Engineering Applications of Artificial Intelligence 89 (2020) 103434

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

Machine-tool condition monitoring with Gaussian mixture models-based dynamic probabilistic clustering☆

Javier Diaz-Rozo [a,b,*], Concha Bielza [b], Pedro Larrañaga [b]

[a] Aingura IIoT, San Antolin 3, Elgoibar 20870, Spain
[b] Departmento de Inteligencia Artificial, Universidad Politécnica de Madrid, Madrid, Spain

ARTICLE INFO

Keywords:
Concept drift
Condition monitoring
Data stream
Dynamic clustering
Gaussian mixture model
Machining operation

ABSTRACT

The combination of artificial intelligence with data, computing power, and new algorithms can provide important tools for solving engineering problems, such as machine-tool condition monitoring. However, many of these problems require algorithms that can perform in highly dynamic scenarios where the data streams have extremely high sampling rates from different types of variables. The unsupervised learning algorithm based on Gaussian mixture models called Gaussian-based dynamic probabilistic clustering (GDPC) is one of these tools. However, this algorithm may have major limitations if a large amount of concept drifts associated with transients occurs within the data stream. GDPC becomes unstable under these conditions, so we propose a new algorithm called GDPC+ to increase its robustness. GDPC+ represents an important improvement because we introduce: (a) automatic selection of the number of mixture components based on the Bayesian information criterion (BIC), and (b) concept drift transition stabilization based on Cauchy–Schwarz divergence integrated with the Dickey–Fuller test. Thus, GDPC+ can perform better in highly dynamic scenarios than GDPC in terms of the number of false positives. The behavior of GDPC+ was investigated using random synthetic data streams and in a real data stream-based condition monitoring obtained from a machine-tool that produces engine crankshafts at high speed. We found that the initial temporal window size can be used to adapt the algorithm to different analytical requirements. The clustering results were also investigated by induction of the rules generated by the repeated incremental pruning to produce error reduction (RIPPER) algorithm in order to provide insights from the underlying monitored process and its associated concept drifts.

1. Introduction

The applications of artificial intelligence in engineering are increasing rapidly as changes in computing power allow algorithms to be applied in highly dynamic scenarios, such as in the manufacturing sector, where rapid responses to highly complex queries are required, e.g., machine-tool condition monitoring during operation. As explained by Larrañaga et al. (2019), machine learning algorithms can be applied to solve specific engineering needs at any level with respect to the component, machine, production, and logistics, where condition monitoring can arise at any level.

Specifically, a machine-tool condition monitoring system can be focused on different elements, e.g., tool, spindle head, servomotors, linear axis, chatter (dynamic stability), etc. Additionally, it can have different objectives, such as detection, diagnosis and prognosis. For example, the failure prediction of ball-screw can be done with a linear axis monitoring system. These types of solutions are mainly obtained off-line, where very large data sets are analyzed based on supervised or unsupervised learning.

However, some engineering applications such as degradation analysis for production assets or condition monitoring require online analysis, where the algorithm must be able to process data in real time without any previous knowledge of the degradation patterns. In this scenario, there are some approaches where classifiers such as rClass (Pratama et al., 2019a) and pENsemble+ (Pratama et al., 2019b) have been applied for tool condition monitoring, using machine data streams. However, these models need to be trained with different tool states (classes or labels) to be able to classify. These classes might be defined as the degradation state of the tool, e.g., in terms of its cutting edge: good condition, fair condition and worn-out condition. For tool condition monitoring, the generation of relatively balanced datasets for training is feasible as machine tool life is short. Analyzing degradation in other (non-consumable) elements with useful life measured in months or years, unsupervised learning such as clustering with

☆ No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.engappai.2019.103434.

* Corresponding author at: Aingura IIoT, San Antolin 3, Elgoibar 20870, Spain.
E-mail address: jdiaz@ainguraiiot.com (J. Diaz-Rozo).

https://doi.org/10.1016/j.engappai.2019.103434
Received 8 March 2019; Received in revised form 21 November 2019; Accepted 18 December 2019
Available online 5 January 2020
0952-1976/

Table 1
Clustering algorithms for data streams.

Type	Algorithm name, authors	Description
K-means	BIRCH, Zhang et al. (1997).	Incremental algorithm based on a data structure called *CF-Tree* for summarizing the shape of the clusters generated by K-means.
	CluStream, Aggarwal et al. (2003).	This algorithm divides the clustering process into two steps: online for storing summary statistics from the data and offline where the summary statistics are used to increase the processing efficiency in the data stream, i.e., processes with high computational cost are run offline. In this algorithm, standard K-means is used in the online step to initialize the micro-clusters and modified K-means is then employed to create a macro-cluster.
	DGClust, Gama et al. (2011).	Distributed algorithm that monitors data changes on acquisition nodes and reports them to a central server by ranking the nodes with the most changes, where adaptive K-means is employed to find the clusters.
	StreamKM++, Ackermann et al. (2012).	Similar to the BIRCH algorithm, but performs a non-uniform sampling and the K-means++ algorithm is applied to generate the clusters. A different data structure is used compared with CluStream, which is based on a merge-and-reduce technique (Har-Peled and Mazumdar, 2004) for summarizing the size and shape of the clusters.
	SWClustering, Zhou et al. (2008).	Similar to BIRCH algorithm but the data structure employed is based on histograms for tracking the evolution of clusters.
	AutoClust, Lughofer and Sayed-Mouchaweh (2015).	This algorithm compresses data streams with the support of well-defined *prototypes* in terms of local regions, compact and unique clusters. If the complexity of those clusters increases, then more clusters are produced splitting cluster shapes into ellipsoidal pieces. Then a single-pass merging operation is done, solving overlapping (instances split up into distinct clusters).
Hierarchical	ODAC, Rodrigues et al. (2008).	The algorithm maintains a tree-like hierarchy of clusters, which evolves with the data. The dynamic split and merge process are triggered by changes in the diameters of the clusters.
DBSCAN	D-Stream, Chen and Tu (2007).	Similar to CluStream, the algorithm divides the clustering process into two steps: online for mapping each instance into a grid in the data space and offline by using DBSCAN to compute the grid density and the clusters in the grids.
	DenStream, Cao et al. (2006).	Two structures called *core-micro-cluster* and *outlier-micro-cluster* are created using DBSCAN and employed to summarize the clusters with arbitrary shape and outliers, respectively.
GMMs	GDPC, Diaz-Rozo et al. (2018).	GMMs are used as the data density function. Cluster evolution is detected when the GMM no longer faithfully represents the data coming from the data stream.
Hybrid	ClusTree, Kranen et al. (2011).	This algorithm uses a mixture of hierarchical and K-means clustering, where it monitors the age of the data and assigns more importance to the newest data. Therefore, the data stream summary is maintained by an adaptive ranking structure.

data stream processing capabilities is more appropriate. Table 1 summarizes the different methods that can be employed with data stream clustering where they are grouped according to the well-known static counterparts: K-means, hierarchical, density-based spatial clustering of applications with noise (DBSCAN), Gaussian mixture models (GMM), and any hybrid of these.

As explained by Diaz-Rozo et al. (2017), probabilistic clustering based on GMMs (McLachlan and Peel, 2004) can extract valuable information from data by producing interpretable mappings from the clustering results to the behavioral pattern of the machine under analysis. In addition, GMM-based clustering is referred to as soft clustering, where an instance could belong to all components (clusters) in the mixture with different probabilities, thereby providing more information regarding the instance assignments that could be useful for understanding, e.g., instances with a high probability of being part of two components, which is impossible with other clustering techniques.

Nevertheless, GMM-based clustering has a high computational burden because its internal expectation–maximization (EM) algorithm (Dempster et al., 1977) requires complex iterations to estimate the Gaussian mixture parameters. In addition, variations in the data distribution in the data stream can affect the accuracy of the clustering process with a fixed GMM model. Model updating is required to overcome this issue, which demands extra processing time. These variations in the data distribution are referred to as concept drift (Schlimmer and Granger, 1986; Widmer and Kubat, 1996; Khamassi et al., 2018), which was defined by Gama et al. (2011, 2014) as a change over time in the shift of the relationship between the input data and the target variable. In this case, the target is represented by the clusters and their underlying model, i.e., concept drift in GMM-based clustering is found when the data are no longer represented by the current GMM.

According to Diaz-Rozo et al. (2018), Yang et al. (2018), and Sidhu and Bhatia (2018), concept drift detection is an important feature and it must be handled in real-time during data stream analysis to avoid using models that neglect the shifts between the input and output, thereby producing inaccurate results. Concept drifts can also represent an important source of information regarding the process under analysis.

The Gaussian-based dynamic probabilistic clustering (GDPC) algorithm proposed by Diaz-Rozo et al. (2018) (see Table 1) can exploit knowledge obtained from engineering processes at different levels and work with data streams, where the EM algorithm is launched only if a concept drift is detected. Nevertheless, GDPC works with parameters such as the number of components that need to be estimated by preliminary data acquisition and based on user expertise regarding the process under analysis. For example, if a machining cycle has three states comprising *stopped*, *dry-cycle*, and *in-cycle*, and the problem involves determining the pattern of each state, then the algorithm user needs to know in advance that the number of components must be three, which might be unreliable if other interesting patterns exist. In addition, the model updating algorithm can be unstable after a concept drift because non-stationary data can still undergo relationship shifts between the input and output, thereby triggering further updates, reducing the computing performance (i.e., requiring more EM algorithm calculations), and increasing the number of false positives (i.e., false concept drift alarms).

Therefore, the main objective of this study was to improve the GDPC algorithm to overcome these limitations. The new algorithm called GPDC+ improves the GDPC by adding two new capabilities, as follows.

- Reducing the requirement for prior user knowledge and testing needs. The amount of training data is the only parameter that needs to be controlled by the user and it is directly related to the analysis objective, which is determined by univariate testing.
- Minimizing the instability of the GDPC caused by non-stationary data during concept drifts.

Thus, the GDPC+ can operate under completely unknown scenarios to hopefully reduce the number of false positives and improve its suitability for real scenarios.

The remainder of this paper is organized as follows. In Section 2, we review the GDPC and explain the improvements to the original algorithm in terms of dynamic component selection and non-stationary data processing. In Section 3, we present a performance assessment in order to demonstrate the improvements obtained due to the new characteristics of the algorithm. In addition, we propose a criterion for parameter selection to make the algorithm more suitable for real applications. In Section 4, we present the application of the algorithm to a real data stream coming from a crankshaft manufacturing machine tool, and we analyze the types of actionable insights that can be obtained from the improved algorithm and how their evolution is detected online. In Section 5, we give our conclusions and suggestions for further research.

2. GDPC+ methodology

According to Diaz-Rozo et al. (2018), the GDPC algorithm pipeline is based on six steps, as shown in Fig. 1a, which are summarized as follows.

1. Offline training of a GMM with an initial data set $\mathscr{D} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\}$ of size N, where each instance is assigned to K components.
2. Fitting test for new instances coming from the data stream into the current trained model, which is measured based on the mean log-likelihood.
3. Outlier detection for mean log-likelihood measurements using the Page–Hinkley test.
4. Concept drift detection based on the Chernoff bound, which sets the maximum number of outliers allowed within an evaluation window.
5. If no concept drift is detected, then for each instance, we compute each cluster membership probability with the responsibilities provided by the EM algorithm according to the GMM current model.
6. If a concept drift is found, the model is re-adjusted using the last window $\mathscr{D}' = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$ of size n, with $n < N$.

However, this pipeline makes two assumptions that lead to important issues when the GDPC is operating on data streams with unknown behavior, which is the main objective of unsupervised algorithms. Therefore, improving the GDPC involves developing new strategies to increase the overall performance of the algorithm under conditions with unknown behavior.

The first assumption is related to steps 1 and 4 in the pipeline (Fig. 1a), where each instance is assigned to K components and K is kept constant for the whole algorithm throughout data stream processing. In this case, if the data are represented by a GMM, the number of components that best fit the data could change over time because cluster merging and splitting phenomena occur during the modeling process (Spiliopoulou et al., 2006). An initial estimation of the number of components is needed from the user, who may have previous knowledge of the process under analysis. Measures such as the Bayesian information criterion (BIC) (Schwarz, 1978) may be used to dynamically determine the number of components each time that a GMM is updated. The modification is shown in Fig. 1b for steps 2 and 8, which are employed for dynamic component estimation.

Therefore, the BIC criterion is launched in step 2 as part of the offline training and in step 8 after a concept drift is detected. As described in the GDPC algorithm, the Chernoff bound is part of the concept drift detection step and it is based on the minimum number of instances (s) that are not outliers, according to

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

and an adaptive window of size n,

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

where $0 < \epsilon < 1$ is the additive error bound, $0 < \phi < 2$ is a constant to control the probability of an instance to be successfully clustered and p is the probability to receive a non-outlier instance.

The second assumption is related to step 6 in the pipeline (Fig. 1a), where a concept drift has been detected and a model is updated. After the concept drift is detected, the model is re-estimated from the last data window. A concept drift is produced by a change in the pattern of the data, so the last and next windows could include non-stationary data and other noisy data that feed into the updating process. Depending on the nature of the concept drift, this updating process could be unstable over time, thereby requiring the launch of a model update at each subsequent iteration until the new data reach a steady state.

Therefore, a monitoring system is proposed in order to control the instability of the algorithm. This monitoring system measures the divergence between two GMMs (GMM before concept drift and GMM after model update because a concept drift) using the closed-form Cauchy–Schwarz divergence described by Kampa et al. (2011) and its variation over time using the Dickey–Fuller test (Dickey and Fuller, 1979). Thus, step 9 is added in Fig. 1b where a change in the trend of the divergence measurement determines the model updating process.

Detailed descriptions of each new step are given as follows.

2.1. Dynamic component estimation

From the GDPC algorithm, the density in the kth cluster is $f_k(\boldsymbol{x}; \boldsymbol{\theta}_k)$ and the GMM model is given by:

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

with the parameters $\boldsymbol{\Psi} = (\pi_1, \dots, \pi_K, \boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_K)$, where π_k is the weight of component K within the mixture, $\boldsymbol{\theta}_k = (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, and $K = 1, \dots, K$ and $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$ are the vector of means and the covariance matrix of component K, respectively.

According to McLachlan and Peel (2004), an information criterion for model selection can be obtained based on the bias-corrected log likelihood ($\log L$) given by:

$$\log L(\hat{\boldsymbol{\Psi}}) - b, \tag{3}$$

where $\hat{\boldsymbol{\Psi}}$ is the estimation of the GMM unknown parameter $\boldsymbol{\Psi}$ obtained by using maximum likelihood estimation or EM algorithm. The term b is the bias.

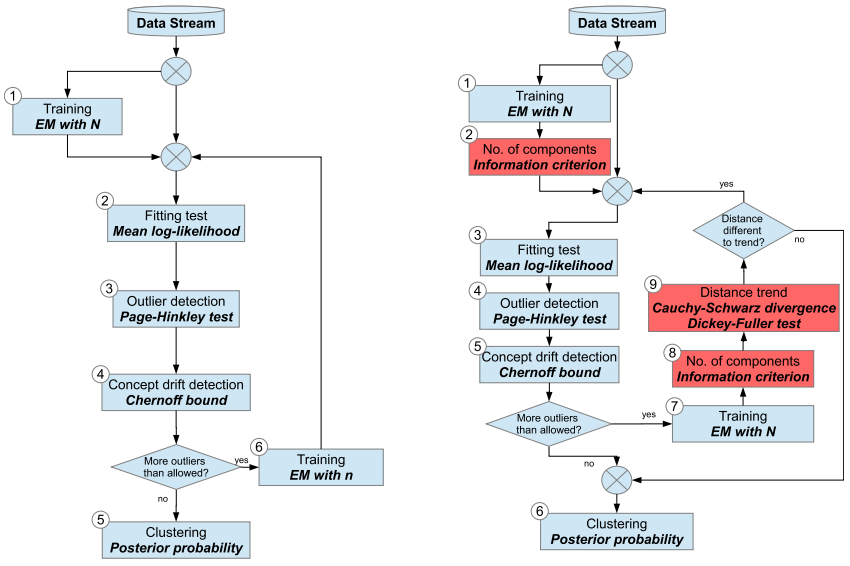
The information criterion for Eq. (3) is generally expressed as:

$$-2 \log L(\hat{\boldsymbol{\Psi}}) + 2C, \tag{4}$$

where the first term measures the lack of fit and $C > 0$ is a penalty to account for the model's complexity. Choosing a model depends on the minimization of Eq. (4). For BIC, Eq. (4) is expressed as:

$$-2 \log L(\hat{\boldsymbol{\Psi}}) + dim(\hat{\boldsymbol{\Psi}}) \log N, \tag{5}$$

where N is the sample size. The minimization of Eq. (5) provides a model where fitting and complexity are balanced. Therefore, the number of components is the value of K that minimizes Equation (5) for a constant value of N.



(a) GDPC (b) GDPC+

Fig. 1. Algorithm schemes. The new features in GDPC+ are shown in red boxes. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

In the GDPC+ algorithm, the minimum BIC value is sought within a range of possible number of components, where the lowest value BIC* is preferred. Then, this value is:

$$\text{BIC}^* = \min_{\hat{\Psi}} \left(-2 \log L(\hat{\Psi}) + dim(\hat{\Psi}) \log N\right).$$

Lughofer (2012) also used the BIC criterion during a splitting phase of already extracted clusters (mixture components). A merging phase, not using the BIC, is then applied, both phases leading to find the number of clusters. This implicit drift reaction differs from GDPC+ that first explicitly detects a concept drift and then finds the number of clusters by minimizing the BIC criterion.

2.2. Concept drift transient stabilization

In this case, the improvement of the algorithm is based on the hypothesis that a concept drift should be represented by a large divergence between the current GMM and the GMM calculated after the concept drift. To measure this divergence between two Gaussian mixtures with different parameters $\hat{\Psi}$ and different number of components, we use the closed-form Cauchy–Schwarz divergence defined by Kampa et al. (2011). Therefore, we start from two GMMs:

$$p(\boldsymbol{x}; \Psi) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{6}$$

and

$$q(\boldsymbol{x}; \Psi) = \sum_{m=1}^{M} \tau_m \mathcal{N}(\boldsymbol{x}; \boldsymbol{\nu}_m, \boldsymbol{\Lambda}_m), \tag{7}$$

where K is the number of components and π_k, $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$ are the parameters for GMM $p(\boldsymbol{x}; \Psi)$, and M is the number of components and τ_m, $\boldsymbol{\nu}_m$, $\boldsymbol{\Lambda}_m$ are the parameters for GMM $q(\boldsymbol{x}; \Psi)$. Each component in the corresponding GMM is given by:

$$\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{(2\pi)^{D/2} |\boldsymbol{\Sigma}_k|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_k)\right) \tag{8}$$

and

$$\mathcal{N}(\boldsymbol{x}; \boldsymbol{\nu}_m, \boldsymbol{\Lambda}_m) = \frac{1}{(2\pi)^{D/2} |\boldsymbol{\Lambda}_m|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\nu}_m)^T \boldsymbol{\Lambda}_m^{-1} (\boldsymbol{x} - \boldsymbol{\nu}_m)\right), \tag{9}$$

with $\boldsymbol{x} \in \Re^D$.

The Cauchy–Schwarz divergence measure between $p(\boldsymbol{x}; \Psi)$ and $q(\boldsymbol{x}; \Psi)$ is given by:

$$D_{CS}(p, q) = -\log\left(\frac{\int p(\boldsymbol{x}; \Psi) q(\boldsymbol{x}; \Psi) d\boldsymbol{x}}{\sqrt{\int p^2(\boldsymbol{x}; \Psi) d\boldsymbol{x} \int q^2(\boldsymbol{x}; \Psi) d\boldsymbol{x}}}\right),$$

which can be rewritten as

$$D_{CS}(p, q) = -\log\left(\int p(\boldsymbol{x}; \Psi) q(\boldsymbol{x}; \Psi) d\boldsymbol{x}\right) + \frac{1}{2}\log\left(\int p^2(\boldsymbol{x}; \Psi) d\boldsymbol{x}\right) + \frac{1}{2}\log\left(\int q^2(\boldsymbol{x}; \Psi) d\boldsymbol{x}\right).$$

By distributing the integrals into the sum and using Eqs. (6), (7), (8), and (9), the closed-form expression is given by:

$$\begin{aligned} D_{CS}(p, q) = &-\log\left(\sum_{k=1}^{K}\sum_{m=1}^{M} \pi_k \tau_m z_{km}\right) \\ &+ \frac{1}{2}\log\left(\sum_{k=1}^{K} \frac{\pi_k^2}{(2\pi)^{D/2}|\boldsymbol{\Sigma}_k|^{1/2}} + 2\sum_{k=1}^{K}\sum_{k'<k} \pi_k \pi_{k'} z_{kk'}\right) \\ &+ \frac{1}{2}\log\left(\sum_{m=1}^{M} \frac{\tau_m^2}{(2\pi)^{D/2}|\boldsymbol{\Lambda}_m|^{1/2}} + 2\sum_{m=1}^{M}\sum_{m'<m} \tau_m \tau_{m'} z_{mm'}\right), \end{aligned} \tag{10}$$

where

$$z_{km} = \mathcal{N}\left(\boldsymbol{\mu}_k; \boldsymbol{\nu}_m, (\boldsymbol{\Sigma}_k + \boldsymbol{\Lambda}_m)\right),$$
$$z_{kk'} = \mathcal{N}\left(\boldsymbol{\mu}_k; \boldsymbol{\mu}_{k'}, (\boldsymbol{\Sigma}_k + \boldsymbol{\Sigma}_{k'})\right), \text{ y}$$
$$z_{mm'} = \mathcal{N}\left(\boldsymbol{\nu}_m; \boldsymbol{\nu}_{m'}, (\boldsymbol{\Lambda}_m + \boldsymbol{\Lambda}_{m'})\right).$$

After measuring the divergence between two GMMs, it is important to monitor its size. Thus, Mushtaq (2011) proposed the augmented Dickey–Fuller (ADF) test for this purpose, which is a hypothesis test that is generally used to test trends over macroeconomic data and to determine whether the data are stationary or not. The data are defined as stationary when the mean and covariance parameters are time invariant during a time lag.

According to Cheung and Lai (1995), if $\Delta D_{CS_t}(p, q) = D_{CS_t}(p, q) - D_{CS_{t-1}}(p, q)$ is the Cauchy–Schwarz divergence change over time t, then

the ADF test involves the regression:

$$\begin{aligned}\Delta D_{CS_t}(p,q) =& \alpha + \beta t + \gamma D_{CS_{t-1}}(p,q) \\ &+ \delta_1 \Delta D_{CS_{t-1}}(p,q) + \cdots + \delta_{u-1}\Delta D_{CS_{t-u+1}}(p,q) \\ &+ \epsilon t\end{aligned} \tag{11}$$

where α is a constant, β is the time coefficient, ϵ is the error, u is the lag order of the autoregressive process, δ_i comprises the autoregressive coefficients ($i = 1, \ldots, u-1$), and γ is a coefficient, which its negativity over the time t is examined.

Intuitively, if $\gamma \propto \frac{1}{D_{CS_{t-1}}(p,q)}$, a large value of the divergence $D_{CS_{t-1}}(p,q)$ does not provide information related to its change in time t as it minimizes γ, and thus it is defined as stationary. An increase in the negativity of γ is related to a small value of $D_{CS_{t-1}}(p,q)$ providing more information to its change in time t. To define a threshold where the divergence is non-stationary a hypothesis test, the ADF test, is carried out.

Therefore, the ADF test is as follows:

$$\begin{cases} H_0 : & \gamma = 0 \\ H_1 : & \gamma < 0 \end{cases}$$

where different values of $D_{CS_t}(p,q)$ are recorded and $\Delta D_{CS_t}(p,q)$ calculated during a lag of size u. $\alpha \neq 0$ and $\beta \neq 0$ constants are also calculated during this lag.

If H_0 is rejected, then the change in the Cauchy–Schwarz divergence, $\Delta D_{CS_t}(p,q)$, is non-stationary, and thus there is a large change between the distributions, i.e., the previous value $\Delta D_{CS_{t-1}}(p,q)$ is relevant. A concept drift is confirmed at this time. Therefore, monitoring γ and providing a confidence value to reject the null hypothesis avoids updating the GMM during non-stationary periods of the data stream and filtering unstable effects.

In addition, different non-stationary behaviors can be studied based on Eq. (11):

- If $\beta = 0$ and $\alpha = 0$, then the ADF test is conducted for stationary behavior.
- If $\beta = 0$ and $\alpha \neq 0$, then the ADF test is conducted for stationary behavior with drift.
- If $\beta \neq 0$ and $\alpha \neq 0$, then the ADF test is conducted for stationary behavior with drift and a time trend.

In order to demonstrate how these new features help to improve the GDPC, we assessed its performance with simulated data streams, as described in the following.

3. GDPC+ performance assessment

The improved performance of GDPC+ compared with the GDPC algorithm was evaluated in terms of different figures of merit. We used the accuracy, sensitivity, specificity, recall, and F-score. A synthetic data set comprising 20 variables and 20 000 instances was produced from a GMM with different parameter sets using a continuous distribution to draw:

- a vector of 20 different values for the mean,
- a matrix of 20 by 20 values for the covariance, and
- the number of components from 2 to 9.

To evaluate the detection of concept drift, four concept drifts were introduced by randomly changing the distribution parameters and finally located at instance number 6133, 6393, 9819, and 18 400.

The following measures were used to estimate the figures of merit.

1. Instances where a concept drift is detected (n_{CD}): The instance number where concepts drifts are detected within a data stream.
2. True concept drifts (CD_{true}): The amount of true concept drifts compared with $CD_{set} = \{6133, 6393, 9819, 18\,400\}$.

Table 2
Performance results obtained with static and dynamic component selection.

K components	$Inst/s$	CD_{true}	n_{CD}	Accu. [%]	Recall [%]	Spec. [%]	F-score [%]
3	1293.19	0	12	99.90	0.00	99.90	0.00
4	2238.38	0	104	99.50	0.00	99.50	0.00
5	676.44	4	383	98.10	98.10	98.10	99.00
6	1239.82	4	383	98.10	98.10	98.10	99.00
7	591.78	4	383	98.10	98.10	98.10	99.00
9	1014.21	4	383	98.10	98.10	98.10	99.00
Dynamic	358.12	4	383	98.10	98.10	98.10	99.00

The performance was assessed using the figures of merit derived from concept drift detection, where we compared true concept drifts and concept drifts detected by the model to estimate the accuracy, sensitivity, specificity, recall, and F-score.

3.1. Dynamic component selection performance

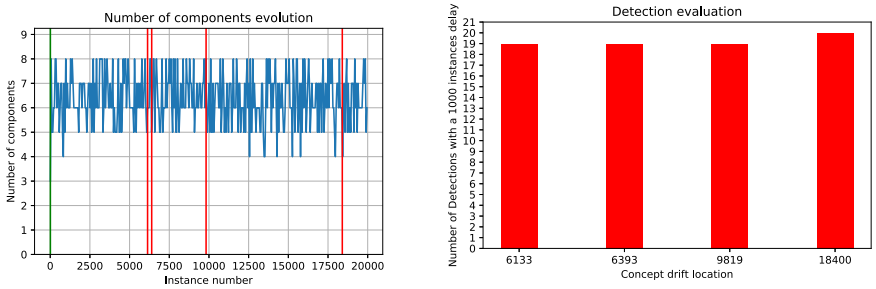
The number of components selected was determined dynamically by using the BIC score, as described in Section 2.1, and the performance was analyzed based on comparisons with GDPC (static component setting) by using different component values from $K = \{3, 4, 5, 6, 7, 9\}$ and a temporal window $N = 10$. BIC analysis was conducted using the `bic()` function from the Scikit-Learn library (Pedregosa et al., 2011).

The results are shown in Table 2. It should be noted that dynamic component selection obtained the same performance as static analysis when K took values of 5, 6, 7 and 9. However, the processing frequency (as the number of instances per second) was slower because the proposed method aims to minimize the BIC value, whereas the static version GDPC does not need to perform a search as the value is fixed in advance.

Analysis of dynamic model selection in terms of the changes in the number of components showed that the values oscillates between $K = 4$ and $K = 9$ (Fig. 2a). Using these values, true concept drift detection began to occur with GDPC, except when $K = 4$ (Table 2). Algorithm performance has high values for accuracy, recall, specificity and F-score after $K > 4$ with detection of the concept drifts, i.e., with $K = 3$ and $K = 4$ not true concept drift is detected, only false positives, $n_{CD} = 12$ and 104 respectively. From $K = 5$ to $K = 9$, performance values are not 100% because of the large amount of false positives detected, i.e., $n_{CD} = 383$ false concept drifts each. However, all available true concept drifts ($CD_{true} = 4$) are detected. Therefore, when working with an unknown data stream (i.e., no information was available about the data structure to help estimate the number of components), dynamic component selection could obtain the same results as setting the components before running the algorithm. This is an interesting result because dynamic component selection may allow the GPDC+ algorithm to work without initial estimations of the number of components given by an expert, and with low computational costs according to the processing frequency.

Table 2 shows that after the algorithm began to detect concept drifts, the number of false positives started to increase and the specificity indicator for the GDPC algorithm decreased from 99.90% to 98.10%, which could lead to poor performance when the algorithm is required to detect degradation behavior because it would trigger false alarms continuously. A deeper analysis of this behavior showed that multiple concept drift detections were found in the range between CD_{set} and $CD_{set} + 1000$ instances, where 19 concept drift detections were signaled at instance number 6133, 6393, and 9819, and 20 concept drift detections were signaled at instance number 18 400 instances (see Fig. 2b).

The multiple signaling events were instabilities that occurred after concept drifts were detected, where the new model used non-stationary data for updating. After a new model was calculated, the next instances coming from the data stream could also have unstable values because a concept drift occurred, thereby triggering a new model calculation.



(a) (b)

Fig. 2. (a) Dynamic component selection evolution based on the data stream. Red lines indicate real concept drift locations. (b) Number of concept drifts detected with a 1000 instances delay.

This problem makes the algorithm inefficient because of the increased amount of false positives. Therefore, stabilization with a filtering capability technique was introduced into the GPDC+ algorithm in order to reduce the effects of concept drift on the algorithm's performance. The concept drift stabilization feature is introduced in the next section.

3.2. Concept drift transient stabilization performance

When concept drifts are detected by the algorithm, a new GMM is estimated for the data stream. It is assumed that if multiple concept drifts are sufficiently close together, then the change in the Cauchy–Schwarz divergence value $\Delta D_{CS_t}(p,q)$ between the GMMs (see Eq. (11)) is sufficiently stable, thereby giving $\gamma \approx 0$ with a confidence value and H_0 is true in the ADF test, and thus the data are stationary and the GMM is not updated, so concept drift detection can be neglected. However, if the value of $\Delta D_{CS_t}(p,q)$ changes abruptly, $\gamma < 0$ has a confidence value and a concept drift is signaled, so the GMM is updated.

This method is used together with the dynamic component selection process described in Section 3.1, so $D_{CS}(p,q)$ must be calculated between GMMs with different number of instances and components during specific instance temporal windows. The Cauchy–Schwarz divergence (see Eq. (10)) can operate under this condition.

After calculating the divergence, the rate of change between two distribution divergences, $\Delta D_{CS_t}(p,q)$, must be measured in order to determine their magnitudes and to define the operating range within a specific data stream or application. This trend should be analyzed and a concept drift is triggered when non-stationary behavior is detected. Non-stationary behavior is defined as the change between two distribution divergences being sufficiently different from previous measurements to cause new concept drift detection. The ADF test described in Section 2.2 is used to measure the non-stationary trends based on divergence measurements. Its implementation is described in Appendix A.

From the design of experiments described in Appendix B, test 1 and 9 are selected and summarized in Table 3.

These experimental results demonstrate that a considerable reduction in the number of false positives was obtained, regardless of the parameters used in the ADF test. Compared with the results in Table 2, the reduction in the number of concept drift detections (see n_{CD} values) was around one order of magnitude, and the GPDC+ algorithm improved the specificity compared with GDPC.

In addition, the speed of the algorithm was maintained in terms of the processing frequency, which was always between 629.98 and 1116.29 instances per second, and this is acceptable for working on a millisecond sampling time basis.

Therefore, the parameters were set as the constant (`c`) regression type, with `AIC` as the information criterion for minimizing the lag, and with 8 as the maximum lag. This lag ensured that the $\boldsymbol{D}_{CS}$ vector would require around 20 divergence measurements (see Fig. A.8) to identify non-stationary behavior, which is affordable in terms of data storage. Using these parameters, the filtering capacity when the ADF test rejected model updating when the data were non-stationary is shown in Fig. 3a. A significant improvement in the stability was obtained compared with the results shown in Fig. 2a, where only one detection per concept drift was obtained within a range of 1000 instances (see Fig. 3b).

4. Application of GDPC+ to real engineering scenarios

Condition monitoring is a useful engineering application in real scenarios, such as machining processes, where the algorithm parameters are difficult to estimate for experts and they should be obtained automatically from the data.

4.1. Data set description

In order to test the suitability of GDPC+ for use in real scenarios, a data stream with 31 machining cycles performed by a real machine-tool (Fig. 4a) was obtained as part of the Industrial Internet Consortium testbed.[1] In this case, the machine-tool was used to manufacture 31 crankshafts (Fig. 4b), where the process variables comprised the angular speed, temperature, power, and torque taken from each of the two spindle heads of the machine (Fig. 4c). Each cycle required around 30 s and 3000 instances.

The machining cycles are shown in Fig. 5 in terms of the angular speed (Fig. 5a), power (Fig. 5c). and temperature (Fig. 5e). In order to obtain the cycle details, Figs. 5b, 5d, and 5f show the data for these three variables during two machining cycles and 6000 instances.

4.2. GDPC+ concept drift results

The data stream was tested using the parameters that obtained the best performance in the ADF test (Table B.8, test 1). To study different actionable insights from the data stream, we used the GDPC+ parameter referred to as the temporal window length N (see box 1 in Fig. 1b). This parameter considers the amount of data to generate the first model, which is then used as the reference for the subsequent online analysis, i.e., the data used for training. Therefore, several values of N were selected depending on the behavior of the machine, i.e., before machining started: $N = 1000, 3000$; after one machining cycle: $N = 5000$; and after three machining cycles: $N = 10\,000$.

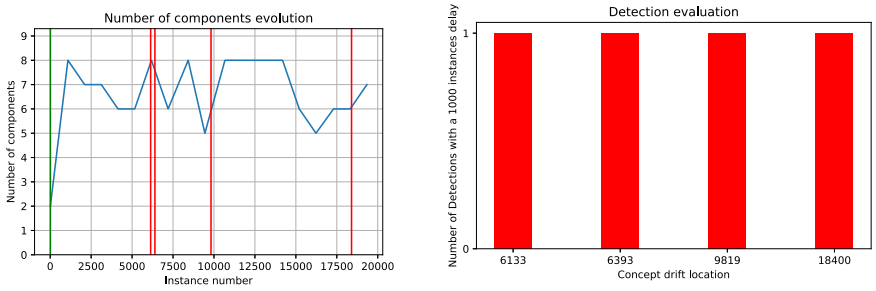
The results are shown in Table 4 and they provided the following insights.

1. Before the machining process begins (tests 1 and 2): in this case, the algorithm learned using the data from the standby machine. With $N = 1000, 3000$, the amounts of concept drifts (68) were the same with slightly different processing frequencies (688.59

[1] https://www.iiconsortium.org/smart-factory-machine-learning.htm.

Table 3
Results of the GDPC+ tests 1 and 9 with different parameters and $N = 10$.

.0 Test	`regression`	`autolag`	`maxlag`	$Inst/s$	CD_{true}	n_{CD}	Accu. [%]	Recall [%]	Spec. [%]	F-score [%]
1	c	AIC	8	779.56 ± 138.17	3.10 ± 0.94	13.00 ± 2.28	99.93 ± 0.05	99.94 ± 0.07	99.95 ± 0.05	84.92 ± 18.05
9	cdt	BIC	8	782.47 ± 127.27	3.20 ± 0.87	13.50 ± 2.16	99.91 ± 0.03	99.89 ± 0.05	99.91 ± 0.03	87.08 ± 14.42

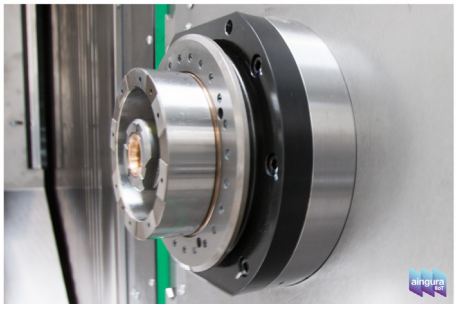


(a) (b)

Fig. 3. (a) Dynamic component selection and concept drift stabilization evolution based on the data stream. (b) Number of instances detected as concept drift with a delay of 1000 instances.

(a) Crankshaft machine-tool

(b) Crankshaft (credit: Abilene Machine (CC BY 2.0))



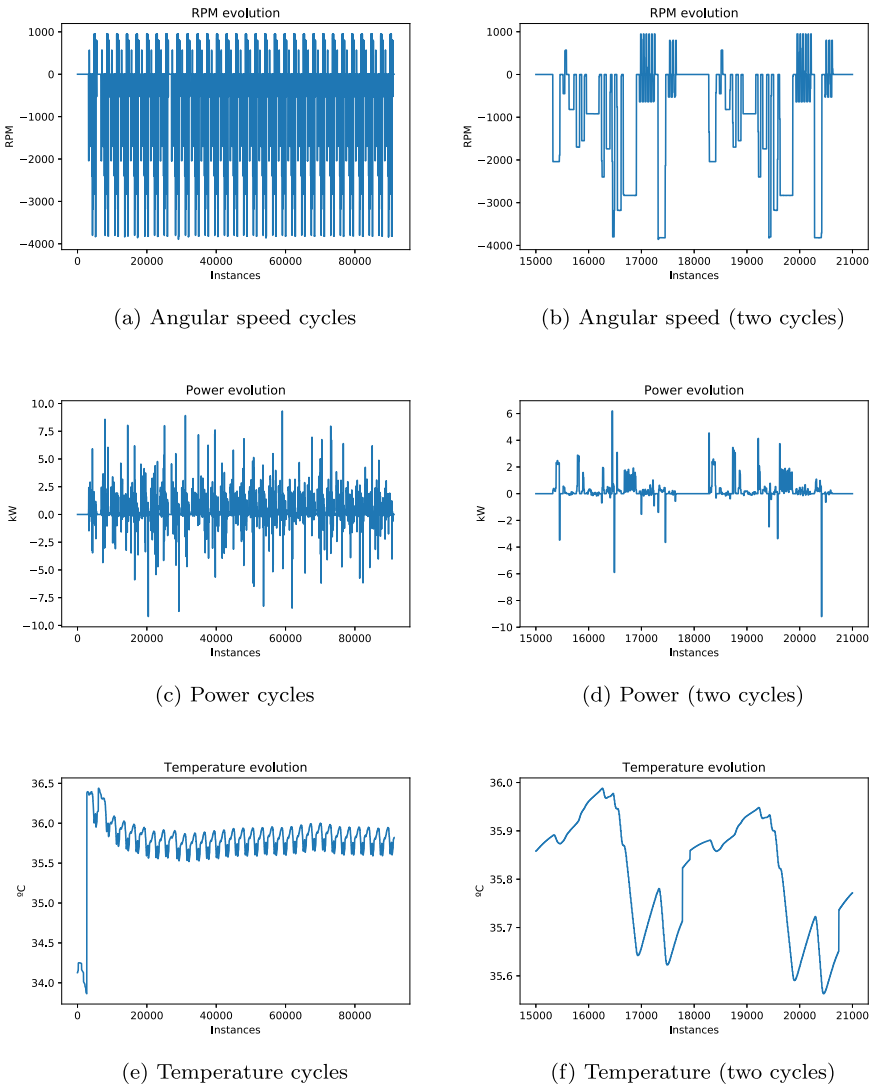
(c) Spindle head

Fig. 4. Elements of the real engineering application.

Table 4
Actionable insights results for `regression` = `c`, `autolag` = `AIC`, `maxlag` = 8.

Test	N	$Inst/s$	n_{CD}
1	1000	688.59	65.0
2	3000	698.64	64.0
3	5000	668.64	62.0
4	10 000	1737.84	0.0

and 698.64) because of the increased initial temporal window size, i.e., more time was required for training. The changes in the numbers of components are shown in Figs. 6a and 6c for $N = 1000$ and $N = 3000$, respectively. The number of components varied around six with a minimum of five for $N = 1000$ and five for $N = 3000$, and the maximum was eight in both cases. The concept drifts detected with both values for N are shown in Fig. 6b ($N = 1000$) and Fig. 6d ($N = 3000$) versus the power consumption signal, which is the most critical variable for the machining cycles. It should be noted that the drifts were located at different power values with similar spacings between them. Thus, during the timing within the cycle (the machine conducted the same process), the power fluctuations from the machine were controlled to maintain the same cutting conditions, thereby explaining the concept drifts. At this level of analysis, the N values for tests 1 and 2 configured the algorithm to detect concept drifts in the control system in terms of energy delivery, i.e., control system anomalies. In addition, other types



(a) Angular speed cycles

(b) Angular speed (two cycles)

(c) Power cycles

(d) Power (two cycles)

(e) Temperature cycles

(f) Temperature (two cycles)

Fig. 5. Data stream from a real engineering application.

of degradation could be monitored such as those due to the tool, ball-bearings, and electrical windings.

2. After one machining cycle (test 3): with this type of training, the algorithm started detecting concept drifts at similar locations to those with $N = 3000$. However, the amount of concept drifts was slightly lower (62 rather than 64). Fig. 6f shows that the regions were similar but shifted in time. The number of components varied around six (Fig. 6e), but the behavioral pattern was similar to that with $N = 1000$. Thus, small amounts of training data could be used and selection could be conducted in terms of the processing frequency, with $N = 1000$ as the best value.
3. After three machining cycles (test 4): no concept drift was detected with this parameter. The work pieces were produced without any quality issues, which is the expected result if the algorithm is applied to detect degradation of the machining cycles. It is important to note that further tests with data from more machining cycles performed in exactly the same manner and led to the detection of zero concept drifts.

4.3. Comparison of GDPC and GDPC+

In order to qualitatively analyze the differences in performance by the GDPC and GDPC+ algorithms when operating in unknown scenarios, we applied the data stream used in Section 4.2 to GDPC with different values for the number of components $K = \{3, 5\}$. The training window was set to $N = 1000$ to obtain a detailed view of the machining process with three machining cycles (see test 1, Table 4), and the results are shown in Table 5.

Table 5
Benchmarking results for GDPC and GDPC+.

Number of components	N	$Inst/s$	n_{CD}
GDPC; $d = 3$	1000	2014.84	1497
GDPC; $d = 5$	1000	1600.91	1711
GDPC+	1000	883.97	66

GDPC performed poorly in terms of the number of false positives because concept drifts were not expected to be detected, thereby yielding a high amount of concept drift detections, which were unmanageable from a machining process viewpoint. GDPC+ could obtain effective control due to the stability of the algorithm and it was two times faster than GPDC.

4.4. GDPC+ clustering results

After analyzing the performance of GDPC+ at concept drift detection, we interpreted the clusters to obtain insights into machining

Table 6
Rules extracted from GDPC+ results.

Concept drift	Component	Rule	Number of instances
3898	1	Power $\neq 0$ W Temperature $\in [34.12, 34.16]$ °C Torque ≈ 0 N m Angular speed ≈ 0 RPM	359
	2	Temperature ≈ 34.2 °C Torque > 0 N m Angular speed $\geqslant 85.6$ RPM	339
	3	Power $\neq 0$ W Temperature $\in [34.12, 34.16]$ °C Torque $\in [0.23, 0.31]$ N m and $[-0.12, -1.03]$ N m	374
	4	Other	1717
	5	Power $\geqslant 0$ Temperature $\in [34.14 - 34.25]$ °C Torque $\geqslant 0.26$ N m	42
	6	Temperature ≈ 34.2 °C Torque ≈ 0 N m	305
	7	Temperature ≈ 34.2 °C Torque $\geqslant -0.20$ N m Angular speed $\leqslant -57.12$ RPM	745
5215	1	Temperature $\geqslant 36.3$ °C Torque ≈ -4.42 N m	9
	2	Temperature $\leqslant 36.0$ °C Torque $\in [4, 6]$ N m and $[21.3, 36.0]$ N m	61
	3	Other	634
	4	Angular speed $\leqslant -1580.6$ RPM	521
	5	Temperature ≈ 36.4 °C Torque $\leqslant 0.77$ N m	60
	6	Temperature $\geqslant 36.4$ °C Torque $\leqslant -6.7$ N m	10
	7	Temperature ≈ 36.4 °C	17
6235	1	Angular speed $\neq 3820$ RPM	10
	2	Power $\leqslant 0.16$ W Angular speed $\leqslant -3820$ RPM	20
	3	Other	936
	4	Temperature $\leqslant 36.0$ °C Angular speed $\leqslant -3819$ RPM	32
	5	Angular speed ≈ -3819 RPM	18
7469	1	Other	788
	2	Power $\neq 0$ W Temperature $\neq 36.3$ °C Torque ≈ 0 N m	166
	3	Torque $\neq 0$ N m Angular speed $\geqslant 316.6$ RPM	92
	4	Temperature $\geqslant 36.4$ °C Torque $\geqslant 0$ N m Angular speed $\geqslant 0$ RPM	175
	5	Temperature ≈ 36.4 °C Angular speed $\geqslant 0$ RPM	21

(*continued on next page*)

process and its evolution, especially after the occurrence of concept drifts. As an example, we considered the results for $N = 1000$ and the concept drifts shown in Fig. 6b located at instances 3898, 5215, 6235, 7469, and 8738.

To interpret the results, we selected the cluster label as the supervised class variable to induce a set of rules using the repeated incremental pruning to produce error reduction (RIPPER) learner (Hall et al., 2009) implemented in WEKA as JRip. RIPPER constructed the classification rules based on the information gain and then simplified the rules using a pruning strategy. The rules shown in Table 6 were obtained with a classification accuracy of 100% by using all of instances as the training set.

According to the rules in Table 6, for the first concept drift at 3898, Cluster 1 represented the state of a stopped machine, which did not exist in the following concept drifts. Thus, one of the causes of concept drift was the machine starting at below the operating temperature. Clusters 2, 3, 5, 6, and 7 at 3898 were similar to cluster 2 at 5215 and they represented the machine working at low temperatures, i.e., it was starting to warm up. After a certain temperature was achieved, the current GMM was no longer valid, so a model update was launched at 6235. Therefore, clusters 1, 4, 5, 6, and 7 at 5215 can be interpreted as the machine operating during the warm up, which required around 1 s at concept drift 6235. This concept drift was similar to that located at instance 8738.

At 6235, the machine operated at high angular velocities and it also finished the machining cycle. Thus, a concept drift was triggered because the stopping of the machine was detected. Cluster 4 with low power was related to a high angular speed where only spinning was

Table 6 (*continued*).

Concept drift	Component	Rule	Number of instances
8738	1	Temperature ≈ 36.3 °C Torque $\in [-2.3, 4.2]$ N m Angular speed $\leqslant -920.2$ RPM	877
	2	Power $\neq 0$ W Temperature $\in [35, 36]$ °C Torque 0.1, 0.2, 0.3 and $\geqslant 4.6$ N m Angular speed $\in [0 - 445, 53.5]$ RPM	585
	3	Power $\leqslant 0$ W Temperature ≈ 36.0 °C Torque $\leqslant -22.8$ N m Angular speed $\in [-743, 16.8]$ RPM	672
	4	Other	1253
	5	Power $\geqslant 0.8$ W Temperature $\leqslant 36.3$ °C Angular speed ≈ -3821 RPM	53
	6	Power $\in [-3, 0.2]$ W Temperature $\leqslant 36.3$ °C Torque $\in [-3.6, -0.4]$ N m Angular speed $\in [-3818, -919.8]$ RPM	521

conducted without machining in a breaking process. This cluster was found only in 20 instances, thereby indicating the high efficiency of the machine and an excessive amount of spinning without work was avoided.

The machine was already warm at 7469 but the high variability in the cluster values in terms of the torque, power, and angular velocity indicated that the GDPC+ algorithm detected the completion of the machining cycle and that it started again. Therefore, a concept drift was launched during the transition from one machining cycle to the next. During this period, a large amount of position operations were conducted while a new work piece was introduced into the machine. Cluster 3 provided information about the acceleration required to achieve the cutting conditions. Therefore, the clusters described different stages of acceleration, where the temperature increased for cluster 2 and cluster 4 when the spindle stopped, and the spindle started to move and achieve the drilling speed for cluster 4.

The clusters obtained from the concept drift located at 8738 were similar to that at 5215, where the machining operation was conducted. Clusters 5 and 6 were similar to Clusters 1 and 4 where the machining process was performed.

The different number of clusters between concept drifts provided insights into the different machine states that were no longer available or new states that had never occurred before when a concept drift was detected. For example, between 5215 and 6235, a new cluster was generated with data related to a stopped spindle, i.e., cluster 3. This is an important feature obtained by GDPC+ because a new process behavior would not be labeled if an algorithm such as GDPC was employed with a fixed number of clusters.

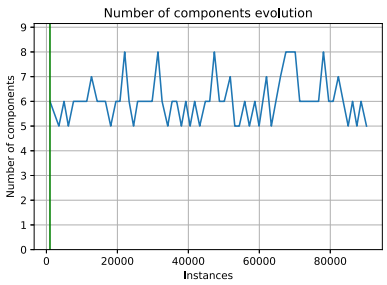
A schematic view of the interpreted results from Table 6 is presented in Fig. 7, where the angular speed from the process was used as a reference to label each cluster and each concept drift.

According to these results, selecting $N = 1000$ in the GDPC+ algorithm was sufficient to differentiate small details, such as the machine stopping, machining, changing the work-piece (cycle starts and finishes), and no-load spindle spinning. Thus, this configuration is useful for exploring patterns that may occur within a machining cycle. However, a configuration with a small temporal window, such as $N = 1000$, could lead to many false positives because of the detection of the small variations related to the different work-pieces, tool degradation, machine temperature, etc. According to Table 4, for $N = 10\,000$, there were no concept drifts and the algorithm was less sensitive to small changes that could be neglected depending on the exploratory analysis required.

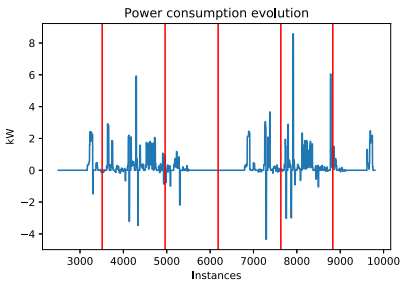
5. Conclusion and further work

In this study, we made two major improvements to the GDPC algorithm at two levels: component selection and stabilization during transients. Thus, the dynamic component selection feature can detect the best number of components for the GMM without requiring an initial expert estimation. In addition, the Cauchy–Schwarz divergence combined with the ADF test can control the amount of false positives triggered by non-stationary data during concept drift. Therefore, the GPDC+ algorithm is proposed.

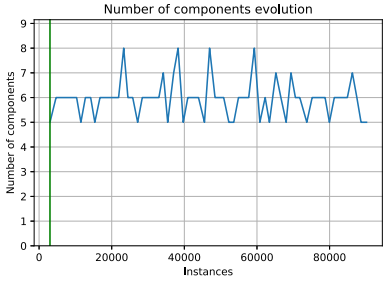
The number of parameters and their selection is reduced to only one because of these improvements. In order to use GDPC+ as a knowledge discovery tool, the main parameter that needs to be controlled is the size of the initial training window, N, which can help the end user to select an appropriate granularity depending on the condition monitoring requirements. The granularity of the analysis must be known *a*



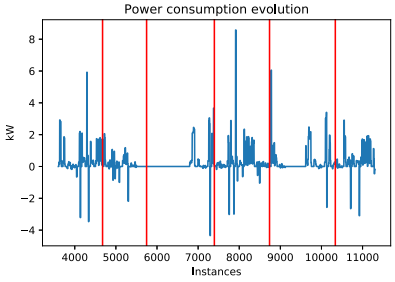
(a) $N = 1000$



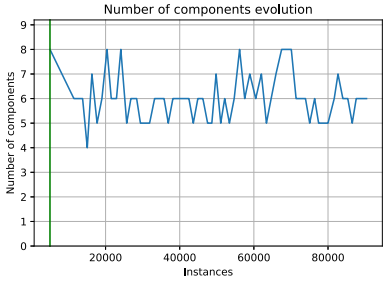
(b) Concept drifts location ($N = 1000$)



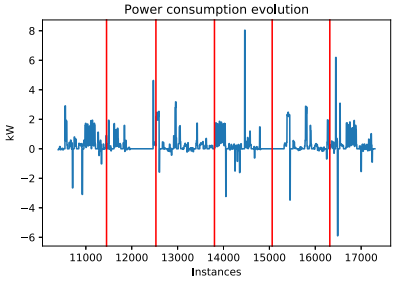
(c) $N = 3000$



(d) Concept drifts location ($N = 3000$)



(e) $N = 5000$



(f) Concept drifts location ($N = 5000$)

Fig. 6. Changes in the number of components.

priori in order to correctly define the amount of data that needs to be fed into the algorithm and to make its application highly flexible.

Based on the tests conducted with the GDPC+ algorithm, we can conclude the following.

- The concept drifts detected by the GDPC+ algorithm could be compared effectively with actual physical phenomena detected.
- The GDPC+ algorithm can operate under data stream conditions with analysis frequencies as high as 0.3 to 1.5 kHz.
- GDPC+ outperforms GDPC under completely unknown conditions, where it can update the GMM model automatically with a significant reduction in the number of false positives.
- Model updating produces a large range of behavior rules, which may contain information for each stationary state as well as information about concept drifts.

In future research, we will investigate the online implementation of the algorithm to work effectively under data stream conditions from a machine while using the lowest amount of computing power and storage as possible. Thus, the algorithm will be deployed in an embedded system with a Zynq® Ultrascale+™ MPSoC as the processing unit in a commercial device design, which will be developed by Aingura IIoT. After the algorithm operates correctly in this device, we will conduct a long-term validation study over longer production times.

CRediT authorship contribution statement

Javier Diaz-Rozo: Investigation, Writing - original draft. **Concha Bielza:** Writing - review & editing. **Pedro Larrañaga:** Writing - review & editing.

Acknowledgments

This study was supported by the Spanish Centre for the Development of Industrial Technology (CDTI) through the IDI-20180156 LearnIIoT project, and partially supported and by the Spanish Ministry of Economy and Competitiveness through the TIN2016-79684-P project. We would like to thank Etxe-Tar for advice regarding the machine hardware and Ikergune for support during data stream acquisition.

Appendix A. ADF Test implementation

In this case, the ADF test is run using the open source Statsmodels[2] Python module and its function `statsmodels.tsa.stattools.adfuller()`. According to this function, several parameters can influence the detection of concept drifts, as follows.

[2] www.statsmodels.org.

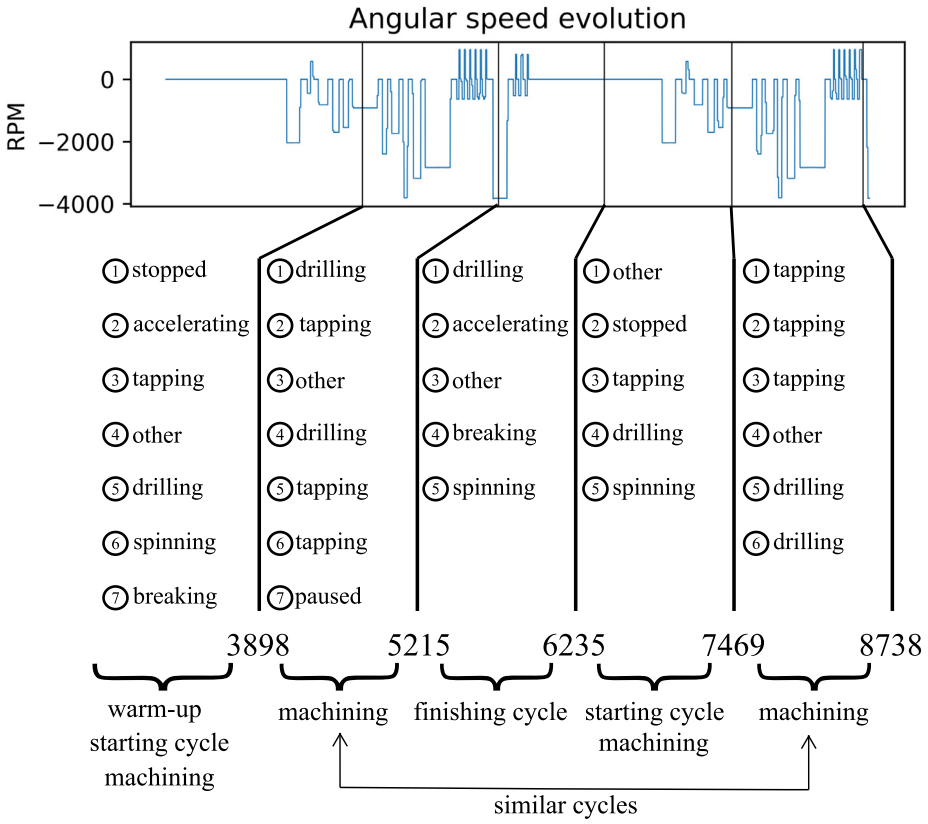


Fig. 7. Schematic interpretation of the results in Table 6.

- $\boldsymbol{D}_{CS}$ is the vector of divergence values in $\mathfrak{R}$ between the current GMM, $p(\boldsymbol{x};\boldsymbol{\Psi})$, and the updated GMM, $q(\boldsymbol{x};\boldsymbol{\Psi})$.
- `regression` is the order of Eq. (11) with the following different options.
  - `c` constant, i.e., $\beta = 0$ and $\alpha = 0$
  - `cd` constant and drift, i.e., $\beta = 0$ and $\alpha \neq 0$
  - `cdt` constant, drift, and time trend, i.e., $\beta \neq 0$ and $\alpha \neq 0$
- `maxlag` is the maximum lag in the test and it is calculated by:

$$\texttt{maxlag} = 12 \times \left(\frac{\text{length of } \boldsymbol{D}_{CS} \text{ vector}}{100} \right)^{1/4}$$

  with the shape shown in Fig. A.8. One of the objectives of GDPC+ algorithm is to significantly reduce the requirement for data storage, so the values of `maxlag` should be kept small.
- `autolag` is used to select the lag p from Eq. (11), as follows.
  - `None` when the `maxlag` value is used
  - `AIC` (Akaike's information criterion) or `BIC` when the lag is selected to minimize the corresponding information criterion

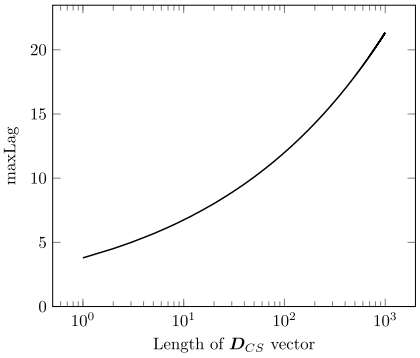


Fig. A.8. Maximum lag vs. size of the $\boldsymbol{D}_{CS}$ vector.

Table B.7
DoE: parameters and levels.

Parameters	Level 1	Level 2	Level 3
`regression`	`c`	`cd`	`cdt`
`maxlag`	8	10	12
`autolag`	`None`	`AIC`	`BIC`

Appendix B. Design of experiments for ADF

Similar to the GDPC approach proposed by Diaz-Rozo et al. (2018), a Taguchi design is used to understand how these parameters influence the results. This is a fractional factorial and orthogonal design of experiemtns (DoE) where the factors (variables for analysis) and levels (value ranges for each factor) are balanced to give all of the contrasts needed to obtain the information from the experiment. The entry key input employed to select the Taguchi design is the required number of parameters. We considered three ADF test parameters: `regression`, `maxlag`, and `autolag`, where each had three levels. Therefore, the most suitable design to fit this number of parameters and levels is the orthogonal Taguchi L9 design (Taguchi and Wu, 1979) with three parameters at four levels, i.e., a 4^3 design with nine observations. The values for each parameter are shown in Table B.7.

Random processes within the GDPC+ algorithm might influence the ADF test, so the experiments were run 10 times, before averaging the performance indicators and determining their standard deviations. The results are shown in Table B.8, where column denoted as CD_{det} indicates when GDPC+ detected a concept drift.

Table B.8
Results of the GDPC+ tests 1 to 9 with different parameters and $N = 10$.

Test	regression	autolag	maxlag	$Inst/s$	CD_{true}	n_{CD}	Accu. [%]	Recall [%]	Spec. [%]	F-score [%]
1	c	AIC	8	779.56 ± 138.17	3.10 ± 0.94	13.00 ± 2.28	99.93 ± 0.05	99.94 ± 0.07	99.95 ± 0.05	84.92 ± 18.05
2	cd	BIC	10	677.31 ± 47.33	0.70 ± 0.78	2.90 ± 1.14	100.00 ± 0.00	50.00 ± 50.00	100.00 ± 0.00	25.34 ± 26.98
3	cdt	None	12	818.66 ± 255.93	1.00 ± 0.45	1.80 ± 0.40	100.00 ± 0.00	90.00 ± 30.00	100.00 ± 0.00	38.67 ± 15.15
4	c	BIC	12	750.97 ± 69.70	0.00 ± 0.00	0.80 ± 0.40	100.00 ± 0.00	0.00 ± 0.00	100.00 ± 0.00	0.00 ± 0.00
5	cd	None	8	835.40 ± 280.89	2.60 ± 1.02	14.30 ± 3.55	99.91 ± 0.03	89.89 ± 29.96	99.91 ± 0.03	74.73 ± 26.59
6	cdt	AIC	10	732.83 ± 146.84	0.60 ± 0.66	4.80 ± 1.25	99.97 ± 0.05	50.00 ± 50.00	100.00 ± 0.00	22.67 ± 23.89
7	c	None	10	734.14 ± 43.03	1.30 ± 0.46	5.60 ± 0.49	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	48.01 ± 12.24
8	cd	AIC	12	690.87 ± 37.38	0.50 ± 1.02	1.10 ± 1.04	100.00 ± 0.00	20.00 ± 40.00	100.00 ± 0.00	15.24 ± 30.77
9	cdt	BIC	8	782.47 ± 127.27	3.20 ± 0.87	13.50 ± 2.16	99.91 ± 0.03	99.89 ± 0.05	99.91 ± 0.03	87.08 ± 14.42

According to Table B.8, test 1 obtained the highest amount of true detection results for the GPDC+ algorithm. However, the variance in CD_{true} denotes that all four concept drifts were not detected in some cases. Similar performance was obtained in test 9 but with slightly lower accuracy, specificity, and recall values. In addition, tests 1 and 9 obtained similar narrow values in terms of their variance, which indicates that they were more robust than the other options. Test 5 obtained similar performance but with only three detections.

References

Ackermann, M.R., Märtens, M., Raupach, C., Swierkot, K., Lammersen, C., Sohler, C., 2012. StreamKM++: A clustering algorithm for data streams. J. Exp. Algorithmics 17, 2–4.

Aggarwal, C.C., Yu, P.S., Han, J., Wang, J., 2003. A framework for clustering evolving data streams. In: Proceedings of the 2003 Very Large Data Bases Conference. Elsevier, pp. 81–92.

Cao, F., Ester, M., Qian, W., Zhou, A., 2006. Density-based clustering over an evolving data stream with noise. In: Proceedings of the 2006 International Conference on Data Mining. Society for Industrial and Applied Mathematics, vol. 6, pp. 328–339.

Chen, Y., Tu, L., 2007. Density-based clustering for real-time stream data. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM Press, pp. 133–142.

Cheung, Y.W., Lai, K.S., 1995. Lag order and critical values of the augmented Dickey–Fuller test. J. Bus. Econom. Statist. 13 (3), 277–280.

Dempster, A.P., Laird, N.M., Rubin, D.B., 1977. Maximum likelihood from incomplete data via the EM algorithm. J. R. Stat. Soc. Ser. B Stat. Methodol. 39 (1), 1–38.

Diaz-Rozo, J., Bielza, C., Larrañaga, P., 2017. Machine learning-based CPS for clustering high throughput machining cycle conditions. Procedia Manuf. 10, 997–1008.

Diaz-Rozo, J., Bielza, C., Larrañaga, P., 2018. Clustering of data streams with dynamic Gaussian mixture models: An IoT application in industrial processes. IEEE Internet Things J. 5 (5), 3533–3547.

Dickey, D.A., Fuller, W.A., 1979. Distribution of the estimators for autoregressive time series with a unit root. J. Amer. Statist. Assoc. 74 (366), 427–431.

Gama, J., Rodrigues, P.P., Lopes, L., 2011. Clustering distributed sensor data streams using local processing and reduced communication. Intell. Data Anal. 15 (1), 3–28.

Gama, J., Žliobaite, I., Bifet, A., Pechenizkiy, M., Bouchachia, A., 2014. A survey on concept drift adaptation. ACM Comput. Surv. 46 (4), 44:1–44:37.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H., 2009. The WEKA data mining software: An update. SIGKDD Explor. 11 (1), 10–18.

Har-Peled, S., Mazumdar, S., 2004. On coresets for k-means and k-medians clustering. In: Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing. ACM, pp. 291–300.

Kampa, K., Hasanbelliu, E., Principe, J.C., 2011. Closed-form Cauchy-Schwarz PDF divergence for mixture of Gaussians. In: Proceedings of the 2011 International Joint Conference on Neural Networks. IEEE, pp. 2578–2585.

Khamassi, I., Sayed-Mouchaweh, M., Hammami, M., Ghédira, K., 2018. Discussion and review on evolving data streams and concept drift adapting. Evolv. Syst. 9 (1), 1–23.

Kranen, P., Assent, I., Baldauf, C., Seidl, T., 2011. The ClusTree: Indexing micro-clusters for anytime stream mining. Knowl. Inf. Syst. 29 (2), 249–272.

Larrañaga, P., Atienza, D., Diaz-Rozo, A., Puerto-Santana, C.E., Bielza, C., 2019. Industrial Applications of Machine Learning. CRC Press.

Lughofer, E., 2012. A dynamic split-and-merge approach for evolving cluster models. Evolv. Syst. 3 (3), 135–151.

Lughofer, E., Sayed-Mouchaweh, M., 2015. Autonomous data stream clustering implementing split-and-merge concepts – Towards a plug-and-play approach. Inform. Sci. 304, 54–79.

McLachlan, G., Peel, D., 2004. Finite Mixture Models. John Wiley & Sons.

Mushtaq, R., 2011. Augmented Dickey Fuller Test. SSRN Scholarly Paper ID 1911068, Social Science Research Network; Université Paris.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: Machine learning in Python. J. Mach. Learn. Res. 12, 2825–2830.

Pratama, M., Dimla, E., Lai, C.Y., Lughofer, E., 2019a. Metacognitive learning approach for online tool condition monitoring. J. Intell. Manuf. 30 (4), 1717–1737.

Pratama, M., Dimla, E., Tjahjowidodo, T., Pedrycz, W., Lughofer, E., 2019b. Online tool condition monitoring based on parsimonious ensemble+. IEEE Trans. Cybern. 1–14.

Rodrigues, P., Gama, J., Pedroso, J., 2008. Hierarchical clustering of time-series data streams. IEEE Trans. Knowl. Data Eng. 20 (5), 615–627.

Schlimmer, J.C., Granger, R.H., 1986. Incremental learning from noisy data. Mach. Learn. 1 (3), 317–354.

Schwarz, G., 1978. Estimating the dimension of a model. Ann. Statist. 6 (2), 461–464.

Sidhu, P., Bhatia, M.P.S., 2018. A novel online ensemble approach to handle concept drifting data streams: Diversified dynamic weighted majority. Int. J. Mach. Learn. Cybern. 9 (1), 37–61.

Spiliopoulou, M., Ntoutsi, I., Theodoridis, Y., Schult, R., 2006. MONIC: Modeling and monitoring cluster transitions. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM Press, pp. 706–711.

Taguchi, G., Wu, Y., 1979. Introduction to Off-Line Quality Control. Central Japan Quality Control Association.

Widmer, G., Kubat, M., 1996. Learning in the presence of concept drift and hidden contexts. Mach. Learn. 23 (1), 69–101.

Yang, H., Li, P., Guo, X., Chen, H., Lin, Z., 2018. A review: The effects of imperfect data on incremental decision tree. Int. J. Inf. Commun. Technol. 12 (1–2), 162–174.

Zhang, T., Ramakrishnan, R., Livny, M., 1997. BIRCH: a new data clustering algorithm and its applications. Data Min. Knowl. Discov. 1 (2), 141–182.

Zhou, A., Cao, F., Qian, W., Jin, C., 2008. Tracking clusters in evolving data streams over sliding windows. Knowl. Inf. Syst. 15 (2), 181–214.