Concept Drift Detection in Histogram-Based Straightforward Data Stream Prediction

Jorge Casillas
Dept. Computer Science & AI,
University of Granada, Spain
Email: casillas@decsai.ugr.es

Shuo Wang
School of Computing & Digital Technology,
Birmingham City University, UK
Email: shuo.wang@bcu.ac.uk

Xin Yao
CERCIA, School of Computer Science,
The University of Birmingham, UK
Email: x.yao@cs.bham.ac.uk

***Abstract*—Mainstream research in concept drift detection for on-line classification focuses on monitoring a measure of the learner's performance, thus assuming that a reduction of its prediction ability implies a change in the relation between the input variables and the class labels. This approach makes the detector highly dependent on the learner, which can be problematic in some situations, for example, when the learner includes adaptability mechanisms, when it is unable to converge, or when it shows a high overfitting. Ultimately, the concept drift is something that happens in the data, not learners, so detecting drifts indirectly through a learner's performance adds bias into the result. Besides, it makes the process highly inefficient.**

This paper proposes a new mechanism to detect concept drifts without supervising the learning process. The data distribution is summarized in univariate histograms based on which an on-line straightforward prediction is made. It proves us a measure that is monitored over time for detecting changes in data. The method is extremely efficient, as the time and space complexity to process each sample is linear to the number of input variables multiplied by the number of classes. A thorough analysis on synthetic and real-world data streams shows that the proposed method very efficiently makes a reliable (low false alarm ratio) and effective (high true detection ratio) drift detection that outperforms other well-known methods.

I. Introduction

Every day is more usual to deal with data science problems that generate the information incrementally as structured record sequences. This sequence is commonly non-stationary, in the sense that the dynamics of the problem changes over time, thus requiring specific techniques (known as data stream mining) able to perform an on-line learning that adapts to detect and adapt to these changes in an online fashion. The key characteristic of these systems is that the data records are not stored permanently. Instead, they are processed on-the-fly. Each record is analyzed, processed and then forgotten, making the management of the huge amount of data in real-time possible, even with reduced storage and processing capacities.

A large amount of research in data stream mining focuses on supervised classification, which assumes that a stream of samples with class labels is continuously received. A classification model is built and maintained online while dealing with different types of changes in the data distribution (known as concept drifts [1]). It is expected to react as efficiently and effectively as possible to these drifts. There are to ways to deal with concept drift in data streams: explicitly detecting these concept drifts and resetting the learning (active methods); implicitly adapting the learning process (passive methods).

The aim of any drift detector is to recognize changes in non-stationary data streams accurately and timely. To do so, most of research monitors the reduction in the predictive capability as an indicator of changes in the data [2], [3], [4]. Such approaches relate drift detection to the learner's performance. However, it can occur the paradoxical situation that learners able to autonomously adapt to concept drifts may not be useful to detect weak drifts if their prediction capabilities are not radically affected. Besides, having the knowledge about when the concept drifts occur is also valuable information for decision-making in real-world problems. At the end, it is expected that an effective detection of these drifts would help advanced learners to react even quicker and better.

Another drawback of existing approaches is the poor efficiency. Tracking a learner's performance for drift detection significantly slows down the process, which make these methods less applicable to real-world problems receiving a large number of samples per second.

Therefore, we aim to develop a concept drift detection method that does not rely on the learner's performance; meanwhile, it is expected to react to the drift accurately and promptly by issuing a drift alert. Our method analyzes the input data instead of monitoring the learner. It produces histograms that summarize the data distributions, and uses a straightforward prediction based on this knowledge to oversee changes in the decision boundaries. This method has the following advantages as a drift detector:

- It has time and space complexity of $\mathcal{O}(n \cdot m)$—with n being the number of features and m the number of classes—at each time step. The proposal is extremely efficient. This aspect is essential in data stream mining as it allows to deal with high speed data, thus providing more applicability in real-world problems compared to conventional drift detectors. Low memory consumption is another benefit of our method, which can be crucial for mobile, wearable and IoT devices.
- The implementation and integration of the proposed method is agile, thus facilitating prototyping and exploratory analysis of new data stream problems. Its design is simple, so it is easy to understand its behavior and debug the process.

- It can find out the concept drift accurately even when the performance of the learner does not significantly degrade. It helps us gain further insight about the problem nature.
- In the last term, the proposed method is compatible with traditional drift detectors based on learner's performance, so they can be used together to complement with each other and to dynamically adapt themselves to the problem.

The rest of the paper is organized as follows: Section II presents the fundamentals of the proposal based on histograms to represent data distributions and a straightforward prediction over them to measure changes in the decision boundaries. Section III introduces the proposed algorithm to detect drifts from this measurement. Section IV carries out the experiments in different synthetic data streams and analyzes its behavior in comparison with other well-known drift detectors in depth. Section V analyzes the efficiency of the proposed approach. Finally, Section VI concludes and draws some future lines of research.

II. Drift Detection in Histogram-Based Classification

This section introduces the background of the histogram computation that can on-line record the data distribution, a straightforward mechanism for setting decision boundaries and an on-line performance measure that allow the algorithm to detect changes in these boundaries and so potential concept drifts.

A. Synopsis by Histograms

Histograms can be used to summarize data considering a data stream scenario where data are not stored. Let $\mathbf{x}^t = (x_1^t, \ldots, x_i^t, \ldots, x_n^t)$ be the vector of n input variables for the sample received in the instant t and let y^t be the class of this sample, which takes values in $\{c_1, \ldots, c_j, \ldots, c_m\}$ in a problem with m different classes. When a new sample $\mathbf{x}^t$ arrives and its corresponding class y^t is known in the instant $t+1$, m histograms (one per class) are updated for each input variable as follows:

$$\mathbf{H}_{ij}^t = \alpha(t) \cdot \mathbf{H}_{ij}^{t-1} + \mathbf{I}_{ij}^t, \quad \mathbf{H}_{ij}^0 = 0 \tag{1a}$$

$$\mathbf{I}_{ij}^t = \left(K_{A_1^i}(x_i^t, y^t, c_j), \ldots, K_{A_{b_i}^i}(x_i^t, y^t, c_j) \right) \tag{1b}$$

$$K_A(x, y, c) = \begin{cases} 1, & y = c \text{ and } x \in A \\ 0, & \text{otherwise} \end{cases} \tag{1c}$$

$$A_k^i = \begin{cases} \left[a_{k-1}^i, a_k^i\right), & \forall k \in \{1, \ldots, b_i - 1\} \\ \left[a_{k-1}^i, a_k^i\right], & k = b_i \end{cases} \tag{1d}$$

$$a_k^i = min_i + k \cdot s_i, \quad s_i = \frac{max_i - min_i}{b_i} \tag{1e}$$

where $\alpha(t) \in [0, 1]$ is the retain factor in time t (i.e., $1 - \alpha(t)$ as forgetting/decay factor) and b_i is the number of bins of the m histograms of the ith input variable (which is ranged in $[min_i, max_i]$).

Note that we do not use the traditional delta rule function $x^t = \alpha \cdot x^{t-1} + (1-\alpha) \cdot \chi^t \;\equiv\; x^t = x^{t-1} + (1-\alpha) \cdot (\chi^t - x^{t-1})$ as it estimates the expected value of variable χ. The measure we need is $\mathbf{H}$ as an additive value of the last received data. Note also that (1a) holds

$$\sum_k H_{ijk}^t = \frac{1}{1 - \alpha(t)} = \delta \tag{2}$$

provided $\alpha(t)$ is a constant.

Therefore, with a given $\alpha(t) = \frac{\delta - 1}{\delta}$, we can ensure $\mathbf{H_{ij}^t}$ constantly sums δ ($\forall t$), being δ a good indicator of the memory persistence we want to preserve in $\mathbf{H}$. To do so:

$$\alpha(t) = \begin{cases} 1, & 1 \le t \le \delta \\ \dfrac{\delta - 1}{\delta}, & t > \delta \end{cases} \tag{3}$$

As higher δ is, more stable and accurate $\mathbf{H}$ is. However, it will become slower to react to changes in data streams. By contrast, low values of δ ensures a quick reaction to changes but with a unstable behavior. As δ is defined in terms of number of received samples (analogously it can be defined in terms of the time window), it is a parameter easy to be set by the user according to problem requirements.

B. Straightforward Data Stream Prediction

The histograms described in Section II-A can be used to predict the class of the received sample, thus developing a straightforward data stream prediction with the advantage of being efficient in time and memory. This prediction neither acts as a classifier nor reaches sound accuracies, but it can be used to monitor changes in decision boundaries which would eventually lead to a real concept drift. In other words, the accuracy achieved by this prediction is not the key; its change affected by the concept drift is useful. When the decision boundaries change because of a redistribution of the data that affects posterior probability $P(y|\mathbf{x})$, the performance of the straightforward prediction based on histograms will vary, and will be so until the histograms are restored by the forgetting mechanism regulated by α and δ.

The proposed prediction is designed as follows. Firstly, it is determined for each variable the relative frequency (h_{ij}) of the received sample in each class according to the historical histograms. For a better estimation, in continuous variables an interpolation (with weight w_i) between the bin where the sample lies (ϕ_i) and its nearest bin (φ_i) is performed. In categorical variables, the number of bins should be equal to the number of categories (if time and memory resources permit) and no interpolation is applied.

$$\phi_i = k \;\text{ s.t. }\; x_i \in A_k^i \tag{4a}$$

$$\varphi_i = \begin{cases} k - 1, & x_i \in A_k^i, k > 1, x_i < \frac{a_{k-1}^i + a_k^i}{2} \\ k + 1, & x_i \in A_k^i, k < b_i, x_i > \frac{a_{k-1}^i + a_k^i}{2} \\ k, & \text{otherwise} \end{cases} \tag{4b}$$

$$w_i = \begin{cases} 0.5 + \frac{x_i - a^i_{k-1}}{max_i - min_i}, & \varphi_i = k-1 \\ 1.5 + \frac{a^i_{k-1} - x_i}{max_i - min_i}, & \varphi_i = k+1 \\ 1, & \text{otherwise} \end{cases} \tag{5a}$$

$$h_{ij} = \frac{w_i H_{ij\phi_i} + (1-w_i) H_{ij\varphi_i}}{\sum_k H_{ijk}}, \quad \forall j \in \{1, \dots, m\} \tag{5b}$$

Next, these values are used to vote for the winner class in each variable.

$$\nu_{ij} = \frac{h_{ij}}{\sum_k h_{ik}}, \quad \vartheta_j = \max_i \nu_{ij} \tag{6}$$

Finally, the predicted class is determined by the class that receives the highest score:

$$\hat{y} = c_k \quad \text{s.t.} \quad k = \arg\max_j \vartheta_j \tag{7}$$

Note that maximizing ν_{ij} intrinsically seeks for cases with the highest separability. In last term, the proposed prediction assigns the class label of the most frequent one in the very near neighbour (with a radius defined by s_i in (1e), which depends of the number of bins b_i) of the received samples that are summarized in the histograms. Therefore, the proposed online predictor is an univariate k-NN algorithm for data stream mining based on data synopses in the form of histograms that does not need to store past samples.

C. Prediction Performance

The performance of the above mentioned predictor can be used to detect concept drifts. To achieve this, we propose to estimate its accuracy with a forgetting mechanism. Let $\hat{y}^t$ be the predicted class for sample $\mathbf{x}^t$. In case of binary classification ($m = 2$), accuracy of the prediction is based on *G-mean* as follows:

$$G^t = \sqrt{\frac{M^t_{tp}}{M^t_{tp} + M^t_{fn}} \cdot \frac{M^t_{tn}}{M^t_{tn} + M^t_{fp}}} \tag{8a}$$

$$\mathbf{M}^t = \left(M^t_{tp}, M^t_{fn}, M^t_{tn}, M^t_{fp}\right), \ \mathbf{M}^0 = 0 \tag{8b}$$

$$M^t_{tp} = \alpha \cdot M^{t-1}_{tp} + tp^t, \ \ M^t_{fn} = \alpha \cdot M^{t-1}_{fn} + fn^t, \tag{8c}$$

$$M^t_{tn} = \alpha \cdot M^{t-1}_{tn} + tn^t, \ \ M^t_{fp} = \alpha \cdot M^{t-1}_{fp} + fp^t \tag{8d}$$

$$tp^t = \begin{cases} 1, & y^t = \hat{y}^t = c_1 \\ 0, & \text{otherwise} \end{cases}, \ fn^t = \begin{cases} 1, & y^t = c_1, \hat{y}^t = c_2 \\ 0, & \text{otherwise} \end{cases} \tag{8e}$$

$$tn^t = \begin{cases} 1, & y^t = \hat{y}^t = c_2 \\ 0, & \text{otherwise} \end{cases}, \ fp^t = \begin{cases} 1, & y^t = c_2, \hat{y}^t = c_1 \\ 0, & \text{otherwise} \end{cases} \tag{8f}$$

As the higher is G in $[0, 1]$, the more accurate is the classification model in predicting both positive and negative classes. Note that G is not sensitive to class distribution.

III. Drift Detector Algorithm

This section proposes HSP (histogram-based straightforward prediction) drift detector method to effectively identify variations in the decision boundary that would lead to concept drifts. The main idea is we use the straightforward prediction to monitor the dynamics of the decision boundary. We consider both increases and decreases of the error as concept drift contrary to the conventional approach focusing only on reductions of the prediction accuracy. Besides, the histograms retain the distributions of recent samples (which is controlled by δ parameter), so the detector method must be adapted to let the process gathering enough samples as to ensure the histograms are consistent. Finally, the predictor shows a sensitive reaction that suggests the need of including a tolerance threshold to make the process more reliable.

To use the predictor for concept drift detection, we firstly compute mean and standard deviation (p_i and σ_i) of *G-mean* with decay factor following the incremental algorithm originally proposed in [5]—called ISMD in this paper—, so the most recent samples have a higher weight when computing these measures. This permits the algorithm a quicker reaction to changes and, more importantly, a better detection of light variations of prediction performance for a higher true detection rate.

HSP drift detector method is described in Algorithm 1. The main adaptations are as follows. Condition of line 6 ensures the mean (p_i) and standard deviation (σ_i) computation begins after sufficient number of samples is received for estimating the error (the histograms have enough information for the prediction). Condition of line 11 avoids early updates of minimum and maximum values until p_i and σ_i have been updated with enough amount of samples. Because we are interested in fluctuations of error, we monitor not only increasing trends but also decreasing ones. The maximum levels are maintained in line 15.

As ensured by line 20a, the concept drift is triggered when both increasing and decreasing trends happen from the instant when the levels start to be updated (i.e., from $0.5 \cdot \delta$ samples since last concept drift). The reason for considering both directions is to have the detection process stable before varying again, so we can distinguish error variations caused by new concept drifts from the delayed effect of the previously detected concept drift. This conjunctive condition ensures the curve adopts a valley or peak shape.

A second condition is also permitted to trigger concept drifts as fixed in line 20b. This is included to distinguish two different situations where both increasing and decreasing trends are activated. In a conventional procedure, decreases of the error are not considered as a sign of concept drift, only that the learner is improving its performance after gathering more and more samples. However, in our case, we can get this trend when a concept drift happens that let the histograms to have a better class discrimination. So, condition of line 20b triggers the drift detection if the decreasing is too relevant as to be caused by a concept drift and not an improvement of

Algorithm 1: HSP drift detector method

```
Parameters: $\delta$, $\alpha = \frac{\delta-1}{\delta}$, $b_i$, $\epsilon = 0.01$
Static variables: $t \leftarrow 1$, $t_{CD}$, $p_i$, $\sigma_i$, $p_{min}$, $\sigma_{min}$, $p_{max}$, $\sigma_{max}$, *first_max*
Data: $\mathbf{H}^{t-2}$, $\mathbf{M}^{t-2}$, $\mathbf{x}^{t-1}$
Input: $\mathbf{x}^t$, $y^{t-1}$
Output: $(t, \{\text{true, false}\})$
1  begin
2    concept_drift $\leftarrow$ false
3    if $t > 1$ then
4      Obtain $\hat{y}^{t-1}$ from $\mathbf{H}^{t-2}$ and $\mathbf{x}^{t-1}$    // eqs. (4-7)
5      Update $\mathbf{M}^{t-1}$ from $\mathbf{M}^{t-2}$, $\hat{y}^{t-1}$, and $y^{t-1}$
         // eqs. (8b-8f)
6      if $t_{CD} > 0.25 \cdot \delta$ then
7        Compute $G^{t-1}$ from $\mathbf{M}^{t-1}$    // eq. (8a)
8        $(p_i, \sigma_i) \leftarrow \texttt{IMSD}(1 - G^{t-1}, t_{CD}, \alpha)$
9      else
10       $p_i \leftarrow 0.25$, $\sigma_i \leftarrow 0$
11     if $(t_{CD} > 0.5 \cdot \delta)$ then
12       if $p_i + \sigma_i < p_{min} + \sigma_{min}$ then
13         $p_{min} \leftarrow p_i$, $\sigma_{min} \leftarrow \sigma_i$
14       if $p_i - \sigma_i > p_{max} - \sigma_{max}$ then
15         $p_{max} \leftarrow p_i$, $\sigma_{max} \leftarrow \sigma_i$
16         if $t_{CD} \leq 0.5 \cdot \delta + 1$ then
17           first_max $\leftarrow p_{max}$
18     increase $\leftarrow$
         $(p_i + \sigma_i \geq p_{min} + 3\sigma_{min} + \epsilon$ and $t_{CD} > \delta)$
19     decrease $\leftarrow$
         $(p_i - \sigma_i \leq p_{max} - 3\sigma_{max} - \epsilon$ and $t_{CD} > \delta)$
20     if decrease and ( increase or
         $((p_{min} - p_i > 2\epsilon)$ and $(p_{max} -$ first_max $< 2\epsilon))$ ) then
21       concept_drift $\leftarrow$ true
22   if $t = 1$ or concept_drift then
23     $p_i \leftarrow 0$, $\sigma_i \leftarrow 0$
24     $p_{min} \leftarrow 1$, $\sigma_{min} \leftarrow 0$
25     $p_{max} \leftarrow 0$, $\sigma_{max} \leftarrow 0$
26     first_max $\leftarrow 0$
27     $\mathbf{M}^{t-1} \leftarrow 0$
28     $\mathbf{H}^{t-1} \leftarrow 0$
29     $t_{CD} \leftarrow 0$
30     Reset static variables in IMSD
31   else
32     Update $\mathbf{H}^{t-1}$ from $\mathbf{H}^{t-2}$ and $(\mathbf{x}^{t-1}, y^{t-1})$    // eq. (1)
33   $t \leftarrow t + 1$
34   $t_{CD} \leftarrow t_{CD} + 1$
35   return $(t - 1,$ concept_drift$)$
```

the prediction.

When a concept drift is detected, lines 23-30 reset all the variables, including the histograms and the ones used for incremental computing of *G-mean*, its decayed mean and standard deviation. This procedure allows a quicker update of the histograms and performance estimation of the new concept.

Note that the time complexity of processing each sample is $\mathcal{O}(n \cdot m)$ (with n being the number of input variables or features and m the number of classes), which is due to lines 4 and 32. The space complexity is also $\mathcal{O}(n \cdot m)$, which is to store $\mathbf{H}$. This makes HSP extremely efficient.

IV. Experiments with Synthetic Data Streams

This section analyzes the behavior of the proposed HSP drift detector and compares it with other methods. For a clear observation, we need to know when the concept drifts are happening. Therefore, we design and discuss on synthetic data streams in this section. We choose the methods in MOA software [6] DDM, ADWIN, EDDM, CUSUM Test, and Page-Hinckley Test (PHT) to join our comparison. These well-known methods represent the different approaches to detect changes (sequential, control charts and monitoring two distributions) presented in [1]. All of them use Oza and Russell's incremental on-line bagging [7] with 10 trees as the base learner. All the methods are run with their default parameters.

A. SINE1 and SEA with strong/weak drifts and sudden/gradual changes

1) Data stream generators: This first experiment is based on two synthetic datasets (SINE1 and SEA) with the following data generators:

$$y_P(p,r) = \begin{cases} +1, & r < p \\ -1, & r \geq p \end{cases}, \; y_N(p,r) = \begin{cases} -1, & r < p \\ +1, & r \geq p \end{cases} \quad (9a)$$

$$y_{\text{SINE1}}(p, \mathbf{r}, \theta) = \quad (9b)$$

$$\begin{cases} [\mathbf{x}, y_P(p, r_2)] \text{ s.t.} \\ x_2 < sin(x_1 + \theta), \mathbf{x} \sim U([0,1])^2 \end{cases}, \quad r_1 < 0.5 \\ \begin{cases} [\mathbf{x}, y_N(p, r_2)] \text{ s.t.} \\ x_2 \geq sin(x_1 + \theta), \mathbf{x} \sim U([0,1])^2 \end{cases}, \quad r_1 \geq 0.5$$

$$y_{\text{SEA}}(p, \mathbf{r}, \theta) = \quad (9c)$$

$$\begin{cases} [\mathbf{x}, y_P(p, r_2)] \text{ s.t.} \\ x_1 + x_2 \leq \theta, \mathbf{x} \sim U([0,10])^3 \end{cases}, \quad r_1 < 0.5 \\ \begin{cases} [\mathbf{x}, y_N(p, r_2)] \text{ s.t.} \\ x_1 + x_2 > \theta, \mathbf{x} \sim U([0,10])^3 \end{cases}, \quad r_1 \geq 0.5$$

A real concept drift refers to a change in posterior probability $P(y|\mathbf{x})$ that involves an alteration of the true decision boundary. We will simulate this concept drift by two different settings.

The first setting is to switch the class concepts, which leads to a *strong drift* as the input area that previously belonged to class +1 (-1), now belongs to class -1 (+1). The second setting is to move the decision boundary, which leads to a *weak drift* where the input area is partially switched but still keeping some parts of the original class assignment. Weak drift is more difficult to be detected as the prediction performance degrades less, so delaying the drift detection and reducing the true detection rate.

We also consider two different drift settings in terms of the transition time. The first setting is to apply a *sudden change*, i.e., from the first received sample after drift happened the new labeling criterion (concept) is being applied. The second setting is to apply a *gradual change* where the probability of using the new labeling criterion gradually increases as samples are received. Gradual change adds another difficulty to drift detection as it may lead to delays and more false alarms.

Therefore, a total of eight data streams are included in this study by combining {SINE1, SEA}, {Strong, Weak}, and {Sudden, Gradual}. These data streams are generated by eqs. (10-17).

$$\text{SINE1-SS}(t) = \begin{cases} y_{\text{SINE1}}\big(1, U([0,1))^2, 0\big), & 1 \le t \le 1500 \\ y_{\text{SINE1}}\big(0, U([0,1))^2, 0\big), & 1501 \le t \le 3000 \end{cases} \tag{10}$$

$$\text{SEA-SS}(t) = \begin{cases} y_{\text{SEA}}\big(1, U([0,1))^2, 7\big), & 1 \le t \le 1500 \\ y_{\text{SEA}}\big(0, U([0,1))^2, 7\big), & 1501 \le t \le 3000 \end{cases} \tag{11}$$

$$\text{SINE1-SG}(t) = \begin{cases} y_{\text{SINE1}}\big(1, U([0,1))^2, 0\big), & 1 \le t \le 1500 \\ y_{\text{SINE1}}\big(1 - \frac{t-1500}{500}, U([0,1))^2, 0\big), & 1501 \le t \le 2000 \\ y_{\text{SINE1}}\big(0, U([0,1))^2, 0\big), & 2001 \le t \le 3000 \end{cases} \tag{12}$$

$$\text{SEA-SG}(t) = \begin{cases} y_{\text{SEA}}\big(1, U([0,1))^2, 7\big), & 1 \le t \le 1500 \\ y_{\text{SEA}}\big(1 - \frac{t-1500}{500}, U([0,1))^2, 7\big), & 1501 \le t \le 2000 \\ y_{\text{SEA}}\big(0, U([0,1))^2, 7\big), & 2001 \le t \le 3000 \end{cases} \tag{13}$$

$$\text{SINE1-WS}(t) = \begin{cases} y_{\text{SINE1}}\big(1, U([0,1))^2, 0\big), & 1 \le t \le 1500 \\ y_{\text{SINE1}}\big(1, U([0,1))^2, \pi - 1\big), & 1501 \le t \le 3000 \end{cases} \tag{14}$$

$$\text{SEA-WS}(t) = \begin{cases} y_{\text{SEA}}\big(1, U([0,1))^2, 7\big), & 1 \le t \le 1500 \\ y_{\text{SEA}}\big(1, U([0,1))^2, 13\big), & 1501 \le t \le 3000 \end{cases} \tag{15}$$

$$\text{SINE1-WG}(t) = \begin{cases} y_{\text{SINE1}}\big(1, U([0,1))^2, 0\big), & 1 \le t \le 1500 \\ y_{\text{SINE1}}\Big(1, U([0,1))^2, \frac{(\pi-1)\cdot(t-1500)}{500}\Big), & 1501 \le t \le 2000 \\ y_{\text{SINE1}}\big(1, U([0,1))^2, \pi - 1\big), & 2001 \le t \le 3000 \end{cases} \tag{16}$$

$$\text{SEA-WG}(t) = \begin{cases} y_{\text{SEA}}\big(1, U([0,1))^2, 7\big), & 1 \le t \le 1500 \\ y_{\text{SEA}}\big(1, U([0,1))^2, 7 + 6 \cdot \frac{t-1500}{500}\big), & 1501 \le t \le 2000 \\ y_{\text{SEA}}\big(1, U([0,1))^2, 13\big), & 2001 \le t \le 3000 \end{cases} \tag{17}$$

2) Parameter setting: The following parameter setting is used in our algorithm: $\delta = 300$ (therefore, $\alpha = \frac{\delta-1}{\delta} \approx 0.99667$), $b_i = 20$, and $\epsilon = 0.01$. Parameter δ, which defines the latency of the histograms, is critical to determine drift detection resolution. In a real-world problem, it can be also defined in terms of time (e.g., 20 seconds, 1 minute, etc.) depending of the required reaction time. Shorter values allow a faster reaction but make the performance more unstable. In our laboratory experiments with a total of 3 000 samples, we considered 300 (10% of the total sample size) as a plausible bracketing with a good balance between time reaction and stability.

A resolution of 20 bins is adopted. We experimentally found that at least 10 bins are required for a good level of sensitivity to detect concept drift. More than 30 bins are only required when small variations of the input values greatly change the assigned class.

For categorical variable, the number of bins is recommended to be equal to the number of categories if memory and time allow. The error tolerance ϵ is set to 0.01, i.e., 1% of *G-mean*. Higher values would make the method more reliable by reducing false alarms, but there would be more risk of missing true drifts.

3) Measures to assess concept drift detection performance: As pointed out in the introduction section, traditional drift analysis based on learner's performance may include some bias that distort the true behavior of the drift detector. Of course, in a real-world problem without any clue about when concept drifts are happening, analyzing the drift detector behavior indirectly through the prediction performance may be a reasonable approach. However, in synthetic data streams where know when and how the true concept drift happens, it is more appropriate to analyze the success rate in detecting the drifts as well as the time reaction. Moreover, average values also hide important information about the detection behavior so a direct look to how the time stamps of the detections are distributed can also provide valuable information. We will follow this analysis in this work.

We consider true detection to the first detection made with $t \ge 1501$. Other detections are considered false alarms, either because the detection is made before the drift or because it is the second or successive detections after the drift. Different measures that quantify the drift detection performance are analyzed, including:

1) *Delay* computes the average time stamp of the true detections minus 1 501, so as closer this value is to zero, quicker the reaction of the algorithm to the drift is, which is a desirable behavior (known as MTD—mean time to detection—in [8]).
2) The coefficient of variation (CV), i.e., the ratio of the sample standard deviation to the sample mean. The closer this value is to zero, the more robust the true detection behavior is.
3) True detection percentage (TD) is also computed; the closer the value is to 100%, the more effective the algorithm when detecting the drift is. It is known as 1-MDR–one minus missed detection rate—in [8].
4) FA assess the ratio of false alarms after the drift boundary; the lower is this value, the more reliable the detector is.
5) Besides, we use FA_0 as the false alarm ratio before the drift boundary. While FA_0 measures the sensitivity of the method in stationary data, FA represents the ability of the method to recover from a concept drift.
6) MTFA (mean time between false alarms) [8] is computed in our case (100 runs over 3 000 samples) as $\text{MTFA} = \frac{300000}{(\text{FA}+\text{FA}_0+1)}$.
7) The new measure MTR (mean time ratio) proposed in [9] to quantify the fast detection vs. false alarm

trade-off is also analyzed. It is computed as MTR $=$ $(\text{MTFA}/delay) \cdot (\text{TD}/100)$.

4) Results and analysis with SINE1 and SEA: It is common to perform the analysis over a unique instance of the data stream but with multiple runs for stochastic machine learners. The approach, however, may cause biases int the drift detection analysis as the randomness when generating a unique data set may include more or less difficulties. Instead, average results in many random data streams on the same data generator are analyzed in this paper. Ideally, multiple runs of stochastic learners should also be applied in each data set, but in terms of experimental plausibility, a single run can be accepted but with a different seed in each generation of data stream. In our study, we apply this methodology in 100 data streams for each of the 8 data generators. The same data stream instances are used in all algorithms.

The behavior of the proposed algorithm HSP is compared with other well-known approaches. The result is shown in Table I.

To facilitate comparison among the different methods, we have computed relative normalized values as follows. Let $f(p, m)$ be a specific measure given by method m in problem p. For example, $delay(\text{SINE1-SS}, \text{HSP}) = 39.52$ or $CV(\text{SEA-SG}, \text{DDM}) = 0.0122$ at Table I. For each problem and algorithm, we compute $\hat{f}(p, m) = \max_{\forall q, \forall r} f(q, r) - f(p, m)$, i.e., the maximum value obtained by any method in any problem minus the specific value in problem p and method m. It is applied in such measures to be minimized (*delay*, CV, FA, and FA_0). If the measure must be maximized (TD), it is computed as $\hat{f}(p, m) = f(p, m) - \min_{\forall q, \forall r} f(q, r)$. Then, average results of these relative measures for each method in all the problems is computed as $\overline{f}(m) = \frac{1}{8} \cdot \sum_{\forall p} \hat{f}(p, m)$. Finally, these values are normalized $\tilde{f}(m) = \frac{\overline{f}(m)}{\max_{\forall r} \overline{f}(r)}$.

With this procedure we can obtain a normalized measure that explains how close is each method to the best one, which gets value 1. We can then study the performance of the different drift detectors from four different properties: reactivity (by *delay*), robustness (by CV), effectiveness (by TD), and reliability (by $\text{FA}+\text{FA}_0$). Table II collects these relative results as well as the overall behavior computed as average MTR and its average rank.

We can observe that HSP, DDM, ADWIN and EDDM effectively detect TD. However, this can provide a misleading information if the method also get a high FA (poor reliability) since detecting many drifts necessarily allows to hit a true detection every once in a while. It is specially severe in EDDM, with a high FA_0 ratio that masks the true detection. This is the reason why EDDM shows the lower delay (quickest reactivity). This method, however, is not recommended in the analyzed problems. Despite CUSUM ranks the 3rd in MTR (because of its excellent reliability), it is achieved at the expense of a very poor reactivity and effectiveness, which makes this method inadvisable in these problems. ADWIN fails in reliability and is slightly worse than HSP and DDM when it comes to robustness. Indeed, HSP and DDM are the best two methods in these problems. The decision about which detector is preferable is difficult to make as while DDM shows a slightly quicker reaction than HSP, it makes it a bit more unstable with a greater FA ratio (poorer reliability). The very low FA ratio of HSP with a moderate delay makes this method as the best one according to MTR measure.

In conclusion, we can say that HSP behaves very good in the analyzed problems with a competitive performance compared with other well-known drift detectors, showing the best effectiveness and reliability with a satisfactory reactivity.

TABLE I: Results in the eight SINE1 and SEA problems

ds	method	delay	CV	TD	FA	FA_0	MTFA	MTR	rank
SINE1-SS	HSP	39.52	0.0035	100	0	0	300000.0	7591.09	1
	DDM	19.81	0.0019	100	17	4	13636.4	688.36	2
	ADWIN	35.68	0.0021	100	155	21	1694.9	47.50	4
	EDDM	32.26	0.0772	100	146	128	1090.9	33.82	5
	CUSUM	574.50	0.0153	100	0	0	300000.0	522.19	3
	PHT	827.88	0.0873	99	153	101	1176.5	1.41	6
SEA-SS	HSP	38.58	0.0042	100	0	0	300000.0	7776.05	1
	DDM	19.76	0.0023	100	19	0	15000.0	759.11	2
	ADWIN	36.00	0.0000	100	141	2	2083.3	57.87	5
	EDDM	19.49	0.0059	100	112	85	1515.2	77.74	4
	CUSUM	541.54	0.0102	100	0	0	300000.0	553.98	3
	PHT	757.91	0.0796	100	158	108	1123.6	1.48	6
SINE1-SG	HSP	161.06	0.0119	100	14	0	20000.0	124.18	2
	DDM	152.34	0.0139	100	52	4	5263.2	34.55	3
	ADWIN	165.92	0.0182	100	572	21	505.1	3.04	5
	EDDM	125.36	0.0193	100	204	128	900.9	7.19	4
	CUSUM	854.36	0.0158	100	0	0	300000.0	351.14	1
	PHT	1129.97	0.0778	97	130	101	1293.1	1.11	6
SEA-SG	HSP	159.47	0.0130	100	7	0	37500.0	235.15	2
	DDM	151.68	0.0122	100	60	0	4918.0	32.42	3
	ADWIN	166.24	0.0164	100	536	2	556.6	3.35	5
	EDDM	120.56	0.0233	100	204	85	1034.5	8.58	4
	CUSUM	825.28	0.0115	100	0	0	300000.0	363.51	1
	PHT	1071.43	0.0778	100	154	108	1140.7	1.06	6
SINE1-WS	HSP	84.02	0.0115	100	0	0	300000.0	3570.58	1
	DDM	52.44	0.0078	100	49	4	5555.6	105.94	2
	ADWIN	64.12	0.0173	99	143	21	1818.2	28.07	3
	EDDM	51.88	0.0189	100	146	128	1090.9	21.03	4
	CUSUM	1207.00	0.0477	2	0	0	300000.0	4.97	5
	PHT	1035.41	0.1257	17	9	101	2702.7	0.44	6
SEA-WS	HSP	110.45	0.0180	100	0	0	300000.0	2716.16	1
	DDM	72.79	0.0148	100	36	0	8108.1	111.39	2
	ADWIN	82.08	0.0165	100	154	2	1910.8	23.28	5
	EDDM	57.58	0.0183	100	116	85	1485.1	25.79	4
	CUSUM	1402.87	0.0223	15	0	0	300000.0	32.08	3
	PHT	1239.30	0.0977	23	14	108	2439.0	0.45	6
SINE1-WG	HSP	219.46	0.0271	100	18	0	15789.5	71.95	1
	DDM	186.35	0.0248	100	147	4	1973.7	10.59	2
	ADWIN	234.46	0.0497	99	179	21	1492.5	6.30	3
	EDDM	148.80	0.0330	100	215	128	872.1	5.86	5
	CUSUM	1443.00	0.0132	3	0	0	300000.0	6.24	4
	PHT	1095.38	0.0737	8	8	101	2727.3	0.20	6
SEA-WG	HSP	362.79	0.0275	100	0	0	300000.0	826.92	1
	DDM	259.03	0.0223	100	47	0	6250.0	24.13	2
	ADWIN	291.36	0.0320	100	252	2	1176.5	4.04	5
	EDDM	192.90	0.0363	100	139	85	1333.3	6.91	3
	CUSUM	1462.00	0.0128	2	0	0	300000.0	4.10	4
	PHT	1248.18	0.0912	11	2	108	2702.7	0.24	6

TABLE II: Relative results $\tilde{f}(m)$ of the analyzed methods in the eight SINE1 and SEA problems, average MTR and its rank

	Delay	CV	TD	FA+FA$_0$	MTR	
	reactivity	robustness	effectiveness	reliability	overall	rank
HSP	0.96104	0.98155	1.00000	0.99178	2864.01	1.250
DDM	0.98489	1.00000	1.00000	0.90662	220.81	2.250
ADWIN	0.97013	0.94253	0.99745	0.53120	21.68	4.375
EDDM	1.00000	0.85408	1.00000	0.55017	23.36	4.125
CUSUM	0.30925	0.94594	0.51786	1.00000	229.78	3.000
PHT	0.30058	0.32522	0.55995	0.69140	0.80	6.000

B. Rotating checkerboard

The proposed HSP method is based on univariate histograms, so it is mandatory to analyze its behavior in problems where the decision boundaries can be ortogonal to the variable axes, as in these cases the histograms could be evenly distributed in all the classes and so the straightforward prediction would be unable to effectively separate the class. Nevertheless, recall this prediction mechanism is not designed to do a sound classification, only to monitor changes of the decision boundaries.

Therefore, if the currently coming data stream cannot be correctly predicted, *G-mean* would stay around 0.5. What we expect to happen is that this measure will grow as soon as a drift occurs and so, HSP should detect the change and issue the concept drift alarm.

For this experiment, we find the rotating checkerboard (CB) problem very interesting, which consists of two variables labeled with canonical XOR where the decision boundaries can be rotated.

1) CB data stream generator: CB data can be generated with (18). The concept drift is introduced by (19), which produces drifts with variable intensity but sudden change every 2 500 samples for a total of three concept drifts after 10 000 samples.

$$\begin{aligned} s(\mathbf{x},\theta) &= \left\lfloor \frac{x_1 \cdot \cos\theta - x_2 \cdot \sin\theta}{0.5} \right\rfloor + \left\lfloor \frac{x_1 \cdot \cos\theta + x_2 \cdot \sin\theta}{0.5} \right\rfloor \\ y(\mathbf{x},\theta) &= 2 \cdot (|s(\mathbf{x},\theta)| \,\%\, 2) - 1 \qquad (18) \\ y_{\mathrm{CB}}(\theta) &= [\mathbf{x}, y(\mathbf{x},\theta)]\,, \mathbf{x} \sim U([0,1])^2 \end{aligned}$$

$$\text{CB-VS}(t) = \begin{cases} y_{\mathrm{CB}}\left(\frac{3\pi}{4}\right), & 1 \le t \le 2500 \\ y_{\mathrm{CB}}\left(\frac{\pi}{2}\right), & 2501 \le t \le 5000 \\ y_{\mathrm{CB}}\left(\frac{5\pi}{4}\right), & 5001 \le t \le 7500 \\ y_{\mathrm{CB}}\left(\frac{7\pi}{4}\right), & 7501 \le t \le 10000 \end{cases} \qquad (19)$$

Transitions from concept 1 to 2 and from 2 to 3 are weak drifts (in the sense that not all the input area changes the assigned class after drift) while the change from concept 3 to 4 is a strong drift (all the input area switches the class). Therefore, concept drifts 1-2 and 2-3 are more difficult to be detected than concept drift 3-4. Besides, concept 2 presents decision boundaries ortogonal to the axes, which makes the straightforward prediction unable to discriminate the classes. Now let us see how HSP detects concept drifts.

The same parameter setting used for HSP in the previous experiment applies here ($\epsilon = 0.01$ and $b_i = 20$) with the exception of δ (and subsidiarily α), that needs to be proportional to the drift resolution we want to detect; $\delta = 500$ is used in CB-VS.

Figure 1 presents the obtained results in CB-VS. HSP shows very good detection results, which is the only method to be able to detect 100% of TD in concept drift 1-2 and 3-4. Besides, the detection locations are compact (low CV) and with a short delay (good reactivity). HSP also shows 0% as FA ratio (high reliability). ADWIN is the second best method detecting TD, but with high FA ratio and CV (specially after the two first weak concept drifts). DDM works fine in terms of reliability (low FA ratio) but fails to detect TD in the two weak concept drifts at $t = 2501$ and $t = 5001$. EDDM generates again high FA that reduces its reliability in this problem.

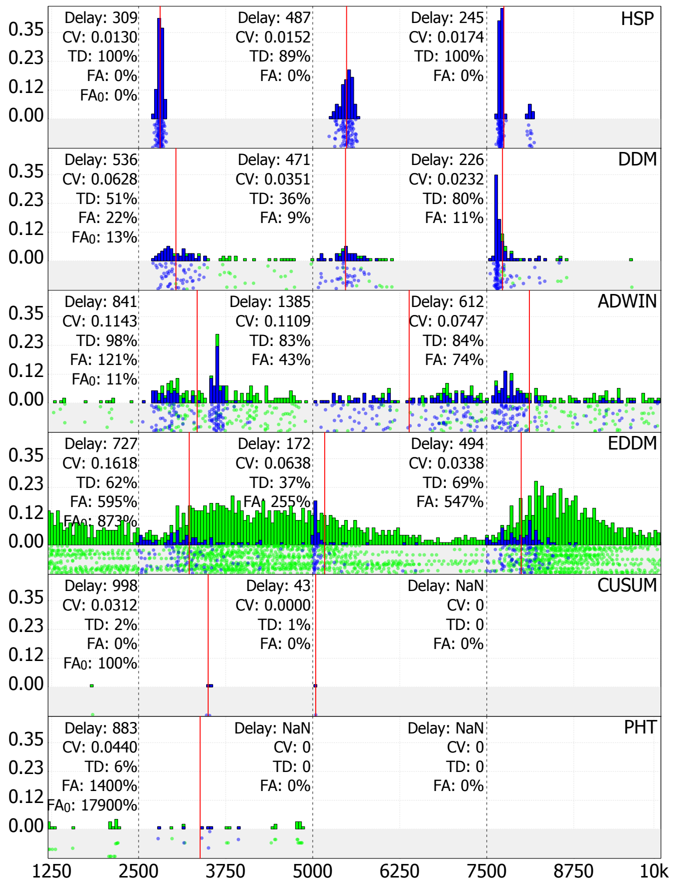

Fig. 1: Results in CB-VS (variable drift, sudden change). A row per algorithm is included. The abscissa shows the time stamp, with a dashed line marking the time when the concept drifts start. The gray area in each row includes a circle for each concept drift detected (with ordinate axis corresponding to the run when it was detected). Blue circles correspond to true detection while the green ones are false alarms. The white areas include the stacked histograms (resolution of 200 bins) of the detection locations with the same blue and green colors in order to see the distribution of true and false alarms. The vertical red lines show the average value of true detections. The measures that summarizes the results are located at the left side of each concept drift boundary

TABLE III: Efficiency of HSP and DDM methods in the analyzed problems (time in milliseconds)

Data stream				HSP					DDM	
name	class labels	features	samples	bins	time/ samples	time/ (samples·bins)	time	Hz	time	Hz
SINE1-SS	2	2	3 000	80	0.02466667	0.00030833	74	40 541	988	3 036
SINE1-SG	2	2	3 000	80	0.02466667	0.00030833	74	40 541	1 008	2 976
SINE1-WS	2	2	3 000	80	0.02466667	0.00030833	74	40 541	1 077	2 786
SINE1-WG	2	2	3 000	80	0.02466667	0.00030833	74	40 541	973	3 083
SEA-SS	2	3	3 000	120	0.02933333	0.00024444	88	34 091	1 040	2 885
SEA-SG	2	3	3 000	120	0.02933333	0.00024444	88	34 091	967	3 102
SEA-WS	2	3	3 000	120	0.02933333	0.00024444	88	34 091	987	3 040
SEA-WG	2	3	3 000	120	0.02933333	0.00024444	88	34 091	1 155	2 597
CB-VS	2	2	10 000	80	0.02520000	0.00031500	252	39 683	15 146	660
Electricity	2	7	45 312	310	0.05863789	0.00018915	2 657	17 054	596 081	76
Airlines	2	7	539 378	250	0.05240295	0.00020961	28 265	19 083	53 244 877	10

V. Efficiency Analysis

The previous section proved the proposed HSP method can effectively detect concept drifts with a competitive performance compared to other well-known drift detectors. This section looks into its efficiency, particularly the runtime and memory usage of HSP. Efficient methods are always desirable in machine learning, but in the context of data stream mining this issue is mandatory as this will define the rate of arrival data the method can process within a limited memory usage, and its applicability in real-world applications.

We have included two real-world data stream problems (Electricity and Airlines) [6]. The task of Electricity is to predict tendency in electricity market with a total of 45 312 samples (more than two years and half). Each sample is collected every 30 minutes, including 7 numeric variables (*date* feature is ignored), and two classes. The Airlines problem is about flight delays and contains a total of 539 378 samples (values of 31 days with about 17 400 flights on average per day). Each sample has 7 variables (4 categorical and 3 numeric), and two classes. The same $\epsilon = 0.01$ is again used here. The number of bins are again $b_i = 20$ in numeric variables and categorical ones except those with a low cardinality: in Electricity, 7 bins are used in *weekday* and 48 in *period*; in Airlines, 18 bins are used in the feature *airline* and 7 in *day of week*.

Table III collects some information about dimensionality of the data and efficiency of HSP and DDM methods. In HCS, *bins* refers to the total number of bins used (memory usage). Time is in milliseconds and Hz refers to the number of samples that the method can process per second. The experiments where run in a laptop with Intel® Core$^{\text{TM}}$ i7-4500U processor.

We can observe HSP runs 10 times faster than DDM in SINE1 and SEA, 60 times faster in CB, 200 times faster in Electricity, and 1 800 times faster in Airlines. This dramatic reduction in time makes HSP able to process between 7 000 and 40 000 samples per second, which definitely allows us to deal with high-speed data stream problems unachievable for other concept drift detectors.

VI. Conclusion

The paper proposed a new drift detector method (HSP) for data stream mining. Different from the conventional approaches, it does not track the performance of the learner. Instead, it employs histograms to summarize data distributions per class and develops a simple class prediction to measure changes in decision boundaries. HSP is proved to perform better than other well-known drift detectors in several synthetic data streams. Its performance in real-world data streams is also analyzed. One advantage of HSP is its extraordinary efficiency in time and memory, which enables it to process thousands of samples per second, being up to 1 800 times faster than other drift detectors in the performed experiments.

As our further work, we plan to analyze the method in problems with more than two classes and to study the impact of this drift detector on the learner's performance.

References

[1] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, pp. 44:1–44:37, Mar. 2014.

[2] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Advances in Artificial Intelligence – SBIA 2004: 17th Brazilian Symposium on Artificial Intelligence, Sao Luis, Maranhao, Brazil, September 29 - October 1, 2004. Proceedings*, A. L. C. Bazzan and S. Labidi, Eds. Berlin, Heidelberg: Springer, 2004, pp. 286–295.

[3] M. Baena-García, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavaldá, and R. Morales-Bueno, "Early drift detection method," in *Fourth International Workshop on Knowledge Discovery from Data Streams*, 2006.

[4] A. Bifet and R. Gavaldà, *Adaptive Learning from Evolving Data Streams*. Berlin, Heidelberg: Springer, 2009, pp. 249–260.

[5] E. Ruiz and J. Casillas, "Adaptive fuzzy partitions for evolving association rules in big data stream," *International Journal of Approximate Reasoning*, vol. 93, pp. 463–486, 2018.

[6] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "MOA: massive online analysis," *Journal of Machine Learning Research*, vol. 1, pp. 1601–1604, 2010.

[7] N. C. Oza and S. Russell, "Online bagging and boosting," in *Artificial Intelligence and Statistics*. Morgan Kaufmann, 2001, pp. 105–112.

[8] F. Gustafsson, *Adaptive Filtering and Change Detection*. John Wiley & Sons, Ltd, 2001.

[9] A. Bifet, J. Read, B. Pfahringer, G. Holmes, and I. Žliobaitė, "CD-MOA: Change detection framework for massive online analysis," in *Advances in Intelligent Data Analysis XII: 12th International Symposium, IDA 2013, London, UK, October 17-19, 2013. Proceedings*, A. Tucker, F. Höppner, A. Siebes, and S. Swift, Eds. Berlin, Heidelberg: Springer, 2013, pp. 92–103.