Int J Data Sci Anal (2017) 3:183–212
DOI 10.1007/s41060-017-0043-4

REGULAR PAPER

Fading histograms in detecting distribution and concept changes

Raquel Sebastião[1,2] · **João Gama**[2,3] · **Teresa Mendonça**[4,5]

Received: 29 June 2016 / Accepted: 25 January 2017 / Published online: 23 February 2017
© Springer International Publishing Switzerland 2017

Abstract The remarkable number of real applications under dynamic scenarios is driving a novel ability to generate and gather information. Nowadays, a massive amount of information is generated at a high-speed rate, known as data streams. Moreover, data are collected under evolving environments. Due to memory restrictions, data must be promptly processed and discarded immediately. Therefore, dealing with evolving data streams raises two main questions: (i) how to remember discarded data? and (ii) how to forget outdated data? To maintain an updated representation of the time-evolving data, this paper proposes fading histograms. Regarding the dynamics of nature, changes in data are detected through a windowing scheme that compares data distributions computed by the fading histograms: the adaptive cumulative windows model (ACWM). The online monitoring of the distance between data distributions is evaluated using a dissimilarity measure based on the asymmetry of the Kullback–Leibler divergence. The experimental results support the ability of fading histograms in providing an updated representation of data. Such property works in favor of detecting distribution changes with smaller detection delay time when compared with standard histograms. With respect to the detection of concept changes, the ACWM is compared with 3 known algorithms taken from the literature, using artificial data and using public data sets, presenting better results. Furthermore, we the proposed method was extended for multidimensional and the experiments performed show the ability of the ACWM for detecting distribution changes in these settings.

Keywords Data streams · Fading histograms · Data monitoring · Distribution changes · Concept changes

✉ Raquel Sebastião
raquel.sebastiao@ua.pt

João Gama
jgama@fep.up.pt

Teresa Mendonça
tmendo@fc.up.pt

1 Institute of Electronics and Informatics Engineering of Aveiro (IEETA) & Department of Electronics, Telecommunications and Informatics (DETI), University of Aveiro, Universitário de Santiago, 3810-193 Aveiro, Portugal

2 LIAAD INESC Tec, INESC, Campus da FEUP, Rua Dr. Roberto Frias, 4200-465 Porto, Portugal

3 School of Economics, University of Porto, Porto, Portugal

4 Department of Mathematics, Faculty of Sciences, University of Porto, Rua do Campo Alegre, s/n, 4169-007 Porto, Portugal

5 SYSTEC - Research Center for Systems and Technologies, FEUP, Rua Dr. Roberto Frias, i219, 4200-464 Porto, Portugal

1 Introduction

The most recent developments in science and information technology are spreading the computational capacity of smart devices, which are capable to produce massive amounts of information at a high-speed rate, known as data streams. A data stream is a sequence of information in the form of transient data that arrives continuously (possibly at varying times) and is potentially infinite. Therefore, it is unreasonable to assume that machine learning systems have sufficient memory capacity to store the complete history of the stream. Indeed, stream learning algorithms must process data promptly and discard it immediately. In this context, it is essential to create synopses structures of data, keeping only a small and finite representation of the gathered information and allowing the discarded data to be remembered.

Along with this, as data flow continuously for large periods of time, the process generating data is not strictly stationary

and evolves over time. Therefore, it is of utmost importance to maintain a stream learning model consistent with the most recent data. When dealing with data streams in dynamics environments, besides remembering discarded data, it is necessary to forget outdated data. To accomplish such assignments, this paper advances fading histograms, which weight data examples according to their age. Thus, while remembering the discarded data, fading histograms gradually forget old data.

Moreover, the dynamics of environments faced nowadays raise the need of performing online change detection tests. In this context, the delay between the occurrence of a change and its detection must be minimal. When data flow over time and for large periods of time, it is unlikely the assumption that the observations are generated, at random, according to a stationary probability distribution [5]. As the underlying distribution of data may change over time, old observations do not describe the current state of nature and are useless. Therefore, it is of paramount interest to perceive if and when there is a change.

Despite aging, the problem of dealing with changes in a signal has caught the attention of the scientific community in recent years due to the emergence of real word applications. The online analysis of the gathered signals is of foremost importance, especially in those cases where actions must be taken after the occurrence of a change. From this point of view, it is essential to detect a change as soon as possible, ideally immediately after it occurs. Minimizing the detection delay time is of great importance in applications such as real-time monitoring in biomedicine and industrial processes, automatic control, fraud detection, safety of complex systems and many others. Widely used in the data stream context [7,13,25,35], windowing approaches for detecting changes in data consist of monitoring distributions over two different time-windows, performing tests to compare distributions and decide if there is a change. This paper proposes a windowing model for change detection, which evaluates, through a dissimilarity measure based on the asymmetry of the Kullback–Leibler divergence, the distance between data distributions provided by fading histograms.

Previous work, contributions and paper outline

A previous work [37] presented a detailed description of the construction of fading histograms and compared the performance of these with sliding histograms, both feasible approaches to cope with the problem of remember data in the context of high-speed and massive data streams and to forget outdated data when in dynamic scenarios. Other previous work [36] introduces an adaptive model for detecting changes in data distribution, employing this summarization approach to compute distributions.

The main contribution of this paper is the detailed description of the adaptive cumulative windows model (ACWM) for detecting data distribution and concept changes and the introduction of this model in multidimensional settings. Moreover, the experimental section evaluates the overall performance of the ACWM in detecting distribution changes in different evolving scenarios and it is compared with other algorithms when detecting concept drift.

This paper is organized as follows. It starts with the problem of constructing fading histograms from data streams. Section 3 addresses the problem of detecting distribution and concept changes. In Sect. 4 windowing schemes for change detection are presented and Sect. 5 proposes a windowing model to compare data for detecting changes in data distribution. Section 6 evaluates the performance of the ACWM with respect to the ability to detect distribution changes, in artificial and real-world data sets and compares results with those obtained with the Page–Hinkley Test (PHT). In Sect. 7, the ability to detect concept changes is assessed and the presented approach is compared with 3 algorithms: DDM (Drift Detection Method), ADWIN (ADaptive WINDdowing) and PHT. The performance of the ACWM when detecting changes in multidimensional data is also evaluated. Finally, Sect. 8 presents conclusions on the ACWM and advances directions for further research.

2 Data summarization

When very large volumes of data arrive at a high-speed rate, it is impractical to accumulate and archive in memory all observations for later use. Nowadays, the scenario of finite stored data sets is no longer appropriate because information is gathered assuming the form of transient and infinite data streams and may not even be stored permanently. Therefore, it is unreasonable to assume that machine learning systems have sufficient memory capacity to store the complete history of the stream.

This implies to create compact representations of data when dealing with massive data streams. Memory restrictions preclude keeping all received data in memory. These restrictions impose that in the data stream systems, the data elements are quickly and continuously received, promptly processed and discarded immediately. Since data elements are not stored after being processed it is necessary to use synopses structures to create compact summaries of data, keeping only a small and finite representation of the received information. As a result of the summarization process, the size of a synopsis structure is small in relation to the length of the data stream represented. Reducing memory occupancy is of utmost importance when handling a huge amount of data. Along with this, without the need of accessing the entire stream, data synopses allow fast and relative approximations to be obtained in a wide range of problems, such as

range queries, selectivity estimation, similarity searching and database applications, classification tasks, change detection and concept drift.

As for the wide range of problems in which data synopses are useful, it is of paramount interest that these structures have broad applicability. This is a fundamental requirement for using the same data synopsis structure in different applications, reducing time and space efficiency in the construction process. The data stream context under which these synopses are used also imposes that their construction algorithms must be single pass, time efficient and have, at most, space complexity linear in relation to the size of the stream. Moreover, in most cases, data are not static and evolves over time. Synopses construction algorithms must allow online updates on the synopses structures to keep up with the current state of the stream.

Different kinds of summarization techniques can be considered in order to provide approximated answers to different queries. The online update of such structures in a dynamic scenario is also a required property. Sampling [39], hot lists [11,30], wavelets [9,18,24], sketches [10] and histograms [21–23] are examples of synopses methods to obtain fast and approximated answers.

Fading histograms

A histogram is a synopsis structure that allows accurate approximations of the underlying data distribution and provides a graphical representation of a random variable. Histograms are widely applied to compute aggregate statistics, to approximate query answering, query optimization and selectivity estimation [22].

Consisting of a set of k non-overlapping intervals (also known as buckets or bins), a histogram is visualized as a bar graph that shows frequency data. The values of the random variable are placed into non-overlapping intervals, and the height of the bar drawn on each interval is proportional to the number of observed values within that interval.

To construct histograms in the stream mining context, there are some requirements that need to be fulfilled: The algorithms must be one-pass, supporting incremental maintenance of the histograms, and must be efficient in time and space [20,21]. Moreover, the updating facility and the error of the histogram are the major concerns to embrace when constructing online histograms from data streams.

In the proposed histograms, the definition of the number of buckets is related with the error of the histogram: following the equi-width strategy, the number of buckets is chosen under error constraints [37]. Let ε be the admissible error for the mean square error of a histogram H_k with k buckets. Then, considering that R is the admissible range of the variable under study, the mean square error of an equi-width histogram with at least $\frac{R}{2\sqrt{\varepsilon}}$ buckets is, at most, ε. With respect to the binning strategy, the equi-width histograms were chosen based on the following reasons:

- The construction is effortless: It simply divides the admissible range R of the random variable into k non-overlapping intervals with equal width.
- The updating process is easy: Each time a new data observation arrives, it just identifies the interval where it belongs and increments the count of that interval.
- Information visualization is simple: The value axis is divided into buckets of equal width.

Let i be the current number of observations of a given variable X from which a histogram is being constructed. A histogram H_k is defined by a set of k buckets $B_1, \ldots, B_k$ in the range of the random variable and a set of frequency counts $F_1(i), \ldots, F_k(i)$.

Definition 1 Let k be the number of non-overlapping intervals of a histogram. For each time instance i, the histogram frequencies are defined as:

$$F_j(i) = \frac{\sum_{l=1}^{i} C_j(l)}{i}, \quad \forall j = 1, \ldots, k \tag{1}$$

where C_j is the count of bucket B_j:

$$C_j(i) = \begin{cases} 1 & if\ x_i \in B_j \\ 0 & \text{otherwise} \end{cases} \quad \forall j = 1, \ldots, k$$

A standard histogram attributes the same importance to all observations. However, in dynamic scenarios, recent data are usually more important than old data. Therefore, outdated data can be gradually forgotten attributing different weights to data observations. In an exponential approach, the weight of data observations decreases exponentially with time. Exponential fading factors have been applied successfully in data stream evaluation [16]. Figure 1 illustrates the weight of examples according to their age, considering an exponential approach.

Following an exponential forgetting, histograms can be computed using fading factors, henceforth referred to as fading histograms. In this sense, data observations with high weight (the recent ones) contribute more to the fading histogram than observations with low weight (the old ones).

Definition 2 Let k be the number of buckets of a fading histogram. For each observation x_i of a given variable X, the histogram α-frequencies are defined as:

$$\begin{aligned} F_{\alpha,j}(i) &= \frac{\sum_{l=1}^{i} \alpha^{i-l} C_j(l)}{\sum_{j=1}^{k} \sum_{l=1}^{i} \alpha^{i-l} C_j(l)} \\ &= \frac{\sum_{l=1}^{i} \alpha^{i-l} C_j(l)}{\frac{1-\alpha^i}{1-\alpha}}, \forall j = 1, \ldots, k, \end{aligned} \tag{2}$$

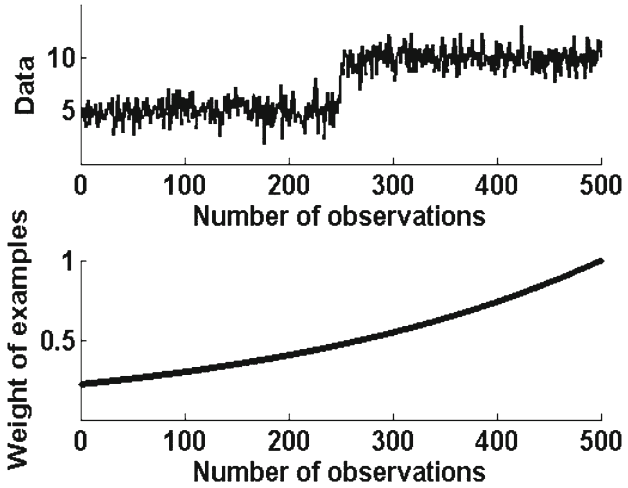

Fig. 1 The weight of examples as a function of age, in an exponential approach

where α, real number parameter, is the exponential fading factor such that $0 \ll \alpha < 1$.

According to this definition, old data are forgotten gradually, since it contributes less than recent data. Assuming that the observation x_i belongs to bin m (with $1 \leq m \leq k$), the recursive form enables the construction of the fading histograms counts in the flow:

$$C_j = \alpha * C_{j-1}, \forall j = 1, \ldots, k$$
$$C_m = C_m + 1, 1 \leq m \leq k \quad (3)$$

To exemplify the forgetting ability of fading histograms with respect to histograms constructed over the entire stream, artificial data were generated from two normal distributions. The initial 2500 observations follow a normal distribution with mean 5 and standard deviation 1 and the remaining 2500 observations follow a normal distribution with mean 10 and the same standard deviation.

For illustrative purposes, the number of buckets in each histogram was set to 20 (considering an admissible error $\varepsilon = 0.1$ for the mean square error of the histograms and using the approach proposed in [37]) and the value of the fading factor for the fading histograms was set to $\alpha = 0.997$.

Figure 2 (top) shows the artificial data with a change at observation 2500. The remaining plots display fading histograms and histograms constructed over the entire stream, at different observations: 2000, 3000 and 4000.

From the first representations, while in the presence of a stationary distribution, it turns out that both histograms produce similar representations of the data distribution. The second and the third representations present a different scenario. At observation 3000, after the change occurred at observation 2500, the representations provided by both histograms strategies become quite different. It can be observed that in the fading histogram representation, the buckets for the second distributions are reinforced, which does not occur on the histogram constructed over the entire stream. Indeed, contrary to the histograms constructed over the entire stream, fading histograms capture the change better as there is an enhancement of the fulfillment of the buckets for the second distribution. At observation 4000, it can be seen that the fading histogram produces a representation that keeps up with the current state of nature, forgetting outdated data (it must be pointed out that although they appear to be empty, the buckets for the first distribution present very low frequencies). At the same observation, in the histogram constructed over the entire stream, the buckets for the first distribution still quite filled, which is not in accordance with the current observations. From these representations, it can be observed the ability of fading histograms to forget outdated data, since the buckets from the initial distribution presented smaller values than the corresponding ones in the histogram constructed over the entire data, while the buckets from the second distribution have higher values. Indeed, fading histograms reinforces the capture of changes in evolving stream scenarios.

3 The change detection problem

A data stream is a sequence of information in the form of transient data that arrives continuously (possibly at varying times) and is potentially infinite. Along with this, as data flow for long periods of time, the process generating data is not strictly stationary and evolves over time.

Despite aging, the problem of detecting changes in a signal has caught the attention of the scientific community in recent years due to the emergence of real-world applications. Such applications require the online analysis of the gathered signals: especially in those cases where actions must be taken after the occurrence of a change. From this point of view, it is essential to maintain a stream learning model consistent with the most recent data, forgetting outdated data. Moreover, it is of utmost importance to detect a change as soon as possible, ideally immediately after it occurs. This reduces the delay time between the occurrence of the change and its detection. Minimizing the detection delay time is of great importance in applications such as real-time monitoring in biomedicine and industrial processes, automatic control, fraud detection, safety of complex systems and many others.

3.1 Distribution changes

In the dynamic scenarios faced nowadays, it is an unlikely the assumption that the observations are generated, at random, according to a stationary probability distribution [5]. Changes in the distribution of the data are expected. As the

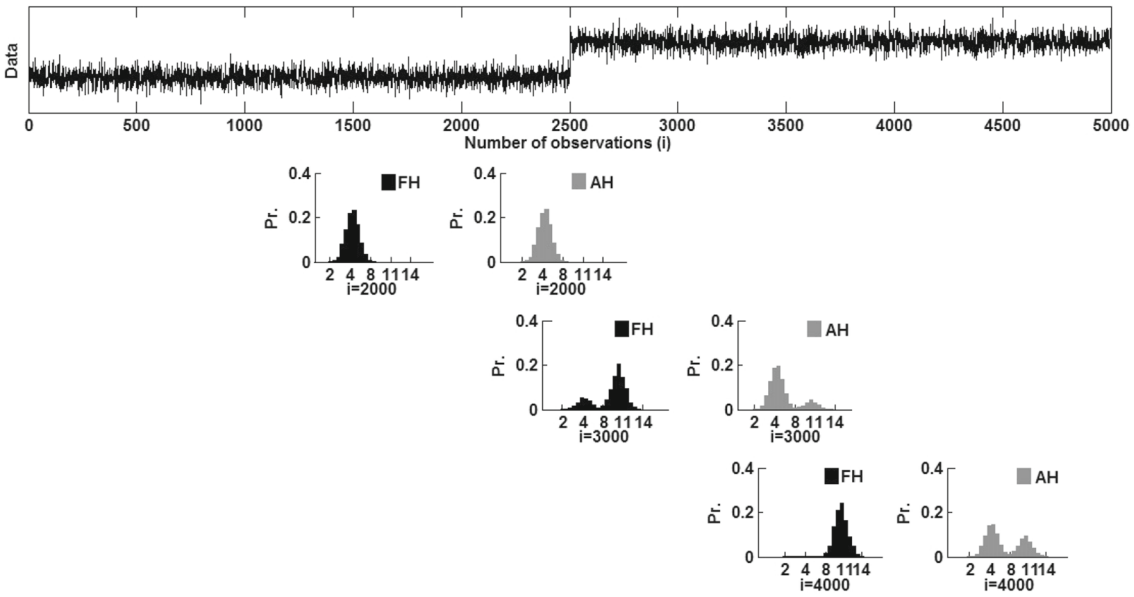

Fig. 2 Comparison of histograms computed with a fading factor of $\alpha = 0.997$ (FH) and histograms constructed over the entire stream (AH)

underlying distribution of data may change over time, it is of utmost importance to perceive if and when there is a change.

The distribution change detection problem is concerned with the identification of the time of occurrence of a change (or several changes) in the probability distribution of a data sequence. Figure 3 illustrates this problem. In this example, P_0 is the probability distribution of the observations seen in the past and P_1 is the probability distribution of the most recent observed data.

Consider that $x_1, x_2, \ldots$ is a sequence of random observations, such that $x_t \in \mathbb{R}, t = 1, 2, \ldots$ (unidimensional data stream). Consider that there is a change point at time t^* with $t^* \geq 1$, such that the subsequence $x_1, x_2, \ldots, x_{t^*-1}$ is generated from a distribution P_0 and the subsequence $x_{t^*}, x_{t^*+1}, \ldots$ is generated from a distribution P_1.

A change is assigned if the distribution P_0 differs significantly from the distribution P_1. In this context, it means that the distance between both distributions is greater than a given threshold.

The change detection problem relies on testing the hypothesis that the observations are generated from the same distribution and the alternative hypothesis that they are generated from different distributions: $H_0 : P_0 \equiv P_1$ versus $H_1 : P_0 \ncong P_1$. The goal of a change detection method is to decide whether or not to reject H_0.

Whenever the alternative hypothesis is verified, the change detection method reports an alarm. The correct detection of a change is a hit; a non-detection of an occurred change is a miss or a false positive. Incorrectly detecting a change that does not occur is a false alarm or false negative. An effective change detection method must present few false events and detect changes with a short delay time.

The essence of a distribution change can be categorized according to three main characteristics:

- *Rate* The rate of a change (also known as speed) is extremely important in a change detection problem, describing whether a signal changes between distributions suddenly, incrementally, gradually or recurrently. Besides the intrinsic difficulties that each of these kinds of rates impose to change detection methods, real data streams often present several combinations of different rates of change.
- *Magnitude* The magnitude of change (also known as severity) is also a characteristic of paramount importance. In the presence of a change, the difference between distributions of the signal can be high or low. Despite being closely related, the magnitude and the rate of a change describe a different pattern of a change. Abrupt changes are easily observed and detected. Hence, in most cases, they do not pose great difficulties to change detection methods. What is more, these changes are the most critical ones because the distribution of the signal changes abruptly. However, smooth changes are more difficult to be identified. At least in the initial phases, smooth changes can easily be confused with noise [14]. Since noise and examples from another distribution are differentiated by permanence, the detection of a smooth change in an early phase, tough to accomplish, is of foremost interest.
- *Source* Besides other features that also describe a distribution of a data set (such as skewness, kurtosis, median, mode), in most cases, a distribution is characterized by the mean and variance. In this sense, a change in data

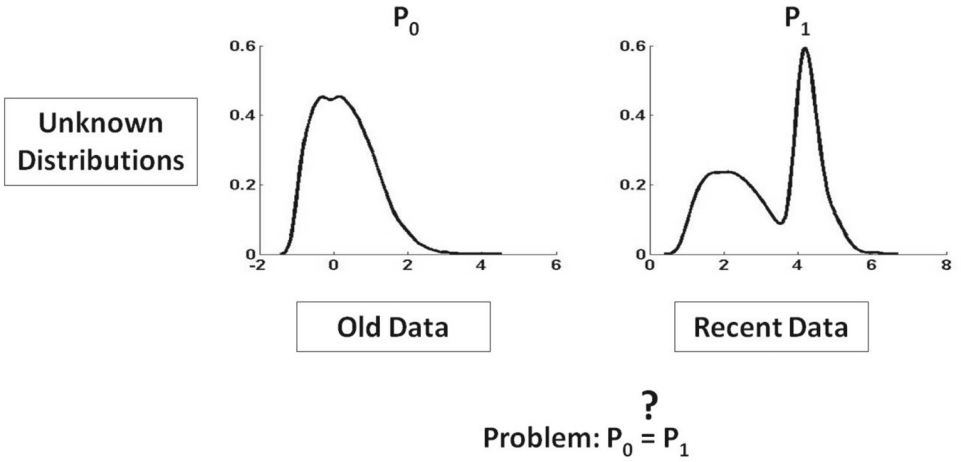

Fig. 3 Illustration of a distribution change

distribution can be translated by a change in the mean or by a change in variance. While a change in the mean does not pose great challenges to a change detection method, a change in the variance tends to be more difficult to detect (considering that both presented similar rate and magnitude).

3.2 Concept changes

The concept change problem is found in the field of machine learning and is closely related to the distribution change problem. A change in the concept means that the underlying distribution of the target concept may change over time [40]. In this context, concept change describes changes that occur in a learned structure.

Consider a learning scenario, where a sequence of instances $\mathbf{X}_1, \mathbf{X}_2, \ldots$ is being observed (one at a time and possibly at varying times), such that $\mathbf{X}_t \in \mathbb{R}^p, t = 1, 2, \ldots$ is an instance p-dimensional feature vector and y_t is the corresponding label, $y_t \in \{C_1, C_2, \ldots, C_k\}$. Each example $(\mathbf{X}_t, y_t), t = 1, 2, \ldots$ is drawn from the distribution that generates the data $P(\mathbf{X}_t, y_t)$. The goal of a stream learning model is to output the label y_{t+1} of the target instance $\mathbf{X}_{t+1}$, minimizing the cumulative prediction errors during the learning process. This is remarkably challenging in environments where the distribution that is generating the examples changes: $P(\mathbf{X}_{t+1}, y_{t+1})$ may be different from $P(\mathbf{X}_t, y_t)$.

For evolving data streams, some properties of the problem might change over time, namely the target concept on which data are obtained may shift from time to time, on each occasion after some minimum of permanence [14]. This time of permanence is known by context and represents a set of examples from the data stream where the underlying distribution is stationary. In learning scenarios, changes may occur due to modifications in the context of learning (caused by changes in hidden variables) or in the intrinsic properties of the observed variables.

Concept change can be formalized as a change in the joint probability distribution $P(\mathbf{X}, y)$:

$$P(\mathbf{X}, y) = P(y|\mathbf{X}) \times P(\mathbf{X})$$

Therefore, a concept change can be explained through a change in the class-conditional probability (conditional change) and/or in the feature probability (feature change) [17].

Concept changes can be addressed by assessing changes in the probability distribution (class-conditional distributions or prior probabilities for the classes), changes due to different feature relevance patterns, modifications in the learning model complexity and increases in the classification accuracy [27].

In a supervised learning problem, at each time stamp t, the class prediction $\hat{y}_t$ of the instance $\mathbf{X}_t$ is outputted. After checking the class y_t, the error of the algorithm is computed. For consistent learners, according to the *Probability Approximately Correct* (PAC) learning model [31] if the distribution of examples is stationary, the error rate of the learning model will decrease when the number of examples increases.

Detecting concept changes under non-stationary environments is, in most of the cases, inferred by monitoring the error rate of the learning model [4,15,33]. In such problems, the key to figuring out if there is a change in the concept is to monitor the evolution of the error rate. A significant increase in the error rate suggests a change in the process generating data. For long periods of time, it is reasonable to assume that the process generating data will evolve. When there is a concept change, the current learning model no longer corresponds to the current state of the data. Indeed, whenever new concepts replace old ones, the old observations become irrelevant and thus the model will become inaccurate. Therefore, the predictions outputted are no longer correct and the error rate will increase. In such cases, the learning model

must be adapted in accordance with the current state of the phenomena under observation.

As well as for the distribution changes, concept change can also be categorized according to the rate, magnitude and source of concept change. Although, regarding the source, a change in the concept can be translated as a change in the mean, variance and correlation of the feature value distribution. Moreover, the literature categorizes concept changes into concept drift and concept shift according to the rate and magnitude of the change. A concept drift occurs when the change presents a sudden rate and an high magnitude, while a concept shift designates a change with gradual rate and low magnitude.

4 Windowing methods for change detection

A windows-based change detection method consists of monitoring distributions over two different time-windows, performing tests to compare distributions and decide if there is a change. It assumes that the observations in the first window of length L_0 are generated according to a stationary distribution P_0 and that the observations in the second window of length L_1 are generated according to a distribution P_1. A change is assigned if the distribution P_0 differs significantly from the distribution P_1:

$D_{t^*}(P_0||P_1) = \max_{L_0<t} D_t(P_0||P_1) > \lambda$, where λ is known as the detection threshold.

The method outputs that distribution changes at the change point estimate t^*. In this context, it means that the distance between both distributions is greater than a given threshold.

In such models, the data distribution on a reference window, which usually represents past information, is compared to the data distribution computed over a window from recent examples [13,25,35]. Within a different conception, Bifet and Gavaldà [7] propose an adaptive windowing scheme to detect changes: the ADaptive WINDdowing (ADWIN) method. The ADWIN keeps a sliding window W with the most recently received examples and compares the distribution in two sub-windows (W_0 and W_1) of the former. Instead of being fixed a priori, the size of the sliding window W is determined online according to the rate of change observed in the window itself (growing when the data are stationary and shrinking otherwise). Based on the use of the Hoeffding bound, whenever two *large enough* sub-windows, W_0 and W_1, exhibit *distinct enough* averages, the older sub-window is dropped and a change in the distribution of examples is assigned. When a change is detected, the examples inside W_0 are thrown away and the window W slides keeping the examples belonging to W_1. With the advantage of providing guarantees on the rates of false positives and false negatives, the ADWIN is computationally expensive, as it compares all possible sub-windows of the recent window. To cutoff the number of possible sub-windows in the recent window, the authors have enhanced ADWIN. Using a data structure that is a variation of exponential histograms and a memory parameter, ADWIN2 reduces the number of possible sub-windows within the recent window.

The windows-based approach proposed by Kifer et al. [25] provides statistical guarantees on the reliability of detected changes and meaningful descriptions and quantification of these changes. The data distributions are computed over an ensemble of windows with different sizes, and the discrepancy of distributions between two pairs of windows (with the same size) is evaluated performing statistical hypothesis tests, such as Kolmogorov–Smirnov and Wilcoxon, among others. Avoiding statistical tests, the adjacent windows model proposed by Dasu et al. [13] measures the difference between data distributions by the Kullback–Leibler distance and applies bootstrapping theory to determine whether such differences are statistically significant. This method was applied to multidimensional and categorical data, showing to be efficient and accurate in higher dimensions.

Addressing concept change detection, the method proposed by Nishida and Yamauchi [33] detects concept changes in online learning problems, assuming that the concept is changing if the accuracy of the classifier in a recent window of examples decreases significantly compared to the accuracy computed over the stream hitherto. This method is based on the comparison of a computed statistic, equivalent to the Chi-square test with Yates's continuity correction and the percentile of the standard normal distribution. Using two levels of significance, the method stores examples in short-term memory during a warning period. If the detection threshold is reached, the examples stored are used to rebuild the classifier and all variables are reset. Later, Bach and Maloof [3] propose paired learners to cope with concept drifts. The stable learner predicts based on all examples, while the active learner predicts based on a recent window of examples. Using differences in accuracy between the two learners over the recent window, drift detection is performed and whenever the target concept changes the stable learner is replaced by the reactive one.

The work presented in Kuncheva [27] goes beyond the methods addressed in this section. Instead of using a change detector, it proposes an ensemble of windows-based change detectors. Addressing adaptive classification problems, the proposed approach is suitable for detecting concept changes either in labeled and unlabeled data. For the labeled data, the classification error is recorded and a change is signaled comparing the error on a sliding window with the mean error hitherto. For labeled data, computing the classification error is straightforward; hence, it is quite common to monitor the error or some error-based statistic to detect concept drift on the assumption that an increase in the error results from a change. However, when the labels of the data are not avail-

able, the error rate cannot be used as a performance measure of drifts. Therefore, changes in unlabeled data are handled by comparing cluster structures from windows with different length sizes. The advantage of an ensemble of change detectors is disclosed by their ability to effectively detect different kinds of changes.

The evaluation of the performance of change detection methods in time-changing environments is quantitatively assessed by measuring the following standard criteria:

- *Detection delay time* The number of examples required to detect a change after the occurrence of one.
- *Missed detections* Ability to not fail the detection of real changes.
- *False alarms* Resilience to false alarms when there is no change, which means that the change detection method is not detecting changes under static scenarios.

5 Adaptive cumulative windows model (ACWM)

The ACWM for change detection is based on online monitoring of the distance between data distributions (provided by fading histograms), which is evaluated using the Kullback–Leibler divergence (KLD) [26]. Within this approach, the reference window (RW) has a fixed length and reflects the data distribution observed in the past. The current window (CW) is cumulative, and it is updated sliding forward and receiving the most recent data. The evaluation step is determined automatically depending on the data similarity.

In change detection problems, it is mandatory to detect changes as soon as possible, minimizing the delay time in detection. Along with this, the false and the missed detections must be minimal. Therefore, the main challenge when proposing an approach for change detection is reaching a trade-off between the robustness to false detections (and noise) and sensitivity to true changes.

It must be pointed out that the ACWM is a nonparametric approach, which means that it makes no assumptions on the form of the distribution. This is a property of major interest, since real data streams rarely follow known and well-behavior distributions.

Figure 4 shows the workflow of the ACWM for change detection. The histograms representations were constructed from the observed data, with different number of observations. At every evaluation step, the data distribution in the Current Window (CW) is compared with the distribution of the data in the Reference Window (RW). If a change in the distribution of the data in the CW with respect to the distribution of the data in the RW is not detected, the CW is updated with more data observations. Otherwise, if a change is detected, the data in both windows are cleaned, new data are used to fulfill both windows and a new comparison starts.

5.1 Distance between distributions

From information theory [6], the relative entropy is one of the most general ways of representing the distance between two distributions [13]. Contrary to the mutual information, this measure assesses the dissimilarity between two variables. Also known as the Kullback–Leibler divergence, it measures the distance between two probability distributions and therefore is suitable for use in comparison purposes.

Assuming that the data in the reference window have distribution P_{RW} and that data in the current window have distribution P_{CW}, the Kullback–Leibler divergence (KLD) is used as a measure to detect whenever a change in the distribution has occurred.

Considering two discrete distributions with empirical probabilities $P_{RW}(i)$ and $P_{CW}(i)$, the relative entropy of P_{RW} with respect to P_{CW} is defined by:

$$KLD(P_{RW}||P_{CW}) = \sum_i P_{RW}(i) log \frac{P_{RW}(i)}{P_{CW}(i)}.$$

Since it is asymmetric, the Kullback–Leibler divergence is a quasi-metric:

$$KLD(P_{RW}||P_{CW}) \neq KLD(P_{CW}||P_{RW}).$$

Nevertheless, it satisfies many important mathematical properties: is a nonnegative measure, it is a convex function of $P_{RW}(i)$ and equals zero only if $P_{RW}(i) = P_{CW}(i)$.

Consider a reference window with empirical probabilities $P_{RW}(i)$, and a current sliding window with probabilities $P_{CW}(i)$. Taking into account the asymmetric property of the Kullback–Leibler divergence and that the minimal value of the absolute difference between $KLD(P_{RW}||P_{CW})$ and $KLD(P_{CW}||P_{RW})$, which is zero, is achieved when P=Q: Smaller values of this difference correspond to smaller dispersion between both data distributions, meaning that the data are similar; and higher values of this difference suggest that distributions are further apart. Other metrics, namely the entropy absolute difference and the cosine distance, were considered in Sebastião and Gama [35]. When compared with the Kullback–Leibler divergence, this measure outperforms the others.

5.2 Decision rule

Consider a reference window with empirical probabilities $P_{RW}(i)$ and a current sliding window with probabilities $P_{CW}(i)$: Lower values of $KLD(P_{RW}||P_{CW})$ correspond to smaller dispersion between both data distributions, meaning that the data are similar. A higher value of $KLD(P_{RW}||P_{CW})$ suggests that distributions are further apart. Therefore, due to the asymmetric property of the KLD, if the distributions are

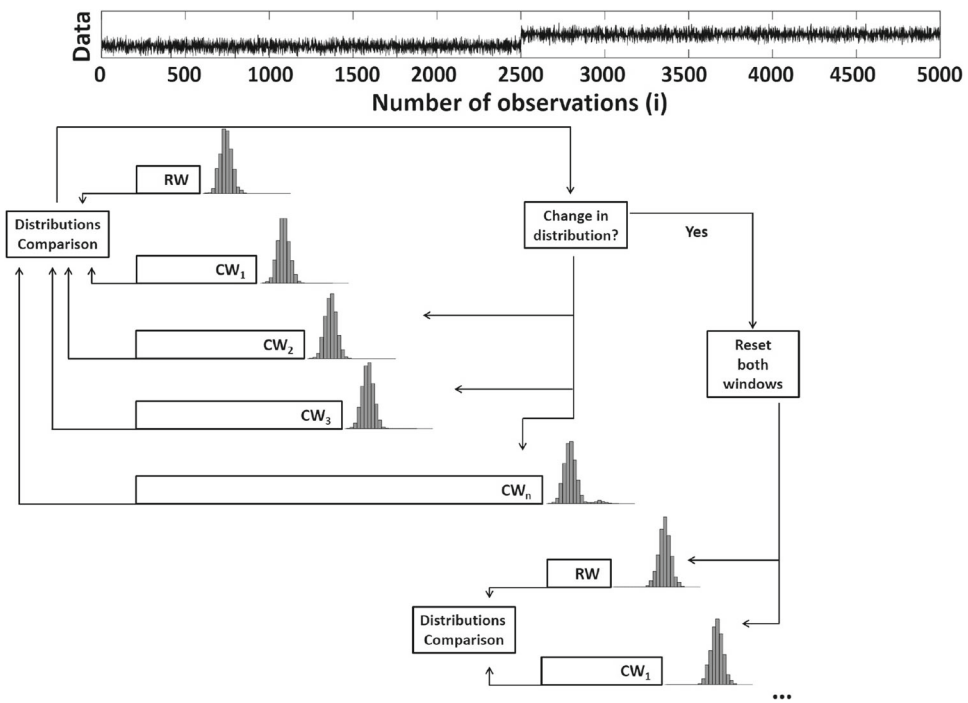

Fig. 4 Workflow of the adaptive cumulative windows model (ACWM) for change detection

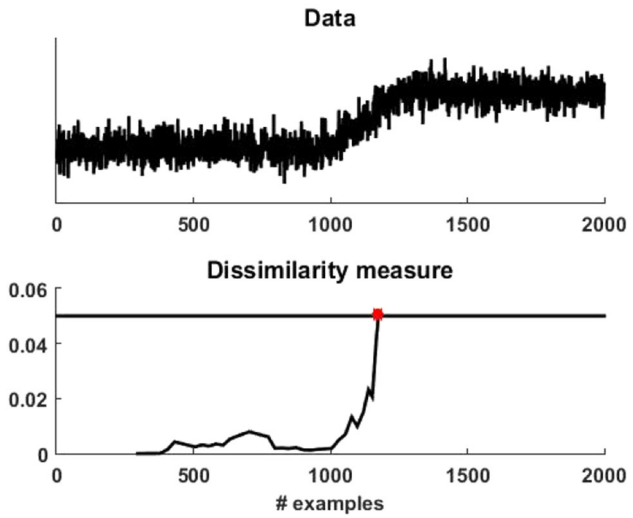

Fig. 5 Behavior of the dissimilarity measure for detecting changes

similar, the absolute difference between $KLD(P_{RW}||P_{CW})$ and $KLD(P_{CW}||P_{RW})$ is small. In the ACWM, the decision rule used to assess changes in data distribution is a dissimilarity measure based on the asymmetry of the Kullback–Leibler divergence. It is defined that a change has occurred in the data distribution of the current window relatively to the data distribution of the reference window, if:

$$|KLD(P_{RW}||P_{CW}) - KLD(P_{CW}||P_{RW})| > \delta,$$

where δ is a user defined threshold, empirically defined and that establishes a trade-off between the false alarms and the missed detections. Increasing δ will entail fewer false alarms, but might miss or delay some changes.

If a change in the distribution of the data in the CW with respect to the distribution of the data in the RW is not detected, the CW is updated with more data observations. Otherwise, if a change is detected, the data in both windows are cleaned, new data are used to fulfill both windows and a new comparison starts.

Figure 5 presents the dissimilarity measure against the detection threshold to detect a change. As desired, it can be observed that this dissimilarity highly increases in the presence of a change.

5.3 Evaluation step for data distributions comparison

In the ACWM, the evaluation step is the increment of the cumulative current window. When comparing data distributions over sliding windows, at each evaluation step the change detection method is induced by the examples that are included in the sliding window. Here, the key difficulty is how to select the appropriate evaluation step. A small evaluation step may ensure fast adaptability in phases where the data distribution changes. However, a small evaluation step implies that more data comparisons are made. Therefore, it tends to be computationally costly, which can affect the overall performance of the change detection method. On the other hand, with a large evaluation step, the number of data distribution comparisons decreases, increasing the performance of the change detection method in stable phases but not allowing quick reactions when a change in the distribution occurs.

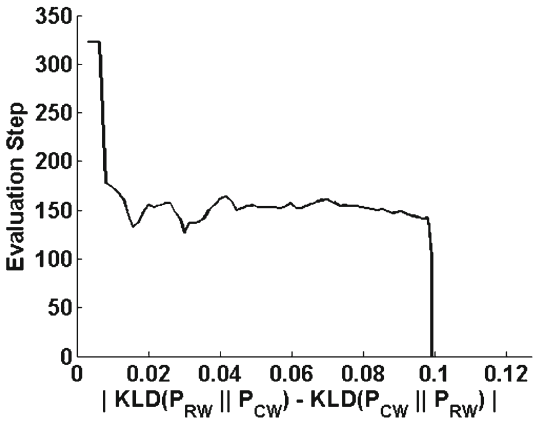

Fig. 6 Representation of the evaluation step for data distributions comparison with respect to the absolute difference between $KLD(P_{RW}||P_{CW})$ and $KLD(P_{CW}||P_{RW})$ (for a change detection threshold of $\delta = 0.1$)

Therefore, the a priori definition of the evaluation step to perform data distribution comparisons is a compromise between computational costs and detection delay time. In the proposed approach, the evaluation step, instead of being fixed and selected by the user, is automatically defined according to the data stationarity and to the distance between data distributions. Starting with an initial evaluation step of $IniEvalStep$, the step length is increased if the distance between distributions is small (which suggests that data are generated according to a similar distribution, hence it is a stationary phase) and is decreased if the distance between distributions is high (which means that data from both windows are further apart), according to the following relation:

$$EvalStep = max\left(1, round\left(IniEvalStep * \left(1 - \frac{1}{\delta}\right) * |KLD(P_{RW}||P_{CW}) - KLD(P_{CW}||P_{RW})|\right)\right).$$

Figure 6 illustrates the dependency of the evaluation step on the distance between data distributions of an artificial data set (for a change detection threshold $\delta = 0.1$).

5.4 Pseudocode

The presented adaptive cumulative windows model (ACWM) was designed to detect changes online in the distribution of streams of data. The data distribution is computed by the histograms presented in Sect. 2. In order to detect changes, the data distributions in two time-windows are compared using the Kullback–Leibler divergence. Algorithm 1 presents the pseudocode for this ACWM.

Algorithm 1 ACWM

Input: Data set: $x_1, x_2, \ldots$
Number of buckets in the histogram $nBuckets$
Length of the Reference window: L_{RW}
Initial evaluation step $IniEvalStep$
Change detection threshold δ
Output: Time of the detected changes: t^*
$t_i \leftarrow 0$
$Init \leftarrow True$
while not at the end of the stream **do**
  if $init = True$ **then**
    Initialize the histogram in the reference window (P_{RW}) as empty
    Initialize the histogram in the current window (P_{CW})as empty
    Define the first evaluation point: $EvalPoint = L_{RW} + IniEvalStep$
    $Init \leftarrow False$
  end if
  if $t \leq t_i + L_{RW}$ **then**
    $Init \leftarrow False$
    Compute the histogram in the RW: P_{RW}
    Compute the histogram counts for the CW
  else if $t = EvalPoint$ **then**
    Compute the histogram in the CW: P_{CW}
    Compute the next evaluation step:
    $EvalStep = max(1, round(IniEvalStep(1 - \frac{1}{\delta}) * |KLD(P_{RW}||P_{CW}) - KLD(P_{CW}||P_{RW})|))$
    Compute the next evaluation point:
    $EvalPoint = EvalPoint + EvalStep$
    if $|KLD(P_{RW}||P_{CW}) - KLD(P_{CW}||P_{RW})| > \delta$ **then**
      $t_i \leftarrow i$
      report a change at time t: $t^* = t$
      $Init \leftarrow True$
    end if
  else
    Compute the histogram counts for the CW
  end if
end while

5.5 Complexity analysis

The complexity of the proposed CWM is linear with the number of observations (n). To show that the CWM is $O(n)$, experiments were performed using artificial data with and without changes. The data set without change was generated according to a normal distribution with zero mean and standard deviation equals to 1. The data set with changes (which was forced at the middle of the data set) was generated from a normal distribution with standard deviation equals to 1 and with zero mean in the first part and with mean equals to 10 in the second part. For both cases, with and without changes, the size was increased from 1.000 to 10.000.000 examples, and 10 different data streams were generated with different seeds. Table 1 shows that the execution time increases linearly with

Table 1 Execution time of the FCWM and ACWM when detecting changes in artificial data with different sizes, with and without changes (average and standard deviation of 10 runs)

Data size	Execution time (s)			
	Fixed step		Adaptive step	
	No change	Change	No change	Change
1000	0.04 ± 0.04	0.01 ± 0	0.02 ± 0.01	0.01 ± 0
10,000	0.23 ± 0.01	0.10 ± 0	0.25 ± 0.01	0.11 ± 0
100,000	2.30 ± 0.04	1.02 ± 0.01	2.35 ± 0.11	1.06 ± 0.04
1,000,000	22.70 ± 0.24	10.19 ± 0.04	22.89 ± 0.94	10.31 ± 0.22
10,000,000	215.51 ± 4.49	102.29 ± 0.94	241.55 ± 43.32	104.19 ± 2.55

the size of the data (average and standard deviation of 10 runs on data generated with different seeds), either for the cases with or without changes and using a fixed or an adaptive evaluation step. Moreover, it can be observed that the execution time when using fixed or adaptive evaluation step is similar.

5.6 Multidimensional setting

The proposed approach for detecting changes in multidimensional data assumes the independence between the features. For multidimensional data sets, the ACWM is computed as follows:

- For each dimension:
  - computes the probabilities $P_{RW}(i)$ and $P_{CW}(i)$ in the reference and in the current sliding windows, respectively;
  - computes

$$absKLD_d = |KLD(P_{RW}||P_{CW}) - KLD(P_{CW}||P_{RW})|;$$

- Computes the mean of the $absKLD_d$, considering all the dimensions:

$$M\_absKLD = \frac{\sum_{d=1}^{D} absKLD_d}{D},$$

  where D is the number of dimensions.
- Signals a change if $M\_absKLD > \delta$, where δ is the change detection threshold.

6 Results on distribution change detection

This section presents the performance of the cumulative windows model (CWM) in detecting distribution changes, using a fixed and an adaptive evaluation step. To detect distribution changes, the model is evaluated using artificial data, presenting distribution changes with different magnitudes and rates, and using real-world data from an industrial process and a medical field. The efficiency of ACWM is also compared with the PHT when detecting distribution changes on a real data set.

The artificial data were obtained in MATLAB [29]. All the experiments were implemented in MATLAB, as well as the graphics produced.

6.1 Experiments with artificial data

The data sets and the experimental designs were outlined in order to evaluate the overall performance of the ACWM in detecting distribution changes in different evolving scenarios, namely to:

1. Evaluate the advantage of using an adaptive evaluation step instead of a fixed one.
2. Evaluate the benefit, in detection delay time, of using fading histograms when comparing data distributions to detect changes.
3. Evaluate robustness to detect changes against different amounts of noise.
4. Evaluate the stability in static phases with different lengths and how it affects the ability to detect changes.

The data sets were generated according to a normal distribution with certain parameters. Both the mean and the standard deviation parameters were varied, generating 2 different problems according to the source of the change. Each data stream consists of 2 parts, where the size of the second is N.

Two data sets were generated. In the first data set, the length of the first part of data streams was set to N. In the second data set, the length of the first part was set to $1N$, $2N$, $3N$ $4N$ and $5N$, in order to simulate different lengths of static phases.

The first data set was used to carry out the first, the second and the third experimental designs, and the second data set was used to perform the fourth experimental design,

Table 2 Magnitude levels of the designed data sets

Parameter changed	Value of the fixed parameter	Parameter variation (before → after change)	Magnitude of change
μ	$\sigma = 1$	$\mu = 0 \rightarrow \mu = 5$	High
		$\mu = 0 \rightarrow \mu = 3$	Medium
		$\mu = 0 \rightarrow \mu = 2$	Low
σ	$\mu = 0$	$\sigma = 1 \rightarrow \sigma = 5$	High
		$\sigma = 1 \rightarrow \sigma = 3$	Medium
		$\sigma = 1 \rightarrow \sigma = 2$	Low

evaluating the effect of different extensions of the stationary phase on the performance of the ACWM in detecting changes.

Within each part of the data streams, the parameters stay the same, which means that only 1 change happens between both parts and different changes were simulated by varying among 3 levels of magnitude (or severity) and 3 rates (or speed) of change, obtaining a total of 9 types of changes for each changing source (therefore, a total of 18 data streams with different kind of changes). Although there is no golden rule to classify the magnitude levels, they were defined in relation to one another, as high, medium and low according to the variation of the distribution parameter, as shown in Table 2. For each type of changes, 30 different data streams were generated with different seeds.

The rates of change were defined assuming that the examples from the first part of data streams are from the old distribution and the $N - ChangeLength$ last examples are from the new distribution, where $ChangeLength$ is the number of examples required before the change is complete. During the length of the change, the examples from the new distribution are generated with probability $p_{new}(t) = \frac{t-N}{ChangeLength}$ and the examples from the old distribution are generated with probability $p_{old}(t) = 1 - p_{new}(t)$, where $N < t < N + ChangeLength$. As for the magnitude levels, the rates of change were defined in relation to one another, as sudden, medium and low, for a $ChangeLength$ of 1, $0.25N$ and $0.5N$, respectively. The value of N was set to 1000.

Therefore, the first data set is composed by a total of 540 data streams with 2000 examples each, and the second data set consists of a total of 2700 data streams with five different lengths, 540 data streams of each length.

Setting the parameters of the CWM and of the online histograms

The CWM and the online histograms require the setting of the following parameters:

- L_{RW} Length of the reference window (CWM);
- $IniEvalStep$ Initial evaluation step (CWM);
- δ Change detection threshold (CWM);
- ε Admissible mean square error of the histogram;

As stated before, the number of buckets is chosen under error constraints [37] and is computed as $k = \frac{R}{2\sqrt{\varepsilon}}$, where R is the admissible range of the variable and ε is the admissible mean square error of the histogram. It was established that 5% was an admissible mean square error of the histogram. To investigate the values for the remaining parameters, an experiment was performed on a training data set with the same characteristics as the first data set, varying the L_{RW} within $1k, 2k, \ldots 10k$ (where k is the number of buckets in the online histograms) and δ within 0.01, 0.05, 0.1, 0.2. However, in this training data set, only 10 data streams were generated with different seeds for each type of drift, obtaining a total of 118 data streams with length $2N$ ($N = 1000$). In this experiment, the CWM was performed with a unitary evaluation step and the summary results were analyzed. Table 3 presents the precision, recall and F_1 score for a reference window of length $50k$ and $10k$ and for a change detection threshold of 0.05 and 0.1, for a total of 180 true changes. Although compromising the delay time in change detection, the best F_1 score is obtained for a reference window of $10k$ examples and a change detection threshold of 0.05.

Figure 7 shows the detection delay time (average of 10 runs) and the total number of false alarms (FA) and missed detections (MD), for a total of 180 true changes, depending on the length of reference window of length (L_{RW}) and on the change detection threshold (δ). It can be observed that an increase in the detection delay time, controlled by the value of δ, is followed by a decrease in the number of false alarms (and an increase in the number of missed detections). However, for $L_{RW} = 10k$ and $\delta = 0.05$, the false alarm rate (3/180) and the miss detection rate (5/180) are admissible.

From now forward, unless otherwise stated, the settings for the parameters of the CWM were the following:

Table 3 Precision, recall and F_1 score, obtained when performing the CWM, with a reference window of length $5k$ and $10k$ and a change detection threshold of 0.05 and 0.1

L_{RW}	
$5k$	$10k$
δ	
0.05	
$Precision = 0.87$	$Precision = 0.98$
$Recall = 1$	$Recall = 0.97$
$F_1 = 0.93$	$F_1 = 0.98$
0.1	
$Precision = 0.97$	$Precision = 1$
$Recall = 0.96$	$Recall = 0.88$
$F_1 = 0.97$	$F_1 = 0.93$

– The length of the reference window was set to $10k$ ($L_{RW} = 10k$);
– The initial evaluation interval was set to k ($IniEval\ Step = k$);
– The threshold for detecting changes was set to 5% ($\delta = 0.05$);

Evaluate the advantage of using an adaptive evaluation step instead of a fixed one

This experiment was designed to study the advantage of performing the CWM with an adaptive evaluation step (ACWM) against a fixed evaluation step (FCWM). Figure 8 shows the advantage, in detection delay time, of an adaptive evaluation step over the fixed one (average results for 30 runs on data generated with different seeds). Except for the distribution change in the mean parameter, with low magnitude and sudden rate, the detection delay time is shorter when performing the ACWM. This decrease in the detection delay time, when performing ACWM, is obtained without compromising the false alarm and miss detection rates (except for one case: change with low magnitude and sudden rate, in the mean parameter, see Table 4).

Using the results of the 30 replicas of the data, a paired, two-sided Wilcoxon signed-rank test was performed to assess the statistical significance of the comparison results. It was tested the null hypothesis that the difference between the detection delay times of the ACWM and the FCWM comes from a continuous, symmetric distribution with zero median, against the alternative that the distribution does not have zero median. For all types of changes in both mean and standard deviation parameters, the null hypothesis of zero median in the differences was rejected, at a significance level of 1%. Therefore, considering the very low p values obtained, there is strong statistical evidence that the detection delay time of ACWM is smaller than of FCWM.

In Table 4, besides the detection delay time using the CWM with an adaptive and a fixed evaluation steps, the total number of missed detections and the total number of false

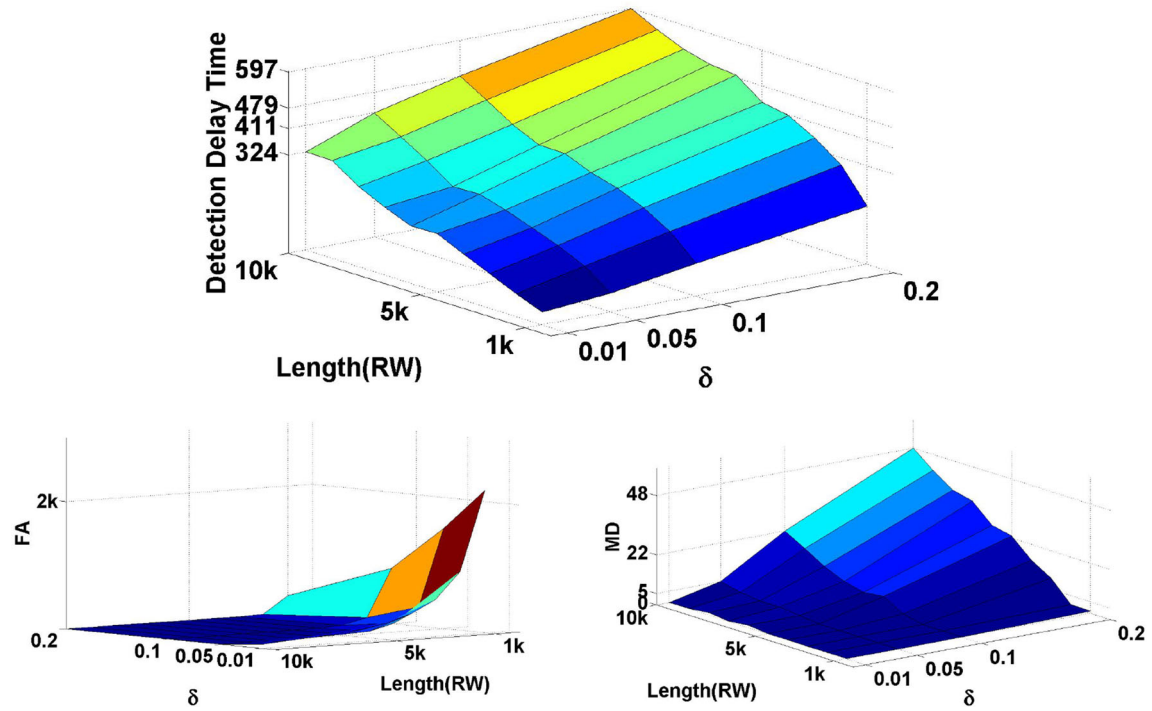

Fig. 7 Detection delay time, total number of false alarms (FA) and missed detections (MD), depending on the L_{RW} and δ

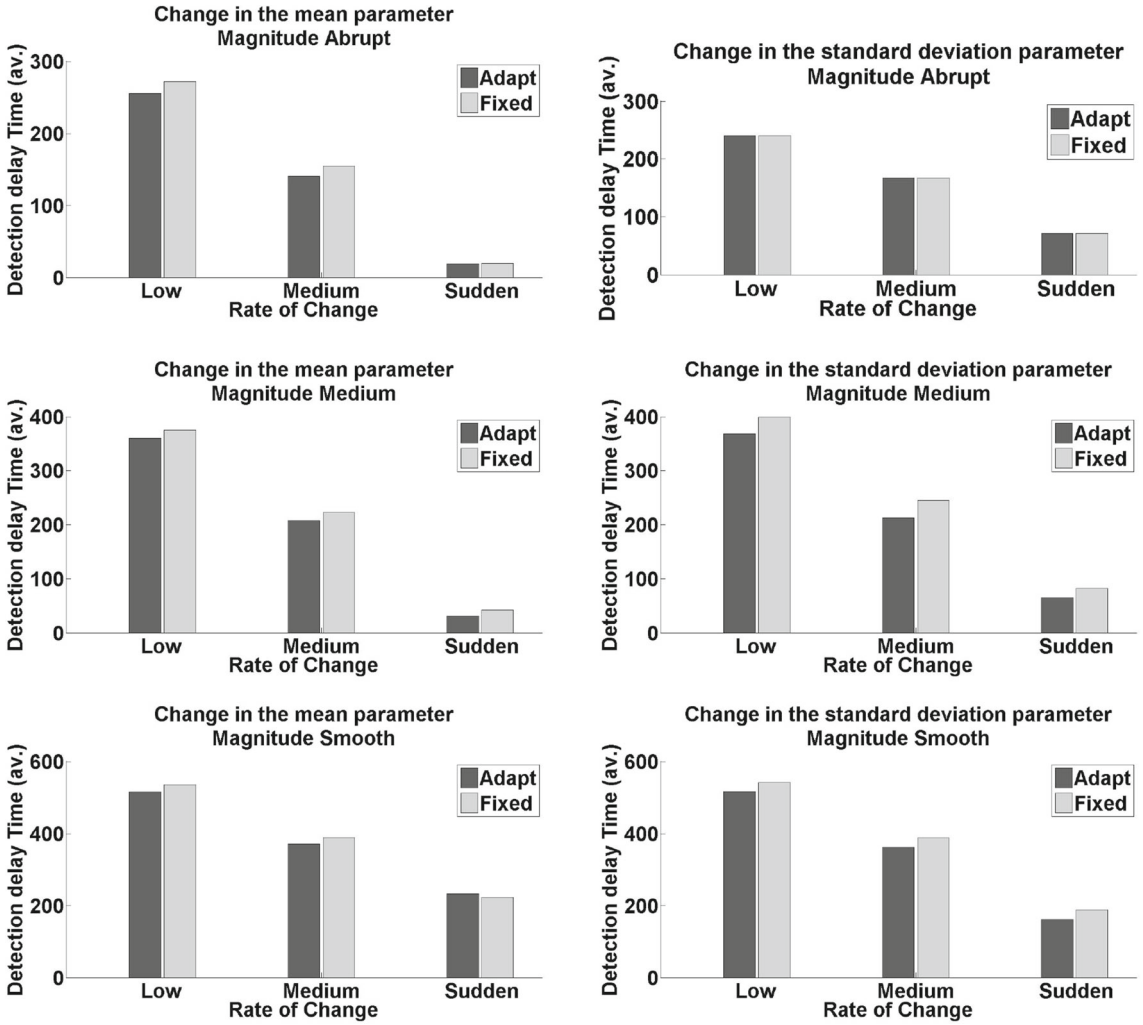

Fig. 8 Detection delay time (average of 30 runs) using the CWM with an adaptive (ACWM) and a fixed (FCWM) evaluation steps

alarms are also presented. The results report the average and standard deviation of 30 runs.

As expected, greater distribution changes (high magnitudes and sudden rates) are easier to detect by the CWM, either using an adaptive or a fixed evaluation step. On the other hand, for smaller distribution changes (low magnitudes and low rates) the detection delay time increases. The decrease in the detection delay time in this experiment sustains the use of an adaptive evaluation step, when performing the CWM. Although the decrease in detection delay time is small, these results must be taken into account that the length of the data was also small. With data with higher length, the decrease in detection delay time will be reinforced. Moreover, for both strategies, the execution time of performing the CWM is comparable.

Evaluate the advantage, in detection delay time, of using fading histograms when comparing data distributions to detect changes

As stated before, fading histograms attribute more weight to recent data. In an evolving scenario, this could be a huge advantage since it enhances small changes. Therefore, when comparing data distributions to detect changes, the detection of such changes will be easier. This experimental design intends to evaluate the advantage of using fading histograms as a synopsis structure to represent the data distributions that will be compared, for detecting changes, within the ACWM (which will be referred to as ACWM-fh). Thus, data distributions, within the reference and the current windows, were computed using fading histograms with different values of

Table 4 Detection delay time (DDT) using the ACWM and the FCWM

Parameter changed	Mag.	Rate	Adaptive step DDT ($\mu \pm \sigma$)	Fixed step DDT ($\mu \pm \sigma$)
Mean	High	Low	260 ± 57	275 ± 53 (0;0)
		Medium	153 ± 24 (1;1)	178 ± 59 (0;1)
		Sudden	19 ± 4 (0;1)	24 ± 0 (0;1)
	Medium	Low	410 ± 131 (0;1)	424 ± 138 (0;1)
		Medium	242 ± 125 (0;1)	259 ± 122 (0;1)
		Sudden	36 ± 22 (0;1)	52 ± 26 (0;1)
	Low	Low	516 ± 171 (7;2)	535 ± 177 (7;2)
		Medium	371 ± 233 (5;0)	389 ± 232 (5;0)
		Sudden	233 ± 229 (1;0)	223 ± 193 (3;0)
STD	High	Low	240 ± 34	284 ± 42
		Medium	168 ± 16	198 ± 30
		Sudden	71 ± 10	104 ± 0
	Medium	Low	368 ± 87	399 ± 93
		Medium	213 ± 28	245 ± 32
		Sudden	65 ± 15	83 ± 27
	Low	Low	517 ± 158	542 ± 154
		Medium	362 ± 127	387 ± 129
		Sudden	162 ± 60 (1;0)	189 ± 63 (1;0)

The results report the average and standard deviation of 30 runs. In parenthesis is the number of runs, if any, where the algorithm misses detection or signals a false alarm: They are in the form (Miss; False Alarm)

fading factors: 1 (no forgetting at all), 0.9994, 0.9993, 0.999, 0.9985 and 0.997.

Using the results of the 30 replicas of the data, a paired, two-sided Wilcoxon signed-rank test (with Bonferroni correction for multiple comparisons) was performed to assess the statistical significance of differences between ACWM and ACWM-fh. With the exception of the change in the mean parameter with high magnitude and sudden rate (for the fading factors tested except 0.997), for all the other types of changes in both mean and standard deviation parameters, the null hypothesis of zero median in the differences between detection delay times was rejected, at a significance level of 1%. Therefore, considering the very low p values obtained, there is strong statistical evidence that the detection delay time of ACWM-fh is smaller than of ACWM.

Table 5 presents a summary of the detection delay time (average and standard deviation from 30 runs on data generated with different seeds) using the ACWM-fh for comparing the data distributions. The total number of missed detections and the total number of false alarms are also presented. This experiment underlines the advantage of using fading histograms to compute the data distributions: The detection delay time decreases by decreasing the fading factor and without compromising the number of missed detections and false alarms (except when using a fading factor of 0.997). The increase in false alarms when using a fading factor of 0.997 suggests that fading histograms computed with this value are over reactive; therefore, fading factors of values equal or smaller than 0.997 are not suitable for use in this data set.

The detection delay time (average of 30 runs on data generated with different seeds) of this experimental design is shown in Fig. 9. It can be observed that the advantage of using fading histograms is strengthened when detecting small changes, which is explained by the greater importance attributed to recent examples that enhances a change and eases its detection by the ACWM.

Evaluate the robustness to detect changes against different amounts of noise

Within this experimental design, the robustness of the ACWM against noise was evaluated. Noisy data were generated by adding different percentages of Gaussian noise with zero mean and unit variance to the original data set. Figure 10 shows the obtained results by varying the amount of noise from 10% to 50%.

The detection delay time (average of 30 runs on data generated with different seeds) of this experimental design is shown in Fig. 10. The ACWM presents a similar performance along the different amounts of noise, with the exception of a change in the standard deviation parameter with high magnitude and medium and sudden rates (for a level of noise of 30%). In these cases, the average detection delay time

Table 5 Detection delay time (average and standard deviation), using the ACWM and computing data distributions with fading histograms (with different fading factors)

Parameter changed	Mag.	Rate	Fading factor					
			1	0.9994	0.9993	0.999	0.9985	0.997
Mean	High	Low	260 ± 57	246 ± 64	246 ± 64	241 ± 68	233 ± 77	226 ± 70 (0;5)
		Medium	153 ± 24 (1;1)	145 ± 27 (0;1)	150 ± 33 (0;2)	140 ± 31 (0;2)	140 ± 32 (0;2)	125 ± 36 (0;2)
		Sudden	19 ± 4 (0;1)	19 ± 5 (0;1)	19 ± 5 (0;1)	18 ± 6 (0;1)	16 ± 6 (0;1)	13 ± 6 (0;1)
	Medium	Low	410 ± 131 (0;1)	387 ± 142 (0;1)	385 ± 144 (0;1)	381 ± 151 (0;1)	365 ± 148 (0;2)	311 ± 96 (0;5)
		Medium	242 ± 125 (0;1)	215 ± 69 (0;2)	213 ± 69 (0;2)	211 ± 58 (0;3)	205 ± 58 (0;3)	186 ± 54 (0;5)
		Sudden	36 ± 22 (0;1)	27 ± 16 (0;1)	25 ± 15 (0;1)	23 ± 11 (0;1)	22 ± 9 (0;1)	36 ± 110 (0;2)
	Low	Low	516 ± 171 (7;2)	510 ± 202 (4;2)	496 ± 204 (4;2)	496 ± 197 (3;2)	448 ± 173 (2;2)	369 ± 150 (0;9)
		Medium	371 ± 233 (5;0)	338 ± 208 (3;0)	327 ± 193 (3;0)	324 ± 209 (1;0)	289 ± 168	250 ± 151 (0;4)
		Sudden	233 ± 229 (1;0)	165 ± 170 (1;0)	159 ± 168 (1;0)	139 ± 159 (1;0)	138 ± 180	66 ± 76 (0;1)
STD	High	Low	240 ± 34	219 ± 36	216 ± 38	208 ± 41	204 ± 45	186 ± 52
		Medium	168 ± 16	157 ± 18	155 ± 19	151 ± 22	143 ± 25	138 ± 40
		Sudden	71 ± 10	60 ± 13	58 ± 14	52 ± 16	42 ± 19	23 ± 18
	Medium	Low	368 ± 87	327 ± 76	322 ± 76	310 ± 76	294 ± 79	249 ± 92
		Medium	213 ± 28	185 ± 30	180 ± 30	170 ± 32	159 ± 35	140 ± 40
		Sudden	65 ± 15	53 ± 13	52 ± 14	49 ± 13	43 ± 13	39 ± 14
	Low	Low	517 ± 158	445 ± 140	435 ± 137	411 ± 136	380 ± 145	316 ± 113 (0;2)
		Medium	362 ± 127	305 ± 101	295 ± 92	278 ± 88	260 ± 85	204 ± 52
		Sudden	162 ± 60 (1;0)	116 ± 41 (1;0)	122 ± 74	109 ± 74	87 ± 46	62 ± 40

The results report the average and standard deviation of 30 runs. In parenthesis is the number of runs, if any, where the ACWM-fh misses detection or signals a false alarm: They are in the form (Miss; False Alarm)

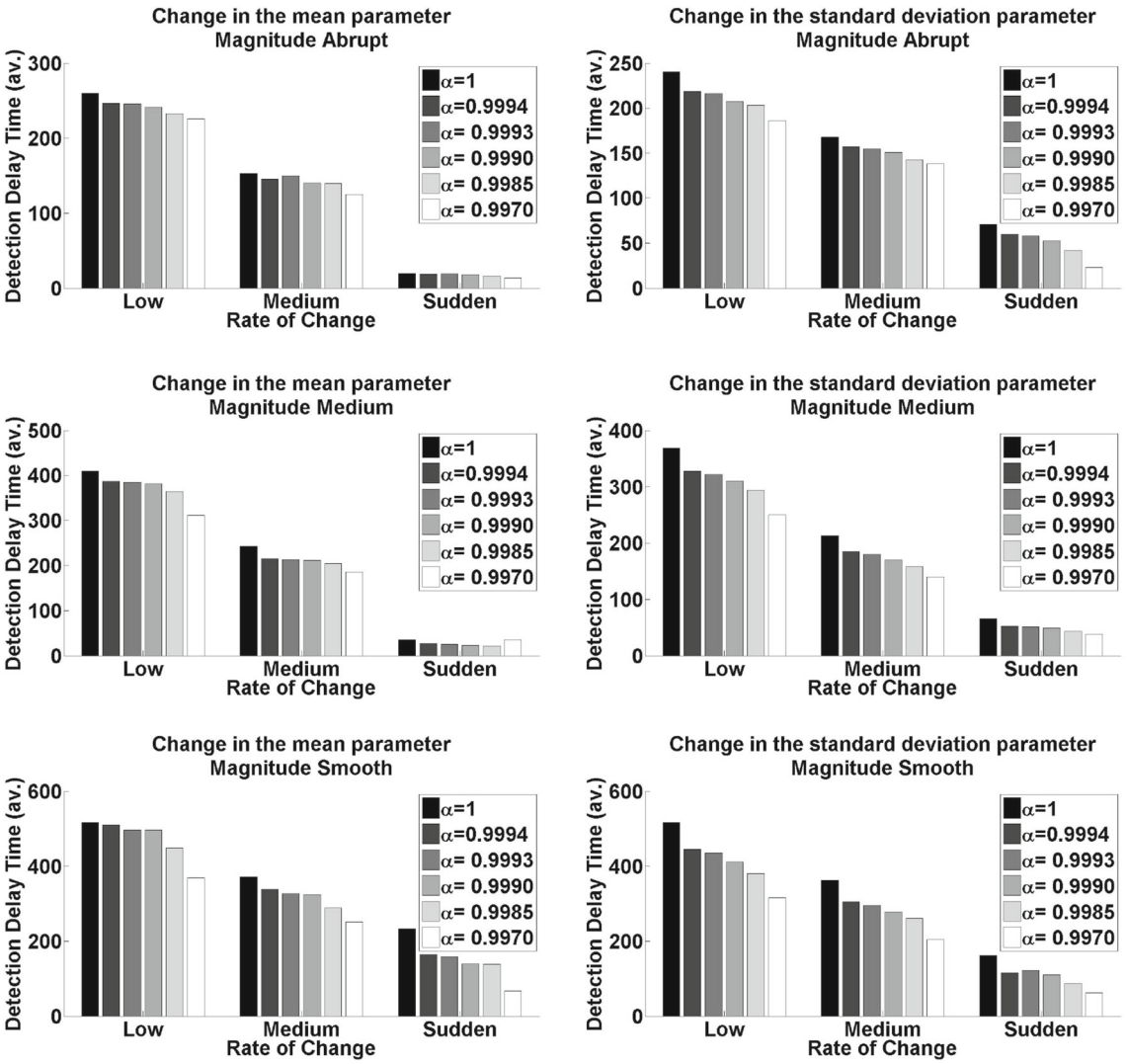

Fig. 9 Detection delay time (average of 30 runs) of the ACWM-fh

increases when compared with other amounts of noise. This experiment sustains the argument that the ACWM is robust against noise while effectively detects distribution changes in the data.

Table 6 presents a summary of the detection delay time (average and standard deviation from 30 runs on data generated with different seeds) using the ACWM for comparing the data distributions in the first data set. The total number of missed detections and the total number of false alarms are also presented. Regarding the total number of missed detections and false alarms, with an amount of 50% of noise a slight increase is noticeable for both, mainly for changes in the mean parameter.

To assess the statistical significance of differences between the detection delay time of the ACWM when performed on data without and with different amounts of noise, a paired, two-sided Wilcoxon signed-rank test (with Bonferroni correction for multiple comparisons) was performed. For most of the cases, at a significance level of 1%, there are no statistical evidence to reject the null hypothesis of zero median in the differences (exceptions are indicated in Table 6 with **). This experiment sustains the argument that the ACWM is robust against noise, while effectively detects distribution changes in the data.

Evaluate the stability in static phases with different lengths and how it affects the ability to detect changes

This experiment was carried out with the second data set. The performance of the ACWM was evaluated varying the length of stationary phases from $1N$ to $5N$ ($N = 1000$).

Overall, it can be observed that the detection delay time for the ACWM increases within the increase in the stationary phase. This is even more evident in distribution changes with

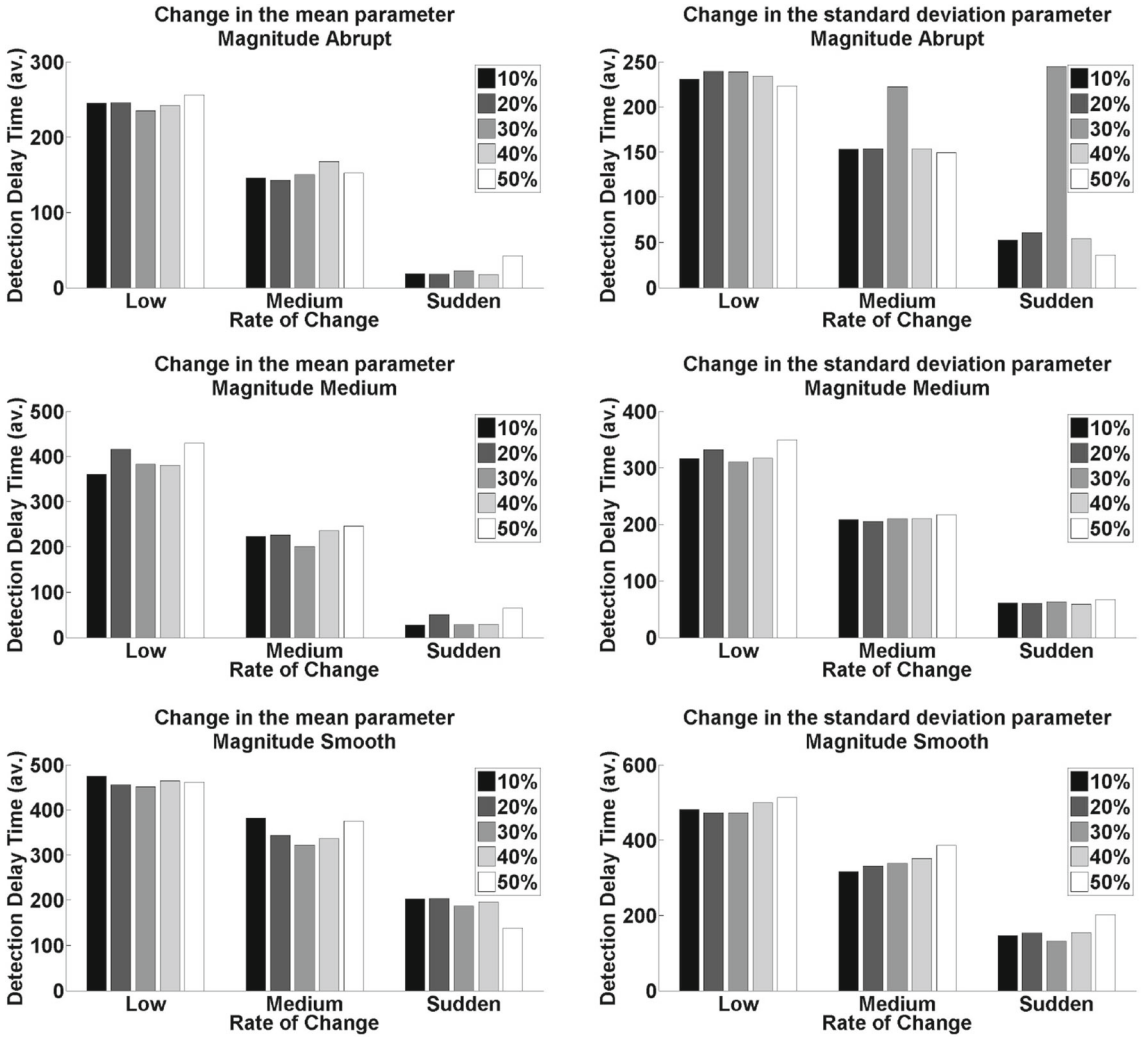

Fig. 10 Detection delay time (average of 30 runs) of the ACWM with different amounts of noise

sudden rates. Indeed, the stability of the ACWM in stationary phases compromises the ability to effectively detect changes. However, this can be overthrown by using fading histograms to compute the data distributions, as shown in Fig. 11.

Actually, in stationary phases, the ability of the fading histograms to forget outdated data works in favor of the change detection model, by decreasing the detection delay time. However, a decrease in the value of the fading factor results in the increase in the number of false alarms. Table 7 presents the detection delay time (average and standard deviation of 30 runs on data generated with different seeds for the 9 types of changes for each source parameter) using the ACWM-fh in different stationary phases. The total number of missed detections and the total number of false alarms are also presented. From the results presented, it can be noted that a decrease in the detection delay time is achieved, establishing a commitment with respect to the number of false alarms and missed detections.

6.2 Experiments with an industrial data set

This industrial data set was obtained within the scope of the work presented in Correa et al. [12], with the objective of designing different machine learning classification methods for predicting surface roughness in high-speed machining. Data were obtained by performing tests in a Kondia HS1000 machining center equipped with a Siemens 840D open-architecture CNC. The blank material used for the tests was 170 × 100 × 25 aluminum samples with hardness ranging from 65 to 152 Brinell, which is a material commonly used in automotive and aeronautical applications. These tests were done with different cutting parameters, using sensors for registry vibration and cutting forces. A multi-component dynamometer with an upper plate was used to measure the in-process cutting forces and piezoelectric accelerometers in the X and Y axis for vibrations measures. Each record includes

Table 6 Detection delay time (average and standard deviation), using the ACWM with different amounts of noise

Parameter changed	Mag.	Rate	Noise scale				
			10%	20%	30%	40%	50%
Mean	High	Low	245 ± 62	246 ± 67	235 ± 44	242 ± 82	256 ± 75 (0;3)
		Medium	146 ± 25 (0;2)	143 ± 29 (0;1)	150 ± 30 (0;1)	167 ± 148 (0;2)	152 ± 36 (0;2)
		Sudden	19 ± 5	18 ± 4	22 ± 6	17 ± 4	42 ± 150 (0;1)
	Medium	Low	360 ± 152 (1;0)	416 ± 123 (1;2)	383 ± 90 (1;1)	380 ± 127 (1;1)	429 ± 144 (1;4)
		Medium	223 ± 77 (0;3)	225 ± 70 (0;1)	201 ± 50	235 ± 72 (0;2)	246 ± 112 (1;5)
		Sudden	27 ± 12 (0;1)	51 ± 138 (0;2)	28 ± 9 (0;1)	29 ± 13 (0;1)	65 ± 120 (0;1)
	Low	Low	475 ± 201 (4;1)	456 ± 165 (6;1)	451 ± 137 (5;0)	464 ± 138 (7;2)	461 ± 144 (6;5)
		Medium	382 ± 237 (6;1)	344 ± 206 (5;1)	322 ± 166 (5;1)	336 ± 237 (4;1)	375 ± 205 (5;1)
		Sudden	203 ± 226 (0;1)	204 ± 234 (3;1)	187 ± 235 (1;0)	197 ± 221 (2;1)	139 ± 203 (3;9)
STD	High	Low	231 ± 35	240 ± 38	239 ± 36	234 ± 38	223 ± 53
		Medium	153 ± 17 **	154 ± 15 **	222 ± 89 (1;0) **	153 ± 17 **	150 ± 27 **
		Sudden	53 ± 14 **	61 ± 14 **	245 ± 163 (2;0) **	54 ± 13 **	36 ± 7 **
	Medium	Low	316 ± 47	332 ± 49	310 ± 63 **	317 ± 64 **	349 ± 79
		Medium	208 ± 24	205 ± 31	210 ± 29	209 ± 33	217 ± 56
		Sudden	61 ± 16	60 ± 18	63 ± 14	59 ± 14	67 ± 17
	Low	Low	481 ± 117	472 ± 129	472 ± 141 (1;0)	499 ± 128 (1;0)	513 ± 181 (2;1)
		Medium	316 ± 142 (1;0)	331 ± 107	338 ± 104	350 ± 134	386 ± 210 (0;1)
		Sudden	146 ± 80	153 ± 69	132 ± 63	154 ± 77	202 ± 152 (1;0)

The results report the average and standard deviation of 30 runs. In parenthesis is the number of runs, if any, where the ACWM misses detection or signals a false alarm: They are in the form (Miss; False Alarm)

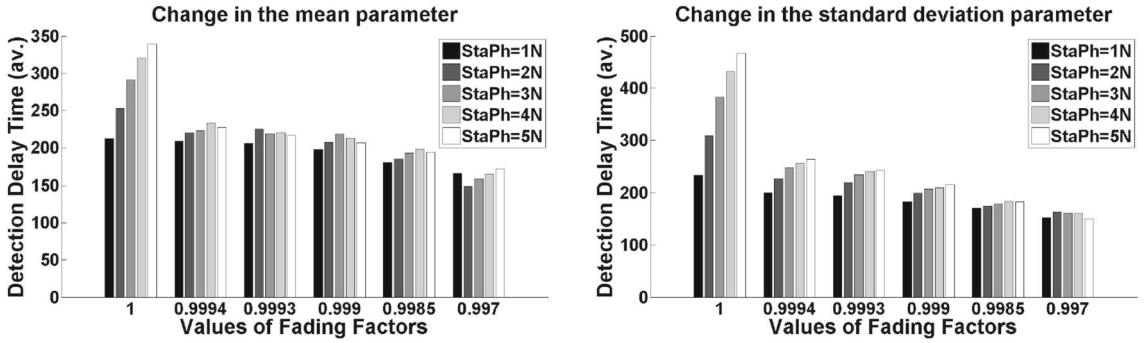

Fig. 11 Detection delay time (average of 30 runs for the 9 types of changes) of the ACWM-fh with different lengths of stationary phases

Table 7 Detection delay time (average and standard deviation) using the ACWM-fh in different stationary phases

Parameter changed	Fading factor	Length of stationary phase				
		$1k$	$2k$	$3k$	$4k$	$5k$
Mean	1	249 ± 166 (14;7)	282 ± 172 (25;11)	300 ± 171 (31;7)	334 ± 183 (31;7)	365 ± 189 (35;7)
	0.9994	228 ± 163 (8;8)	248 ± 163 (12;14)	260 ± 159 (14;15)	258 ± 164 (12;17)	254 ± 159 (15;21)
	0.9993	224 ± 159 (8;9)	246 ± 170 (9;16)	247 ± 151 (14;18)	252 ± 162 (10;21)	250 ± 162 (11;24)
	0.999	219 ± 160 (5;10)	228 ± 161 (9;17)	229 ± 150 (10;23)	227 ± 153 (10;30)	228 ± 155 (11:34)
	0.9985	206 ± 146 (2;11)	221 ± 157 (4;20)	219 ± 162 (3;33)	228 ± 164 (1;47)	223 ± 163 (4;60)
	0.997	176 ± 125 (0;34)	188 ± 121 (0;79)	182 ± 127 (3;121)	177 ± 131 (0;146)	187 ± 122 (0;182)
Standard deviation	1	241 ± 150 (1;0)	317 ± 185 (4;0)	385 ± 203 (11;0)	441 ± 208 (23;0)	495 ± 220 (34;0)
	0.9994	207 ± 131 (1;0)	235 ± 141 (1;0)	251 ± 148 (1;0)	257 ± 151 (1;0)	263 ± 152 (1;0)
	0.9993	204 ± 127	227 ± 137 (1;0)	238 ± 142 (1;0)	243 ± 143 (1;0)	246 ± 144 (1;0)
	0.999	193 ± 122	208 ± 128 (1;0)	212 ± 130 (1;0)	213 ± 129 (1;1)	214 ± 131 (1;1)
	0.9985	179 ± 116	188 ± 118	186 ± 117 (1;1)	189 ± 124 (2;6)	187 ± 118 (0;9)
	0.997	151 ± 99 (0;2)	155 ± 96 (1;10)	160 ± 97 (1;21)	162 ± 105 (4;42)	162 ± 103 (3;60)

The results report the average and standard deviation of 30 runs for the 9 types of changes for each source parameter. In parenthesis is the number of runs, if any, where the ACWM-fh misses detection or signals a false alarm: They are in the form (Miss; False Alarm)

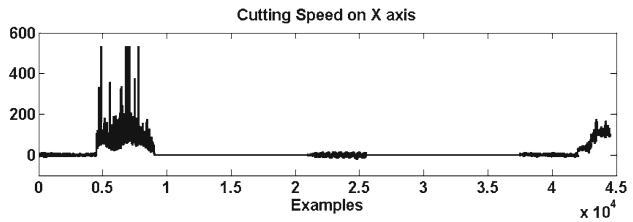

Fig. 12 The cutting speed on X axes from 7 tests sequentially joined

information on several variables used in a cutting process, and the measurements for each test were saved individually.

For change detection purposes, the measurements of the cutting speed on X axes from 7 tests were joined sequentially in order to have only one data set with 6 changes with different magnitudes and sudden and low rates. Figure 12 shows this data set.

The goal of this experiment is to evaluate the feasibility of the proposed ACWM with an industrial problem and comparing the advantage of using fading histograms. To this end, data distributions, within the reference and the current windows, were computed using fading histograms with different values of fading factors: 1 (no forgetting at all), 0.999994 and 0.99999. Furthermore, the ability for detecting changes in data distribution of the ACWM-fh was also compared with the Page–Hinkley Test (PHT) [34].

The PHT is a sequential analysis technique typically used for monitoring change detection in the average of a Gaussian signal [32]. The two-sided PHT tests increases and decreases in the mean of a sequence. For testing online changes, it runs two tests in parallel, considering a cumulative variable U_T defined as the accumulated difference between the observed values and their mean until the current moment. The tests performed are the following:

At every observation, the two PH statistics (PH_U and PH_L) are monitored and a change is reported whenever one of them rises above a given threshold λ. The threshold λ depends on the admissible false alarm rate. A higher λ will guarantee few false alarms, but it may lead to missed detections or delay them.

For increase cases:	For decrease cases:
$U_0 = 0$	$L_0 = 0$
$U_T = (U_{T-1} + x_T - \bar{x}_T - \delta)$	$L_T = (L_{T-1} + x_T - \bar{x}_T + \delta)$
($\bar{x}_T$ is the mean of the signal until the current example.)	
$m_T = min(U_t, t = 1 \ldots T)$	$M_T = max(L_t, t = 1 \ldots T)$
$PH_U = U_T - m_T$	$PH_L = M_T - L_T$

To adjust the PHT input parameters, an analysis was previously conducted on collected data with similar characteristics. From that the parameters δ and λ were set equal to 1 and 1000, respectively.

The ACWM-fh is able to detect the 6 changes in the data with smaller detection delay time than when using histograms constructed over the entire data. Moreover, with both approaches for data representations, the model did not miss any change. Although data have different kinds of changes, either ACWM or ACWM-fh presented a performance which was highly resilient to false alarms. Although detecting all changes, the PHT presented 18 false alarms in this experiment. Moreover, the average detection delay time obtained with the PHT is greater than when performing the ACWM-fh. Concerning the fourth change, all methods require too much examples to detect it. This is reasonable since before and after the fourth change, the average of the data is similar and the change in the standard deviation is also small. Therefore, all methods analyzed several examples before signed a change. Although, it should be noticed that the PHT detected this change with almost half the examples than the others. With respect to the third and the fifth change, the PHT required much more examples than ACWM or ACWM-fh to detect these changes, which are similar: The average of data before and after the change is similar and the standard deviation slightly increases. Therefore, the high delay in detecting these kind of changes is due to the design of the PHT.

Regarding the delay between the occurrence of changes and the detections, the number of false alarms and the missed detections, the ACWM-fh outperforms the ACWM and the PHT.

Table 8 presents the detection delay time of the compared methods when applied to this industrial data set. It can be observed that ACWM presents a high delay time when detecting the fourth change (with low magnitude).

Table 8 Detection delay time of the ACWM, of the ACWM-fh and of the PHT, on the industrial data set

True change	ACWM	ACWM-fh		PHT
		$\alpha = 0.999994$	$\alpha = 0.99999$	
45,000	906	910	959	190
90,000	988	1331	988	270
210,000	2865	2100	1749	7060
255,000	9142	8452	7900	4500
375,000	2496	1806	1493	8890
420,000	1340	1018	1268	500
Average	2956	2603	2393	3568

6.3 Experiments with a medical data set—CTGs

The CWM was evaluated on five fetal cardiotocographic (CTG) problems, collected at the Hospital de São João, in Porto. Fetal cardiotocography is one of the most important methods of assessment of fetal well-being. CTG signals contain information about the fetal heart rate (FHR) and uterine contractions (UC).

Five antepartum FHR with a median duration of 70 min (min–max: 59–103) were obtained and analyzed by the SisPorto® system. These cases corresponded to a median gestational age of 39 weeks and 5 days (min–max: 35 weeks and 4 days–42 weeks and 1 day).

The SisPorto® system, developed at INEB (Instituto Nacional de Engenharia Biomédica), starts the computer processing of CTG features automatically after 11 min of tracing acquisition and updates it every minute [2], providing the FHR baseline estimation, identifying accelerations and decelerations and quantifying short-term and long-term variability according to algorithms described in Ayres-de Campos et al. [1]. Along with these features, the system also triggers alerts, such as "Normality criteria met alert", "Non-reassuring alerts" and "Very non-reassuring alerts" (further details can be founded in Ayres-de Campos et al. [1]). However, the system usually takes about 10 min to detect these different behaviors. In the "Normal" stage of FHR tracing four different patterns may be considered [19]:

- *A* Corresponding to calm or non-eye movement (REM) sleep;
- *B* Active or rapid eye movement (REM) sleep;
- *C* Calm wakefulness;
- *D* Active wakefulness;

Figure 13 shows an example of the analysis of a CTG exam exactly as it is produced by the SisPorto® system. The FHR and UC tracings are represented at the top and at the bottom, respectively. The FHR baseline estimation, accelerations and decelerations and different alerts stages also can be observed in this figure. The "Normal" stage is represented with a green bar in between the FHR and UC tracings. The "Suspicious" stage is represented with yellow and orange bars and the "Problematic" stage with a red bar.

The aim is to apply the FCWM and the ACWM for this clinical data and assess whether the changes detected are in accordance with the changes identified by the SisPorto®

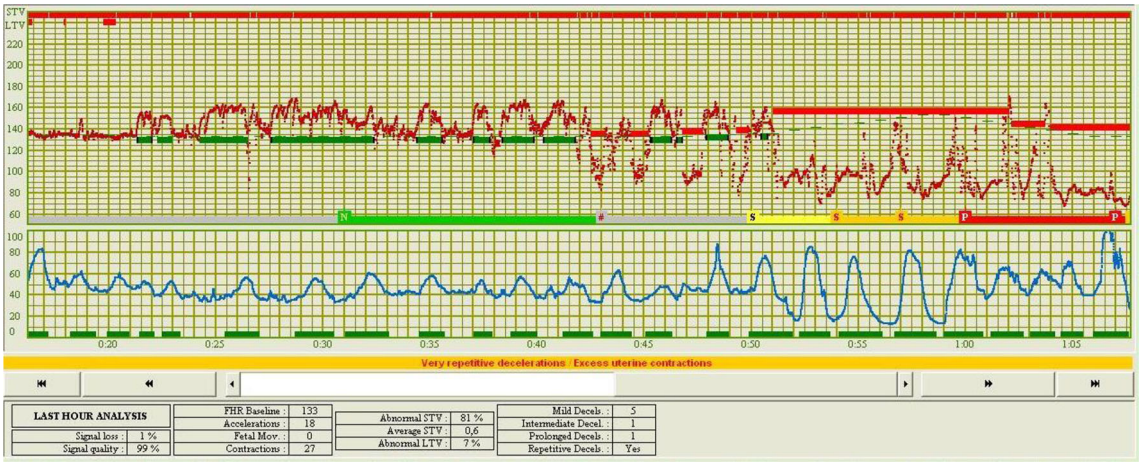

Fig. 13 FHR (*top*) and UC (*bottom*) tracings. This figure also includes the FHR baseline estimation, accelerations and decelerations and patterns classification

system. Ideally, these changes should be detected earlier with CWM. The CWM was applied to the FHR tracings.

The achieved results are consistent with the system analysis and the CWM detects the changes between the different stages earlier than the SisPorto® system. Further than the analysis of this program, the method is able to detect some changes between different patterns of the "Normal" stage. Due to difficulties in ascertaining the exact change points between these behaviors it is not possible to perform a detection delay evaluation. However, the preference of the ACWM is again supported by the detections results in this data set.

7 Results on concept change detection

Concerning the detection of concept changes, a comparison of the ACWM with three well-known methods taken from the literature was undertaken, namely:

- Drift Detection Method (DDM), presented by Gama et al. [15];
- ADaptive WINDdowing (ADWIN) method, introduced by Bifet and Gavaldà [7];
- Page–Hinkley Test (PHT), described in Page [34];

Such comparison was done using artificial data and public data sets. This section ends presenting results on the ability to detect changes of the ACWM under multidimensional settings.

The artificial data were obtained in MATLAB, and all the experiments were implemented in MATLAB, as well as the graphics produced.

Drift Detection Method (DDM)

This online drift detection method monitors the trace of the error rate of an online classifier, for streaming observations, and considers that the error rate follows the binomial distribution. At each time t, the error rate of the online classifier is the probability of misclassifying, p_t, with a standard deviation $s_t = \sqrt{p_t(1-p_t)/t}$. According to the *Probability Approximately Correct* (PAC) learning model, this method assumes that in a stationary concept, the error rate decreases with the number of observations. Therefore, an increase in the error rate indicates a change in the concept. While monitoring the error rate, the DDM stores p_{min} and s_{min}, which correspond to the minimum probability and minimum standard deviation (respectively), and are obtained when $p_t + s_t$ reaches its minimum value. Based on these minimum values, the DDM establishes two levels as follows:

- The warning level: when $p_t + s_t \geq p_{min} + 2s_{min}$;
- The drift level: $p_t + s_t \geq p_{min} + 3s_{min}$;

When the error rate exceeds the lower threshold, the system enters in a warning mode and stores the observations within the warning level in a short-term memory. If the error drops below the threshold again, the warning mode is canceled. However, if the error increases reaching the second (higher) threshold, a change in the concept is assigned. The online classifier is retrained using only the observations in the buffer and reinitializes the variables.

ADaptive WINdowing (ADWIN)

The ADaptive WINDdowing method keeps a sliding window W (with length n) with the most recently received examples and compares the distribution on two sub-windows of W. Whenever two *large enough* sub-windows, W_0 and W_1, exhibit *distinct enough* averages, the older sub-window is dropped and a change in the distribution of examples is assigned. The window cut threshold is computed as follows:

Table 9 Average detection delay time (DDT), number of false alarms (#FA) and the number of missed detections (#MD), for the four methods, using the data streams with lengths 2.000, 5.000 and 10.000 and with different slopes in the Bernoulli parameter distribution

Length	Slope	ADWIN			DDM			PHT			ACWM-fh ($\alpha = 0.9994$)		
		DDT	#FA	#MD	DDT	#FA	#MD	DDT	#FA	#MD	DDT	#FA	#MD
2.000	0	(n.a.)	5	(n.a.)	(n.a.)	0	(n.a.)	(n.a.)	4	(n.a.)	(n.a.)	0	(n.a.)
	1×10^{-4}	582	0	3	627	0	2	573	0	3	629	100	5
	2×10^{-4}	578	0	0	687	0	16	523	0	0	620	0	0
	3×10^{-4}	428	0	0	537	0	0	397	0	0	550	0	0
	4×10^{-4}	359	0	0	534	0	0	331	0	0	430	0	0
5.000	0	(n.a.)	17	(n.a.)	(n.a.)	17	(n.a.)	(n.a.)	41	(n.a.)	na	0	(n.a.)
	1×10^{-4}	722	16	30	866	21	77	650	23	13	849	0	27
	2×10^{-4}	512	13	13	732	19	37	463	25	0	632	0	0
	3×10^{-4}	383	14	14	668	20	17	337	32	0	539	0	0
	4×10^{-4}	320	10	10	587	9	12	279	29	0	273	0	0
10.000	0	(n.a.)	15	(n.a.)	(n.a.)	44	(n.a.)	(n.a.)	68	(n.a.)	(n.a.)	20	(n.a.)
	1×10^{-4}	722	19	35	829	39	94	650	60	10	828	14	54
	2×10^{-4}	505	19	19	843	56	57	466	71	0	678	15	5
	3×10^{-4}	401	17	17	720	29	53	344	68	0	576	16	1
	4×10^{-4}	327	23	23	642	52	41	280	66	0	507	22	6

For $slope = 0$ (no change) the measurements DDT and #MD are not applicable

$\varepsilon_{cut} = \sqrt{\frac{1}{2m} ln \frac{4}{D}}$, with $m = \frac{1}{1/n_0 + 1/n_1}$, where n_0 and n_1 denote the lengths of W_0 and W_1.

A confidence value D is used within the algorithm, which establishes a bound on the false positive rate. However, as this first version was computationally expensive, the authors propose to use a data structure (a variation of exponential histograms), in which the information on the number of 1's is kept as a series of buckets (in the Boolean case). It keeps at most M buckets of each size 2^i, where M is a user defined parameter. For each bucket, two (integer) elements are recorded: *capacity* and *content* (*size* or the number of 1s it contains).

Page Hinkley Test (PHT)

To detect increases, the Page–Hinkley Test (PHT) computes the minimum value of cumulative variable: $m_T = min(U_t, t = 1 \ldots T)$ and monitors the difference between U_T and m_T: $PH_T = U_T - m_T$. When the difference PH_T is greater than a given threshold (λ) a change in the distribution is assigned. Controlling this detection threshold parameter makes it possible to establish a trade-off between the false alarms and the missed detections.

7.1 Experiments with artificial data

To assess the performance of these methods in detecting concept changes in different scenarios, different experiments were carried out. The number of false alarms, the missed detections and detection delay time were evaluated using data underlying a Bernoulli distribution and public data sets.

This set of experiments uses data streams of lengths L = 2.000, 5.000 and 10.000, underlying a stationary Bernoulli distribution of parameter $\mu = 0.2$ during the first $L - 1.000$ examples. During the last 1.000 examples, the parameter is linearly increased to simulate concept drifts with different magnitudes. The following slopes were used: 0 (no change), $10^{-4}, 2.10^{-4}, 3.10^{-4}$ and 4.10^{-4}. For each type of simulated drift, 100 data streams were generated with different seeds. These experiments also allow to analyze the influence, in the delay time until detections, of the length of the stationary part (the first $L - 1.000$ samples). With respect to the ACWM-fh, since in this experiment data were binary, the number of bins in the histograms is $k = \frac{R}{2\sqrt{\varepsilon}} = 3$ and, therefore, the length of the reference window was set as $L/5$ (instead as k) and the initial evaluation step ($IniEvalStep$) was set to 50. Note that the choice of $IniEvalStep$ does not great affect the detection delay time results. The change detection threshold (δ) was set to 10^{-4}. The parameters δ and λ of the PHT were set equal to 0.05 and 10, respectively. For the ADWIN, the values of 5 and 0.05 were used for the parameters M and δ, in order to present similar false alarm rates to DDM. These parameters values were decided using similar training data.

Table 9 shows a summary of the performance of the four methods compared: ADWIN, DDM, PHT and ACWM-fh (with fading factor $\alpha = 0.9994$). The rows are indexed by the value of L and corresponding slope, presenting the delay time (DDT) until the detection of the change that occurs at

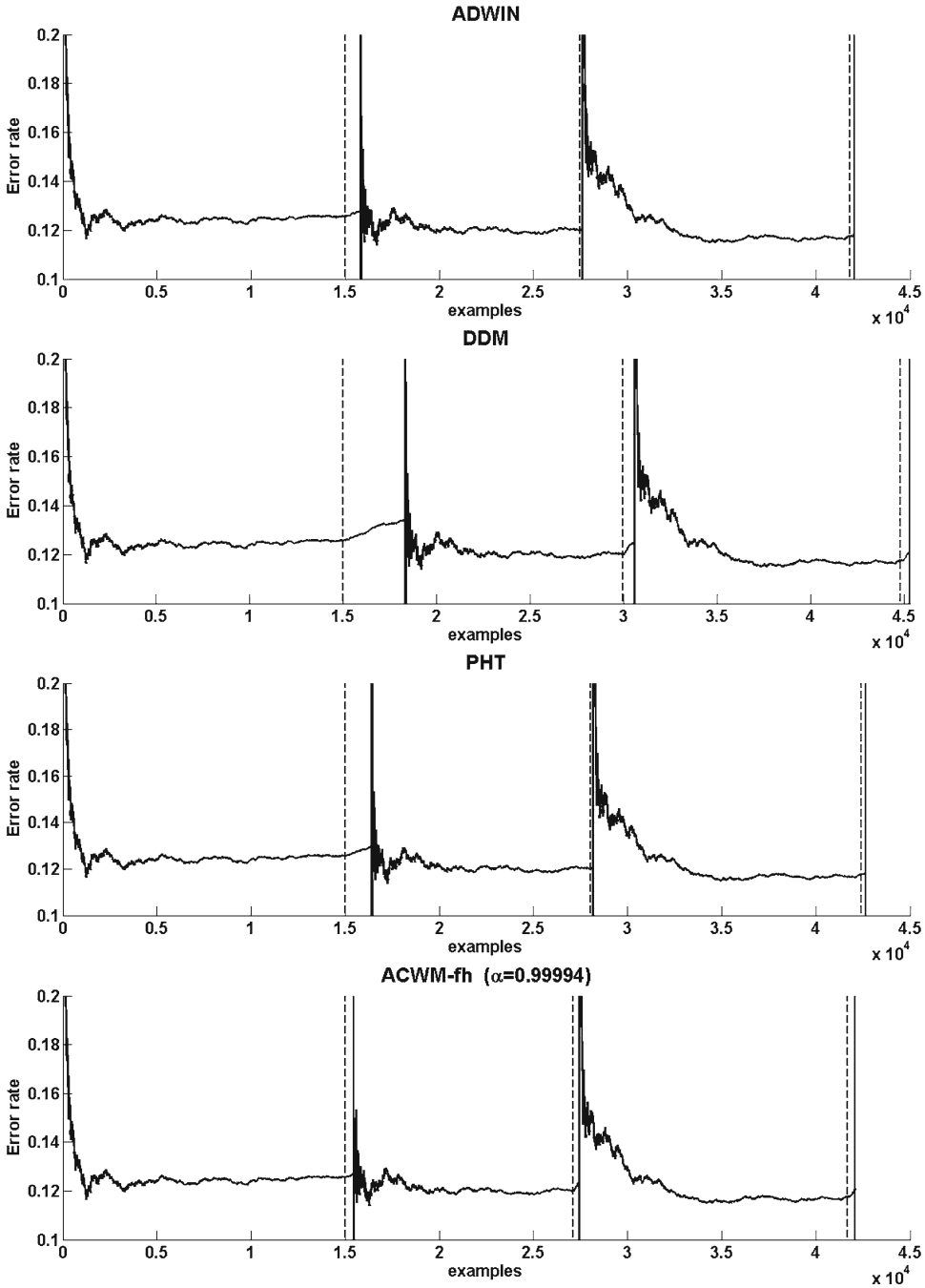

Fig. 14 Evolution of the error rate and the delay times in drift detection using the four presented methods (ACWM was performed using fading factors—$\alpha = 0.9994$). *Vertical dashed lines* indicate drift in data, and vertical *solid lines* indicate when drift was detected

time stamp $L - 1.000$ (averaged over the 100 runs), the total number of missed detections (#MD) and the total number of false alarms (#FA).

For different stream lengths, the first row (slope 0) gives the number of false alarms. The PHT tends to present more false alarms than any of the other methods. The ACWM-fh only presents false alarms for streams with a length of 10.000. For these cases, the number of false alarms of ACWM-fh and ADWIN is similar.

In general, the increase in the data streams length leads to an increase in the number of false alarms and missed detections. As is reasonable for all the methods, the increase in the slope of Bernoulli's parameter contributes to a decrease in the time until the change is detected. Fewer false alarms and missed detections resulted also from slope increases. For all the streams and looking at the detection delay time, the ADWIN wins over DDM, presenting a similar number of false alarms and missed detections. For detection delay time, in all the cases, the PHT outperforms the ADWIN. Additionally, in most cases, PHT does not miss changes. However, PHT results are compromised with the highest number of false alarms among the four methods. In a concept drift prob-

Table 10 Detection delay time of the compared methods, when performed in the SEA data set

# Drift	Detection delay time			
	ADWIN	DDM	PHT	ACWM-fh
1	826	3314	1404	553
2	115	607	118	234
3	242	489	258	288

Table 11 MATLAB execution time when performing the methods analyzed

# Drift	Execution time			
	ADWIN	DDM	PHT	ACWM-fh
1	0.9424	0.9387	2.4780	0.0607
2	0.6145	0.4774	1.2605	0.2386
3	0.7360	0.6219	1.8829	0.0715

lem, when a change detector is embedded in a learning algorithm, this is a huge drawback. The occurrence of a concept drift implies the relearning of a new model in order to keep up with the current state of nature. In the presence of a false detection, the model, which is updated, will be unnecessarily replaced by a new one. On the other hand, in learning scenarios, missed detections are also harmful. They entail outdated models that are not describing the new evolving data.

Regarding this trade-off between false alarms and missed detections, the ACWM-fh presents the best results, with detection delay times almost as low as the ADWIN.

7.2 Experiments on a public data set

In the previous experiment, the data set did not allow the performance of the different change detection methods to be evaluated in large problems, which is important since concept drift mostly occurs in huge amounts of data arriving in the form of streams. To overcome this drawback, an evaluation of the change detection algorithms was performed using the SEA concepts data set [38], a benchmark problem for concept drift. Figure 14 shows the error rate (computed using a naive-Bayes classifier), which presents three drifts. The drifts and the corresponding detections, signed by the analyzed methods, are represented by dashed and solid lines, respectively. Concerning the ACWM-fh, since $k = 3$ and considering the length of the stream, the length of the reference window was set as $1000k$ (instead as k) and the initial evaluation step ($IniEvalStep$) was set to $100k$. The change detection threshold (δ) was set to 10^{-4}. The input parameters for the other three methods remain the same as in the previous experiment.

Table 10 presents the delay time in detecting concept drifts in this data set. In can be seen that all the algorithms require too many examples to detect the first drift. The exception is ACWM-fh ($\alpha = 0.9994$), which takes only almost half the examples of the second "best" method (ADWIN) to detect the first drift. For all the methods, the resilience to false alarms and the ability to reveal changes without missing detections must be stressed.

Comparing the results in detecting the first drift, the ACWM-fh has a clear advantage, significantly reducing the detection delay time. For the second drift, with respect to detection delay times, the performance of ADWIN and PHT is similar, and smaller than the one presented by ACWM-fh. Concerning the third drift, the performance of ADWIN, PHT and ACWM-fh is similar. These three methods clearly perform better than DDM. It must be pointed out that, with the exception of PHT (which presents 1 false alarm), the other three methods were resilient to false alarms.

In evolving learning scenarios, the time required to process examples plays an important role. When comparing the MATLAB execution time of these methods, the ACWM-fh presents smaller execution time against the other methods (for the three drifts), as shown it Table 11. It must be pointed out that ADWIN was performed in MATLAB, but the code was implemented in JAVA, which may increase the execution time.

7.3 Experiments on generated data streams

To compare the performance of the different change detection methods in drift scenarios, several data streams were generated. The data stream generators are Waveform, LED, RT and RBF as implemented in MOA [8]. The Waveform stream is a three class problem defined by 21 numerical attributes, the RBF and the RT streams are two-class problems defined by 10 attributes. The data were generated by emulating a concept drift event as a combination of two distributions. For the Waveform and RBF data sets, the first distribution was generated with the WaveformGenerator and the RandomRBFGenerator (respectively) and the second distribution was generated with the WaveformGeneratorDrift and the RandomRBFGeneratorDrift (respectively). For second stream of Waveform data set, the number of attributes with drift was set to 21. The second RBF stream was generated setting the seed for the random generation of the model to 10 and adding speed drift to the centroids of the model (0.01). For the RT data set, both distributions were generated with the RandomTreeGenerator, varying the seed of the second concept. For all the streams, the change occurs at example $32k$. The learning algorithms are VFDT majority class (VFDT-MC) and VFDT Naive-Bayes adaptive (VFDT-NBa) as implemented in MOA. Figure 15 presents, for both

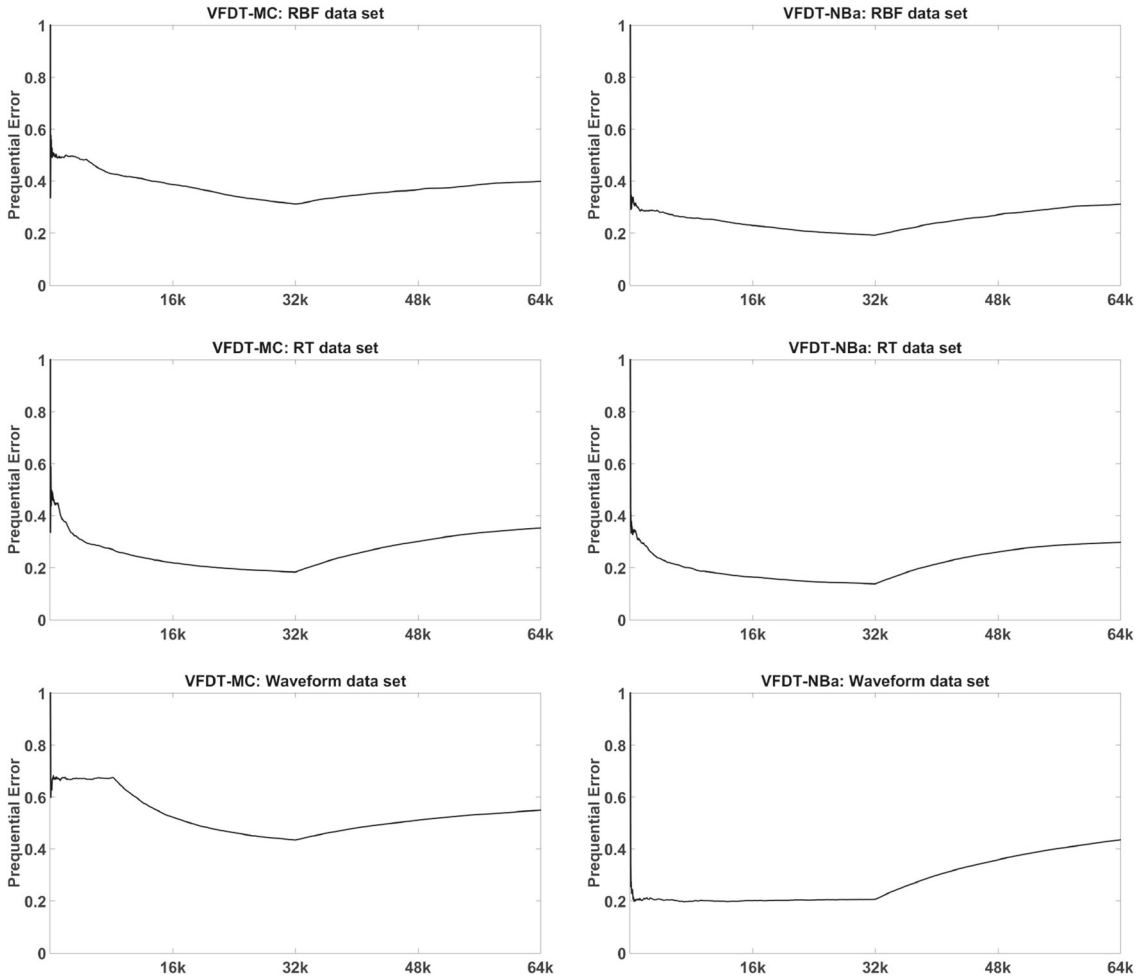

Fig. 15 Prequential error, for VFDT-MC and VFDT-NBa learning algorithms, for the 3 data sets considered

learning algorithms, the prequential error in each data set. After an initial and abrupt decrease, it is possible to observe that around example $32k$, the prequential error increases as expected and as a result of the concept drift emulated.

Table 12 presents a summary of the performance of the four methods compared: ADWIN, DDM, PHT and ACWM-fh (with fading factor $\alpha = 0.9994$). Besides the detection delay time and the execution time, the total number of missed detections and the total number of false alarms are also shown. The results report the average and standard deviation of 5 runs.

From Table 12 it can be noticed that the drift in the RBF data set is the most difficult to detect, since when using the error from both the learning algorithms, all the 4 methods presented higher detection delay time (with the exception of the ACWM-fh, with fading factor $\alpha = 0.9994$, when applied to the error of the VFDT-NBa algorithm). It also be stressed out the ability of the 4 change detection methods in detecting drifts and their resilience to false alarms (only the DDM presented 3 false alarms when detecting drifts on the waveform error of the VFDT-NBa algorithm). It can also be observed that for the 3 data sets and both learning algorithms, the ADWIN outperforms the other 3 methods, presenting smaller detection delay time. However, it requires more time to process the data than the other 3 methods (it must be pointed out that ADWIN was performed in MATLAB, but the code was implemented in JAVA, which may increase the execution time). With respect to the other 3 methods, regarding the detection delay and execution times, the ACWM-ff presents the smallest results.

Moreover, as stated before, when learning in dynamic scenarios, the time required to process examples is of utmost importance. When embedding a change detection method in a learning algorithm it must be taking into account the trade-off between detection delay and execution times. The overall accuracy of the learning algorithm will depend of the earlier adaptation in the presence of drifts. On the other hand, the time required to process examples must be minimal to

Table 12 Detection delay time (DDT) and execution time for the 4 compared change detection methods

	ADWIN		DDM		PHT		ACWD-fh (0.994)	
	DDT	ExTime (av.)	DDT	ExTime	DDT	ExTime	DDT	ExTime
VFTD-MC								
RBF	109 ± 56	6.09	1144 ± 139	4.94	767 ± 156	1.53	365 ± 117	0.41
RT	31 ± 14	4.73	436 ± 78	5.51	317 ± 57	1.64	249 ± 71	0.53
WAVE	46 ± 16	3.82	787 ± 120	5.53	458 ± 89	1.74	361 ± 125	0.65
VFTD-Nba								
RBF	36 ± 16	5.52	615 ± 67	5.07	443 ± 55	1.35	206 ± 64	0.22
RT	26 ± 15	4.47	351 ± 38	5.51	280 ± 26	1.67	302 ± 157	0.12
WAVE	22 ± 10	3.36	220 ± 186 (0;3)	5.01	281 ± 202	1.30	189 ± 102	0.62

The results report the average and standard deviation of 5 runs. In parenthesis is the number of runs, if any, where the algorithm misses detection or signals a false alarm: They are in the form (Miss; False Alarm)

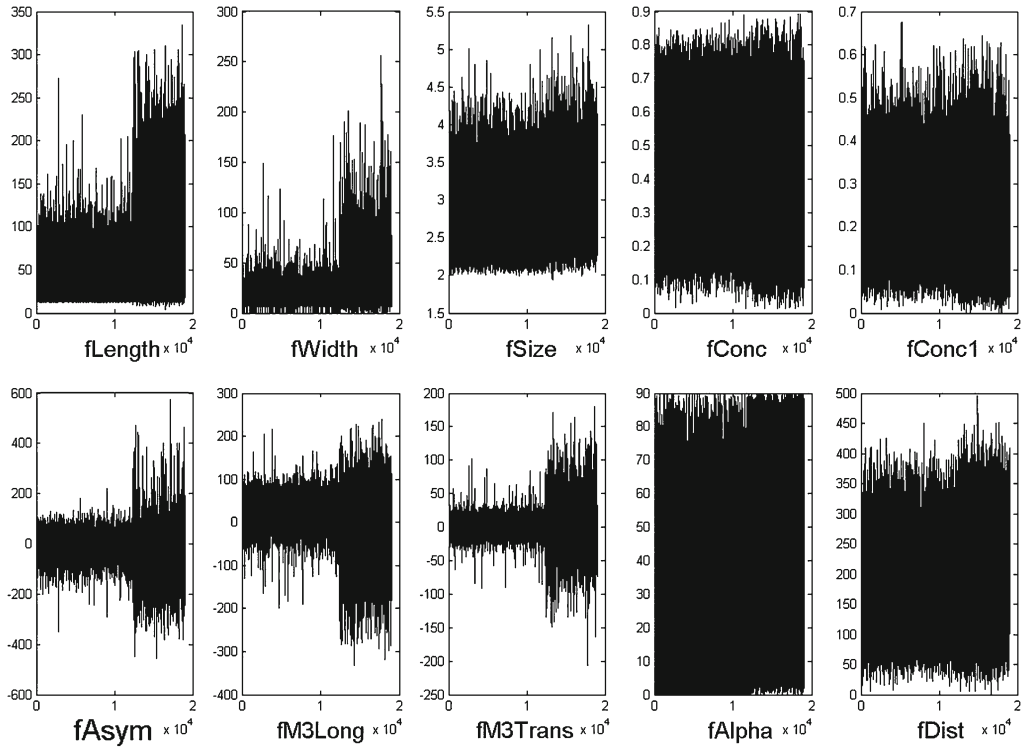

Fig. 16 Data distribution of each attribute in the MAGIC gamma telescope data set

guarantee that the process is performed at the arrival rate of data. Considering the results from this experiment, it seems that the ACWM is the best method to embed in a learning algorithm.

7.4 Numerical high dimensional data set: MAGIC gamma telescope benchmark

To evaluate the performance of the ACWM in multidimensional data, the UCI MAGIC gamma telescope [28] data set was used. The use of such data set has the advantage of consisting of two known classes, allowing an easy validation of the splits. This data set, which consists of 19,020 data points in 2 classes with 10 numerical (real) attributes ('fLength', 'fWidth', 'fSize', 'fConc', 'fConc1', 'fAsym', 'fM3Long', 'fM3Trans', 'fAlpha' and 'fDist'). To transform this data set into a data stream with a change, the data set was split on class label, simulating a single stream composed first by the examples of class 'gamma' (12,332 data points) and followed by the examples of class 'hadron' (6688 data points), and the

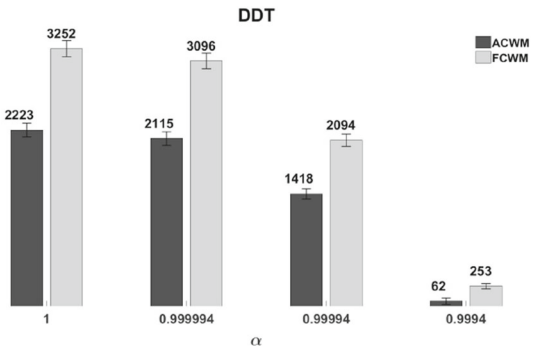

Fig. 17 Detection delay time (average over 100 runs) in the MAGIC gamma telescope data set

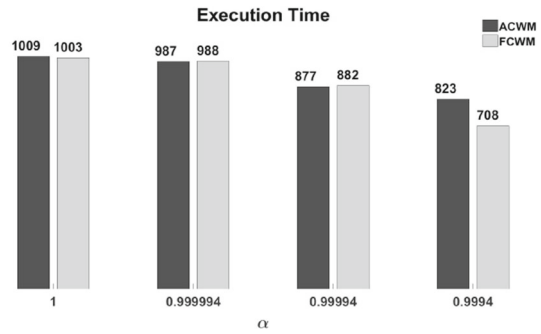

Fig. 18 Execution time (over 100 runs) in the MAGIC gamma telescope data set

class labels were removed. Figure 16 shows the modified data for each attribute in this data set, in some attributes the change is easily observed, while for other is not noticeable.

Since the data are not 'time labeled' and to obtain results independent from the examples order, for each attribute, the examples within each class were shuffled. This strategy was repeated obtaining 100 different ordered data sets.

This approach for detecting changes was evaluated in all these simulated data sets. Performing the change detection test (using ACWM and FCWM), it was expected to detect changes around the class change point (12,332). In multidimensional settings, the comparison of distributions must be performed at the same evaluation point for each attribute. Therefore, the initial evaluation step was set to $IniEvalStep = \min_{i=1,\ldots D}(k_i)$, where D is the number of dimensions. In this experiment, the length of the reference window needed to be adjusted because the range of attribute 'fAsym' was around 1000. Therefore, the L_{RW} was set to $3k$. The remaining parameters were not adjusted.

Figures 17 and 18 present the detection delay time (average results for 100 runs) and execution time, respectively, of the CMW when performed in this multidimensional data set. It can be observed that, along the different fading factors used in the histograms, the ACWM outperforms FCWM (with similar execution time, as show in Fig. 18). The use of fading factors reveals to be advantageous, significantly reducing the detection delay time. It must be pointed out that this decrease in the detection delay time was obtained in similar execution time as presented in Fig. 18. The resilience to false alarms and the ability to reveal changes without missing detections must be stressed (only in the ACWM-fh with $\alpha = 0.9994$ was obtained 2 false alarms and 2 missed detections).

8 Conclusions and further research

This paper presents a windowing scheme to detect distribution and concept changes. The ACWM is based on the online monitoring of the distance between data distributions, which is evaluated through a dissimilarity measure based on the asymmetry of the Kullback–Leibler divergence. The novelty relies on the approach that provides the representation of the data distribution. The fading histograms provide a more updated representation of the data since outdated data are gradually forgotten. The advantage of such a representation structure works in favor of the detection of changes. The experimental results on artificial and real data show that when using fading histograms to represent data instead of standard histograms, the time to detect a change is significantly reduced. The obtained results also sustain that the ACWM to detect distribution changes is robust to noise and exhibit a good performance under several stationary phases. Moreover, the experiments carried out with respect to the detection of concept changes show that, considering both the false alarms and the missed detections, the ACWM-fh outperforms the other methods and presents detection delay times similar to ADWIN. The advantage of using fading histograms to compare data distributions is also disclosed in a multidimensional data set. Overall, the ACWM-fh presented a performance which was highly resilient to false alarms, under different kinds of distribution changes. However, this windowing model detects changes but does not provide insights on the description of a change. Nowadays, data are becoming increasingly evolved, making mandatory to go beyond the detection of changes and performing change analysis. For instance, it is important to identify if a change is an increase or a decrease and explore if there are relations or explanations on the occurrence of changes. Furthermore, the construction of the fading histograms must take into consideration that in dynamic scenarios, the range of the variable may shrinks or stretches over time. Therefore, the intervals must be adaptive, evolving over time. Moreover, when embracing the multidimensional data, the ACWM must take into consideration possible correlations between features.

Acknowledgements This work was supported by the Portuguese Science Foundation (FCT) through national funds, and co-funded by the FEDER, within the PT2020 Partnership Agreement and COMPETE2020 under projects IEETA (UID/CEC/00127/2013), SYSTEC

(POCI-01-0145-FEDER- 006933), Cloud Thinking (funded by the QREN Mais Centro program, ref. CENTRO-07-ST24- FEDER-002031) and VR2market (funded by the CMU Portugal program, CMUPERI/FIA/0031/2013). The postdoc grant of R. Sebastião (BPD/UI62/6777/2015) is also acknowledged. J. Gama acknowledges the support given by European Commission through MAESTRA (ICT- 2013-612944) and the Project TEC4Growth financed by the North Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement.

Compliance with ethical standards

Conflicts of interest The authors declare that they have no conflict of interest.

References

1. Ayres-de Campos, D., Bernardes, J., Garrido, A., Marquesde-de Sà, J., Pereira-Leite, L.: Sisporto 2.0: a program for automated analysis of cardiotocograms. J. Matern. Fetal Med. **9**(5), 311–318 (2000)
2. Ayres-de Campos, D., Sousa, P., Costa, A., Bernardes, J.: Omniview-SisPorto 3.5—a central fetal monitoring station with online alerts based on computerized cardiotocogram+ST event analysis. J. Perinatal Med. **36**(3):260–264. doi:10.1515/JPM.2008.030, http://www.ncbi.nlm.nih.gov/pubmed/18576938 (2008)
3. Bach, S., Maloof, M.: Paired learners for concept drift. In: Eighth IEEE International Conference on Data Mining, 2008. ICDM '08, pp. 23–32 (2008). doi:10.1109/ICDM.2008.119
4. Baena-García, M., Campo-Ávila, J.D., Fidalgo, R., Bifet, A., Gavaldà, R., Morales-Bueno, R.: Early drift detection method. In: In 4th ECML PKDD International Workshop on Knowledge Discovery from Data Streams, pp. 77–86 (2006)
5. Basseville, M., Nikiforov, I.: Detection of Abrupt Changes: Theory and Applications. Prentice-Hall, Englewood Cliffs (1993)
6. Berthold, M., Hand, D.J. (eds.): Intelligent Data Analysis: An Introduction, 1st edn. Springer, New York, Inc., Secaucus, NJ, USA (1999)
7. Bifet, A., Gavaldà, R.: Learning from time-changing data with adaptive windowing. In: In SIAM International Conference on Data Mining, Berlin, Heidelberg (2007)
8. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: Moa: Massive online analysis. J. Mach. Learn. Res. **11**, 1601–1604 (2010)
9. Chakrabarti, K., Garofalakis, M.N., Rastogi, R., Shim, K.: Approximate query processing using wavelets. In: Abbadi, A.E., Brodie, M.L., Chakravarthy, S., Dayal, U., Kamel, N., Schlageter, G., Whang, K.Y. (Eds.) VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10–14, 2000, Cairo, Egypt, Morgan Kaufmann, pp. 111–122 (2000)
10. Cormode, G., Muthukrishnan, S.: An improved data stream summary: the count-min sketch and its applications. J Algorithms **55**(1), 58–75 (2005). doi:10.1016/j.jalgor.2003.12.001
11. Cormode, G., Muthukrishnan, S.: What's hot and what's not: tracking most frequent items dynamically. ACM Trans. Database Syst. **30**(1), 249–278 (2005). doi:10.1145/1061318.1061325
12. Correa, M., Bielza, C., Pamies-Teixeira, J.: Comparison of bayesian networks and artificial neural networks for quality detection in a machining process. Expert Syst. Appl. **36**(3):7270–7279. http://dblp.uni-trier.de/db/journals/eswa/eswa36.html#CorreaBP09 (2009)
13. Dasu, T., Krishnan, S., Venkatasubramanian, S., Yi, K.: An information-theoretic approach to detecting changes in multi-dimensional data streams. In: Proceedings of the Symposium on the Interface of Statistics, Computing Science, and Applications (2006)
14. Gama, J.: Knowledge Discovery from Data Streams, 1st edn. Chapman & Hall/CRC, London (2010)
15. Gama, J., Medas, P., Castillo, G., Rodrigues, P.P.: Learning with drift detection. In: In SBIA Brazilian Symposium on Artificial Intelligence, Springer Verlag, pp. 286–295 (2004)
16. Gama, J., Sebastião, R., Rodrigues, P.P.: On evaluating stream learning algorithms. Mach. Learn. **90**(3), 317–346 (2013)
17. Gao, J., Fan, W., Han, J., Yu, P.S.: A general framework for mining concept-drifting data streams with skewed distributions. In: In Proceedings of SDM'07, pp. 3–14. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.74.3098 (2007)
18. Gilbert, A.C., Kotidis, Y., Muthukrishnan, S., Strauss, M.J.: One-pass wavelet decompositions of data streams. IEEE Trans. Knowl. Data Eng. **15**(3), 541–554 (2003). doi:10.1109/TKDE.2003.1198389
19. Gonçalves, H., Bernardes, J., Paula Rocha, A., Ayres-de Campos, D.: Linear and nonlinear analysis of heart rate patterns associated with fetal behavioral states in the antepartum period. Early Hum. Dev. **83**(9), 585–591 (2007)
20. Guha, S., Shim, K., Woo, J.: Rehist: Relative error histogram construction algorithms. In: Proceedings of the 30th International Conference on. Very Large Data Bases, pp. 300–311 (2004)
21. Guha, S., Koudas, N., Shim, K.: Approximation and streaming algorithms for histogram construction problems. ACM Trans. Database Syst. **31**(1), 396–438 (2006). doi:10.1145/1132863.1132873
22. Ioannidis, Y.: The history of histograms (abridged). In: Proceedings of the 29th International Conference on Very Large Data Bases—Volume 29, VLDB Endowment, VLDB '03, pp. 19–30. http://dl.acm.org/citation.cfm?id=1315451.1315455 (2003)
23. Jagadish, H.V., Koudas, N., Muthukrishnan, S., Poosala, V., Sevcik, K.C., Suel, T.: Optimal histograms with quality guarantees. In: Proceedings of the 24rd International Conference on Very Large Data Bases, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, VLDB '98, pp. 275–286. http://dl.acm.org/citation.cfm?id=645924.671191 (1998)
24. Karras, P., Mamoulis, N.: Hierarchical synopses with optimal error guarantees. ACM Trans. Database Syst. **33**, 1–53 (2008). doi:10.1145/1386118.1386124
25. Kifer, D., Ben-David, S., Gehrke, J.: Detecting change in data streams. In: Proceedings of the Thirtieth international conference on Very large data bases—Volume 30, VLDB Endowment, VLDB '04, pp. 180–191. http://dl.acm.org/citation.cfm?id=1316689.1316707 (2004)
26. Kullback, S., Leibler, R.A.: On information and sufficiency. Ann. Math. Stat. **22**, 49–86 (1951)
27. Kuncheva, L.I.: Classifier ensembles for detecting concept change in streaming data: overview and perspectives. In: 2nd Workshop SUEMA 2008 (ECAI 2008), pp. 5–10. (2008)
28. Lichman, M.: UCI Machine Learning Repository. http://archive.ics.uci.edu/ml (2013)
29. MATLAB®& Simulink®. Student Version R2007a. The MathWorks Inc., Natick, Massachusetts (2007)
30. Misra, J., Gries, D.: Finding repeated elements. Sci. Comput. Program. **2**, 143–152 (1982). doi:10.1016/0167-6423(82)90012-0
31. Mitchell, T.M.: Machine Learning, 1st edn. McGraw-Hill Inc., New York (1997)
32. Mouss, H., Mouss, D., Mouss, N., Sefouhi, L.: Test of page-hinckley, an approach for fault detection in an agroalimentary production system. In: Control Conference, 2004. 5th Asian, vol. 2, pp. 815–818 (2004)
33. Nishida, K., Yamauchi, K.: Detecting concept drift using statistical testing. In: Proceedings of the 10th International Conference on Discovery Science, Springer-Verlag, Berlin, Heidelberg, DS'07,

pp. 264–269. http://dl.acm.org/citation.cfm?id=1778942.1778972 (2007)
34. Page, E.S.: Continuous inspection schemes. Biometrika **41**(1–2), 100–115 (1954). doi:10.1093/biomet/41.1-2.100
35. Sebastião, R., Gama, J.: Change detection in learning histograms from data streams. In: Proceedings of the 13th Portuguese Conference on Artificial Intelligence, Springer-Verlag, Berlin, Heidelberg, EPIA'07, pp. 112–123. http://dl.acm.org/citation.cfm?id=1782254.1782265 (2007)
36. Sebastião, R., Gama, J., Mendonça, T.: Comparing data distribution using fading histograms. In: ECAI 2014—21st European Conference on Artificial Intelligence, 18–22 August 2014, Prague, Czech Republic—Including Prestigious Applications of Intelligent Systems (PAIS 2014), pp. 1095–1096 (2014)
37. Sebastião, R., Gama, J., Mendonça, T.: Constructing fading histograms from data streams. Prog. Artif. Intell. **3**(1), 15–28 (2014). doi:10.1007/s13748-014-0050-9
38. Street, W.N., Kim, Y.: A streaming ensemble algorithm (sea) for large-scale classification. In: Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM Press, pp. 377–382 (2001)
39. Vitter, J.S.: Random sampling with a reservoir. ACM Trans. Math. Softw. **11**(1), 37–57 (1985). doi:10.1145/3147.3165
40. Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. Mach. Learn. **23**(1), 69–101 (1996). doi:10.1023/A:1018046501280