Clustering of Data Streams With Dynamic Gaussian Mixture Models: An IoT Application in Industrial Processes

Javier Diaz-Rozo, Concha Bielza, and Pedro Larrañaga

***Abstract*—In industrial Internet of Things applications with sensors sending dynamic process data at high speed, producing actionable insights at the right time is challenging. A key problem concerns processing a large amount of data, while the underlying dynamic phenomena related to the machine is possibly evolving over time due to factors, such as degradation. This makes any actionable model become obsolete and necessary to be updated. To cope with this problem, in this paper we propose a new unsupervised learning algorithm based on Gaussian mixture models called Gaussian-based dynamic probabilistic clustering (GDPC) mainly based on integrating and adapting three well known algorithms for use in dynamic scenarios: the expectation-maximization (EM) algorithm to estimate the model parameters and the Page-Hinkley test and Chernoff bound to detect concept drifts. Unlike other unsupervised methods, the model induced by the GDPC provides the membership probabilities of each instance to each cluster. This allows to determine, through a Brier score analysis, the robustness of the instance assignment and its evolution each time a concept drift is detected. Also, the algorithm works with very little data and significantly less computing power being able to decide whether (and when) to change the model. The algorithm is tested using synthetic data and data streams from an industrial testbed, where different operational states are automatically identified, giving good results in terms of classification accuracy, sensitivity, and specificity.**

***Index Terms*—Concept drift, data stream, dynamic clustering, Gaussian mixture models (GMM), industrial Internet of Things (IIoT).**

I. Introduction

NOWADAYS, Internet of Things (IoT) has opened a wide range of applications, where sensor data and other contextual data combined with computational models are able to produce actionable insights, which can be used as new control or monitoring systems and even new services [1]. Moreover, the application of these technologies to the industry, called industrial IoT (IIoT) [2] presents new opportunities to reduce the downtime of the machines and increase their availability, mainly channeled through predictive maintenance, where the failures need to be predicted as soon as possible [3].

Machine learning is highly used within this context. Algorithms are highly applicable to solve specific predictive maintenance problems, such as the remaining useful life (RUL) for a specific asset, failure diagnosis and prognosis, process optimization, testing, visual inspection, and quality control, among others. As a result, models that are able to predict quality deviations or failures during industrial processes can be obtained. Nevertheless, almost all of them are based on the analysis of datasets, where large and complex data storage and strategies are needed. Industrial problems need an actionable insight at the required time, e.g., to avoid further damage when a failure has occurred in a machine [4]. However, algorithm capabilities are limited by their own complexity and also by the computing power.

Additionally, this type of algorithms relies on the assumption that those models are stable throughout the industrial process (e.g., machining, painting, welding, etc.), basically meaning that the boundary conditions (i.e., degradation, raw material, tooling, temperature, etc.) of the specific problem are invariant over time. This assumption is not held in most industrial applications, where variations of boundary conditions are common across productive assets. Data heterogeneity makes real-time data streams analysis difficult to perform [5]. Therefore, we describe techniques that operate with adaptive window size as a possible solution for improving performance. In this case, data stream analysis is an important tool to detect these variations and the potential implied model in dynamic environments [6]. Machine learning algorithms can then be adapted to this setting.

In clustering, data stream analysis is still under development and limited to specific algorithm types: density- and distance-based [7], [8]. In this survey about data stream clustering, the most relevant algorithms are based on the following.

1) The k-means algorithm, e.g., BIRCH [9], CluStream [10], DGClust [11], StreamKM++ [12], scalable k-means [13], single-pass k-means [14], and SWClustering [15].
2) Hierarchical clustering, e.g., ODAC [16].

Manuscript received November 30, 2017; revised April 10, 2018 and April 26, 2018; accepted May 19, 2018. Date of publication May 24, 2018; date of current version November 14, 2018. This work was supported in part by the Spanish Centre for the Development of Industrial Technology through LearnIIoT Project under Grant IDI-20180156, in part by the Spanish Ministry of Economy and Competitiveness under Project TIN2016-79684-P, in part by the Regional Government of Madrid under Project S2013/ICE-2845-CASICAM-CM, and in part by the Fundación BBVA Grants to Scientific Research Teams in Big Data 2016. *(Corresponding author: Javier Diaz-Rozo.)*

J. Diaz-Rozo is with Aingura IIoT, 20870 Elgoibar, Spain, and also with the Department of Artificial Intelligence, Technical University of Madrid, 28660 Madrid, Spain (e-mail: jdiaz@ainguraiiot.com).

C. Bielza and P. Larrañaga are with the Department of Artificial Intelligence, Technical University of Madrid, 28660 Madrid, Spain.

Digital Object Identifier 10.1109/JIOT.2018.2840129

3) DBSCAN, e.g., D-Stream [17] and DenStream [18].
4) Hybrid algorithms, e.g., ClusTree [19], which is a hybrid of *k*-means and DBSCAN.

In all these approaches the underlying algorithm, i.e., *k*-means, hierarchical and DBSCAN, provides a crisp assignment of instances to clusters without more information about the data and how it is assigned to each cluster. However, Gaussian mixture model (GMM)-based clustering [20] adds the mixture model itself, the posterior probability that an instance has to be assigned to each component (an instance could belong to all components with different probabilities) or the possibility of evaluating the strength of the instance assignment to each component via the Brier score [21].

In the context of real engineering processes, such as machining operation data, [22] showed that GMMs had the ability to clearly discriminate what is happening inside a process much better that the *k*-means and hierarchical agglomerative algorithms which assume a static picture of the problem, where the variation of clusters (shape, number and position) over time was not taken into account. In this case, partition algorithms, such as *k*-means, tend to cluster based on underlying variables, which are not included in the analysis, such as angular speed, which do not leverage any new knowledge about the analyzed process. Moreover, agglomerative algorithms perform similarly.

From this point of view, data analysis performed with GMMs has a proven ability to leverage knowledge discovery about the engineering process, giving more information than others. However, the expressiveness of probability-based clustering algorithms is obscured by their inability, to the best of our knowledge, to deal with data streams. For example, GMMS are used as a simple method for dealing with data streams in the absence of concept drift [23]. Additionally, GMMs are usually applied to support other models, such as the evolving Bayesian network capable of dealing with data streams [24]. Nevertheless, the associated distribution parameters are estimated offline, running the EM algorithm on historical data.

A concept drift is defined in [25] as a change over time in the shift of relation between input data and the target variable. This change could be triggered by specific changes in behavior, such as degradation, extremely common in many industrial environments and helpful for early failure detection. It is also useful for detecting sensor failure and degradation [26]. As data streams and the process dynamics are an important consideration for the industry, the work in this paper is driven by the need for GMM-based probabilistic clustering, which is capable of dealing with the dynamic evolution and drifts of the industrial processes, providing a new analytics tool for IIoT.

As one of the objectives of this paper is to detect degradation of the process, a new GMM-based clustering algorithm adapted to dynamic environments called Gaussian-based dynamic probabilistic clustering (GDPC) is proposed. Based on a continuous stream of data, GDPC provides the probabilities of each instance being a member of each cluster. A measure of the cluster assignment robustness and a procedure to detect when data are not well represented by the model, i.e., there is a concept drift. Section II explains the steps of the GMM-based dynamic algorithm from training to component assignment. In Section III, a performance assessment with different parameters is carried out with experimentation to study their influence. In Section IV, a model for parameter selection for high performance is developed to provide the algorithm with real applicability. In Section V, the capacity of the algorithm to assign instances to each component during data stream monitoring is studied. This is in order to assess the ability to increase the adjustment level of the instance to each Gaussian mixture component by updating the Gaussian mixtures model. In Section VII, the algorithm is tested using real data from an industrial testbed and, in Section VIII, the conclusions and further work are given.

II. Methodology

Inspired by the general data stream algorithm pipeline, described by [10] and [27], the following steps are defined.

1) GMM offline training with an initial dataset $\mathscr{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ of size N, where each instance has d components, $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{id})$ (see Section II-A).
2) Fitting test of new incoming instances from the data stream to the current model trained in step 1 or 7, measured by the mean log-likelihood (see Section II-B).
3) Outlier detection (see Section II-C).
4) Concept drift detection with a cumulative value of outliers found, detect if there is a concept drift (see Section II-D).
5) If a concept drift is detected go to step 7, otherwise continue with step 6 for clustering.
6) If no concept drift is detected, assign the instance to each component of the GMM depending on its posterior probability (see Section II-E).
7) If concept drift is found, the model is readjusted using the last Chernoff bound window $\mathscr{D}' = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ of size n (see Section II-D).

The sequence of the algorithm steps is described in Fig. 1. A detailed description of these steps is given in the following sections.

A. GMM Offline Training

This step may be considered *static* since the algorithm must be trained with a data batch of size N. The idea is to create an initial GMM from these data to be used as reference.

Assuming the density in the kth cluster is given by $f_k = (\boldsymbol{x}; \boldsymbol{\theta}_k)$ the GMM model is given by

$$f(\boldsymbol{x}; \boldsymbol{\Psi}) = \sum_{k=1}^{K} \pi_k f_k(\boldsymbol{x}; \boldsymbol{\theta}_k) \tag{1}$$

with parameters

$$\boldsymbol{\Psi} = (\pi_1, \ldots, \pi_K, \theta_1, \ldots, \theta_K)$$

where π_k are the specific weights of each component k within the mixture, $\theta_k = (\mu_k, \Sigma_k)$ with $k = 1, \ldots, K$ and $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$

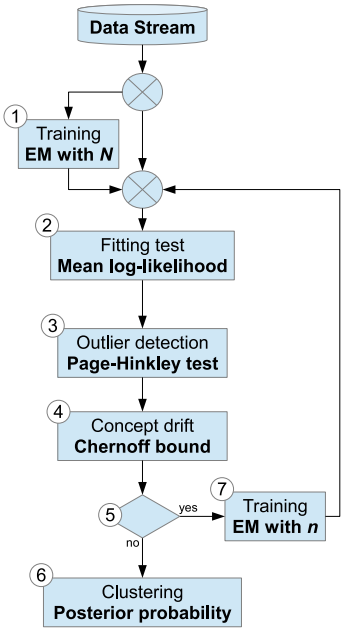


Fig. 1. Algorithm scheme.

are the vector of means and the covariance matrix of each component k, respectively.

The unknown parameter $\boldsymbol{\Psi}$ is estimated by maximum likelihood method. This estimation can be approached using the EM algorithm [20], [28]. The EM algorithm is an iterative method that has two main steps: 1) expectation or E-step, where a function for the expectation of the log-likelihood is created and 2) maximization or M-Step, where the parameters are computed maximizing the expected log-likelihood. These steps are summarized below.

1) E-Step: Having the likelihood for $\boldsymbol{\Psi}$ defined by

$$L(\boldsymbol{\Psi}) = \prod_{i=1}^{N}\left(\sum_{k=1}^{K}\pi_k f_k(\boldsymbol{x}_i;\boldsymbol{\theta}_k)\right)$$

then, the log-likelihood is

$$\begin{aligned}\log L(\boldsymbol{\Psi}) &= \sum_{i=1}^{N}\log\left(\sum_{k=1}^{K}\pi_k f_k(\boldsymbol{x}_i;\boldsymbol{\theta}_k)\right)\\ &= \sum_{k=1}^{K}\sum_{i=1}^{N} z_{ki}\{\log\pi_k + \log f_k(\boldsymbol{x}_i;\boldsymbol{\theta}_k)\}\end{aligned} \tag{2}$$

where z_{ki} contains 0 or 1 depending on the assignment of each input instance i to each component. Its corresponding random variable Z_{ki} is defined as

$$Z_{ki} = \begin{cases} 1, & \text{if } \boldsymbol{x}_i \text{ is assigned to component } k\\ 0, & \text{otherwise.}\end{cases}$$

The value of Z_{ki} is estimated by the E-Step at each iteration.

The subsequent iterations for E-Step only require finding the posterior probability that the ith instance is assigned to the kth component, i.e., $\tau_k^{(t)}(\boldsymbol{x}_i;\boldsymbol{\Psi}^{(t)})$. Therefore,

$$\begin{aligned}\mathbb{E}_{\boldsymbol{\Psi}^{(t)}}\{Z_{ki}|\mathscr{D}\} &= P_{\boldsymbol{\Psi}^{(t)}}\{Z_{ki}=1|\mathscr{D}\}\\ &= \tau_k^{(t)}\left(\boldsymbol{x}_i;\boldsymbol{\Psi}^{(t)}\right)\end{aligned}$$

where

$$\tau_k^{(t)}\left(\boldsymbol{x}_i;\boldsymbol{\Psi}^{(t)}\right) = \frac{\pi_k^{(t)} f_k\left(\boldsymbol{x}_i;\boldsymbol{\theta}_k^{(t)}\right)}{f\left(\boldsymbol{x}_i;\boldsymbol{\Psi}^{(t)}\right)}. \tag{3}$$

For a GMM, (3) is defined as

$$\tau_k^{(t)}\left(\boldsymbol{x}_i;\boldsymbol{\Psi}^{(t)}\right) = \frac{\pi_k^{(t)}\mathcal{N}\left(\boldsymbol{x}_i|\boldsymbol{\mu}_k^{(t)},\boldsymbol{\Sigma}_k^{(t)}\right)}{\sum_{j=1}^{K}\pi_j^{(t)}\mathcal{N}\left(\boldsymbol{x}_i|\boldsymbol{\mu}_j^{(t)},\boldsymbol{\Sigma}_j^{(t)}\right)}.$$

Consequently, using (2) and (3) it is possible to define the conditional expectation of $\log L(\boldsymbol{\Psi})$ given $\mathscr{D}$ as

$$\begin{aligned}\mathbb{E}_{\boldsymbol{\Psi}^{(t)}}\{\log L(\boldsymbol{\Psi})|\mathscr{D}\} &= Q\left(\boldsymbol{\Psi};\boldsymbol{\Psi}^{(t)}\right)\\ &= \sum_{k=1}^{K}\sum_{i=1}^{N}\tau_k^{(t)}\left(\boldsymbol{x}_i;\boldsymbol{\Psi}^{(t)}\right)\\ &\quad\times\{\log\pi_k + \log f_k(\boldsymbol{x}_i;\boldsymbol{\theta}_k)\}.\end{aligned}$$

Once the expectation is found at iteration t, the M-Step is applied.

2) M-Step: The $t+1$ iteration for the M-Step searches for the maximization of $Q(\boldsymbol{\Psi};\boldsymbol{\Psi}^{(t)})$ with respect to $\boldsymbol{\Psi}$ in order to update the value of $\boldsymbol{\Psi}^{(t+1)}$. The estimated specific weights, $\pi_k^{(t+1)}$, are found independently of the vector of unknown parameters $\boldsymbol{\theta}^{(t+1)}$ by means of

$$\pi_k^{(t+1)} = \frac{1}{N}\sum_{i=1}^{N}\tau_k^{(t)}\left(\boldsymbol{x}_i;\boldsymbol{\Psi}^{(t)}\right), \quad k=1,\ldots,K.$$

To obtain $\boldsymbol{\theta}^{(t+1)}$ we solve

$$\sum_{k=1}^{K}\sum_{i=1}^{N}\tau_k^{(t)}\left(\boldsymbol{x}_i;\boldsymbol{\Psi}^{(t)}\right)\frac{\partial\log f_k(\boldsymbol{x}_i;\boldsymbol{\theta}_k)}{\partial\boldsymbol{\theta}_k} = 0. \tag{4}$$

As the likelihood increases monotonically with the iterations of the EM algorithm, a standard stopping criterion is to cease when

$$L\left(\boldsymbol{\Psi}^{(t+1)}\right) - L\left(\boldsymbol{\Psi}^{(t)}\right)$$

is small enough to indicate the convergence of the sequence of parameters $\{\boldsymbol{\Psi}^{(t)}\}$.

For GMM

$$\begin{aligned}f_k(\boldsymbol{x}_i;\boldsymbol{\theta}_k) &= f_k\left(\boldsymbol{x}_i;\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k\right)\\ &= (2\pi)^{-\frac{d}{2}}|\boldsymbol{\Sigma}_k|^{-\frac{1}{2}}\\ &\quad\times\exp\left\{-\frac{1}{2}\left(\boldsymbol{x}_i-\boldsymbol{\mu}_k\right)^T\boldsymbol{\Sigma}_k^{-1}\left(\boldsymbol{x}_i-\boldsymbol{\mu}_k\right)\right\}\end{aligned} \tag{5}$$

where parameters $\boldsymbol{\mu}_k^{(t+1)}$ and $\boldsymbol{\Sigma}_k^{(t+1)}$ $(k=1,\ldots,K)$ are estimated using (4)

$$\boldsymbol{\mu}_k^{(t+1)} = \frac{\sum_{i=1}^{N}\tau_k^{(t+1)}\boldsymbol{x}_i}{\sum_{i=1}^{N}\tau_k^{(t+1)}\left(\boldsymbol{x}_i;\boldsymbol{\Psi}_k^{(t)}\right)}, \quad k=1,\ldots,K$$

and

$$\boldsymbol{\Sigma}_k^{(t+1)} = \frac{\sum_{i=1}^{N} \tau_k^{(t+1)}\left(\boldsymbol{x}_i; \boldsymbol{\Psi}_k^{(t)}\right)\left(\boldsymbol{x}_i - \boldsymbol{\mu}_k^{(t+1)}\right)\boldsymbol{x}_i - \boldsymbol{\mu}_k^{(t+1)}\Big)^T}{\sum_{i=1}^{N} \tau_k^{(t+1)}\left(\boldsymbol{x}_i; \boldsymbol{\Psi}_k^{(t)}\right)}, \quad k = 1, \ldots, K.$$

Having the reference model (induced from initial dataset $\mathscr{D}$), a fitting test is done with new instances coming from the data stream.

B. Fitting Test

The fitting test checks if the new incoming data stream instance is well represented by the current GMM. To measure this, we use the mean log-likelihood criterion [29], which is calculated from (2) and (5)

$$\overline{\log L}(\boldsymbol{\Psi}) = \frac{1}{N}\sum_{i=1}^{N} \log\left(\sum_{k=1}^{K} \pi_k f_k\left(\boldsymbol{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)\right) \tag{6}$$

and compare it with the log-likelihood $\log L_w(\boldsymbol{\Psi})$ of the new instance $\boldsymbol{x}_w$. A significant small value of $\log L_w(\boldsymbol{\Psi})$ in comparison with $\overline{\log L}(\boldsymbol{\Psi})$ indicates that the new data is badly represented by the current mixture model. Next, the values of $\overline{\log L_i}(\boldsymbol{\Psi})$ are used to detect outliers (see the following section).

C. Outlier Detection

Once the fitting test result for the data stream is found, i.e., the value of (2), it is necessary to know the value of $\overline{\log L}(\boldsymbol{\Psi})$ that shows if the instance $\boldsymbol{x}_j$ is correctly represented by the model or not. When an instance is not represented by the current GMM, it is labeled as an outlier, and it is used afterward in the concept drift detection step (Section II-D).

Sebastiao and Gama [27] and Borchani *et al.* [30] successfully applied the Page-Hinkley test [31], sequential analysis of the data stream, taking the sample variance of the analyzed data as the key measure. As explained by [32], the test associates the sample variance with the normal behavior of the data stream. Therefore, any disturbance, such as degradation, in this variance is associated with an abrupt change in the data stream behavior. In this case, the null hypothesis of the test is that the instance $\boldsymbol{x}_j$ is correctly represented by the GMM.

To perform the Page-Hinkley test, as new instances arrive from the data stream process, $\overline{\log L_i}(\boldsymbol{\Psi})$ is calculated from (6) and stored on a vector defined by

$$\overline{\boldsymbol{\log L}}(\boldsymbol{\Psi}) = \left\{\overline{\log L_1}(\boldsymbol{\Psi}), \ldots, \overline{\log L_n}(\boldsymbol{\Psi})\right\} \tag{7}$$

where n is explained by [33]–[35], as an adaptive window size bounded by

$$n \leq \frac{3(1+\epsilon)}{(1-\epsilon)\epsilon^2 p} \ln\left(\frac{2}{\phi}\right) \tag{8}$$

where $0 < \epsilon < 1$ is the additive error bound, $0 < \phi < 2$, a constant to control of the probability of an instance to be successfully clustered and p is the probability that the instance $\boldsymbol{x}_j$ that has arrived, in this case, number of nonoutlier instances over the total available instances. The window size n changes inversely to p, meaning that if a new instance is well represented (large p) or badly represented (small p), the window size is small or large, respectively. This adaptive window helps to optimize the amount of data stored, i.e., if a new data instance fits the model, there is no need to have a large amount of historical data.

Next, the accumulated difference between mean log-likelihood and the accumulated mean value (CUM_j) is defined by

$$\text{CUM}_n = \frac{1}{n}\sum_{j=1}^{n}\left(\overline{\log L_j}(\boldsymbol{\Psi}) - \text{mean}_j\right) - \delta \tag{9}$$

where $\text{mean}_j = (1/j)\sum_{t=1}^{j} \overline{\log L_t}(\boldsymbol{\Psi})$ is the mean value of the first j components of the vector defined in (7), and δ is a positive random parameter of tolerance for the maximum magnitude of accepted changes. The way to select this value is described in Section III. Additionally, a value MAX_j is defined as

$$\text{MAX}_j = \max\left\{\text{CUM}_j, \quad j = 1, \ldots, n\right\}.$$

Therefore, threshold PH_j is defined as $PH_j = \text{MAX}_j - \text{CUM}_j$ ($j = 1, \ldots, n$), and so

$$\text{reject}_j = \begin{cases} 1, & \text{if } PH_j > \lambda \\ 0, & \text{otherwise} \end{cases}$$

where $\lambda > 0$ is the value that controls the null hypothesis, where 1 means that instance $\boldsymbol{x}_j$ is labeled as an outlier (the hypothesis is rejected). Therefore, a vector reject_j of size n stores 1 or 0 if instance $\boldsymbol{x}_j$ is an outlier or not, respectively.

D. Concept Drift Detection

The aim of this step is to differentiate between outliers produced by noise in the data stream (data acquisition system uncertainty) and those related to the GMM, where a group of instances, whose size is larger than a defined threshold, no longer fit properly. The latter case is defined as concept drift.

In order to detect a concept drift, we adapt the approach proposed by [35] to apply Chernoff bounds to define the minimum number of instances s that are not outliers in a defined window, according to

$$s = \frac{3(1+\epsilon)}{\epsilon^2} \ln\left(\frac{2}{\phi}\right) \tag{10}$$

and the size of adaptive data stream windows from (8) to control the amount of stored instances for training if a concept drift is detected. The main idea of this approach is to determine the total amount of outliers needed to indicate that the GMM has expired and a new one should be learned from data. Now, if $r = \sum_{i=1}^{n}(1 - \text{reject}_i) < s$, a concept drift is detected.

As soon as this model change is detected, a new GMM is estimated from the adaptive window data using the algorithm described in Section II-A, with N defined as the lowest n, n being defined by (8). r is initialized as 0 and increases its value each time an instance is correctly represented by the GMM. p is r/n and the initial value for $p = 0$ and it is updated each cycle.

E. Instance Assignment

The next step in the GDPC algorithm is to assign an instance to each component via the posterior probability that instance $\boldsymbol{x}_i$ belongs to component k of the mixture, as reflected in (3). That is

$$z_{kj} = \begin{cases} 1, & \text{if } k = \arg\max_h \tau_h(\boldsymbol{x}_j; \boldsymbol{\Psi}) \\ 0, & \text{otherwise.} \end{cases}$$

F. Algorithm Wrap-Up

It is now necessary to wrap-up the isolated modules described in Sections II-A–II-E to provide a clear view of their operation and interactions that make up the GDPC algorithm backbone.

Once the modules are bonded together, the algorithm is ready to receive a data stream coming from a specific process and initialize the parameters related with each module: δ, λ, ϵ, ϕ, and N. These parameters are selected following the approach presented in Section IV. Using these values, the minimum allowed window size, s, is calculated using (10). Other parameters, such as the probability, p, and the number of instances without outliers, r, are initialized to zero.

After initialization, a training data set $\mathscr{D}$ is filled with data stream until the number of instances is equal to the parameter N. Then, the initial GMM can be trained with $\mathscr{D}$ using the EM algorithm to estimate the Gaussian mixture parameters. From this step, the algorithm starts to analyze data stream, processing one instance at a time using the GMM as the statistical summary that represents the analyzed data.

The data stream analysis start by finding the adaptive window of size n using Chernoff bound that works as a temporary buffer. It stores the newest n instances incoming from the data stream. Meanwhile, each data stream instance is checked against the current GMM using the mean log-likelihood. This mean value is monitored by the Page-Hinkley test that searches for outliers. If this hypothesis test is not rejected, the instance is correctly represented by the GMM and it is recorded by updating r.

The next mainly compares r against s. If the number of correctly represented instances r is larger than s, a concept drift turning point in the data stream is detected. In this case, a new GMM model is fitted using the newest data stored in the adaptive window and the process continues with the next instance from the data stream. If r remains smaller than s (i.e., no concept drift) the probability, p, is updated with r/n and the instance is assigned to a GMM component using its posterior probability and continues with the next instance from the data stream.

The complete process is summarized in Algorithm 1.

Algorithm 1 GDPC

Require: Data stream
Require: N
Require: δ and λ
Require: Initialize s with $0 < \epsilon < 1$ and $0 < \phi < 2$
Require: Initialize $p = 0$ and $r = 0$
while $i <= N$ **do**
  Instance is assigned to training data set: $\mathscr{D} \leftarrow \boldsymbol{x}_i$
function GMM TRAINING($\mathscr{D}$)
  Initial GMM model is trained with $\mathscr{D}$ using EM algorithm
while data stream **do**
  Initialize window size n with Eq. (8)
  function FITTING TEST($\boldsymbol{x}_i$)
    Measure $\overline{\log L}(\boldsymbol{x}_i)$
  function OUTLIER DETECTION($\overline{\log L}(\boldsymbol{x}_i)$)
    CUM_j is calculated using Eq. (9)
    Add the CUM_j value to the vector MAX_j
    Start the Page-Hinkley test with $PH_i = MAX_i - CUM_i$
    if $PH_i > \lambda$ **then**
      An outlier is detected
      $outliers+ = 1$
      $r \leftarrow (1 - outliers)$
  function CONCEPT DRIFT DETECTION(r, $\mathscr{D} \leftarrow \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$)
    Compute s with Eq. (10)
    if $r < s$ **then**
      A concept drift is detected: train GMM model with $\mathscr{D}$
    else if $r > s$ **then**
      Update probability $p \leftarrow r/n$
      Assign new instance

III. DYNAMIC PROBABILISTIC CLUSTERING PARAMETER PERFORMANCE ASSESSMENT

According to the proposed approach, it is necessary to assess the influence of the parameters described below on the GDPC results so that it can serve as a configuration guide. These parameters are mostly related to the hypothesis tests that are posed, and grouped as follows.

1) GMM offline training (Section II-A).
   a) *N:* Number of initial training instances.
2) Page-Hinkley hypothesis test (Section II-C).
   a) λ*:* Positive threshold that defines the value from which the null hypothesis of an instance $\boldsymbol{x}_i$ to be correctly represented by the GMM is rejected.
   b) δ*:* Positive tolerance to changes that are to be absorbed (9).
3) Chernoff bound (Section II-D).
   a) ϵ: Additive error bound ($0 < \epsilon < 1$).
   b) ϕ*:* Constant to control the probability of a instance to be successfully clustered ($0 < \phi < 2$).

A. Experimental Datasets

To study the parameter effect on the concept drift detection capability, synthetic datasets are created by simulating random values from a mixture of Gaussian distributions. As the principal objective of these experiments is to calibrate the GDPC's concept drift detection capability, it is critical to know the concept drift locations within the data stream beforehand. This outputs a reference data stream, with a predefined variety

TABLE I
CHARACTERISTICS OF THE SUBSETS WITH SIMULATED INSTANCES

Subset No.	l_{subset}	π_l	μ_l	Σ_l
0	15,000	π	μ	Σ
1	7,000	π	$0.75 \cdot \mu$	$0.75 \cdot \Sigma$
2	10,000	π	$1.25 \cdot \mu$	$1.25 \cdot \Sigma$
3	5,000	π	$noise_{\sigma=0.25} \cdot \mu$	$noise_{\sigma=0.25} \cdot \Sigma$
4	7,000	π	$0.90 \cdot \mu$	$0.90 \cdot \Sigma$
5	10,000	π	$1.10 \cdot \mu$	$1.10 \cdot \Sigma$
6	5,000	π	$noise_{\sigma=0.10} \cdot \mu$	$noise_{\sigma=0.10} \cdot \Sigma$
7	7,000	π	$0.95 \cdot \mu$	$0.95 \cdot \Sigma$
8	10,000	π	$1.05 \cdot \mu$	$1.05 \cdot \Sigma$
9	5,000	π	$noise_{\sigma=0.05} \cdot \mu$	$noise_{\sigma=0.05} \cdot \Sigma$

of concept drifts, useful for comparing the algorithm results in terms of concept drift detection and its detection delay.

Therefore, data streams are created concatenating different subsets of instances each of which is simulated from GMMs with different parameter value combinations. Due to the variation of the parameters, different GMMs along the dataset simulate the concept drifts.

In the synthetic model with $K = 3$ components for this particular scenario

$$\boldsymbol{\theta} = \left(\pi_1, \pi_2, \pi_3, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\mu}_3, \boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2, \boldsymbol{\Sigma}_3\right)$$

where $\boldsymbol{\mu}_k \quad (k = \{1, 2, 3\})$ is the parameter vector of mean values and $\boldsymbol{\Sigma}_k \quad (k = \{1, 2, 3\})$ is the parameter matrix of variances.

Different values of offsets and Gaussian noise, selected to artificially produce different models, are applied equally to all the parameter dimensions: offset $= \pm 5\%, \pm 10\%$, and $\pm 25\%$. A random variable is defined as noise_σ, where $\sigma = \{0.05, 0.10, 0.25\}$ when its random value is simulated with Gaussian distributions: $\mathcal{N}(0, 0.05)$, $\mathcal{N}(0, 0.10)$, and $\mathcal{N}(0, 0.25)$, respectively. Parameter magnitudes are changed in order to study detection capabilities depending on the similarity of the models, i.e., offset $= \pm 5\%$ or $\mathcal{N}(0, 0.05)$ means that the drift is more similar to the original model than offset $= \pm 25\%$ or $\mathcal{N}(0, 0.25)$.

Once the parameters of the mixtures are varied, for example: using $\boldsymbol{\Sigma}_l = \text{offset} \cdot \boldsymbol{\Sigma}$, instances are simulated producing a subset of length l_{subset}. Accordingly, an original subset (without variations) plus nine subsets of instances with different lengths and variations of their parameters are generated. The description of the way parameters are modified for obtaining different subsets is shown in Table I.

The concatenation of the different subsets of instances described in Table I enables different data streams to be obtained for experimentation. In this way, there are three data streams created from these concatenations shown in Table II. Data streams have positive and negative offsets and Gaussian noise depending on the magnitude defined in Table I. These data streams have an equal number of instances, that is, $l_{\text{datastream}} = 37\,000$ with three additional concept drifts (CD_{set}).

TABLE II
SYNTHETIC DATA STREAMS

Data stream No.	Sequence of subsets (Table 1)	Concept drift location ($\boldsymbol{CD_{set}}$)
1	{0, 1, 2, 3}	{15000, 22000, 32000}
2	{0, 4, 5, 6}	{15000, 22000, 32000}
3	{0, 7, 8, 9}	{15000, 22000, 32000}

TABLE III
DESIGN OF EXPERIMENT PARAMETERS AND LEVELS

Parameters	Level 1	Level 2	Level 3	Level 4
λ	0.10	1.00	10.0	100
δ	0.01	0.10	1.00	10.0
ϵ	0.10	0.25	0.75	0.90
ϕ	0.01	0.27	0.54	0.80
N	10	100	1000	10000

B. Design of Experiments

To efficiently explore the parameter effect on model behavior, we require a fractional factorial and orthogonal design in which factors (variables for analysis) and levels (value ranges of each factor) are balanced. This type of designs are called Taguchi methods. The entry key input to select the Taguchi design is the required number of parameters. In this case, we studied five GDPC parameters: λ, δ, ϵ, ϕ, and N; each with four levels. Therefore, the design that best fits this number of parameters and levels is the orthogonal Taguchi L16 (type B) design [36] with five four-level parameters, i.e., a 4^5 design with 16 observations. Table III shows the parameters and levels to be used in the experimentation.

To study the influence of these parameters, we determine the following measures to be extracted from the experiment.

1) *Detected Concept Drifts (CD_{det}):* The amount of concepts drifts detected within a data stream.
2) *Instance Marked As Concept Drift Turning Point (n_{CD}):* The first instance, where the algorithm detects a concept drift CD_{det}, defined as a turning point.
3) *True Concept Drifts (CD_{true}):* From CD_{det}, the amount of true concept drifts measured by their location n_{CD} compared to CD_{set} in Table II.
4) *Processing Time in Seconds (t):* The time required by the algorithm to process the data stream.

From observations, performance is assessed using the figures of merit derived from the concept drift detection confusion matrix, i.e., accuracy, sensitivity, and specificity. The confusion matrix is detailed in Table IV, where a comparison between a true concept drift class (c_t) and concept drift detected by the model class (c_m) can be made.

1) *c_t Is Positive and c_m Is Positive:* The instances detected by the model are true concept drifts, i.e., CD_{true}.
2) *c_t Is Negative and c_m Is Positive:* Not all the instances detected by the model are true concept drifts, i.e., $\text{CD}_{\text{det}} - \text{CD}_{\text{set}}$.
3) *c_t Is Positive and c_m Is Negative:* Not all the instances with true concept drifts are detected, i.e., $\text{CD}_{\text{set}} - \text{CD}_{\text{true}}$.
4) *c_t Is Negative and c_m Is Negative:* The amount of instances not detected as concept drift, i.e., $N - \text{CD}_{\text{det}}$.

TABLE IV
CONFUSION MATRIX

		Truth	
		+	-
Predicted	+	CD_{true}	$CD_{det} - CD_{set}$
	-	$CD_{set} - CD_{true}$	$N - CD_{det}$

From Table IV, the figures of merit are defined as follows.

1) *Accuracy:* $[(\mathrm{CD_{true}} - \mathrm{CD_{det}} + N)/N]$.
2) *Sensitivity—True Positives (TP):* $(\mathrm{CD_{true}}/\mathrm{CD_{set}})$.
3) *Specificity—True Negatives:* $[(N - \mathrm{CD_{det}})/(N - \mathrm{CD_{set}})]$.
4) *Recall:* $[TP/(TP + FN)]$.
5) *F-Score:* $[(2x\mathrm{Recall}x\mathrm{Precision})/(\mathrm{Recall} + \mathrm{Precision})]$.

Where

1) *False Positives (FP):* $[(\mathrm{CD_{set}} - \mathrm{CD_{true}})/\mathrm{CD_{set}}]$;
2) *False Negatives (FN):* $[(\mathrm{CD_{det}} - \mathrm{CD_{set}})/(N - \mathrm{CD_{set}})]$;
3) *Precision:* $[TP/(TP + FP)]$.

C. Experimental Results

Three experimental repetitions are run with each data stream defined in Table II. The results of these experiments are shown in Tables V–VII.

In Table V, the results with data stream 1 are shown. In this case, the algorithm is able to detect the first concept drift with delays not larger than 591 instances for experiment 14 (turning point detected in instance 15 591) in the complete set of experiments. Experiments 1 and 12, where the only experiments without a correct true concept drift ($\mathrm{CD_{true}}$) detection, with only one and two detections, respectively. On the other hand, experiments 6–11 and 13–16 achieved 100% accuracy without any FPs. Therefore, 87.5% of the experiments have parameter combinations that are able to detect true concept drifts (100% sensitivity), but only 62.5% of them with 100% accuracy.

Table VI shows the results for data stream 2, and the situation is slightly different: 62.5% of the experiments have 100% sensitivity and only 50% of them have 100% accuracy (experiments 6, 9, 10, 12, and 14), meaning that fewer combinations of parameters are able to detect the predefined concept drifts. These results show that the algorithm is sensitive to the difference between GMM, simulated by the offset and noise magnitude from Table I, detected as the concept drift.

In the experiment with data stream 3 (Table VII), the results are more critical because some parameter combinations are not able to detect any concept drift, i.e., experiments 13–16. In this case, 56% of the experiments have 100% sensitivity with only 55% of them with 100% accuracy, i.e., experiments 6, 8–10, and 12. For this data stream, the experiments that could not detect any concept drift are differentiated by the parameter λ, which has the largest level (100). As a preliminary analysis for parameter selection, large levels of λ are not optimal when GMM changes (concept drifts) are small.

Regarding processing time, experiments 1–4 with $\lambda = 0.1$ from any data stream have the highest value, not always guaranteeing the concept drift detection. As a preliminary insight, $\lambda > 1$ is able to give similar results with 50% of processing time. Other experiments need around 7.21 s to process 37 000 instances independently from the data stream used.

In summary, parameter combinations 9, 10, and 12 are the only combinations able to detect concept drifts in all data streams, meaning that it is possible to use optimal parameters in the proposed algorithm. It is interesting to note that these combinations have the same $\lambda = 10$, giving some clues to understand the optimal magnitude for this parameter.

However, these are intuitive results that can help to understand the parameter influence in the algorithm performance. As a result, in Section IV a complete study, using the experimental results to build a supervised model to select parameters to improve the figure of merit, is presented.

IV. PARAMETER SELECTION FOR HIGH PERFORMANCE

From the experimentation results shown in the previous section, optimal parameter selection is studied to find the underlying common characteristics between parameter combinations 9, 10, and 12. Therefore, data from experiments is fed into traditional models for experiment result analysis, such as linear or polynomial regressions or the signal-to-noise ratio test. However, those models have not given the desired result in relation to their accuracy due to a low coefficient of determination.

One of the principal issues that could affect these underperforming models is the lack of enough data. Consequently, taking advantage of algorithm efficiency and its ability to obtain good performance with small amounts of data, naïve Bayes classifier [37] is used. This classifier is built upon the assumption of conditional independence of the predictive variables given the class, where the maximum posterior probability assignment of the class variable is calculated as prediction.

Three classifiers are designed for each figure of merit: accuracy, sensitivity, and specificity, taken as the class variables. The predictor variables are the five univariate Gaussian parameters λ, δ, ϵ, ϕ, and N.

Because class variables are continuous, the values in each experiment are categorized using three different groups.

1) *Low*, for values smaller than 50%.
2) *Average*, for values that are greater than 50% and less than 99%.
3) *High*, for values greater than 99%.

Therefore, the naïve Bayes model is defined by

$$c^* = \arg\max_c p(c|\mathbf{par}) = \arg\max_c p(c) \prod_{i=1}^{5} f(\mathrm{par}_i|c)$$

where c^* is the Bayesian classifier selected as the most probable hypothesis with the maximum *a priori* or MAP and $\boldsymbol{par} = (\lambda, \delta, \epsilon, \phi, N)$, and c is the class variable of each figure of merit previously described.

A. Parameter Ranking by Relevance

The first step in the experiment results analysis with the naïve Bayes classifier is to rank the relevance of each parameter for each figure of merit. Experimental data from Section III-C is fed into the model and the posterior probabilities for *high* value of class variable are shown

TABLE V
EXPERIMENTAL RESULTS FOR DATA STREAM 1

Test	λ	δ	ϵ	ϕ	N_{train}	t	CD_{det}	CD_{true}	n_{CD}	Accu.	Sens.	Spec.	Recall	F-score
1	0.1	0.01	0.10	0.01	10	17.27	15	1	{15546}	0.889	0.333	1.000	1.000	0.500
2	0.1	0.10	0.25	0.27	100	18.72	230	3	{15140, 22020, 32100}	0.994	1.000	0.994	0.994	0.997
3	0.1	1.00	0.75	0.54	1000	18.45	717	3	{15009, 22021, 32021}	0.981	1.000	0.981	0.981	0.990
4	0.1	10.00	0.90	0.80	10000	16.40	320	3	{15059, 22002, 32043}	0.992	1.000	0.991	0.991	0.995
5	1.0	0.01	0.25	0.54	10000	9.66	35	3	{15127, 22205, 32145}	0.999	1.000	0.999	0.999	0.999
6	1.0	0.10	0.10	0.80	1000	8.23	4	3	{15070, 22105, 32155}	1.000	1.000	1.000	1.000	1.000
7	1.0	1.00	0.90	0.01	100	8.03	3	3	{15352, 22477, 32773}	1.000	1.000	1.000	1.000	1.000
8	1.0	10.00	0.75	0.27	10	8.28	12	3	{15069, 22073, 32324}	1.000	1.000	1.000	1.000	1.000
9	10.0	0.01	0.75	0.80	100	8.17	3	3	{15026, 22030, 32061}	1.000	1.000	1.000	1.000	1.000
10	10.0	0.10	0.90	0.54	10	8.08	4	3	{15105, 22103, 32202}	1.000	1.000	1.000	1.000	1.000
11	10.0	1.00	0.10	0.27	10000	9.36	3	3	{15138, 22478, 32770}	1.000	1.000	1.000	1.000	1.000
12	10.0	10.00	0.25	0.01	1000	8.18	2	2	{15382, 22150}	0.800	0.667	1.000	1.000	0.800
13	100.0	0.01	0.90	0.27	1000	7.91	3	3	{15140, 22140, 32370}	1.000	1.000	1.000	1.000	1.000
14	100.0	0.10	0.75	0.01	10000	9.66	3	3	{15591, 22463, 32544}	1.000	1.000	1.000	1.000	1.000
15	100.0	1.00	0.25	0.80	10	9.29	3	3	{15048, 22056, 32057}	1.000	1.000	1.000	1.000	1.000
16	100.0	10.00	0.10	0.54	100	9.97	3	3	{15460, 22180, 32260}	1.000	1.000	1.000	1.000	1.000

TABLE VI
EXPERIMENTAL RESULTS FOR DATA STREAM 2

Test	λ	δ	ϵ	ϕ	N_{train}	t	CD_{det}	CD_{true}	n_{CD}	Accu.	Sens.	Spec.	Recall	F-score
1	0.1	0.01	0.10	0.01	10	24.34	19	1	{15546}	0.909	0.333	1.000	1.000	0.500
2	0.1	0.10	0.25	0.27	100	17.47	228	3	{15140, 22020, 32007}	0.994	1.000	0.994	0.994	0.997
3	0.1	1.00	0.75	0.54	1000	21.85	719	3	{15003, 22011, 32039}	0.981	1.000	0.981	0.981	0.990
4	0.1	10.00	0.90	0.80	10000	22.37	332	3	{15018, 22047, 32032}	0.991	1.000	0.991	0.991	0.995
5	1.0	0.01	0.25	0.54	10000	10.04	24	3	{15185, 22080, 32028}	0.999	1.000	0.999	0.999	0.999
6	1.0	0.10	0.10	0.80	1000	8.85	8	3	{15079, 22114, 32164}	1.000	1.000	1.000	1.000	1.000
7	1.0	1.00	0.90	0.01	100	9.39	10	2	{15909, 32839}	0.923	0.667	1.000	1.000	0.800
8	1.0	10.00	0.75	0.27	10	8.87	27	3	{15076, 22183, 32093}	0.999	1.000	0.999	0.999	0.999
9	10.0	0.01	0.75	0.80	100	7.96	4	3	{15026, 22030, 32026}	1.000	1.000	1.000	1.000	1.000
10	10.0	0.10	0.90	0.54	10	7.94	4	3	{15103, 22099, 32178}	1.000	1.000	1.000	1.000	1.000
11	10.0	1.00	0.10	0.27	10000	8.82	3	2	{15138, 32754}	0.833	0.667	1.000	1.000	0.800
12	10.0	10.00	0.25	0.01	1000	8.41	3	3	{15382, 22573, 32302}	1.000	1.000	1.000	1.000	1.000
13	100.0	0.01	0.90	0.27	1000	8.23	1	1	{32993}	0.500	0.333	1.000	1.000	0.500
14	100.0	0.10	0.75	0.01	10000	8.55	3	3	{15387, 22389, 32239}	1.000	1.000	1.000	1.000	1.000
15	100.0	1.00	0.25	0.80	10	8.93	2	1	{15048}	0.600	0.333	1.000	1.000	0.500
16	100.0	10.00	0.10	0.54	100	8.16	1	1	{15460}	0.500	0.333	1.000	1.000	0.500

TABLE VII
EXPERIMENTAL RESULTS FOR DATA STREAM 3

Test	λ	δ	ϵ	ϕ	N_{train}	t	CD_{det}	CD_{true}	n_{CD}	Accu.	Sens.	Spec.	Recall	F-score
1	0.1	0.01	0.10	0.01	10	19.46	19	1	{15546}	0.909	0.333	1.000	1.000	0.500
2	0.1	0.10	0.25	0.27	100	17.51	230	3	{15140, 22020, 32100}	0.994	1.000	0.994	0.994	0.997
3	0.1	1.00	0.75	0.54	1000	21.81	719	3	{15034, 22042, 32001}	0.981	1.000	0.981	0.981	0.990
4	0.1	10.00	0.90	0.80	10000	15.91	264	3	{15055, 32031, 32095}	0.993	1.000	0.993	0.993	0.996
5	1.0	0.01	0.25	0.54	10000	9.16	24	3	{15079, 22047, 32077}	0.999	1.000	0.999	0.999	0.999
6	1.0	0.10	0.10	0.80	1000	9.13	9	3	{15070, 22105, 32155}	1.000	1.000	1.000	1.000	1.000
7	1.0	1.00	0.90	0.01	100	10.35	7	1	{32928}	0.800	0.333	1.000	1.000	0.500
8	1.0	10.00	0.75	0.27	10	9.42	19	3	{15089, 22186, 32078}	1.000	1.000	1.000	1.000	1.000
9	10.0	0.01	0.75	0.80	100	8.60	3	3	{15142, 22040, 32159}	1.000	1.000	1.000	1.000	1.000
10	10.0	0.10	0.90	0.54	10	8.64	4	3	{15170, 22455, 32121}	1.000	1.000	1.000	1.000	1.000
11	10.0	1.00	0.10	0.27	10000	14.37	2	2	{15897, 22503}	0.800	0.667	1.000	1.000	0.800
12	10.0	10.00	0.25	0.01	1000	9.19	3	3	{15382, 22643, 32372}	1.000	1.000	1.000	1.000	1.000
13	100.0	0.01	0.90	0.27	1000	9.02	0	0	-	NaN	NaN	1.000	NaN	NaN
14	100.0	0.10	0.75	0.01	10000	9.70	0	0	-	NaN	NaN	1.000	NaN	NaN
15	100.0	1.00	0.25	0.80	10	8.73	0	0	-	NaN	NaN	1.000	NaN	NaN
16	100.0	10.00	0.10	0.54	100	9.01	0	0	-	NaN	NaN	1.000	NaN	NaN

in Table VIII. The independent probability for each class to take high, average, and low values is shown, along with the prediction accuracy as naïve Bayes validation in Table IX.

From Table VIII, the most relevant parameter, with the maximum posterior probability for high figures of merit (accuracy, sensitivity, and sensibility) is *N*. Page-Hinkley test (Section II-C) parameters δ and λ are the second most relevant

TABLE VIII
MAXIMUM VALUES OF DENSITY $f(\text{par}_i|c = \text{high})$ FOR c DENOTING ACCURACY (SECOND COLUMN), SENSITIVITY (THIRD COLUMN), AND SPECIFICITY (FOURTH COLUMN) CONSIDERING ALL THE EXPERIMENTS SHOWN IN TABLES V–VII MERGED INTO ONE DATA STREAM

Parameter	Accuracy	Sensitivity	Specificity
ϕ	0.0002	0.0002	0.0001
ϵ	0.0002	0.0002	0.0002
δ	0.0013	0.0013	0.0011
λ	0.0079	0.0076	0.0101
N	0.9904	0.9907	0.9885

TABLE IX
PROBABILITY FOR EACH FIGURE OF MERIT

Figure of Merit	*High*	*Average*	*Low*	Prediction Accuracy
Accuracy	0.63	0.25	0.12	58.3%
Sensitivity	0.69	0.10	0.22	68.8%
Specificity	0.92	0.08	0.00	93.8%

TABLE X
MEAN AND STANDARD DEVIATION OF EACH PARAMETER GIVEN THE FIGURE OF MERIT IN ITS BEST VALUE ESTIMATED ACCORDING TO THE NAÏVE BAYES MODEL

Parameter	Accuracy	Sensitivity	Specificity
λ	22.6 ± 38.7	20.6 ± 37.5	29.6 ± 43.0
δ	3.65 ± 4.79	3.41 ± 4.63	2.89 ± 4.34
ϵ	0.52 ± 0.32	0.54 ± 0.31	0.48 ± 0.34
ϕ	0.46 ± 0.29	0.46 ± 0.28	0.39 ± 0.30
N	2837 ± 4310	2675 ± 4143	2896 ± 4347

and Chernoff bound (Section II-D) parameters ϕ and ϵ are the least relevant. Analyzing these results, there is a sharp dependency of the algorithm on the size of the training window to be able to increase the figures of merit. However, parameter N is the most relevant in terms of training time, as the algorithm has to wait until the window is filled to continue with the data stream analysis. Additionally, large amounts of N could cause over-fitting, where concept drifts could be masked during the training.

These results are important to conclude that previous knowledge about the analyzed processes is needed to calibrate the existence of concept drifts. For example, if the RUL is expected to be 8 and 2000 h for a tool or a motor, respectively, the size of N should be lower for the tool.

With a clear understanding of each parameter role within the overall dynamic clustering algorithm given by experimentation and posterior analysis with naïve Bayes, mean values and standard deviation for each parameter are given in terms of the figure of merit that is optimized. Therefore, we estimate the mean and the standard deviation of the Gaussian distribution of each parameter for each figure of merit at its best value, i.e., the mean and the standard deviation of the distribution of any parameter given that the figure of merit takes its *high* value. A summary of the parameter values for high accuracy, sensitivity, and specificity is given in Table X.

From Section III-C, where the most accurate experiments (9, 10, and 12) were found, a comparison with the optimal parameters described in Table X can be made. Thus, almost all parameters from those experiments in Section III-C are within the values shown in Table X, meaning that it is possible to optimally tune the algorithm and control the initial training size.

To check the performance of the model built with the optimal parameters, an experiment is designed selecting the mean values for each parameter (Table X). Then, the parameters are introduced into the naïve Bayes model to check the classification process in terms of *low*, *average*, or *high* categories for each figure of merit with its corresponding posterior probability.

The experiment results are shown in Table XI, where the selected parameters are able to predict the highest possible values related to the corresponding figure of merit. In fact, as shown in the table, the posterior probability for the classification is 1.

Next, the parameters are checked using predefined data streams 1–3. Results are shown in Tables XII–XIV for data streams 1 and 2, 100% for accuracy, sensitivity, and specificity is obtained. It is important to stress that the largest detection delay is 135, meaning that it only needs 135 instances to detect the concept drift.

However, for data stream 3 (Table XIV), the parameters for high sensitivity have problems to detect the last concept drift located at instance 32 000. This last concept drift is related to a normal distribution noise with standard deviation 0.05 (Table I), explaining the difficulties for the algorithm to detect those small changes with these parameters.

As a conclusion of this section, it can be seen that algorithm parameters are sensitive to the analyzed scenario, where a tuning is needed the first time it is used. The idea is to select parameters, where concept drifts are detected. Therefore, as a procedure to fine-tune the algorithm parameters depending on the application, a Taguchi L16 design is used to analyze the response and to build a model. In this case, a naïve Bayes is used to select the best parameters for high accuracy, sensitivity, and/or specificity.

Another aspect is the ability of the algorithm to assign the instances to each component of the mixture model (cluster), which is the ultimate goal. The goodness of this assignment process is analyzed in the following section.

V. MODEL ADJUSTMENT MEASUREMENT

The clustering step in the GDPC assigns an instance to each component via the posterior probability that the instance $\boldsymbol{x}_i$ belongs to component k of the mixture, as reflected in (3).

With the aim of studying how the GDPC performs with data streams containing concept drifts during the cluster assignment, we measure the posterior probability that instance $\boldsymbol{x}_i$ belongs to the evolution of component k during concept drift detection. Therefore, it is important to monitor how the model assigns instances to each component and how it begins to mismatch due to a concept drift.

Intuitively, one expects to find a better adjustment once the concept drift is detected and the model is recalculated, i.e., the posterior probabilities that an instance belongs to each

TABLE XI
CLASSIFICATION WITH THE NAÏVE BAYES MODEL AND THE OPTIMAL PARAMETERS

Test	λ	δ	ϵ	ϕ	N	Probability for *low*	Probability for *medium*	Probability for *high*	Prediction Naïve Bayes
Accuracy	22.6	3.65	0.52	0.46	2837	0	0	1	*high*
Sensitivity	20.6	3.41	0.54	0.46	2675	0	0	1	*high*
Specificity	29.6	2.89	0.48	0.39	2896	0	0	1	*high*

TABLE XII
RESULTS FOR OPTIMAL PARAMETERS WITH DATA STREAM 1

Test	λ	δ	ϵ	ϕ	N_{train}	t	CD_{det}	CD_{true}	n_{CD}	Accu.	Sens.	Spec.	Recall	F-score
High accuracy	22.6	3.65	0.52	0.46	2837	8.83	3	3	{15135, 22107, 32054}	1.0	1.0	1.0	1.0	1.0
High sensitivity	20.6	3.41	0.54	0.46	2675	8.53	3	3	{15130, 22030, 32030}	1.0	1.0	1.0	1.0	1.0
High specificity	29.6	2.89	0.48	0.39	2896	9.94	3	3	{15100, 22061, 32114}	1.0	1.0	1.0	1.0	1.0

TABLE XIII
RESULTS FOR OPTIMAL PARAMETERS WITH DATA STREAM 2

Test	λ	δ	ϵ	ϕ	N_{train}	t	CD_{det}	CD_{true}	n_{CD}	Accu.	Sens.	Spec.	Recall	F-score
High accuracy	22.6	3.65	0.52	0.46	2837	8.52	3	3	{15123, 22070, 32082}	1.0	1.0	1.0	1.0	1.0
High sensitivity	20.6	3.41	0.54	0.46	2675	9.57	3	3	{15122, 22115, 32079}	1.0	1.0	1.0	1.0	1.0
High specificity	29.6	2.89	0.48	0.39	2896	7.85	3	3	{15091, 22051, 32072}	1.0	1.0	1.0	1.0	1.0

TABLE XIV
RESULTS FOR OPTIMAL PARAMETERS WITH DATA STREAM 3

Test	λ	δ	ϵ	ϕ	N_{train}	t	CD_{det}	CD_{true}	n_{CD}	Accu.	Sens.	Spec.	Recall	F-score
High accuracy	22.6	3.65	0.52	0.46	2837	8.83	3	3	{15088, 22114, 32059}	1.0	1.000	1.0	1.0	1.0
High sensitivity	20.6	3.41	0.54	0.46	2675	8.54	2	2	{15088, 22038}	0.8	0.667	1.0	1.0	0.8
High specificity	29.6	2.89	0.48	0.39	2896	7.62	3	3	{15076, 22036, 32166}	1.0	1.000	1.0	1.0	1.0

component k ($k = 1, \ldots, K$) will be more similar between them (close to a uniform distribution over the components), so that allocation is no longer so clear.

A. Brier Score

In order to assess the assignment performance of instances to clusters, the Brier score [21] is used. It allows us to measure the deviation of posterior probabilities from those that are expected. Therefore,

$$\text{Brier} = \frac{1}{n}\sum_{j=1}^{K}\sum_{i=1}^{n}\left(P_{ij} - e_{ij}\right)^2 \tag{11}$$

where P_{ij} is the calculated posterior probability of instances $\boldsymbol{x}_i$ and component j, predicted by the model, and e_{ij} takes a value of 1 or 0 if it is the expected component to be allocated or not. For example, if the P_{ij} values of an instance $\boldsymbol{x}_i$ for three components are [0.33, 0.34, 0.33], then e_{ij} is [0, 1, 0]. Accordingly, using (11)

$$\text{Brier} = \frac{1}{1}\Big[(0.33-0)^2 + (0.34-1)^2 + (0.33-0)^2\Big] = 0.65.$$

The lower the values the better, i.e., the assignment probability is more concentrated and less distributed between components. The value of the n-sized window, with n obtained from (8) of Chernoff bound, will be used as the calculation window of the Brier score.

To study how the concept drift affects the Brier score, the data streams used in the previous sections are used below.

B. Experimentation Results and Analysis

The Brier score is monitored over the three data streams to analyze its behavior regarding how the algorithm adjusts a better model after a concept drift detection. For these experiments, the parameters for high accuracy obtained in Section IV are taken ($\lambda = 22.6$, $\delta = 3.65$, $\epsilon = 0.52$, $\phi = 0.46$, $N = 2837$) with the three data streams defined in Table II.

The results of the experiments are shown in Fig. 2, where the GDPC is able to detect the three concept drifts. In each case, every time the model changes (instance 15 000, 22 000, and 32 000), the Brier score changes its behavior and a general stabilization of the Brier score is obtained after the concept drift turning point is detected. As shown in the figures, the Brier score is stable at each leg of different variations described in Table I, which means that the model is constantly assigning, with the same confidence, the instances to specific components of the mixture, only changing if a concept drift is detected.

Therefore, the Brier score can be used to assess the quality of the new model detected after a concept drift, where the idea of dynamic behavior is to adjust the model during data stream processing to guarantee high clustering quality taking into account system changes over time, e.g., degradation.

VI. COMPARISON OF GDPC WITH STATE-OF-THE-ART ALGORITHMS

Using the optimal parameters defined for high accuracy, labeled λ, δ, ϵ, ϕ, and N (Table XI) and data stream no. 3 (Table II), we compared GDPC against some well-known

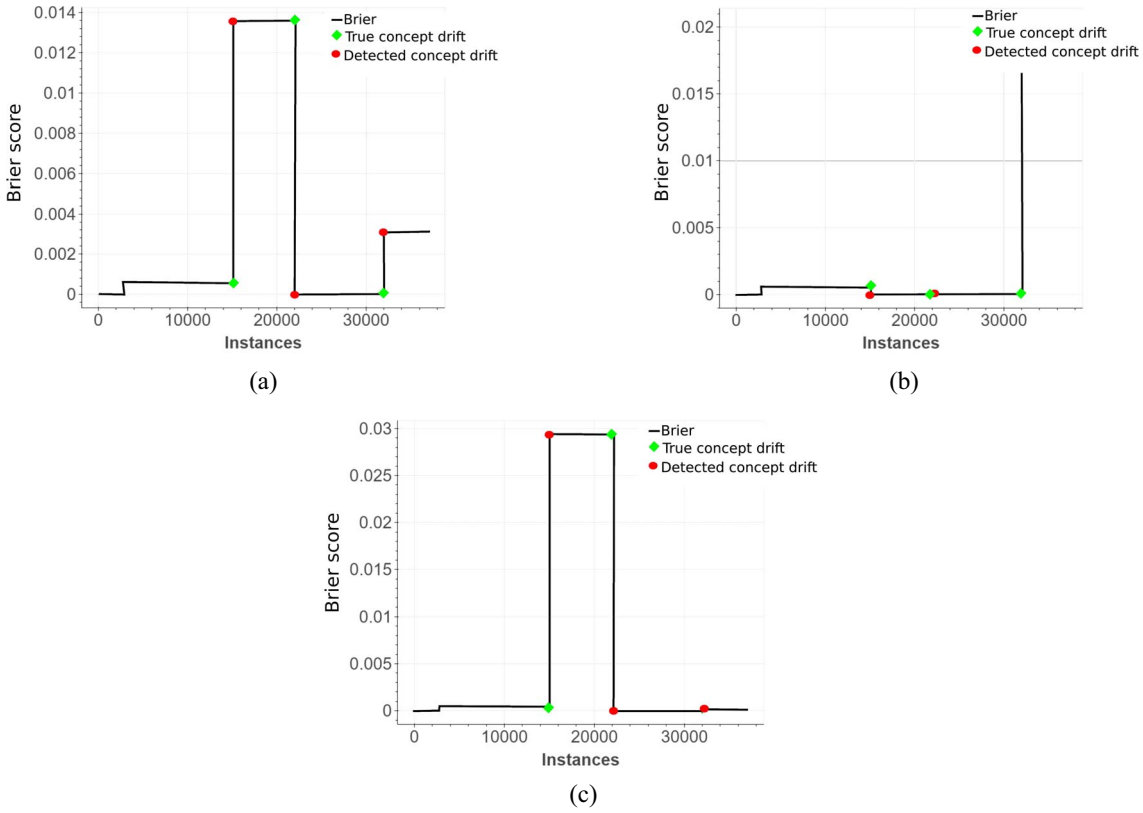


Fig. 2. Brier score evolution for each data stream. (a) Data stream 1. (b) Data stream 2. (c) Data stream 3.

data stream clustering algorithms described in Section I: CluStream (*k*-means-based), D-Stream (DBSCAN-based), and ClusTree (hybrid algorithm with *k*-means and DBSCAN). These algorithms are available at the massive online analysis framework [38].

The analysis was performed on the synthetic data stream for three drifts at instances 15 000, 22 000, and 32,000. True instance cluster labels were output from the synthetic data stream. From the clustering results, the recall and F-score were computed in 1000-instance batches and then plotted to check the evolution throughout the data stream. The analysis results are shown in Fig. 3(a) and (b).

Looking at Fig. 3, D-Stream performance is comparable to GDPC in terms of recall during the first part of the stream. After the first concept drift occurs, the performance drops below the other algorithms, meaning that D-Stream does not have robust capabilities for concept drift detection. In terms of the F-score, the value for D-Stream is below par throughout the data stream with the lowest value of all four analyzed algorithms.

CluStream and ClusTree performed similarly, exhibiting low true positive rates throughout the entire data stream. However, ClusTree does react when the concept drift occurs, although it misses the concept drift located at instance 15 000. Both algorithms performed the best during the last leg of the data stream, though it is unclear whether this behavior is because the algorithm is not concept drift sensitive.

The GDPC algorithm does react to the concept drift, adapting online to the data stream to maintain the highest true positive rate. The algorithm detects the concept drift located at instance 15 000 perfectly, though it is noted that its performance drops in some instances. Once a new GMM model is calculated, the recall increases again, i.e., the algorithm adapts the model to account for the new data from the stream after the concept drift occurs. The next concept drift, at instance 22 000 is also detected, and a new GMM model is launched. This further increases the true positive rate to almost one. The last concept drift, instance 32 000, is detected and the GMM is launched again. In this case, the new model performs worse than CluStream and ClusTree.

Table XV provides a summary of these results.

VII. Dynamic Probabilistic Clustering Applied to Data Streams

A. Data Acquisition

In order to analyze the algorithm performance with data streams using a real machinery, an experimental testbed is used. This testbed, shown in Fig. 4, has the ability to reproduce the real behavior of a machine and all its subsystems from data acquisition to analytics as described in [39]. To simulate concept drifts produced by the machinery, the testbed is programmed to perform a servo-motor movement with different steps of angular velocity, expressed in revolutions per minute (RPM), in order to recreate different states or phases. Thus, the change in angular velocity defined as a concept drift is located every 4500 instances, totaling 25.

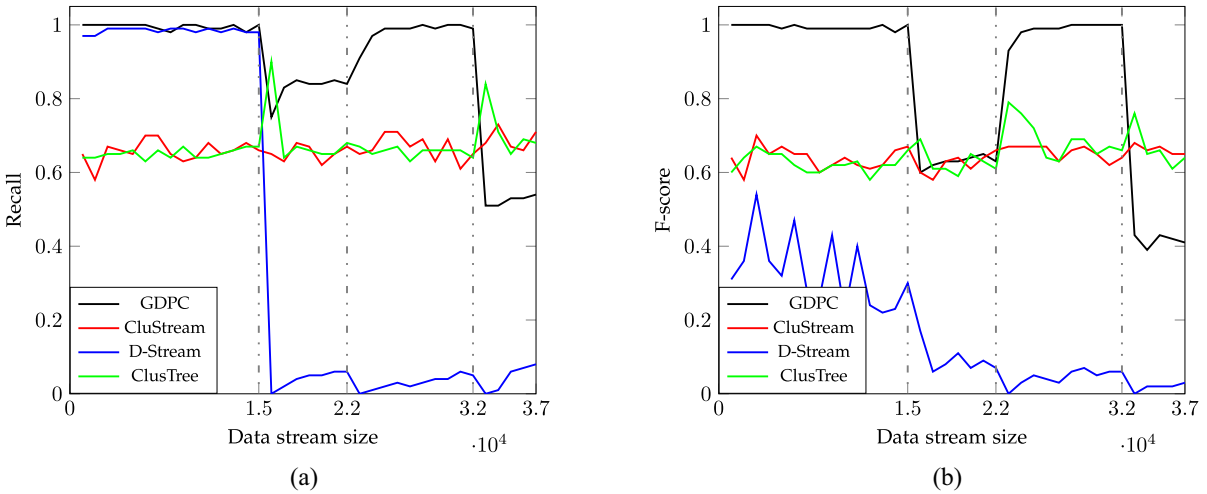


Fig. 3. Recall and F-score for data stream clustering algorithms. (a) Recall. (b) F-score.

TABLE XV
COMPARISON BETWEEN ALGORITHMS

Algorithm	Type	Concept Drift Detection	On-line Adaptation
GDPC	Probability	Yes	Yes
CluStream	Distance	No	Yes
D-Stream	Density	No	No
ClusTree	Hybrid	No	Yes

Fig. 4. Industrial Internet Consortium Testbed.

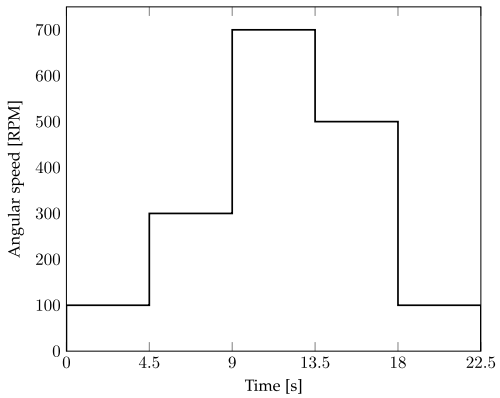


Fig. 5. Testbed angular velocity profile.

The programmed velocity profile is shown in Fig. 5, where the servomotor rotates at 100, 300, 700, 500, and 100 RPM, 4.5 s, respectively. This cycle is repeated five times with the same structure. From each cycle, four variables are acquired: 1) time; 2) active power; 3) angular speed; and 4) torque. When the angular velocity changes, the servomotor variables also vary simulating an alteration in behavior denoted as a concept drift. Consequently, a data stream with 117 822 instances has been generated with a sampling rate of 100 ms.

B. Results and Discussion

Using the algorithm mean parameters for high accuracy, high sensitivity, and high specificity from Table X, the data stream is analyzed. Results are shown in Table XVI. The algorithm shows a high degree of accuracy for concept drift detection (98.7%). In this case, parameters for high accuracy and sensitivity have the best algorithm performance, achieving 98.7% accuracy and sensitivity of 96%, meaning that almost all detections are TP. However, there are about twice the amount of instances detected as turning points for concept drift. Further work on the algorithm development should be oriented to find an effective way to reduce that detection in real/unknown environments.

From the time t required to process the complete data stream of 117 822 instances and 25 concept drifts, the total effective processing time (required time upon data instance arrival) is 22 s, which means that the algorithm needs an average of 185 μs per instance, including the time taken to recalculate the model due to concept drift. In this particular scenario, where monitoring is done at 100 ms, real-time performance is achieved.

Additionally, taking the parameters for high accuracy, the Brier score is monitored (Fig. 6) to check how the concept drift is detected and then the model is adjusted decreasing the value of the score, so there is a high confidence in the assignment of instances to each component.

From the results, it can be seen that the algorithm has the capability to cluster the data stream based on a GMM with a

TABLE XVI
EXPERIMENTAL RESULTS FOR THE TESTBED DATA STREAM

Test	λ	δ	ϵ	ϕ	N_{train}	t	CD_{det}	CD_{true}	n_{CD}	Accu.	Sens.	Spec.	Recall	F-score
High accuracy	22.6	3.65	0.52	0.46	2837	21.07	50	24	{4770, 9215, 13706, 18237, 22775, 22877, 27467... }	0.987	0.96	1.0	1.0	0.98
High sensitivity	20.6	3.41	0.54	0.46	2675	21.69	56	24	{4780, 9239, 13689, 18236, 22686, 22836, 27378... }	0.987	0.96	1.0	1.0	0.98
High specificity	29.6	2.89	0.48	0.39	2896	21.13	49	24	{4696, 9198, 13819, 18199, 22699, 22881, 27437... }	0.986	0.96	1.0	1.0	0.98

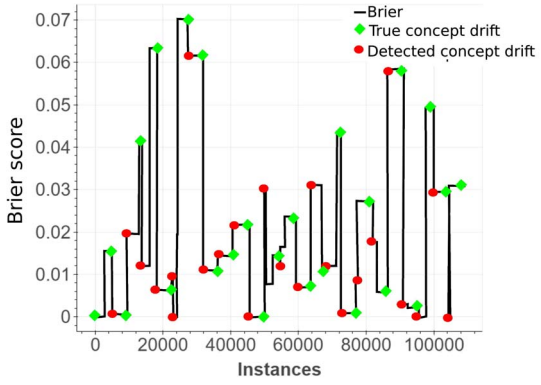


Fig. 6. Testbed results of the Brier score.

reasonable accuracy of around 99% and at the required time. This has been achieved with only 2% of the total available instances. Almost all of the 25 induced concept drifts have been detected along with others changes. These detected concept drifts could alert a monitoring system that the process is changing over time probably due to degradation or some other abnormal behavior. The knowledge obtained from these concept drifts and their evolution over time could be used for further analysis related to the aforementioned RUL.

VIII. CONCLUSION

The main conclusions of this paper are as follows.

1) A new algorithm called GDPC has been developed, which is capable of dealing with data streams with a limited amount of historical data and time needs. It detects concept drifts and adjusts the current model with fitting guarantee.
2) The easiest concept drift detected in a data stream is the first change, regardless of its type. From this point onward, the following changes represent a detection challenge. This behavior is caused by the window size used by the Chernoff bound, which tends to be smaller than the first training set as the algorithm always uses the smallest possible window. This effect has to be controlled using minimal values of ϵ and ϕ to maximize the window size.
3) The algorithm parameters, especially the training window size N, have a critical effect on the result, basically, in its ability to recognize concept drift but also in its accuracy. A slight variation in some of the parameters can cause big changes in the figures of merit.
4) The existing relationship between the algorithm parameters does not allow a simple representation with traditional tools for experimental analysis, this is due to the high dependency of all of them. That is why a naïve Bayes classifier can help to find a good parameter configuration in an automatic way.
5) Regardless of the model used, the size of the training set is the most critical factor and Chernoff bound parameters is the least critical factor.
6) Algorithm performance is related to the nature of the concept drift. Small changes, such as the introduction of noise at low standard deviation, are difficult to detect.
7) GMM clustering is a really powerful tool that is able give useful information for posterior analysis, such as Brier score, mixture parameters, probability of assignment to the cluster, among others.
8) The Brier score is a proper tool to evaluate the ability of dynamic clustering algorithms to adjust the models during the data stream analysis. If the new model is not able to reduce the Brier score, the algorithm performance is not good enough.
9) Real-time applicability of the algorithm to the industry is possible because there is a small footprint in terms of the amount of data needed and processing time. That is, the algorithm uses a small size of data stream window that is continuously adapted to the data stream in order to increase efficiency. From this window, the algorithm is able to update the model if a change occurs without taking excessive processing time, as with the traditional static GMM. This represents a first step to deploying a probability-based clustering technique at the edge, where actionable insights are needed at the right time and accessibility to large computing systems is limited.

Further work related to the development of the algorithm is directed to parameter selection improvement with a more advanced classifier than naïve Bayes, which does not take into account the relationship between variables (parameters). Additionally, specific work should be done in order to optimize whether small changes, such as Gaussian noise with low standard deviation must be detected. On the other hand, work has to be done to increase algorithm features, such as the estimation for the number of components K depending on the training data, where new components could arise or be deleted during the data stream. This new step can leverage the algorithm to be used in novelty detection. Furthermore, the different Gaussian mixtures models produced during concept drift need be studied to differentiate between drift magnitudes, i.e., measure the distance between models. This is especially useful when the algorithm is used in real unsupervised environments, where the concept drift turning point is unknown.

As this algorithm is designed primarily for deployment in a IoT platform, further research has to be

conducted with regard to implementation in terms of computing power footprint. Some preliminary tests with GMMs have been conducted on system-on-chip devices, such as Zynq®Ultrascale+™MPSoC, where programmable logic can accelerate part of the algorithm without the need of complex distributed infrastructures.

ACKNOWLEDGMENT

The authors would like to thank Etxe-Tar for the machine hardware advice and Ikergune for the support during data stream acquisition.

REFERENCES

[1] H. Kopetz, "Internet of Things," in *Real-Time Systems*. Boston, MA, USA: Springer, 2011, pp. 307–323.
[2] S. Jeschke, C. Brecher, T. Meisen, D. Özdemir, and T. Eschert, "Industrial Internet of Things and cyber manufacturing systems," in *Industrial Internet of Things*. Cham, Switzerland: Springer, 2017, pp. 3–19.
[3] R. K. Mobley, *An Introduction to Predictive Maintenance*. Amsterdam, The Netherlands: Butterworth-Heinemann, 2002.
[4] D. Isaacs, J. Diaz, A. Astarloa, and B. Arejita, *Making Factories Smarter Through Machine Learning*, Ind. Internet Consortium, Needham, MA, USA, Jan. 2017. [Online]. Available: https://www.iiconsortium.org/news/joi-articles/2017-Jan-Making-Factories-Smarter-through-Machine-Learning.pdf
[5] Ş. Kolozali, D. Puschmann, M. Bermudez-Edo, and P. Barnaghi, "On the effect of adaptive and nonadaptive analysis of time-series sensory data," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1084–1098, Dec. 2016.
[6] J. Gama, *Knowledge Discovery From Data Streams*. London, U.K.: CRC Press, 2010.
[7] J. A. Silva *et al.*, "Data stream clustering: A survey," *ACM Comput. Surveys*, vol. 46, no. 1, p. 13, 2013.
[8] D. Puschmann, P. Barnaghi, and R. Tafazolli, "Adaptive clustering for dynamic IoT data streams," *IEEE Internet Things J.*, vol. 4, no. 1, pp. 64–74, Feb. 2017.
[9] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: A new data clustering algorithm and its applications," *Data Min. Knowl. Disc.*, vol. 1, no. 2, pp. 141–182, 1997.
[10] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in *Proc. 29th Int. Conf. Very Large Data Bases*, vol. 29. Berlin, Germany, 2003, pp. 81–92.
[11] J. Gama, P. P. Rodrigues, and L. Lopes, "Clustering distributed sensor data streams using local processing and reduced communication," *Intell. Data Anal.*, vol. 15, no. 1, pp. 3–28, 2011.
[12] M. R. Ackermann *et al.*, "StreamKM++: A clustering algorithm for data streams," *J. Exp. Algorithmics*, vol. 17, pp. 2–4, Jul. 2012.
[13] P. S. Bradley, U. M. Fayyad, and C. Reina, "Scaling clustering algorithms to large databases," in *Proc. Knowl. Disc. Databases*, New York, NY, USA, 1998, pp. 9–15.
[14] F. Farnstrom, J. Lewis, and C. Elkan, "Scalability for clustering algorithms revisited," *ACM SIGKDD Explor. Newslett.*, vol. 2, no. 1, pp. 51–57, 2000.
[15] A. Zhou, F. Cao, W. Qian, and C. Jin, "Tracking clusters in evolving data streams over sliding windows," *Knowl. Inf. Syst.*, vol. 15, no. 2, pp. 181–214, 2008.
[16] P. P. Rodrigues, J. Gama, and J. Pedroso, "Hierarchical clustering of time-series data streams," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 5, pp. 615–627, May 2008.
[17] Y. Chen and L. Tu, "Density-based clustering for real-time stream data," in *Proc. 13th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, San Jose, CA, USA, 2007, pp. 133–142.
[18] F. Cao, M. Ester, W. Qian, and A. Zhou, "Density-based clustering over an evolving data stream with noise," in *Proc. Int. Conf. Data Min.*, vol. 6, Soc. Ind. Appl. Math., 2006, pp. 328–339.
[19] P. Kranen, I. Assent, C. Baldauf, and T. Seidl, "The ClusTree: Indexing micro-clusters for anytime stream mining," *Knowl. Inf. Syst.*, vol. 29, no. 2, pp. 249–272, 2011.
[20] G. McLachlan and D. Peel, *Finite Mixture Models*. Hoboken, NJ, USA: Wiley, Mar. 2004.
[21] G. W. Brier, "Verification of forecasts expressed in terms of probability," *Monthly Weather Rev.*, vol. 78, no. 1, pp. 1–3, Jan. 1950.
[22] J. Diaz-Rozo, C. Bielza, and P. Larrañaga, "Machine learning-based CPS for clustering high throughput machining cycle conditions," *Procedia Manuf.*, vol. 10, pp. 997–1008, Jan. 2017.
[23] M.-R. Bouguelia, A. Karlsson, S. Pashami, S. Nowaczyk, and A. Holst, "Mode tracking using multiple data streams," *Inf. Fusion*, vol. 43, pp. 33–46, Sep. 2018.
[24] Y. Wang, G. Chen, and Z. Wang, "A streaming data prediction method based on evolving Bayesian network," in *Web and Big Data* (Lecture Notes in Computer Science). Cham, Switzerland: Springer, 2017, pp. 294–302.
[25] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surveys*, vol. 46, no. 4, pp. 1–37, 2014.
[26] C. Frederickson *et al.*, "Adding adaptive intelligence to sensor systems with MASS," in *Proc. IEEE Sensors Appl. Symp. (SAS)*, Glassboro, NJ, USA, 2017, pp. 1–6.
[27] R. Sebastiao and J. Gama, "A study on change detection methods," in *Proc. 14th Portuguese Conf. Artif. Intell.*, 2009, pp. 12–15.
[28] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Stat. Soc. B (Methodol.)*, vol. 39, no. 1, pp. 1–38, 1977.
[29] H. Akaike, "A new look at the statistical model identification," *IEEE Trans. Autom. Control*, vol. AC-19, no. 6, pp. 716–723, Dec. 1974.
[30] H. Borchani, P. Larrañaga, J. Gama, and C. Bielza, "Mining multi-dimensional concept-drifting data streams using Bayesian network classifiers," *Intell. Data Anal.*, vol. 20, no. 2, pp. 257–280, Jan. 2016.
[31] D. V. Hinkley, "Inference about the change-point from cumulative sum tests," *Biometrika*, vol. 58, no. 3, pp. 509–523, 1971.
[32] H. Mouss, D. Mouss, N. Mouss, and L. Sefouhi, "Test of Page–Hinckley, an approach for fault detection in an agro-alimentary production system," in *Proc. 5th Asian Control Conf.*, vol. 2, Jul. 2004, pp. 815–818.
[33] H. Chernoff, "A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations," *Ann. Math. Stat.*, vol. 23, no. 4, pp. 493–507, 1952.
[34] O. Watanabe, "Simple sampling techniques for discovery science," *IEICE Trans. Inf. Syst.*, vol. E83-D, no. 1, pp. 19–26, Jan. 2000.
[35] D. Barbará and P. Chen, "Tracking clusters in evolving data sets," in *Proc. 14th Florida Artif. Intell. Res. Soc. Conf.*, 2001, pp. 239–243.
[36] G. Taguchi and Y. Wu, *Introduction to Off-Line Quality Control*. Nagoya, Japan: Central Japan Qual. Control Assoc., 1979.
[37] M. Minsky, "Steps toward artificial intelligence," *Proc. IRE*, vol. 49, no. 1, pp. 8–30, Jan. 1961.
[38] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "MOA: Massive online analysis," *J. Mach. Learn. Res.*, vol. 11, pp. 1601–1604, Mar. 2010.
[39] *Smart Factory Machine Learning for Predictive Maintenance Testbed*, Ind. Internet Consortium, Needham, MA, USA, 2017. [Online]. Available: http://www.iiconsortium.org/smart-factory-machine-learning.htm

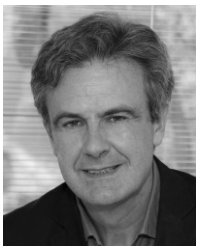
Javier Diaz-Rozo received the M.Eng. degree in mechanical engineering from the University of Los Andes, Bogotá, Colombia, in 2001, and the M.Sc. degree in advanced manufacturing technology and systems management from the University of Manchester, Manchester, U.K., in 2003. He is currently pursuing the Ph.D. degree in artificial intelligence at the Universidad Politécnica de Madrid, Madrid, Spain.

He was with Aingura IIoT, Elgoibar, Spain, as the IIoT Team Leader. He has accumulated nearly 15 years of industrial experience researching in different positions related with research and development: Research and Development Project Manager with Ikergune, the Etxe-Tar Group Research and Development Unit, responsible for the advanced manufacturing research area, Senior Consultant in a research and development consulting firm from 2010 to 2014, Research and Development Director in a business group mainly dedicated to the wind energy sector from 2008 to 2010, and the Director for the Advanced Manufacturing Area with the ASCAMM Technology Centre, Barcelona, Spain, from 2006 to 2008.

Concha Bielza received the M.S. degree in mathematics from the Universidad Complutense de Madrid, Madrid, Spain, in 1989, and the Ph.D. degree in computer science from the Universidad Politécnica de Madrid, Madrid, in 1996.

Since 2010, she has been a Full Professor of statistics and operations research with the Departamento de Inteligencia Artificial, Universidad Politécnica de Madrid, where she co-leads the Computational Intelligence Group. She has authored or co-authored over 100 papers in impact-factor journals and has supervised 9 Ph.D. theses. Her current research interests includes probabilistic graphical models, machine learning, decision analysis, metaheuristics for optimization, classification models, and real applications, like biomedicine, bioinformatics, neuroscience, industry 4.0, and sport analytics.

Dr. Bielza was a recipient of the Extraordinary Doctorate Award for her Ph.D. degree and the 2014 UPM Research Prize.



Pedro Larrañaga received the M.Sc. degree in mathematics (statistics) from the University of Valladolid, Valladolid, Spain, and the Ph.D. degree in computer science from the University of the Basque Country (UPV-EHU), Leioa, Spain.

His academic career was developed with the UPV-EHU with several faculty ranks: an Assistant Professor from 1985 to 1998, an Associate Professor from 1998 to 2004, and a Full Professor from 2004 to 2007. He earned the habilitation qualification for Full Professor in 2003. He has been a Full Professor in computer science and artificial intelligence with the Universidad Politécnica de Madrid, Madrid, Spain, since 2007, where he co-leads the Computational Intelligence Group. He has served as an Expert Manager of the Computer Technology Area with the Deputy Directorate of Research Projects, Spanish Ministry of Science and Innovation from 2007 to 2010. He has been a member of the Advisory Committee 6.2 (Communication, Computing, and Electronics Engineering) of the CNEAI, Spanish Ministry of Education from 2010 to 2011. He has authored or co-authored over 200 papers in impact factor journals and has supervised 25 Ph.D. theses. His current research interests include probabilistic graphical models, data science, metaheuristics, and real applications, like biomedicine, bioinformatics, neuroscience, industry 4.0, and sports.

Dr. Larrañaga was a recipient of the 2013 Spanish National Prize in Computer Science and the Spanish Association for Artificial Intelligence Prize in 2018, and the Excellence Award for his Ph.D. degree. He has been a Fellow of the European Association for Artificial Intelligence since 2012.