Expert Systems with Applications 41 (2014) 8144–8156

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Review

A comparative study on concept drift detectors

Paulo M. Gonçalves Jr. [a,*], Silas G.T. de Carvalho Santos [b], Roberto S.M. Barros [b], Davi C.L. Vieira [b]

[a] Instituto Federal de Educação, Ciência e Tecnologia de Pernambuco, Brazil
[b] Centro de Informática, Universidade Federal de Pernambuco, Brazil

ARTICLE INFO

Article history:
Available online 19 July 2014

Keywords:
Data streams
Time-changing data
Concept drift detectors
Comparison

ABSTRACT

In data stream environments, drift detection methods are used to identify when the context has changed. This paper evaluates eight different concept drift detectors (DDM, EDDM, PHT, STEPD, DOF, ADWIN, Paired Learners, and ECDD) and performs tests using artificial datasets affected by abrupt and gradual concept drifts, with several rates of drift, with and without noise and irrelevant attributes, and also using real-world datasets. In addition, a 2^k factorial design was used to indicate the parameters that most influence performance which is a novelty in the area. Also, a variation of the Friedman non-parametric statistical test was used to identify the best methods. Experiments compared accuracy, evaluation time, as well as false alarm and miss detection rates. Additionally, we used the Mahalanobis distance to measure how similar the methods are when compared to the best possible detection output. This work can, to some extent, also be seen as a research survey of existing drift detection methods.

1. Introduction

Learning from data is a topic that is addressed by several research fields, such as data mining, machine learning, and pattern recognition. Traditionally, this learning is performed in static environments, where a dataset is available for the classifier to read as many times as needed for training. Also, another characteristic is that the target concept to be learned is fixed. Several classifiers have been proposed and, nowadays, many efficient classifiers are available.

Currently, however, several expert systems have to deal with data that flow continuously. They can therefore not be stored for later analysis and must be processed as they arrive. Examples of applications that have to handle this requirement, known as *data streams*, are TCP/IP traffic, GPS data, sensor networks, and customer click streams (Gama, 2010). Compared to batch learning, data stream processing imposes restrictions on memory usage, limited testing time and learning, and one-time reading data.

Another difficulty of data streams is that the target concept may change in time, usually after a minimum stability period (Gama, 2010), a problem known as *concept drift*. The problem of concept drift has received a lot of attention over the past few years, mainly because it negatively impacts the accuracy of the classifiers that learned on the basis of past training instances. Some examples of situations where concept drifts may occur are "spam, fraud or climate change detection" (Elwell & Polikar, 2011). Concept drift may be classified in terms of the speed of change and the reason of change. Considering the speed of change, an *abrupt concept drift* occurs when a change between two contexts happens suddenly, while a *gradual concept drift* describes the case where the transition between two contexts occurs smoothly.

With respect to the reason of change, a *real concept drift* occurs "when a set of examples has legitimate class labels at one time and has different legitimate labels at another time" (Kolter & Maloof, 2007). On the other hand, a *virtual concept drift* occurs when "the target concepts remain the same but the data distribution changes" (Delany, Cunningham, Tsymbal, & Coyle, 2005). In practice, however, "virtual concept drift and real concept drift often occur together" (Tsymbal, Pechenizkiy, Cunningham, & Puuronen, 2008).

Several approaches have been proposed to deal with concept drift, including (a) adapting a classifier's internal structure and (b) using ensemble classifiers. In this paper, we focus on the methods used to identify the occurrence of a concept drift. Based on the concept drift identification, these methods can be used inside other classifiers to modify either the internal structure or the number of classifiers in an ensemble. Examples of classifiers that internally use concept drift detection methods are Diversity for Dealing with Drifts (DDD) (Minku & Yao, 2012) and Recurring Concept Drifts (RCD) (Gonçalves & Barros, 2013). In addition, they can be used in conjunction with any classifier to identify a drift. Drift detection

* Corresponding author.
E-mail addresses: paulogoncalves@recife.ifpe.edu.br (P.M. Gonçalves Jr.), sgtcs@cin.ufpe.br (S.G.T. de Carvalho Santos), roberto@cin.ufpe.br (R.S.M. Barros), dclv@cin.ufpe.br (D.C.L. Vieira).

http://dx.doi.org/10.1016/j.eswa.2014.07.019
0957-4174/

methods usually use a specific classifier (also called "base learner") to analyze its accuracy and indicate when a drift has occurred.

The objective of this paper is to compare several concept drift detection methods and analyze under which conditions they perform well. In every study that proposes a new drift detection method a number of tests are performed to verify its usefulness. Unfortunately, the base learner, the datasets, the metrics, and the types of drifts used in the experiments vary considerably, as well as the other methods with which they are compared. This makes it difficult to choose which method to use in different situations.

Thus, in this paper eight concept drift detection methods (DDM (Gama, Medas, Castillo, & Rodrigues, 2004), EDDM (Baena-García et al., 2006), PHT (Page, 1954), ADWIN (Bifet & Gavaldà, 2007), Paired Learners (Bach & Maloof, 2008), ECDD (Ross, Adams, Tasoulis, & Hand, 2012), DOF (Sobhani & Beigy, 2011), and STEPD (Nishida & Yamauchi, 2007)) were compared in terms of accuracy, evaluation time, false alarm and miss detection rates, as well as distance to the drift point. These methods were selected from the ones with the highest number of citations, provided that there was a freely available implementation or at least a detailed description of the algorithm to allow a direct implementation and avoid incorrect results and/or slow performance. The experiments were performed using the most cited datasets in the area, including artificial datasets affected by abrupt and gradual concept drifts, with variable rates of drift, and real-world ones. To the best of our knowledge, this is the broadest experiment to date comparing a wide range of concept drift detectors in datasets with different characteristics and using different metrics.

The rest of this paper is organized as follows: Section 2 presents the drift detection methods used in the experiments and the techniques they use to identify concept drifts. Section 3 describes the parameters used in the drift detectors, the datasets used in the experiments, and the adopted evaluation methodology. The results obtained in the experiments are analyzed in Section 4. Finally, Section 5 presents our conclusions.

2. Background

Concept drift detection methods use a base learner (classifier) to classify incoming instances. For each instance, it outputs a class prediction, which is usually compared to the true class label. Based on the classification result (true for correct classification and false otherwise), the drift detection method can indicate whether a drift has occurred or not. Finally, the base learner is trained on the instance. This process is repeated for each incoming instance.

Several concept drift detection methods have already been proposed, one of which is the Drift Detection Method (DDM) (Gama et al., 2004). This uses a base learner to classify incoming instances and the classification result is used to compute the online error-rate of the base learner. The classification result indicates whether the base learner classified the arriving instance correctly or not. If the base learner correctly classifies the actual instance, the error-rate decreases. DDM considers that, when the concept changes, the base learner will incorrectly classify the arriving instances that are created based on a different data distribution. Thus, if the error-rate increases, it is an indication of a concept drift.

On the other hand, while the distribution is stationary, i.e., it remains unchanged, the error rate decreases. Therefore, the error rate (p_i) and the standard deviation $\left(s_i = \sqrt{p_i(1-p_i)/i}\right)$ are computed and these values are stored when $p_i + s_i$ reaches its minimum (obtaining p_{min} and s_{min}). When $p_i + s_i \geqslant p_{min} + 2 \cdot s_{min}$, a warning level is reached and instances are stored in anticipation of a possible concept drift. If $p_i + s_i \geqslant p_{min} + 3 \cdot s_{min}$, a drift level is reached, indicating a context change. The base learner and the values of p_{min} and s_{min} are then reset and a new base learner is trained on the examples stored since the warning level.

The Early Drift Detection Method (EDDM) (Baena-García et al., 2006) is similar to DDM but, instead of using the error rate, it uses the distance-error-rate of the base learner to identify whether a drift has occurred. This metric computes the number of examples between two classification errors. When there is no concept drift, the base learner improves its predictions and the distance between errors increases. On the other hand, when a concept drift occurs, the base learner makes more mistakes and the distance between error decreases. As described in the original paper, EDDM is best suited to dealing with slow gradual concept drifts. The average distance between two errors (p_i) and its standard deviation (s_i) are computed. These values are stored when $p_i + 2 \cdot s_i$ reaches its maximum value (obtaining p_{max} and s_{max}). This value indicates that the base learner best approximates the current concept.

Like DDM, EDDM defines two thresholds. When $(p_i + 2 \cdot s_i)/(p_{max} + 2 \cdot s_{max}) < \alpha$, the warning level is reached and the instances are stored anticipating a concept drift. The drift level is reached when $(p_i + 2 \cdot s_i)/(p_{max} + 2 \cdot s_{max}) < \beta$, indicating a change in the context. The values of α and β are 0.95 and 0.9, respectively, the same chosen in the original paper after some experimentation. The base learner and the values of p_{max} and s_{max} are reset and a new base learner is trained on the examples stored since the warning level.

The Page-Hinkley Test (PHT) (Page, 1954) is a sequential analysis technique that can be used as a concept drift detector. It computes the observed values (here we used the actual accuracy of the classifier) and their mean up to the current moment. When a concept drift occurs, the base learner will fail to correctly classify incoming instances, making the actual accuracy decrease. As a result, the average accuracy up to the current moment also decreases. The cumulative difference between these two values (U_T) and the minimum difference between these two values (m_T) are computed. Higher U_T values indicate that the observed values differ considerably from their previous values. When the difference between U_T and m_T is above a specified threshold corresponding to the magnitude of changes that are allowed (λ), a change in the distribution is detected. Higher λ values result in fewer false alarms, but might miss or delay some changes.

Adaptive Windowing (ADWIN) (Bifet & Gavaldà, 2007) is another drift detection method. It uses sliding windows of variable size, which are recomputed online according to the rate of change observed from the data in these windows. The algorithm dynamically enlarges the window (W) when there is no apparent change in the context, and shrinks it when a change is detected. The algorithm attempts to find two sub-windows of W that exhibit distinct averages. If that occurs, it concludes that the corresponding expected values are different, meaning that the older portion of the window is based on a data distribution different than that of the present one, and is therefore dropped. The maximum length of the window is "statistically consistent with the hypothesis that there has been no change in the average value inside the window" (Bifet, Holmes, Pfahringer, & Frank, 2010). Additionally, ADWIN provides rigorous guarantees of its performance, in the form of limits on the rates of false positives and false negatives.

The Paired Learners (PL) (Bach & Maloof, 2008) method, as the name suggests, uses two learners: a stable and a reactive one. The stable learner predicts based on all of its experience, while the reactive one predicts based on a window of recent examples. A circular list of bits with length w (the same length as the window) stores the value *one* if an instance was incorrectly classified by the stable learner and correctly classified by the reactive learner, and *zero* otherwise. If the number of bits of this circular list set to *one* exceeds a parameterized value θ, it indicates that the reactive learner, trained on recent instances, has a better predictive

accuracy than the stable learner. Thus, the stable learner is replaced by the reactive one, and all the bits in the list are set to *zero*.

It should be noted that lower w values are best suited to identifying abrupt concept drifts because the reactive learner will rapidly beat the performance of the stable one, replacing it more rapdly, but might also increase the number of false alarms. Lower θ values might make the reactive learner replace the stable one too early, raising more false alarms. The reactive learner can be implemented in two different ways: (a) by rebuilding the learner with the last w examples, or (b) by using a retractable learner that can unlearn examples. The former has the advantage of being general and working for all kinds of learners (batch and online) and the disadvantage of increasing the running time with larger w values.

Concept Drift Detection (ECDD) (Ross et al., 2012) is a proposal for a drift detection method based on Exponentially Weighted Moving Average (EWMA) (Roberts, 1959), used for identifying an increase in the mean of a sequence of random variables. Considering that, in EWMA, the probability of incorrectly classifying an instance before the change point and the standard deviation of the stream are known in advance, an EWMA change detector for the Bernoulli distribution was proposed (Yeh, Mcgrath, Sembower, & Shen, 2008). This is the probability distribution of a random variable which takes value 1 with success probability p and value 0 with failure probability $q = 1 - p$. Here, the success probability is the probability of correctly classifying an instance. In ECDD, these values are computed online, based on the classification accuracy of the base learner in the actual instance, together with an estimator of the expected time between false positive detections. Two estimations are compared: one with more weight on recent examples and another with similar emphasis on both recent and old data. When the difference between these two estimations exceeds a certain parameterized threshold, a concept drift is identified. The authors also mention a warning level that is raised when the difference between these two estimations is approaching the drift level.

Statistical Test of Equal Proportions (STEPD) (Nishida & Yamauchi, 2007) computes the accuracy of the base learner in the W most recent instances and compares it to its overall accuracy from the beginning of the learning process. It assumes that "the accuracy of a classifier for recent W examples will be equal to the overall accuracy from the beginning of the learning if the target concept is stationary; and a significant decrease of recent accuracy suggests that the concept is changing." (Nishida & Yamauchi, 2007). A chi-square test is performed by computing a statistic and its value is compared to the percentile of the standard normal distribution to obtain the observed significance level. If this value is less than a significance level, then the null-hypothesis is rejected, assuming that a concept drift has occurred. STEPD also uses warning and drift thresholds, similar to the ones presented by DDM, EDDM, PHT, and ECDD.

The Degree of Drift (DOF) (Sobhani & Beigy, 2011) method detects drifts by processing data chunk by chunk, computing the nearest neighbor in the previous batch for each instance in the current batch and comparing their corresponding labels. A distance map is created, associating the index of the instance in the previous batch and the label computed by its nearest neighbor. A metric known as degree of drift is computed based on the distance map. The average and standard deviation of all degrees of drift are computed and, if the current value is away from the average more than s standard deviations, a concept drift is raised, where s is a parameter of the algorithm. As described in the original paper (Sobhani & Beigy, 2011), "the proposed drift detection algorithm is more effective for problems with well separated and balanced classes".

It is important to note that the ability of the algorithms to deal with continuous or nominal data depends mainly on the classifier used as base learner, which can be naive Bayes, a decision tree, a neural network, or others. The only exception is DOF, which internally computes the distance between instances based on the type of the attribute. If it is continuous, the Euclidean distance is computed (without the square root). If it is nominal and the values are different, the distance is incremented by 1.0 (one). The values are added and then squared to compute the final distance. Thus, all the methods presented here can deal with both continuous and nominal data.

An example of an expert system that deals with concept drift and could thus benefit from the use of a drift detection method is known as SpamHunting (Fdez-Riverola, Iglesias, Díaz, Méndez, & Corchado, 2007), used to tackle concept drift in spam filtering. A concept drift detection method can be used to detect when the context has changed, for example, the type of messages a given user receives are different due to new job, studies, hobbies, etc., and therefore the base learner must be updated or replaced by a new one trained on recent data. Another example of an expert system tries to mine decision rules that govern the concept drift (Tsai, Lee, & Yang, 2009). Here, again, there is a clear need to identify when a concept drift has occurred in order to select the correct instances to be used to extract the decision rules.

3. Experiment configuration

In this section we describe the datasets and the parametrization used in the drift detectors as well as the evaluation methodology used in the experiments.

3.1. Datasets

The datasets chosen for the experiments have all been used previously in the concept drift research area. To analyze the performance of the methods, artificial datasets affected by abrupt and gradual concept drifts, as well as real-world datasets, were used. The synthetic datasets allow us to analyze how the methods deal with the types of drift included in the datasets, as it is known in advance when the drifts begin and end.

Two datasets with abrupt concept drifts were selected: Sine (Baena-García et al., 2006; Gama et al., 2004; Ross et al., 2012) and Stagger (Schlimmer & Granger, 1986; Bach & Maloof, 2008). Sine presents the problem of identifying the position of coordinates, represented by two attributes, in relation to the curve $y = \sin(x)$. In the first context, points below the curve are classified as positive. After each concept drift, the classification is reversed. Each coordinate has values uniformly distributed in the [0,1] interval. Two other attributes, filled with random data in the same interval, are added to each instance with no influence on the classification function (irrelevant data). Positive and negative examples are interchanged to ensure a stable learning environment similar to the ones described at (Baena-García et al., 2006; Gama et al., 2004).

The Stagger dataset has three attributes, each of which has three possible values, and one class with two values. Each example consists of the following attributes: shape $\in$ {triangle, circle, rectangle}, color $\in$ {green, blue, red}, and size $\in$ {small, medium, large}. The Stagger concepts are represented by three boolean functions: (1) color = red $\wedge$ size = small, (2) color = green $\vee$ shape = circle, and (3) size = medium $\vee$ size = large. This dataset can generate 27 different combinations of the three attributes (3^3). The three concepts are not mutually exclusive, i.e., some combinations of the attributes maintain the same class as the previous concept while others change. For example, from concept 1 to concept 2, 11 combinations share the same class while in the other 16 combinations the class values are different. From concept 2 to concept 3, 14 combinations are similar, while the other 13 have different class values,

indicating that the change from concept 2 to 3 is more complex than the change from concept 1 to 2. This is a noise-free dataset.

Two datasets affected by gradual concept drifts were chosen: Hyperplane (Bifet, Holmes, Pfahringer, Kirkby, & Gavaldà, 2009; Hulten, Spencer, & Domingos, 2001) and Mixed (Baena-García et al., 2006; Gama et al., 2004). Hyperplane is a two-class dataset that models a rotating hyperplane in a d-dimensional space. It is represented by the set of points x that satisfy $\sum_{i=1}^{d} w_i x_i = w_0$, where x_i is the ith coordinate of x. Examples below w_0 are labeled negative; the others are labeled positive. In the following experiments, this dataset consists of 10 drifting attributes, 5% of noise, and 10% of chance that the direction of change is reversed.

The Mixed dataset has two boolean (v and w) and two numerical (x and y) attributes. Three conditions are verified for these attributes: v, w, and $y < 0.5 + 0.3\sin(3\pi x)$. If at least two of these conditions are satisfied, the example is considered positive. When a drift occurs, the classification is reversed. Concept drift is produced by gradually choosing examples from the old and new concepts, reducing the probability of selecting examples from the old context while increasing the probability of selecting them from the new context. In this dataset, we have an initial stable context that gradually changes to a new stable context, unlike Hyperplane, which is always changing. This is a noise-free dataset.

Experiments were also performed using four real-world datasets, which are described below. The first three can be obtained at http://moa.cms.waikato.ac.nz/datasets/ and the last one at http://users.rowan.edu/polikar/research/nse/.

Covertype (Bifet et al., 2010) contains 581,012 instances and 54 attributes including both numeric and categoric ones. It represents the forest cover type for 30 m^2 cells obtained from the US Forest Service Region 2 Resource Information System data. The goal is to predict the forest cover type from cartographic variables.

The Electricity dataset (Baena-García et al., 2006; Gama et al., 2004; Kolter & Maloof, 2007), consisting of 45,312 instances and 8 attributes, presents the problem of predicting whether the price in the Australian New South Wales Electricity Market will increase or decrease. In this market, the prices are variable, based on market demand and supply, and are set every five minutes. The class label identifies the change in the price related to a moving average over the last 24 h.

The Poker Hand (Bifet et al., 2010) dataset represents the problem of identifying the hand in a Poker game. It consists of 829,217 instances and 10 attributes: each instance represents a hand comprising of five cards, each of which is described by a rank and a suit. In the experiments we used a version in which the cards were sorted by rank and suit, and duplicates were removed.

The Nebraska Weather Prediction (Elwell & Polikar, 2011), part of the U.S. National Oceanic and Atmospheric Administration, is the last real dataset tested. It provides 50 years of daily measurements with several meteorological data such as wind speed, pressure, temperature, among others, offering diverse weather patterns.

3.2. Evaluation setup

To evaluate the drift detection methods using the presented datasets, we used a methodology similar to that described by (Elwell & Polikar, 2011). For each time step T, a specified amount of instances of the training set (t) are read and used to train the classifier. Next, other examples of the testing set (d) are used to test the classifier. This procedure is repeated 50 times, and a confidence interval is computed. The values of t and d are different for each dataset. For each one, two configurations were tested to analyze the influence of (a) the number of training examples in the datasets with abrupt drifts and (b) the speed of change in the ones with gradual drifts.

We used Naive Bayes (NB) as base learner in all tested drift detection methods because of its speed, simplicity, freely available implementations, and widespread use in experiments in the data stream research area. The experiments described here used the Massive Online Analysis (MOA) framework (Bifet, Holmes, Kirkby, & Pfahringer, 2010). `SingleClassifierDrift` is a classifier implemented in MOA which tests each incoming instance using a base learner, which returns a boolean value indicating whether it correctly classified the instance or not. This value is passed to a parameterized drift detector, which will flag the example for no drift, warning level, or drift level. If no drift is identified, the base learner is trained on the instance that has just arrived. If the warning level is reached, a new learner is built and both are trained. If the drift level is reached, the learner built on instances obtained in the warning level is used as the new base learner.

Two conditions must therefore be met to use `SingleClassifierDrift`: (a) the drift detection method has to flag each incoming instance based solely on the boolean value indicating if it was correctly classified or not, and (b) the drift detection method has to use the three flags (no drift, warning, and drift).

MOA already had implementations of DDM and EDDM as drift detection methods of `SingleClassifierDrift`. Three other methods that satisfy these requirements were implemented: ECDD, PHT, and STEPD. The PL and DOF methods have also been implemented in MOA by us, but as single classifiers because they do not meet condition (a). Finally, the ADWIN method had already been implemented in MOA, but had mainly been used in other classifiers. As ADWIN does not have a warning level, it was not possible to directly port it to use `SingleClassifierDrift`. Thus, we implemented a new classifier that uses ADWIN internally.

3.3. Parameter configuration

We initially performed experiments to identify the best parameter settings for each of the methods. Specifically, we adopted a novel use of the 2^k factorial design to discover which of the parameters have the greatest influence on the results. It is important to note that none of the drift detection methods presented has performed such a task, nor have subsequent papers. We argue this is of the utmost importance to avoid wasting time adjusting irrelevant or unimportant parameters. After identifying the parameters that most influence performance in the 2^k factorial design, we tested other values for these parameters attempting to find values that outperform the ones used previously.

We statistically compared the tested versions in the presented datasets computing the F_F statistic (Iman & Davenport, 1980), derived from the Friedman non-parametric statistical test (Friedman, 1940) as described by Demšar (2006). This test informs us whether any of the methods have a performance statistically different from the others. If this is the case, we perform two post hoc tests. The first of these is the Nemenyi test (Nemenyi, 1963) with 95% confidence. It is used to identify the methods with statistical differences in accuracy, comparing all methods against each other. The second one is the Holm procedure (Holm, 1979). This is a test that controls the family-wise error when comparing all methods with a specific one (here we chose NB as we wanted to verify whether the methods really behave better than the base learner). These tests were performed to finally select the best parameter set to be used by each drift detection method in the experiments.

The DDM implementation available at MOA offers only one parameter: the minimum number of instances before permitting the detection of a change (n). We modified the method to add another two parameters, including the number of standard deviations to warn (w) and detect (d) drifts. The original method uses two standard deviations for the warning level, three for the drift level, and $n = 30$. Tests were performed using the following values for the

parameters: $n \in \{15, 30\}, d \in \{2.5, 3.0\}$, and $w \in \{1.5, 2.0\}$. Our results indicated that the drift level was the parameter with the greatest influence. The influence of the parameters in the artificial datasets was small but in the real-world ones d accounted for more than 52% of the differences in performance, followed by n, with 16.55% and the combined influence of d and n, with 11.47%. Thus, we varied the d value (adjusting the value of w accordingly, as d must be higher than w) and performed more tests. After computing F_F, comparing all tested versions of the DDM method, results indicated that there were versions with statistically different results. The analysis of the tested versions with the Holm procedure revealed that the parameter set with the best results was $d = 2.5, w = 2$, and $n = 30$, thus changing only the d value, of the parametrization used in the original paper.

The EDDM method available at MOA offers no parameters. Analyzing its code, it was possible to parameterize four values that were constants: the minimum number of instances before permitting the detection of change (similar to the one of DDM), the minimum number of incorrectly classified instances before raising a warning or drift (e), the drift detection level (d), and the warning level (w). The values used in the 2^k factorial design were: $d \in \{0.8, 0.85\}, w \in \{0.9, 0.95\}, n \in \{15, 30\}$, and $e \in \{15, 30\}$. Again, the results are similar to the ones obtained with the DDM method. The drift detection level was the most important parameter considering all datasets, specially in the real-world ones, with 56.30%. In the abrupt drifts datasets, e was the most important parameter, accounting for 23.74% of the differences in performance. After changing the values of these parameters (again adjusting the value of w accordingly, as d must be lower than w) to find better configurations and comparing them using the F_F statistic, our test results indicated no statistical differences between any of them. Thus, we used the default values as presented in the MOA framework: $n = 30, e = 15, d = 0.9$, and $w = 0.95$.

The ECDD method has four parameters: the average run length that informs the rate of false positive alarms per data point (a), how much weight is given to more recent data compared to older data (λ), the warning threshold (w), and the minimum number of instances before permitting the detection of a drift (n). For the first three parameters, we used two values based on (Ross et al., 2012): $a \in \{100, 1000\}, \lambda \in \{0.1, 0.3\}$, and $w \in \{0.5, 0.8\}$. The last of these used the same values tested in the DDM and EDDM methods: $n \in \{15, 30\}$. The results indicated that in the abrupt datasets, λ was the most important parameter, followed by a. In the gradual and real-world datasets, the opposite occurred. Considering all datasets, a accounted for 34.86% of the differences in performance, followed by λ with 29.96% and the combination of a and λ, with 7.75%. By varying the parameters a and λ, we obtained the following configuration for the ECDD method, which presented the best results using the F_F statistic: $a = 400, \lambda = 0.1, w = 0.5$, and $n = 30$, quite similar to the proposed values in the original paper, as used in the 2^k factorial design.

In the case of the PHT method, as presented by (Gama, Sebastião, & Rodrigues, 2009), two parameters were used: allowed magnitude of change (m) and the drift detection threshold (δ). Two other parameters were added to the method: the warning level (w) and the minimum number of incorrectly classified instances before raising a warning or drift (n). Initial tests were performed using the following values: $\delta \in \{1.5, 2.5\}, m \in \{0.001, 0.01\}, w \in \{0.5, 0.8\}$, and $n \in \{15, 30\}$. The results of the 2^k factorial design indicated that δ was the most important parameter, closely followed by m. Subsequent tests modifying these parameters and computing the F_F statistic presented the best set to use: $n = 15, \delta = 1.5, w = 0.5$, and $m = 0.001$.

In the case of the STEPD method, three parameters described in the original paper were used: the window size (w), and the significance levels for warning (m) and for drift (d). Initial tests to perform the 2^k factorial design were carried out with the following values: $w \in \{20, 100\}, d \in \{0.02, 0.03\}$, and $m \in \{0.05, 0.08\}$ (the d value must be lower than m). Our results indicated that w was the most important parameter, followed by d and the interaction of the two. Subsequent tests were performed modifying these two parameters. The computed F_F statistic indicated no differences in performance between any of them. Comparing each configuration to the others, and selecting the one with the highest accuracy in the greatest number of datasets, the final configuration was: $w = 20, d = 0.03$, and $m = 0.08$.

Unlike the other methods that were built to identify concept drifts based on the results of a base learner's classification (true or false), PL identifies a drift when the difference in performance between its two learners (stable and reactive) is above a parameterized threshold on a window with the last tested instances. Thus, PL is not implemented using `SingleClassifierDrift`, but as a pure classifier. This classifier has two parameters: w, indicating the window size, and θ, the threshold to inform that a drift has occurred. Tests were performed using the following values: $w \in \{6, 12\}$ and $t \in \{0.1, 0.2\}$. w was the most important parameter to set with 46.82% while θ accounted for 28.21% of the differences in performance. This is an expected behavior as θ must be set based on the value of w as they are inversely proportional. Tests in 12 other configurations of PL, including the ones presented in the original paper, indicated differences in performance with the best parameter set being $w = 10$ and $\theta = 0.2$, close to the ones presented in the original paper.

The ADWIN implementation available in the MOA framework does not offer any parametrization. Two values were made available as parameters and had their influence analyzed in the resulting accuracy: maximum global error (δ) and frequency of window reduction (M). Initial tests were performed using the following values: $\delta \in \{0.002, 0.01\}$ and $M \in \{32, 64\}$. δ was the most important parameter in the real-world datasets, with 66.76%. In the artificial datasets, the influence of the parameters was negligible. By varying the values of δ and M, F_F indicated no differences in performance. The values used in these experiments were the default ones presented in MOA: $\delta = 0.02$ and $M = 16$.

Finally, to compute the 2^k factorial design for the DOF method, two parameters, as described in the original paper, were used: the number of standard deviations for detecting drifts (s) and the window size (w). The values used were: $w \in \{50, 100\}$ and $s \in \{2.0, 4.0\}$. The results indicated that s was the most relevant parameter. Subsequent tests were performed and the F_F statistic was computed, resulting in no differences in performance. The configuration with the highest accuracy in most datasets was chosen: $s = 2.0$ and $w = 15$.

Initial conclusions on the parametrization of the methods can be drawn. The drift detection level was the most important parameter in DDM, EDDM, STEPD, and the PHT methods. The n parameter used in DDM, EDDM, ECDD, and PHT was considered unimportant. Parametrization was more important in the real-world datasets; in the artificial datasets, differences in performance were almost always negligible.

It is important to note that ECDD, PHT, STEPD, DOF, and PL were not available in MOA. We therefore provided complete implementations and made them available as extensions of the MOA framework. These, together with ADWIN, as well as the Sine and Mixed datasets, can be downloaded from http://sites.google.com/site/moaextensions/, including the instructions on how to use them.

4. Experimental results

In this section we present the results of the experiments with the concept drift detection methods in the presented datasets regarding accuracy, evaluation time, false alarm and miss detection rates, as well as distance to the drift point.

4.1. Predictive accuracy

The predictive accuracies of the drift detection methods were computed by the average normalized Area Under the Curve (AUC), with 95% confidence intervals using the trapezoid rule on adjacent pairs of accuracies. The results are presented in Table 1, with the best results written in boldface. Fig. 1 presents graphics of the accuracies in the artificial datasets.

Sine(1000) is the Sine dataset in which each context contains 1000 instances, $t = 1$, and $d = 100$. It is similar to the dataset known as SINIRREL1 described by (Gama et al., 2004), and its results are presented in Fig. 1(a). Sine(5000), presented in Fig. 1(b), is similar to Sine(1000), but each context is represented by 5000 instances and $t = 10$. It is based on the SINE200 dataset presented by (Ross et al., 2012), and the graphical curves are almost the same as the ones presented in Fig. 1(a).

It is possible to observe that the performance in Sine(5000) was better than in the Sine(1000) dataset in all drift methods. This is an expected behavior because more instances are used in the training phase of Sine(5000), allowing the base learners to better represent the hidden context. The majority of the methods recovered very rapidly from the drift in both configurations of the Sine dataset, unlike NB, whose performance decreased considerably after the drift. This is also an expected behavior because the dataset has only four attributes, of which only two are relevant, there are only two classes, and the hidden context is represented by a simple function.

The accuracies of the methods in each dataset are very similar, usually differing by less than 1%. This is due to the fact that the function that generates the target attribute is fairly simple and has many instances to train after the drift point.

The first tested Stagger configuration, here known as Stagger(1), had already been used in previous experiments (Bach & Maloof, 2008; Kolter & Maloof, 2007). The three contexts are represented by 40 instances, $t = 1$, and $d = 100$. Fig. 1(c) present the accuracies on Stagger(1). It is worth pointing out that, around time steps 40 and 80, when the concept drifts occur, the accuracies of all the methods decreased, until their base learners changed to adapt to the new context.

DOF had the worst performance in this dataset. This can be explained by two facts: (a) DOF only checks for drifts every w instances (in the tested configuration, $w = 15$), reducing its capacity to detect drifts at any moment; (b) it needs a sequence of degrees of drift to correctly compute its average and standard deviation and identify whether it has changed. As the dataset size is small and it only checks for drifts every 15 instances, DOF did not identify either of the concept drifts that occurred.

ADWIN performed practically the same as the base learner, being the second worst drift detection method, not identifying the first concept drift and identifying the second drift only in 6% of the repetitions at time step 96. This occurred because, in the configuration tested, ADWIN only computes its two internal windows every 16 instances. At the first drift point (time step 40), the window consists of classification results from time steps 33 to 48, 7 from the first context and the remaining from the new context. However, as the first and second contexts are not mutually exclusive, some instances have the same classification results in both contexts. Thus, ADWIN has difficulty in differentiating data from both contexts in the window. The moment of the drift point is important, as well as the window size. For example, increasing the window size to 32 prevents ADWIN from identifying any of the two drifts, because the drift occurs at the beginning of the window, storing too few instances of the old context.

Differences in the curves of the Sine and Stagger configurations with $t = 1$ are mainly due to the number of training instances before drift: 1000×40. In the Sine dataset, the drift occurred after a stable period in a single concept (starting around $T = 200$), while in Stagger the base learner was still converging.

Stagger(20) presents a modified configuration of Stagger(1), with $t = 20$. Now, all the methods diverge from the base learner much earlier. Again, increasing t had a positive influence on the evaluation of the testing instances in all methods, with ADWIN (by 33.72%) and STEPD (by 33.61%) being the ones that benefited most. DOF, which was the most affected method in the Stagger(1) dataset when testing in a single instance, with the lowest accuracy, yields results comparable to those presented by PL, DDM, ECDD, and EDDM in Stagger(20).

Based on the analysis of the results of the drift detection methods in the abrupt concept drift datasets, PL presented the lowest average rank, while DDM presented the highest average accuracies, closely followed by PL. The parametrization chosen for PL in the experiments performed here, with a small window size, yields better results in datasets affected by abrupt concept drifts. On the other hand, DOF presented the highest average rank and the lowest average accuracies.

To analyze how the methods perform in the presence of gradual drifts, two configurations of the Mixed dataset were evaluated with different widths of change: 200, a faster gradual configuration, and 1000, a slower one. In the Mixed(200) dataset, the drift takes 200 instances to completely change the context, between time steps 47 and 53, with $t = d = 40$. In the Mixed(1000) dataset, a configuration similar to the SINE1G dataset presented by (Baena-García et al., 2006) was used: each context is made up of 2000 instances and a slow gradual drift lasts for 1000 instances, between time steps 37 and 62. The results of Fig. 1(e) and (f) show that the performance of all methods declines when the concept starts to drift but recovers slightly before the new context is completely

Table 1
Accuracy average normalized AUC with 95% confidence intervals.

Dataset	NB	PL	ADWIN	DOF	DDM	ECDD	EDDM	PHT	STEPD
Sine(1000)	78.28 ± 0.67	90.90 ± 0.16	90.72 ± 0.16	89.68 ± 0.15	91.05 ± 0.16	**91.22 ± 0.16**	90.73 ± 0.16	88.60 ± 0.18	91.21 ± 0.16
Sine(5000)	79.01 ± 1.00	92.52 ± 0.07	92.52 ± 0.07	91.05 ± 0.05	92.50 ± 0.07	92.51 ± 0.07	92.40 ± 0.07	91.50 ± 0.09	**92.61 ± 0.07**
Stagger(1)	72.78 ± 1.40	80.08 ± 0.47	72.83 ± 1.39	72.78 ± 1.40	**80.28 ± 0.46**	76.82 ± 0.71	76.86 ± 0.70	74.95 ± 0.94	73.19 ± 1.32
Stagger(20)	82.65 ± 2.20	**97.80 ± 0.26**	97.39 ± 0.23	95.57 ± 0.25	97.77 ± 0.26	97.77 ± 0.26	97.51 ± 0.24	94.73 ± 0.34	97.79 ± 0.26
Mixed(200)	76.55 ± 3.35	86.21 ± 1.75	**88.11 ± 1.79**	85.83 ± 1.74	87.93 ± 1.79	87.43 ± 1.78	87.31 ± 1.77	86.72 ± 1.80	88.05 ± 1.79
Mixed(1000)	75.79 ± 3.20	81.30 ± 1.86	83.10 ± 1.93	81.79 ± 1.83	**83.53 ± 1.88**	83.03 ± 1.87	82.73 ± 1.86	83.49 ± 1.86	83.07 ± 1.93
Hyp(0.1)	74.75 ± 1.53	75.94 ± 1.59	76.08 ± 1.59	78.21 ± 1.67	77.21 ± 1.64	**79.02 ± 1.70**	78.06 ± 1.66	76.11 ± 1.59	74.76 ± 1.53
Hyp(0.001)	87.73 ± 1.78	79.32 ± 1.60	87.72 ± 1.78	83.11 ± 1.67	**87.74 ± 1.79**	86.69 ± 1.76	86.83 ± 1.80	87.64 ± 1.79	87.73 ± 1.78
Covertype	66.22	70.34	70.77	68.00	69.52	69.23	69.49	**74.02**	69.50
Electricity	74.65	73.98	76.45	74.65	73.46	73.87	72.74	**76.83**	76.29
PokerHand	58.24	**72.16**	71.05	65.71	70.45	70.86	70.11	69.20	70.88
Weather	68.84	70.86	69.13	68.24	71.20	70.53	**71.43**	70.40	70.23

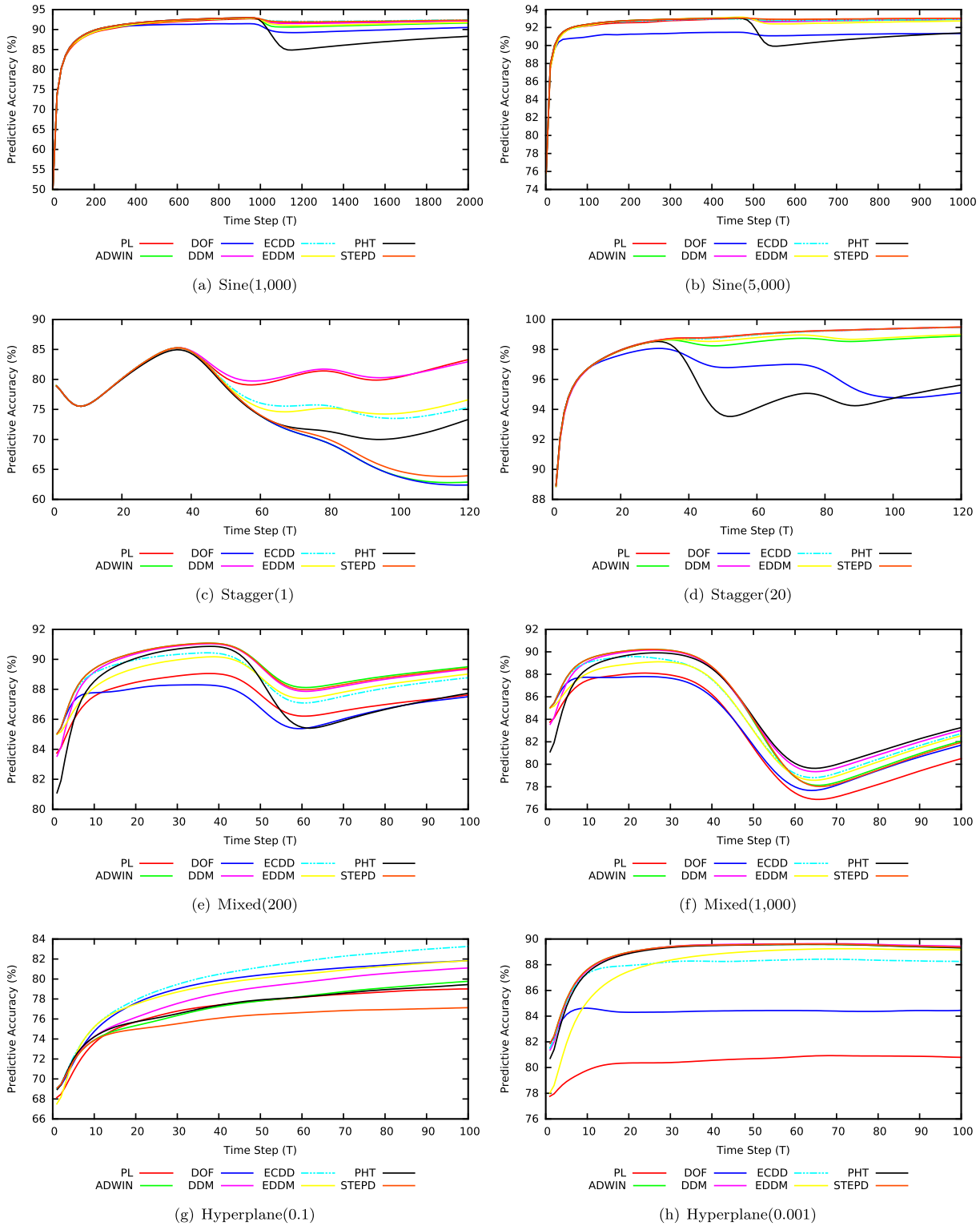


Fig. 1. Accuracies in the artificial datasets.

present. When we compared both configurations of the Mixed dataset, we found that all the methods had lower accuracies when the width of change increased.

Unlike the Mixed dataset, in which there are two stable periods, i.e., before and after the gradual drift, the Hyperplane dataset is in a continuous state of drift. Also, two versions of Hyperplane were tested with different speeds of change, based on the magnitude of the change for every example: 0.001 (slower change) and 0.1 (faster change), during 100,000 instances, and $t = d = 100$.

The Hyp(0.001) is similar to that used by (Bifet et al., 2010). All the methods performed better in the slower drifts version of this dataset. Because the drift is very slow, it almost resembles a single context dataset, as can be seen in Fig. 1(h), where the accuracies are almost constant after convergence of the base learner.

PL was the method with the lowest accuracy in practically all datasets affected by gradual concept drifts (only better than STEPD in the Hyp(0.1) dataset and DOF in Mixed(200)). It was also the method with the highest average rank. On the other hand, DDM was the method with the best performance, presenting the lowest average rank and the highest average accuracies.

For all the real-world datasets we used the evaluation methodology presented by (Elwell & Polikar, 2011), with $t = d = 30$. At each time step, the values previously used for testing are now used for training and the following 30 instances are used for testing. This is just a slight modification of the evaluation methodology used in the abrupt concept drift datasets. The graphical results are presented in Fig. 2.

On analysis of the real-world datasets, PHT was the best method in the Covertype and Electricity datasets. PL was the best method in Poker Hand and second best in Weather. EDDM had the highest accuracy in the Weather dataset, as can be observed in Fig. 2(d). PL was the method with the lowest average rank, while PHT presented the highest average accuracies. DOF was the worst method considering both the average rank and accuracy.

Finally, we used the F_F statistic to compare the base learner and the drift methods in the 12 datasets tested. It revealed that some methods are significantly better than others. To identify which ones are better, we performed the Nemenyi test. The results, presented in Fig. 3, indicated that PL, ADWIN, DDM, ECDD, and STEPD were significantly better than the base learner. All the drift detection methods have similar statistical performances. The Holm procedure was used to compare the methods with NB. It indicated that all methods have similar statistical performances and are better than the base learner, except for DOF, which presented statistical results similar to NB.

4.2. Evaluation time

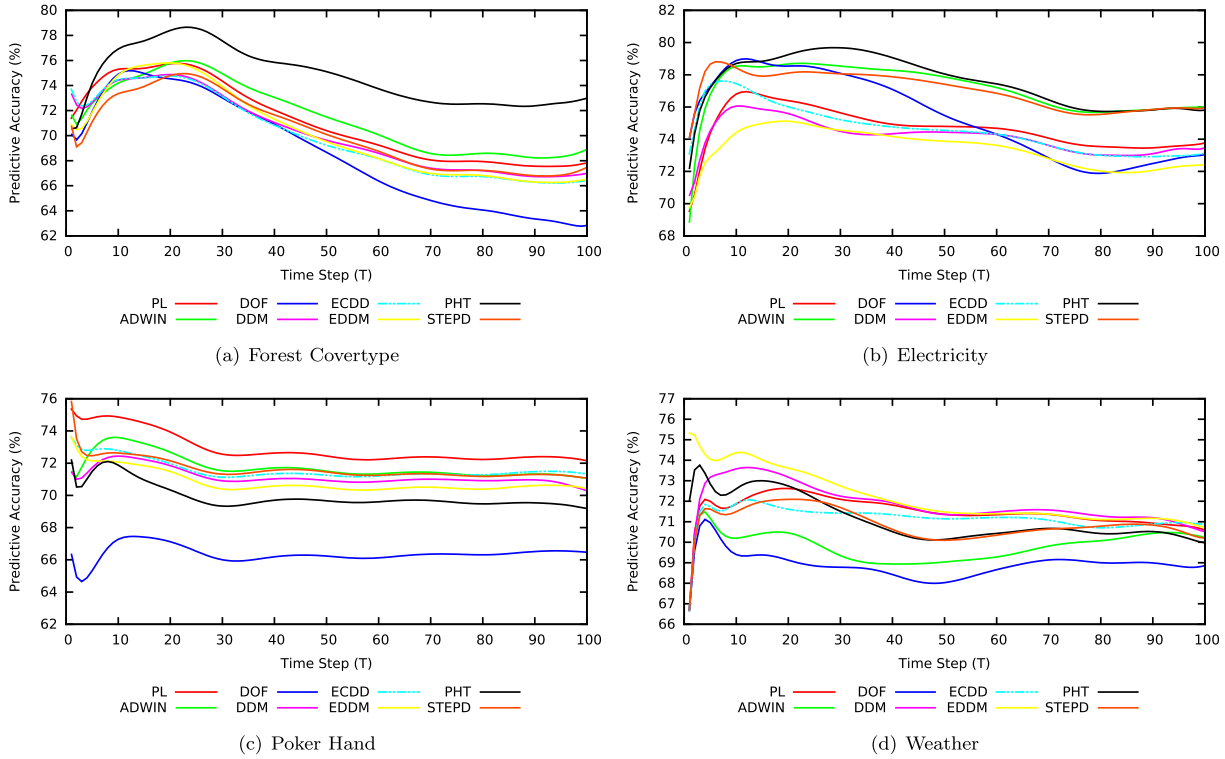
Tests were performed in a computer with a Core i3 330 M processor, 4 GB of main memory, running the Ubuntu 13.10 64 bit operating system. The average evaluation times are presented in Table 2, together with the 95% confidence intervals for the artificial datasets. The methods with the best results in performance are highlighted in boldface.

ADWIN was the fastest method in half of the tested datasets, including all abrupt datasets, and second best in two. PL had the highest average evaluation times, mainly because, for each arriving instance, it has to create a new learner trained on the last w instances. If the retractable version were used, the evaluation times might be reduced, but at the cost of being more specific, as not all online methods have a retractable version. DOF was the second slowest method, especially in the gradual and real-world datasets. The main reason is that DOF has to compute the nearest neighbor of each instance for a current and a previous data chunk, a computation that takes longer than the other methods, which store statistics only on the arriving instances.

The computed F_F statistic also displayed differences in performance. The Nemenyi test identified drift detection methods with different performances: the group composed of ADWIN, DDM, EDDM, and PHT was statistically faster than PL, while ADWIN, DDM, and PHT were statistically faster than DOF. NB was statistically faster than PL, DOF, STEPD, and ECDD. The Holm procedure confirms the results indicated by the Nemenyi test when comparing all methods to NB.

4.3. Drift identification

Table 3 presents the mean distances up to the drift detection (μ DT), the standard deviations (σ DT), the false alarm rates (FA), the miss detection rates (MD), and the Mahalanobis Distances (MaD) (Mahalanobis, 1936) for the abrupt drift datasets. In these datasets, we know exactly where the concept changes occur and can thus use this information to check which method most closely identified the correct drift positions, as well as which had the best false alarm and miss detection rates. The drift points are $T = 1001$, for Sine(1000), $T = 501$ for Sine(5000), and $T = 41$ and $T = 81$ for both Stagger configurations. In the metrics presented at Table 3, lower values (indicated in bold) signify better results.



(a) Forest Covertype (b) Electricity

(c) Poker Hand (d) Weather

Fig. 2. Accuracies in the real-world datasets.

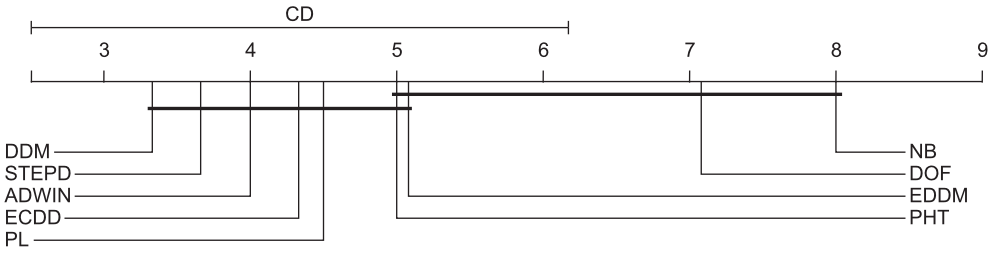


Fig. 3. Comparison of all classifiers against each other with the Nemenyi test. Groups of classifiers that are not significantly different (at $p = 0.05$) are connected.

Table 2
Evaluation time with 95% confidence intervals.

Dataset	NB	PL	ADWIN	DOF	DDM	ECDD	EDDM	PHT	STEPD
Sine(1000)	1.60 ± 0.17	2.65 ± 0.19	**2.14 ± 0.18**	2.46 ± 0.18	2.42 ± 0.20	2.72 ± 0.22	2.51 ± 0.20	2.40 ± 0.19	2.54 ± 0.19
Sine(5000)	0.53 ± 0.03	1.09 ± 0.04	**0.82 ± 0.03**	0.96 ± 0.03	0.97 ± 0.04	1.06 ± 0.04	1.00 ± 0.04	0.95 ± 0.04	1.05 ± 0.04
Stagger(1)	0.04 ± 0.00	0.09 ± 0.00	**0.06 ± 0.00**	0.09 ± 0.00	0.07 ± 0.00	0.08 ± 0.00	0.08 ± 0.00	0.08 ± 0.00	0.08 ± 0.00
Stagger(20)	0.04 ± 0.00	0.09 ± 0.00	**0.07 ± 0.00**	0.09 ± 0.00	0.08 ± 0.00	0.09 ± 0.00	0.08 ± 0.00	0.08 ± 0.00	0.09 ± 0.00
Mixed(200)	0.03 ± 0.00	0.10 ± 0.00	**0.06 ± 0.00**	0.10 ± 0.00	0.07 ± 0.00	0.09 ± 0.00	0.08 ± 0.00	0.07 ± 0.00	0.08 ± 0.00
Mixed(1000)	0.04 ± 0.00	0.11 ± 0.00	**0.07 ± 0.00**	0.10 ± 0.00	0.07 ± 0.00	0.09 ± 0.00	0.08 ± 0.00	0.07 ± 0.00	0.08 ± 0.00
Hyp(0.1)	0.49 ± 0.12	1.87 ± 0.45	0.94 ± 0.23	1.51 ± 0.36	0.94 ± 0.23	1.09 ± 0.26	1.00 ± 0.24	**0.90 ± 0.22**	1.06 ± 0.26
Hyp(0.001)	0.49 ± 0.12	1.87 ± 0.45	0.93 ± 0.22	1.51 ± 0.36	0.95 ± 0.23	1.09 ± 0.26	1.00 ± 0.24	**0.92 ± 0.22**	1.06 ± 0.26
Covertype	19.93	38.79	19.54	37.45	**17.05**	17.95	17.95	20.58	18.63
Electricity	0.34	0.89	0.49	0.66	0.47	0.54	0.47	**0.46**	0.50
PokerHand	8.71	14.27	7.21	11.19	6.66	7.01	**5.95**	6.71	6.89
Weather	0.37	0.48	0.46	0.79	0.24	0.27	0.25	**0.23**	0.26

Table 3
Statistics in the abrupt datasets.

Methods	μ DT	σ DT	FA	MD	MaD
	Sine(1000)				
PL	7.26	3.49	0.006	0.839	0.515
ADWIN	22.68	2.26	0.005	0.915	1.616
DOF	238.30	408.73	0.006	0.889	0.944
DDM	12.04	3.46	0.005	0.900	0.447
ECDD	22.36	141.09	**0.004**	**0.820**	0.276
EDDM	12.58	7.41	0.006	0.875	0.577
PHT	104.36	11.12	0.005	0.917	0.549
STEPD	**6.58**	**1.49**	**0.004**	0.822	**0.115**
	Sine(5000)				
PL	0.08	0.27	0.014	0.138	1.202
ADWIN	1.80	0.61	0.001	0.643	0.948
DOF	48.38	116.92	0.018	0.583	1.691
DDM	1.82	0.98	0.002	0.632	0.643
ECDD	11.64	71.18	0.007	0.074	0.776
EDDM	2.70	3.26	0.014	0.603	1.260
PHT	19.54	2.94	0.002	0.667	0.856
STEPD	**0.00**	**0.00**	**0.000**	**0.000**	**0.080**
	Stagger(1)				
PL	**23.52**	9.22	0.156	0.875	0.377
ADWIN	116.64	12.17	0.153	0.917	0.168
DOF	120.00	**0.00**	0.153	0.917	**0.141**
DDM	24.44	17.78	**0.149**	**0.861**	0.243
ECDD	60.44	22.59	0.156	0.902	0.292
EDDM	56.74	10.34	0.161	0.911	0.393
PHT	64.12	28.24	0.159	0.902	0.344
STEPD	113.14	18.44	0.154	0.917	0.181
	Stagger(20)				
PL	0.02	0.14	0.001	0.019	0.058
ADWIN	0.76	0.66	0.006	0.432	0.499
DOF	22.76	26.13	0.033	0.640	0.952
DDM	**0.00**	**0.00**	0.001	0.000	0.035
ECDD	0.80	5.66	0.001	0.020	0.035
EDDM	0.74	0.53	0.007	0.425	0.531
PHT	8.76	0.94	0.017	0.667	1.097
STEPD	**0.00**	**0.00**	**0.000**	**0.000**	**0.000**

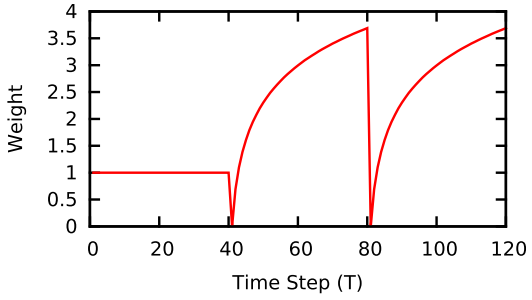


Fig. 4. Weight function for the Stagger(20) dataset.

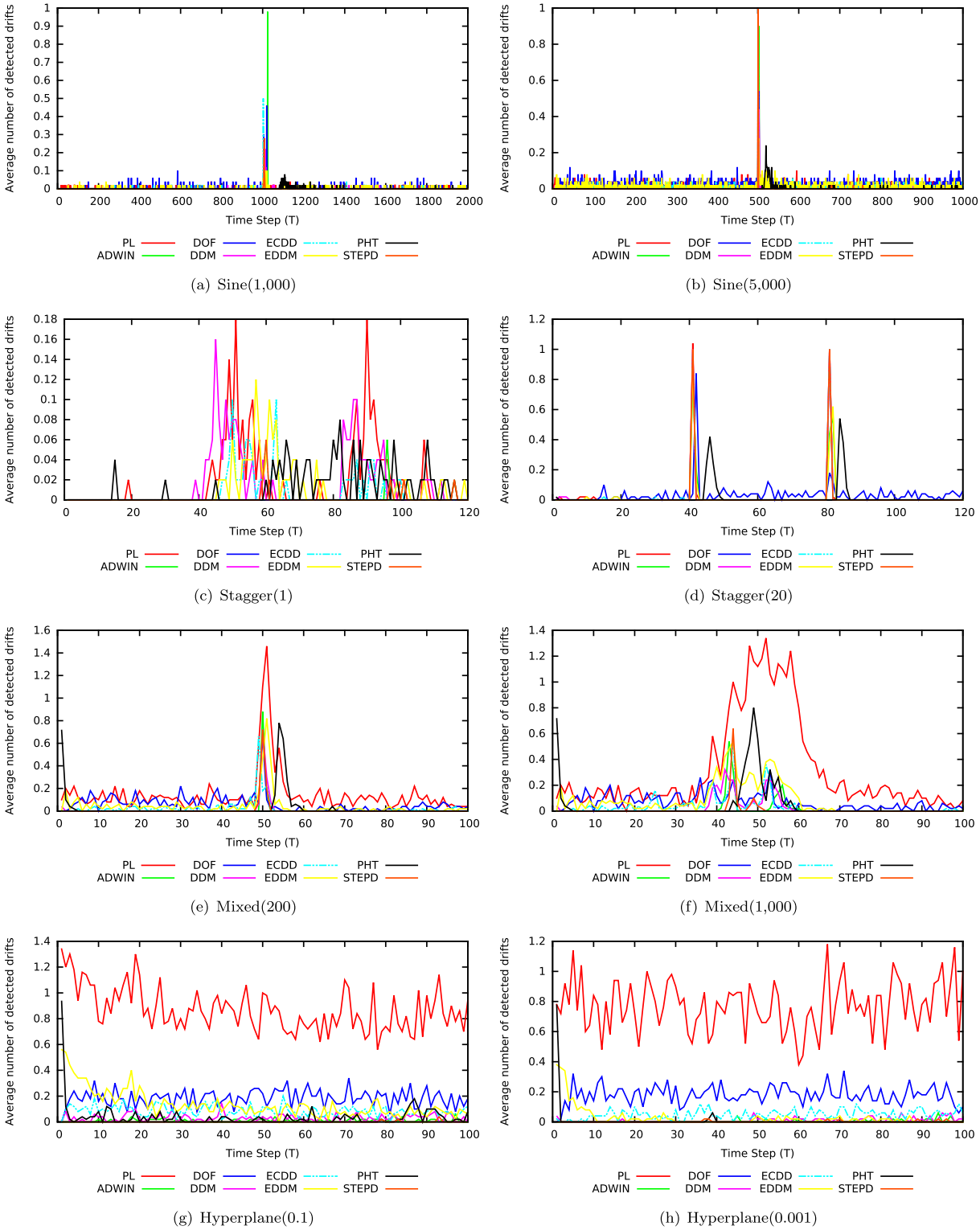
Considering the first metric, lower distances signify that the methods identified the drift closer to the actual change point. STEPD presented good results, with the lowest mean distance in Sine(1000), Sine(5000), and Stagger(20). In the last two datasets, the distance to the drift point was zero, indicating that STEPD identified the drift point in the correct position in all 50 repetitions. On the other hand, DOF was the method that took longest to identify that a drift had occurred in all datasets.

The standard deviation allows us to analyze the stability of the methods in relation to the detected drift position. It informs us how confident we can be that a drift detection method execution identifies the context change in the distance obtained by the previous metric. STEPD performed best in the Sine(1000), Sine(5000), and Stagger(20) datasets, for the reasons described previously: all the repetitions identified the drift in the correct position. DOF also achieved a zero standard deviation in Stagger(1), while DDM had this same result in the Stagger(20) dataset. On the other hand, in the other datasets, DOF was the least unstable method.

The false alarm rate indicates the rate of identified drifts when none occurred. The values are low, indicating that the methods

usually identify drifts correctly. Higher rates in the Stagger(1) dataset can be explained by the occurrence of two concept drifts in a small number of training instances (120). STEPD, DDM, and ADWIN presented the lowest false alarm rates, while EDDM, DOF, and PL presented the highest ones.

Considering the miss detection rate, in the Sine(1000) and Stagger(1) datasets, the methods train on a single instance before performing the testing phase. Thus, it is quite difficult for any method to detect a drift at the exact moment when it occurs based on only one instance. Methods usually consider it to be noise. Several instances are needed to update the internal base learner until it realizes that the context has changed. We can confirm this statement by analyzing the mean distance taken to identify a concept drift. In both Sine(1000) and Stagger(1), the mean distances are much higher than in the configurations in which the methods are trained on more instances before testing.



(a) Sine(1,000) (b) Sine(5,000)

(c) Stagger(1) (d) Stagger(20)

(e) Mixed(200) (f) Mixed(1,000)

(g) Hyperplane(0.1) (h) Hyperplane(0.001)

Fig. 5. Average number of detected concept drifts in the artificial datasets.

Table 4
Summary of the experiments.

Methods	Metrics				
	Abrupt Datasets Accuracy	Gradual Datasets Accuracy	Real-World Datasets Accuracy	Overall Accuracy	Evaluation Time
PL	⇑	⇓	⇑	⇑	⇓
ADWIN	⇓	⇑	⇑	⇑	⇑
DOF	⇓	⇓	⇓	⇓	⇓
DDM	⇑	⇑	⇓	⇑	⇑
ECDD	⇑	⇓	⇓	⇑	⇓
EDDM	⇓	⇓	⇓	⇓	⇑
PHT	⇓	⇓	⇑	⇓	⇑
STEPD	⇑	⇓	⇑	⇑	⇓
	μ DT	σ DT	FA	MD	MaD
PL	⇑	⇑	⇓	⇑	⇓
ADWIN	⇓	⇑	⇑	⇓	⇓
DOF	⇓	⇓	⇓	⇓	⇓
DDM	⇑	⇑	⇑	⇑	⇑
ECDD	⇓	⇓	⇓	⇑	⇑
EDDM	⇑	⇓	⇓	⇓	⇓
PHT	⇓	⇓	⇓	⇓	⇓
STEPD	⇑	⇑	⇑	⇑	⇑

Training on single instances raises a question: how should the miss detection rates be computed? To expect that the methods could correctly identify drifts based on one instance training of a new context is implausible: the miss detection rate can easily reach 100%. Therefore, to avoid this problem, in Sine(1000) and Stagger(1), if the method identifies a drift in the first 10 instances following the correct drift point, *no* miss detection is computed. In the other abrupt datasets, as they train on at least 10 instances, the miss detection rate *is* computed if the drift identification does not occur at the exact time step where it took place.

Concerning the miss detection rates, STEPD, PL and ECDD presented the best results, while DOF and PHT presented the worst rates. The methods presenting high miss detection rates are a result of late detection, as previously described. They usually identify the drift far from the drift point, increasing the miss detection rates.

In addition to these metrics, we also computed the distance of the detected concept drifts at each time step of each method to the benchmark of each dataset using the Mahalanobis Distance (Mahalanobis, 1936). This metric allows us to verify which of the compared methods most closely resembles the correct number of drifts at each time step, i.e., identifying the drift point in the correct position and not identifying drifts at wrong positions. In a sense, it can be viewed as a summary of the metrics presented above.

To use this metric, we must inform the covariance matrix to be used in the computation. In this matrix, we inform in the main diagonal the weight that each value must have in relation to its time step. The weight function (w) used is described at Eq. (1), and a graph showing its values is presented in Fig. 4 for the Stagger(20) dataset. Variable t is the time step and d is the time step where the drift occurs. Higher weight values result in longer distances. Before the drift point, all false alarms are treated similarly with weight 1. In the drift points, the weight is not zeroed to prevent the covariance matrix zeroing the distance in cases where a method identifies no drift. Thus, a small weight is used. After the drift points, the weights increase using a logarithm function. The choice of such a function is related to the nature of the drift identification. We wanted a function to model a behavior in which the differences in the identification of the drift points reduces the weight increase as it moves away from the correct time step. For example, in the Stagger datasets, if DDM identifies a drift at time step 42 and EDDM at time step 43, the difference in the weight is 0.405, while at time steps 65 and 66, the difference is 0.039. STEPD had the best results in the Sine(1000), Sine(5000), and Stagger(20) datasets, and was third best in Stagger(1). In Stagger(20), STEPD performed a perfect match, identifying the drifts in their correct positions (in all repetitions), without false alarms. In Stagger(1), DOF did not identify any drifts, yielding an average distance to the drift point equal to the dataset size, zeroing the standard deviation, and resulting in the lowest Mahalanobis distance, as it only differs from the benchmark at the two drift points.

$$w(t) = \begin{cases} 1 & \text{if} \quad t < d \\ 0.01 & \text{if} \quad t = d \\ \ln(t - d + 1) & \text{if} \quad t > d \end{cases} \tag{1}$$

Fig. 5 presents the average number of detected concept drifts per time step for the artificial datasets in the 50 repetitions of the experiments. For example, observing Fig. 5(d), we can see that, right after time steps 40 and 80, there is an increase in the number of detected concept drifts, indicating that the methods generally identify the drift in the right position.

In the Mixed(1000) dataset, presented in Fig. 5(f), the methods usually identified the drifts near the correct positions, but we can observe that PL detected far more drifts in the stable contexts than the other methods. A similar behavior occurs in the other configurations of the gradual drift datasets. This behavior indicates that PL did not handle the gradual drift datasets well, which is confirmed by its predictive accuracies.

Table 4 presents a summary of the results obtained in the experiments, comparing all concept drift detection methods in all the metrics presented. The average rank of the methods in each metric was computed, as well as the standard deviation. The methods lying between the result of the best method and a standard deviation were considered the best ones; the others were considered the worst. The best results are shown with the symbol ⇑ and the worst ones with the symbol ⇓.

5. Conclusion

In this paper, several concept drift detectors were compared and analyzed with respect to how they behaved in the presence of abrupt and gradual concept drifts, with different speeds of change, in artificial and real-world datasets. In a sense, this paper may be regarded as a literature review of existing drift detection methods.

Five different drift detection methods (PHT, ECDD, STEPD, DOF, and Paired Learners) and two artificial datasets (Sine and Mixed) were implemented in the MOA framework and made freely available to the public, allowing further comparison by other researchers.

Initially, a 2^k factorial design was performed at each drift method to identify the parameters that most influence the predictive accuracy. To the best of our knowledge, this is the first time that this technique has been used to indicate the relevance of each parameter and of its interactions. This can considerably reduce the space search for the best parameter combination. The results indicated that the parameters used to set the warning level and the minimum number of stored instances before permitting the detection of a change usually have little influence on the accuracies of the methods.

After identifying the best parameters for each drift method, several other values for these parameters were used to select the best configuration on the tested datasets. This selection was performed by computing the F_F statistic, based on the Friedman non-parametric statistical test, and by using two post hoc tests, namely Nemenyi and Holm. The best parametrization of each drift method was used to compare them against each other.

The results of the predictive accuracies indicated that:

- PL was the method that presented the lowest average rank in datasets with abrupt concept drifts, while DDM presented the highest average accuracies. DOF was the worst in both metrics;
- DDM was the best method in datasets affected by gradual concept drifts, considering both the average rank and the accuracies, while PL was the worst in both metrics;
- In the real-world datasets, PL presented the lowest average rank and PHT the highest average accuracies, while DOF was the worst in both metrics;
- Taking all tested datasets into account, DDM was the best method considering both the average rank and the accuracies.

Regarding evaluation time, even though ADWIN was the fastest drift detection method in 6 out of 12 datasets tested, being even faster than the base learner in some real-world ones, PHT and DDM presented the lowest average ranks. On the other hand, PL was the slowest in 11 out of 12 datasets and DOF was the second slowest.

In the abrupt drift datasets, PL was the best method for identifying the drift position close to the actual drift points. STEPD presented the lowest standard deviation, identifying drifts practically always at the same time step, while DOF presented a highly unstable behavior, almost always indicating a context change at each time step. The methods that returned the lowest number of false alarms were STEPD, DDM, and ADWIN, whereas ECDD, PL, and STEPD presented the lowest miss detection rates.

Finally, we also computed the Mahalanobis distance of the methods and the best possible drift identification pattern. We proposed a weighting function that yields zero if the method matches the benchmark and higher values as the drift identification deviates from the benchmark. STEPD yielded the lowest distance to the expected behavior of the drift methods. Again, we believe that this is the first time that this technique has been used in this context.

As can be seen, each drift detection method performs best in datasets with different characteristics. Based on the work described in this paper, a number of research topics could be investigated in future work.

Firstly, an ensemble of drift detection methods similar to the ensemble classifier approach could be tested. It is likely that such an ensemble would improve the results of the single methods, probably using the concept of diversity (Minku, White, & Yao, 2010), by testing the data stream using different methods. The best criteria for the warning and drift levels would also be subject of investigation and such levels could be indicated when one or some or all the methods so indicate.

Another possible direction would be to try to automatically identify the type of drift (abrupt or gradual), based on the results of the drift detection methods, as well as the most likely position of the drift based on a confidence interval.

Based on the experiments, it might also be possible to identify the characteristics of the dataset based on the results of the methods. For example, using STEPD and ECDD to process a dataset and then analyze their results, identifying the method that yields the best results may be a good approximation of deciding whether the dataset is affected by abrupt or gradual concept drifts.

Since DOF presented the worst results when compared to the other methods, we could try to improve its predictive accuracy by testing other distance measures in addition to the Euclidean distance. It is possible that, besides its accuracy, this could also make it run faster. Another possible improvement would be to define a threshold to indicate a drift, rather than computing its average and standard deviation. Again, this could also make DOF run faster because it would execute fewer calculations.

To conclude, it is important to point out that the experiments reported in this paper represent a first step towards identifying which drift detection methods are best suited to solving different kinds of problems. More methods could be tested, for example, the Fixed Cumulative Windows Model (FCWM) (Sebastião, Gama, Rodrigues, & Bernardes, 2010), as well as other base learners.

References

Bach, S. H., & Maloof, M. A. (2008). Paired learners for concept drift. In *IEEE international conference on data mining, ICDM'08* (pp. 23–32). Los Alamitos, CA, USA: IEEE Computer Society. <http://dx.doi.org/10.1109/ICDM.2008.119>.

Baena-García, M., Del Campo-Ávila, J., Fidalgo, R., Bifet, A., Gavaldà, R., & Morales-Bueno, R. (2006). Early drift detection method. In *International workshop on knowledge discovery from data streams, IWKDDS'06* (pp. 77–86). http://eprints.pascal-network.org/archive/00002509/.

Bifet, A., & Gavaldà, R. (2007). Learning from time-changing data with adaptive windowing. In *Proceedings of the seventh SIAM international conference on data mining, SDM'07* (pp. 443–448). Lake Buena Vista, Florida, USA: SIAM.

Bifet, A., Holmes, G., Kirkby, R., & Pfahringer, B. (2010). MOA: Massive online analysis. *Journal of Machine Learning Research, 11*, 1601–1604. <http://portal.acm.org/citation.cfm?id=1859890.1859903>.

Bifet, A., Holmes, G., Pfahringer, B., & Frank, E. (2010). Fast perceptron decision tree learning from evolving data streams. In M. Zaki, J. Yu, B. Ravindran, & V. Pudi (Eds.), *Advances in knowledge discovery and data mining. Lecture Notes in computer science* (Vol. 6119, pp. 299–310). Berlin/ Heidelberg: Springer. <http://dx.doi.org/10.1007/978-3-642-13672-6_30>.

Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., & Gavaldà, R. (2009). New ensemble methods for evolving data streams. In *Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining, KDD'09* (pp. 139–148). New York, NY, USA: ACM. <http://dx.doi.org/10.1145/1557019.1557041>.

Delany, S. J., Cunningham, P., Tsymbal, A., & Coyle, L. (2005). A case-based technique for tracking concept drift in spam filtering. *Knowledge-Based Systems, 18*(4-5), 187–195. <http://dx.doi.org/10.1016/j.knosys.2004.10.002> [al-2004, Cambridge, England, 13th–15th December 2004].

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research, 7*, 1–30. <http://dl.acm.org/citation.cfm?id=1248547.1248548>.

Elwell, R., & Polikar, R. (2011). Incremental learning of concept drift in nonstationary environments. *IEEE Transactions on Neural Networks, 22*(10), 1517–1531. <http://dx.doi.org/10.1109/TNN.2011.2160459>.

Fdez-Riverola, F., Iglesias, E. L., Díaz, F., Méndez, J. R., & Corchado, J. M. (2007). Applying lazy learning algorithms to tackle concept drift in spam filtering. *Expert Systems with Applications, 33*(1), 36–48. <http://dx.doi.org/10.1016/j.eswa.2006.04.011>.

Friedman, M. (1940). A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics, 11*(1), 86–92. <http://www.jstor.org/stable/2235971>.

Gama, J. (2010). *Knowledge discovery from data streams*. Chapman & Hall/CRC.

Gama, J., Medas, P., Castillo, G., & Rodrigues, P. (2004). Learning with drift detection. In A. Bazzan & S. Labidi (Eds.), *Advances in Artificial Intelligence – SBIA 2004. Lecture Notes in Computer Science* (Vol. 3171, pp. 66–112). Berlin/ Heidelberg: Springer. <http://dx.doi.org/10.1007/978-3-540-28645-5_29>.

Gama, J., Sebastião, R., & Rodrigues, P. P. (2009). Issues in evaluation of stream learning algorithms. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD'09* (pp. 329–338). New York, NY, USA: ACM. <http://doi.acm.org/10.1145/1557019.1557060>.

Gonçalves, P. M., Jr., & Barros, R. S. M. (2013). RCD: A recurring concept drift framework. *Pattern Recognition Letters, 34*(9), 1018–1025. <http://dx.doi.org/10.1016/j.patrec.2013.02.005>.

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics, 6*(2), 65–70. <http://www.jstor.org/stable/4615733>.

Hulten, G., Spencer, L., & Domingos, P. (2001). Mining time-changing data streams. In *Proceedings of the seventh ACM SIGKDD international conference on knowledge discovery and data mining, KDD'01* (pp. 97–106). New York, NY, USA: ACM. <http://dx.doi.org/10.1145/502512.502529>.

Iman, R. L., & Davenport, J. M. (1980). Approximations of the critical region of the fbietkan statistic. *Communications in Statistics – Theory and Methods, 9*(6), 571–595. <http://dx.doi.org/10.1080/03610928008827904>.

Kolter, J. Z., & Maloof, M. A. (2007). Dynamic weighted majority: An ensemble method for drifting concepts. *Journal of Machine Learning Research, 8*, 2755–2790. <http://dl.acm.org/citation.cfm?id=1314498.1390333>.

Mahalanobis, P. C. (1936). On the generalized distance in statistics. *Proceedings of the National Institute of Sciences of India, 2*(1), 49–55. <http://www.new.dli.ernet.in/rawdataupload/upload/insa/INSA_1/20006193_49.pdf>.

Minku, L. L., White, A. P., & Yao, X. (2010). The impact of diversity on online ensemble learning in the presence of concept drift. *IEEE Transactions on Knowledge and Data Engineering, 22*(5), 730–742. <http://dx.doi.org/10.1109/TKDE.2009.156>.

Minku, L. L., & Yao, X. (2012). DDD: A new ensemble approach for dealing with concept drift. *IEEE Transactions on Knowledge and Data Engineering, 24*(4), 619–633. <http://dx.doi.org/10.1109/TKDE.2011.58>.

Nemenyi, P. B. (1963). *Distribution-free multiple comparisons* (Ph.D. thesis). Princeton University.

Nishida, K., & Yamauchi, K. (2007). Detecting concept drift using statistical testing. In V. Corruble, M. Takeda, & E. Suzuki (Eds.). *Proceedings of the 10th international conference on discovery science, DS'07* (Vol. 4755, pp. 264–269). Berlin, Heidelberg: Springer-Verlag. <http://portal.acm.org/citation.cfm?id=1778942.1778972>.

Page, E. S. (1954). Continuous inspection schemes. *Biometrika, 41*(1/2), 100–115. <http://www.jstor.org/stable/2333009>.

Roberts, S. W. (1959). Control chart tests based on geometric moving averages. *Technometrics, 1*(3), 239–250. <http://www.jstor.org/stable/1266443>.

Ross, G. J., Adams, N. M., Tasoulis, D. K., & Hand, D. J. (2012). Exponentially weighted moving average charts for detecting concept drift. *Pattern Recognition Letters, 33*(2), 191–198. <http://dx.doi.org/10.1016/j.patrec.2011.08.019>.

Schlimmer, J. C., & Granger, R. H. (1986). Incremental learning from noisy data. *Machine Learning, 1*(3), 317–354. <http://dx.doi.org/10.1023/A:1022810614389>.

Sebastião, R., Gama, J., Rodrigues, P., & Bernardes, J. (2010). Monitoring incremental histogram distribution for change detection in data streams. In M. Gaber, R. Vatsavai, O. Omitaomu, J. Gama, N. Chawla, & A. Ganguly (Eds.), *Knowledge discovery from sensor data. Lecture notes in computer science* (Vol. 5840, pp. 25–42). Berlin/ Heidelberg: Springer. <http://dx.doi.org/10.1007/978-3-642-12519-5_2>.

Sobhani, P., & Beigy, H. (2011). New drift detection method for data streams. In A. Bouchachia (Ed.), *Adaptive and intelligent systems. Lecture notes in computer science* (Vol. 6943, pp. 88–97). Berlin Heidelberg: Springer. <http://dx.doi.org/10.1007/978-3-642-23857-4_12>.

Tsai, C.-J., Lee, C.-I., & Yang, W.-P. (2009). Mining decision rules on data streams in the presence of concept drifts. *Expert Systems with Applications, 36*(2, Part 1), 1164–1178. <http://dx.doi.org/10.1016/j.eswa.2007.11.034>.

Tsymbal, A., Pechenizkiy, M., Cunningham, P., & Puuronen, S. (2008). Dynamic integration of classifiers for handling concept drift. *Information Fusion, 9*(1), 56–68. <http://dx.doi.org/10.1016/j.inffus.2006.11.002> [special Issue on Applications of Ensemble Methods].

Yeh, A. B., Mcgrath, R. N., Sembower, M. A., & Shen, Q. (2008). Ewma control charts for monitoring high-yield processes based on non-transformed observations. *International Journal of Production Research, 46*(20), 5679–5699. <http://dx.doi.org/10.1080/00207540601182252>.