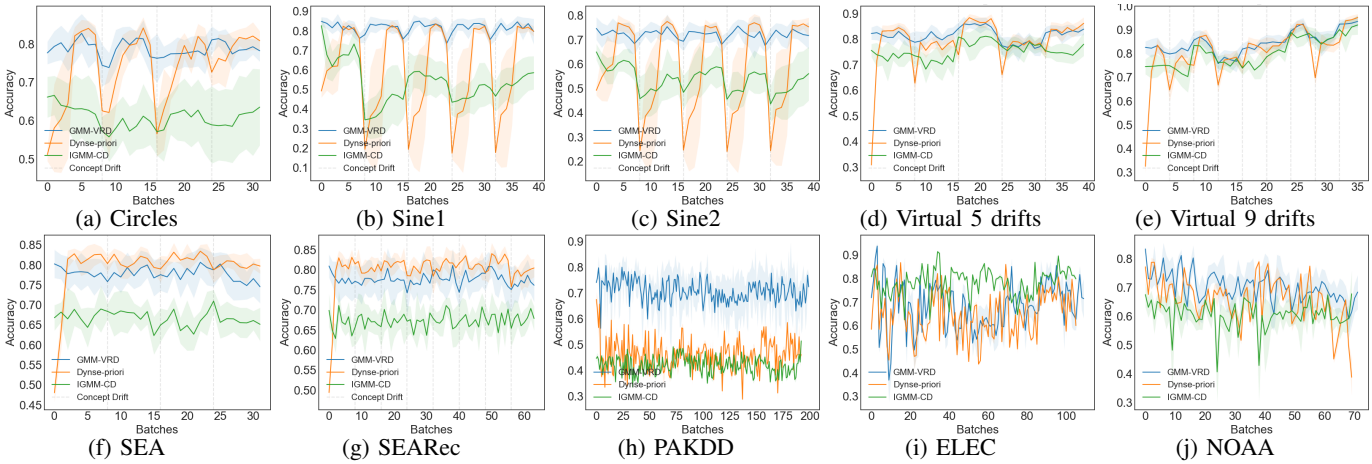
A. Comparison of Predictive Performance

The results of the comparison of the approaches are presented in Table III. The proposed method (GMM-VRD) obtained the best results in 5 out of 7 synthetic datasets, and in 2 out of 3 real datasets. Friedman detects significant differences in average accuracy with a p-value of 7.51e-18. The Nemenyi post-tests are shown in Figure 5. According to the test, the proposed method's average accuracy was significantly better than that of the other approaches.

To better understand the reason for the better performance of the proposed method we illustrate the behavior of the approaches using the accuracy over time in Figure 6.

In the datasets with abrupt drifts (Figures 6b and 6c) we note that the proposed method has less degradation compared to the other approaches. One of the reasons for this is that the model is constantly updated due to its mechanism of adaptation to virtual drifts. Different from GMM-VRD, Dynse has a sharp drop in accuracy upon abrupt drifts. This is because Dynse still does not have a classifier trained on this concept. Once a new classifier is trained on the new concept, its predictive performance increases again. Since IGMM-CD does not have a mechanism to recycle its knowledge more quickly, it tends to accumulate obsolete Gaussians which tend to decrease its performance.

When analyzing the accuracy of the approaches on the virtual datasets (Figures 6d and 6e), we observe that the proposed GMM-VRD obtained good results, presenting less accuracy degradation than the other approaches, highlighting the benefits of its virtual drift handling mechanism. We also observe that IGMM-CD obtained more competitive performance compared to the other approaches than for the abrupt



(a) Circles (b) Sine1 (c) Sine2 (d) Virtual 5 drifts (e) Virtual 9 drifts

(f) SEA (g) SEARec (h) PAKDD (i) ELEC (j) NOAA

Fig. 6: Mean of accuracy over time for all methods on each dataset. The standard deviation is represented by shadow lines of the same color. Each point represents the accuracy for a batch with 250 observations.

drifts discussed in the previous paragraph. Its procedure of creating new Gaussians accommodates well the drifts existing in the virtual drifts datasets.

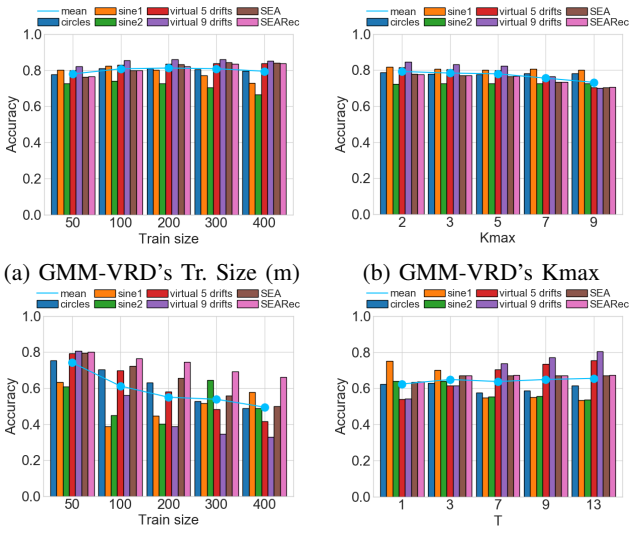
In the datasets 6f and 6g, we observe that Dynse obtained the best results. This is explained by the fact that the drifts are gradual, not offering so much degradation to the models. Therefore, Dynse, which maintains a pool of models and uses a DCS technique to choose the best classifier for each test instance, is an adequate strategy. Although GMM-VRD's accuracy does not degrade with drifts, a single classifier generally tends to perform less well than a pool.

Finally, when analyzing the real datasets, we observe that the proposed approach GMM-VRD performed well for PAKDD (Figure 6h). One of the reasons for this is the complexity of the data and drifts of this dataset. As GMM-VRD is able to model complex distributions and always recycles Gaussians when needed, it tends to be accurate. The other two approaches, which do not have this capability, tend to perform poorly. On the other hand, in the ELEC dataset (Figure 6i), IGMM-CD performed best. The ELEC dataset has gradual changes happening often [21]. This favors IGMM-CD, due to its ability to create Gaussians faster. In the NOAA dataset (Figure 6j), Dynse's and IGMM-CD's accuracy declined at certain points, demonstrating less robustness to drifts.

B. Parameter Sensitivity Analysis

Figure 7 presents the average accuracy of each approach on each dataset, when varying their parameters.

For GMM-VRD, the train size parameter (Figure 7a) 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. 7b) 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.



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

(c) Dynse's Batch Size (m) (d) IGMM-CD's T

Fig. 7: 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.

For Dynse (Figure 7c), we observe that if the size of the training batch is increased, the system tends to lose performance. Dynse depends on the quick creation of the classifiers to be able to adapt to drifts. A larger batch means that it takes longer time to create new classifiers, hindering predictive performance in non-stationary environments. Besides that, the Dynse is more sensitive to this parameter than GMM-VRD was to its parameters m and $Kmax$, given the larger changes in accuracy obtained when varying m.

For IGMM-CD (Figure 7d), the more Gaussians are created, the faster it adapts to concept drifts. Therefore, larger T values allowed it to recycle its knowledge faster, leading to better predictive performance.

VII. CONCLUSION

We proposed an approach called Gaussian Mixture Model for Dealing With Virtual and Real Concept Drifts (GMM-VRD). Our experiments show that GMM-VRD obtained the best overall accuracy across datasets because of the different strategies used for virtual and real drifts. We observed that the proposed approach performed less well than the others in terms of accuracy over time in the datasets SEA and SEARec. Despite the fact that the proposed approach sometimes performed less well, its stability across datasets was higher – the other approaches sometimes performed very well and sometimes very poorly.

We also investigated the impact of parameters in the approaches's average accuracies. In particular, for GMM-VRD we showed that: (i) The use of many Gaussians can represent complex distributions but can lead to overfitting. (ii) The use of more instances to train can improve the model stability increasing the accuracy over time.

Future work includes a study of GMM-VRD on performance critical applications; an analysis of GMM-VRD's time complexity; other virtual drift adaptation mechanisms; an approach to create Gaussians more quickly; a study using other drift detection methods; a pool to store old Gaussians for reuse on similar concepts; and an approach to deal with real drifts by excluding only the more degraded Gaussians.

ACKNOWLEDGMENT

The authors would like to thank CNPq and FACEPE (Brazilian Research Agencies) for their financial support. Leandro Minku was supported by EPSRC Grant No. EP/R006660/1.

REFERENCES

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

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

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

[4] P. M. Gonçalves Jr, S. G. de Carvalho Santos, R. S. Barros, and D. C. Vieira, "A comparative study on concept drift detectors," *Expert Systems with Applications*, vol. 41, no. 18, pp. 8144–8156, 2014.

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

[6] T.-W. Lee, M. S. Lewicki, and T. J. Sejnowski, "Ica mixture models for unsupervised classification of non-gaussian classes and automatic context switching in blind signal separation," *IEEE TPAMI*, vol. 22, no. 10, pp. 1078–1089, 2000.

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

[8] S. W. Choi, J. H. Park, and I.-B. Lee, "Process monitoring using a gaussian mixture model via principal component analysis and discriminant analysis," *Computers & chemical engineering*, vol. 28, no. 8, pp. 1377–1387, 2004.

[9] T. Chen and J. Zhang, "On-line multivariate statistical monitoring of batch processes using gaussian mixture model," *Computers & chemical engineering*, vol. 34, no. 4, pp. 500–507, 2010.

[10] G. Ditzler and R. Polikar, "Semi-supervised learning in nonstationary environments," in *IJCNN, The 2011 IJCNN on*. IEEE, 2011, pp. 2741–2748.

[11] P. R. Almeida, L. S. Oliveira, A. S. Britto Jr, and R. Sabourin, "Adapting dynamic classifier selection for concept drift," *Expert Systems with Applications*, vol. 104, pp. 67–85, 2018.

[12] T. K. Moon, "The expectation-maximization algorithm," *IEEE Signal processing magazine*, vol. 13, no. 6, pp. 47–60, 1996.

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

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

[15] G. H. Oliveira, R. C. Cavalcante, G. G. Cabral, L. L. Minku, and A. L. Oliveira, "Time series forecasting in the presence of concept drift: A pso-based approach," in *ICTAI, 2017 IEEE 29th International Conference on*. IEEE, 2017, pp. 239–246.

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

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

[18] "Real datasets with concept drift. [online]," *https://en.wikipedia.org/wiki/Concept_drift#Real*.

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

[20] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine learning research*, vol. 7, no. Jan, pp. 1–30, 2006.

[21] I. Zliobaite, "How good is the electricity benchmark for evaluating concept drift adaptation," *arXiv preprint arXiv:1301.3524*, 2013.