McDiarmid drift detection methods for evolving data streams

Pesaranghader, Ali; Viktor, Herna; Paquet, Eric

Publisher's version / Version de l'éditeur:

Statistics, 2017-10-05

McDiarmid Drift Detection Methods for Evolving Data Streams

Ali Pesaranghader [1] (✉), Herna L. Viktor [1], Eric Paquet [1,2]

[1] *School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, ON K1N 6N5, Canada*
{apesaran, hviktor}@uottawa.ca

[2] *National Research Council, 1200 Montreal Road, Ottawa, ON K1A 0R6, Canada*
eric.paquet@nrc-cnrc.gc.ca

***Abstract*—Increasingly, Internet of Things (IoT) domains, such as sensor networks, smart cities, and social networks, generate vast amounts of data. Such data are not only unbounded and rapidly evolving. Rather, the content thereof dynamically evolves over time, often in unforeseen ways. These variations are due to so-called concept drifts, caused by changes in the underlying data generation mechanisms. In a classification setting, concept drift causes the previously learned models to become inaccurate, unsafe and even unusable. Accordingly, concept drifts need to be detected, and handled, as soon as possible. In medical applications and military zones, for example, change in behaviors should be detected in near real-time, to avoid potential loss of life. To this end, we introduce the McDiarmid Drift Detection Method (MDDM), which utilizes McDiarmid's inequality [1] in order to detect concept drift. The MDDM approach proceeds by sliding a window over prediction results, and associate window entries with weights. Higher weights are assigned to the most recent entries, in order to emphasize their importance. As instances are processed, the detection algorithm compares a weighted mean of elements inside the sliding window with the maximum weighted mean observed so far. A significant difference between the two weighted means, upper-bounded by the McDiarmid inequality, implies a concept drift. Our extensive experimentation against synthetic and real-world data streams show that our novel method outperforms the state-of-the-art. Specifically, MDDM yields shorter detection delays as well as lower false negative rates, while maintaining high classification accuracies.**

I. Introduction

A proliferation of Internet-enabled devices such as smartphones, tablets, and smartwatches are ubiquitous in our society. These devices continuously generate vast amounts of data in the form of infinite streams. In most applications, sensors are responsible for collecting the data from their surrounding environment, in order to facilitate data-driven decision-making. For instance, smart houses make use of various types of sensors for adapting the domestic services to the changing needs of their inhabitants. Automobiles are equipped with onboard chips that uninterruptedly monitor vehicle-health, fuel consumption and driver behavior, while detecting unexpected events on the road [2].

Learning from data streams coming from a dynamic sensor network is a challenging task for various reasons: (1) sensors are generally not synchronized, (2) they have access to a limited amount of computational and memory resources, (3) high throughput external processing is restricted by the low transmission bandwidth of the sensors, (4) data are generated at high rate; and (5) sensors usually operate in evolving environments [2]. The first three challenges may be overcome by topological or hardware solutions; whereas, the last two may be addressed by online and adaptive learning algorithms. This paper focuses on the last two issues, namely the high rate of arrival and the evolving nature of such environments.

Traditionally, machine learning algorithms assume that the data are generated by a stationary distribution and that all the data are collected prior to learning. Yet, these assumptions are not valid in evolving environments, where the underlying distributions may change over time: a phenomenon known as *concept drift* [3]. As a consequence, model accuracy decreases as concept drifts arise. Therefore, adaptation to new distributions (or situations) is essential to ensure the efficiency of the decision-making process. An adaptive learning algorithm may utilize a drift detection method for observing concept drifts in a data stream. Once the drift detector signals the presence of a concept drift, the learning algorithm updates and adapts its current model by taking into account the new distribution. For the learning process to be efficient, the drift detector must detect concept drifts rapidly, while maintaining low false negative and false positive rates. In this paper, we introduce the McDiarmid Drift Detection Method (MDDM) which applies the McDiarmid inequality [1] and various weighting schemes in order to detect rapidly and efficiently concept drifts. Through numerous experiments, we show that MDDM finds abrupt and gradual concept drifts with shorter delays and with lower false negative rates, compared to the state-of-the-art.

This paper is organized as follows. *Data stream classification* and *concept drift* are formally defined in Sections II and III, respectively. Section IV describes adaptive learning as a form of incremental learning from evolving data streams. Section V reviews the state-of-the-art for concept drift detection. In Section VI, we introduce the McDiarmid Drift Detection Methods (MDDMs). Next, in Section VII, our approaches are compared with the start-of-the-art for both synthetic and real-world data streams. We conclude the paper and discuss future work in Section VIII.

II. Data Stream Classification

The primary objective of data stream classification is to build a model *incrementally*, using the (current) available data, the so-called training data, for predicting the label of unseen examples. Data stream classification may be defined as follows:

> Let a stream S be a sequence of instances: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_t, y_t)$. The pair $(\mathbf{x}_t, y_t)$ represents an instance arriving at time t, where $\mathbf{x}_t$ is a vector containing k attributes: $\mathbf{x} = (x_1, x_2, ..., x_k)$, while y_t is a *class label* which belongs to a *finite* set of size m, $y_t \in \{c_1, c_2, ..., c_m\}$. Assume a target function $y_t = f(\mathbf{x}_t)$ which maps an input vector to a particular class label. The learning task consists of incrementally building a model $\tilde{f}$ that approximates the function f at all time. Naturally, an approximation which *maximizes* the classification accuracy is preferred [4].

As suggested in the literature [5], [6], for data stream classification, incremental learning algorithms should fulfill four essential requirements: (1) the examples should be processed one-by-one and only once in the order of their arrival, (2) memory usage should be constrained as the size of a data stream is typically substantially larger than the size of the available memory, (3) all the calculations should be performed in real-time or at least, in near real-time, and (4) the outcome of the classification process should be available at any time.

III. Concept Drift Definition

The Bayesian Decision Theory is commonly employed in describing classification processes based on their prior probability distribution of classes, i.e. $p(y)$, and the class conditional probability distribution, i.e. $p(\mathbf{x}|y)$ [3], [7]. The classification decision is related to the posterior probabilities of the classes. The posterior probability associated with class c_i, given instance $\mathbf{x}$, is obtained by:

$$p(c_i|\mathbf{x}) = \frac{p(c_i) \cdot p(\mathbf{x}|c_i)}{p(\mathbf{x})} \quad (1)$$

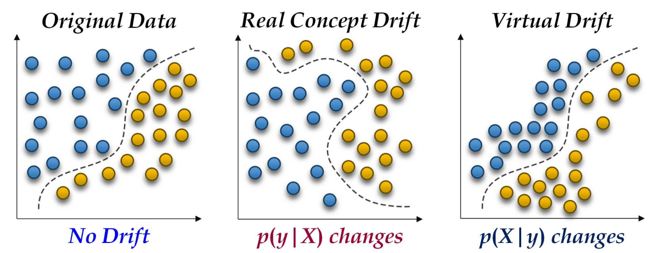
where $p(\mathbf{x}) = \sum_{i=1}^{m} p(c_i) \cdot p(\mathbf{x}|c_i)$ is the marginal probability distribution. Formally, if a concept drift occurs in between time t_0 and t_1 we have:

$$\exists \mathbf{x} : p_{t_0}(\mathbf{x}, y) \neq p_{t_1}(\mathbf{x}, y) \quad (2)$$

where p_{t_0} and p_{t_1} represent the joint probability distributions at time t_0 and t_1, respectively [3]. Eq. (2) implies that the data distribution at times t_0 and t_1 are distinct, as their joint probabilities differ. From Eq. (1), it may be observed that a concept drift may occur [3]:

- As a result of a change in the prior probability distribution of the classes $p(y)$,
- As a result of a change in the class conditional probability distributions $p(\mathbf{x}|y)$,
- As a result of a change in the posterior probability distribution of the classes $p(y|\mathbf{x})$, thus affecting the classification decision boundaries.

Gama et al. [3] and Žliobaitė [7] classify changes into two types, namely *real concept drift* and *virtual concept drift*. A *real concept drift* refers to the changes in $p(y|\mathbf{x})$ which affects the decision boundaries or the target concept (as shown in Fig. 1 (b)). On the other hand, *virtual drift* is the result of a change in $p(\mathbf{x})$, and subsequently in $p(\mathbf{x}|y)$, but not in $p(y|\mathbf{x})$. That is, a virtual drift is a change in the distribution of the incoming data which implies that the decision boundaries remain unaffected (as in Fig. 1 (c)).



(a) Original Data (b) Real Drift (c) Virtual Drift

Fig. 1: Real Concept Drift vs. Virtual Concept Drift (Similar to Fig. 1 in [3])

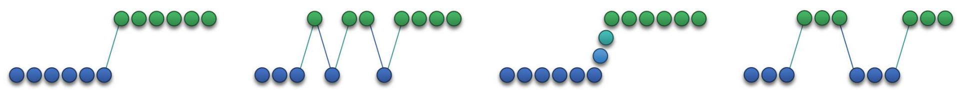
In practice, a virtual concept drift may appear in conjunction with a real concept drift. As a result, the class boundary is also altered. Consequently, from a predictive perspective, adaptation is required once a real concept drift occurs, since the current decision boundary turns out to be obsolete [3]. By adaptation, we mean updating the classification model according to the new distribution in order to maintain a high classification accuracy. Adaptive learning is discussed in Section IV.

A. Concept Drift Patterns

A concept drift may appear in different patterns [7]; as illustrated in Fig. 2. An *abrupt* concept drift results from a sudden change in the data distribution. On the other hand, a gradual concept drift results from a slow transition from one data distribution to the next. The two patterns may coexist concurrently (Fig. 2 (b)). In an incremental concept drift, a sequence of data distributions appear during the transition. In re-occurring concept drift, a previously active concept reappears after some time, as shown in Fig. 2 (d). In practice, a mixture of different concept drifts may be present.

IV. Adaptive Data Stream Learning

As learning algorithms are often trained in non-stationary environments, where concept drift is inevitable, they must have the capacity to adapt to new situations. Adaptive learning is defined as a form of advance incremental learning in which



(a) Abrupt (b) Gradual (c) Incremental (d) Re-occurring

Fig. 2: Patterns of Concept Drifts (Similar to Fig. 2 in [3], and colors represent different distributions)

concept drifts are detected while the classification models are updated accordingly [3]. Adaptation methods fall into two categories known as *blind* and *informed*.

Blind Adaptation – Blind adaptive algorithms adapt the classification model without any concept drift detection. These algorithms often use a sliding window mechanism to hold the most recent examples from a stream. As the window is displaced over the examples, the model is periodically updated.

Informed Adaptation – Informed adaptive algorithms are reactive, meaning that they adapt their models once an alarm has been triggered by a concept drift detector [3]. The reaction to an alarm for a concept drift may be in the form of *global replacement* or *partial replacement* [3]. In *global replacement*, the outdated model is discarded, and a new model is trained from scratch. This strategy is suitable for *global classifiers*, e.g. Naive Bayes learners, as well as for *granular classifiers*, such as Decision Trees [3]. In contrast, in *partial replacement*, the adaptation is limited to some parts of the classification model in order to better represent the data space that has been partially affected by a concept drift. This strategy is restricted to *granular classifiers* where only some parts of the models must be updated.

A. Adaptive Learning Requirements

Recall that classical incremental algorithms should process each example only once, use a limited amount of memory, converge in a limited amount of time, and be ready to perform a prediction at any time. Adaptive learning algorithms must fulfill the following requirements in order to maintain high predictive performances [8], [9], [10]: (1) *Minimum false positive and false negative rates* – an adaptive algorithm must detect concept drifts with a small number of false positives and false negatives. A high false positive rate involves more model retraining which in turn requires more computational resources [11]. On the other hand, a high false negative rate reduces the classification accuracy, as the current model does not reflect the new distribution. (2) *Short drift detection delay* – An adaptive learning algorithm should detect concept drifts rapidly, and update its predictive model in quasi real-time in order to maintain the classification accuracy. (3) *Robustness to noise* – adaptive learners must be able to distinguish concept drift from noise. Indeed, no adaptation is required if noise is present in a stream.

V. Concept Drift Detection Methods

Change detection methods refer to techniques and algorithms that detect concept drifts and distributional changes explicitly. Drift detection methods characterize and quantify concept drifts by discovering the change points or small time intervals during which concept drifts occur. Gama et al. [3] classify concept drift detectors into three groups:

1) *Sequential Analysis based Methods* sequentially evaluate prediction results as they become available. They alarm for concept drifts when a pre-defined threshold is met. The Cumulative Sum (CUSUM) and its variant PageHinkley (PH) [12], as well as Geometric Moving Average (GMA) [13] are representatives of this group.
2) *Statistical based Approaches* analyze statistical parameters such as the mean and the standard deviation associated with the predicted results in order to detect concept drifts. The Drift Detection Method (DDM) [14], Early Drift Detection Method (EDDM) [15], Exponentially Weighted Moving Average (EWMA) [16], and Reactive Drift Detection Method (RDDM) [17] are members of this group.
3) *Windows based Methods* usually utilize a fixed reference window for summarizing the past information and a sliding window for summarizing the most recent information. A significant difference in between the distributions of these two windows implies the occurrence of a concept drift. Statistical tests or mathematical inequalities, with the null-hypothesis indicating that the distributions are equal, are employed. The Adaptive Windowing (ADWIN) [18], the SeqDrift detectors [8], the Drift Detection Methods based on Hoeffding's Bound ($\text{HDDM}_{\text{A-test}}$ and $\text{HDDM}_{\text{W-test}}$) [4], and the Adaptive Cumulative Windows Model (ACWM) [19] are members of this family.

CUSUM and its variant PageHinkley (PH) are some of the pioneer methods in the community. DDM, EDDM, and ADWIN have frequently been considered as benchmarks in the literature [4], [9], [15], [18], [20]. SeqDrift2, HDDMs, and RDDM present similar performances. Therefore, all these methods are evaluated in our experiments. These methods are described in more detail below:

- Cumulative Sum (CUSUM), by Page [12], alarms for a change when the mean of the input data significantly deviates from zero. The input of CUSUM may be, for instance, the prediction error from a Kalman filter [3]. The CUSUM test has the form $g_t = max(0, g_{t-1} + (x_t - \delta))$, and alarms for concept drift when $g_t > \lambda$. Here, x_t is the current observed value, δ specifies the magnitude of allowed changes, $g_0 = 0$, and λ is a user-defined threshold. The accuracy of CUSUM depends on both δ and λ. Lower values of δ result in a faster detection, but at the cost of an increasing number of false alarms.

- PageHinkley (PH), by Page [12], is a variant of CUSUM employed for change detection in signal processing [3]. The test variable m_T is defined as the cumulative difference between the observed values and their mean until the current time T; and is evaluated by $m_T = \sum_{t=1}^{T}(x_t - \bar{x}_T - \delta)$, where $\bar{x} = \frac{1}{T}\sum_{t=1}^{T} x_t$ and where δ controls the variation range. The PH method also updates the minimum of m_T, denoted as M_T, using $M_T = min(m_t, t = 1...T)$. Significant difference in between m_T and M_T, i.e. $PH_T : m_T - M_T > \lambda$ where λ is a user-defined threshold, indicates a concept drift. A large value of λ typically results in fewer false alarms, but at the expense of increasing the false negative rate.
- Drift Detection Method (DDM), by Gama et al. [14], monitors the error-rate of the classification model to detect drifts. On the basis of the probably approximately correct (PAC) learning model [21], the method considers that the error-rate of a classifier decreases or stays constant as the number of instances increases. Otherwise, it suggests the occurrence of a drift. Let p_t be the error-rate of the classifier with a standard deviation of $s_t = \sqrt{(p_t(1-p_t)/t)}$ at time t. As instances are processed, DDM updates two variables p_{min} and s_{min} when $p_t + s_t < p_{min} + s_{min}$. DDM warns for a drift when $p_t + s_t \geq p_{min} + 2 * s_{min}$, and while a drift is detected when $p_t + s_t \geq p_{min} + 3 * s_{min}$. The variables p_{min} and s_{min} are reset when a drift occurs.
- Early Drift Detection Method (EDDM), by Baena-Garcia et al. [15], evaluates the distances between wrong predictions to detect concept drifts. The algorithm is based on the observation that a drift is more likely to occur when the distances between errors are smaller. EDDM calculates the average distance between two recent errors, i.e. p'_t, with its standard deviation s'_t at time t. It updates two variables p'_{max} and s'_{max} when $p'_t + 2 * s'_t > p'_{max} + 2 * s'_{max}$. The method warns for a drift when $(p'_t + 2 * s'_t)/(p'_{max} + 2 * s'_{max}) < \alpha$, and indicates that a concept drift occurred when $(p'_t + 2 * s'_t)/(p'_{max} + 2 * s'_{max}) < \beta$. The authors set α and β to 0.95 and 0.90, respectively. The p'_{max} and s'_{max} are reset only when a drift is detected.
- Reactive Drift Detection Method (RDDM), by Barros et al. [17], addresses a performance loss problem of Drift Detection Method (DDM) when the sensitivity of the method deteriorates over time, particularly in the context of large concept drifts. As in DDM, RDDM evaluates two variables p_t and s_t over time; and updates p_{min} and s_{min} if $p_t + s_t < p_{min} + s_{min}$. The constants α_w and α_d represent the warning and drift levels, respectively. Additionally, RDDM holds three variables: max (the maximum size of a concept), min (the reduced size of a stable concept), and $warnLimit$ (the maximum number of instances that limits the warning level). The method warns for a drift when $p_t + s_t > p_{min} + \alpha_w * s_{min}$, and alarms for a drift when either of following three conditions occur (1) $p_t + s_t > p_{min} + \alpha_d * s_{min}$, (2) $num\_instances > max$, or (3) $num\_warnings > warnLimit$.
- Adaptive Windowing (ADWIN), by Bifet and Gavalda [18], slides a window w as the predictions become available, in order to detect drifts. The method examines two sub-windows of sufficient length, i.e. w_0 of size n_0 and w_1 of size n_1 where $w_0 \cup w_1 = w$. A significant difference between the means of two sub-windows indicates a concept drift, i.e. when $|\hat{\mu}_{w_0} - \hat{\mu}_{w_1}| \geq \varepsilon$ where $\varepsilon = \sqrt{\frac{1}{2m}\ln\frac{4}{\delta'}}$, m represents the harmonic mean of n_0 and n_1, and $\delta' = \delta/n$. Here δ is the confidence level while n is the size of window w. Once a drift is detected, elements are removed from the tail of the window until no significant difference is observed.
- SeqDrift2, by Pears et al. [8], relies on a reservoir sampling method [22], as an adaptive sampling strategy, for random sampling from input data. SeqDrift2 stores entries into two repositories called *left* and *right*. As entries are processed over time, the left repository forms a combination of old and new entries by applying the reservoir sampling strategy, while the right repository collects the new entries. SeqDrift2 subsequently finds an upper-bound for the difference in between the means of the two repositories, i.e. $\hat{\mu}_l$ for the left repository and $\hat{\mu}_r$ for the right repository, using the Bernstein inequality [23]. Finally, a significant difference between the two means suggests a concept drift.
- $HDDM_{\text{A-test}}$, by Frías-Blanco et al. [4], detects concept drifts by comparing moving averages. A significant difference between them indicates a concept drift. $HDDM_{\text{W-test}}$, a variant of $HDDM_{\text{A-test}}$, employs the EMWA forgetting scheme [16] to weight the moving averages. Then, the weighted moving averages are compared to detect concept drifts. In both cases, the Hoeffding inequality [24] determines an upper-bound for the difference between the two averages. The authors have suggested that the first and the second methods are ideal for detecting abrupt and gradual drifts, respectively.

Discussion – CUSUM and PageHinkley (PH) detect concept drifts from the deviation of the observed values from their mean and alarm for a drift when this difference exceeds a user-defined threshold. These algorithms are sensitive to the parameter values, resulting in a trade-off between false alarms and detecting true drifts. DDM and EDDM require less memory as only a small number of variables is maintained [3]. On the other hand, the ADWIN and SeqDrift2 approaches necessitate multiple subsets of the stream which lead to more memory consumption. They are also computationally more expensive, due to the sub-window compression or reservoir sampling procedures. Barros et al. [17] observed that, in general, RDDM leads to a higher classification accuracy compared to DDM, especially against datasets with gradual concept drift, despite an increase in false positives. EDDM may frequently alarm for concept drift in the early stages of learning if the distances in between wrong predictions are small. HDDM employs the Hoeffding inequality to detect concept drifts. Recall that SeqDrift2 employs the Bernstein inequality in order to detect concept drift. SeqDrift2 uses the sample variance, and assumes that the sampled data follow a normal distribution. This assumption may be too restrictive, in real-world domains. Further, the Bernstein inequality is conservative and requires a variance parameter, in contrast to, for instance, the Hoeffding inequality. These shortcomings may lead to longer detection delay and a potential loss of accuracy. Moreover, our preliminary experimentation confirmed that these methods may cause long drift detection delay, as well as high false positive and false negative rates.

In Section VI, we introduce the McDiarmid Drift Detection Methods (MDDM), which extends our Fast Hoeffding Drift Detection Method (FHDDM) which was introduced recently [9]. FHDDM slides a window over the stream, in order to detect concept drift. Two variables are maintained, namely the mean of the elements inside the window at the current time as well as the maximum mean observed so far. FHDDM employs the Hoeffding inequality [24] to detect concept drifts. FHDDM and MDDM are detailed in the next section.

VI. McDiarmid Drift Detection Methods

In a streaming environment, one may assume that old examples are either obsolete or outdated. Therefore, incremental learners should rely on the most recent examples for training, as the latter reflect the current situation more adequately. Fading or weighting approaches are typically used by online learning algorithms to increase the weight attributed to the most recent instances [3]. This is important from an adaptive learning perspective, especially when a transition between two contexts is occurring. For instance, Klinkenberg [25] relies on an exponential weighting scheme $w_\lambda(\mathrm{x}_i) = exp(-\lambda i)$, where λ is a parameter and i is the entry index, to assign lower weights to old examples. Assigning higher weights to recent predictions results in a faster detection of concept drifts. In this section, we introduce the McDiarmid Drift Detection Methods (DMMDs) which utilizes a weighting scheme to ponderate the elements of the window. We, first, introduce the Fast Hoeffding Drift Detection Method (FHDDM) [9] since our MDDM approaches extend the former.

A. Fast Hoeffding Drift Detection Method (FHDDM)

Fast Hoeffding Drift Detection Method (FHDDM) uses the Hoeffding inequality [24] to detect drifts in evolving data streams. The FHDDM algorithm slides a window of size n over the prediction results. The algorithm inserts a 1 into the window if the prediction result is *correct* and 0 otherwise. As inputs are processed, the mean of the elements of the sliding window is calculated, i.e. μ^t, as well as the maximum mean observed so far, i.e. μ^m. As indicated in Eq. (3).

$$\text{if} \quad \mu^m < \mu^t \Rightarrow \mu^m = \mu^t \tag{3}$$

On the basis of the PAC learning model [21], the classification accuracy should increase or stay constant as the number of instances increases; otherwise, the possibility of facing drifts increases [14]. Thus, the value of μ^m should increase or remain constant as more instances are processed. In other words, the possibility of facing a concept drift increases if μ^m does not change and μ^t decreases over time. Finally, as shown by Eq. (4), a significant difference in between μ^m and μ^t indicates the occurrence of a drift:

$$\Delta\mu = \mu^m - \mu^t \geq \varepsilon_d \Rightarrow \text{Drift := True} \tag{4}$$

In [9], the value of ε_d was evaluated by the Hoeffding inequality, i.e. Eq. (7). The Hoeffding inequality has the very attractive property that no particular probability distribution is assumed for the data [4], [24], [26]. The Hoeffding inequality assigns an upper-bound for the deviation between the *empirical mean* of n random variables and their corresponding expected value.

Theorem I: Hoeffding's Inequality – Let $X_1, X_2, ..., X_n$ be n independent random variables such that $X_i \in [a_i, b_i]$, where $i \in \{1, ..., n\}$. Then, given any $\varepsilon_H > 0$, we have for the difference between the empirical mean $\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$ and its expectation $E[\overline{X}]$:

$$\Pr\{E[\overline{X}] - \overline{X} \geq \varepsilon_H\} \leq \exp\left(-\frac{2n^2\varepsilon_H^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right) \tag{5}$$

As a result of this theorem, given a confidence level of δ_H, the value of ε_H is determined by:

$$\varepsilon_H = \sqrt{\frac{1}{2n}\ln\frac{1}{\delta_H}} \tag{6}$$

Corollary I: FHDDM test – In a stream setting, assume μ^t is the mean of a sequence of n random entries, each belonging to $\{0, 1\}$, at time t, and μ^m is the maximum probability observed so far. Let $\Delta\mu = \mu^m - \mu^t \geq 0$ be the difference between these two quantities. Given the desired δ_d, i.e. the probability of error allowed, the Hoeffding inequality entails that a drift is detected if $\Delta\mu \geq \varepsilon_d$, where:

$$\varepsilon_d = \sqrt{\frac{1}{2n}\ln\frac{1}{\delta_d}} \tag{7}$$

B. McDiarmid Drift Detection Methods (MDDMs)

This section presents our MDDM approach, which extends the above-mentioned FHDDM. MDDM is based on the assumption that by weighting the prediction results associated with a sliding window, and by putting more emphasis on the most recent elements, concept drift could be detected faster and more efficiently. This concept is illustrated in Fig. 3. Given the rule $w_i < w_{i+1}$, the elements at the head of the window have higher weights than those located at the tail. Different weighting schemes have been considered including *arithmetic* and *geometric* schemes. The arithmetic scheme is given by $w_i = 1 + (i - 1) * d$, where $d \geq 0$ is the difference between two consecutive weights. The geometric scheme is given by $w_i = r^{(i-1)}$, where $r \geq 1$ is the ratio of two consecutive weights. In addition, we employ the Euler scheme which is defined by $r = e^\lambda$ where $\lambda \geq 0$. We have implemented three weighted drift detection methods based on these three schemes: MDDM-A (A for arithmetic), MDDM-G (G for geometric), and MDDM-E (E for Euler)[1]. All these methods are described below. As the prediction results are processed one-by-one, the algorithm calculates the weighted average of the elements inside the sliding window, and simultaneously updates two variables μ_w^t (i.e. the current weighted average) and μ_w^m (i.e. the maximum weighted average observed so far). Similar to FHDDM, a significant difference between μ_w^m and μ_w^t implies a concept drift. The McDiarmid inequality [1] is employed to determine if the difference is deemed significant.

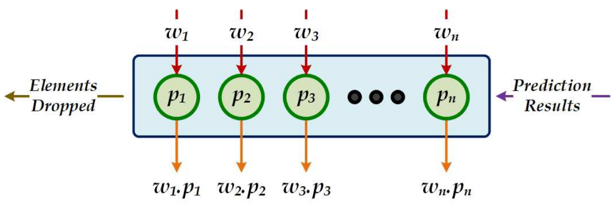


Fig. 3: McDiarmid Drift Detection Method (General Scheme)

Theorem II: McDiarmid's Inequality – Let $X_1, X_2, ..., X_n$ be n independent random variables all taking values in the set χ. Further, let $f : \chi^n \mapsto \mathbb{R}$ be a function of $X_1, ..., X_n$ that satisfies $\forall i, \forall x_1, ..., x_n, x_i' \in \chi$,

$$|f(x_1, ..., x_i, ..., x_n) - f(x_1, ..., x_i', ..., x_n)| \leq c_i.$$

This implies that replacing the i^{th} coordinate x_i by some arbitrary value changes the function f by at most c_i. Then, for all $\varepsilon_M > 0$, we have:

$$\Pr\{E[f] - f \geq \varepsilon_M\} \leq \exp\left(-\frac{2\varepsilon_M^2}{\sum_{i=1}^{n} c_i^2}\right) \tag{8}$$

Consequently, for a given confidence level δ_M, the value of ε_M is obtained by:

$$\varepsilon_M = \sqrt{\frac{\sum_{i=1}^{n} c_i^2}{2} \ln \frac{1}{\delta_M}} \tag{9}$$

Corollary II: MDDM test – In a stream context, assume μ_w^t is the weighted mean of a sequence of n random entries, at time t, and μ_w^m is the maximum weighted mean observed so far. Recall that each entry p_i is in $\{0, 1\}$ and has a weight of w_i. Let $\Delta\mu_w = \mu_w^m - \mu_w^t \geq 0$ be the difference between these two weighted means. Given the confidence level δ_w, the McDiarmid inequality detects a drift if $\Delta\mu_w \geq \varepsilon_w$, where

$$\varepsilon_w = \sqrt{\frac{\sum_{i=1}^{n} v_i^2}{2} \ln \frac{1}{\delta_w}} \tag{10}$$

and where v_i is given by

$$v_i = \frac{w_i}{\sum_{i=1}^{n} w_i} \tag{11}$$

All three MDDM approaches apply Corollary II in order to detect concept drift.

[1]The source codes are available at https://www.github.com/alipsgh/ (One may use them with the MOA framework [27]).

The pseudocode for the MDDM algorithm appears in Algorithm 1. Firstly, the INITIALIZE function initializes the parameters, including the window size n, confidence level δ_w, ε_w, the sliding window win, and μ_w^m. As the data stream examples are processed, the prediction results are pushed into the window (lines 8-14). The algorithm updates the variables μ_w^t and μ_w^m over time (lines 15-17). Finally, a drift is detected if $(\mu_w^m - \mu_w^t) \geq \varepsilon_w$ (lines 18-21). Recall that we have $w_i = 1+(i-1)*d$ for MDDM-A, $w_i = r^{(i-1)}$ for MDDM-G, and $w_i = e^{\lambda(i-1)}$ for MDDM-E.

Algorithm 1 McDiarmid Drift Detection Method

1: **function** INITIALIZE($windowSize, delta$)
2: $(n, \delta_w) \leftarrow (windowSize, delta)$
3: ε_w = CALCULATEEPSILON()
4: RESET()
5: **function** RESET()
6: win = [] ▷ Creating an empty sliding window.
7: $\mu_w^m = 0$
8: **function** DETECT(pr) ▷ *pr* is 1 for correct predictions, 0 otherwise.
9: **if** win.$size() = n$ **then**
10: win.$tail.drop()$ ▷ Dropping an element from the tail.
11: win.$push(pr)$ ▷ Pushing an element into the head.
12: **if** win.$size() < n$ **then**
13: **return** False
14: **else**
15: μ_w^t = GETWEIGHTEDMEAN()
16: **if** $\mu_w^m < \mu_w^t$ **then**
17: $\mu_w^m = \mu_w^t$
18: $\Delta\mu_w = \mu_w^m - \mu_w^t$
19: **if** $\Delta\mu_w \geq \varepsilon_w$ **then**
20: RESET() ▷ Resetting parameters.
21: **return** True ▷ Signaling for an alarm.
22: **else**
23: **return** False
24: **function** CALCULATEEPSILON()
25: $S = \sum_{i=1}^{n} v_i^2$ ▷ $v_i = w_i / \sum_{i=1}^{n} w_i$
26: **return** $\sqrt{\frac{S}{2} \ln \frac{1}{\delta_w}}$
27: **function** GETWEIGHTEDMEAN()
28: **return** $\sum_{i=1}^{n}(p_i \times w_i) / \sum_{i=1}^{n} w_i$

C. *Comparison of Hoeffding's and McDiarmid's Inequalities*

Recall that the Hoeffding inequality [24] bounds the difference in between the 'empirical mean' of n random variables and their expectation. This means that all the weights, which are implicit, are equal to one. Consequently, the Hoeffding inequality cannot be applied to a weighted average (see Eq. (5) to (7)). Alternatively, the McDiarmid inequality may be applied to an arbitrary function including a weighted average (see Eq. (8) to (11)). The Hoeffding inequality is a special case of the McDiarmid inequality which occurs when $\forall i, c_i = 1/n$ meaning that all the weights are equal to 1.

D. *Discussion On Variants of MDDM*

Recall that the MDDM-A approach employs an arithmetic scheme $w_i = 1+(i-1)*d$, where $w_{i+1} - w_i = d$ meaning that the weights increase *linearly*. On the the other hand, MDDM-G applies the geometric scheme $w_i = r^{(i-1)}$, indicating that the weights increase *exponentially* with $w_{i+1}/w_i = r$ (Note that $r = e^{\lambda}$ for MDDM-E). The linear or exponential nature of the weighting scheme affects the detection delay and the false positive rate. That is, the exponential weighting scheme often results in a faster concept drift detection, but at the expense of a higher false positive rate if compared to the linear weighting scheme. These statements are supported by experimental results in Section VII.

E. Parameters Sensitivity Analysis

Recall that the classic Fast Hoeffding Drift Detection Method (FHDDM) has two parameters: n and δ_d. These parameters are inversely proportional with respect to ε_d. That is, as the value of n increases, the value of ε_d decreases. This implies that, as observations become available, a more optimistic error bound should be applied. On the other hand, as the value of δ_d decreases, the values of ε_d increases (i.e. the bound becomes more conservative). These observations also apply to the McDiarmid Drift Detection Methods (MDDMs). The parameter d in MDDM-A controls the scale of the weights assigned to the sliding window elements. The value of ε_w increases, as the value of d increases. Larger values of d lead to faster drift detection, since higher weights are assigned to the element located at the head of the window; however, the false positive rate may increase. MDDM-G and MDDM-E behave similarly; the scale of their weight is determined by their parameters r and λ, respectively. That is, a higher r or λ leads to a shorter detection delay, but at the expense of a higher false positive rate. In order to set the default values of these parameters, we conducted a number of experiments against various synthetic data streams. We gradually increased the values of their parameters to find the optimal values: $\delta_d, \delta_w = 10^{-6}$, $d = 0.01$, $r = 1.01$, and $\lambda = 0.01$.

VII. Experimental Evaluation

A. Benchmarking Data Streams

1) Synthetic Data Streams: We generated four synthetic data streams from Sine1, Mixed, Circles and LED, which are all widely found in the literature [4], [9], [14], [28]. Each data stream consists of $100,000$ instances. A class noise of 10% was added to each stream in order to evaluate the robustness of the drift detectors against noisy data[2]. The synthetic data streams are described below.

- **Sine1** · *with abrupt drift*: It has two attributes x and y uniformly distributed on the interval $[0, 1]$. The classification function is $y = sin(x)$. Instances are classified as positive if they are under the curve, otherwise they are negative. At a drift point, the classification is reversed.
- **Mixed** · *with abrupt drift*: The dataset has two numeric attributes x and y distributed in $[0, 1]$ with two boolean attributes v and w. The instances are classified as positive if at least two of the three following conditions are satisfied: $v, w, y < 0.5+0.3*sin(3\pi x)$. The classification is reversed when drift points occur.
- **Circles** · *with gradual drift*: It has two attributes x and y distributed in $[0, 1]$. The classification function is the circle equation $(x - x_c)^2 + (y - y_c)^2 = r_c^2$ where (x_c, y_c) and r_c are the center and the radius of the circle, respectively. Instances inside the circle are classified as positive. Four different circles are employed in order to simulate concept drift.
- **LED** · *with gradual drift*: The objective of this dataset is to predict the digit on a seven-segment display, where each digit has a 10% chance of being displayed. The dataset has 7 class attributes, and 17 irrelevant ones. Concept drift is simulated by interchanging relevant attributes.

Concept Drift Simulation – Following [29], we used the sigmoid function to simulate *abrupt* and *gradual* concept drifts. The function determines the probability of belonging to a new context during a transition between two concepts. The transition length ζ allows to simulate abrupt or gradual concept drifts. The value was set to 50 for abrupt concept drifts, and to 500 for gradual concept drifts in all our experiments. To summarize, the drifts occur at every $20,000$ instances in Sine1 and Mixed with $\zeta = 50$ for *abrupt drift*, and at every $25,000$ instances in Circles and LED with $\zeta = 500$ for *gradual drift*.

2) Real-world Data Streams: We extended our experiments to real-world data streams[3]; which are frequently employed in the online learning and adaptive learning literature [4], [14], [15], [28], [29]. Three data streams were selected in our comparative study.

- **Electricity** · It contains $45,312$ instances, with 8 input attributes, recorded every half hour for two years by the Australian New South Wales Electricity. The classifier must predict a rise (*Up*) or a fall (*Down*) in the electricity price. The concept drift may result from changes in consumption habits or unexpected events [30].
- **Forest Covertype** · It consists of 54 attributes with $581,012$ instances describing 7 forest cover types for 30×30 meter cells obtained from US Forest Service (USFS) Region 2 Resource Information System (RIS) data, for 4 wilderness areas located in the Roosevelt National Forest of Northern Colorado [31].
- **Poker hand** · It is composed of $1,000,000$ instances, where each instance is an example of five cards drawn from a standard 52 cards deck. Each card is described by two attributes (suit and rank), for a total of ten predictive attributes. The classifier predicts the poker hand [32].

B. Experiment Settings

We used the MOA framework [27] for all our experiments. We selected Hoeffding Tree (HT) [26] and Naive Bayes (NB) [6], [33] as our incremental classifiers; and compared MDDMs and FHDDM with CUSUM, PageHinkley, DDM,

[2]Available at: https://www.github.com/alipsgh/data_streams/.

[3]Available at: https://moa.cms.waikato.ac.nz/datasets/2013/.

EDDM, RDDM, ADWIN, SeqDrift2, and HDDMs. The default parameters were employed for both the classifiers and the drift detection methods. The algorithms were evaluated *prequentially* which means that an instance is first tested and then used for training. Pesaranghader et al. [9] introduced the *acceptable delay length* notion for measuring detection delay and for determining true positive (TP), false positive (FP), and false negative (FN) rates. The acceptable delay length Δ is a threshold that determines how far a given alarm should be from the true location of a concept drift to be considered a true positive. Therefore, the number of true positives is incremented when the drift detector alarm occurs within the acceptable delay range. Otherwise, the number of false negatives is incremented as the alarm occurred too late. In addition, the false positive value is incremented when a false alarm occurs outside the acceptable delay range. Following this approach, we set Δ to 250 for the SINE1, MIXED, and to 1000 for the CIRCLES and LED data streams. A longer Δ should be considered for data streams with gradual drifts in order to avoid a false negative increase [9]. For FHDDM and MDDMs, the window size was set to 25 for the SINE1 and MIXED, and to 100 for the CIRCLES and LED data streams. We used a wider window for the CIRCLES and LED data streams in order to better detect gradual concept drifts. These window sizes were chosen in order to have shorter detection delay, as well as lower false positive and false negative rates. Experiments were performed on an Intel Core i5 @ 2.8 GHz with 16 GB of RAM running Apple OS X Yosemite.

C. Experiments and Discussion

1) Synthetic Data Streams: Our experimental results against the synthetic data streams are presented in Tables I to IV, and discussed below:

- **Discussion I - SINE1 and MIXED (Abrupt Drift):** As represented in Tables I and II, $HDDM_{W\text{-test}}$ and MDDMs detected concept drifts with shorter delays against SINE1 and MIXED data streams. FHDDM and MDDM resulted in the lowest false positive rates, followed by CUSUM and $HDDM_{A\text{-test}}$. This observation may indicate that FHDDM, MDDM, CUSUM, and $HDDM_{A\text{-test}}$ are more accurate. Although RDDM had shorter detection delays and false negative rates compared to DDM and EDDM, it caused higher false positive rates. EDDM and ADWIN had the highest false positive rates. Moreover, EDDM had the highest false negative rates since it could not detect concept drifts within the acceptable delay length. ADWIN showed the highest execution runtime because it compares all sub-windows of its sliding window for drift detection. SeqDrift2 had the highest memory usage, followed by RDDM and ADWIN as a considerable number of prediction results must be stored in their repositories or sliding window. MDDMs and FHDDM showed comparable results against the other methods. As shown in Table I, for the Hoeffding Tree classifier, the highest classification accuracies was obtained with MDDMs and FHDDM, since they detected drifts with the shortest delays and the lowest false positive rates. Similar observations apply to the Naive Bayes classifier. From Tables I and II, it may be noticed that the false positive rate is lower for Naive Bayes. This suggests that the Naive Bayes classifier represented the decision boundaries more accurately for noisy SINE1 and MIXED data streams.
- **Discussion II - CIRCLES and LED (Gradual Drift):** Tables III and IV show the experimental results with the Hoeffding Tree and Naive Bayes classifiers against the CIRCLES and LED data streams. The MDDM algorithms resulted in the shortest concept drift detection delays, followed by FHDDM and $HDDM_{W\text{-test}}$. Compared to FHDDM, MDDMs detected concept drifts faster because of its weighting schemes which favor the most recent elements. On the other hand, EDDM produced the longest drift detection delays. It also had the highest false negative rates. MDDMs, FHDDM, CUSUM, RDDM, and HDDMs had the highest true positive rates. EDDM and ADWIN showed the highest false positive rates against the CIRCLES data streams. ADWIN alarmed falsely for having a non-conservative test. We achieved higher accuracies with Hoeffding Tree than with Naive Bayes against the CIRCLES data stream. In the case of the LED data streams, ADWIN and SeqDrift2 triggered a relatively large number of false alarms. Although RDDM outperformed DDM in all cases, both in terms of drift detection delays and false negative rates, it showed higher false positive rates. SeqDrift2 caused fewer false positives compared to ADWIN, since it applies a more conservative test (Bernstein's inequality). ADWIN is computationally expensive since all possible sub-windows of the sliding window must be compared for drift detection. As in the previous experiments, SeqDrift2 had a high memory consumption, followed by RDDM and ADWIN. Finally, MDDMs, FHDDM, and HDDMs led to the highest accuracies.
- **Discussion III - MDDM Variants:** Frequently, MDDM-G and MDDM-E have shorter drift detection delays than MDDM-A. The reason is to be found in the fact that they both utilize an exponential weighting scheme (i.e. more weight is put on the most recent entries which are the ones required for faster detection) as opposed to MDDM-A which has a linear one. The reader will notice that the false positive rates of these two variants against the two streams with gradual change, namely CIRCLES and LED, were higher than those of MDDM-A. This is a consequence of the fact that MDDM-A put more emphasis on the older entries in the window, which, in these cases are beneficial to the learning process. All three variants had comparable levels of accuracy. MDDM-E had slightly longer runtimes, due to the overhead incurred when calculating e^{λ}. In general, one may observe that an exponential-like scheme is beneficial in scenarios when faster detection is required. It follows that the optimal shape for the weighting function is data, context and application dependent.

TABLE I: Hoeffding Tree (HT) with Drift Detectors against Synthetic Data Streams with Abrupt Change

	Detector	Delay	TP	FP	FN	Runtime	Memory	Accuracy
SINE1 (Abrupt)	MDDM-A	**38.60 ± 3.38**	4.00	**0.21 ± 0.43**	0.00	20.00 ± 4.70	232.00	**87.07 ± 0.16**
	MDDM-G	**38.56 ± 3.36**	4.00	**0.20 ± 0.42**	0.00	14.33 ± 3.47	232.00	**87.07 ± 0.16**
	MDDM-E	**38.56 ± 3.36**	4.00	**0.20 ± 0.42**	0.00	33.43 ± 5.95	232.00	**87.07 ± 0.16**
	FHDDM	40.65 ± 3.15	4.00	**0.10 ± 0.33**	0.00	10.78 ± 3.25	232.00	**87.07 ± 0.16**
	CUSUM	86.89 ± 4.47	4.00	0.24 ± 0.47	0.00	15.26 ± 4.27	128.00	86.94 ± 0.15
	PageHinkley	**229.24 ± 13.20**	2.30 ± 1.07	1.71 ± 1.08	1.70 ± 1.07	12.08 ± 3.59	136.00	86.46 ± 0.17
	DDM	163.11 ± 22.73	3.36 ± 0.77	3.30 ± 2.20	0.64 ± 0.77	15.12 ± 3.67	136.00	86.06 ± 1.34
	EDDM	**243.83 ± 14.25**	**0.22 ± 0.44**	**33.90 ± 11.61**	**3.78 ± 0.44**	10.69 ± 3.71	136.00	84.71 ± 0.55
	RDDM	93.63 ± 7.57	4.00	4.72 ± 3.58	0.00	16.01 ± 4.57	**7224.00**	86.79 ± 0.18
	ADWIN	47.32 ± 2.78	4.00	**23.77 ± 5.75**	0.00	**> 2295.0**	**> 2050.0**	86.75 ± 0.21
	SeqDrift2	**200.00**	4.00	4.83 ± 1.16	0.00	38.31 ± 6.34	**> 82270.0**	86.53 ± 0.15
	HDDM$_{\text{A-test}}$	57.62 ± 11.81	4.00	0.71 ± 0.89	0.00	38.53 ± 5.92	160.00	87.01 ± 0.16
	HDDM$_{\text{W-test}}$	**35.70 ± 2.95**	4.00	0.46 ± 0.68	0.00	32.64 ± 4.95	136.00	**87.07 ± 0.15**
MIXED (Abrupt)	MDDM-A	**38.38 ± 3.66**	4.00	**1.11 ± 1.15**	0.00	19.68 ± 4.12	232.00	**83.36 ± 0.11**
	MDDM-G	**38.28 ± 3.64**	4.00	**1.19 ± 1.21**	0.00	15.31 ± 4.06	232.00	**83.36 ± 0.11**
	MDDM-E	**38.28 ± 3.64**	4.00	**1.19 ± 1.21**	0.00	35.06 ± 6.06	232.00	**83.36 ± 0.11**
	FHDDM	40.55 ± 3.70	4.00	**0.65 ± 0.94**	0.00	11.19 ± 3.68	232.00	**83.39 ± 0.10**
	CUSUM	90.90 ± 6.13	4.00	**0.32 ± 0.58**	0.00	13.70 ± 4.11	128.00	83.27 ± 0.12
	PageHinkley	**229.91 ± 13.27**	2.26 ± 0.98	1.74 ± 0.98	1.74 ± 0.98	12.33 ± 3.74	136.00	82.88 ± 0.11
	DDM	**195.73 ± 22.12**	2.76 ± 1.01	2.91 ± 1.96	1.24 ± 1.01	14.74 ± 4.03	136.00	81.78 ± 2.06
	EDDM	**248.46 ± 7.69**	**0.05 ± 0.22**	**21.51 ± 7.70**	**3.95 ± 0.22**	11.59 ± 3.66	136.00	80.65 ± 0.82
	RDDM	106.68 ± 11.26	3.99 ± 0.10	3.49 ± 2.47	0.01 ± 0.10	15.87 ± 4.42	**7224.00**	83.16 ± 0.12
	ADWIN	52.40 ± 2.86	4.00	**20.94 ± 5.70**	0.00	**> 2315.0**	**> 2070.0**	83.29 ± 0.12
	SeqDrift2	**200.00**	4.00	4.98 ± 1.20	0.00	37.92 ± 6.10	**> 81470.0**	82.91 ± 0.11
	HDDM$_{\text{A-test}}$	69.42 ± 15.51	4.00	**1.28 ± 1.09**	0.00	38.98 ± 5.89	160.00	83.31 ± 0.11
	HDDM$_{\text{W-test}}$	**35.56 ± 3.50**	4.00	3.23 ± 1.95	0.00	31.96 ± 5.22	136.00	83.27 ± 0.12

TABLE II: Naive Bayes (NB) with Drift Detectors against Synthetic Data Streams with Abrupt Change

	Detector	Delay	TP	FP	FN	Runtime	Memory	Accuracy
SINE1 (Abrupt)	MDDM-A	**38.55 ± 3.35**	4.00	**0.13 ± 0.34**	0.00	19.04 ± 4.32	232.00	**86.08 ± 0.25**
	MDDM-G	**38.47 ± 3.35**	4.00	**0.14 ± 0.35**	0.00	14.18 ± 3.79	232.00	**86.08 ± 0.25**
	MDDM-E	**38.46 ± 3.35**	4.00	**0.14 ± 0.35**	0.00	33.37 ± 5.33	232.00	**86.08 ± 0.25**
	FHDDM	40.48 ± 3.37	4.00	**0.04 ± 0.20**	0.00	10.89 ± 3.65	232.00	**86.08 ± 0.25**
	CUSUM	83.27 ± 6.96	3.99 ± 0.10	0.71 ± 0.86	0.01 ± 0.10	13.83 ± 3.70	128.00	85.96 ± 0.25
	PageHinkley	**175.07 ± 24.72**	3.71 ± 0.50	0.35 ± 0.54	0.29 ± 0.50	11.83 ± 3.35	136.00	85.69 ± 0.27
	DDM	**179.18 ± 26.83**	2.87 ± 0.84	3.09 ± 1.88	1.13 ± 0.84	15.13 ± 3.42	136.00	82.39 ± 4.32
	EDDM	**234.28 ± 22.22**	**0.57 ± 0.64**	**33.53 ± 11.50**	**3.43 ± 0.64**	11.97 ± 3.38	136.00	83.44 ± 2.87
	RDDM	89.72 ± 16.45	3.99 ± 0.10	3.93 ± 2.91	0.01 ± 0.10	16.25 ± 4.12	**7224.00**	85.98 ± 0.27
	ADWIN	48.61 ± 2.65	4.00	**20.27 ± 5.23**	0.00	**> 2325.0**	**> 2070.0**	85.97 ± 0.24
	SeqDrift2	**200.00**	4.00	4.26 ± 0.58	0.00	35.96 ± 6.58	**> 83665.0**	85.59 ± 0.25
	HDDM$_{\text{A-test}}$	88.03 ± 25.73	3.97 ± 0.17	0.35 ± 0.55	0.03 ± 0.17	38.70 ± 6.04	160.00	85.95 ± 0.25
	HDDM$_{\text{W-test}}$	**35.52 ± 3.10**	4.00	0.41 ± 0.58	0.00	33.36 ± 5.93	136.00	**86.09 ± 0.25**
MIXED (Abrupt)	MDDM-A	**38.52 ± 3.81**	4.00	**0.69 ± 0.89**	0.00	20.30 ± 4.13	232.00	**83.37 ± 0.09**
	MDDM-G	**38.41 ± 3.81**	4.00	**0.70 ± 0.89**	0.00	15.32 ± 4.16	232.00	**83.37 ± 0.09**
	MDDM-E	**38.41 ± 3.81**	4.00	**0.70 ± 0.89**	0.00	34.25 ± 5.94	232.00	**83.37 ± 0.09**
	FHDDM	40.56 ± 3.72	4.00	**0.25 ± 0.48**	0.00	11.50 ± 3.44	232.00	**83.38 ± 0.08**
	CUSUM	88.23 ± 8.97	3.99 ± 0.10	**0.35 ± 0.54**	0.01 ± 0.10	13.59 ± 3.71	128.00	83.27 ± 0.08
	PageHinkley	**198.79 ± 18.72**	3.56 ± 0.65	0.44 ± 0.65	0.44 ± 0.65	11.67 ± 3.47	136.00	82.97 ± 0.10
	DDM	**192.99 ± 23.82**	2.78 ± 1.00	2.41 ± 1.44	1.22 ± 1.00	15.48 ± 4.56	136.00	80.28 ± 4.11
	EDDM	**247.47 ± 8.60**	**0.11 ± 0.31**	**20.22 ± 7.66**	**3.89 ± 0.31**	11.55 ± 2.89	136.00	80.30 ± 2.32
	RDDM	104.97 ± 12.06	3.99 ± 0.10	1.86 ± 1.65	0.01 ± 0.10	17.75 ± 3.77	**7224.00**	83.24 ± 0.09
	ADWIN	52.35 ± 2.88	4.00	**20.27 ± 5.11**	0.00	**> 2320.0**	**> 2070.0**	83.31 ± 0.08
	SeqDrift2	**200.00**	4.00	4.39 ± 0.79	0.00	38.08 ± 5.80	**> 83520.0**	82.91 ± 0.08
	HDDM$_{\text{A-test}}$	83.71 ± 19.46	3.96 ± 0.20	**0.48 ± 0.64**	0.04 ± 0.20	38.47 ± 5.24	160.00	83.28 ± 0.09
	HDDM$_{\text{W-test}}$	**35.75 ± 3.94**	4.00	1.77 ± 1.39	0.00	32.00 ± 5.33	136.00	**83.36 ± 0.09**

TABLE III: Hoeffding Tree (HT) with Drift Detectors against Synthetic Data Streams with Gradual Change

Dataset	Detector	Delay	TP	FP	FN	Runtime	Memory	Accuracy
CIRCLES (Gradual)	MDDM-A	**71.98 ± 22.19**	3.00	**0.27 ± 0.51**	0.00	46.42 ± 6.24	528.00	**86.58 ± 0.16**
	MDDM-G	**69.42 ± 22.09**	3.00	**0.36 ± 0.61**	0.00	30.60 ± 5.24	528.00	**86.58 ± 0.17**
	MDDM-E	**69.52 ± 22.12**	3.00	**0.37 ± 0.61**	0.00	50.25 ± 5.98	528.00	**86.57 ± 0.17**
	FHDDM	79.28 ± 20.64	3.00	**0.17 ± 0.40**	0.00	15.15 ± 4.06	528.00	**86.58 ± 0.13**
	CUSUM	220.07 ± 31.79	2.99 ± 0.10	**0.04 ± 0.20**	0.01 ± 0.10	14.30 ± 4.16	128.00	86.51 ± 0.13
	PageHinkley	**855.37 ± 56.27**	1.79 ± 0.45	1.24 ± 0.47	1.21 ± 0.45	11.87 ± 3.40	136.00	85.96 ± 0.15
	DDM	487.97 ± 82.24	2.78 ± 0.52	1.41 ± 1.24	0.22 ± 0.52	15.71 ± 4.51	136.00	86.21 ± 0.47
	EDDM	**987.61 ± 54.35**	**0.07 ± 0.26**	**24.61 ± 14.48**	**2.93 ± 0.26**	10.68 ± 3.49	136.00	84.89 ± 0.29
	RDDM	293.80 ± 38.52	2.98 ± 0.14	0.79 ± 1.25	0.02 ± 0.14	17.71 ± 4.30	**7224.00**	86.46 ± 0.16
	ADWIN	214.47 ± 137.38	2.66 ± 0.49	**19.46 ± 6.03**	0.34 ± 0.49	**> 2420.0**	**> 2140.0**	85.64 ± 0.18
	SeqDrift2	202.67 ± 16.11	3.00	3.08 ± 0.90	0.00	37.87 ± 6.21	**> 104370.0**	86.47 ± 0.14
	$HDDM_{A\text{-test}}$	111.96 ± 68.22	2.96 ± 0.20	0.65 ± 0.92	0.04 ± 0.20	39.19 ± 6.10	160.00	86.52 ± 0.20
	$HDDM_{W\text{-test}}$	94.03 ± 57.61	2.98 ± 0.14	0.73 ± 0.87	0.02 ± 0.14	34.23 ± 5.67	136.00	86.53 ± 0.18
$LED_{0.3.1.3}$ (Gradual)	MDDM-A	**210.31 ± 73.05**	2.98 ± 0.14	**0.03 ± 0.17**	0.02 ± 0.14	55.26 ± 7.71	528.00	**89.56 ± 0.04**
	MDDM-G	**208.65 ± 73.05**	2.98 ± 0.14	**0.03 ± 0.17**	0.02 ± 0.14	30.53 ± 5.77	528.00	**89.56 ± 0.04**
	MDDM-E	**208.61 ± 73.05**	2.98 ± 0.14	**0.03 ± 0.17**	0.02 ± 0.14	48.92 ± 7.79	528.00	**89.56 ± 0.04**
	FHDDM	220.40 ± 76.00	2.97 ± 0.22	**0.03 ± 0.22**	0.03 ± 0.22	16.70 ± 3.96	528.00	**89.56 ± 0.04**
	CUSUM	300.68 ± 50.30	3.00	**0.00**	0.00	12.11 ± 3.55	128.00	**89.56 ± 0.03**
	PageHinkley	**560.30 ± 79.43**	2.95 ± 0.26	0.04 ± 0.24	0.05 ± 0.26	10.37 ± 3.21	136.00	89.35 ± 0.04
	DDM	444.13 ± 79.82	2.97 ± 0.17	0.32 ± 0.58	0.03 ± 0.17	13.39 ± 3.66	136.00	89.47 ± 0.56
	EDDM	**954.97 ± 62.98**	**0.66 ± 0.71**	5.97 ± 1.69	**2.34 ± 0.71**	9.99 ± 3.11	136.00	88.33 ± 0.50
	RDDM	321.88 ± 50.94	2.98 ± 0.14	0.61 ± 0.96	0.02 ± 0.14	15.70 ± 4.14	**7224.00**	**89.63 ± 0.04**
	ADWIN	**541.71 ± 213.81**	2.45 ± 0.67	**718.85 ± 24.77**	0.55 ± 0.67	**> 1170.0**	**> 1390.0**	**72.14 ± 0.44**
	SeqDrift2	426.00 ± 173.31	2.78 ± 0.44	**277.06 ± 47.48**	0.22 ± 0.44	37.58 ± 6.13	**> 15260.0**	**76.51 ± 2.28**
	$HDDM_{A\text{-test}}$	295.03 ± 85.29	2.98 ± 0.20	0.16 ± 0.44	0.02 ± 0.20	34.89 ± 6.83	160.00	**89.58 ± 0.05**
	$HDDM_{W\text{-test}}$	259.18 ± 87.25	2.95 ± 0.26	0.08 ± 0.31	0.05 ± 0.26	28.86 ± 5.42	136.00	**89.56 ± 0.04**

TABLE IV: Naive Bayes (NB) with Drift Detectors against Synthetic Data Streams with Gradual Change

Dataset	Detector	Delay	TP	FP	FN	Runtime	Memory	Accuracy
CIRCLES (Gradual)	MDDM-A	**161.25 ± 87.26**	2.95 ± 0.22	**0.63 ± 0.70**	0.05 ± 0.22	50.12 ± 7.32	528.00	**84.14 ± 0.12**
	MDDM-G	**161.73 ± 89.49**	2.94 ± 0.24	**0.80 ± 0.73**	0.06 ± 0.24	31.83 ± 5.98	528.00	**84.14 ± 0.12**
	MDDM-E	**161.74 ± 89.49**	2.94 ± 0.24	**0.81 ± 0.73**	0.06 ± 0.24	49.84 ± 6.40	528.00	**84.14 ± 0.12**
	FHDDM	166.13 ± 83.84	2.96 ± 0.20	**0.43 ± 0.60**	0.04 ± 0.20	15.91 ± 4.47	528.00	84.14 ± 0.13
	CUSUM	299.78 ± 52.29	3.00	**0.40 ± 0.62**	0.00	14.45 ± 4.15	128.00	84.08 ± 0.12
	PageHinkley	**677.32 ± 76.30**	2.11 ± 0.55	0.93 ± 0.53	0.89 ± 0.55	12.61 ± 3.91	136.00	83.94 ± 0.13
	DDM	**703.59 ± 122.67**	**1.92 ± 0.72**	2.33 ± 1.49	1.08 ± 0.72	16.06 ± 4.44	136.00	83.18 ± 1.61
	EDDM	**938.27 ± 106.60**	**0.35 ± 0.50**	**31.09 ± 18.14**	**2.65 ± 0.50**	11.58 ± 3.22	136.00	83.12 ± 0.40
	RDDM	406.50 ± 69.40	2.99 ± 0.10	2.15 ± 1.94	0.01 ± 0.10	17.76 ± 4.31	**7224.00**	84.05 ± 0.11
	ADWIN	200.80 ± 63.23	2.98 ± 0.14	**16.58 ± 5.02**	0.02 ± 0.14	**> 2420.0**	**> 2150.0**	84.12 ± 0.12
	SeqDrift2	276.67 ± 91.10	2.92 ± 0.27	2.49 ± 0.97	0.08 ± 0.27	36.36 ± 4.88	**> 106125.0**	**84.13 ± 0.14**
	$HDDM_{A\text{-test}}$	306.91 ± 107.78	2.91 ± 0.29	**0.49 ± 0.69**	0.09 ± 0.29	39.85 ± 6.17	160.00	84.09 ± 0.12
	$HDDM_{W\text{-test}}$	242.43 ± 134.19	2.73 ± 0.44	1.59 ± 1.00	0.27 ± 0.44	33.03 ± 5.80	136.00	84.11 ± 0.13
LED (Gradual)	MDDM-A	**210.31 ± 73.05**	2.98 ± 0.14	**0.03 ± 0.17**	0.02 ± 0.14	54.31 ± 7.38	528.00	**89.57 ± 0.04**
	MDDM-G	**208.65 ± 73.05**	2.98 ± 0.14	**0.03 ± 0.17**	0.02 ± 0.14	29.78 ± 5.33	528.00	**89.57 ± 0.04**
	MDDM-E	**208.61 ± 73.05**	2.98 ± 0.14	**0.03 ± 0.17**	0.02 ± 0.14	49.31 ± 6.52	528.00	**89.57 ± 0.04**
	FHDDM	220.40 ± 76.00	2.97 ± 0.22	**0.03 ± 0.22**	0.03 ± 0.22	15.90 ± 3.99	528.00	89.57 ± 0.04
	CUSUM	300.61 ± 50.30	3.00	**0.00**	0.00	11.90 ± 3.51	128.00	89.57 ± 0.03
	PageHinkley	**559.27 ± 78.99**	2.95 ± 0.26	**0.04 ± 0.24**	0.05 ± 0.26	9.89 ± 2.73	136.00	89.36 ± 0.04
	DDM	446.23 ± 82.12	2.96 ± 0.20	0.33 ± 0.58	0.04 ± 0.20	12.57 ± 2.98	136.00	89.29 ± 1.15
	EDDM	**949.61 ± 68.94**	**0.70 ± 0.73**	6.33 ± 1.96	**2.30 ± 0.73**	9.35 ± 3.14	136.00	88.32 ± 0.53
	RDDM	321.80 ± 50.94	2.98 ± 0.14	0.61 ± 0.96	0.02 ± 0.14	15.66 ± 4.30	**7224.00**	**89.63 ± 0.04**
	ADWIN	**550.01 ± 211.60**	2.36 ± 0.66	**697.51 ± 19.71**	0.64 ± 0.66	**> 1180.0**	**> 1400.00**	**72.72 ± 0.47**
	SeqDrift2	445.33 ± 192.27	2.75 ± 0.46	**278.82 ± 47.50**	0.25 ± 0.46	36.20 ± 5.71	**> 14790.0**	**76.54 ± 2.25**
	$HDDM_{A\text{-test}}$	295.85 ± 83.23	2.98 ± 0.20	0.17 ± 0.47	0.02 ± 0.20	34.56 ± 6.34	160.00	**89.58 ± 0.05**
	$HDDM_{W\text{-test}}$	259.17 ± 87.21	2.95 ± 0.26	**0.05 ± 0.22**	0.05 ± 0.26	29.22 ± 6.05	136.00	**89.56 ± 0.03**

2) Real-world Data Streams: There is a consensus among researchers that the locations and/or the presence of concept drift in the ELECTRICITY, FOREST COVERTYPE, and POKER HAND data streams are not known [4], [9], [20], [29]. This implies, in turn, that the drift detection delay as well as the false positive and false negative rates cannot be determined since the knowledge of the drift locations is necessary in order to evaluate these quantities. Consequently, our evaluation is based on the overall accuracy and the number of alarms for concept drifts issued by each drift detector. We have also considered *blind adaptation* and *no detection* approaches as benchmarks for our experiments. In the *blind adaptation*, the classifier is retrained ab initio at every 100 instances. The classifiers are trained without drift detectors in the case of *no detection*. A window of size 25 was selected for both MDDM and FHDDM; our experiments have shown that this choice is optimal in terms of accuracy.

Table V presents the experimental results for ELECTRICITY, FOREST COVERTYPE, and POKER HAND data streams with the Hoeffding Tree (HT) and Naive Bayes (NB) classifiers. Firstly, the Hoeffding Tree classifier showed higher classification accuracies compared to Naive Bayes when executed without drift detector. This suggests that the Hoeffding Tree classifier could branch out and adequately reflect the new patterns. Secondly, both classifiers achieved higher classification accuracies by using drift detection. Although this observation indicates that using drift detection methods is beneficial compared to the *no detection* case, it does not necessarily mean that a drift detector outperforms the others. Indeed, in a recent study by Bifet et al. [10], it was found that blind detection has the highest classification accuracies, against the ELECTRICITY and FOREST COVERTYPE data streams. Based on multiple experiments, Bifet et al. [10] concluded that this behavior may be explained by the temporal dependencies in between the various instances of the streams. As shown in Table V, a drift detection method with a higher number of alarms usually led to a higher classification accuracy. In such a case, a classifier learns from a small portion of the data stream where almost all instances are labeled with a common label (this refers to temporal dependencies among examples as stated by Bifet et al. [10]). To support this observation, as mentioned earlier, we considered a blind adaptation as a benchmark. As shown in the same table, the blind adaptation led to the highest or the second highest classification accuracies. We further extended our experiments by running MDDMs and FHDDM with higher values of δ_w and δ_d. Recall that a higher δ implies that the drift detection technique is less conservative. As indicated in the table, as MDDMs and FHDDM became less conservative, the number of alarms as well as the classification accuracies increased. Therefore, because of temporal dependencies, both classifiers repeatedly learned from instances presenting the same labels between two consecutive alarms.

In summary, we concluded that using drift detection methods against real-world data streams is beneficial. Nevertheless, we are not in a position to make a strong statement based solely on the accuracy because (1) the location of the drift is unknown, and (2) because of the temporal dependencies in between instances [10]. HDDM and FHDDM consistently led to higher classification accuracies. Particularly, MDDM achieved the highest classification accuracies in all cases when the value of δ_w increased from 10^{-6} to 0.001 and 0.01.

VIII. CONCLUSION

Sensor networks, smart houses, intelligent transportation, autopilots are examples of technologies operating in evolving environments where experiencing concept drifts over time is commonplace. In order for the learning process to be more efficient, concept drifts should be detected rapidly with false negative rate as small as possible. In this research paper, we introduced the McDiarmid Drift Detection Methods (MDDMs) for detecting concept drifts with shorter delays and lower false negative rates. We conducted various experiments to compare MDDMs against the state-of-the-art. Our experimental results indicated that MDDMs outperformed existing methods in terms of drift detection delay, false negative, false positive, and classification accuracy rates.

In this paper, we considered incremental learning against a single stream; and accordingly evaluated the drift detection methods. However, in most real-world applications, including sensor networks, one must analyze concurrently multiple data streams. Therefore, we plan to extend our methods to multiple data streams by applying Data Stream Processing (DSP) systems [34]. Further, we aim to investigate streams with heterogeneous concept drifts, i.e. where different drift types and rates overlap. We also interested to assess the performance of drift detection methods against time-series data streams.

ACKNOWLEDGMENT

The authors wish to acknowledge financial support by the Canadian Natural Sciences and Engineering Research Council (NSERC) as well as the Ontario Trillium Scholarship (OTS).

REFERENCES

[1] C. McDiarmid, "On the method of bounded differences," *Surveys in combinatorics*, vol. 141, no. 1, pp. 148–188, 1989.

[2] J. Gama and M. M. Gaber, *Learning from data streams: processing techniques in sensor networks*. Springer, 2007.

[3] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys (CSUR)*, vol. 46, no. 4, p. 44, 2014.

[4] I. Frías-Blanco, J. del Campo-Ávila, G. Ramos-Jiménez, R. Morales-Bueno, A. Ortiz-Díaz, and Y. Caballero-Mota, "Online and non-parametric drift detection methods based on hoeffdings bounds," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 3, pp. 810–823, 2015.

[5] J. Gama, R. Sebastião, and P. P. Rodrigues, "Issues in evaluation of stream learning algorithms," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 329–338.

[6] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "Data stream mining a practical approach," 2011.

[7] I. Žliobaitė, "Learning under concept drift: an overview," *arXiv preprint arXiv:1010.4784*, 2010.

[8] R. Pears, S. Sakthithasan, and Y. S. Koh, "Detecting concept change in dynamic data streams," *Machine Learning*, vol. 97, no. 3, pp. 259–293, 2014.

TABLE V: Hoeffding Tree (HT) and Naive Bayes (NB) against Real-world Data Stream

	Detector	Electricity HT Alarms	Electricity HT Acc.	Electricity NB Alarms	Electricity NB Acc.	Forest Covertype HT Alarms	Forest Covertype HT Acc.	Forest Covertype NB Alarms	Forest Covertype NB Acc.	Poker hand HT Alarms	Poker hand HT Acc.	Poker hand NB Alarms	Poker hand NB Acc.
	MDDM-A	105	84.60	126	83.47	1963	85.33	2022	85.38	2036	76.89	2145	76.83
	MDDM-G	105	84.60	126	83.47	1966	85.35	2025	85.39	2034	76.89	2149	76.83
	MDDM-E	105	84.60	126	83.47	1966	85.35	2025	85.39	2034	76.89	2149	76.83
	FHDDM	90	84.59	109	83.13	1794	85.08	185	85.09	1876	76.72	1928	76.68
	CUSUM	22	81.71	28	79.21	226	83.01	286	81.55	617	72.85	659	72.54
	PageHinkley	6	81.95	10	78.04	90	81.65	117	80.06	403	71.30	489	70.67
	DDM	169	**85.41**	143	81.18	4301	**87.35**	4634	**88.03**	1046	72.74	433	61.97
	EDDM	191	84.91	203	**84.83**	2466	86.00	2416	86.08	4806	**77.30**	4863	**77.48**
	RDDM	143	85.18	164	84.19	2671	86.42	2733	86.86	2512	76.70	2579	76.67
	ADWIN	110	83.40	128	81.63	2341	83.68	2636	83.59	2373	74.56	2453	74.60
	SeqDrift2	59	82.83	60	79.68	710	82.85	757	82.44	1322	72.51	1395	72.25
	$\text{HDDM}_{\text{A-test}}$	210	**85.71**	211	**84.92**	3695	87.24	3284	87.42	2565	76.40	2615	76.48
	$\text{HDDM}_{\text{W-test}}$	117	85.06	132	84.09	2342	85.97	2383	86.22	2211	77.11	2312	77.11
	$\text{Blind}_{\|w\| = 100}$	453	84.26	453	84.82	5810	**87.24**	5810	**87.70**	8292	**77.96**	8292	**78.18**
	No Detection	—	79.20	—	73.36	—	80.31	—	60.52	—	76.07	—	59.55
$\delta = 0.001$	MDDM-A	180	**85.79**	208	**85.00**	3253	**87.03**	3221	**87.27**	4320	**77.82**	4378	**78.03**
$\delta = 0.001$	MDDM-G	182	**85.78**	209	**85.01**	3270	**87.05**	3231	**87.29**	4370	**77.83**	4425	**78.05**
$\delta = 0.001$	MDDM-E	182	**85.78**	209	**85.01**	3270	**87.06**	3234	**87.29**	4369	**77.83**	4427	**78.06**
$\delta = 0.001$	FHDDM	170	85.67	193	84.83	3067	86.74	3027	86.99	4010	77.73	4082	77.87
$\delta = 0.01$	MDDM-A	256	**85.86**	265	**85.60**	3884	**87.63**	3791	**87.95**	6075	**78.18**	6099	**78.51**
$\delta = 0.01$	MDDM-G	252	**85.98**	265	**85.78**	3969	**87.73**	3856	**88.05**	6095	**78.21**	6118	**78.54**
$\delta = 0.01$	MDDM-E	252	**85.97**	266	**85.77**	3980	**87.73**	3864	**88.05**	6091	**78.21**	6116	**78.54**
$\delta = 0.01$	FHDDM	251	**85.86**	261	**85.65**	3869	**87.63**	3780	**87.95**	6035	**78.16**	6092	**78.49**

[9] A. Pesaranghader and H. L. Viktor, “Fast hoeffding drift detection method for evolving data streams,” in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2016, pp. 96–111.

[10] A. Bifet, “Classifier concept drift detection and the illusion of progress,” in *International Conference on Artificial Intelligence and Soft Computing*. Springer, 2017, pp. 715–725.

[11] I. Žliobaitė, M. Budka, and F. Stahl, “Towards cost-sensitive adaptation: when is it worth updating your predictive model?” *Neurocomputing*, vol. 150, pp. 240–249, 2015.

[12] E. Page, “Continuous inspection schemes,” *Biometrika*, vol. 41, no. 1/2, pp. 100–115, 1954.

[13] S. Roberts, “Control chart tests based on geometric moving averages,” *Technometrics*, vol. 42, no. 1, pp. 97–101, 2000.

[14] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, “Learning with drift detection,” in *Brazilian Symposium on Artificial Intelligence*. Springer, 2004, pp. 286–295.

[15] M. Baena-Garcıa, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavalda, and R. Morales-Bueno, “Early drift detection method,” in *Fourth international workshop on knowledge discovery from data streams*, vol. 6, 2006, pp. 77–86.

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

[17] R. S. Barros, D. R. Cabral, S. G. Santos *et al.*, “Rddm: Reactive drift detection method,” *Expert Systems with Applications*, 2017.

[18] A. Bifet and R. Gavalda, “Learning from time-changing data with adaptive windowing,” in *SDM*, vol. 7. SIAM, 2007, p. 2007.

[19] R. Sebastião, J. Gama, and T. Mendonça, “Fading histograms in detecting distribution and concept changes,” *International Journal of Data Science and Analytics*, pp. 1–30, 2017.

[20] D. T. J. Huang, Y. S. Koh, G. Dobbie, and A. Bifet, “Drift detection using stream volatility,” in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2015, pp. 417–432.

[21] T. M. Mitchell, “Machine learning,” 1997.

[22] J. S. Vitter, “Random sampling with a reservoir,” *ACM Transactions on Mathematical Software (TOMS)*, vol. 11, no. 1, pp. 37–57, 1985.

[23] S. Bernstein, “The theory of probabilities,” 1946.

[24] W. Hoeffding, “Probability inequalities for sums of bounded random variables,” *Journal of the American statistical association*, vol. 58, no. 301, pp. 13–30, 1963.

[25] R. Klinkenberg, “Learning drifting concepts: Example selection vs. example weighting,” *Intelligent Data Analysis*, vol. 8, no. 3, pp. 281–300, 2004.

[26] P. Domingos and G. Hulten, “Mining high-speed data streams,” in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2000, pp. 71–80.

[27] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, “Moa: Massive online analysis,” *Journal of Machine Learning Research*, vol. 11, no. May, pp. 1601–1604, 2010.

[28] A. Pesaranghader, H. L. Viktor, and E. Paquet, “A framework for classification in data streams using multi-strategy learning,” in *International Conference on Discovery Science*. Springer, 2016, pp. 341–355.

[29] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavaldà, “New ensemble methods for evolving data streams,” in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 139–148.

[30] I. Žliobaitė, “How good is the electricity benchmark for evaluating concept drift adaptation,” *arXiv preprint arXiv:1301.3524*, 2013.

[31] J. A. Blackard and D. J. Dean, “Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables,” *Computers and electronics in agriculture*, vol. 24, no. 3, pp. 131–151, 1999.

[32] M. K. Olorunnimbe, H. L. Viktor, and E. Paquet, “Dynamic adaptation of online ensembles for drifting data streams,” *Journal of Intelligent Information Systems*, pp. 1–23, 2017.

[33] P. Langley, W. Iba, and K. Thompson, “An analysis of bayesian classifiers,” in *AAAI*, vol. 90, 1992, pp. 223–228.

[34] S. Zhang, B. He, D. Dahlmeier, A. C. Zhou, and T. Heinze, “Revisiting the design of data stream processing systems on multi-core processors,” in *Data Engineering (ICDE), 2017 IEEE 33rd International Conference on*. IEEE, 2017, pp. 659–670.