Fast Hoeffding Drift Detection Method for Evolving Data Streams

Ali Pesaranghader(✉) and Herna L. Viktor

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

Abstract. Decision makers increasingly require near-instant models to make sense of fast evolving data streams. Learning from such evolving environments is, however, a challenging task. This challenge is partially due to the fact that the distribution of data often changes over time, thus potentially leading to degradation in the overall performance. In particular, classification algorithms need to adapt their models after facing such distributional changes (also referred to as concept drifts). Usually, drift detection methods are utilized in order to accomplish this task. It follows that detecting concept drifts as soon as possible, while resulting in fewer false positives and false negatives, is a major objective of drift detectors. To this end, we introduce the Fast Hoeffding Drift Detection Method (FHDDM) which detects the drift points using a sliding window and Hoeffding's inequality. FHDDM detects a drift when a significant difference between the maximum probability of correct predictions and the most recent probability of correct predictions is observed. Experimental results confirm that FHDDM detects drifts with less detection delay, less false positive and less false negative, when compared to the state-of-the-art.

Keywords: Data Stream Mining, Concept Drift, Hoeffding's Inequality, Evolving Environments.

1 Introduction

Learning in evolving environments is a challenging task. This difficulty is caused, not only by the speed and volume of data arrival, but also by the changes in the distribution that may occur. Intuitively, distributional changes may cause degradation in the performance of classification models. To this end, adaptive learning algorithms utilize drift detection methods to detect such changes and then take appropriate actions [1]. Typically, classification models are updated or, alternatively, retrained when a drift has been detected. Alternatively, ensemble learning algorithms are employed in an attempt to maintain the accuracy [2-6].

It follows that, in such a setting, drift detection methods resulting in fewer false positives and less false negatives are preferred. Such detectors should also

detect drifts as soon as they arrive. A drift detector with a high false positive number (or rate) causes frequent retraining, leading to more resources being used [7, 8]. On the other hand, a drift detector with a high false negative number causes decay in the accuracy of classification, since it does not detect drift points. These types of oversights are costly and should be avoided in many applications, e.g. in fraud detection and emergency response settings. Moreover, drift detectors should detect drifts with the least possible delay. Correct approximation of a drift point, i.e. detecting the drift with less delay, is necessary because it helps not only to make maximum usage from the data, but also aids us to realize how the drift happened. Such insights are crucial in Business Intelligence (BI) applications. Accordingly, *false positive*, *false negative* and *detection delay* are considered as evaluation measures for drift detection methods [23, 24].

We introduce the Fast Hoeffding Drift Detection Method (FHDDM) based on our requirement that the accuracy of classification models should stay steady, or increase, as more instances are processed. Otherwise, the degradation in accuracy may indicate that we face concept drifts. The FHDDM algorithm, in a novel way, uses a sliding window and Hoeffding's inequality [9] to calculate and compare the maximum probability of correct predictions observed so far with the most recent probability of correct predictions for the purpose of drift detection. We will show that the FHDDM algorithm results in less detection delay, less false positive and less false negative, when compared to the state-of-the-art.

The remainder of this paper is organized as follows: We talk about related works to concept drift detection in Section 2. We describe the Fast Hoeffding Drift Detection Method (FHDDM) algorithm in Section 3. Section 4 presents an approach for evaluating drift detectors on the basis of detection delay. We conduct our experiments on synthetic and real-world datasets in Section 5. Finally, we conclude the paper and discuss future works in Section 6.

2 Related Works

Gama et al. [1] classified concept drift detectors into three general groups of: 1) *Sequential Analysis based Methods*: These methods sequentially evaluate prediction results as they become available, and they alarm for drifts when a predefined threshold is met. The Cumulative Sum (CUSUM) [10] and Geometric Moving Average (GMA) [11] are members of this group. 2) *Statistical based Methods*: These methods probe the statistical parameters such as mean and standard deviation of prediction results to detect drifts in a stream. The Drift Detection Method (DDM) [12], Early Drift Detection Method (EDDM) [13] and Exponentially Weighted Moving Average (EWMA) [14] are placed in this group. 3) *Windows based Methods*: They usually use a fixed reference window summarizing the past information and a sliding window summarizing the most recent information. A significant difference between the distributions of these windows suggests the occurrence of a drift. Statistical tests or mathematical inequalities, with the null-hypothesis saying that the distributions are equal, can be used to decide the level of difference. Kifer's [15], Nishida's [16], Bach's [17], the Adaptive

Windowing (ADWIN) [18], the Hoeffding Drift Detection Methods ($\text{HDDM}_{\text{A-test}}$ and $\text{HDDM}_{\text{W-test}}$) [19], and SeqDrift detectors [23, 24] are members of this group. As discussed in [1], drift detectors in the second and third groups have shown better performances and have been frequently considered as benchmarks in the literature [7, 13, 16, 18, 19]. We will, thus, compare our FHDDM with DDM, EDDM, ADWIN, $\text{HDDM}_{\text{A-test}}$ and $\text{HDDM}_{\text{W-test}}$. We describe each one below:

DDM: Drift Detection Method – DDM, by Gama et al. [12], monitors the error-rate of the classification model to detect drifts. On the basis of PAC learning model [20], 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. Consider p_t as the error-rate of the classifier with 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 it detects a drift when $p_t + s_t \geq p_{min} + 3 * s_{min}$. The p_{min} and s_{min} are reset in the case of drift detection.

EDDM: Early Drift Detection Method – EDDM, by Baena-Garcia et al. [13], checks the distances between wrong predictions to detect concept drifts. The algorithm is based on the observation that facing a drift is likely 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 $p'_t + 2 * s'_t > p'_{max} + 2 * s'_{max}$. It warns for a drift if $(p'_t + 2 * s'_t)/(p'_{max} + 2 * s'_{max}) < \alpha$, and it detects a drift if $(p'_t + 2 * s'_t)/(p'_{max} + 2 * s'_{max}) < \beta$. They set α and β to 0.95 and 0.90 respectively. The p'_{max} and s'_{max} are reset if a drift is detected.

ADWIN: Adaptive Sliding Window – ADWIN, by Bifet et al. [18], slides the window w on the results of predictions to detect drifts. It examines two large enough sub-windows, i.e. w_0 with size n_0 and w_1 with size n_1, of w for drift detection where $w_0 \cdot w_1 = w$. A significant difference between the means of two sub-windows suggests a concept drift, i.e. $|\mu_{w_0} - \mu_{w_1}| \geq \varepsilon$ where $\varepsilon = \sqrt{\frac{1}{2m} \ln \frac{4}{\delta'}}$, m is the harmonic mean of n_0 and n_1, $\delta' = \delta/n$, δ is the confidence level and n is the size of window w. After a drift detection, elements are dropped from the tail of the window until no significant difference is seen.

$\text{HDDM}_{\text{A-test}}$ $\diamond$ $\text{HDDM}_{\text{W-test}}$ – $\text{HDDM}_{\text{A-test}}$ and $\text{HDDM}_{\text{W-test}}$ are proposed by Frias-Blanco et al. [19]. The former compares the moving averages to detect drifts. The latter uses the EMWA forgetting scheme [14] to weight the moving averages. Then, weighted moving averages are compared to detects the drift. For both cases, Hoeffding's inequality [9] is used to set an upper bound to the level of difference between averages. The authors noted that the first and the second methods are ideal for detecting abrupt and gradual drifts, respectively.

The pros and cons of all methods will be discussed in more details in Section 5. However, during our preliminary experiments, we observed that the aforementioned methods may cause high numbers of false positives and false negatives. Some resulted in long detection delays, though they had short detection runtimes. In the next section, we introduce our Fast Hoeffding Drift Detection Method (FHDDM), developed to address these shortcomings.

3 Fast Hoeffding Drift Detection Method

We present our Fast Hoeffding Drift Detection Method (FHDDM) which uses Hoeffding's inequality [9] to detect drifts in evolving data streams[1]. The FHDDM algorithm slides a window with a size of n on the classification results. Subsequently, it inserts a 1 into the window if the prediction result is *true*, otherwise it inserts 0. As inputs are processed, it calculates the probability of observing 1s, i.e. p_t^1, in the sliding window at time t, and also keeps the maximum probability of 1s occurring, i.e. p_{max}^1. Equation (1) shows if the value of p_1 at time t is greater than the value of p_{max}^1 then the value of p_{max}^1 will be updated.

$$if\ p_{max}^1 < p_t^1 \Rightarrow p_t^1 \rightarrow p_{max}^1 \tag{1}$$

On the basis of the probably approximately correct (PAC) learning model [20], the accuracy of classification would increase or stay steady as the number of instances increases; otherwise the possibility of facing drifts increases [12]. Thus, the value of p_{max}^1 should increase or remain steady as we process instances. In other words, the possibility of facing a concept drift increases if p_{max}^1 does not change and p_t^1 decreases over time. Eventually, as in Equation (2), a significant difference between p_{max}^1 and p_t^1 indicates the occurrence of a drift in the stream.

$$\Delta p = p_{max}^1 - p_t^1 \geq \varepsilon_d \Rightarrow Drift := True \tag{2}$$

We use the Hoeffding inequality to define the value of ε_d, Equation (4). The Hoeffding inequality has a very attractive property that it is independent of the probability distribution generating the data [9, 19, 21]. That is, it assigns an upper bound for the deviation between the mean of n random variables and its expected value.

Hoeffding's Inequality Theorem: Let $X_1, X_2, ..., X_n$ be n independent random variables such that $X_i \in [0, 1]$, then with probability at most δ, the difference between the empirical mean $\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$ and the true mean $E[\overline{X}]$ is at least ε_H, i.e. $Pr(|\overline{X} - E[\overline{X}]| \geq \varepsilon_H) \leq \delta$, where:

$$\varepsilon_H = \sqrt{\frac{1}{2n} \ln \frac{2}{\delta}} \tag{3}$$

[1] The source code is available online: https://github.com/alipsgh/

Corollary (FHDDM test): In a stream setting, assume p_t^1 is the probability of observing 1s in a sequence of n random entries, each in $\{0, 1\}$, at time t, and p_{max}^1 is the maximum probability observed so far. Let $\Delta p = p_{max}^1 - p_t^1 \geq 0$ be the difference between those two probabilities. Then, given the desired δ, i.e. the probability of error allowed, Hoeffding's inequality guarantees a drift has happened if $\Delta p \geq \varepsilon_d$, where:

$$\varepsilon_d = \sqrt{\frac{1}{2n} \ln \frac{1}{\delta}} \tag{4}$$

Figure 1 depicts an illustrative example of the FHDDM algorithm. In this example, n and δ are set to 10 and 0.2, respectively. Using Equation (4), the value of ε_d will be equal to 0.28. In this example, a real drift occurs right after the 12^{th} instance. The values of p^1 and p_{max}^1 are null and zero until 10 elements are inserted into the window. We have seven 1s in the window after reading the first 10 elements, and so the p_{10}^1 is equal to 0.7. The value of p_{max}^1 is set to 0.7 too. The 1^{st} element is dropped out from the window before the 11^{th} prediction status is inserted. Since the value of prediction status is 0, the value of p^1 decreases to 0.6. The value of p_{max}^1 stays the same, because it is greater than the current p^1. This progress is continued until the 18^{th} is inserted. At this moment the difference between p_{max}^1 and p_{18}^1 becomes more than the value of ε_d. In this case, the FHDDM algorithm alarms for a drift.

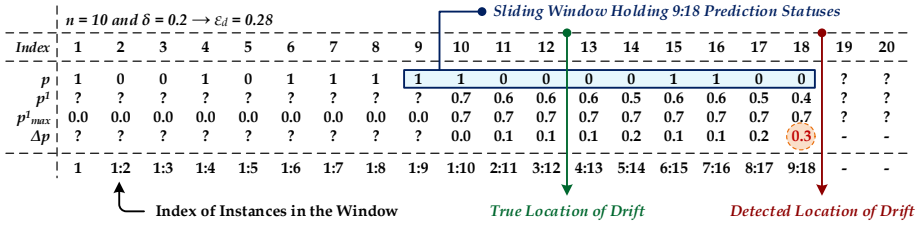

Fig. 1. Illustration of How FHDDM Works

We present the pseudocode of the FHDDM approach in Algorithm 1. First, we need to instantiate an object from FHDDM and then call its DETECT function. The result of prediction, i.e. p, is sent to the DETECT function as an input in order to determine whether a drift has occurred, in line 11. The oldest element is dropped out from the sliding window if it is full; then, a new element is pushed into it, as shown in lines 12 to 15. The algorithm returns *False* in the case of having not enough elements in the window, as depicted in lines 16 and 17. Next, the values of p^1, p_{max}^1, and Δp are calculated or updated (lines 19 to 23). In the case of having $\Delta p \geq \varepsilon_d$, it resets its parameters and alarms for a drift by returning *True*.

Algorithm 1 Pseudocode of Fast Hoeffding Drift Detection Method (FHDDM)

function INITIALIZE($windowSize, delta$)
  $n = windowSize$
  $\delta = delta$
  $\varepsilon_d = \sqrt{\frac{1}{2n} \ln \frac{1}{\delta}}$
  RESET()
end function
function RESET()
  $w = []$ ▷ Creating an empty sliding window.
  $p^1_{max} = 0$
end function
function DETECT(p) ▷ p is 1 for the correct predictions, 0 otherwise.
  if $w.size() = n$ **then**
    $w.tail.drop()$ ▷ Dropping an element from the tail.
  end if
  $w.push(p)$ ▷ Pushing an element into the head.
  if $w.size() < n$ **then**
    return False
  else
    $p_1 = w.count(1)/w.size()$ ▷ The recent probability of seeing 1s.
    if $p^1_{max} < p^1$ **then**
      $p^1_{max} = p^1$
    end if
    $\Delta p = p^1_{max} - p^1$
    if $\Delta p \geq \varepsilon_d$ **then**
      RESET() ▷ Resetting parameters.
      return True ▷ Signalling for an alarm.
    else
      return False
    end if
  end if
end function

Window-based approaches [15-18] usually compare two (sub)windows, e.g. w_1 and w_2, leading to a considerable memory usage [1]. That is, one window is used to maintain historic information (from the beginning) and the second maintains the most recent information. In contrast, FHDDM compares the current accuracy of the classifier with its best accuracy, i.e. the best experience, observed so far using one sliding window size of n. Thus, it occupies only one register, i.e. p^1_{max}, and a sliding window size of n where $n \ll |w_1|$ or $|w_2|$. Eventually, unlike [12-14], as we apply the Hoeffding inequality, our method is independent of the probability distribution of data. The Hoeffding inequality assumes instances are independent of each other that makes the bound independent of the probability distribution.

4 On Evaluation of Concept Drift Detectors

True Positive, False Positive and False Negative numbers are useful to evaluate the performance of concept drift detectors. Intuitively, a drift detector with the highest true positive, the lowest false positive and the lowest false negative values is preferred. Huang et al. [7] and Bifet et al. [18] used three types of tests to measure true positive, false positive, and false negative values of a drift detector. For instance, to measure the false positive, they generated a stream of bits from a stationary Bernoulli distribution. If the detector alarms for drifts, one false positive is counted for each alarm. Thus, one may use three of such tests to count true positive, false positive, and false negative numbers. However, having an approach able to count them in one test, for any stream generated by any probability distribution, is preferred. To this end, we introduce an approach to count true positive, false positive and false negative by defining the *acceptable delay length* Δ. The acceptable delay length is a threshold set to determine how far the detected drift could be from the true location of drift, for being considered as true positive. Considering the acceptable delay length Δ, we describe the true positive, false positive and false negative calculations as follows:

- *True Positive (TP)*: A drift detector truly detects a drift occurred at time t if it alarms for that at anytime in $[t - \Delta, t + \Delta]$. We call this range as the *acceptable detection interval* of true positive. Eventually, the true positive rate is defined as the number of drifts correctly identified over the total number of drifts in a stream. For evaluating reactive concept drift detectors, the acceptable detection interval is $[t, t + \Delta]$.
- *False Positive (FP)*: A drift detector falsely alarms for a drift if it detects that outside of the acceptable detection intervals. The false positive rate is defined as the number of points incorrectly considered as drifts over the total number of points which are not drifts.
- *False Negative (FN)*: A drift detector falsely overlooks a drift occurred at time t if it does not alarm for that at anytime in $[t - \Delta, t + \Delta]$. The false negative rate is defined as the number of drifts incorrectly left unidentified over the total number of drifts in a stream. For the reactive concept drift detectors, the range is $[t, t + \Delta]$.

Figure 2, as an example, illustrates how the true positive, false positive and false negative are counted. The upper stream shows the real locations of drifts, i.e. the squares with D inside, and the lower stream shows the result of detection at each location. The squares with T inside represent the drifts detected correctly (true positive), the squares with F inside represent the points incorrectly considered as drift points (false positive), and the squares with N inside indicate undiscovered drifts (false negative). The drift detector signals for a drift within the first acceptable detection interval and so the true positive number increases. Subsequently, it incorrectly alarms for a drift and the false positive number increases. Since the detector does not alarm for a drift within the second acceptable detection interval, the false negative number increases. The figure shows that the detector incorrectly alarms for a drift at the very end of the stream.

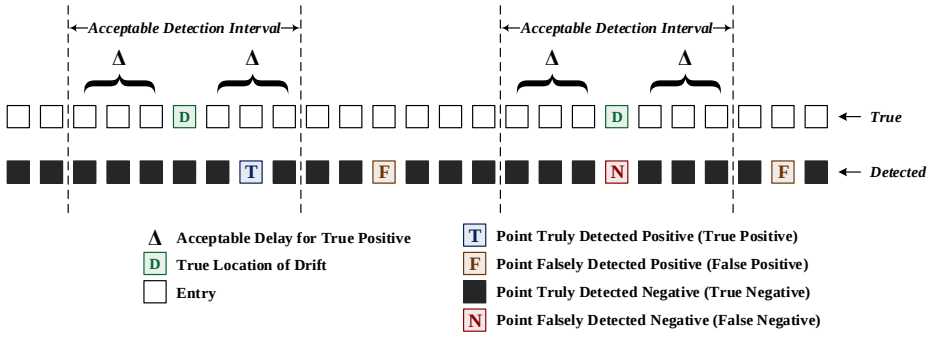

Fig. 2. Illustration of Counting True Positive, False Positive and False Negative

For data stream mining, the usage of resources will be high if the drift detector incorrectly alarms for drift repeatedly. Further, the error-rate or cost of classification would be high if the drift detector could not correctly detect the location of drifts. In other words, the error-rate of classification typically increases as does the false negative number [7, 8, 18]. Therefore, false positive and false negative are essential measures for evaluating concept drift detectors.

The *delay of detection* may be considered as a performance measure for drift detectors. Less detection delay results in losing less data for learning, it means more instances from the new distribution can be used for learning. The *detection runtime* and *detection memory usage* of drift detectors can be also used as performance measures. Intuitively, a drift detector able to correctly find drifts with less delays faster by consuming less resources is preferred.

5 Experimental Analysis

We discuss our experimental results by comparing the performance of FHDDM against that of DDM, EDDM, ADWIN, $\text{HDDM}_{\text{A-test}}$ and $\text{HDDM}_{\text{W-test}}$. We ran the experiments on synthetic and real-world datasets often used in concept drift detection research [6, 7, 12, 13, 14, 19]. We have considered Hoeffding Tree (HT), also known as VFDT, and Naive Bayes (NB) as our incremental classifiers; they are frequently used in the literature [6, 7, 12, 13, 16, 18, 19]. In all experiments, we ran the Hoeffding Tree with $\delta = 10^{-7}, \tau = 0.05$ and $n_{min} = 200$ as used in the [21]. Instances are processed prequentially, which means they are first tested and then used for training. We used MOA [22], a framework for data stream mining, to implement FHDDM in and compare it with other drift detectors. Experiments are run on Intel Core i5 @ 2.8 GHz with 16 GB of RAM running Apple OS X Yosemite.

5.1 Experiments on Synthetic Datasets

Synthetic Datasets – We generated three synthetic datasets of SINE1, MIXED and CIRCLES, as originally described in [25] and used in the literature [12, 13, 16], containing 100,000 instances with 2 classes. We also added 10% noise to each dataset. In this way, we can consider how robust drift detectors are against noisy data streams by distinguishing noises from drifts. One of the advantages of synthetic datasets is being aware of the location of drifts. Therefore, we can measure the detection delay, true positive, false positive and false negative numbers (or rates). The datasets are described below:

- SINE1 · *with abrupt concept drift*: The dataset has two attributes x and y uniformly distributed in [0, 1]. The classification function is $y = sin(x)$. Before the first drift, instances under the curve are classified as positive and others as negative. At a drift point the classification is reversed. We put the drifts at every 20,000 instances.
- MIXED · *with abrupt concept drift*: The dataset has two numeric attributes x and y uniformly distributed in [0, 1] as well as two boolean attributes v and w. The instances are classified as positive if at least two of the three following conditions are satisfied: $v, w, y < 0.5 + 0.3 * sin(2\pi x)$. The classification is reversed after drifts. Drifts happen at every 20,000 instances.
- CIRCLES · *with gradual concept drift*: It has two attributes x and y uniformly distributed in [0, 1]. The function of a circle $<(x_c, y_c), r_c>$ is $(x - x_c)^2 + (y - y_c)^2 = r_c^2$ where (x_c, y_c) is its centre and r_c is the radius. Four circles of $<(0.2, 0.5), 0.15>$, $<(0.4, 0.5), 0.2>$, $<(0.6, 0.5), 0.25>$, and $<(0.8, 0.5), 0.3>$ classify instances in order. Instances inside the circle are classified as positive. A drift happens when the classification function, i.e. circle function, changes. Drifts occur at every 25,000 instances.

Experiments – We ran Hoeffding Tree (HT) and Naive Bayes (NB) with each drift detector for 100 times and then averaged the detection delays, true positives, false positives, false negatives, detection runtimes (in millisecond), memory usage (in bytes) of drift detectors as well as the accuracies of classifiers. The acceptable drift detection delay length, i.e. Δ, was set to 250 for the SINE1 and MIXED datasets and to 1000 for the CIRCLES dataset. We consider a longer Δ for the CIRCLES dataset because it contains gradual concept drifts. Preliminary experiments and inspections confirmed that a longer Δ should be considered for gradual drifts, otherwise the false negative numbers would increase. We ran FHDDM with a sliding window size of 25 on the SINE1 and MIXED datasets, and with a sliding window size of 100 on the CIRCLES dataset. We considered a wider sliding window size for the CIRCLES dataset to make sure we have enough examples in the window as we are facing with the gradual drifts. Preliminary inspections helped us to adjust our window sizes for resulting in less detection delay, less false positive and less false negative. Since FHDDMs' sliding windows are small and they compare p_t^1 with the p_{max}^1, we need to set δ to a small value to make sure the ε_d is big enough. It was, therefore, set δ to 10^{-7} for our experiments. All other drift detectors were run with the default parameters as set in MOA (or as in the original papers).

Table 1 (a) represents the results of experiments on the SINE1 dataset. FHDDM has the lowest false positive and false negative averages with both classifiers. $\text{HDDM}_{\text{W-test}}$ results in the lowest delay followed by FHDDM with small margins. DDM and EDDM exhibit the highest detection delay. They are also the only two drift detectors with false negative averages. EDDM and ADWIN have considerable false positive averages. As shown in Table 1 (b), we achieve the highest classification accuracies by FHDDM and $\text{HDDM}_{\text{W-test}}$ with both classifiers. It is clearly seen that ADWIN has the longest runtimes and the highest memory usages with considerable margins.

Table 1. Results of Experiments on SINE1 Dataset (10% Noise)

(a) Drift Detection Delay, True Positive, False Positive and False Negative

Classifier	Detector	Delay	TP	FP	FN
HT	FHDDM	16.58 ± 1.39	**4.0**	**0.02 ± 0.14**	**0.0**
	DDM	139.82 ± 23.87	3.42 ± 0.72	2.93 ± 1.87	0.58 ± 0.72
	EDDM	242.81 ± 16.42	0.20 ± 0.40	32.67 ± 12.97	3.80 ± 0.40
	ADWIN	21.23 ± 1.39	**4.0**	21.74 ± 5.72	**0.0**
	$\text{HDDM}_{\text{A-test}}$	32.2 ± 12.69	**4.0**	0.73 ± 1.01	**0.0**
	$\text{HDDM}_{\text{W-test}}$	**11.62 ± 1.00**	**4.0**	0.6 ± 0.72	**0.0**
NB	FHDDM	16.84 ± 1.02	**4.0**	**0.0**	**0.0**
	DDM	213.05 ± 69.57	3.39 ± 0.71	2.19 ± 1.51	0.61 ± 0.71
	EDDM	412.74 ± 73.03	1.65 ± 0.99	31.01 ± 12.15	2.35 ± 0.99
	ADWIN	22.13 ± 1.50	**4.0**	17.89 ± 5.10	**0.0**
	$\text{HDDM}_{\text{A-test}}$	59.34 ± 21.05	**4.0**	0.27 ± 0.53	**0.0**
	$\text{HDDM}_{\text{W-test}}$	**11.66 ± 1.29**	**4.0**	0.39 ± 0.69	**0.0**

(b) Drift Detection Runtime and Memory Usage with Classification Accuracy

Classifier	Detector	Runtime (ms)	Memory (bytes)	Accuracy
HT	FHDDM	35.02 ± 5.93	352	**87.07% ± 0.15**
	DDM	17.15 ± 4.47	**160**	86.16% ± 1.01
	EDDM	**11.89 ± 3.51**	**160**	84.62% ± 0.54
	ADWIN	2538.31 ± 103.53	1399.7 ± 81.43	86.77% ± 0.18
	$\text{HDDM}_{\text{A-test}}$	43.19 ± 6.35	168	86.99% ± 0.14
	$\text{HDDM}_{\text{W-test}}$	36.39 ± 5.75	**160**	87.05% ± 0.15
NB	FHDDM	35.22 ± 5.72	352	86.08% ± 0.21
	DDM	17.23 ± 4.69	**160**	81.70% ± 4.49
	EDDM	**13.39 ± 3.8**	**160**	83.67% ± 2.27
	ADWIN	2532.92 ± 89.88	1356.32 ± 65.79	85.99% ± 0.21
	$\text{HDDM}_{\text{A-test}}$	43.92 ± 6.16	168	85.96% ± 0.21
	$\text{HDDM}_{\text{W-test}}$	35.23 ± 6.55	**160**	**86.09% ± 0.21**

We show the results of experiments on the MIXED dataset in Table 2. FHDDM and ADWIN have the highest true positive averages without causing any false negatives. FHDDM has the smallest false positive averages while EDDM and ADWIN have the highest averages. $\text{HDDM}_{\text{W-test}}$ and FHDDM have the shortest detection delays. As represented in Table 2 (b), we achieve the highest classification accuracies by FHDDM with both classifiers. EDDM and ADWIN result in the shortest and longest detection runtimes, respectively. ADWIN considerably occupies the memory.

Table 2. Results of Experiments on MIXED Dataset (10% Noise)

(a) Drift Detection Delay, True Positive, False Positive and False Negative

Classifier	Detector	Delay	TP	FP	FN
HT	FHDDM	16.05 ± 1.28	**4.0**	**0.24 ± 0.51**	**0.0**
	DDM	181.43 ± 24.08	2.69 ± 0.9	2.95 ± 1.92	1.31 ± 0.90
	EDDM	248.01 ± 8.95	0.07 ± 0.26	20.81 ± 7.18	3.93 ± 0.26
	ADWIN	23.85 ± 1.68	**4.0**	18.96 ± 5.27	**0.0**
	$HDDM_{A\text{-test}}$	41.12 ± 16.89	3.99 ± 0.10	1.16 ± 1.14	0.01 ± 0.10
	$HDDM_{W\text{-test}}$	**11.57 ± 5.92**	3.99 ± 0.10	3.15 ± 2.04	0.01 ± 0.10
NB	FHDDM	16.15 ± 1.43	**4.0**	**0.03 ± 0.17**	**0.0**
	DDM	178.08 ± 27.77	2.77 ± 0.93	2.26 ± 1.33	1.23 ± 0.93
	EDDM	248.13 ± 7.55	0.11 ± 0.34	20.71 ± 7.94	3.89 ± 0.34
	ADWIN	23.79 ± 1.61	**4.0**	18.24 ± 5.28	**0.0**
	$HDDM_{A\text{-test}}$	58.10 ± 23.56	3.99 ± 0.10	0.60 ± 0.76	0.01 ± 0.1
	$HDDM_{W\text{-test}}$	**11.48 ± 6.12**	3.99 ± 0.10	1.63 ± 1.16	0.01 ± 0.1

(b) Drift Detection Runtime and Memory Usage, and Classification Accuracy

Classifier	Detector	Runtime (ms)	Memory (bytes)	Accuracy
HT	FHDDM	35.98 ± 5.32	352	**83.40% ± 0.12**
	DDM	17.35 ± 4.28	**160**	81.77% ± 1.74
	EDDM	**13.09 ± 3.56**	**160**	80.58% ± 0.89
	ADWIN	2417.56 ± 89.81	1343.63 ± 69.07	83.28% ± 0.13
	$HDDM_{A\text{-test}}$	41.69 ± 6.54	168	83.30% ± 0.13
	$HDDM_{W\text{-test}}$	33.07 ± 5.69	**160**	83.27% ± 0.13
NB	FHDDM	35.71 ± 6.04	352	**83.39% ± 0.09**
	DDM	17.65 ± 4.76	**160**	80.74% ± 3.52
	EDDM	**13.38 ± 3.81**	**160**	80.53% ± 1.89
	ADWIN	2715.83 ± 99.69	1349.26 ± 58.26	83.31% ± 0.09
	$HDDM_{A\text{-test}}$	43.64 ± 6.45	168	83.27% ± 0.11
	$HDDM_{W\text{-test}}$	36.89 ± 5.87	**160**	83.37% ± 0.09

Tables 3 (a) and (b) hold the experiments results on the CIRCLES dataset. FHDDM results in the shortest detection delay, the highest false positive, the lowest false positive and the lowest false negative with Hoeffding Tree. ADWIN has the shortest detection delay and the highest true positive average with Naive Bayes and it is followed by FHDDM. EDDM and ADWIN have the highest false positive averages. In the terms of classification accuracies, we achieve the highest ones by FHDDM with either of classifiers. Like the previous experiments, EDDM has the shortest detection runtimes and ADWIN has the highest memory occupations.

We compared FHDDM with existing drift detection methods on the synthetic datasets containing abrupt and gradual concept drifts. In conclusion, FHDDM had the first or second shortest detection delay, the highest true positive average, the lowest false positive average, and the lowest false negative average. Further, the detection runtime and memory occupation was comparable to $HDDM_{A\text{-test}}$'s and $HDDM_{W\text{-test}}$'s. Importantly, FHDDM led to the highest classification accuracies with both Hoeffding Tree and Naive Bayes.

Table 3. Results of Experiments on CIRCLES Dataset (10% Noise)

(a) Drift Detection Delay, True Positive, False Positive and False Negative

Classifier	Detector	Delay	TP	FP	FN
HT	FHDDM	**94.14 ± 28.05**	**3.0**	**0.04 ± 0.20**	**0.0**
	DDM	604.7 ± 118.48	2.26 ± 0.8	1.74 ± 1.58	0.74 ± 0.8
	EDDM	981.60 ± 62.64	0.11 ± 0.31	22.75 ± 10.83	2.89 ± 0.31
	ADWIN	249.45 ± 159.06	2.58 ± 0.55	17.9 ± 5.66	0.42 ± 0.55
	$HDDM_{A\text{-test}}$	123.57 ± 72.89	2.95 ± 0.22	0.48 ± 0.77	0.05 ± 0.22
	$HDDM_{W\text{-test}}$	101.84 ± 70.79	2.96 ± 0.2	0.54 ± 0.75	0.04 ± 0.20
NB	FHDDM	270.98 ± 125.98	2.83 ± 0.38	**0.20 ± 0.40**	0.17 ± 0.38
	DDM	830.15 ± 139.60	1.35 ± 0.90	2.50 ± 1.58	1.65 ± 0.90
	EDDM	949.50 ± 96.80	0.32 ± 0.51	34.84 ± 19.30	2.68 ± 0.51
	ADWIN	**225.25 ± 56.39**	**3.0**	17.84 ± 5.28	**0.0**
	$HDDM_{A\text{-test}}$	494.95 ± 155.27	2.63 ± 0.52	0.68 ± 0.68	0.37 ± 0.52
	$HDDM_{W\text{-test}}$	316.24 ± 149.4	2.68 ± 0.51	1.21 ± 1.06	0.32 ± 0.51

(b) Drift Detection Runtime and Memory Usage, and Classification Accuracy

Classifier	Detector	Runtime (ms)	Memory (bytes)	Accuracy
HT	FHDDM	64.38 ± 6.97	952	**86.66% ± 0.14**
	DDM	16.76 ± 4.39	**160**	85.84% ± 0.81
	EDDM	**11.92 ± 3.57**	**160**	84.96% ± 0.27
	ADWIN	2690.09 ± 129.28	1716.69 ± 106.23	86.04% ± 0.23
	$HDDM_{A\text{-test}}$	43.85 ± 6.83	168	86.60% ± 0.19
	$HDDM_{W\text{-test}}$	35.67 ± 5.73	**160**	86.58% ± 0.17
NB	FHDDM	64.54 ± 7.56	952	**84.31% ± 0.14**
	DDM	16.66 ± 0.90	**160**	82.73% ± 1.57
	EDDM	**13.25 ± 3.60**	**160**	83.30% ± 0.40
	ADWIN	2818.84 ± 41.55	1777.86 ± 77.53	**84.31% ± 0.12**
	$HDDM_{A\text{-test}}$	42.43 ± 6.57	168	84.24% ± 0.14
	$HDDM_{W\text{-test}}$	35.81 ± 5.67	**160**	84.27% ± 0.16

5.2 Experiments on Real-World Datasets

Real-World Datasets – We considered the AIRLINES [26], POKER HAND [27] and ELECTRICITY [28] datasets widely used in concept drift research [6, 7, 12, 13, 18, 19]. The preprocessed and normalized version of datasets are available at MOA website[2]. The datasets are described below:

- AIRLINES: This dataset was created to be used as a non-stationary data stream for evaluating learning algorithms [26]. It contains 539,383 records of flight schedules defined by 7 attributes. The task is to predict if a flight is delayed or not. Concept drift could appear as the result of changes in the flights schedules, e.g. changes in day, time, and the length of flights.
- POKER HAND: It comprises 1,000,000 instances with 11 attributes. Each instance is an example of a hand consisting of five playing cards drawn from a standard deck of 52. Each card is described by two attributes (suit and rank), for ten predictive attributes. The class predicts the poker hand. Concept drift happens as changing the card at hand, i.e. the poker hand [4].
- ELECTRICITY: It has 45,312 instances, with 8 input attributes, recorded every half an hour for a period of two years from Australian New South Wales

[2] Real-world datasets: http://moa.cms.waikato.ac.nz/datasets/

Table 4. The Results of Experiments on AIRLINES Dataset

Classifier	Detector	Runtime (ms)	Memory (bytes)	Num. Drifts	Accuracy
HT	FHDDM	118.05 ± 12.04	352	**339**	**65.66%**
	DDM	49.75 ± 9.87	160	14	**65.29%**
	EDDM	**41.60 ± 8.37**	160	54	65.06%
	ADWIN	13439.70 ± 96.48	1879.23	**341**	**65.25%**
	$HDDM_{A\text{-test}}$	146.45 ± 12.08	168	88	64.99%
	$HDDM_{W\text{-test}}$	107.55 ± 10.67	160	652	65.02%
	No Detection	—	—	—	65.07%
NB	FHDDM	114.15 ± 7.93	352	**297**	**66.44%**
	DDM	48.65 ± 6.13	**160**	13	65.33%
	EDDM	**39.55 ± 7.17**	**160**	23	65.18%
	ADWIN	13034.20 ± 77.13	1896.08	**300**	**66.79%**
	$HDDM_{A\text{-test}}$	139.80 ± 13.34	168	72	**67.22%**
	$HDDM_{W\text{-test}}$	86.70 ± 9.63	**160**	620	65.34%
	No Detection	—	—	—	64.55%

Electricity. The classification task is to predict a rise (*Up*) or a fall (*Down*) in the electricity price. The concept drift may happen because of changes in consumption habits, unexpected events and seasonality [29].

Experiments – The ground truth for drifts is not available for the real-world datasets. This implies that we do not know whether drifts occur in these datasets or where they occur [6, 7]. We, therefore, cannot measure the detection delay, true positive, false positive, and false negative numbers of drift detectors in this section. We only evaluate the number of drifts detected and the accuracy of classification. All classifiers and drift detectors were run with the default parameters. For FHDDM, we only present the results obtained by the sliding window size of 25 because we usually obtained better classification accuracies with size 25 on the real-world datasets in our preliminary experiments, though the margins of differences were small.

Tables 4 to 6 summarize the results of experiments on the aforementioned datasets. The accuracy of classification has improved by using drift detectors. The classification accuracy with FHDDM is among the highest ones. It also detects less drifts compared to ADWIN, $HDDM_{A\text{-test}}$ and $HDDM_{W\text{-test}}$ while their classification accuracies are similar. As argued in [7], there are two possible cases if a drift detector detects less number of drifts compared to the other drift detectors while they all lead to similar classification accuracies: 1) That drift detector caused less false positive compared to others, or 2) Not detected drifts, i.e. false negatives, were less significant drifts. The second case implicitly says having less number of drifts detected leading to lower classification accuracy suggests significant false negatives. Therefore, based on these arguments, it is more likely that FHDDM caused fewer false positives. Its detection runtime is also comparable with $HDDM_{A\text{-test}}$'s and $HDDM_{W\text{-test}}$'s. In all cases, FHDDM resulted in shorter detection runtimes, less memory occupations and higher classification accuracies compared to ADWIN.

Table 5. The Results of Experiments on POKER HAND Dataset

Classifier	Detector	Runtime (ms)	Memory (bytes)	Num. Drifts	Accuracy
HT	FHDDM	173.85 ± 16.54	352	**1557**	**76.45%**
	DDM	62.00 ± 7.78	**160**	1046	72.74%
	EDDM	**56.50 ± 7.05**	**160**	4806	**77.30%**
	ADWIN	12725.25 ± 108.71	1464.42	**2373**	74.56%
	$HDDM_{A\text{-test}}$	200.10 ± 8.81	168	**2565**	**76.40%**
	$HDDM_{W\text{-test}}$	147.65 ± 12.84	**160**	**2211**	**77.11%**
	No Detection	—	—	—	76.07%
NB	FHDDM	166.60 ± 12.60	352	**1660**	**76.30%**
	DDM	69.05 ± 7.53	**160**	433	61.97%
	EDDM	**56.60 ± 8.43**	**160**	4863	**77.48%**
	ADWIN	12650.00 ± 248.42	1453.93	**2453**	74.60%
	$HDDM_{A\text{-test}}$	195.85 ± 13.74	168	**2615**	**76.48%**
	$HDDM_{W\text{-test}}$	125.55 ± 13.26	**160**	**2312**	**77.11%**
	No Detection	—	—	—	59.55%

Table 6. The Results of Experiments on ELECTRICITY Dataset

Classifier	Detector	Runtime (ms)	Memory (bytes)	Num. Drifts	Accuracy
HT	FHDDM	22.70 ± 4.31	352	**77**	84.38%
	DDM	15.70 ± 4.79	**160**	169	84.41%
	EDDM	**10.40 ± 3.37**	**160**	191	84.91%
	ADWIN	738.05 ± 23.28	1468.29	**110**	83.40%
	$HDDM_{A\text{-test}}$	28.05 ± 6.67	168	210	**85.71%**
	$HDDM_{W\text{-test}}$	24.35 ± 4.66	**160**	**117**	**85.06%**
	No Detection	—	—	—	79.20%
NB	FHDDM	23.90 ± 4.17	352	**96**	82.69%
	DDM	13.25 ± 4.90	**160**	143	81.18%
	EDDM	**8.60 ± 2.24**	**160**	203	**84.83%**
	ADWIN	691.85 ± 17.82	1408.25	**128**	81.63%
	$HDDM_{A\text{-test}}$	25.40 ± 5.23	168	211	**84.92%**
	$HDDM_{W\text{-test}}$	25.45 ± 5.84	**160**	**132**	84.09%
	No Detection	—	—	—	73.36%

6 Conclusion and Future Work

Adapting classification learners is essential when they are used to learn from data in evolving environments. In this paper, we introduced a new concept drift detection method, so-called FHDDM, that uses Hoeffding's inequality. The method works based on the fact that the accuracy of a classifier should increase or stay steady as more instances arrive; otherwise it implies the existence of drift points in the stream. FHDDM slides a window with a size of n on the stream and measures the p_t^1, i.e. the probability of correct classification predictions in the most recent n instances at time t. It updates the value of p_{max}^1 that holds the maximum probability of correct predictions seen so far. A significant difference, bounded by Hoeffding's inequality, between p_t^1 and p_{max}^1 suggests a drift. In addition, we introduced an approach to count true positive, false positive and false negative of drift detectors by considering their delay of detection for evolving data streams.

We experimentally evaluated our method on the synthetic and real-world datasets. Experiments on the synthetic datasets indicated that FHDDM detects drifts with a shorter delay, leading to the highest true positive, the lowest false positive and the lowest false negative, when compared to the state-of-the-art. When considering real-world datasets, the classification accuracies of our method were consistently high.

In the future, we will investigate the performance of our FHDDM approach on imbalanced and highly noisy data streams as well as streams containing outliers. We will also consider implementing an adaptable window size, as based on the trends of prediction results. In addition, we plan to study the sensitivity of FHDDM's parameters, i.e. *size of sliding window* and *confidence level*, along with other drift detectors' and consider their performances in different domains. It would also be worthwhile to compare FHDDM with other drift detectors, as proposed in [14, 24], amongst others. Finally, we intend to use our proposed method in anomaly detection and business intelligence applications.

References

1. Gama, J. A., Zliobaite, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A Survey on Concept Drift Adaptation. ACM Computing Surveys, **46**(4), 44:1-44:37, (2014)
2. Ditzler, G., Roveri, M., Alippi, C., Polikar, R.: Learning in Nonstationary Environments: A survey. Computational Intelligence Magazine, **10**(4), pp. 12–25 (2015)
3. Alippi, C., Boracchi, G., Roveri, M.: Just-in-time Ensemble of Classifiers. In: International Joint Conference on Neural Networks, pp. 1–8 (2012)
4. Olorunnimbe, M. K., Viktor, H. L., Paquet, E.: Intelligent Adaptive Ensembles for Data Stream Mining: A High Return on Investment Approach. In: International Workshop on New Frontiers in Mining Complex Patterns, pp. 61–75, Springer International Publishing (2015)
5. Kuncheva, L. I.: Classifier Ensembles for Detecting Concept Change in Streaming Data: Overview and Perspectives. In: 2nd Workshop SUEMA, pp. 5–9 (2008)
6. Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., Gavalda, R.: New Ensemble Methods for Evolving Data Streams. In: 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 139–148, ACM (2009)
7. Huang, D. T. J., Koh, Y. S., Dobbie, G., Bifet, A.: Drift Detection Using Stream Volatility. In: Machine Learning and Knowledge Discovery in Databases, pp. 417–432, Springer International Publishing (2015)
8. Zliobaite, Indre, Marcin Budka, and Frederic Stahl.: Towards Cost-Sensitive Adaptation: When is it Worth Updating Your Predictive Model? Neurocomputing **150**, pp. 240–249 (2015)
9. Hoeffding, W.: Probability Inequalities for Sums of Bounded Random Variables. American Statistical Association **58** (301), pp. 13–30 (1963)
10. Page, E. S.: Continous Inspection Schemes, Biometrika **41**, pp. 100-115 (1954)
11. Roberts, S. W.: Control Chart Tests Based on Geometric Moving Averages. Technometrics **42**(1), pp. 97–101 (2000)
12. Gama, J., Medas, P., Castillo, G., Rodrigues, P.: Learning with Drift Detection. In: Advances in Artificial Intelligence, pp. 286–295, Springer Berlin Heidelberg (2004)

13. Baena-Garcia, M., del Campo-Avila, J., Fidalgo, R., Bifet, A., Gavalda, R., Morales-Bueno, R.: Early Drift Detection Method. In: 4th International Workshop on Knowledge Discovery from Data Streams, vol. 6, pp. 77–86 (2006)
14. Ross, G. J., Adams, N. M., Tasoulis, D. K., Hand, D. J.: Exponentially Weighted Moving Average Charts for Detecting Concept Drift. Pattern Recognition Letters **33**(2), pp. 191–198 (2012)
15. Kifer, D., Ben-David, S., Gehrke, J.: Detecting Change in Data Streams. In: 30th International Conference on Very Large Data Bases, vol. 30, pp. 180–191, (2004)
16. Nishida, K., Yamauchi, K.: Detecting Concept Drift Using Statistical Testing. In: Discovery Science, pp. 264–269, Springer Berlin Heidelberg (2007)
17. Bach, S. H., Maloof, M. A.: Paired Learners for Concept Drift. In: 8th IEEE International Conference on Data Mining ICDM'08, pp. 23–32 (2008)
18. Bifet, A., Gavalda, R.: Learning from Time-Changing Data with Adaptive Windowing. In: SIAM International Conference on Data Mining, pp. 443–448 (2007)
19. Frias-Blanco, I., del Campo-Avila, J., Ramos-Jimenez, G., Morales-Bueno, R., Ortiz-Diaz, A., Caballero-Mota, Y.: Online and Non-parametric Drift Detection Methods based on Hoeffding's Bounds. IEEE Transactions on Knowledge and Data Engineering **27**(3), pp. 810–823 (2015)
20. Mitchell, T.: Machine Learning. McGraw Hill (1997)
21. Domingos, P., Hulten, G.: Mining High-Speed Data Streams. In: 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 71–80 (2000)
22. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: MOA: Massive Online Analysis. Machine Learning Research **11**, pp. 1601–1604 (2010)
23. Sakthithasan, S., Pears, R., Koh, Y. S.: One Pass Concept Change Detection for Data Streams. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 461–472. Springer Berlin Heidelberg (2013)
24. Pears, R., Sakthithasan, S., Koh, Y. S.: Detecting Concept Change in Dynamic Data Streams. Machine Learning **97**(3), pp. 259–293 (2014)
25. Kubat, M., Widmer, G.: Adapting to Drift in Continuous Domains. In: European Conference on Machine Learning, pp. 307–310, Springer Berlin Heidelberg (1995)
26. Ikonomovska, E.: Airline Dataset. http://kt.ijs.si/elena_ikonomovska/data.html (2011) (Last Visit Happened on 03/15/2016)
27. Cattral, R., Oppacher, F., Deugo, D.: Evolutionary Data Mining with Automatic Rule Generalization. Recent Advances in Computers, Computing and Communications, pp. 296–300 (2002)
28. Harries, M., Wales, N. S.: Splice-2 Comparative Evaluation: Electricity pricing. Technical Report, University of New South Wales, Australia (1999)
29. Zliobaite, I.: How Good is The Electricity Benchmark for Evaluating Concept Drift Adaptation. arXiv preprint arXiv:1301.3524 (2013)