Online and Non-parametric Drift Detection Methods Based on Hoeffding's Bounds

Isvani Frías-Blanco, José del Campo-Ávila, Gonzalo Ramos-Jiménez, Rafael Morales-Bueno, Agustín Ortiz-Díaz and Yailé Caballero-Mota

Abstract—Incremental and online learning algorithms are more relevant in the data mining context because of the increasing necessity to process data streams. In this context, the target function may change over time, an inherent problem of online learning (known as concept drift). In order to handle concept drift regardless of the learning model, we propose new methods to monitor the performance metrics measured during the learning process, to trigger drift signals when a significant variation has been detected. To monitor this performance, we apply some probability inequalities that assume only independent, univariate and bounded random variables to obtain theoretical guarantees for the detection of such distributional changes. Some common restrictions for the online change detection as well as relevant types of change (abrupt and gradual) are considered. Two main approaches are proposed, the first one involves moving averages and is more suitable to detect abrupt changes. The second one follows a widespread intuitive idea to deal with gradual changes using weighted moving averages. The simplicity of the proposed methods, together with the computational efficiency make them very advantageous. We use a Naïve Bayes classifier and a Perceptron to evaluate the performance of the methods over synthetic and real data.

Index Terms—concept drift, control chart, incremental learning, weighted moving average.

1 Introduction

LEARNING from data streams, the target concept of which can change over time, is a research area of growing interest. In these situations, large volumes of data are acquired over time, possibly at a high incoming rate. For example, such changes can emerge due to changing clothing preferences (e.g. given by a season change), news preferences, energy consumption, etc. Spam filtering is another example: spammers try to elude filters by disguising their emails while spam filters must be updated to successfully identify spam over time. So it is possible that a learning model previously induced may be inconsistent with the current data, making an update necessary. This problem is commonly known as *concept drift*. Many learning algorithms have been implemented as the base model for handling concept drift, such as rule-based systems, decision trees, Naïve Bayes, support vector machines, instance based learning, and ensemble of classifiers [1], [2].

In supervised incremental learning, a well-extended approach to handle concept drift constantly monitors a performance measure (e.g. accuracy) of the learning model. If a significant drop in this measure is estimated a concept drift is assumed and some actions are defined to update the model according to the latest data. In this strategy, change detectors that are independent of the learning algorithm play a crucial role [3], [4], [5]. These detectors often operate over a stream of real values (corresponding to a given performance measure); due to the difficulty of detecting online distributional changes, most existing approaches monitor changes in a suitable statistic, such as the mean or median [6]. Thus, the problem of concept drift detection is reduced to estimating significant changes in the statistic calculated from the sequence of values that measure a performance characteristic.

Often, this stream of real values is also large (possibly infinite), since the learning model is monitored over time. Therefore, it is common to impose restrictions on these online change detectors [6], [7]. The computational complexity required to process each performance value must be constant and methods should be single-pass, where each performance value is processed once and then discarded. These change detectors must also deal with common types of change prevalent in many real-world data [2]. Under these conditions, many traditional statistical approaches that assume a fixed size of the input data for estimating distributional changes are not suitable. Some of the most studied parametric schemes to detect changes online are Shewhart's control charts, exponentially weighted moving average (EWMA) control charts and Page's cumulative sum (CUSUM) procedure [8]. However, in many situations the data are not ruled by these well-known probability distributions, and non-parametric approaches are more suitable.

Some existing approaches do not fulfill the afore-

- *I. Frías-Blanco is with the Regional Faculty of Granma, University of Computer Sciences, Granma, Cuba, e-mail: ifriasb@grm.uci.cu.*
- *J. del Campo-Ávila, G. Ramos-Jiménez and R. Morales-Bueno are with the Department of Lenguajes y Ciencias de la Computación, University of Málaga, Complejo Tecnológico, 29071 Málaga, Spain, e-mail: jcampo@lcc.uma.es, ramos@lcc.uma.es, morales@lcc.uma.es.*
- *A. Ortiz-Díaz is with the Department of Computer Science, University of Granma, Granma, 85100 Cuba, e-mail: agustin@udg.co.cu.*
- *Y. Caballero-Mota is with the Department of Computer Science, University of Camagüey, Camagüey, Cuba, e-mail: yailec@yahoo.com.*

Digital Object Identifier no. 10.1109/TKDE.2014.2345382

mentioned computational restriction [5], [9]. For example, ADWIN2 [5] keeps a window of length W with $O(\log W)$ memory and update time, where W is the number of measured values generated after the last drift detection. Another related methods, DDM [3], EDDM [4] and ECDD [10] do not provide rigorous guarantees of performance; whilst DDM [3] and ECDD [10] assume measured values given according to a Bernoulli distribution, so they are restricted to a stream of bits. ECDD, which uses the EWMA estimator for concept drift detection, only focuses on the false positives rate.

This paper concerns about the problem of detecting concept drift in supervised incremental learning. Specifically, we propose a family of methods for monitoring over time the mean as estimated from a sequence of real values (corresponding to the performance measure) in order to detect significant changes. Thus, we extend some methods for detecting change in data streams by removing any assumption related with the probability density function that generates these measured values. Instead, we assume that these values are given according to independent and bounded random variables. Additionally, the proposed methods take care of important issues of online learning: they are single pass, process each incoming measured value in constant time and space complexity, and provide rigorous guarantees of performance in the form of bounds on false positives and false negatives rate.

Although in this paper we study the concept drift problem assuming that all the incoming instances are labeled, the proposed change detectors can also be applied to more realistic scenarios in which instances arrive online but labels are harder to obtain [11].

This paper is structured as follows. Section 2 provides the problem statement and its main peculiarities. Later, in Section 3 we review some outstanding research work dealing with the change detection problem in data streams. Then, a nonparametric two-sample test involving moving averages with strong probabilistic guarantees is discussed in Section 4. In Section 5 this test is generalized to improve the detection when the change is gradual by means of weighted moving averages. The algorithm that detects changes online from these tests is shown in Section 6. Section 7 presents a study that shows the performance of the methods over both synthetic and real data. Finally, we conclude this paper in Section 8, summarizing the most salient results and proposing future work.

2 Notations and Definitions

Consider the following incremental learning scenario. A very large (or possibly infinite) sequence $S = (\vec{a}_1, c_1); (\vec{a}_2, c_2); \ldots$ of pairs $(\vec{a}_i, c_i)$, named *examples* (or *instances*), arrives over time, where $\vec{a}_i \in \overrightarrow{\mathcal{A}}$ is a vector in which each component is called *attribute* and $c_i \in \mathcal{C}$ is its corresponding *class label* taken from a finite set $\mathcal{C}$ named *class*. Assuming the existence of a target function $c_i = f(\vec{a}_i)$, the incremental learning task is to obtain a model $\hat{f}$ that approximates f, so that $\hat{f}$ maximizes the prediction accuracy. Often it is also assumed that the examples are regulated by a probability density function $P(\overrightarrow{\mathcal{A}}, \mathcal{C})$. *Concept* refers to the whole distribution $P(\overrightarrow{\mathcal{A}}, \mathcal{C})$ of the problem at a certain point in time. Therefore, a change in the whole distribution of the problem is called *concept change* or concept drift.

2.1 The Speed of Change

Speed of change refers to the transition period between consecutive concepts. For example, tracking abrupt and gradual changes is an important issue since these types of change are prevalent in many real-world data. Often they are handled separately on different learning algorithms or by using different change detectors [3], [4]. If $S = (\vec{a}_1, c_1); (\vec{a}_2, c_2); \ldots; (\vec{a}_t, c_t); ...; (\vec{a}_n, c_n)$ is a sequence of n examples where each $(\vec{a}_i, c_i)$ has associated with it the concept $P_i = P(\vec{a}_i, c_i)$, a change is abrupt if there is a *change point* $t+1$ such that P_i remains constant for $0 < i \leq t$, and changes at this point (t+1) to P'_j remaining constant for $t < j \leq n$.

On the other hand, some research studies have identified various types of gradual changes [2], [12], [13], [14]. For instance, Helmbold and Long [13] assume a possible permanent but slow concept drift defining $P(f_i(\overrightarrow{\mathcal{A}}) \neq f_{i+1}(\overrightarrow{\mathcal{A}})) \leq \Delta$, where $\Delta > 0$ is a bound for the drift rate on the sequence of target functions $f_i(\overrightarrow{\mathcal{A}})$. Other theoretical results [12] have assumed that examples are generated independently and randomly from a sequence P_i of joint distributions over $\overrightarrow{\mathcal{A}} \times \{0, 1\}$ supposing that consecutive pairs of distributions have at most γ total variation distance ($0 < \gamma \leq 1$).

2.2 Monitoring Statistics

The change detectors proposed in this paper can be applied to a well-known scheme for handling concept drift with two modules [3], [4], [5]: the change detector and the learning algorithm. Change detectors can alternate between two states: *in-control* where it is estimated that the concept is stable and *out-control* achieved when estimating a concept change. As each example arrives the learning algorithm makes a prediction $\hat{c}_i \in \mathcal{C}$ based on the vector of attributes $\vec{a}_i$. Then, the change detector is updated via a loss function $L(\hat{c}_i, c_i)$ defined as $L : \mathcal{C} \times \mathcal{C} \to \mathbb{R}$ (e.g. 0-1 loss function with $L(\hat{c}_i, c_i) = 1$ if $\hat{c}_i \neq c_i$ and $L(\hat{c}_i, c_i) = 0$ otherwise). Next, the aforementioned example $(\vec{a}_i, c_i)$ is provided for the learning algorithm to continue its training.

Thus, the change detector receives as input a stream of real values (calculated from the values estimated by the loss function) and gives as output the information of the current status estimated by the detector. The family of methods proposed in this article applies to this scheme if $L(\hat{c}_i, c_i)$ corresponds to independent and bounded random variables.

Following the PAC learning model, if the distribution of the examples is stationary, the error rate of the learning algorithm will decrease when the number of examples increases [3], [4]. This way, a significant increase in the mean (e.g. in the error rate of the learning algorithm) as estimated when tracking $L(\hat{c}_i, c_i)$ over time, may in fact signify a concept drift.

As we have already mentioned, a very widespread intuitive idea for detecting concept drift is to monitor some convenient statistic $\hat{X}$ over time, computed over a stream $x_1, x_2, ..., x_t, x_{t+1}, ..., x_n$ of performance values. In this paper we study moving averages and weighted moving averages, which have the form $\hat{X} = \sum_{i=1}^{n} v_i x_i$ (e.g. on moving averages $\forall i,\ v_i = 1/n$ and $\hat{X} = \overline{X}$); additionally, we assume performance values generated according to n independent random variables $X_1, X_2, ..., X_t, X_{t+1}, ..., X_n$ bounded, real-valued and univariate. The problem in hand is to estimate if $\hat{X}$ changes significantly over time. In abrupt changes (that is to say $E(X_i) = \mu_0$ for $0 < i \leq t$ and $E(X_j) = \mu_1 \neq \mu_0$ for $t < j \leq n$), the problem is to estimate weather such a change point $t+1$ exists in this sequence. Unfortunately, the assumption of abrupt change is often unrealistic.

Frequently, with gradual changes there is not a defined change point but a transition period between consecutive concepts. Slow and permanent gradual changes are more difficult to track. In these cases, the population mean of the performance values (e.g. the accuracy) often varies slowly and continuously.

3 Related Work

Although the problem of detecting arbitrary distributional changes has been extensively studied in the statistical community, far fewer approaches have studied it in data stream environments. In the following subsections we mention relevant algorithms used in online learning, which fall in two main categories: window strategies and weighted-instance approaches.

3.1 Window Strategies

Due to the huge amount of data, it is important to define how the incoming data will be recorded. In the area of machine learning and data mining, the incoming data are generally stored in a buffer called *time window*. Gama and Rodrigues [15] also mention three relevant models within this category: landmark, sliding and tilted windows. In incremental learning, time windows have been used to process the input examples, associating with each example a time stamp that defines its age. Since it is not feasible to exactly calculate statistics relative to several sub-windows, relevant algorithms have been proposed to approximately maintain statistics, offering mathematical guarantees to bound the error of the estimation [7].

These results have also been extended to detect significant changes in the population mean. For instance, ADWIN (ADaptive WINdowing) [5] eliminates an obsolete part of a window (sub-window) when it concludes that this part has a different distribution. ADWIN2 [5], a more efficient version, does not examine all the window divisions in order to find a good cut point quickly.

However, the temporal complexity of these window-based algorithms depends on the number of values seen so far. Other approaches are capable of detecting change in a single-pass through the observed values [3], [4], guaranteeing a theoretical limit to the maximum processing time of each incoming value. For instance, following the idea of some classical approaches to detect changes online [8], DDM [3] identifies a single cut point in the sequence of incoming values, by counting the number of errors. To improve the detection on gradual and slow types of change, Baena *et al.* [4] consider the distance between two classification errors instead of considering only the number of errors.

3.2 Weighted-Instance Strategies

Lazarescu and Venkatesh [16] show some disadvantages of window strategies. First, there is no single window size to deal with rapid and slow types of concept drift. Even with dynamic adjustable windows, it is difficult to handle continuous and slow changes at the same time. In machine learning, another prominent strategy weights the data [17] or parts of the hypothesis [1], [18] according to their age or current utility.

A widespread solution uses weighted moving averages, where recent performance values receive more weight (since they have more probability of occurrence) than the previous ones. A well-known area involves the EWMA statistic, which is an optimal mean estimator when the mean follows a first-order integrated moving average model [19] and when the mean is subject to random step changes [14]. EWMA is as simple to apply as CUSUM, it can be used to estimate the current mean and performs as well as the CUSUM procedure [20]. Weights on the EWMA mean estimator $\hat{X}_t$ decrease exponentially as in a geometric series on time: $\hat{X}_t = (1-\lambda)\hat{X}_{t-1} + \lambda X_t$, where λ acts as the forgetting factor ($0 < \lambda \leq 1$), X_t is the current value of a sequence of random variables and $\hat{X}_0$ can be taken to be the average of preliminary data. Usually, if the statistic exceeds a specified control limit a change is estimated. Anyway, most research studies assume that the random variables X_t are ruled by a known probability distribution function [8].

Nuñez *et al.* [21] implement a variant of the EWMA procedure to detect concept drift inducing decision trees. Specifically, they consider a concept change if a performance measure has a persistent descent; this persistence being the key to distinguishing between noise and a true concept change.

A recent method, ECDD [10], uses an EWMA chart to monitor the misclassification rate of an online classifier. The control limit is computed by using a Monte Carlo approach and a regression method to fit a polynomial under various conditions. These polynomials, which are computed considering the parameter λ and the required false positives rate, must be stored in a "look-up table" in order to detect change during the stream monitoring.

In the information filtering domain, Klinkenberg [17] indicates that users may change their interests in a specific topic slowly. To deal with this type of change, he also suggests decreasing the importance of older examples by a weighting scheme. In this sense, *global weighting* selects the weight of the example x_t based on their respective age using an exponential aging function $w_\eta(x_t) = \exp(-\eta t)$, where the parameter η performs as the forgetting factor ($\eta \geq 0$). Optimal weights in *local example weighting* are estimated by the performance of the learner on the newest batch of data.

Cohen and Strauss [22] formalize the problem of maintaining sums and statistics of a data stream. They examine some families of decay functions and explore storage requirement mainly for exponential scenarios ($\exp(-\eta t)$), polynomial scenarios ($1/t^\alpha$) and sliding window scenarios. They also propose desired properties for a given family of decay functions, among them a sufficiently rich class of decay rates as particular applications depend on the time scales of correlations between values.

As we discussed, most studied parametric schemes to detect changes online assume an underlying probability density function, but real data rarely follow these well-known parametric distributions. Other methods do not fulfill common computational restrictions for online change detection or do not have strong guarantees of performance. The methods derived in the following sections can detect changes from statistics involving some of the aforementioned weighting schemes. These methods can be applied straightforwardly in online change detectors that consider a single cut point in the stream of real values. In the next section we obtain a more simple non-parametric test from the Hoeffding's inequality concerning moving averages.

4 A-TEST: BOUNDING MOVING AVERAGES

There are methods applied to learning in data stream scenarios that compute confidence intervals for different parameters (e.g. error rate) considering well-known distributions. For example, the normal distribution has been assumed given that the sample size (n) is large enough to approximate an unknown probability distribution function to the normal distribution [5].

Other methods do not assume any probability distribution function and use various probability inequalities. For example, IADEM algorithm family [23] induces decision trees without instances memory using Chernoff's and Hoeffding's bounds; it only stores in the nodes the relevant information in terms of relative frequencies. Then, given a desired confidence ($1-\delta$), it uses concentration bounds to estimate which is the maximum error (ε) and then calculates which is the confidence interval ($[E(\overline{X})-\varepsilon, E(\overline{X})+\varepsilon]$) for all stored parameters and measures. Thus, the induction of the tree considers enriched information to execute different actions (expansion of a node, selection of most appropriate attribute to expand, etc.).

Therefore, concentration bounds have been already used in data stream scenarios, and one of the most extended inequality is the one proposed by Hoeffding [24].

Theorem 1 (Hoeffding's inequality). *Let $X_1, X_2, \dots, X_n$ be independent random variables such that $X_i \in [a_i, b_i]$, where $i \in \{1, \dots, n\}$. Let $\overline{X}=\frac{1}{n}\sum_{i=1}^n X_i$. Then for any $\varepsilon > 0$,*

$$\Pr\left\{\overline{X} - E\left[\,\overline{X}\,\right] \geq \varepsilon\right\} \leq e^{-2n^2\varepsilon^2/\sum_{i=1}^n(b_i-a_i)^2}$$

From this theorem and considering the average $\overline{X}$, we can estimate the error ε_δ, given a significance level of at most δ:

$$\varepsilon_\delta = \sqrt{\frac{1}{2n}\ln\frac{1}{\delta}}$$

As we can see, Hoeffding's inequality assumes only independent and bounded random variables, but none probability function is assumed. The involved statistic ($\overline{X}$) and the error bound (ε_δ) can be computed in $O(1)$ time and space computational complexity at each incoming value, what makes it applicable to learning from data streams [23]. Particularly, a corollary proposed by Hoeffding [24, page 16] can be applied to the detection of significant changes in the moving averages of streaming values.

Corollary 2. *If $X_1, \dots, X_n, Y_1, \dots, Y_m$ are independent random variables with values in the interval $[a, b]$, and if $\overline{X} = \frac{1}{n}\sum_{i=1}^n X_i$, $\overline{Y} = \frac{1}{m}\sum_{i=1}^m Y_i$, then for $\varepsilon > 0$:*

$$\Pr\left\{\overline{X}-\overline{Y}-\left(E\left[\,\overline{X}\,\right] - E\left[\,\overline{Y}\,\right]\right) \geq \varepsilon\right\} \leq e^{\frac{-2\varepsilon^2}{(n^{-1}+m^{-1})(b-a)^2}} \quad (1)$$

Analogously to Theorem 1, but considering instead the difference between averages, we can estimate the error ε_α, given a significance level of at most α:

$$\varepsilon_\alpha = (b-a)\sqrt{\frac{n^{-1}+m^{-1}}{2}\ln\frac{1}{\alpha}} \quad (2)$$

Therefore, Corollary 2 can also be used with $O(1)$ time and space computational complexity in order to detect distributional changes. From Corollary 2 we can obtain a statistical test bounding the probability of type I and type II errors. We call this test the A-test (A because it involves Averages). Let the null hypothesis $H_0 : E\left[\,\overline{X}\,\right] \leq E\left[\,\overline{Y}\,\right]$ against the alternative one $H_1 : E\left[\,\overline{X}\,\right] > E\left[\,\overline{Y}\,\right]$ and let $\overline{X}-\overline{Y} \geq \varepsilon_\alpha$ the rule to reject H_0. Next corollary bounds the probability of type I and type II errors for this statistical test.

Corollary 3 (A-test). *Under the conditions on Corollary 2, for ε_α defined in the equation (2):*

1) *if $E\left[\,\overline{X}\,\right] \leq E\left[\,\overline{Y}\,\right]$, then $\Pr\left\{\overline{X}-\overline{Y} \geq \varepsilon_\alpha\right\} \leq \alpha$ (bound for the type I error),*
2) *if $E\left[\,\overline{X}\,\right] \geq E\left[\,\overline{Y}\,\right] + \varsigma$, and $\varsigma > \varepsilon_\alpha$ then*

$$\Pr\left\{\overline{X}-\overline{Y} < \varepsilon_\alpha\right\} \leq e^{\frac{-2(\varsigma-\varepsilon_\alpha)^2}{(n^{-1}+m^{-1})(b-a)^2}}$$

(bound for the type II error).

In an obvious way we can derive a similar test for the null hypothesis $H_0 : E\left[\overline{X}\right] \geq E\left[\overline{Y}\right]$ against the alternative one $H_1 : E\left[\overline{X}\right] < E\left[\overline{Y}\right]$, being $\overline{Y} - \overline{X} \geq \varepsilon_\alpha$ the rule to reject H_0. Thus, a two-tailed test can also be obtained. This way, given a sequence of random variables $X_1, \ldots, X_n, Y_1, \ldots, Y_m$, we can detect changes in the population mean by monitoring the difference between moving averages from the A-test.

5 W-test: Bounding Weighted Moving Averages

In this section we derive a more general statistical test for weighted moving averages, although equally efficient and simple. In this case, recent incoming real values will have more weight than the older ones, assuming that they have more probability of occurrence.

McDiarmid [25] generalized the Hoeffding's inequality for dependent random variables, specifically for a martingale difference sequence, and we have the following interesting result:

Theorem 4 (McDiarmid's inequality)**.** *Let $X_1, X_2, \ldots, X_n$ be independent random variables such that $X_i \in A_i \in [a_i, b_i]$, where $i \in \{1, \ldots, n\}$. Let $g : A_1 \times \ldots \times A_n \to \mathbb{R}$ be a measurable function that satisfies the independent bounded differences condition, i.e., there exists a vector $\mathbf{d} = (d_1, \ldots, d_n)$ such that $\left|g\left(\overrightarrow{X}\right) - g(\overrightarrow{X}')\right| \leq d_i$ for all vectors $\overrightarrow{X}$ and $\overrightarrow{X}'$ that differ only at the i-th coordinate. Let $\overline{Y}$ be a random variable defined as $\overline{Y} = g(X_1, X_2, \ldots, X_n)$. Then for any $\varepsilon > 0$,*

$$Pr\left\{\overline{Y} - E\left[\overline{Y}\right] \geq \varepsilon\right\} \leq e^{-2\varepsilon^2/\sum_{i=1}^n d_i^2} \quad (3)$$

Being in mind Theorem 4, we can define $g(\overrightarrow{X})$ as a function which aims to give more weight to the most recently arrived examples. We propose monitoring a weighted sum updated at every arrived value. However, since not all examples are available in the data stream context before the processing, but arrive over time, we have defined $g(\overrightarrow{X})$ to be efficient in terms of time and space complexity. This way, we examine some weight functions whose respective weighted sums can be updated in $O(1)$ computational time for each new value.

If we assume $X_1, X_2, \ldots, X_n$ to be independent random variables the values of which are in the interval $\forall i\, A_i \in [a, b]$, additionally we take $\overrightarrow{X}$ to be a random vector defined as $\overrightarrow{X} = (X_1, X_2, \ldots, X_n)$ and consider the weights $v_i \in (0, 1]$ that can be generated by a time-decay function; then we can define the weighted moving average

$$g(\overrightarrow{X}) = \hat{X}_n = \sum_{i=1}^n v_i X_i \quad (4)$$

that satisfies the independent bounded differences condition (needed in Theorem 4): as $X_i \in [a, b]$ and $v_i \in (0, 1]$ ($i \in \{1, \ldots, n\}$), therefore $d_i = (b - a)v_i$.

In general, if on a given weighted scheme the weights v_i are not restricted in the interval $(0, 1]$ (e.g. $0 < v_i < \infty$), they can be normalized as follows.

Consider $E[X_i] = \mu$ for all $i \in \{1, \ldots, n\}$, we can present $E[g(\overrightarrow{X})]$ like

$$E\left[\sum_{i=1}^n v_i X_i\right] = \sum_{i=1}^n v_i E[X_i] = \mu \sum_{i=1}^n v_i$$

To conveniently monitor changes in the average value in a simple way, we can make $\sum_{i=1}^n v_i = 1$ and consequently $E[g(\overrightarrow{X})] = \mu$. This way, we rewrite equation (4) in a more general form:

$$\hat{X}_n = \frac{1}{\mathcal{W}_n} \sum_{i=1}^n w_i X_i \quad (5)$$

where $\mathcal{W}_n = \sum_{i=1}^n w_i$ and $v_i = w_i/\mathcal{W}_n$. Additionally, written in the following recurrent form, equation (5) can be computed in $O(1)$ time complexity at each incoming value (note that $\mathcal{W}_n = \mathcal{W}_{n-1} + w_n$; $\hat{X}_1 = X_1$):

$$\hat{X}_n = \frac{\mathcal{W}_{n-1}}{\mathcal{W}_n}\hat{X}_{n-1} + \frac{w_n}{\mathcal{W}_n} X_n.$$

Furthermore, we can compute $\mathcal{D}_n = \sum_{i=1}^n d_i^2$ on equation (3) with $O(1)$ time complexity at each incoming value ($\mathcal{D}_1 = (b - a)^2$):

$$\begin{aligned} \mathcal{D}_n &= \sum_{i=1}^n d_i^2 = \left(\frac{b-a}{\mathcal{W}_n}\right)^2 \sum_{i=1}^n w_i^2 \\ \mathcal{D}_n &= \left(\frac{\mathcal{W}_{n-1}}{\mathcal{W}_n}\right)^2 \mathcal{D}_{n-1} + (b-a)^2 \left(\frac{w_n}{\mathcal{W}_n}\right)^2. \end{aligned}$$

Logically, we can obtain a confidence interval from Theorem 4 for the weighted moving average $\hat{X}_n$ with respect to its expected value.

5.1 Bounding the Difference between Weighted Moving Averages

By following the idea shown in Section 4, we can obtain a more general statistical test with probabilistic guarantees for the type I and II errors analogously bounding the difference between two weighted moving averages. We need to use the following corollary:

Corollary 5. *If $X_1, X_2, \ldots, X_n, Y_1, Y_2, \ldots, Y_m$ are independent random variables bounded in the interval $[a, b]$, and if $\hat{X}_n = \sum_{i=1}^n v_i X_i$, $\hat{Y}_m = \sum_{i=1}^m v_i' Y_i$, such that $\hat{X}_n$ and $\hat{Y}_n$ satisfy the independent bounded differences condition, then for $\varepsilon > 0$:*

$$\Pr\left\{\hat{X}_n - \hat{Y}_m - \left(E\left[\hat{X}_n\right] - E\left[\hat{Y}_m\right]\right) \geq \varepsilon\right\} \leq e^{-\frac{2\varepsilon^2}{\mathcal{D}_{n,m}}} \quad (6)$$

where $\mathcal{D}_{n,m} = (b-a)^2\left[\sum_{i=1}^n (v_i)^2 + \sum_{i=1}^m (v_i')^2\right]$.

Consider now the null hypothesis $H_0 : E[\hat{X}_n] \leq E[\hat{Y}_n]$ against $H_1 : E[\hat{X}_n] > E[\hat{Y}_n]$, and let $\hat{X}_n - \hat{Y}_n > \hat{\varepsilon}_\alpha$ the rule to reject H_0, where α is the significance level and ε_α is the error bound defined as

$$\hat{\varepsilon}_\alpha = \sqrt{\frac{\mathcal{D}_{n,m}}{2} \ln \frac{1}{\alpha}} \tag{7}$$

Obviously, equation (6) is a generalization of equation (1) and the last statistical test is also a generalization of the A-test presented in Section 4. Similarly, bounds for the probability of the type I and type II errors can also be guaranteed. We name this test W-test (W because it involves Weighted averages).

Corollary 6 (W-test)**.** *Under the conditions on Corollary 5, for $\hat{\varepsilon}_\alpha$ defined in the equation (7):*

1) *if $E\left[\hat{X}_n\right] \leq E\left[\hat{Y}_m\right]$, then $\Pr\left\{\hat{X}_n - \hat{Y}_m \geq \hat{\varepsilon}_\alpha\right\} \leq \alpha$ (bound for the type I error),*
2) *if $E\left[\hat{X}_n\right] \geq E\left[\hat{Y}_m\right] + \varsigma$, and $\varsigma > \varepsilon_\alpha$ then*

$$\Pr\left\{\hat{X}_n - \hat{Y}_m < \hat{\varepsilon}_\alpha\right\} \leq e^{\frac{-2(\varsigma - \varepsilon_\alpha)^2}{\mathcal{D}_{n,m}}}$$

(bound for the type II error).

Proof: The proof is analogous to that of Corollary 3. □

5.2 Weighting Schemes Compatible with the W-Test

Let us show a motivating example where we obtain a confidence interval from Theorem 4 for the EWMA statistic $\hat{X}_n = (1 - \lambda)\hat{X}_{n-1} + \lambda X_t$ with respect to its expected value. To unify notations, we equivalently define $\hat{X}_n$ for $n > 0$ and we take $\hat{X}_1 = X_1$.

Example 7. When $n \to \infty$, if $X_1, X_2, \ldots, X_n$ are random variables bounded in the interval $[a, b]$, and $\hat{X}_n$ is the EWMA statistic; then

$$\Pr\left\{\hat{X}_n - E\left[\hat{X}_n\right] \geq \varepsilon\right\} \leq e^{-\frac{2\varepsilon^2(2-\lambda)}{\lambda(b-a)^2}} \tag{8}$$

It is easy to show that on the EWMA statistic (see Example 7)

$$\lim_{n,m \to \infty} \hat{\varepsilon}_\alpha = (b - a)\sqrt{\frac{\lambda}{\lambda + 2} \ln \frac{1}{\alpha}} \tag{9}$$

In the statistical process control field, we analogously think about $\hat{\varepsilon}_\alpha$ as a control limit. For large n and the EWMA statistic, if the random variables follow a normal distribution (in this case we denote $\hat{\varepsilon}_\alpha$ as $\hat{\varepsilon}_\mathcal{N}$), $\hat{\varepsilon}_\mathcal{N} = L\sigma\sqrt{\lambda/(\lambda + 2)}$ is a classical control limit [8], where σ is the standard deviation and L is a multiplicative factor to give performance related to both type I and type II errors. Then, we can show the asymptotic relationship between α on the W-test and L bearing in mind the equation (9), precisely $\alpha = \exp^{-1}(L^2\sigma^2/(b - a))$.

For example, assuming normality, a typical value $L = 3$ (*three-sigma control limits* [8]) makes the probability of the type I error approximately equal to 0.001. However, considering random variables bounded in the interval $[0, 1]$ and the worst-case variance $\sigma^2 = 1/4$; we can easily check that for large n, setting over this configuration $L = 3$ is similar to configuring $\alpha = 0.11$ (i.e. the bound for the probability of type I error) on the W-test.

On the other hand, another scheme that has been used previously is to weight past values according to an exponential aging function, that is to say $w_i = \beta^{n-i}$ $(0 < \beta \leq 1)$. For instance $\beta = exp(-\eta)$ [17], [22]. We name the corresponding statistic *exponential aging*.

Example 8. When $n \to \infty$, if $X_1, X_2, \ldots, X_n$ are random variables bounded in the interval $[a, b]$, and $\hat{X}_n$ is the exponential aging statistic; then

$$\Pr\left\{\hat{X}_n - E\left[\hat{X}_n\right] \geq \varepsilon\right\} \leq e^{-\frac{2\varepsilon^2(1+\beta)}{(1-\beta)(b-a)^2}}$$

Consider now the configuration in Example 7 and a moving average $\overline{X}$ computed over the most recent m random variables. From the equation (8) and the Hoeffding's inequality, we can say that $\Pr(\overline{X} - E[\overline{X}] > \varepsilon)$ and $\Pr(\hat{X} - E[\hat{X}_n] > \varepsilon)$ are bounded identically selecting λ on the EWMA statistic $\hat{X}_n$ as $\lambda = 2/(m + 1)$. Analogously, for the exponential aging statistic on the Example 8 we can fix $\beta = (m - 1)/(m + 1)$. EWMA statistic and the exponential aging statistic are also related asymptotically by $\lambda = 1 - \beta$.

Polynomial weighted schemes have been less studied in the literature. Anyway, the aforementioned formulation of weighted moving averages is also feasible for weightings polynomial weighting. An elegant solution is to define $w_i = i^p$. For instance, for convenient small values of $p \in \mathbb{Z}^+$, the sums $\mathcal{W}_n$ and $\mathcal{D}_n$ can be represented in a closed form expression using the *Faulhaber formula*. This way, it can be unnecessary to maintain these sums updated since for large n, even for small p they can grow quickly.

5.3 Relationship between the A-Test and the W-Test

Let us now compare the A-test and the W-test supposing that we know exactly where the change point is located. Consider the scenario with an abrupt change at n random variables (i.e. all X_i are *i.i.d.*, there is a change in the mean value just after X_n arrives, and again all Y_j are *i.i.d.*). Informally, we can expect that $\hat{X}_n \approx \overline{X}$ and $\hat{Y}_n \approx \overline{Y}$ since in this case $E[\hat{X}_n] = E[\overline{X}]$ and $E[\hat{Y}_n] = E[\overline{Y}]$. However, $\varepsilon_\alpha \leq \hat{\varepsilon}_\alpha$ (ε_α was defined in equation 2) since $\mathcal{D}_{n,m}$ has a minimum if all $v_i = 1/n$ and all $v'_i = 1/m$. Consequently, in abrupt changes the A-test must detect the change more quickly than the W-test involving weighted moving averages.

On the other hand, often in gradual changes it is the case that not all incoming values have the same importance, but rather that the importance can decrease over time. Weighting can be viewed as changing the probability distribution $P(X_i)$ in order to place higher weight on values with a greater probability of occurrence [17]. In this setting, using estimators involving weighted moving averages can be more appropriate. Nevertheless, an additional problem emerges related with the weighting estimation. As we have discussed, some research studies fix some tuning parameter with an extensive empirical evaluation [21], other weights have been adapted with respect to the actual performance of the learner and/or the age of the input values [17], and other weights have also been computed to be optimal, assuming an underlying structure of the change [14]. In any case, the W-test can be straightforwardly applied to many weighted schemes, principally due to the general formulation of the weighted moving averages $\hat{X}_n$ and $\hat{Y}_n$, as well as to the probability inequality given in the Corollary 5.

Finally, as both the A-test and the W-test can be performed with constant time and space complexity, all that is left to do is to identify a relevant cut point in the sequence of values in order to detect significant changes online. Again, a very simple but effective method is used.

6 The Algorithm to Detect Changes Online

A very extended method to detect the occurrence of a change is monitoring the evolution of some indicators or measures. Many options can be considered for this monitoring, and one approach that fits in very well with the use of interval estimations is based on the idea of statistical process control [3], [8]. In this approach two different levels, warning and drift, are defined on the basis of the probability density function of a normal distribution. Thus, the method tries to detect when a statistic (p) computed from the last observed values is statistically far from the expected one (μ), specifically it sets the warning level at 95% ($\Pr\{\mu - 2\sigma \le p \le \mu + 2\sigma\} \approx 0.95$, where σ is the standard deviation) and the drift level at 99% ($\Pr\{\mu - 3\sigma \le p \le \mu + 3\sigma\} \approx 0.99$).

We can adapt this idea, relaxing the assumption of normality, and substituting the estimation of the interval that uses the standard deviation (σ) with another one that fixes the desired significance level (α) and estimates the confidence interval (by means of ε_α or $\hat{\varepsilon}_\alpha$). In our case, we can define two different confidence values, one corresponding to the warning level (α_W) and another to the drift level (α_D). In Figure 1 we present a simple method to differentiate between three separate states: $STABLE$, when there seems to be no change; $WARNING$, when it seems that a possible concept drift may appear; and $DRIFT$, when the drift is clearly identified. The information given by the method in the variable $STATE$ can be used in many ways and our proposal does not limit the actions to be executed when warning or drift states are detected. However, one of the most direct usages is the following: *a)* if the warning level is exceeded a possible drift will arrive and, consequently, new observed examples can be buffered and used to train an alternative classifier; *b)* when a drift signal is triggered a hypothetical alternative classifier could replace the old one to adapt the learner using the buffered examples.

```
Require: $x_1, x_2, ...$: potentially not-ended stream of real
  values where $\forall i,\ x_i \in [a, b]$
Require: $\alpha_W \in (0, 1]$: confidence for the warning level
Require: $\alpha_D \in (0, 1]$: confidence for the drift level
Ensure: $STATE \in \{STABLE, WARNING, DRIFT\}$
  /* Variables declaration, at n incoming real values: */
  $\hat{X}_{cut}$: statistic computed from $x_1, x_2, ..., x_{cut}$
  $\hat{Y}_{n-cut}$: statistic computed from $x_{cut+1}, ..., x_n$
  $\hat{Z}_n$: statistic computed from $x_1, x_2, ..., x_n$
  $\varepsilon_{\hat{X}_{cut}}$, $\varepsilon_{\hat{Y}_{n-cut}}$ and $\varepsilon_{\hat{Z}_n}$: error bounds in correspondence
  with the statistic used
  init()      /* variables are reset */
  for all $x_i$ arrival on stream $x_1, x_2, ..., x_i, ...$ do
    /* update statistics and confidence intervals */
    update $\hat{Y}_{i-cut}$ , $\hat{Z}_i$, $\varepsilon_{\hat{Y}_{i-cut}}$, and $\varepsilon_{\hat{Z}_i}$ having a new real
    value $x_i$
    /* update the cut point */
    if $\hat{Z}_i + \varepsilon_{\hat{Z}_i} \le \hat{X}_i + \varepsilon_{\hat{X}_i}$ then
      $\hat{X}_{cut} = \hat{Z}_i$ and $\varepsilon_{\hat{X}_{cut}} = \varepsilon_{\hat{Z}_i}$
      reset $\hat{Y}_{i-cut}$ and $\varepsilon_{\hat{Y}_{i-cut}}$
    end if
    /* determine the current state of the data stream */
    if $H_0 : E[\hat{X}_{cut}] \ge E[\hat{Y}_{i-cut}]$ is rejected with size $\alpha_D$
    then
      $STATE \leftarrow DRIFT$
      init()
    else
      if $H_0 : E[\hat{X}_{cut}] \ge E[\hat{Y}_{i-cut}]$ is rejected with size
      $\alpha_W$ then
        $STATE \leftarrow WARNING$
      else
        $STATE \leftarrow STABLE$
      end if
    end if
  end for
```

Figure 1. HDDM: Drift Detection Method based on the Hoeffding's inequality.

Consider a sequence of random variables $X_1, X_2, \ldots, X_{cut}, X_{cut+1}, \ldots, X_n$ and the problem of detecting a significant increment in the mean value of this sequence (e.g. an increment of the error rate of a learning algorithm). The first task is to estimate a relevant cut point in this sequence, that we called cut, in order to later carry out either the A-test or the W-test over the samples $X_1, X_2, \ldots, X_{cut}$ and $X_{cut+1}, \ldots, X_n$. Consider the expression $\hat{X}_i + \varepsilon_{\hat{X}_i}$ $(1 \le i \le n)$, where $\hat{X}_i$ can be either a moving average or a weighted moving average, and $\varepsilon_{\hat{X}_i}$ is its corresponding error bound computed from Theorem 1 or from Theorem 4. When there is no change in the population mean, $\hat{X}_i$ must keep approximately constant and consequently $\hat{X}_i + \varepsilon_{\hat{X}_i}$ must diminish its value (in the case of A-test) or keep approximately constant (for the W-test). However, at a given increment in the population mean, the value of the mean estimator $\hat{X}_i$ and as consequence $\hat{X}_i + \varepsilon_{\hat{X}_i}$ must increase. Following this idea, a relevant cut point in this sequence can be estimated from the minimum value of $\hat{X}_i + \varepsilon_{\hat{X}_i}$ $(1 \le i \le n)$ [3], [4], [8].

Then, we can apply the respective statistical test (A-test or W-test) to estimate the actual state ($STABLE$, $WARNING$ or $DRIFT$) of the change detector from α_W and α_D over the samples $X_1, X_2, \ldots, X_{cut}$ and $X_{cut+1}, \ldots, X_n$. This way, if the null hypothesis is rejected with size α_W the current status is set to $WARNING$. Similarly, if the null hypothesis is rejected with size α_D, the change detector reaches the $DRIFT$ level and all counters are reset. Otherwise the null hypothesis is accepted and the current status is set to $STABLE$. We have called this online change detector HDDM because it is similar to DDM but uses instead the Hoeffing's inequality for the two-sample statistical test.

To perform the statistical test, Figure 1 maintains three counters ($\hat{X}_{cut}$, $\hat{Y}_{n-cut+1}$ and $\hat{Z}_n$). For the A-test, this method can be slightly improved by applying the following corollary (equivalent to Corollary 2) derived from the Hoeffding's inequality. This way, only two counters (e.g. $\hat{X}_{cut}$ and $\hat{Z}_n$) are necessary.

Corollary 9. *If* $X_1, \ldots, X_n, X_{n+1}, \ldots, X_{n+m}$ *are independent random variables with values in the interval* $[a, b]$, *and if* $\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$, $\overline{Z} = \frac{1}{n+m}\sum_{i=1}^{n+m} X_i$, *then for* $\varepsilon > 0$*:*

$$\Pr\left\{\overline{X}-\overline{Z}-\left(E\left[\overline{X}\right]-E\left[\overline{Z}\right]\right)\geq\varepsilon\right\}\leq e^{-\frac{2\epsilon^2 n(n+m)}{m(b-a)^2}} \qquad (10)$$

In this case, the rule to reject the null hypothesis $H_0 : E\left[\overline{X}\right] \le E\left[\overline{Z}\right]$ against the alternative one $H_1 : E\left[\overline{X}\right] > E\left[\overline{Z}\right]$ will be $\overline{X}-\overline{Z} \geq \varepsilon_\alpha$, where

$$\varepsilon_\alpha = (b-a)\sqrt{\frac{m}{2n(n+m)}\ln\frac{1}{\alpha}}$$

Generally speaking, both the A-test and W-test can be applied to other window techniques in order to consider many cut points [5], [7]. In this case, a *Bonferroni correction* for α can correct multiple tests maintaining bounded the probability of false detection [5]. Specifically, the A-test can easily be extended to the ADWIN2 algorithm, the resulting bound being tighter than the one given by Bifet and Gavaldà [5] in a constant factor and under the same theoretical assumptions.

Furthermore, HDDM in combination with W-test gives place to a more simple method than ECDD [10] to detect change by means of the EWMA statistic (e.g., no "look-up table" is needed nor additional method to estimate confidence intervals).

Afterward we empirically study some particular configurations of these online change detectors taken into account different types of changes and learning algorithms.[1]

7 Empirical Study

Some authors have proposed various performance measures to be considered in the design and evaluation of change detection algorithms [5], [8], [26], [27]. We focus on evaluating the false positives rate (probability of false detection), false negatives rate (probability of non-detection) and mean delay for detection as they are the most considered in incremental learning scenarios [26].

On the other hand, the prequential method is a general methodology for evaluating incremental learning algorithms. Gama *et al.* [27] propose a framework for evaluating the quality of streaming learning algorithms. They defend the use of a predictive sequential error estimated over a sliding window to calculate the performance of algorithms that learn from open-ended data streams in non-stationary environments. At the same time, they present strategies which apply to both supervised and unsupervised problems, given that a proper loss function is defined; and study properties and statistical tests to comparatively assess algorithms' performance.

This way, to evaluate precision in artificial data streams, we periodically test (every 100 training examples) the learning algorithm with other 100 examples only used for testing, taking advantage of stream generators since they can produce a possibly infinite number of examples. Obviously, this strategy is not so useful for real-world datasets, so in this case we use a test-then-train approach [27], [28]. We do not focus on time and space measurements since all the proposed methods have $O(1)$ computational time complexity. Although they do not store examples, the necessary information is kept in a fixed number of counters.

The methods considered in this experimental study are HDDM in combination with the A-test and the W-test, and some others that follow similar strategies and characteristics. Concretely we use a moving average (by means of the A-test) and the EWMA statistic (applying the W-test), both configured with $\alpha_W = 0.005$ and $\alpha_D = 0.001$. We have considered the algorithm DDM [3], because it uses a similar approach and it shows a good performance; it detects abrupt and gradual changes when the change is not very slow, but it does not perform as equally well if the contrary happens [4]. We also include ADWIN2 [5], although it does not process the input performance values with constant time and space complexity, it has a very good performance and strong theoretical guarantees; furthermore, ADWIN2 has been broadly used as a change detector in many learning algorithms. We also incorporate ECDD [10], which is a concept drift detector based on EWMA.

1. The source code (in the Java programming language) of the proposed change detectors, as well as more detailed data of the experiment results are available online at http://www.lcc.uma.es/~jcampo/software/HDDM.

Previously we have shown the relationship between the W-test using the EWMA statistic and the control limit widely studied in the literature assuming that random variables are normally distributed. Likewise, we have established the relationship between the EWMA statistic and another common weighting scheme that we call exponential aging. We have also shown the asymptotic similarity between the EWMA statistic and a moving average over the last m real values (e.g. a sliding window of fixed size). Thus, in the empirical study we do not consider some of these basic algorithms.

7.1 Performance over Streaming Bits

We generate streams of bits from a Bernoulli distribution with parameter μ making 30 drifts separated by 100 and $1\,000$ bits (i.e. the length of stable concepts, see Table 1). We run each algorithm 30 times for every one of these situations. So we assume that in a run, a change detector must raise 30 drift signals and must not raise more than one drift signal per stable mean. This way we can estimate the false positives and false negatives rate. We also distinguish types of change controlling the random difference for mean values: the absolute value of the difference between old mean and the new one restricted in an interval $[a, b]$. This is reflected in Table 1, separating the performance of each algorithm in different columns. For each algorithm we estimate the false positives rate (FP), the false negatives rate (FN) and the average of retardation for the detection of changes (DELAY, that measures for example, how fast the algorithm detects changes). Table 1 shows the averaged results of these 30 runs. Due to the difficulty in estimating these performance measures on gradual changes, we do not consider this type of change in streaming bits.

In this experiment we study different configurations of λ in the EWMA statistic using the W-test on HDDM (this combination is represented as HDDM$_{W\text{-test}(\lambda)}$), the A-test (represented as HDDM$_{A\text{-test}}$), DDM, ADWIN2 with the bound that involves approximation to the normal curve for large sample size, and ECDD. On HDDM$_{W\text{-test}(\lambda)}$, HDDM$_{A\text{-test}}$ and ADWIN2 we fix the size of the test (significance level) to $\alpha = 0.001$. In ECDD, we fix $\lambda = 0.2$ and $ARL_0 = 100$ (which controls the false positives rate).[2]

As we assume bounded random variables, values of λ near to 1 in HDDM$_{W\text{-test}(\lambda)}$ are useless. We can check this observation in equation (9). For example, since $X_i \in [a, b]$, at least $\lim_{n,m\to\infty} \hat{\varepsilon}_\alpha \le b - a$, what leads to $\lambda \le 2/(\ln(1/\alpha) - 1)$. In all the algorithms the size was set to $\alpha = 0.001$ and consequently in HDDM$_{W\text{-test}(\lambda)}$ useful values are $\lambda \le 0.33$. Even so, we have found empirically that configurations for $\lambda \le 0.25$ generally outperforms the variant for $\lambda > 0.25$. This way, we study some settings for $\lambda \le 0.25$.

2. A discussion of the setting of ECDD is given by Ross *et al.* [10]

Similarly, we can explain the defective false negatives rate of HDDM$_{W\text{-test}(\lambda)}$ (see Table 1) on changes with less scale (e.g. $|\mu_{old} - \mu_{new}| \in [0.1, 0.3]$). HDDM$_{W\text{-test}(\lambda)}$ does not detect such changes since the confidence interval for this statistic does not tend to zero as the other change detectors do. However, HDDM$_{W\text{-test}(\lambda)}$ has a low false positives rate in all the configurations for the parameter λ, and a very good behavior when the scale of the change is large.

According to additional experiments, in the EWMA statistic we note that configurations for $\lambda \le 0.1$ have a very good performance in many situations; variants for $0.1 < \lambda \le 0.2$ do not perform so well. In general, configurations for $\lambda \le 0.2$ outperformed the variants for $\lambda > 0.2$.

All proposed methods maintained the ratio of false positives and false negatives according to the theory. When the mean value varies quite frequently (100 bits per stable mean value), detecting slight changes is difficult as confidence intervals are not fit enough. However, when concepts remain constant for a longer time these rates are in fact much smaller than α.

Note that the type of abrupt change considered in this experiment favors the A-test with respect to the W-test (see Subsection 5.3). Table 1 shows that often HDDM$_{A\text{-test}}$ performs better than DDM in the three performance measures, this difference is even more notable in the retardation to detect changes at more bits per concept. Note that ECDD maintains approximately constant the false positives and false negatives rate, what indicates robustness of the method.

Additionally, HDDM$_{A\text{-test}}$ also outperformed ADWIN2 in many cases. In ADWIN2, where multiple cut points are considered, the retardation in the change detection is not as susceptible to the length of the stable concepts as when considering a single cut point. Aside from the highest processing time, to maintain bounded the probability of false positive, the extra logarithmic term $\ln(n)$, where n is the number of cut points, can make the bound much more conservative than HDDM$_{A\text{-test}}$ when n is large. This way, when change detectors that consider a single cut point (e.g. HDDM$_{A\text{-test}}$) are able to estimate precisely where the change point is located, they can detect the corresponding concept drift more quickly because the computed bound is less conservative.

In supplementary experiments (omitted for space reasons) we note that methods also perform according to Table 1 when concepts remain constant for a much longer time (e.g. $100\,000$ bits per stable mean). However, ADWIN2 outperforms all the change detectors when concepts remained constant for a longer period of time (especially considering the DELAY measure). However, its computational cost was much more expensive. As HDDM$_{A\text{-test}}$ and DDM use the average to estimate the change point, older values have the same weight as the new ones. Then, when concepts remain stable for a large number of bits there is a notable difference between

Algorithm		100 bits between changes				1 000 bits between changes			
		[0.1, 0.3]	[0.3, 0.5]	[0.5, 0.7]	[0.7, 0.9]	[0.1, 0.3]	[0.3, 0.5]	[0.5, 0.7]	[0.7, 0.9]
HDDM$_{W\text{-test}(0.025)}$	FP	0	0	0	0	1.48E-04	1.87E-04	9.78E-05	1.44E-05
	FN	0.67	0.49	0.33	0.09	0.088	0.018	0.004	0.002
	DELAY	16.76	30.22	36.04	40.24	105.62	34.52	19.65	15.09
HDDM$_{W\text{-test}(0.050)}$	FP	0	0	0	0	3.11E-04	3.89E-04	1.86E-04	2.33E-05
	FN	0.62	0.16	0	0	0.149	0.023	0.002	0
	DELAY	16.05	28.98	19.80	14.68	110.68	29.64	14.86	11.71
HDDM$_{W\text{-test}(0.150)}$	FP	2.22E-05	2.22E-05	1.11E-05	0	2.62E-04	3.24E-04	1.37E-04	7.78E-06
	FN	0.75	0.351	0.023	0	0.411	0.076	0.003	0
	DELAY	9.99	20.04	16.963	10.71	110.44	56.28	16.95	11.03
HDDM$_{A\text{-test}}$	FP	0	3.33E-05	1.11E-05	1.11E-05	4.33E-05	4.00E-05	3.22E-05	1.22E-05
	FN	0.69	0.138	0	0.001	0.022	0.001	0	0
	DELAY	15.27	28.67	14.05	8.28	157.78	40.38	22.02	13.24
DDM	FP	2.33E-04	0.001	0.001	0.003	3.08E-04	4.53E-04	6.20E-04	0.001
	FN	0.56	0.088	0.037	0.036	0.104	0.003	0.001	0.004
	DELAY	18.68	24.20	12.86	7.94	208.67	109.95	57.34	26.80
ADWIN2	FP	0.004	0.006	0.008	0.023	0.004	0.007	0.008	0.009
	FN	0.70	0.67	0.25	0	0.05	0	0	0
	DELAY	12.27	14.91	46.66	32.30	209.59	56.38	28.26	16.10
ECDD	FP	0.04	0.05	0.05	0.06	0.04	0.05	0.05	0.06
	FN	0.6	0.49	0.53	0.20	0.6	0.45	0.47	0.16
	DELAY	6.70	11.81	11.64	26.12	23.69	95.61	114.11	260.76

Table 1
Mean value of the methods' performances over streaming bits.

the estimated change point and the real one; leading to a considerable retardation in the detection of changes. This way, a weighted scheme used only to estimate this change point should improve the performance of HDDM$_{A\text{-test}}$ on large stable concepts.

Generally speaking, for lesser values of λ the size of the stable concepts is not so influential on HDDM$_{W\text{-test}(\lambda)}$ as in HDDM$_{A\text{-test}}$ and DDM. Thus, the difference in the retardation to detect changes between ADWIN2 and HDDM$_{W\text{-test}(\lambda)}$ is similar for changes with large mean scale. At the same time, this behavior implies that the warning level on HDDM$_{A\text{-test}}$ and DDM should be more effective than on ADWIN2, HDDM$_{W\text{-test}(\lambda)}$ with λ close to zero, and ECDD.

7.2 Artificial Data Streams

This experiment is implemented over MOA [28], a framework for the incremental learning. It provides a collection of evaluation tools, a great variety of algorithms inherent to incremental learning and several methods to generate artificial data streams with the possibility of including concept changes.

We evaluate change detectors with two different incremental learning algorithms [28]: a Naïve Bayes predictor (NB) and a Perceptron (P). Thus, when an example arrives, it is classified so that the error rate can be monitored. When a warning alert is raised an alternative classifier is trained until a stable concept estimation returns. In the case of drift trigger, the alternative replaces the previous one. It is possible to use a more sophisticated strategy in order to handle concept drift in similar conditions (e.g. see Bach and Maloof [29]), however, we again choose the simplest one.

As we have shown in the previous section, on the HDDM$_{W\text{-test}(\lambda)}$ change detector, all the studied configurations for $\lambda \leq 0.1$ (as well as for $0.1 < \lambda \leq 0.2$) had a similar performance. For this reason in this experiment, we only study two variants for the HDDM$_{W\text{-test}(\lambda)}$ change detector: $\lambda = 0.05$ and $\lambda = 0.15$. Obviously, we do not claim that any of these configurations is optimal so it may be possible to improve the accuracy of all the methods. However, since it is not useful in practice we use the above configurations, which perform very well in many circumstances. Considering the change detectors (HDDM$_{W\text{-test}(0.05)}$, HDDM$_{W\text{-test}(0.15)}$, HDDM$_{A\text{-test}}$, DDM, ADWIN2 and ECDD) and the incremental predictors (NB and P) we have tested 12 algorithms combining changes detectors and learning algorithms. Additionally, we have also included the incremental learning algorithms without any method to detect changes.

To be precise, we run the algorithms over three drift generators (LED, STAGGER and AGRAWAL) implemented in MOA [28]. In LED, the goal is to predict the digit displayed on a seven-segment LED display, where each attribute has a 10% chance of being inverted. The particular configuration of the generator used for the experiment produces 24 binary attributes, 17 of which are irrelevant. Drift is simulated by interchanging relevant attributes.

STAGGER concepts are boolean functions of three attributes encoding objects. We carry out drift choosing between the three original target functions.

AGRAWAL generates one of ten different predefined functions. The generator produces a stream containing nine attributes, six numeric and three categorical ones. There are ten functions defined for generating binary class labels from the attributes. Drift is simulated changing the function that classifies the examples.

We measure the performance of the learners under abrupt and gradual types of change. To simulate gradual changes we use the sigmoid function incrementing the probability that at each time the new examples belong

Algorithm			100 examples per concept						1 000 examples per concept					
			Abrupt ($t = 0$)			Gradual ($t = 50$)			Abrupt ($t = 0$)			Gradual ($t = 500$)		
			LED	STA	AGR	LED	STA	AGR	LED	STA	AGR	LED	STA	AGR
HDDM$_{W\text{-test}(0.05)}$	NB	$\overline{x}$	**63.35**	**98.76**	77.36	**63.39**	85.46	**73.76**	**78.75**	**99.83**	**82.07**	70.55	**90.76**	**77.25**
		s	**7.77**	**0.22**	4.46	**5.34**	6.05	**4.36**	**1.44**	**0.11**	**9.48**	3.03	**2.48**	**7.92**
	P	$\overline{x}$	40.14	85.02	*61.03*	*41.21*	77.03	*62.01*	64.48	88.76	57.79	56.77	82.66	58.04
		s	2.69	2.77	*3.72*	*2.22*	3.23	*4.49*	2.77	9.05	16.83	4.30	7.88	15.89
HDDM$_{W\text{-test}(0.15)}$	NB	$\overline{x}$	58.85	94.44	**80.19**	61.41	82.48	72.89	78.62	98.13	80.94	**70.60**	88.84	74.64
		s	8.51	2.13	**2.62**	6.57	3.27	4.83	1.87	3.05	10.27	**2.95**	2.64	7.28
	P	$\overline{x}$	*41.01*	84.49	*61.07*	40.03	*78.26*	58.77	64.14	88.47	60.59	55.65	82.59	58.12
		s	*1.70*	2.59	*4.44*	1.46	*3.65*	4.22	3.74	9.33	16.04	4.39	6.73	15.92
HDDM$_{A\text{-test}}$	NB	$\overline{x}$	56.80	98.38	68.98	55.31	**87.62**	69.58	**78.73**	99.81	81.29	70.40	90.44	74.16
		s	10.39	0.69	6.95	8.64	**4.51**	5.61	**1.47**	0.15	9.81	3.05	3.50	8.28
	P	$\overline{x}$	40.68	85.23	57.48	39.80	77.10	55.12	*64.71*	*88.83*	*62.89*	57.07	82.68	58.51
		s	2.83	2.92	4.11	1.52	3.35	4.04	*3.07*	*9.00*	*15.90*	3.78	7.51	14.65
DDM	NB	$\overline{x}$	54.49	**98.73**	67.76	47.89	86.75	67.59	78.04	**99.83**	68.76	70.14	90.40	66.74
		s	10.54	**0.20**	7.69	12.49	5.00	6.20	2.11	**0.14**	14.81	3.11	3.50	12.64
	P	$\overline{x}$	*41.20*	85.58	60.09	39.91	*78.32*	57.11	63.73	88.15	59.53	*57.83*	*82.76*	60.04
		s	*2.68*	2.81	4.65	1.69	*3.61*	6.13	2.84	8.58	16.84	*3.70*	*7.41*	15.30
ADWIN2	NB	$\overline{x}$	51.74	76.37	66.92	49.71	77.07	67.08	76.99	**99.83**	76.74	69.60	90.63	69.58
		s	12.19	8.85	6.78	11.53	6.99	5.51	2.75	**0.13**	10.78	3.22	2.94	8.82
	P	$\overline{x}$	40.36	85.66	55.29	40.48	77.86	55.75	61.90	*88.84*	61.30	51.32	82.56	*60.41*
		s	1.76	2.88	5.82	1.35	4.02	5.95	3.23	*8.99*	16.49	5.60	7.49	*14.17*
ECDD	NB	$\overline{x}$	48.85	98.54	74.67	47.89	80.49	64.76	70.88	99.85	70.58	66.46	88.37	67.42
		s	13.91	0.36	4.30	12.49	8.29	6.06	3.95	0.11	14.87	4.67	3.63	11.74
	P	$\overline{x}$	40.56	82.15	49.60	40.26	68.32	44.47	51.59	84.91	50.39	44.97	76.56	48.85
		s	2.14	3.36	7.46	1.39	5.77	4.56	11.60	12.55	15.90	12.12	10.26	15.52
No drift detector	NB	$\overline{x}$	48.85	72.93	66.92	47.89	72.42	67.08	41.22	69.12	61.44	40.39	68.83	61.90
		s	13.91	5.94	6.78	12.49	5.25	5.51	16.06	12.08	11.82	15.15	11.64	12.01
	P	$\overline{x}$	40.48	85.66	58.98	40.28	77.70	41.67	44.89	85.37	59.50	44.34	82.33	59.73
		s	1.47	2.88	4.00	1.50	4.13	5.36	15.05	8.84	16.97	12.23	6.91	16.22

Table 2

Experiments with 100 and $1\,000$ examples per concept; 50 and 500 examples in the transition period between concepts on gradual changes ($t = 50$ and $t = 500$). Significant differences with respect to the remaining methods' accuracy are showed in bold for the Naïve Bayes classifier as well as underlined and in italics for the Perceptron.

to the new concept [28]. On gradual changes, we set the transition period between consecutive concepts to 50 and 500 (100 and $1\,000$ examples per concept respectively). There were 30 drifts for each experiment. Each algorithm was evaluated every 100 examples with another 100 examples (only used for testing). Table 2 shows the means and standard deviations of these accuracy measures in one run for each configuration. Naturally, to evaluate the performance of change detectors we do not compare accuracy between different classifiers.

As we expected, the classifier without change detector rarely outclasses any of the methods for detecting drift. Naïve Bayes almost always outperforms Perceptron, so the optimal classification boundary appears to be non-linear in all cases.

Concepts that remain little time are harder to track, classifiers have not much information to learn the current concept and thus the error rate is not so affected at a concept drift. Furthermore, confidence intervals may not be tight enough to detect small changes. If such changes are not detected, classifiers learn a unified concept and its accuracy often drops significantly. In these cases the ability of the classifier to learn quickly is more influential. Table 2 reflects this setting when changes occur every 100 instances. Change detectors based on EWMA, which give more weight to recent values, perform better in these situations (particularly HDDM$_{W\text{-test}(\lambda)}$, whose confidence interval also converge rapidly to its limit, see Example 7). It is important to detect changes as soon as possible when concepts are short because there is little time to learn. The Naïve Bayes classifier usually learns the selected concepts quicker than the Perceptron. In general, EWMA change detectors in combination with the Naïve Bayes classifier outperform the other methods for the above reasons. This difference is not so appreciable when learning with the Perceptron (which does not learn as quickly). Even so, HDDM$_{W\text{-test}(\lambda)}$ never detected all the concept drifts (30 changes) when concepts remained for 100 instances (we do not show this measurements for space reasons). We noted that ECDD had a high false positives rate in all the experiments.

The proposed algorithms also perform well in gradual changes. Note that gradual changes on 100 instances per concept is hard to track, even so all change detectors in combination with Naïve Bayes often improve the performance of the classifier without change detection. However, in 100 instances per concept, methods do not perform so well in combination with the Perpectron, and an alternative strategy may be necessary.

When concepts remained constant for a longer period of time ($1\,000$ instances per concept), the proposed change detectors based on EWMA also performed well, especially HDDM$_{W\text{-test}(0.05)}$. In correspondence with the theory (Section 5) and with the streaming bits experiments, all the proposed methods had a small false positives and false negatives rate, as well as a short delay detection of changes. We also observed that HDDM$_{W\text{-test}(\lambda)}$, HDDM$_{A\text{-test}}$ and DDM approximately detected the same amount of changes, so we estimate that the difference in the prediction accuracy was determined by the delayed detection of such changes (see Table 1).

False positives do not affect the accuracy so much when classifiers can learn the underlying concept fast (even, the accuracy can be improved with false detections). However, Perceptron does not learn the underlying concept so rapidly, causing drift signals to have the additional cost related with slower learning. For example, when concepts are stable for a longer time, change detectors that monitor averages frequently outperform EWMA change detectors in combination with the Perceptron, because EWMA change detectors are more susceptible to noise.

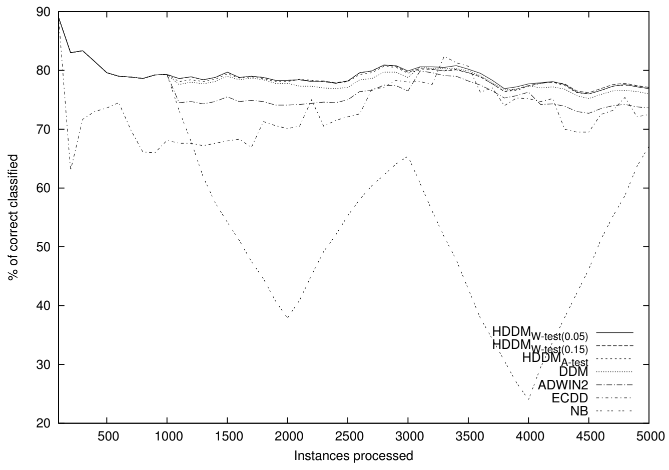

Figure 2. A typical behavior of the methods under abrupt change (LED concepts). Abrupt changes occur every $1\,000$ examples.

Figure 2 represents typical learning curves when change detectors in combination with NB learn LED concepts under abrupt changes. We can observe how the NB accuracy drops when there is not a change detection mechanism: after 1000 examples occurs the first change and the accuracy begins to deteriorate (the model induced for the first concept does not correspond with the second concept), and the performance varies with every new change (the degree of variability mainly depends on the similarity between the current concept and the model previously induced). This drop in the accuracy is not so appreciable for the proposed change detectors because in general, they have a small delay in detecting such concept changes. We can note that DDM and ADWIN2 also have a good performance, as well as how the high false positives rate of ECDD affects the accuracy of the corresponding learning algorithm.

7.3 Real World Data

The selected real world datasets have been used in different studies about concept drift. For these datasets, it is not possible to make strong claims about the presence or type of drift. However, the benefit of evaluating methods with these real-world datasets explains their presence. In all cases we evaluate the methods processing the examples online in their temporal order. At each new example the classifier is first tested and then trained. Table 3 shows the average and standard deviation of the fraction among the number of the well-classified examples and the total of the examples every 100 examples processed; accuracy is computed with respect to a sliding window with size 100 [27], [28].

We experiment with the Electricity Market dataset (ELEC2)[3] and other five datasets obtained from the UCI Machine Learning Repository:[4] two datasets based on the 20 newsgroup collection (USENET1 and USENET2), the Spam Assassin collection (SPAM) [18], the Forest Cover Type (COVERTYPE) and the Nursery dataset.

In this experiment, we omitted the HDDM$_{W\text{-test}(0.15)}$ as it was mostly outperformed by HDDM$_{W\text{-test}(0.05)}$. Again, learning with change detection often improves the accuracy of the learner without change detection. The Perceptron had a poor performance in COVERTYPE, which indicates high non-linearity in its classification boundary.

ECDD had a very good performance, mainly in combination with the Naïve Bayes classifier. As Ross *et al.* [10] pointed out, this performance with a low ARL_0 denotes that changes are occurring quite often. This also shows the ability of Naïve Bayes to learn concepts quite quickly. As we said, when the base classifier does not learn the underlying concept so fast, to raise such drift signals have an additional cost. For example, DDM and HDDM$_{A\text{-test}}$, which have a low false positives rate, show a good performance in combination with the Perceptron.

Table 3 shows that all the proposed methods also has a competitive performance in real data. This time HDDM$_{A\text{-test}}$ mostly outperformed HDDM$_{W\text{-test}(\lambda)}$. We estimate that this behavior is given because many consecutive concepts on these real datasets are similar, causing the changes to have little scale in the error rate. As we have explained in Subsection 7.1, these changes are hard to detect for HDDM$_{W\text{-test}(\lambda)}$. For example, HDDM$_{A\text{-test}}$ usually raises less drift signals than HDDM$_{W\text{-test}(\lambda)}$, but in these real-world data it raised less drift signals only in the NURSERY dataset.

8 Conclusions and Future Work

In this paper we have presented a family of methods to detect concept drift in online learning domains through moving averages and weighted moving averages. Theoretical performance guarantees for these methods are provided using probability inequalities that do not assume knowledge about the probability distribution function. For the weighted moving average, we have

3. http://moa.cms.waikato.ac.nz/datasets/
4. http://www.ics.uci.edu/mlearn/MLRepository.html

Algorithm			ELEC2	SPAM	USENET1	USENET2	COVERTYPE	NURSERY
HDDM$_{W\text{-test}(0.05)}$	NB	$\overline{x}$	84.47	**91.51**	75.07	70.93	86.23	91.71
		s	6.56	**7.35**	11.51	13.32	8.07	7.58
	P	$\overline{x}$	45.01	97.24	75.07	$\underline{74.93}$	0.79	83.65
		s	15.07	3.07	9.95	$\underline{9.04}$	5.35	11.68
HDDM$_{A\text{-test}}$	NB	$\overline{x}$	85.09	90.67	75.20	71.00	87.44	**92.51**
		s	6.32	9.26	11.20	12.84	7.96	**6.48**
	P	$\overline{x}$	$\underline{46.65}$	97.23	$\underline{76.93}$	$\underline{74.93}$	1.68	82.62
		s	$\underline{15.53}$	3.05	$\underline{9.15}$	$\underline{8.82}$	9.15	12.00
DDM	NB	$\overline{x}$	82.70	89.50	73.73	**72.93**	88.03	91.72
		s	8.69	13.82	12.26	**11.68**	8.35	7.09
	P	$\overline{x}$	43.45	$\underline{97.47}$	74.40	$\underline{74.93}$	32.39	83.48
		s	14.47	$\underline{2.97}$	11.46	$\underline{9.12}$	33.76	10.97
ADWIN2	NB	$\overline{x}$	81.01	91.37	70.27	72.13	83.28	91.90
		s	9.44	7.15	15.79	11.15	10.85	6.93
	P	$\overline{x}$	43.08	97.23	71.80	74.87	4.15	$\underline{83.91}$
		s	14.28	3.34	11.54	8.44	15.27	$\underline{10.98}$
ECDD	NB	$\overline{x}$	**87.08**	88.03	**76.53**	66.53	**90.52**	84.72
		s	**4.56**	15.20	**8.63**	15.24	**6.97**	7.17
	P	$\overline{x}$	42.44	95.00	71.73	72.93	36.45	65.74
		s	14.00	5.43	10.55	9.97	29.47	17.27
No drift detector	NB	$\overline{x}$	74.17	90.63	63.33	72.13	60.53	83.35
		s	14.67	10.87	22.84	11.15	21.76	14.88
	P	$\overline{x}$	42.44	97.30	73.20	74.73	$\underline{48.75}$	75.78
		s	14.00	3.25	11.97	8.23	$\underline{32.12}$	19.11

Table 3
Experiments with real data.

mainly studied the exponentially weighted moving average (EWMA), which has been proved to be an optimal mean estimator for some disturbance processes and an effective estimator for various others. However, due to the generality of the presented formulation, it can be applied to many other weighted schemes.

The proposed methods are not dependent of the learning algorithm, and consequently they can be applied to any classifier in order to track concept drift. Another relevant aspect in the area of data stream is that the methods have $O(1)$ complexity in time order and store only the required information in a constant number of counters. All proposed methods improve the learning accuracy of the algorithm when modeling non-stationary problems.

We have already initiated a study of how the proposed change detectors respond to diverse characteristics of concepts and drifts. We have tested our change detectors on streaming bits, synthetic and real time-changing data streams. In streaming bits, we empirically found the false positives and false negatives rate, as well as the detection delay of the drift detection methods. In data streams, we tested these methods by monitoring the error rate of a Naïve Bayes classifier and a Perceptron, a warning signal was used to train an alternative classifier that replaces the original one (who makes predictions) when this warning signal is followed by a drift signal. All experiments show that the resulting algorithms effectively adapt its learning in stables and drifting concepts.

We expect to continue this research with other learning algorithms, different weighting schemes and applications for real-world problems. Preliminary results by means of empirical evaluations over artificial and real data seem to be very promising.

ACKNOWLEDGMENTS

The authors wish to thank the editor and the anonymous revisors for their useful comments and suggestions.

This work was supported in part by the SESAAME project number TIN2008-06582-C03-03 of the MEC, Spain. The authors also wish to thank the support of the AUIP (Asociación Universitaria Iberoamericana de Postgrado).

REFERENCES

[1] H. Wang, W. Fan, P. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003, pp. 226–235.

[2] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys, in press*, 2014.

[3] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Advances in Artificial Intelligence, Proceedings of SBIA 2004*, vol. 3171, 2004, pp. 286–295.

[4] M. Baena, J. del Campo, R. Fidalgo, A. Bifet, R. Gavaldà, and R. Morales, "Early Drift Detection Method," in *4th International Workshop on Knowledge Discovery from Data Streams*, 2006.

[5] A. Bifet and R. Gavaldà, "Learning from time-changing data with adaptive windowing," in *2007 SIAM International Conference on Data Mining*, 2007.

[6] G. Ross, D. Tasoulis, and N. Adams, "Nonparametric monitoring of data streams for changes in location and scale," *Technometrics*, vol. 53, no. 4, pp. 379–389, 2011.

[7] M. Datar, A. Gionis, P. Indyk, and R. Motwani, "Maintaining stream statistics over sliding windows," *SIAM Journal on Computing*, vol. 31, no. 6, pp. 1794–1813, 2002.

[8] D. Montgomery, *Introduction to Statistical Quality Control.* Wiley: New York, 2001.

[9] D. Kifer, S. Ben, and J. Gehrke, "Detecting change in data streams," in *Proceedings of the 30th International Conference on Very Large Data Bases*, vol. 30, 2004, pp. 180–191.

[10] G. Ross, N. Adams, D. Tasoulis, and D. Hand, "Exponentially weighted moving average charts for detecting concept drift," *Pattern Recognition Letters*, vol. 33, no. 2, pp. 191–198, 2012.

[11] I. Žliobaitė, A. Bifet, B. Pfahringer, and G. Holmes, "Active learning with drifting streaming data," *IEEE Transactions on Neural Networks and Learning Systems, in press*, 2013.
[12] P. Bartlett, "Learning with a slowly changing distribution," in *Proceedings of the 5th Annual Workshop on Computational Learning Theory*, 1992, pp. 243–252.
[13] D. Helmbold and P. Long, "Tracking drifting concept by minimizing disagreements," *Machine Learning*, vol. 14, no. 1, pp. 27–45, 1994.
[14] A. Chen and E. Elsayed, "Design and performance analysis of the exponentially weighted moving average mean estimate for processes subject to random step changes," *Technometrics*, vol. 44, no. 4, 2002.
[15] J. Gama and P. Rodrigues, *Learning from Data Streams: Processing Techniques in Sensor Networks*, 1st ed. Springer-Verlag, 2007.
[16] M. Lazarescu and S. Venkatesh, "Using multiple windows to track concept drift," *Intelligent Data Analysis*, vol. 8, no. 1, pp. 29–59, 2004.
[17] R. Klinkenberg, "Learning drifting concepts: example selection vs. example weighting," *Intelligent Data Analysis*, vol. 8, no. 3, pp. 281–300, 2004.
[18] I. Katakis, G. Tsoumakas, and I. Vlahavas, "An ensemble of classifiers for coping with recurring contexts in data streams," in *Proceedings of the 2008 Conference on ECAI 2008: 18th European Conference on Artificial Intelligence*, 2008, pp. 763–764.
[19] G. Box and G. Jenkins, *Time Series Analysis, Forecasting and Control*. Holden-Day, Incorporated, 1990.
[20] J. Lucas, M. Saccucci, R. Baxley, W. Woodall, H. Maragh, F. Faltin, G. Hahn, W. Tucker, J. Hunter, J. MacGregor, and T. Harris, "Exponentially weighted moving average control schemes: properties and enhancements," *Technometrics*, vol. 32, no. 1, pp. 1–29, 1990.
[21] M. Núñez, R. Fidalgo, and R. Morales, "Learning in environments with unknown dynamics: Towards more robust concept learners," *Journal of Machine Learning Research*, vol. 8, pp. 2595–2628, 2007.
[22] E. Cohen and M. Strauss, "Maintaining time-decaying stream aggregates," *Journal of Algorithms*, vol. 59, no. 1, pp. 19–36, 2006.
[23] J. del Campo, G. Ramos, J. Gama, and R. Morales, "Improving the performance of an incremental algorithm driven by error margins," *Intelligent Data Analysis*, vol. 12, no. 3, pp. 305–318, 2008.
[24] W. Hoeffding, "Probabilities inequalities for sums of bounded random variables," *Journal of American Statistical Association*, vol. 58, no. 301, pp. 13–30, 1963.
[25] C. McDiarmid, "On the method of bounded differences," in *Surveys in Combinatorics*, 1989, pp. 148–188.
[26] T. Dasu, S. Krishnan, and G. Pomann, "Robustness of change detection algorithms," in *Proceedings of the 10th International Conference on Advances in Intelligent Data Analysis*, 2011, pp. 125–137.
[27] J. Gama, R. Sebastião, and P. Rodrigues, "On evaluating stream learning algorithms," *Machine Learning*, vol. 90, no. 3, 2013.
[28] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "MOA: Massive Online Analysis," *Journal of Machine Learning Research*, vol. 11, pp. 1601–1604, 2010.
[29] S. Bach and M. Maloof, "Paired learners for concept drift," in *Proceedings of the 8th IEEE International Conference on Data Mining*, 2008, pp. 23–32.

Isvani Frías-Blanco received the B.S. degree in computer science from the University of East, Cuba, in 2005 and the M.Sc. degree in computer science from the University of Granma, Cuba, in 2007. He is an assistant professor in the Regional Faculty of Granma, University of Computer Sciences, Granma, Cuba. His research interests include machine learning and data mining.

José del Campo-Ávila received the Ph.D. degree in software engineering and artificial intelligence from the University of Málaga in 2007. His research interests include incremental learning, mining data streams for classification, and multiple classifier systems, among others. He is an assistant professor in the Department of Lenguajes y Ciencias de la Computación at the University of Málaga and a member of the (IA)^2 research group.

Gonzalo Ramos-Jiménez received the B.S. degree in computer science engineering and the B.S. degree in psychology from the University of Málaga, Málaga, Spain. In 2001 he received the Ph.D. degree in computer science from the same university. He is a full professor in the Department of Lenguajes y Ciencias de la Computación at the University of Málaga. His research interests include machine learning and data mining, among others. He is also interested in cognitive science, and he is a member of EATCS (European Association for Theoretical Computer Science), AEPIA (Asociación Española de Inteligencia Artificial) and the (IA)^2 research group.

Rafael Morales-Bueno received the Ph.D. degree in computer science from the University of Málaga, Málaga, Spain, in 1991. He is a full professor in the Department of Lenguajes y Ciencias de la Computación at the University of Málaga. He is a member of the (IA)^2 research group. His research interests include computational learning, machine learning and data mining, among others. He is a member of EATCS (European Association for Theoretical Computer Science).

Agustín Ortiz-Díaz received the B.S. degree in computer science from the University of East, Cuba, in 2003 and the M.Sc. degree in computer science from the University of Granma, Cuba, in 2007. He is an assistant professor in the Department of Computer Science, University of Granma, Granma, Cuba. His research interests include machine learning and data mining.

Yailé Caballero-Mota received the M.Sc. degree in computer science from the Central University of Las Villas, Villa Clara, Cuba, and the Ph.D. degree in computer science, in 2007, from the same university. She is a full professor in the Department of Computer Science, University of Camagüey, Camagüey, Cuba. Her research interests include artificial neural networks, machine learning and data mining.