Applied Soft Computing 62 (2018) 119–133

Contents lists available at ScienceDirect

Applied Soft Computing

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

DetectA: abrupt concept drift detection in non-stationary environments

Tatiana Escovedo [a], Adriano Koshiyama [b,*], Andre Abs da Cruz [c], Marley Vellasco [a]

[a] Department of Electrical Engineering, PUC-Rio, R. Marquês de São Vicente, 225, Gávea, 22430-060, Rio de Janeiro, Brazil
[b] Department of Computer Science, University College London, London, United Kingdom
[c] MDC Partners, Antwerp, Belgium

ARTICLE INFO

Article history:
Received 13 May 2017
Received in revised form 10 August 2017
Accepted 16 October 2017
Available online 24 October 2017

Keywords:
Concept drift
Drift detection
Proactive approach

ABSTRACT

Almost all drift detection mechanisms designed for classification problems work reactively: after receiving the complete data set (input patterns and class labels) they apply a sequence of procedures to identify some change in the class-conditional distribution – a concept drift. However, detecting changes after its occurrence can be in some situations harmful to the process under analysis. This paper proposes a proactive approach for abrupt drift detection, called DetectA (Detect Abrupt Drift). Briefly, this method is composed of three steps: (i) label the patterns from the test set (an unlabelled data block), using an unsupervised method; (ii) compute some statistics from the train and test sets, conditioned to the given class labels for train set; and (iii) compare the training and testing statistics using a multivariate hypothesis test. Based on the results of the hypothesis tests, we attempt to detect the drift on the test set, before the real labels are obtained. A procedure for creating datasets with abrupt drift has been proposed to perform a sensitivity analysis of the DetectA model. The result of the sensitivity analysis suggests that the detector is efficient and suitable for datasets of high-dimensionality, blocks with any proportion of drifts, and datasets with class imbalance. The performance of the DetectA method, with different configurations, was also evaluated on real and artificial datasets, using an MLP as a classifier. The best results were obtained using one of the detection methods, being the proactive manner a top contender regarding improving the underlying base classifier accuracy.

1. Introduction

Most real-world problems experience a phenomenon known as concept drift [13]. This circumstance occurs in datasets where the joint probability distribution changes arbitrarily over time [34], such as a switch in the conditional probability distribution on a classification problem, or a modification of some moment (such as mean and variance) on a time series forecasting problem [37,15]. Formally speaking, considering the posterior probability of a sample x belonging to a class y, according to [9] concept drift is any scenario in which this probability changes over time, that is: $(Pt+1(y|x) \neq Pt(y|x))$. We can also define concept drift, in a supervised learning scenario, when the relationship between the input data and the target variable changes over time [13]. An environment from which this kind of data is obtained is considered a non-stationary environment. When concepts often evolve, the system may be unable to adapt to the new information, hence dramatically deteriorating its performance [17,40].

One of the most relevant non-stationary scenarios involves classification problems, such as network intrusion detection. This issue is commonly composed by an opponent (a human or a robot) that is seeking ways to deceive the protection method (classifier), introducing a drift component in the test set (new ways to break into the network, for example). Most classifiers are built directly from scratch and readily applied to data not yet seen. If the data distribution varies over time, then the model should be refitted to prevent harmful decisions. However, making constant adjustments to the model is ineffective, and becomes unfeasible in high dimensional and scalable problems [41].

Another practical example of concept drift mentioned in [19,13] is detecting and filtering out spam e-mails. The significance of the two classes "spam" and "ham" may vary over time. They are user specific, and user preferences also vary over time. Moreover, the variables used at time t to classify spam may be irrelevant at $t+k$.

* Corresponding author.
E-mail addresses: tatiana@inf.puc-rio.br (T. Escovedo), a.koshiyama@cs.ucl.ac.uk (A. Koshiyama), andrevpuc@gmail.com (A.A. da Cruz), marley@ele.puc-rio.br (M. Vellasco).

https://doi.org/10.1016/j.asoc.2017.10.031
1568-4946/

In this way, the classifier must deal with the "spammers" who will keep creating new forms to trick the classifier into labelling a spam as a legitimate e-mail. Other examples are consumption patterns that can vary according to season or availability of alternative products, weather prediction, inflation rate, traffic monitoring, and medical decision aiding [37,13,34,17,39].

In general, concept drifts that can occur in the real world can be classified into several types, according to [37,42,13,35,17,29], being the two main types abrupt and gradual. To differentiate both, assume the existence of two data sources, S_1 and S_2:

- **Abrupt**: occurs when a concept A is abruptly switched by another concept B, that is, at time t the origin S_1 is suddenly replaced by source S_2. Some authors, like [17,29] refers this drift type as **sudden** drift.
- **Gradual**: occurs when a concept A is being exchanged for the other B gradually. In this case, while there is no definitive change from concept A to concept B, more and more occurrences of B and fewer occurrences of A are observed. Both sources S_1 and S_2 are active, but as time passes, the sampling probability of the origin S_1 decreases as the sampling probability of the source S_2 increases. At the beginning of this drift, before more instances are observed, one case of the S_2 source can be easily mistaken for random noise.

Models for learning in non-stationary environments may or may not contain drift detection mechanisms. Most of the models found in the literature assume that the changes take place in a hidden context external to the model itself and, therefore, the drift cannot be predicted. For this reason, these models use the passive or reactive approach, where the model's performance is firstly verified and, if a drift is detected, then the model reacts after the error has occurred. Another method would be to do it proactively, that is, by detecting the occurrence of drift in the input data before they are submitted for prediction (i.e., before receiving the true labels). The proactive approach can be more satisfactory since it is possible to refit the model or tweak the data previously, and thereby hope to better cope with the new scenario and avoid a miss-classification [13].

Rooted in the proactive approach, the main objective of this work is to propose a concept drift detection mechanism with the ability to anticipate eventual drifts. As this method is specialised in detecting abrupt drifts, we called this mechanism as DetectA, referring to the terms Detect an Abrupt Drift. In order to assess our contribution, we first performed a sensitivity analysis on the number of attributes, patterns, imbalance rate, and so forth, using artificial datasets with pre-defined abrupt drifts in certain classes and instants. Then, we conducted experiments with artificial and real datasets to verify its performance when coupling with a classifier in non-stationary environments, comparing several different approaches of the proposed method.

We should mention that this work is an extension of a previous contribution [43]. We unfolded this preceding work in multiple directions, but mainly in three aspects: (i) proposing a reactive version of DetectA; (ii) describing two different training strategies when the proactive version of DetectA is deployed jointly with a classifier; and (iii) applying the reactive and proactive versions of DetectA on well-established benchmark datasets from the concept drift detection literature. In (i) we describe a more conventional version of DetectA, by applying only its statistical component to flag concept drifts after having complete knowledge of the new batch labels. Item (ii) refers to different ways to hedge the classifier when the proactive version has signalled a potential drift; hence, we propose two learning strategies that can aid in this task. Finally, (iii) presents new results on coupling a multi-layer perceptron classifier with variations of DetectA (reactive and proactive) and the proposed training strategies (item ii), to provide evidence that when DetectA is deployed jointly with a classifier, it can improve its performance on non-stationary learning scenarios.

We structured this paper in four additional sections. Section 2 presents the fundamentals of concept drift, including definitions and summary of the main algorithms of drift detection in the literature, in order to clarify their main contributions. Section 3 presents the proposed drift detection mechanism (DetectA), outlining the main distinctions between a reactive and proactive detection method. Section 4 presents the sensitivity analysis and discussions on the experiments performed with benchmark databases. Finally, Section 5 concludes this work and discusses future works.

2. Background and literature review

2.1. Existing models for concept drift detection

The term "Change Detection" or "Drift Detection" refers to techniques and mechanisms for detecting drift/change by identifying change points or small intervals during which variations occur, such that the existing models can no longer be effective to predict the behaviour of the current data [13]. Concept drift detectors are methods that can signal that data distributions are changing, based on information about classifier's performance or the incoming data. Such signals usually trigger the need to update, replace or retrain the model [17].

Concept drift can occur in several learning problems, such as classification, regression and time series forecasting. However, this work focuses specifically on classification problems. Typically, concept drift detectors are used together with a classification module, and they measure various properties of the data, such as standard deviation [12], predictive error [5], instance distribution [33], or stability [38]. Any changes observed in these properties are attributed to the potential presence of drift [29].

Several reactive drift detection mechanisms have already been proposed in the literature and can be used to execute the learning process in conjunction with a predictive model. In the case of classification problems, the classifier typically provides the class prediction for each input pattern and then compares its response with the correct class label received to see if the classifier has hit or miss each prediction. Table 1 shows some of these detection methods.

These drift detection methods, as well as most of the methods found in the literature, work reactively, that is, they act after the occurrence of the drift and model error since they depend on the actual class labels of the input patterns. In classification problems, after receiving the complete dataset (patterns and class labels for the training and test sets), the detector applies a sequence of procedures to identify some change in the conditional class distribution – a concept drift.

Few papers use a proactive approach, like in [18] that applies principal component analysis (PCA) for extracting characteristics before change detection. The authors discuss and show evidence that components with lower variance should be stored as extracted features since they are more likely to be affected by the change. The authors then choose a change detection criterion, based on the semiparametric log-likelihood function, which is sensitive to shifts in the mean and variance of multidimensional distributions. Other contribution, described in [44], proposes a new recurrent drift detector which incorporates historical drift rate information that is accurate for streams with reoccurring volatility trends. They have used synthetic and real data to compare their method with three state-of-art detection mechanism, being able to show that their technique is able to lower the rate of false positives. How-

Table 1
Summary of reactive detection methods.

Method	Reference	Drift Type	Metric
Drift Detection Method (DDM)	[12]	Abrupt	Tracks online error and defines tolerance zones
Early Drift Detection Method (EDDM)	[3]	Gradual	Same as DDM, but working with error variance
Statistical Test of Equal Proportions (STEPD)	[27]	Abrupt/Gradual	Compares the accuracy of the same model using different data history sizes
Paired Learners (PL)	[2]	Abrupt/Gradual	Compares the accuracy between two models with different time windows
Exponentially Weighted Moving Average (EWMA)	[30]	Abrupt/Gradual	Monitors the mean of a sequence of random variables
EWMA for Concept Drift Detection (ECDD)	[30]	Abrupt/Gradual	Same as EWMA, but controlling the false positive rate
Resampling	[23]	Abrupt/Gradual	Uses random permutations of the samples, which produce various training-testing splits from the stream of data. The results suggest that it is more robust for noisy changes.
Hierarchical CDT (H–CDT)	[1]	Abrupt	Uses the Hotelling test to check if the present contents in stream before and after the modification differs
Error distance based approach for drift detection and monitoring (EDIST)	[16]	Gradual	Same as EDDM, but takes two data window and traces concept drift by maintaining two windows: one global sliding window and another to store the present example
Hoeffding's Bounds Drift Detection Method (HDDM)	[11]	Abrupt/Gradual	Same as DDM and EDDM, but applying non-parametric methods based on Hoeffding's Bounds
Adaptive cumulative windows model (ACWM)	[31]	Abrupt/Gradual	The online monitoring of the distance between data distributions is evaluated using a dissimilarity measure based on the asymmetry of the Kullback–Leibler divergence
GraphPool Framework	[45]	Recurrent	It extracts a concept representation from the current batch considering the correlation among features. Then, compares the current batch representation to the concept representations in the pool using a statistical multivariate likelihood test
Multidimensional Fourier Transform (MFDT)	[46]	Abrupt/Gradual	Employ Shannon's and Von Neumann's Entropies to quantify variations in data spaces. Also, MDFT allows univariate streams to be reconstructed in phase spaces so their data dependencies can be analyzed decide over concept drifts

ever, authors point out a limitation of the technique, since it only uses drift interval information for predicting future drift locations by matching the drift rate patterns to the pattern network.

Compared to these few proactive methods, DetectA takes a different approach to proactive detection: after grouping the data, the means and covariance matrices of the previous and current blocks are compared using statistical tests, and if this difference exceeds a certain threshold, drift is signalled. Through the information that comes from these statistical tests, we can devise manners to alleviate the impact of such expected drift, being a major difference when compared to current proactive approaches. The DetectA model will be further detailed in Section 3.

As we are going to focus on abrupt drift detection, next subsections present definitions and statistical methods to detect eventual drifts of this kind.

2.2. Wide-sense concept drift detection

Assuming that $\boldsymbol{X}$ conditioned to class k follows a Multivariate Normal Distribution [14], i.e., $\boldsymbol{X}|C_k \sim N_J\left(\boldsymbol{\mu_{C_k}}, \Sigma_{C_k}\right)$, then its conditional joint probability density function is given by:

$$f_{\boldsymbol{X}}(\boldsymbol{x}|C_k) = \frac{1}{(2\pi)^{J/2}|\boldsymbol{\Sigma_{Ck}}|^{J/2}} e^{-\left(\boldsymbol{x}-\boldsymbol{\mu_{C_k}}\right)^T \boldsymbol{\Sigma}_{\boldsymbol{C_k}}^{-1}\left(\boldsymbol{x}-\boldsymbol{\mu_{C_k}}\right)\left(\frac{1}{2}\right)} \quad (1)$$

where:

- $\boldsymbol{\mu_{C_k}} = \left[\mu_{X_1|C_k}, \mu_{X_2|C_k}, \ldots, \mu_{X_J|C_k}\right]^T$ is the conditional mean vector, in which the j-th entry is the conditional mean for the k-th class of the j-th random variable.
- $\Sigma_{C_k} = \begin{bmatrix} \sigma^2_{X_1|C_k} & \sigma_{X_1,X_2|C_k} & \ldots \sigma_{X_1,X_J|C_k} \\ \sigma_{X_2,X_1|C_k} & \ddots & \vdots \\ \vdots & \ldots & \sigma^2_{X_J|C_k} \\ \sigma_{X_J,X_1|C_k} & & \end{bmatrix}$ is the conditional covariance matrix, composed of variance $\left(\sigma^2_{X_j|C_k}\right)$ and covariance $\left(\sigma_{X_l,X_j|C_k}\right)$ terms related to the k-th class. By definition, Σ_{C_k} is symmetric and positive, with $|\Sigma_{C_k}|$ representing the determinant of the covariance matrix.

Based on these parameters – mean vector and covariance matrix – we explore a less strict condition of abrupt concept drift when compared to the one provided in [13] – that is, instead of looking to the harder to estimate joint distribution we rather focus our attention on parameters that are easy to handle statistically.

In this sense, no abrupt drift is observed if both of the following equalities:

$$\boldsymbol{\mu_{C_k}}(t) = \boldsymbol{\mu_{C_k}}(t+1) \quad (2)$$

$$\boldsymbol{\Sigma_{C_k}}(t) = \boldsymbol{\Sigma_{C_k}}(t+1) \quad (3)$$

holds. However, the population parameters $\boldsymbol{\mu_{C_k}}(t)$ and $\boldsymbol{\Sigma_{C_k}}(t)$ are rarely known. In fact, what is observed is a random sample of $\boldsymbol{X}_1^{(t)}, \ldots, \boldsymbol{X}_n^{(t)}$ from the population under analysis. From this sample it is viable to estimate $\boldsymbol{\mu_{C_k}}(t)$ and $\boldsymbol{\Sigma_{C_k}}(t)$ using the maximum likelihood estimators [14] $\bar{\boldsymbol{X}}_{C_k}(t)$ and $\boldsymbol{S}_{C_k}(t)$ respectively, where $\bar{\boldsymbol{X}}_{C_k}(t)$ is a vector composed of arithmetical averages $\left(\bar{\boldsymbol{X}}_{j|C_k}(t)\right)$ for each feature, while $\boldsymbol{S}_{C_k}(t)$ is a matrix, with the main diagonal composed of variances related to the j-th feature $\left(s^2_{j|C_k}(t)\right)$, and the off-diagonal elements are the sample covariance between two different features $\left(s_{j,l|C_k}(t)\right)$. Clearly, all of these values are measured at time t and conditioned to class k.

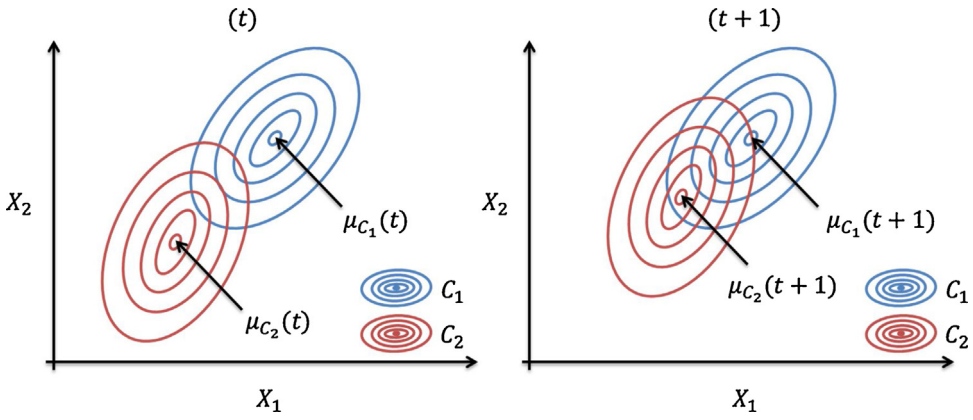


Fig. 1. An example of an abrupt concept drift in the mean vector.

Based on these topics, next topics present a better characterization of abrupt drift and hypothesis tests to detect its occurrence.

2.2.1. Concept drift and hypothesis test on conditional mean vector

Definition 1. An abrupt concept drift in the conditional mean vector occurs when the following equality does not hold:

$$\boldsymbol{\mu}_{\boldsymbol{C_k}}(t) = \boldsymbol{\mu}_{\boldsymbol{C_k}}(t+1) \tag{4}$$

then the mean vectors differ from time t to t + 1.

Observation: the equality in Eq. (4) keeps the same principle of a hypothesis test: first, it is necessary to find an estimator for $\boldsymbol{\mu}_{\boldsymbol{C_k}}(t)$, $\boldsymbol{\mu}_{\boldsymbol{C_k}}(t+1)$ and for other quantities involved; second, submit these to a hypothesis test that measures the probability of non-rejecting the $\boldsymbol{\mu}_{\boldsymbol{C_k}}(t+1)$equality between $\boldsymbol{\mu}_{\boldsymbol{C_k}}(t)$.

If this probability is less or equal to the significance level (α), then an abrupt concept drift in the conditional mean vector has occurred. Fig. 1 exhibits an abrupt concept drift C_2 occurring in the conditional mean vector of class. As can be noted, the decision boundary between the classes have changed, making the problem more challenging; since the current classifier was trained based on the previous boundary, it will tend to erroneous classify more elements as class 1 when they have been generated from class 2.

When Eq. (4) needs to be evaluated from data, we apply the Hotelling's T^2 [14]. Suppose two competing hypothesis:

$$H_0 : \boldsymbol{\mu}_{\boldsymbol{C_k}}(t) = \boldsymbol{\mu}_{\boldsymbol{C_k}}(t+1)$$

$$H_1 : \boldsymbol{\mu}_{\boldsymbol{C_k}}(t) \neq \boldsymbol{\mu}_{\boldsymbol{C_k}}(t+1)$$

If H_0 is rejected, then we consider that an abrupt concept drift in the conditional mean vector has occurred. Given two random samples $\boldsymbol{X}_1^{(t)}, \ldots, \boldsymbol{X}_{n_1}^{(t)}$ and $\boldsymbol{X}_1^{(t+1)}, \ldots, \boldsymbol{X}_{n_2}^{(t+1)}$ of size n_1 and n_2 respectively, with $\boldsymbol{X}^{(t)}|C_k^{(t)} \sim N_J\left(\boldsymbol{\mu}_{\boldsymbol{C_k}}(t), \boldsymbol{\Sigma}_{\boldsymbol{C_k}}(t)\right)$ and $\boldsymbol{X}^{(t+1)}|C_k^{(t+1)} \sim N_J\left(\boldsymbol{\mu}_{\boldsymbol{C_k}}(t+1), \boldsymbol{\Sigma}_{\boldsymbol{C_k}}(t+1)\right)$. The appropriate test statistic is:

$$T^2 = \left(\bar{\boldsymbol{X}}_{C_k}(t) - \bar{\boldsymbol{X}}_{C_k}(t+1)\right)^T \left(\frac{\boldsymbol{S}_{\boldsymbol{C_k}}(t)}{n_1} + \frac{\boldsymbol{S}_{\boldsymbol{C_k}}(t+1)}{n_2}\right)^{-1} \left(\bar{\boldsymbol{X}}_{C_k}(t) - \bar{\boldsymbol{X}}_{C_k}(t+1)\right) \tag{5}$$

which compares the sample mean vectors $\bar{\boldsymbol{X}}_{C_k}(t)$ and $\bar{\boldsymbol{X}}_{C_k}(t+1)$ (estimators for $\boldsymbol{\mu}_{\boldsymbol{C_k}}(\boldsymbol{t})$ and $\boldsymbol{\mu}_{\boldsymbol{C_k}}(t+1)$) in two different instants (blocks), in such manner that if T^2 is "too high", then H_0 must be rejected. $\boldsymbol{S}_{\boldsymbol{C_k}}(t)$ and $\boldsymbol{S}_{\boldsymbol{C_k}}(t+1)$ represent the samples covariance matrices. Given the assumptions behind the random samples, and under H_0, the test statistic $\frac{(n_1+n_2)-J-1}{(n_1-n_2-2)J}T^2$ follows a F distribution with J and $(n_1+n_2)-J-1$ degrees of freedom (J being the number of features). With this knowledge, it is possible to set a rejection zone for H_0 to identify abrupt concept drifts in the mean. Below are the steps to apply Hotteling's T^2 test:

1. Compute $\bar{\boldsymbol{X}}_{C_k}(t)$, $\bar{\boldsymbol{X}}_{C_k}(t+1)$, $\boldsymbol{S}_{C_k}(t)$ and $\boldsymbol{S}_{C_k}(t+1)$.
2. Calculate the test statistic T^2 (Eq. (5)).
3. Drift if $\frac{(n_1+n_2)-J-1}{(n_1-n_2-2)J}T^2 > F_{J,(n_1+n_2)-J-1}(\alpha)$, where $F_{J,(n_1+n_2)-J-1}(\alpha)$ is the upper $(1-\alpha)$ percentile of the $F_{J,(n_1+n_2)-J-1}$ distribution.

2.2.2. Concept drift and hypothesis test on conditional covariance matrix

Definition 2. An abrupt concept drift in the conditional covariance matrix occurs when the equality does not hold:

$$\boldsymbol{\Sigma}_{\boldsymbol{C_k}}(t) = \boldsymbol{\Sigma}_{\boldsymbol{C_k}}(t+1) \tag{6}$$

then the covariance matrix differs from time t to t + 1.

Comment: similarly, this equality can be checked using a hypothesis test. Two possible abrupt concept drifts in the conditional covariance matrix are exhibited in Fig. 2. The conditional variance of X_2 in C_2 has grown, dilating the contour curves in this direction; in C_1, the conditional covariance between X_1 and X_2fades away, implying less association between both variables, thereby rotating the contour curves.

Box-M test [14] is a hypothesis test used to identify significant differences between covariance matrices of normally distributed random variables. This statistical test supposes two competing hypothesis:

$$H_0 : \boldsymbol{\Sigma}_{\boldsymbol{C_k}}(t) = \boldsymbol{\Sigma}_{\boldsymbol{C_k}}(t+1)$$

$$H_1 : \boldsymbol{\Sigma}_{\boldsymbol{C_k}}(t) \neq \boldsymbol{\Sigma}_{\boldsymbol{C_k}}(t+1)$$

If H_0 is rejected, then it is considered that an abrupt concept drift in the conditional covariance matrix has occurred. Given two random samples $\boldsymbol{X}_1^{(t)}, \ldots, \boldsymbol{X}_{n_1}^{(t)}$ and $\boldsymbol{X}_1^{(t+1)}, \ldots, \boldsymbol{X}_{n_2}^{(t+1)}$, with $\boldsymbol{X}^{(t)}|C_k^{(t)} \sim N_J\left(\mu_{C_k}(t), \Sigma_{C_k}(t)\right)$ and $\boldsymbol{X}^{(t+1)}|C_k^{(t+1)} \sim N_J\left(\mu_{C_k}(t+1), \Sigma_{C_k}(t+1)\right)$, consider the likelihood ratio test as:

$$\Lambda = \left(\frac{\det\left(\boldsymbol{S}_{\boldsymbol{C_k}}(t)\right)}{\det\left(\boldsymbol{S}_{\boldsymbol{pool}}\right)}\right)^{\frac{n_1-1}{2}} * \left(\frac{\det\left(\boldsymbol{S}_{\boldsymbol{C_k}}(t+1)\right)}{\det\left(\boldsymbol{S}_{\boldsymbol{pool}}\right)}\right)^{\frac{n_2-1}{2}} \tag{7}$$

where $\boldsymbol{S}_{\boldsymbol{pool}} = \frac{(n_1-1)\boldsymbol{S}_{\boldsymbol{C_k}}(t)+(n_2-1)\boldsymbol{S}_{\boldsymbol{C_k}}(t+1)}{(n_1-1)+(n_2-1)}$ is the pooled covariance matrix. Again, $\boldsymbol{S}_{\boldsymbol{C_k}}(\boldsymbol{t})$ and $\boldsymbol{S}_{\boldsymbol{C_k}}(t+1)$ represent the samples covariance matrices and n_1 and n_2 the number of samples at t and $t+1$, respectively. Box-M test uses the statistic:

$$M = -2\ln\Lambda \tag{8}$$

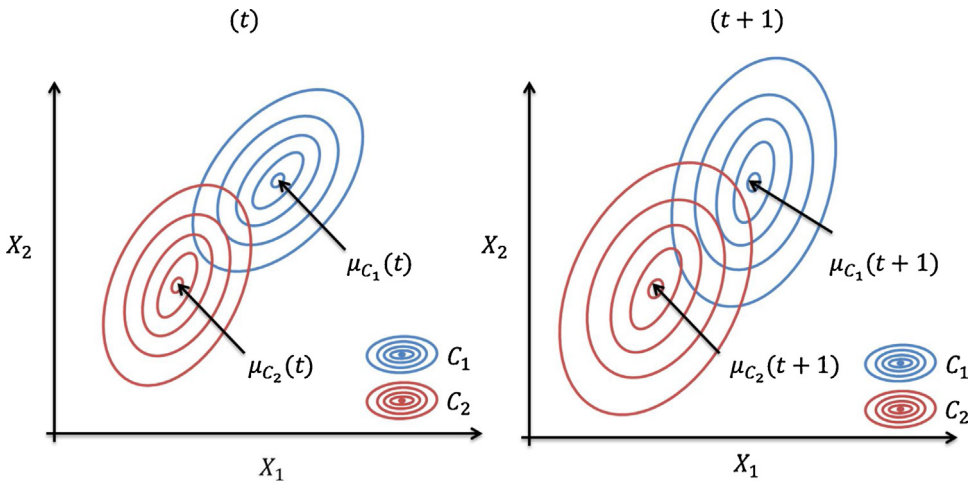


Fig. 2. Examples of an abrupt concept drift in the covariance matrix.

if the null hypothesis is true, the covariance matrices may not substantially differ and, consequently, these will not differ so much from the pooled covariance matrix. Finally, define the quantity u by:

$$u = \left[\frac{1}{(n_1-1)} + \frac{1}{(n_2-1)} - \frac{1}{(n_1-1)+(n_2-1)}\right]\left[\frac{2J^2+3J-1}{6(J+1)}\right] \quad (9)$$

then, $C = (1-u)M$ approximately follows a χ^2 distribution with $\frac{1}{2}J(J+1)$ degrees of freedom. Therefore, it is possible to establish a rejection zone for H_0 to identify abrupt concept drifts in the covariance matrix. As stated in [14], the χ^2 approximation works well when n_1, $n_2 > 20$ and the number of features is below 5. In some case studies, the datasets have more than 5 features, so instead of using the χ^2 distribution, we prefereed the approximation via F distribution (following recommendations by [22]). The necessary steps to execute Box-M test are presented below:

1. Calculate the $\mathbf{S}_{\mathbf{C_k}}(\mathrm{t})$, $\mathbf{S}_{\mathbf{C_k}}(\mathrm{t}+1)$ and $\mathbf{S}_{\mathbf{p}ool}$.
2. Compute M (Eq. (8)) and u (Eq. (9)).
3. Drift if $C > \chi^2_{\frac{1}{2}J(J+1)}(\alpha)$, where $\chi^2_{\frac{1}{2}J(J+1)}(\alpha)$ is the upper (1-α) percentile of the $\chi^2_{\frac{1}{2}J(J+1)}$ distribution.

Based on these two previous hypothesis tests next section describes the proposed DetectA method, with some adaptions to move from reactive to proactive drift detection.

3. Abrupt drift detection method: DetectA

The proposed drift detection method, called DetectA (Detect Abrupt Drift) is basically composed of three steps: (i) the test set patterns are labelled using an unsupervised grouping method; (ii) a series of statistics are computed from the training and test set, both conditioned to the labels settled in the previous stage; then (iii) the conditional means and covariance matrices are compared to the training and test set using the multivariate hypothesis tests outlined in the previous section. After getting the results of such tests, a decision is taken concerning the occurrence or not of drift. In the case of drift, some measures can be applied to adjust and improve the learning process.

Some notation is necessary for the sake of comprehension: consider a collection of n patterns at time t $(\boldsymbol{x}_1(t), \ldots, \boldsymbol{x_n}(t))$, where $\boldsymbol{x_i}(t) = \left[x_{i1}(t), \ldots, x_{ij}(t)\right]$ is the i-th pattern made from observations of each J features (i = 1,. . .,n and j = 1,. . .,J). From this collection, n_k patterns belongs to class k (k = 1,. . .,K). We assume that these n patterns are a realization from the random sample $\boldsymbol{X}_1^{(t)}, \ldots, \boldsymbol{X}_n^{(t)}$ of a population that follows a Multivariate Normal distribution. Although it is possible to execute the previous statistical tests without such assumption, its relevance relies on the correct definition of the probability distribution associated with those test statistics (T^2 or M, for example). When such assumption is not verified in practice, then the probability of wrong conclusion increases (rejecting a true hypothesis, or not rejecting a false). The application of DetectA to real datasets (see subsection 4.2) will provide more information about the impact over the performance when the distribution of the random vector is not idealized.

The following subsections present the two types of abrupt drift detection with DetectA: reactive and proactive. Although the main contribution of this work is the proactive drift detection, we begin with the reactive approach, which will ease the proactive procedure description.

3.1. Reactive detection

The reactive detection implies the existence of patterns classes at instant $t+1$. Consider n labelled patterns $\boldsymbol{x}_1(t+1), \ldots, \boldsymbol{x_n}(t+1)$ and α significance level. For the reactive abrupt drift detection, the following steps must be performed:

- In the conditional mean vector:

(1) Compute $\bar{\boldsymbol{X}}_{C_k}(t)$ and $\bar{\boldsymbol{X}}_{C_k}(t+1)$, as well as $\boldsymbol{S}_{C_k}(t)$ and $\boldsymbol{S}_{C_k}(t+1)$.
(2) Calculate the test statistic T^2 (Eq. (5)).
(3) Define the occurrence of a drift if $\frac{(n_1+n_2)-J-1}{(n_1-n_2-2)J}T^2 > F_{J,(n_1+n_2)-J-1}(\alpha)$

- In the conditional covariance matrix:

(1) Find $\boldsymbol{S}_{C_k}(t)$, $\boldsymbol{S}_{C_k}(t+1)$ and $\boldsymbol{S_{pool}}$.
(2) Compute the test statistic M (Eq. (8)), u (Eq. (9)), and finally $C = (1-u)M$.
(3) Drift in the covariance matrix if $C > \chi^2_{\frac{1}{2}J(J+1)}(\alpha)$

After the drift detection, some measure might be taken, for example, retraining the current classifier or changing some component in the process under analysis. However, such adaptation may take longer than necessary, and the misclassification caused by the late drift detection can reduce the reliability of the whole built framework. Next subsection exhibits the proactive approach, aiming to predict an eventual abrupt drift and adjust the classifier or process before the misclassification occurs.

3.2. Proactive detection

A proactive drift detection means that the detection must occur in the absence of the label from patterns at time $t+1$. That is, the detection should be performed before the classifier commits a mistake. Then, let $\boldsymbol{x}_1(t+1), \ldots, \boldsymbol{x_n}(t+1)$ the set of n unlabeled patterns from instant $t+1$. Commonly, these patterns belong to the test set, thereby waiting for some future moment to obtain its labels. As it is not possible to compute the sample conditional mean vector $\bar{\boldsymbol{X}}_{C_k}(t+1)$ or the conditional covariance matrix $\boldsymbol{S}_{C_k}(t+1)$, it is unfeasible to compare these quantities with those computed at time t $\left(\bar{\boldsymbol{X}}_{C_k}(t)\right)$ and $\left(\boldsymbol{S}_{C_k}(t)\right)$.

Therefore, the proactive detection method must depend on the information contained in $\boldsymbol{x}_1(t+1), \ldots, \boldsymbol{x_n}(t+1)$ and propose a set of labels to these patterns before the actual classification is performed. Approaches that are independent from the class labels but based on the pattern distribution are the clustering algorithms [10].

The proactive approach is, therefore, based on the agglomerative clustering methods for the following reasons: (*i*) the number of groups to be formed known a priori (identical to the number of classes in the problem); (*ii*) the initial condition for the centroid of each group is the conditional mean vector of each class, which helps the algorithm to formulate the group and the subsequent identification of groups as classes; and (*iii*) tends to be computationally more efficient than divisive clustering methods.

There are several agglomerative clustering methods in the literature (such as C-means, Gaussian Mixture Model) [6,4], but as a first approach we used the simple k-means method [6]. This classical method is easy to implement and computationally efficient. Based on this approach, the next steps show (in pseudocode) how to implement the proactive version of DetectA:

1. The initial dataset (t = 1) has n(t) patterns that belong to K classes. This dataset is commonly used to train the classifier.

2. Using the data from this first dataset, the sample conditional mean vectors $\bar{X}_{C_1}(t), \ldots, \bar{X}_{C_K}(t)$, for each available class, are computed.

3. At time (t = t + 1 a new set of n(t) unclassified patterns is received. Then do:

a. Group the n(t) patterns using the k-means, setting the number of groups as the number of classes (groups = K) and the centroids of each group as the vectors $\bar{\boldsymbol{X}}_{C_1}(t-1), \ldots, \bar{\boldsymbol{X}}_{C_K}(t-1)$. We suggest the use of Mahalanobis distance as the dissimilarity metric, aiming to form groups with spherical as well as elliptical shapes [24].

b. After the convergence of the k-means algorithm, define $n_1(t)$ patterns closest to the centre initiated in $\bar{\boldsymbol{X}}_{C_1}(t-1)$ as belonging to class 1, the $n_2(t)$ patterns closest to $\bar{\boldsymbol{X}}_{C_2}(t-1)$ as belonging to class 2, and so on.

c. Then compute the new conditional mean vector and covariance matrix of each class, which are not represented by $\bar{\boldsymbol{X}}_{C_k}(t)$ and $\boldsymbol{S}_{C_k}(t)$ any more, but by $\bar{\boldsymbol{X}}_{\hat{C}_k}(t)$ and $\boldsymbol{S}_{\hat{C}_k}(t)$, estimated based on the predicted class by the k-means algorithm.

4. Consider α as the predefined significance level. Given this level, a proactive abrupt drift detection can be implemented at:

a. Conditional mean vector:

i. Compute $\bar{\boldsymbol{X}}_{C_k}(t-1)$, $\bar{\boldsymbol{X}}_{\hat{C}_k}(t)$, $\boldsymbol{S}_{C_k}(t-1)$ and $\boldsymbol{S}_{\hat{C}_k}(t)$.
ii. Calculate the test statistic T^2 (Eq. (5)).
iii. Drift if $\frac{(n_1+n_2)-J-1}{(n_1-n_2-2)J} T^2 > F_{J,(n_1+n_2)-J-1}(\alpha)$,

b. Conditional covariance matrix:

i. Calculate the $\boldsymbol{S}_{C_k}(t-1)$, $\boldsymbol{S}_{\hat{C}_k}(t)$ and $\boldsymbol{S_{pool}}$.
ii. Compute M (Eq. (8)) and u (Eq. (9)).
iii. Drift if $C > \chi^2_{\frac{1}{2}J(J+1)}(\alpha)$.

5. Return to step 3 whenever is necessary

Note that in this method there is no need to store the previous data blocks, but only the number of patterns of each class, the conditional mean vector and covariance matrix of the previous instant. To illustrate the enunciated steps, Figs. 3 and 4 show a toy example with two classes (Masculine and Feminine) and two features (Height and Weight) problem.

The first step (Fig. 3a) is to cluster all labelled data, forming two groups using as initial centroids the conditional means of each class. After the clustering has converged, it is necessary to store the cluster centroids as well as the current observed conditional means and covariance matrices. The clustering method needs to run again when the new unlabelled data arrives (Fig. 3b), but instead of starting the centroids from scratch, it should use the centroids from the previous run.

After the clustering method has halted (Fig. 4a), we treat all the data closest Mahalanobis distance from each centroid as belonging to a particular class. Obviously, this is an estimation not used to classify the patterns, but to approximate four unknown quantities: $\bar{\boldsymbol{X}}_{\hat{Mas}}(t+1)$, $\boldsymbol{S}_{\hat{Mas}}(t+1)$, $\bar{\boldsymbol{X}}_{\hat{Fem}}(t+1)$ and $\boldsymbol{S}_{\hat{Fem}}(t+1)$. After we have computed these statistics, we apply Hotelling's T^2 and Box-M test to check if some abrupt drift has happened (Fig. 4b).

If an abrupt drift is detected, one of two approaches might be performed to avoid misclassification of the patterns that suffered the drift: the classifier must be retrained, using the labels provided by the clustering algorithms; or the test patterns are used to adjust the current classifier. This adjustment can be realised based on conditioning these new patterns to have a similar distribution to those used during the classifier training. To illustrate this idea, consider the following example:

1. Suppose a problem with two classes and two attributes, plus a set of n test patterns. The proactive detection method was used and has identified the patterns that possibly belong to class 1 and 2.
2. Then, based on the conditional mean vector and covariance matrix of the labelled data in the previous instant, the detection method verifies the presence of an abrupt drift in the conditional mean vector of class 1. It has detected a deviation of 10 units related to the average of the first feature when compared to the previous instant.
3. To "correct the drift", that is, to make the drift unperceived by the classifier, just subtract 10 units in the first feature from the test patterns that are credited to belong to class 1 and use the old classifier with the corrected data.

Therefore, the process of "correcting the drift" is perhaps one of the great novelties of DetectA. This approach is called *Pattern Mean Shift*. However, if the detection is wrong, the correction may be harmful and, in some cases, lead to a possible drift. Therefore, a proactive approach should be conservative and fine-tuned to ensure that such errors are in a situation of low impact to the classifier. Next section displays the experiments performed with DetectA.

4. Results and discussions

This section details the experiments performed with the DetectA method. Two main experiments have been performed: a sensitivity analysis of DetectA on artificially generated datasets with different

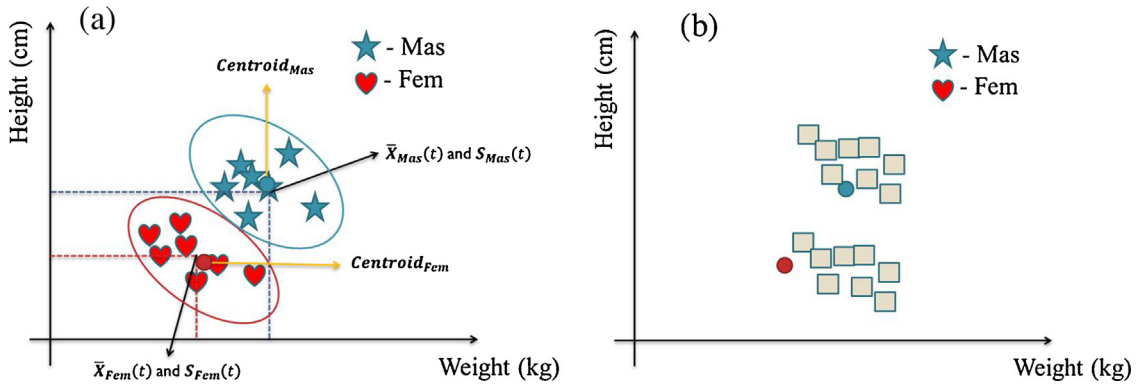


Fig. 3. (a) Clustering of the training set using the known class labels to identify the centroids of each class; (b) The centroids found during the clustering process in (a) are shown (circles) overlaid on the test set examples (squares).

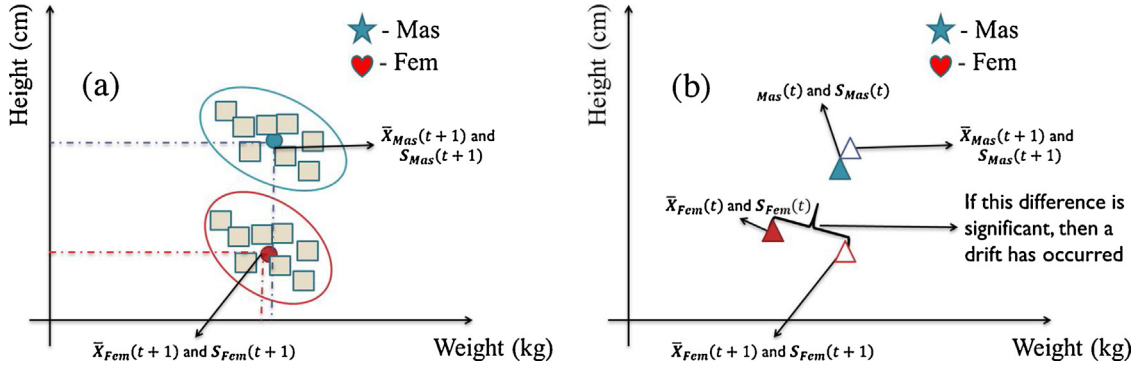


Fig. 4. Clustering the test set and comparing estimated averages and covariance matrices between two iterations.

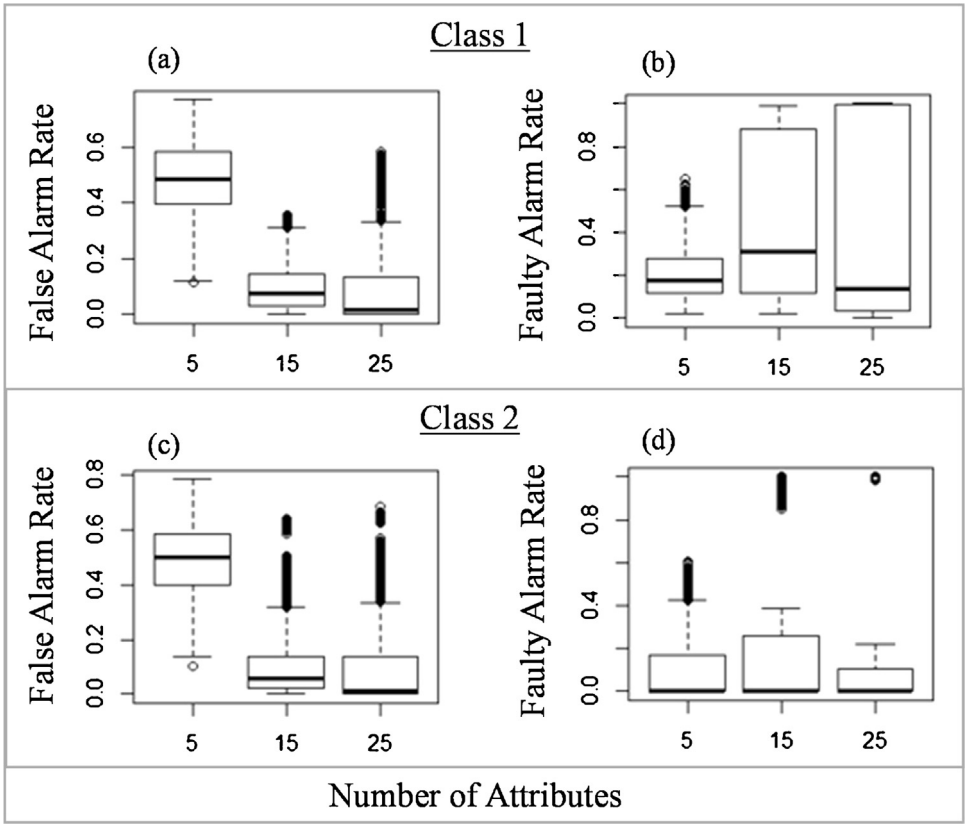


Fig. 5. Boxplots of false and faulty alarm rate break down by number of attributes.

Table 2
Performance metrics for the drift detection process.

Detector	Drift (D)	Absence of Drift (ND)
Alert (A)	#A&D	#A&ND
Idle (NA)	#NA&D	#NA&D

parameters and settings for the data generation process; performance comparison of the various configurations of DetectA with well-established benchmarks in the drift detection literature.

4.1. Experiment 1: sensitivity analysis of DetectA

In this experiment, a sensitivity analysis is performed based on the variation of certain parameters related to the data generating process (number of attributes, the number of patterns, imbalance rate between classes, etc.), to understand the influence of each parameter on the overall performance of DetectA. In this sense, we generated artificial datasets manipulating the instant and the type of drift. The procedures employed to generate these datasets are detailed in [43].

We measured the detection effectiveness by computing its false positives (false alarms) and false negatives (faulty alarms) rates. These metrics can be summarized by a matrix displayed in Table 2. Assuming the two possible situations in the data – drift (D) and absence of drift (ND) – the detector can alert a drift (A) or do not alert (NA), resulting in four possible results:

- #A&D: there is drift (D) and the detector has produced an alert (A)
- #NA D: there is drift (D) but the detector did not produce an alert (NA)
- #A&ND: there is no drift (ND) but the detector produced an alert (A)
- #NA&D: no drift (ND), and the detector did not produce an alert (NA)

The false alarm rate (i.e., false positive rate) is calculated by:

$$FPR = \frac{\#A\&ND}{\#A\&ND + \#NA\&ND} \tag{10}$$

measuring the ratio between the number of cases in which the alerts were performed out of time and the total number of cases without drift. On the other hand, the faulty alarm rate (i.e., false negative rate) is given by:

$$FNR = \frac{\#NA\&D}{\#NA\&D + \#A\&D} \tag{11}$$

relating the number of mistakes made by being idle when a drift happened (#NA&D) and the number of drift available in the data stream. In general terms, a good detector minimises both false positives and false negatives. For simplicity, the experiment is conducted considering binary datasets with the parameters as described in Table 3. For each configuration we generated 100 datasets; therefore, the results presented in the following sections are an average of 100 runs of the detection method.

It is also possible to evaluate the delay time between the occurrence of a drift and its detection, but since the number of blocks used in this experiment is small (only ten blocks), we believe that such metric will not provide substantial information into our analysis. The following subsections present the results of this experiment.

4.1.1. Individual parameters analysis

This experiment involved 4860 different configurations since the combinations of all possible values of all parameters were evaluated. Table 4 presents the influence of each parameter, based on the average of false and faulty alarm rates grouped by each parameter value (average of all possible configurations with each parameter value). In all cases, the observed standard deviation was less than 2%.

Based on Table 4 we verify that an increase in the number of attributes seems to result in a reduction in the false alarm rate, something that is not verified with the faulty alarm rate. We also found out that, as the number of patterns (block size) expands, the false alarm rate modestly increases, whereas the faulty alarm rate tends to decrease.

The false alarm rate for class 1 appears to be decreasing as the rate of imbalance increases, although the variation is not very expressive, and no significant variation was observed for the other indicators. We can also observe that although, in a non-expressive way, the false alarm rate values tend to decrease as the number of blocks with drift in class 1 increases, while the values of the faulty alarm rates remain practically constant. Considering the number of blocks with drift in class 2, the false alarm rate for class 2 is the only one that presents significant variation, decreasing as the number of blocks with drift in class 2 decreases.

Fig. 5 presents four boxplots, each one evaluating how the mismatching rate varies according to the number of attributes. We can observe that the configurations with 5 attributes provides the highest false alarm rates for both classes, concentrating around 40–60% region, while those with 15 and 25 features have their false alarm rates below 20%. However, we can notice a reasonable number of outliers for both classes, mainly when the number of features is equal to 25. Regarding faulty alarm rates, we can perceive that for class 1 the best rates are obtained when the number of features is small, oscillating when this number grows. For class 2, we were unable to spot a clear pattern, but we can point out that overall the rates tend to be at reasonable low levels. In general, when we analyze the median values, it is possible to assert that false alarm rates are negatively correlated with the number of attributes for both classes. Also, the bulk of faulty alarm figures concentrated close to zero.

Fig. 6 outlines four additional boxplots, in this case each depicting how the number of patterns has affected the different false and faulty alarm rates for class 1 and 2. We can verify that the interquartile range tends to increase (bigger dispersion) when the respective block size faced by DetectA grows, contrasting with the falls in the median false alarm rates in both classes – although such reduction seems to be inexpressive. In relation to faulty alarms, the boxplot interquartile range decreases with the increase in the number of patterns; in any scenario the median values settled down at very

Table 3
Parameters analysed during the sensitivity analysis.

Description	Values used
Number of attributes	5, 15, 25
Number of patterns per block (block size)	150, 350, 500
Proportion of the occurrence of class 1 in relation to class 2	0.2, 0.35. 0.5
Number of blocks where there is occurrence of drift in class 1	1, 3, 5, 7
Number of blocks where there is occurrence of drift in class 2	0, 1, 3, 5, 7
Proportion of attributes that will suffer drift within the block	0.2, 0.35, 0.5
Alpha: Level of significance, that is, minimum level that is accepted from the alternative hypothesis H1 (occurrence of drift) is correct	0.01, 0.05, 0.1

Table 4
False and faulty alarm rates grouped for each parameter value.

	Number of attributes			Number of patterns			Imbalance rate			Alpha		
	5	15	25	150	250	500	0.2	0.35	0.5	0.01	0.05	0.1
False Alarm – Class 1	0.48	0.09	0.08	0.19	0.22	0.24	0.25	0.21	0.19	0.15	0.2	0.31
Faulty Alarm – Class 1	0.21	0.44	0.39	0.38	0.34	0.32	0.34	0.35	0.35	0.75	0.21	0.08
False Alarm – Class 2	0.49	0.10	0.09	0.20	0.24	0.26	0.23	0.22	0.23	0.14	0.23	0.32
Faulty Alarm – Class 2	0.11	0.21	0.19	0.18	0.16	0.16	0.16	0.17	0.17	0.38	0.09	0.03
	Number of blocks with drift in class 1				Number of blocks with drift in class 2					Proportion of attributes with drift		
	1	3	5	7	0	1	3	5	7	0.2	0.35	0.5
False Alarm – Class 1	0.25	0.24	0.22	0.17	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22
Faulty Alarm – Class 1	0.34	0.34	0.35	0.35	0.35	0.35	0.34	0.34	0.34	0.35	0.35	0.34
False Alarm – Class 2	0.26	0.24	0.22	0.2	0.27	0.21	0.20	0.18	0.15	0.23	0.23	0.23
Faulty Alarm – Class 2	0.17	0.17	0.17	0.17	0.00	0.33	0.34	0.34	0.34	0.17	0.17	0.16

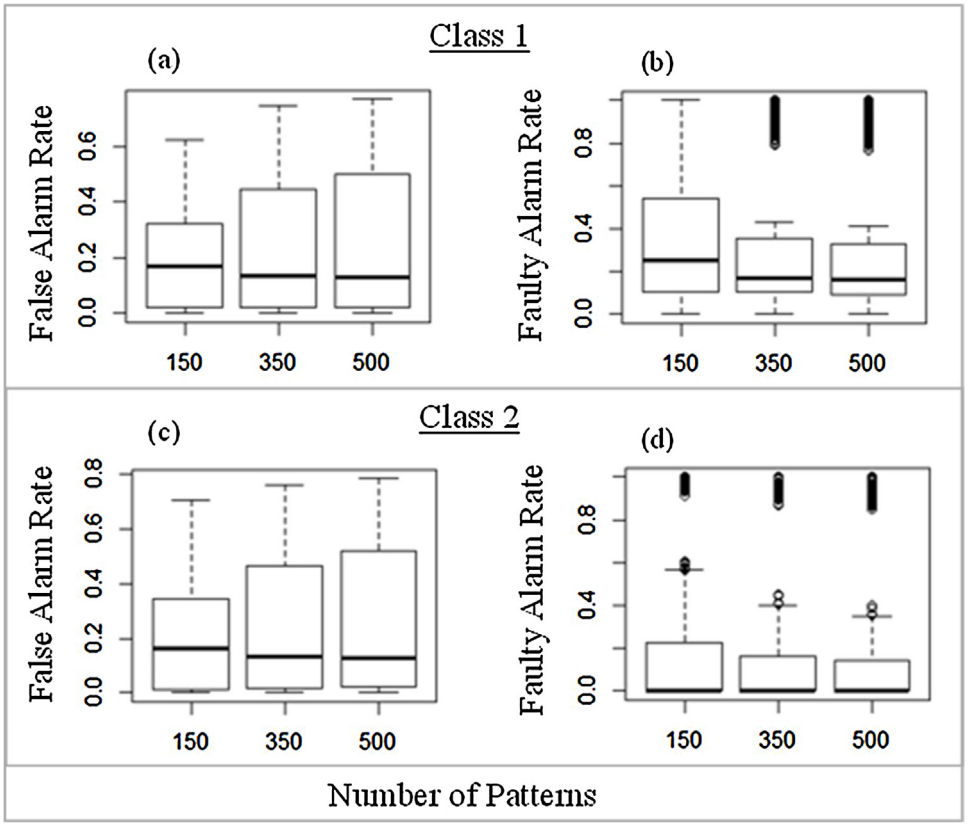


Fig. 6. Boxplots of false and faulty alarm rate break down by number of patterns.

low levels – suggesting the usual negative relationship between number of patterns and faulty alarm rates (false negative rates).

Finally, it is noticed that the false alarms rates increase as alpha rises, which was already expected, because higher alpha values make the model more reactive, with a higher probability of false alarms. The faulty alarm rate, on the other hand, decreases as alpha rises, also showing a higher model reactivity. Overall the results seem reasonably good, especially for the false alarm rates: the mean is 22% for class 1 and 23% for class 2, and the median is 14% for both classes, indicating that the proposed detection model has low occurrences of false alarms. The faulty alarms are also considerably low, with a median of 19%. Table 5 presents the mean, median, and standard deviation for the four indicators.

This subsection presented the main results from an individual parameters perspective, as well as a general view of the indicators. The next subsection presents a quantification analysis considering eventual interactions between parameters (number of attributes and patterns) via the Analysis of Variance method.

Table 5
Statistics aggregated by indicators.

Indicator	Mean	Median	Std Deviation
False Alarm for Class 1	0.22	0.14	0.22
Faulty Alarm for Class 1	0.35	0.19	0.34
False Alarm for Class 2	0.23	0.14	0.22
Faulty Alarm for Class 2	0.17	0.00	0.30

4.1.2. Quantifying each parameter relevance

Analysis of Variance (ANOVA) [26] is a statistical procedure widely applied to detect substantial differences in populations' means across different factor levels. In our context of sensitivity analysis, it allows the identification of the factor levels (number

Table 6
Analysis of variance considering false (FEA) and faulty (FYA) alarm rate. The symbol * denotes p-values less or equal 2.10^{-16}.

		FEA – Class 1		FEA – Class 2		FYA – Class 1		FYA – Class 2	
		S	p	S	p	S	p	S	p
Main	Number of atributes	**209.37**	*	**202.65**	*	**37.70**	*	**7.50**	*
	Number of patterns per block (block size)	2.91	*	4.93	*	4.00	*	0.90	*
	Imbalance rate between classes	5.53	*	0.00	0.78	0.20	*0.04*	0.30	*0.03*
	Number of blocks with drift in class 1	0.01	0.35	0.00	0.86	0.10	0.05	0.10	0.25
	Number of blocks with drift in class 2	**5.94**	*	4.68	*	0.10	0.22	0.00	0.79
	Proportion of attributes with drift within the block	0.00	0,76	**14.69**	*	0.00	0.31	**135.10**	*
	Alpha	**36.84**	*	**40.22**	*	**557.00**	*	**156.00**	*
Interactions	1) Number of patterns × number of attributes	**17.55**	*	**11.44**	*	**7.40**	*	**1.10**	*
	2) Number of patterns × imbalance rate between classes	0.60	*	0.11	*0.01*	0.10	0.25	0.00	0.64
	3) Number of attributes × proportion of attributes with d*rift*	0.01	0.44	0.00	0.62	0.00	0.95	0.00	0.94

of attributes = {5, 15, 25}, for example) that most influence the DetectA's performance. As a byproduct, this analysis makes it possible to determine which variables (number of attributes, patterns, etc.) tend to affect the detection performance with higher intensity.

Table 6 shows the main ANOVA results as a function of the false and faulty alarm rates for each main factor, as well as certain interactions terms that we believe, are the most useful in practice and that seems to have relevant relation: number of patterns with number of attributes, number of patterns with imbalance rate between classes, and finally, number of attributes with proportion of attributes with drift. The most important terms (higher values of sum of squares – S) are highlighted in bold, with the statistically significant terms (p-value <0.05) highlighted in italics.

Considering the false alarm rates sum of squares, we can observe that:

- the number of attributes is the most influential parameter of the model.
- the alpha parameter is directly related to the detector reactiveness, thereby a direct impact on the false alarm rates was expected.
- the variation in the number of blocks with drift tends to easier the detection.

From the p-value perspective, the terms that shows higher influence on the false alarm rates are: number of attributes, number of patterns, imbalance degree between classes (influences only for class 1, because it is the minority class), number of blocks with drift, alpha, 1st interaction (number of patterns with number of attributes) and 2nd interaction (number of patterns with imbalance between classes).

Based on the most relevant terms, we applied the so-called Tukey test [26], a test commonly used to find differences between levels of certain terms. The results are detailed in Appendix A. In this case, we applied the Tukey test to number of attributes, number of patterns, alpha, number of blocks with drift and 1st interaction. Given that, we could derive the following trends:

- the lower the number of attributes, the higher the false alarm rates are;
- as we increase the number of patterns, we observe a larger amount of false alarm rates;
- the higher the alpha, the higher the false alarm rates tend to be;
- as we enlarge the number of blocks with drift, the false alarm rate tends to decrease;
- in the 1st interaction, the variations on the number of attributes are the defining term for the whole interaction.

About the rate of faulty alarms, the alpha parameter is by far the most influential in the model (based on the value of the sum of the squares), followed by number of blocks with drift in class 2, and number of attributes. The other model terms that are statistically significant for the faulty alarm rates are number of patterns, imbalance rate between classes and 1st interaction (number of patterns with attributes). We followed the analysis by applying the Tukey test for the most significant and relevant terms: number of attributes, alpha, number of patterns and 1st interaction. In summary, Tukey test captured the following trends, for both classes:

- the faulty alarm rate does not seem to follow a clear linear relationship with the increase/decrease of the number of attributes;
- when alpha is hiked up, the lower the faulty alarm rates are;
- more patterns tend to imply lower faulty alarm rates;
- about the 1st interaction, the number of attributes is the strongest parameter of the interaction regarding the influence on faulty alarms.

In contrast to the false alarm rates, the Tukey method tended to follow our expectations regarding relationship across factors and faulty alarm rates.

Based on the results obtained, this first experiment suggests that the detector is more efficient for:

- high-dimensional datasets, since false alarm rates decays in the presence of more attributes, as well as faulty alarms do not show a substantial increase;
- intermediate size blocks, as a trade-off between increase/decrease of false and faulty alarm rates;
- datasets with any proportion of drift, since this parameter does not demonstrate the significant influence on the false or faulty alarm rates;
- imbalanced binary classes datasets, since our results have not undergone significant fluctuations with the change in this parameter.

This section presented the results of the first experiment performed with DetectA. The next section presents the comparison of DetectA results with other drift detection methods in the literature.

4.2. Experiment 2: drift detection in datasets

To verify the improvement that DetectA can offer during classification tasks (accuracy and the computational performance), five different datasets were used and several simulations were carried out in various scenarios. These datasets are quite known in the literature and have already been used with different drift detection methods. Next sub-sections briefly describe the five datasets, the simulations carried and the discussion about the results obtained.

Table 7
Block size and number of blocks used in the experiments.

Dataset	Block Size	Number of Blocks
SEA Concepts	250	400
Nebraska	30	583
Electricity	48	944
Cover Type	500	1162
Poker Hand	500	1658

4.2.1. Datasets description

The datasets used in this experiment are the SEA Concepts, an artificial dataset where a more controlled environment about the drifts is provided, and four real datasets (Nebraska, Electricity, Cover Type and Poker Hand), where the exact moment that the drift occurs is not known.

The **SEA Concepts** dataset was artificially created by [32]. It is characterised by extensive periods without major changes in the environment, but with occasional abrupt drifts. The **Nebraska** dataset presents a compilation of climate measurements from the Offutt Air Force Base substation in Bellevue, Nebraska. Its objective is to predict whether a rainfall may appear, using data from the last 30 days. Both datasets are available in [28].

The **Electricity** dataset is extracted from the Australian New South Wales Electricity Market, and the class label defines the price change related to a moving average of the last 24 h. The purpose of the problem is to predict whether the price will go up or down. The **Cover Type** dataset contains information cells corresponding to a forest cover of 30×30 m, extracted from the US Forest Service (USFS). Its goal is to predict the type of forest cover among seven possible values (therefore, a multi-class problem). The **Poker Hand** dataset has as output ten possible categories representing the poker hand, which contains 5 cards. The purpose is to identify the type of a Poker hand among the ten possibilities. These datasets are available in [25].

4.2.2. Experiment description

In order to investigate the influence of the detector on the accuracy and computational performance of a classifier, a simple multi-layer perceptron (MLP) neural network was chosen as the base classifier. We applied it in four different configurations of DetectA: two proactive approaches (Group Label and Pattern Mean Shift, detailed below) and one reactive strategy, already described in Section 3. The proactive approaches are:

- **Group Label**: At each new data block received, a clustering is performed, using as a suggestion the centroids of the previous labelled data block, to determine the predicted classes for each pattern of the new block. Using this clustering step as an input, the detection mechanism checks if a drift occurred about the previous block and, if so, a new MLP is created and trained with the new block, and the class labels suggested in the clustering.
- **Pattern Mean Shift**: Similar to the Group Label approach, with the difference that when a drift is detected, instead of creating a new MLP using the new data block, the old data block is used to train the MLP and the drift is "removed" from the new data block. While in the Group Label approach the new MLP is adjusted to the new data, in Pattern Mean Shift approach the new information is adjusted to the old MLP. This method has already been detailed in Section 3.

For the sake of comparison, experiments were also performed with an MLP without any detector. We decided to use the block approach for the training and testing of the models. The block size and number of blocks used for each dataset are presented in Table 7. We chose the same values already used in literature, such as in [9] and [8].

Two different training approaches were performed for the classifiers with a detection method:

- **Forget the past after detection**: retrains the classifier for every new data blocks using all information available; however, in the case of drift detection, the training is only performed from the detection point onwards.
- **Only retrain after detection**: only retrains the classifier when there is a drift, using the block where the drift was detected.

For the classifier without detection, the traditional training approach was used: for each new block, the classifier is retrained with all the past labelled blocks.

The simulations are then performed using seven variations, as summarised in Table 8. In each variation, 30 simulations were executed by building an MLP with 5 neurons in the hidden layer, and 30 simulations with an MLP with 10 neurons in the hidden layer, totalizing 420 runs for each dataset.

For the **reactive** approach, the detection mechanism is executed when the real labels of the new block (t) arrive, comparing it in relation to block t-1. On the other hand, in all approaches with **proactive** detection, the new block (t) is used to make the clustering and check the occurrence of drift in relation to the previous block. In Group Label approaches, in the case of a drift occurrence, the network is retrained only with the block t and the class labels provided in the clustering. In the Pattern Mean Shift approaches, in the case of drift, the network is tested with the block "t adjusted" towards the detected drift. When the real labels of this block arrive, the classifier is retrained only with the labelled block t. It is worth to mention that in **proactive** detection approaches, whenever the labels of a new block are available, the cluster centroids are adjusted to the next grouping considering the new labelled block.

4.2.3. Results of the experiment

Tables 9 and 10 display the results on average accuracy and execution time in seconds for each of the seven approaches used, respectively. We highlighted the best results by dataset in bold and the worst in italics and underlined. We took care to consider that the best setting provides highest values of accuracy and saving in execution time.

We observe that the pattern is similar considering most of the datasets: the best models are in this order: (i) with reactive detection approach ("forget the past after detection"); (ii) with reactive detection (approach "only retrain after detection"); (iii) proactive detection Pattern Mean Shift (approach "forget the past after detection"); and finally, (iv) with proactive Pattern Mean Shift detection (approach "only retrain after detection").

The main takeaways that can be harnessed from this experiment are:

- In general, the best results are obtained using some detection. As the datasets used have some drift (not necessarily only abrupt), the detection procedure helped to improve the base classifier performance;
- In the realm of proactive approaches, the Pattern Mean Shift approach has shown better performance than Group Label. This outperformance can be explained by the aggressiveness of Group Label approach about training the classifiers, always discarding the old classifier, which seemed not to be the best option for these datasets;
- Comparing "Forget after past detection" approach with "Only retrain after detection", it is observed that the accuracy of the former is favourably higher, while the computational performance of the latter is superior in most cases. Obviously, the first approach

Table 8
Models and approaches used in the experiment.

#	Model	Training approach	Acronym
1	No detection	Traditional	ND
2	Reactive Detection	Forget the past after detection	RD-FPAF
3	Reactive Detection	Only retrain after detection	RD- ORAD
4	Proactive Detection – *Group Label*	Forget the past after detection	PD-GL-FPAF
5	Proactive Detection – *Group Label*	Only retrain after detection	PD-GL- ORAD
6	Proactive Detection – *Pattern Mean Shift*	Forget the past after detection	PD-PMS-FPAF
7	Proactive Detection – *Pattern Mean Shift*	Only retrain after detection	PD-PMS-ORAD

Table 9
Average Accuracy Results. In all cases, the observed standard deviation was less than 2%.

Accuracy	SEA		Neb.		Elec.		Pok.		Cov.	
Hidden Layer Size	5	10	5	10	5	10	5	10	5	10
ND	*0.64*	*0.63*	0.66	0.66	*0.42*	*0.42*	0.61	0.67	0.76	0.82
RD-FPAD	**0.87**	**0.86**	**0.67**	**0.68**	**0.76**	**0.77**	0.67	**0.68**	0.80	**0.84**
RD- ORAD	0.81	0.84	**0.68**	**0.68**	**0.76**	**0.76**	0.66	**0.69**	0.80	**0.83**
PD-GL-FPAD	0.71	0.71	*0.54*	*0.54*	0.68	0.68	*0.13*	*0.14*	*0.58*	*0.60*
PD-GL- ORAD	0.71	0.71	*0.54*	*0.54*	0.70	0.70	*0.13*	*0.13*	*0.58*	*0.60*
PD-PMS-FPAD	0.84	0.83	0.65	0.65	0.73	0.74	0.66	**0.69**	0.80	**0.83**
PD-PMS- ORAD	0.83	0.83	0.64	0.65	0.73	0.73	0.66	**0.69**	0.81	**0.83**

Table 10
Execution Time Results (in seconds). In all cases, the observed standard deviation was less than 2%.

Time in seconds	SEA		Neb.		Elec.		Pok.		Cov.	
Hidden Layer Size	5	10	5	10	5	10	5	10	5	10
ND	42	41	**115**	**116**	371	**164**	**1423**	**1677**	**832**	**869**
RD-FPAD	78	65	210	166	574	537	2007	2419	1242	1150
RD- ORAD	**10**	**7**	173	159	531	**312**	2061	2256	1184	1134
PD-GL-FPAD	*205*	192	*503*	388	651	534	2848	2631	*2154*	1743
PD-GL- ORAD	194	*209*	294	*513*	*875*	*825*	*2923*	*2979*	*2217*	1467
PD-PMS-FPAD	116	104	233	216	464	462	2193	2746	1501	1638
PD-PMS- ORAD	177	188	405	366	691	744	2177	2676	1440	1576

produces better accuracy due to more exhaustive training, with an increase in computational time.

Considering computational time, the fastest approach, as expected, is the one that does not use any detection. Coming in second, we have the reactive approaches, also as expected, because they do not perform any data clustering. Comparing proactive approaches, the Pattern Mean Shift is the one that presented the best computational performance considering all the databases. Forget the Past After Detection approach was even better than the reactive approaches to the Electricity database, indicating that this method could be the more suitable for certain databases when a user are interested in computational time.

It is important to note that although the reactive approaches presented numerically higher accuracy values than the Pattern Mean Shift approach, the difference is not statistically significant (p-value >0.05) for all databases, considering the standard deviation found. This result indicates that the Pattern Mean Shift approach can anticipate drifts, which is the main contribution of the proposed method, although this does not lead to a significantly higher accuracy than the reactive approach.

Finally, it is worth remembering that most existing reactive algorithms assume that real labels are immediately and entirely available, and such assumption is often violated in real-world applications [34]. In many of these situations, accuracy may not be the most important metric, and, because of the unavailability of the real labels or by the urgency of taking action, it is more interesting to detect a drift as soon as possible. For example, in cases of detection of diseases or epidemics, actions must be taken after the occurrence of the change, and it is essential to detect a drift as soon as possible, ideally immediately after it occurs [31]. In these problems, proactive approaches are possibly the best choice since they can detect some early drifts and avoid serious problems.

This section presented the results of the experiments performed with the proposed detection method, DetectA. The next section concludes this work.

5. Conclusions

This work presented a concept drift detection mechanism designed for abrupt concept drift detection, called DetectA. The main novelty of this approach is its proactive feature: it is intended to detect a forthcoming concept drift, as opposed to most of the drift detection procedures that only detects concept drifts after their occurrence. Also, it has been proposed a procedure for creating datasets with pre-defined abrupt drifts. This procedure was used in the sensibility analysis of DetectA, based on variations of the number of attributes, patterns, imbalance rate between classes, among others, to understand the influence degree of each parameter on its final performance. Our results on the sensibility analysis suggested that the detector is efficient and suitable for datasets of high-dimensionality, blocks with medium size, any proportion of drifts and class imbalance. It is important to mention that these results are not exhaustive and further tests using other drift types must be conducted, and we also intend to evaluate the effectiveness of other clustering methods at DetectA mechanism.

We also tested the DetectA algorithm combined with a classification method (MLP) to verify the joint performance of these methods, and we conducted these experiments with artificial and real datasets. The best results were obtained using some detection, being the proactive manner a top contender regarding improving

the underlying base classifier accuracy. In future, we intend to test DetectA combined with a more complex classification method, like ensembles of neural networks or neuro-evolutive approaches.

Finally, a way to prevent the ad-hoc selection of a clustering method is to verify its efficiency in some previous experiments. For this purpose, one might use a set of clustering methods and evaluate their quality, which could be measured using metrics that do not need the pattern labels, such as Silhouette [10], or those that consider like cR [20]. The method with the best performance would be chosen for the proactive detection process.

Another interesting future work would be a hybrid approach combining proactive and reactive methods. Upon receiving a new data block, the proactive method is executed and, if no drift is detected, the classification is performed according to the current model. When the real labels arrive for this data block, the reactive method is executed and if drift is detected, the model is retrained. This hybrid approach is likely to be more accurate as a double-check will be done.

Appendix A. Tukey Test Results

The following tables show the Tukey test for the parameters: number of attributes, number of patterns, alpha, number of blocks with drift in class 1 and 2 and for interaction 1, considering the false alarm rates.

See Tables A1–A9

Table A1
Tukey Test for number os attibutes and false alarms.

False Alarm – Class 1			False Alarm – Class 2		
# of atributes	Difference	p-value	# of atributes	Difference	p-value
15–5	−0.387	<0.05	**15–5**	−0.385	<0.05
25–5	−0.402	<0.05	**25–5**	−0.395	<0.05
25–15	−0.015	<0.05	**25–15**	−0.011	<0.05

Table A2
Tukey Test for number of patterns and false alarms.

False Alarm – Class 1			False Alarm – Class 2		
# of patterns	Difference	p-value	# of patterns	Difference	p-value
350–150	0.030	<0.05	350–150	0.041	<0.05
500–150	0.047	<0.05	500–150	0.061	<0.05
500–350	0.017	<0.05	500–350	0.020	<0.05

Table A3
Tukey Test for *alpha* and false alarms.

False Alarm – Class 1			False Alarm – Class 2		
Alpha	Difference	p-value	*Alpha*	Difference	p-value
0.05–0.01	0.052	<0.05	0.05–0.01	0.083	<0.05
0.1–0.01	0.167	<0.05	0.1–0.01	0.176	<0.05
0.1–0.05	0.115	<0.05	0.1–0.05	0.094	<0.05

Table A4
Tukey Test for number of blocks with drift in class 1–2 and false alarms.

False Alarm – Class 1			False Alarm – Class 2		
# of blocks with drift in class 1	Difference	p-value	# of blocks with drift in class 1	Difference	p-value
3–1	−0.010	<0.05	1–0	−0.067	<0.05
5–1	−0.029	<0.05	3–0	−0.072	<0.05
7–1	−0.076	<0.05	5–0	−0.089	<0.05
5–3	−0.019	<0.05	7–0	−0.123	<0.05
7–3	−0.066	<0.05	3–1	−0.006	<0.05
7–5	−0.047	<0.05	5–1	−0.022	<0.05
			7–1	−0.056	<0.05
			5–3	−0.016	<0.05
			7–3	−0.051	<0.05
			7–5	−0.035	<0.05

Table A5
Tukey Test for Interaction 1 (number of attributes × number of patterns) and false alarms.

False Alarm – Class 1			False Alarm – Class 2		
Number of attributes x number of patterns	Difference	p-value	Number of attributes x number of patterns	Difference	p-value
25:500–5:500	−0.518	<0.05	25:500–5:500	−0.491	<0.05
15:500–5:500	−0.472	<0.05	15:500–5:500	−0.464	<0.05
25:500–5:350	−0.462	<0.05	25:500–5:350	−0.435	<0.05
25:350–5:350	−0.449	<0.05	25:350–5:350	−0.431	<0.05
15:350–5:350	−0.424	<0.05	15:350–5:350	−0.416	<0.05
15:500–5:350	−0.416	<0.05	15:500–5:350	−0.408	<0.05
25:500–5:150	−0.308	<0.05	25:500–5:150	−0.292	<0.05
25:350–5:150	−0.296	<0.05	25:350–5:150	−0.287	<0.05
15:350–5:150	−0.271	<0.05	15:150–5:150	−0.275	<0.05
15:150–5:150	−0.264	<0.05	15:350–5:150	−0.272	<0.05
15:500–5:150	−0.263	<0.05	25:150–5:150	−0.265	<0.05
25:150–5:150	−0.239	<0.05	15:500–5:150	−0.265	<0.05
25:500–25:150	−0.070	<0.05	25:500–15:500	−0.027	<0.05
25:350–25:150	−0.057	<0.05	25:500–25:150	−0.027	<0.05
25:500–15:500	−0.045	<0.05	25:350–25:150	−0.023	<0.05
25:500–15:150	−0.044	<0.05	25:500–15:350	−0.019	<0.05
25:500–15:350	−0.037	<0.05	25:500–15:150	−0.017	<0.05
15:350–25:150	−0.032	<0.05	25:350–15:350	−0.015	<0.05
25:350–15:150	−0.032	<0.05	25:350–15:150	−0.013	<0.05
25:350–15:350	−0.025	<0.05	15:350–25:150	−0.007	<0.05
15:500–25:150	−0.024	<0.05	25:500–25:350	−0.004	<0.05
25:500–25:350	−0.012	<0.05	15:500–25:150	0.000	<0.05
15:350–15:150	−0.007	<0.05	15:350–15:150	0.003	<0.05
15:500–15:150	0.001	<0.05	15:500–15:350	0.007	<0.05
15:500–15:350	0.008	<0.05	25:150–15:150	0.010	<0.05
25:150–15:150	0.025	<0.05	15:500–15:150	0.010	<0.05
15:500–25:350	0.033	<0.05	15:500–25:350	0.023	<0.05
5:500–5:350	0.056	<0.05	5:500–5:350	0.055	<0.05
5:350–5:150	0.154	<0.05	5:350–5:150	0.144	<0.05
5:500–5:150	0.210	<0.05	5:500–5:150	0.199	<0.05
5:350–25:150	0.392	<0.05	5:350–25:150	0.408	<0.05
5:350–15:150	0.418	<0.05	5:350–15:150	0.419	<0.05
5:500–25:150	0.448	<0.05	5:500–25:150	0.464	<0.05
5:500–15:150	0.473	<0.05	5:500–15:350	0.471	<0.05
5:500–15:350	0.480	<0.05	5:500–15:150	0.474	<0.05
5:500–25:350	0.505	<0.05	5:500–25:350	0.486	<0.05

The following tables show the Tukey test for the number of attributes, alpha, number of patterns, and for interaction 1, considering the rates of faulty alarms.

Table A6
Tukey Test for number of atributes and faulty alarms.

Faulty Alarm – Class 1			Faulty Alarm – Class 2		
# of atributes	Difference	p-value	# of atributes	Difference	p-value
15–5	0.223195	<0.05	15–5	0.09902535	<0.05
25–5	0.17056	<0.05	25–5	0.07630743	<0.05
25–15	−0.05264	<0.05	25–15	−0.02271792	<0.05

Table A7
Tukey Test for alpha and faulty alarms.

Faulty Alarm – Class 1			Faulty Alarm – Class 2		
Alpha	Difference	p-value	*Alpha*	Difference	p-value
0.05–0.01	−0.54327	<0.05	0.05−0.01	−0.2948452	<0.05
0.1–0.01	−0.67227	<0.05	0.1−0.01	−0.35625786	<0.05
0.1–0.05	−0.129	<0.05	0.1−0.05	−0.06141266	<0.05

Table A8
Tukey Test for number of patterns and faulty alarms.

Faulty Alarm – Class 1			Faulty Alarm – Class 2		
# of patterns	Difference	p-value	# of patterns	Difference	p-value
350–150	−0.03712	<0.05	350–150	−0.018800632	<0.05
500–150	−0.05492	<0.05	500–150	−0.026536229	<0.05
500–350	−0.0178	<0.05	500–350	−0.007735597	<0.05

Table A9
Tukey Test for Interaction 1 (number of attributes x number of patterns) and false alarms.

Faulty Alarm – Class 1			Faulty Alarm – Class 2		
Number of attributes × number of patterns	Difference	p-value	Number of attributes × number of patterns	Difference	p-value
5:500–15:150	−0.30187	<0.05	5:500–15:150	−0.134824405	<0.05
5:500–15:350	−0.29343	<0.05	5:500–15:350	−0.126413139	<0.05
5:350–15:150	−0.26015	<0.05	5:350–15:150	−0.117690697	<0.05
5:500–25:350	−0.24156	<0.05	5:500–25:350	−0.104551477	<0.05
5:500–25:150	−0.22565	<0.05	5:500–25:150	−0.102978946	<0.05
5:350–25:150	−0.18393	<0.05	5:350–25:150	−0.085845238	<0.05
5:500–5:150	−0.16056	<0.05	5:500–5:150	−0.06669687	<0.05
5:350–5:150	−0.11884	<0.05	5:350–5:150	−0.049563161	<0.05
25:150–15:150	−0.07622	<0.05	25:150–15:150	−0.031845459	<0.05
25:350–15:150	−0.06031	<0.05	25:350–15:150	−0.030272928	<0.05
25:500–15:150	−0.05512	<0.05	25:500–15:150	−0.029601962	<0.05
25:350–15:350	−0.05187	<0.05	25:350–15:350	−0.021861662	<0.05
25:500–15:350	−0.04668	<0.05	25:500–15:350	−0.021190697	<0.05
5:500–5:350	−0.04172	<0.05	5:500–5:350	−0.017133708	<0.05
25:500–15:500	−0.02982	<0.05	15:500–15:150	−0.015155313	<0.05
15:500–15:150	−0.0253	<0.05	25:500–15:500	−0.014446649	<0.05
15:500–15:350	−0.01686	<0.05	15:350–15:150	−0.008411265	<0.05
15:350–15:150	−0.00844	<0.05	15:500–15:350	−0.006744048	<0.05
25:500–25:350	0.005192	<0.05	25:500–25:350	0.000670966	<0.05
25:350–25:150	0.01591	<0.05	25:350–25:150	0.001572531	<0.05
25:500–25:150	0.021103	<0.05	25:500–25:150	0.002243497	<0.05
15:500–25:350	0.035009	<0.05	15:500–25:350	0.015117615	<0.05
15:500–25:150	0.05092	<0.05	15:500–25:150	0.016690146	<0.05
25:150–5:150	0.065087	<0.05	15:350–25:150	0.023434193	<0.05
15:350–25:150	0.06778	<0.05	25:150–5:150	0.036282077	<0.05
25:350–5:150	0.080998	<0.05	25:350–5:150	0.037854608	<0.05
25:500–5:150	0.08619	<0.05	25:500–5:150	0.038525573	<0.05
15:500–5:150	0.116007	<0.05	15:500–5:150	0.052972222	<0.05
15:350–5:150	0.132867	<0.05	15:350–5:150	0.05971627	<0.05
15:150–5:150	0.141307	<0.05	15:150–5:150	0.068127535	<0.05
25:350–5:350	0.19984	<0.05	25:350–5:350	0.087417769	<0.05
25:500–5:350	0.205033	<0.05	25:500–5:350	0.088088735	<0.05
15:500–5:350	0.23485	<0.05	15:500–5:350	0.102535384	<0.05
25:500–5:500	0.246753	<0.05	25:500–5:500	0.105222443	<0.05
15:350–5:350	0.25171	<0.05	15:350–5:350	0.109279431	<0.05
15:500–5:500	0.27657	<0.05	15:500–5:500	0.119669092	<0.05

References

[1] C. Alippi, D. Liu, D. Zhao, L. Bu, Detecting and reacting to changes in sensing units: the active classifier case, IEEE Trans. Syst. Man Cybern.: Syst. 44 (3) (2014) 353–362.

[2] S.H. Bach, M.A. Maloof, Paired learners for concept drift, Proc. of the 8th IEEE Int. Conf. on Data Mining (ICDM) IEEE (2017) 23–32;
Charts for Detecting Concept Drift, Pattern Recogn. Lett. 33 (2) (2012) 191–198.

[3] M. Baena-García, J. Del Campo-Ávila, R. Fidalgo, A. Bifet, Early drift detection method, in: Proc. of the 4th ECML PKDD International Workshop on Knowledge Discovery From Data Streams (IWKDDS'06), Berlin, Germany, 2006, pp. 77–86.

[4] J.C. Bezdek, M.R. Pal, J. Keller, R. Krisnapuram, Fuzzy Models and Algorithms for Pattern Recognition and Image Processing, Kluwer Academic Publishers, Norwell MA, USA, 1999.

[5] A. Bifet, R. Gavaldà, Learning from time-changing data with adaptive windowing, in: Proceedings of the Seventh SIAM International Conference on Data Mining, Minneapolis, Minnesota, USA, 2007, pp. 443–448.

[6] C.M. Bishop, Pattern Recognition and Machine Learning, Springer, New York, 2006.

[8] D. Brzezinski, J. Stefanowski, Reacting to different types of concept drift: the accuracy updated ensemble algorithm, IEEE Trans. Neural Networks Learn. Syst. 25 (1) (2014) 81–94.

[9] R. Elwell, R. Polikar, Incremental learning of concept drift in nonstationary environments, IEEE Trans. Neural Netw. 22 (10) (2011) 1517–1531.

[10] B.S. Everitt, S. Landau, M. Leese, Cluster Analysis, 4th edition, Oxford University Press, Inc., New York; Arnold London, 2001.

[11] I. Frías-Blanco, J. del Campo-Avila, G. Ramos-Jimenez, R. Morales-Bueno, A. Ortiz-Díaz, Y. Caballero-Mota, Online and non-parametric drift detection methods based on Hoeffding's bounds, IEEE Trans. Knowledge Data Eng. 27 (3) (2015) 810–823.

[12] J. Gama, P. Medas, G. Castillo, P.P. Rodrigues, Learning with drift detection. Advances in artificial intelligence – SBIA 2004, in: 17th Brazilian Symposium on Artificial Intelligence, São Luis, Maranhão, Brazil, 29 – October 1, 2004, Proceedings, 2004, pp. 286–295.

[13] J. Gama, I. Žliobaite, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation, ACM Comput. Surv. (CSUR) 46 (4) (2014) 44.

[14] R.A. Johnson, D.W. Wichern, Applied Multivariate Statistical Analysis, Pearson Education Limited, 2014.

[15] M.T. Karnick, M. Ahiskali, M. Muhlbaier, R. Polikar, Learning Concept Drift in Nonstationary Environments Using an Ensemble of Classifiers Based Approach, IJCNN, 2008, pp. 3455–3462.

[16] I. Khamassi, M. Sayed-Mouchaweh, Drift detection and monitoring in non-stationary environments, IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS) IEEE (2014) 1–6.

[17] B. Krawczyk, L.L. Minku, J. Gama, J. Stefanowski, M. Woźniak, Ensemble learning for data stream analysis: a survey, Inf. Fusion 37 (2017) 132–156.

[18] L.I. Kuncheva, W.J. Faithfull, PCA feature extraction for change detection in multidimensional unlabeled data, IEEE Trans. Neural Networks Learn. Syst. 25 (1) (2014) 69–80.

[19] L.I. Kuncheva, Classifier Ensemble for Changing Environments, in Multiple Classifier Systems, vol. 3077, Springer-Verlag, New York, 2004.

[20] L.I. Kuncheva, Combining Pattern Classifiers: Methods and Algorithms, Wiley-Interscience, 2004.

[22] K.V. Mardia, The effect of nonnormality on some multivariate tests and robustness to nonnormality in the linear model, Biometrika 58 (1) (1971) 105–121.

[23] H. Maayan, S. Mannor, R. El-Yaniv, K. Crammer, Concept Drift Detection Through Resampling, ICML, 2014, pp. 1009–1017.

[24] I. Melnykov, V. Melnykov, On k-means algorithm with the use of Mahalanobis distances, Stat. Probab. Lett. 84 (2014) 88–95.

[25] MOA Datasets, MOA – Massive Online Analysis, 2017, At: http://moa.cms.waikato.ac.nz/datasets/ Last Access January 2015.

[26] D.C. Montgomery, Applied Statistics and Probability for Engineers, 6th edition, Wiley, 2013.

[27] K. Nishida, K. Yamauchi, Adaptive classifiers-ensemble system for tracking concept drift, in: Proceedings of the Sixth International Conference on Machine Learning and Cybernetics (ICMLC'07), Honk Kong, 2007, pp. 3607–3612.

[28] R. Polikar, R. Elwell, Benchmark Datasets for Evaluating Concept Drift/NSE Algorithms, 2017, At: http://users.rowan.edu/~polikar/research/NSE Last Access December 2013.

[29] S. Ramírez-Gallego, B. Krawczyk, S. García, M. Woźniak, F. Herrera, A survey on data preprocessing for data stream mining: current status and future directions, in: Neurocomputing, 2017.

[30] G.J. Ross, N.M. Adams, D.K. Tasoulis, D.J. Hand Ross, J. Gordon, et al., Exponentially weighted moving average charts for detecting concept drift, Pattern Recognit. Lett. 33 (2) (2012) 191–198.

[31] R. Sebastião, J. Gama, T. Mendonça, Fading histograms in detecting distribution and concept changes, Int. J. Data Sci. Analyt. (2017) 1–30.

[32] W.N. Street, Y.S. Kim, A streaming ensemble algorithm (SEA) for largescale classification, Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2001) 377–382.

[33] P. Sobolewski, M. Won ïak, Concept drift detection and model selection with simulated recurrence and ensembles of statistical detectors, J. Univ. Comput. Sci. 19 (4) (2013) 462–483.

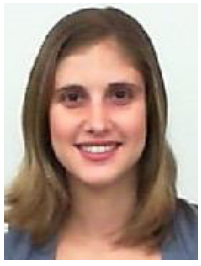
[34] Y. Sun, Z. Wang, H. Liu, C. Du, J. Yuan, Online ensemble using adaptive windowing for data streams with concept drift, Int. J. Distrib. Sens. Netw. 12 (5) (2016) 4218973.

[35] A. Thakre, S. Dongre, Review on concept drift detection techniques, Int. J. Recent Innov. Trends Comput. Commun. 4 (3) (2016) 404–407.

[37] A. Tsymbal, The problem of concept drift: Definitions and related work, in: Tech. Rep., 2004.

[38] R.M.M. Vallim, R.F. de Mello, Proposal of a new stability concept to detect changes in unsupervised data streams, Expert Syst. Appl. 41 (16) (2014) 7350–7360.

[39] G. Widmer, M. Kubat, Learning in the presence of concept drift and hidden contexts, Mach. Learn. 23 (1) (1996) 69–101.
[40] B. Zhang, L. Xue, W. Wang, S. Qin, D. Wang, Model updating mechanism of concept drift detection in data stream based on classifier pool, EURASIP J. Wireless Commun. Network. (1) (2016) 2016.
[41] I. Zliobaite, A. Bifet, M. Gaber, B. Gabrys, J. Gama, L. Minku, K. Musial, Next challenges for adaptive learning systems, ACM SIGKDD Explor. Newslett. 14 (1) (2012) 48–55.
[42] I. Zliobaite, Learning Under Concept Drift: An Overview, Tech. rep. Vilnius University, 2009.
[43] T. Escovedo, A. Koshiyama, M. Vellasco, R. Melo, A.A. da Cruz, A2D2: A pre-event abrupt drift detection, in: International Joint Conference on Neural Networks (IJCNN), Killarney, 2015, pp. 1–8.
[44] K. Chen, K.S. Yun, P. Riddle, Proactive drift detection: predicting concept drifts in data streams using probabilistic networks, International Joint Conference on Neural Networks (IJCNN) IEEE (2016) 780–787.
[45] Z. Ahmadi, S. Kramer, Modeling recurring concepts in data streams: a graph-based framework, Knowledge and Information Systems (2017) 1–30.
[46] G. da C. Fausto, F.S.L.G. Duarte, R.M.M. Vallim, R.F. de Mello, Multidimensional surrogate stability to detect data stream concept drift, Expert Syst. Appl. 87 (2017) 15–29.



Tatiana Escovedo received the BSc and MSc degrees in Computer Science from the Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Brazil, in 2005 and 2007, respectively, and the PhD degree in Electrical Engineering from the Pontifical Catholic University of Rio de Janeiro (PUC-Rio) in 2015. Dr. Escovedo is currently the coordinator of predictive analysis of Petrobras' business risk department in Rio de Janeiro, Brazil, and assistant professor at PUC-Rio. She is the author of several papers in the area of software engineering and machine learning. Her research interests include Data Mining, Artificial Intelligence, Software Engineering, Machine Learning and Business Intelligence.

Adriano Soares Koshiyama received his BSc degree in Economics from UFRRJ and MSc degree in Electrical Engineering from PUC-Rio. Nowadays is a PhD Candidate in Computer Science at University College London (UCL), with its main research subject being in Financial Computing and Analytics. Its main research topics are related to: Machine Learning, Statistical Methods, Optimization and Finance.

Andre Vargas Abs da Cruz received the BSc. in Computer Engineering at the Pontifical Catholic University of Rio de Janeiro (1998), MSc. in Electric Engineering (Support Decision Methods) at the Pontifical Catholic University of Rio de Janeiro (2003) and DSc. in Electric Engineering (Support Decision Methods) at the Pontifical Catholic University of Rio de Janeiro (2007). Did a post-doctoral research in bioinformatics. Has experience on the following subjects: optimization, evolutionary algorithms, quantum computing, bioinformatics and neural networks. He currently works as a Data Scientist for MDC Partners in Antwerp, Belgium.

Marley Maria Bernardes Rebuzzi Vellasco received the BSc and MSc degrees in Electrical Engineering from the Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Brazil, in 1984 and 1987, respectively,and the PhD degree in Computer Science from the University College London (UCL) in 1992. Dr. Vellasco is currently Head of the Electrical Engineering Department of PUC-Rio and of the Computational Intelligence and Robotics Laboratory (LIRA) of PUC-Rio. She is the author of four books and more than 60 papers in professional journals, 340 papers in conference proceedings and 17 book chapters in the area of soft computing and machine learning. Her research interests include Neural Networks, Fuzzy Logic, Neuro-Fuzzy Systems, Neuro-Evolutionary models, Robotics, and Intelligent Agents, applied to decision support systems, pattern classification, time-series forecasting, control, optimization and Data Mining.