Neurocomputing 000 (2017) 1–10

Contents lists available at ScienceDirect

Neurocomputing

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

Wilcoxon Rank Sum Test Drift Detector

Roberto Souto Maior de Barros*, Juan Isidro González Hidalgo, Danilo Rafael de Lima Cabral

Centro de Informática, Universidade Federal de Pernambuco, Cidade Universitária, Recife 50740-560, PE, Brazil

ARTICLE INFO

Article history:
Received 31 January 2017
Revised 6 October 2017
Accepted 29 October 2017
Available online xxx

Communicated by Prof. Zidong Wang

Keywords:
Concept drift
Statistical tests
Drift detection
Data stream
Online learning

ABSTRACT

Online learning regards extracting information from large quantities of data (streams) usually affected by changes in the distribution (concept drift). Drift detectors are software that estimate the positions of these changes to substitute the base learner and ultimately improve accuracy. Statistical Test of Equal Proportions (STEPD) is a simple, well-known, efficient detector which uses a hypothesis test between two proportions to signal the concept drifts. However, despite identifying the existing drifts close to their correct positions, STEPD tends to identify many false positives. This article examines the application of the Wilcoxon rank sum statistical test for concept drift detection, proposing WSTD. Experiments run in the MOA framework using four artificial dataset generators, with abrupt and gradual drift versions of three sizes, as well as seven real-world datasets, suggest WSTD improves the detections of STEPD and other methods as well as their accuracies in many scenarios.

1. Introduction

In data stream environments, large amounts of data (possibly infinite) flow rapidly and continuously. Thus, learning from these streams requires online methods and is often under restrictions regarding memory and run-time usage. Also, each data instance should be read only once. Finally, concept drifts [1], usually seen as changes in data distribution, may happen.

A very common classification of concept drift is based on the speed of change. Sudden and/or rapid changes between concepts are called *abrupt* and transitions from one concept to another over a larger number of instances are called *gradual* [2–5].

Examples of real-world applications with concept drift include the detection of changes in weather or water temperature, monitoring data from sensors [6], and filtering spam in e-mail messages [7], among others [8].

Different directions have been investigated to learn from data streams containing concept drift. One that is very common refers to concept drift detectors [2], lightweight software that focus on identifying changes in data distribution by monitoring the prediction results of a base classifier.

Several ensembles using a base learner have also been proposed to deal with concept drift, sometimes with different strategies and/or weighting functions to compute the resulting classification, including Dynamic Weighted Majority (DWM) [9], Diversity for Dealing with Drifts (DDD) [10], Adaptable Diversity-based On line Boosting (ADOB) [11], and Boosting-like Online Learning Ensemble (BOLE) [12]. Some methods reuse previous classifiers on recurring concepts, e.g. Recurring Concept Drifts (RCD) [5]. Additionally, we point out some of these ensembles also rely on an auxiliary drift detection method [5,10–13].

Ensembles of concept drift detection methods sharing the same base classifier is another approach which was comparatively less explored [14,15].

A number of concept drift detection methods has been proposed over the years and the most well-known are Drift Detection Method (DDM) [3], Early Drift Detection Method (EDDM) [16], Adaptive Windowing (ADWIN) [17], Statistical Test of Equal Proportions (STEPD) [4], Paired Learners (PL) [18], and EWMA for Concept Drift Detection (ECDD) [19]. From these, DDM and STEPD are among the most simple algorithms and, in spite of this, they present good all-round performance [2].

Other more recent concept drift detectors have also been proposed, including Sequential Drift (SeqDrift) [20], SEED Drift Detector (SEED) [21], Drift Detection Methods based on Hoeffdings Bounds (HDDM) [22], and Fast Hoeffding Drift Detection Method (FHDDM) [23].

Inspired on STEPD, this work proposes Wilcoxon Rank Sum Test Drift Detector (WSTD). It changes the statistical test used to signal warnings and drifts and it also limits the size of the older window of STEPD. We also provide an efficient implementation of Wilcoxon's rank sum test [24] to calculate its ranks and p-value directly, without an explicit sort algorithm.

* Corresponding author.
E-mail address: roberto@cin.ufpe.br (R.S.M.d. Barros).

https://doi.org/10.1016/j.neucom.2017.10.051
0925-2312/

WSTD was tested against ADWIN, DDM, ECDD, SEED, SeqDrift2, and STEPD, using the Massive Online Analysis (MOA) framework [25], two different base classifiers, and a reasonably large number of scenarios, with both artificial and real-world datasets. In addition, statistical evaluations and drift identifications analysis of the results have also been performed.

The rest of the document is organized as follows: Section 2 briefly surveys drift detection methods, with special attention given to STEPD; Section 3 describes the Wilcoxon rank sum test and its implementation; Section 4 details WSTD and its abstract pseudo-code; Section 5 shows the experiments configuration, also including brief descriptions of the datasets used in the tests; Section 6 presents the results obtained, evaluating and statistically comparing accuracies and analysing the drift identifications; and, finally, Section 7 gives our conclusions and proposes future work.

2. Related work – drift detection methods

It is fairly common to use a concept drift detection method together with a base classifier to learn from data streams. In general, the drift detector analyses the predictions of the base learner and adopts some decision model to detect changes in the data distribution. Methods that follow this approach include DDM [3], EDDM [16], and STEPD [4].

Different strategies and/or statistics are used to monitor the performance of the base learner and decide when concept drifts have occurred. A lower confidence level to indicate warnings is also common and these warnings mean that concept drifts may have happened. At such points, an alternative classifier is created to be trained in parallel. If a drift is confirmed the new learner replaces the original one and when the warning is found to be a false alarm the new classifier is discarded.

Considering the data stream as a sequence of pairs $(\vec{x}_i, y_i)$, where $\vec{x}_i$ is a set of attributes and y_i is its corresponding class, for each example, the base learner makes a prediction $(\hat{y}_i)$ which is then compared to the actual class label (y_i).

DDM detects concept drifts in streams by analyzing the error rate and its standard deviation. For each position i, DDM defines the error rate p_i as the probability of making an incorrect prediction and its standard deviation as $s_i = \sqrt{p_i \times (1 - p_i) \,/\, i}$. The authors of DDM argue the error rate p_i should decrease as the number of examples i increases, as long as the distribution of the examples remains stationary. If the error rate increases, DDM assumes there was a change in the data distribution and the current base learner is outdated.

EDDM is similar but it monitors the distance between two consecutive errors, rather than the error rate. In this case, the distance between errors tends to increase and drifts are detected when it decreases.

Both methods use parametrized threshold values for the detection of the warning and drift levels and their default values represent 95% and 99% confidence intervals, respectively.

The parameters of DDM are thresholds for warnings and drifts, α_w and α_d, respectively, and the minimum number of instances n before the detection of concept drifts is permitted. Its chosen default values for these parameters are 2.0, 3.0, and 30, respectively.

In EDDM, the parameters and respective default values are the thresholds, $w = 0.95$ and $d = 0.9$, and the minimum number of errors before drift detections are permitted, $e = 30$.

ADWIN [17] uses a sliding window (W) of performance values with a variable size. When drifts occur the size of W is reduced and the longer the concept the larger the size of W. Two dynamically adjusted sub-windows are stored, representing older and recent data. Drifts are detected when the difference on the averages between these sub-windows is higher than a given threshold. The parameters of ADWIN are a confidence level to reduce the window size – $\delta \in (0, 1)$ – and the minimum frequency of instances (f) needed for the window size to be reduced. The default values of ADWIN in its implementation in the MOA framework are $\delta = 0.002$ and $f = 32$.

ECDD [19] adapted Weighted Moving Average (EWMA) [26] to be used in data streams subjected to concept drifts. EWMA detects significant changes in the mean of a sequence of random variables as long as the mean and standard deviation of the data are known in advance. However, in ECDD the mean and standard deviation are not needed. The authors of ECDD defined three parameters but its MOA implementation only has two: the weights used to differentiate recent from old instances (λ) and the minimum number of instances (n) before drifts can be detected. The default parameter values of ECDD in MOA is one of the configurations used by its authors: $\lambda = 0.2$ and $n = 30$.

The authors of SeqDrift [20] wrote it was proposed to improve on some deficiencies of the ADWIN drift detector. It uses two subwindows to represent old and new data. In its newer version, SeqDrift2, an extended version of SeqDrift1 [27], the old data is managed by the use of a reservoir sampling, a one pass method to obtain a random sample of fixed size from a data pool whose size is not known in advance. This technique presents computational efficiency in maintaining and sampling the reservoir. It also uses the Bernstein bound [28] to compare the sample means of both subwindows and, according to the authors, it presents good results compared to other published bounds, specially in distributions with low variance. The proposed parameters and their default values are the size of the pool ($b = 200$) and the drift level ($\delta = 0.01$).

SEED [21] is inspired in ADWIN and also compares two subwindows within a window W. Whenever the two sub-windows of W exhibit distinct averages (higher than a given threshold δ), the older portion of the window (W_L) is dropped. It uses the Hoeffding Inequality with Bonferroni correction proposed in ADWIN to calculate the test statistic. SEED also performs block compressions to eliminate unnecessary cut points and merge blocks that are homogeneous in nature. Its authors claim SEED is faster and more memory efficient than ADWIN. The parameters of SEED and their respective default values in MOA are a block size ($b = 32$), a compress term ($c = 75$), the threshold ($\delta = 0.05$), the growth parameter which controls the magnitude of the increment in a linear fashion ($\alpha = 0.8$), and the base value for the linear function ($\epsilon = 0.01$).

The HDDM authors [22] propose to monitor the performance of the base learner by applying "some probability inequalities that assume only independent, uni-variate and bounded random variables to obtain theoretical guarantees" for the detection of changes. HDDM also provides bounds on false positive and false negative rates. Two approaches have been proposed. The authors claim [22] the first (A_Test) is more suitable to detect abrupt changes and the second (W_Test) is better with gradual changes. Their parameters are the confidence values for drifts ($\alpha_D = 0.001$) and warnings ($\alpha_W = 0.005$), and the direction of the error: $t = 0$, error only increments (default for W_Test) or $t = 1$, error increments and decrements (default for A_Test). $\lambda = 0.05$ controls how much weight is given to more recent data in comparison to older data (only for W_Test).

The algorithm of FHDDM [23] uses a sliding window and HoeffdingâÇÖs inequality [29] to calculate and compare the maximum probability of the correct predictions observed so far (best) with the most recent probability of correct predictions for the purpose of drift detection. The authors claim the FHDDM algorithm results in less detection delay, less false positive and less false negative, when compared to the state-of-the-art. The parameters of FHDDM and their respective default values are the size of

Please cite this article as: R.S.M.d. Barros et al., Wilcoxon Rank Sum Test Drift Detector, Neurocomputing (2017), https://doi.org/10.1016/j.neucom.2017.10.051

the sliding window ($n = 25$) and the probability of error allowed ($\delta = 0.000001$).

2.1. STEPD

The basic idea of STEPD is to monitor the accuracy of a base learner over two windows: a *recent* window, containing the last examples, and an *older* window, with all the other examples seen by the base learner after the last detected drift. The size of the *recent* window (w) is a parameter and its default value is 30.

The method assumes the accuracies of the base classifier over the two aforementioned windows should be the same, as long as the concept remains stationary. Accordingly, the criterion to signal warnings and drifts is a significant decrease in accuracy detected in the examples of the *recent* window.

Similarly to DDM and EDDM, STEPD also has two parametrized thresholds referring to significance levels for the detection of drifts and warnings: $\alpha_d = 0.003$ and $\alpha_w = 0.05$.

In STEPD, the comparison of the precisions over the two windows employs a hypothesis test of equal proportions with continuity correction, as presented in Eq. (1) [4], where r_o is the number of correct predictions in the n_o examples of the *older* window, r_r is the number of correct predictions in the n_r (w) examples of the *recent* window, and $\hat{p} = (r_o + r_r)/(n_o + n_r)$.

$$T(r_o, r_r, n_o, n_r) = \frac{|r_o/n_o - r_r/n_r| - 0.5 \times (1/n_o + 1/n_r)}{\sqrt{\hat{p} \times \left(1 - \hat{p}\right) \times (1/n_o + 1/n_r)}}. \quad (1)$$

The result computed using Eq. (1) is then used to find the p-value in the standard normal distribution table, which is later compared to the significance levels adopted for drifts and warnings. STEPD detects concept drifts when $p\text{-value} < \alpha_d$ and signals warnings when $\alpha_d \leq p\text{-value} < \alpha_w$.

3. Wilcoxon rank sum test and its implementation

In statistics, when tests and models do not conform to parametric standards, i.e. one cannot assume the data satisfy a known distribution, nonparametric standards are used.

The Wilcoxon rank sum test [24] (also called Mann–Whitney U test) is a nonparametric test that can be used to determine whether two *independent* samples come from populations with the same distribution [30].

This statistical test is applicable when the samples are independent and it is useful when measurements can be sorted on ordinal scale, i.e. when the values tend to a continuous variable but may not have a normal distribution.

The test is developed under the null hypothesis that the two samples have the same distribution, against the alternative that they have different distributions.

First, it is necessary to choose the significance level (α), sometimes called risk level, which is the probability of rejecting the null hypothesis when it is true. Then, critical values found in the standard table of normal distribution are established to be compared to the test statistic in order to determine the rejection criterion of the null hypothesis. They are related to α and, so, their values are fixed after α is identified.

After, the $n_1 + n_2$ observations of the two samples are combined and sorted in ascending order, resulting in a rank in the $[1, n_1 + n_2]$ interval for each observation. In the case of ties (identical observations), their ranks are replaced by the mean of the ranks they would have if they were distinguishable. For example, if the seventh and eighth observations were identical, they would both be ranked as 7.5. Also, the sum of the ranks of both samples are calculated and the smallest ($\boldsymbol{R}$) is selected.

The test statistic is calculated by the generalized equation $z = (R - \mu_R) \ / \ \sigma_R$. Note $\mu_R = n_1 \times (n_1 + n_2 + 1)/2$ and $\sigma_R = \sqrt{n_1 \times n_2 \times (n_1 + n_2 + 1) \ / \ 12}$ are the population mean and standard deviation, respectively, and n_1 and n_2 represent the sizes of the smallest and of the largest sample, respectively.

The obtained z value is then used to reject the null hypothesis that the two samples have the same distribution when its value is in the rejection region [31], accepting that the samples come from different distributions.

Another possibility, adopted in this work, is to determine the forcefulness of the null hypothesis by calculating the p-value and estimating the strength of evidence of the respective rejection or not that both samples have the same distribution.

The p-value is the probability obtained by finding the z value in the table of the normal distribution [31]. Thus, the null hypothesis that errors *are* equally distributed on both windows should be rejected *if and only if* the obtained p-value is smaller than the chosen significance level, otherwise it is accepted.

3.1. Implementation

As previously described, computing the Wilcoxon rank sum test is relatively simple, but the prescribed *sorting* of the observations can make it computationally expensive.

In the drift detection scenario, it would mean one ordering of the results for each new instance received, even though the adoption of *insertion sort* could make it fairly efficient.

Other generalizations have also been proposed to decrease the computational cost of the original test [32].

However, note all the observations in the considered scenario are either 0 or 1, because the input to WSTD is only the information regarding the performance of a base learner: is it classifying the instances correctly? Therefore, there will be only two rank values after the application of the test. And since the number of occurrences of both are already calculated for both samples, it is possible to deduce these ranks by using mathematics, more specifically, the formula to calculate the sum of the elements of arithmetic series (AS) – finite arithmetic progressions, making the use of an explicit *sort* unnecessary.

Given that (a) STEPD keeps track of the number of correct predictions r_o and r_r in the n_o and n_r examples of the *older* and *recent* windows, respectively, (b) the corresponding numbers of wrong predictions w_o and w_r are trivially calculated from them, and (c) in MOA 0 means true and 1 means false, the result of the *sort* of the Wilcoxon test is a sequence containing $r_o + r_r$ zeros followed by $w_o + w_r$ ones.

Consequently, the ranks of the first $r_o + r_r$ observations are all equal to the mean of the values of the AS that goes from 1 to $r_o + r_r$ with $r_o + r_r$ elements. Given the sum of the elements of an AS is $(a_1 + a_n) \times n \ / \ 2$, where a_1, a_n, and n are its first, last, and number of elements, its mean value is $(a_1 + a_n) \ / \ 2$, and the resulting rank is $rRanks = (1 + r_o + r_r) \ / \ 2$.

Similarly, the rank of the remaining $w_o + w_r$ elements is given by $wRanks = r_o + r_r + (1 + w_o + w_r) \ / \ 2$.

The sum of the ranks of the elements of both samples are also straightforward: $sum_o = (rRanks \times r_o) + (wRanks \times w_o)$ and $sum_r = (rRanks \times r_r) + (wRanks \times w_r)$.

The rest of the provided implementation simply follows the description of the method and is omitted.

4. WSTD

As discussed in Section 2, STEPD maintains statistics of two windows of data and uses the statistical test of equal proportions to detect changes in the data distribution as the means to signal the warnings and drifts points.

Please cite this article as: R.S.M.d. Barros et al., Wilcoxon Rank Sum Test Drift Detector, Neurocomputing (2017), https://doi.org/10.1016/j.neucom.2017.10.051

Aiming to provide a method that identifies less false positive drifts than STEPD and is also statistically precise, this work proposes WSTD, a method that applies the Wilcoxon rank sum test in the detection of concept drifts.

WSTD works similarly to STEPD: it monitors the predictions of the base learner using two windows (*recent* and *older*), it relies on a statistical test to signal warnings and drifts, and it includes its three parameters and default values, i.e. the size of the *recent* window ($w = 30$) and the significance levels for the detection of drifts ($\alpha_d = 0.003$) and of warnings ($\alpha_w = 0.05$).

The default values of the three parameters were initially set to the same default values of STEPD to allow for a fair comparison of the methods. However, we have later run several exploratory experiments to search for better sets of values for both methods, including the use of a genetic algorithm [33], but we could not find any significantly better set of values over a large collection of datasets.

The main differences between these two are related to (a) the statistical test used to compare the samples and (b) the size of the *older* window. In STEPD, the *older* window covers all the data instances seen before those of the *recent* window. Our original intention was to adopt the same strategy. However, we noticed, experimentally, that the accuracy of the method would often degrade when the concepts were very long, irrespective of base classifier, dataset generator, or type, frequency, and severity of drift. For this reason, we decided to limit the size of the *older* window of WSTD, using a fourth parameter w_2. Its default value was experimentally set to 4000. However, we point out that using the *older* window with 500 instances is enough to deliver similarly good results, sometimes higher, especially if the base classifier is Hoeffding Tree (HT).

Algorithm 1 presents the abstract pseudo-code of WSTD. Note the inputs are a data stream and the four aforementioned parameters.

Algorithm 1: Wilcoxon Rank Sum Test Drift Detector.

```
   Input: Data stream s, recent window size w, drift level α_d, warning level α_w,
          older window size w_2
 1 storedPreds ← new byte [w]
 2 storedPreds_2 ← new byte [w_2]
 3 n_o ← n_r ← w_o ← w_r ← r_o ← r_r ← 0
 4 changeDetected ← false
 5 foreach instance in s do
 6     if changeDetected then
 7         reset storedPreds, storedPreds_2
 8         n_o ← n_r ← w_o ← w_r ← r_o ← r_r ← 0
 9         changeDetected ← false
10     Updates predictions in older and recent windows
11     Updates stats of both windows: n_o, n_r, w_o, w_r, r_o, r_r
12     isWarningZone ← false
13     if n_o ≥ w then
14         rRanks ← (1 + r_o + r_r) / 2
15         wRanks ← r_o + r_r + ((1 + w_o + w_r) / 2)
16         sum_o ← (rRanks × r_o) + (wRanks × w_o)
17         sum_r ← (rRanks × r_r) + (wRanks × w_r)
18         if sum_o < sum_r then
19             R ← sum_o
20         else
21             R ← sum_r
22         aux ← n_o + n_r + 1
23         z ← (R - n_r × aux/2) / sqrt (n_o × n_r × aux / 12)
24         p-value ← normalProbability (|z|)
25         p-value ← 2 × (1 - p-value)
26         if p-value < α_d then
27             changeDetected ← true
28         else if p-value < α_w then
29             isWarningZone ← true
```

Lines 1–4 show a simplified high-level summary of the data that needs to be instantiated in the beginning of the method.

Lines 5–29 refer to the main part of the algorithm. It is worth saying WSTD implements its necessary adjustments after a drift detection when it receives the first instance of the new concept (lines 6–9). Thus, the contents of attribute *changeDetected* at line 6 is the one set at the previous instance. In MOA, changes in the base learner after detecting drifts are *not* directly implemented in the code of any detectors – they only signal the drift points to other shared classes of the MOA implementation.

Line 10 abstracts the updates needed in both windows every time a new instance of data is processed by WSTD: the oldest instance of the *older* window is discarded, the oldest instance of the *recent* window is moved to the *older* window, and this new instance is included in the *recent* window. Line 11 reflects those changes in both windows statistics.

Observe line 13 guarantees that detections only take place after the *older* window has at least w instances, i.e. after $2 \times w$ processed instances, Lines 14–25 detail the calculation of the p-value, and drifts and warnings are detected in lines 26–29.

5. Experimental setting

This section presents all the relevant information about the experiments that were designed to test WSTD.

Firstly, the selected algorithms for the experiments were ADWIN, DDM, ECDD, SEED, SeqDrift2, and STEPD. These form a comprehensive and well-balanced selection of both traditional and newer concept drift detectors that use different strategies, with some emphasis on methods that detect drifts based on two windows of data.

Also, all these methods have been tested with both Hoeffding Tree (HT) and Naive Bayes (NB) as base learners – they are the most frequently used in experiments in the area and their implementations are available in the MOA framework.

The accuracy evaluation was performed using the Prequential methodology [34] with a sliding window of size 1000 as its forgetting mechanism [35]. Also, each incoming instance is used initially for testing and subsequently for training.

Testing the methods using artificial datasets is very important because the number, position, and duration of the concept drifts can be configured to cover different scenarios. Accordingly, we chose four artificial dataset generators, described below, and built abrupt and gradual concept drift versions of three different sizes, for a total of 24 artificial datasets. In all of them, four concept drifts were distributed at regular intervals and the size of the concepts in each dataset version of the same generator is different, covering three different scenarios.

Concept drifts were simulated by joining different concepts. In all the gradual concept drift datasets the changes last for 500 instances and were generated by a probability function, available in the MOA framework. This means that, in the 250 instances before and the 250 instances after the specified points, the class labels are progressively less likely to be based on the older concept and more likely to be based on the newer concept. At the change point, the probability should be 50% for each concept.

In all these artificial datasets, the experiments were executed 30 times to calculate the accuracies of the methods and the mean results were computed with 95% confidence intervals.

In addition to the artificial datasets, seven real-world datasets were chosen to complement the evaluation of the proposed methods and these seven datasets are also described below.

Please cite this article as: R.S.M.d. Barros et al., Wilcoxon Rank Sum Test Drift Detector, Neurocomputing (2017), https://doi.org/10.1016/j.neucom.2017.10.051

Table 1
Average accuracies in percentage (%) using HT, with 95% confidence intervals in the artificial datasets.

Type-Size	Dataset	ADWIN	DDM	ECDD	SEED	SEQDRIFT2	STEPD	WSTD	WSTD1W
Abr-20K	Agrawal	64.27 (+−0.23)	64.93 (+−1.28)	63.83 (+−0.44)	64.45 (+−0.19)	64.15 (+−0.48)	64.96 (+−0.31)	**65.33 (+−0.48)**	65.28 (+−0.48)
	LED	63.22 (+−0.58)	**71.31 (+−0.25)**	68.09 (+−0.42)	55.44 (+−0.61)	60.97 (+−1.17)	65.16 (+−1.59)	70.25 (+−0.60)	70.24 (+−0.60)
	RBF	47.44 (+−0.54)	46.92 (+−0.95)	41.81 (+−0.38)	47.10 (+−0.47)	47.76 (+−0.51)	47.50 (+−0.46)	47.98 (+−0.74)	**48.10 (+−0.58)**
	Sine	88.67 (+−0.14)	89.31 (+−0.14)	86.90 (+−0.19)	88.26 (+−0.19)	86.80 (+−0.10)	89.22 (+−0.20)	**89.93 (+−0.12)**	**89.93 (+−0.12)**
Abr-50K	Agrawal	65.73 (+−0.15)	68.03 (+−1.98)	64.76 (+−0.64)	65.24 (+−0.15)	66.84 (+−0.35)	66.18 (+−0.29)	**69.16 (+−0.72)**	67.46 (+−0.59)
	LED	64.15 (+−0.55)	71.93 (+−0.48)	68.69 (+−0.32)	55.99 (+−0.54)	64.85 (+−1.35)	67.85 (+−1.10)	**71.99 (+−0.31)**	69.15 (+−0.35)
	RBF	49.14 (+−0.36)	49.22 (+−0.62)	45.12 (+−0.32)	48.77 (+−0.38)	**49.92 (+−0.42)**	48.64 (+−0.32)	49.66 (+−0.49)	48.82 (+−0.54)
	Sine	89.88 (+−0.10)	91.06 (+−0.15)	87.12 (+−0.13)	89.24 (+−0.14)	90.23 (+−0.11)	90.37 (+−0.21)	**91.52 (+−0.13)**	87.20 (+−0.27)
Abr-100K	Agrawal	66.48 (+−0.12)	71.01 (+−2.08)	66.25 (+−0.71)	65.59 (+−0.12)	68.38 (+−0.32)	66.89 (+−0.27)	**71.62 (+−0.74)**	69.89 (+−0.63)
	LED	64.79 (+−0.46)	72.65 (+−0.30)	68.97 (+−0.23)	56.59 (+−0.91)	67.90 (+−1.11)	69.04 (+−0.73)	**72.81 (+−0.20)**	72.34 (+−0.26)
	RBF	49.95 (+−0.19)	51.96 (+−0.57)	**52.78 (+−0.85)**	49.37 (+−0.19)	50.85 (+−0.32)	49.05 (+−0.29)	50.42 (+−0.40)	49.94 (+−0.31)
	Sine	90.33 (+−0.08)	92.31 (+−0.09)	87.15 (+−0.10)	89.56 (+−0.11)	91.91 (+−0.10)	90.96 (+−0.16)	**92.59 (+−0.10)**	91.65 (+−0.09)
Grad-20K	Agrawal	63.38 (+−0.24)	64.10 (+−1.17)	62.94 (+−0.32)	63.66 (+−0.18)	62.72 (+−0.65)	64.05 (+−0.21)	**64.61 (+−0.37)**	64.46 (+−0.44)
	LED	62.43 (+−0.66)	**70.54 (+−0.19)**	67.16 (+−0.43)	55.11 (+−0.49)	60.15 (+−0.82)	64.25 (+−1.39)	68.57 (+−0.50)	68.56 (+−0.50)
	RBF	44.84 (+−0.51)	44.95 (+−0.95)	41.76 (+−0.42)	45.56 (+−0.41)	**46.20 (+−0.68)**	44.93 (+−0.58)	45.01 (+−0.86)	45.10 (+−0.86)
	Sine	85.43 (+−0.11)	**86.68 (+−0.14)**	85.04 (+−0.18)	85.34 (+−0.15)	86.55 (+−0.14)	85.89 (+−0.12)	86.67 (+−0.10)	**86.68 (+−0.11)**
Grad-50K	Agrawal	65.32 (+−0.16)	**68.46 (+−1.74)**	64.95 (+−0.74)	64.91 (+−0.13)	66.55 (+−0.28)	65.77 (+−0.25)	67.93 (+−0.65)	66.93 (+−0.53)
	LED	63.84 (+−0.62)	**72.33 (+−0.23)**	68.32 (+−0.33)	55.91 (+−0.48)	64.66 (+−1.29)	67.97 (+−0.93)	71.36 (+−0.32)	68.82 (+−0.30)
	RBF	47.26 (+−0.59)	47.28 (+−0.64)	45.07 (+−0.33)	47.88 (+−0.37)	**48.90 (+−0.58)**	47.61 (+−0.34)	48.15 (+−0.45)	47.45 (+−0.48)
	Sine	88.51 (+−0.10)	**90.27 (+−0.10)**	86.33 (+−0.14)	88.13 (+−0.11)	90.11 (+−0.10)	89.16 (+−0.20)	90.24 (+−0.12)	86.11 (+−0.28)
Grad-100K	Agrawal	66.23 (+−0.12)	**71.72 (+−1.76)**	65.79 (+−0.64)	65.37 (+−0.10)	68.25 (+−0.24)	66.63 (+−0.27)	70.90 (+−0.80)	69.63 (+−0.62)
	LED	64.60 (+−0.50)	**72.54 (+−0.34)**	68.79 (+−0.23)	56.20 (+−0.48)	68.48 (+−0.75)	68.75 (+−0.71)	72.40 (+−0.18)	72.04 (+−0.19)
	RBF	49.19 (+−0.33)	50.78 (+−0.66)	**52.52 (+−0.81)**	49.03 (+−0.19)	50.55 (+−0.38)	48.46 (+−0.30)	49.32 (+−0.32)	49.06 (+−0.35)
	Sine	89.60 (+−0.07)	**92.00 (+−0.09)**	86.80 (+−0.12)	88.88 (+−0.08)	91.89 (+−0.09)	90.38 (+−0.15)	91.93 (+−0.10)	91.00 (+−0.07)
Real	Airlines	65.17	65.30	63.82	**65.44**	65.32	65.37	65.15	65.35
	Connect4	74.42	74.12	74.99	74.92	74.30	75.25	**75.27**	**75.27**
	CovSorted	71.26	**75.64**	70.05	70.93	71.34	70.75	71.08	71.10
	Outdoor	58.57	59.27	58.84	58.67	57.92	59.74	**60.12**	**60.12**
	Pokerhand	73.83	72.73	**78.62**	75.18	72.50	77.12	76.48	76.48
	Spam	92.46	92.11	**92.60**	92.15	91.83	92.01	92.04	92.04
	Usenets	67.08	70.87	**72.44**	67.33	69.03	71.36	71.70	71.70
Rank	–	5.80645	2.91935	5.70968	6.19355	4.80645	4.67742	**2.45161**	3.43548

Table 2
Average accuracies in percentage (%) using NB, with 95% confidence intervals in the artificial datasets.

Type-Size	Dataset	ADWIN	DDM	ECDD	SEED	SEQDRIFT2	STEPD	WSTD	WSTD1W
Abr-20K	Agrawal	64.09 (+−0.17)	63.08 (+−0.59)	62.37 (+−0.15)	64.17 (+−0.18)	63.70 (+−0.34)	64.38 (+−0.18)	**64.48 (+−0.27)**	64.44 (+−0.29)
	LED	63.42 (+−0.57)	**71.32 (+−0.25)**	68.15 (+−0.42)	56.24 (+−0.73)	60.95 (+−1.04)	65.79 (+−1.63)	70.60 (+−0.44)	70.57 (+−0.44)
	RBF	47.68 (+−0.59)	47.38 (+−0.82)	39.38 (+−0.40)	47.43 (+−0.49)	47.96 (+−0.64)	47.86 (+−0.42)	48.04 (+−0.81)	**48.15 (+−0.68)**
	Sine	86.66 (+−0.17)	83.67 (+−1.77)	86.42 (+−0.16)	86.56 (+−0.17)	84.26 (+−0.17)	87.18 (+−0.16)	**87.21 (+−0.18)**	**87.21 (+−0.18)**
Abr-50K	Agrawal	65.51 (+−0.13)	63.64 (+−0.63)	62.80 (+−0.13)	65.35 (+−0.13)	65.55 (+−0.10)	65.12 (+−0.15)	**65.57 (+−0.14)**	63.90 (+−0.20)
	LED	64.76 (+−0.53)	71.66 (+−0.71)	68.73 (+−0.31)	56.69 (+−0.52)	65.00 (+−1.41)	68.73 (+−0.79)	**72.10 (+−0.33)**	69.14 (+−0.32)
	RBF	49.34 (+−0.30)	47.97 (+−1.19)	39.90 (+−0.32)	49.18 (+−0.31)	**50.18 (+−0.46)**	48.85 (+−0.37)	49.77 (+−0.54)	48.76 (+−0.48)
	Sine	87.14 (+−0.12)	84.21 (+−1.32)	86.44 (+−0.11)	87.09 (+−0.11)	86.19 (+−0.12)	87.27 (+−0.12)	**87.40 (+−0.11)**	82.46 (+−0.13)
Abr-100K	Agrawal	66.00 (+−0.08)	64.17 (+−0.68)	62.89 (+−0.08)	65.91 (+−0.11)	**66.06 (+−0.08)**	65.40 (+−0.08)	65.96 (+−0.11)	65.68 (+−0.11)
	LED	65.21 (+−0.53)	72.54 (+−0.40)	69.02 (+−0.22)	57.51 (+−0.74)	67.82 (+−1.18)	69.46 (+−0.60)	**72.85 (+−0.20)**	72.49 (+−0.16)
	RBF	50.22 (+−0.23)	48.91 (+−1.24)	39.91 (+−0.20)	49.90 (+−0.22)	**51.05 (+−0.38)**	49.27 (+−0.34)	50.45 (+−0.36)	49.99 (+−0.34)
	Sine	87.28 (+−0.08)	83.77 (+−1.40)	86.45 (+−0.10)	87.24 (+−0.08)	86.82 (+−0.08)	87.30 (+−0.08)	**87.43 (+−0.09)**	87.39 (+−0.08)
Grad-20K	Agrawal	63.07 (+−0.18)	62.62 (+−0.51)	61.85 (+−0.13)	**63.44 (+−0.20)**	62.25 (+−0.38)	63.31 (+−0.23)	63.15 (+−0.41)	63.19 (+−0.37)
	LED	62.53 (+−0.52)	**70.54 (+−0.19)**	67.21 (+−0.43)	55.85 (+−0.59)	60.56 (+−0.92)	64.11 (+−1.41)	68.68 (+−0.52)	68.68 (+−0.52)
	RBF	45.25 (+−0.60)	44.87 (+−0.95)	39.38 (+−0.40)	45.84 (+−0.39)	**46.27 (+−0.73)**	45.27 (+−0.66)	45.13 (+−0.87)	45.17 (+−0.84)
	Sine	84.03 (+−0.15)	**84.64 (+−0.20)**	84.17 (+−0.15)	83.93 (+−0.19)	84.38 (+−0.19)	84.36 (+−0.19)	84.60 (+−0.16)	84.60 (+−0.16)
Grad-50K	Agrawal	65.21 (+−0.13)	63.92 (+−0.57)	62.53 (+−0.11)	65.07 (+−0.13)	**65.28 (+−0.13)**	64.77 (+−0.14)	65.17 (+−0.14)	63.79 (+−0.20)
	LED	64.49 (+−0.50)	**72.30 (+−0.26)**	68.35 (+−0.33)	56.92 (+−0.63)	64.62 (+−1.26)	68.17 (+−0.80)	71.48 (+−0.28)	68.90 (+−0.29)
	RBF	47.75 (+−0.55)	46.60 (+−0.89)	39.90 (+−0.32)	48.19 (+−0.30)	**49.13 (+−0.59)**	47.71 (+−0.34)	48.06 (+−0.47)	47.37 (+−0.46)
	Sine	86.06 (+−0.09)	86.31 (+−0.26)	85.62 (+−0.12)	86.11 (+−0.11)	86.29 (+−0.12)	86.24 (+−0.12)	**86.63 (+−0.11)**	81.67 (+−0.11)
Grad-100K	Agrawal	65.84 (+−0.09)	64.06 (+−0.63)	62.77 (+−0.08)	65.75 (+−0.11)	**65.88 (+−0.08)**	65.16 (+−0.09)	65.69 (+−0.11)	65.46 (+−0.11)
	LED	65.42 (+−0.47)	72.34 (+−0.51)	68.83 (+−0.23)	57.28 (+−0.45)	68.51 (+−0.77)	69.17 (+−0.62)	**72.46 (+−0.18)**	72.19 (+−0.13)
	RBF	49.04 (+−0.48)	46.70 (+−1.23)	39.91 (+−0.20)	49.30 (+−0.40)	**50.30 (+−0.39)**	48.66 (+−0.33)	49.39 (+−0.39)	49.10 (+−0.36)
	Sine	86.74 (+−0.08)	86.58 (+−0.29)	86.01 (+−0.09)	86.71 (+−0.07)	86.86 (+−0.09)	86.73 (+−0.07)	**87.05 (+−0.09)**	86.98 (+−0.08)
Real	Airlines	66.70	65.35	63.66	**66.71**	66.60	65.73	66.68	**66.71**
	Connect4	74.31	74.47	75.05	74.65	74.07	75.14	**75.17**	**75.17**
	CovSorted	67.73	67.14	67.39	67.32	67.68	67.62	68.15	**68.18**
	Outdoor	58.65	59.49	60.71	59.09	58.10	**61.60**	60.91	60.91
	Pokerhand	73.69	61.98	**79.12**	75.08	72.24	77.18	76.38	76.38
	Spam	**91.87**	89.34	88.39	90.90	89.70	91.42	91.80	91.80
	Usenets	68.41	71.01	**72.75**	68.65	66.31	71.95	71.58	71.58
Rank	–	4.67742	5.48387	6.24194	5.1129	4.48387	4.14516	**2.25806**	3.59677

Please cite this article as: R.S.M.d. Barros et al., Wilcoxon Rank Sum Test Drift Detector, Neurocomputing (2017), https://doi.org/10.1016/j.neucom.2017.10.051

5.1. Datasets

The datasets that were chosen for the experiments reported in Section 6 have all been previously used in the area and are all publicly available, most of them in the MOA framework, from the MOA website, or at https://sites.google.com/site/moaextensions. These are:

- Agrawal generator [36] has information about people willing to receive a loan, who should be classified as belonging to group A or group B. Its attributes are: salary, commission, age, education level, zip code, value of the house, etc. Classification is performed using 10 functions, each with a different form of evaluation. The first five functions (F1–F5) were used to generate the datasets.
- LED generator [5,11,37] represents the problem of predicting the digit shown by a seven-segment LED display. It has 24 categorical attributes (but only seven are relevant) and a categorical class, with ten possible values. Also, each attribute has a 10% probability of being inverted (noise). Concept drifts were simulated by changing the position of the relevant attributes.
- RandomRBF generator [15,37] uses n centroids with their centers, labels and weights randomly defined, and a Gaussian distribution to determine the values of m attributes. The centroid also determines the class label, creating a normally distributed hypersphere of examples surrounding each central point with varying densities. Concept drifts were simulated by changing the seeds as well as the number and positions of the centroids.
- Sine generator [3,16,19] uses two numeric attributes (x, y) and two contexts: Sine1 and Sine2. In Sine1, an instance is positive if the point (x, y) is below the curve $y = sin(x)$, whereas Sine2 uses the condition $y = 0.5 + 0.3 \times \sin(3\pi x)$. Concept drifts can be simulated either by alternating between Sine1 and Sine2 or by reversing the aforementioned conditions, i.e. points below the curves become negative.
- Airlines [33,38] is a binary dataset composed of 539,383 instances. The goal is to predict whether flights are delayed or not, based on a set of flight information. Its attributes are: name of the company, departure time, flight number, duration, airports of origin and destination, and day of the week.
- Connect4 [39] contains all legal 8-ply positions in the game of connect-4 in which neither player has won yet, and in which the next move is not forced. It is composed of 42 attributes and 67,557 instances. The outcome class is the game theoretical value for the first player: win, loss, or draw.
- Covertype [13] contains 30 m × 30 m cells of data from US Forest Service (USFS) Region 2. It has 581,012 instances and 54 attributes, both numeric and categorical ones, and the goal is to predict the forest cover type. We used a version *sorted by the elevation attribute* [40]. It induces gradual concept drifts on the class distribution: depending on the elevation, some types of vegetation disappear while others start to appear.
- Outdoor [41] was obtained from images recorded by a mobile in a garden environment and is available at https://github.com/vlosing/driftDatasets/. The task is to classify 40 different objects, each of them approached 10 times under varying lighting conditions. Each approach represents 10 images in temporal order within the dataset, for a total of 4000 instances.
- Pokerhand [5,13] represents the problem of identifying the value of a five-card hand in the game of Poker. It has five categorical and five numeric attributes and the value of a hand (e.g. a pair, two pairs, etc.) is a categorical class with 10 possible values. In this modified version, available at the MOA website, cards are sorted by rank and suit and duplicates were removed, resulting in 829,201 instances.
- Spam [7] is based on the email messages from the Spam Assassin Collection and is available at http://mlkd.csd.auth.gr/concept_drift.html. This binary dataset was built by Katakis et al. It consists of 9324 instances and 500 attributes (words derived after feature selection). The authors claim the characteristics of these spam messages gradually change with time, i.e. this dataset contains gradual concept drift.
- Usenets was built from the usenet1 and usenet2 datasets [22,42], which are based on the twenty-newsgroups collection [39] and are also available at http://mlkd.csd.auth.gr/concept_drift.html. They simulate streams of messages sequentially presented to a user, who labels them as interesting or junk, according to his/her personal interests. Since these two datasets are fairly small (1500 instances) and they both have 99 attributes, we concatenated them to create a larger dataset.

6. Experimental results and analysis

This section introduces the results of the performed experiments, including analyses of accuracy and drift identifications of the methods over the selected datasets using two different base learners – HT and NB.

6.1. Accuracy results and analysis

Tables 1 and 2 present the accuracy results of the tested methods in all selected datasets as well as their ranks using HT and NB, respectively. Note WSTD1W is the previous/alternative/simpler implementation of WSTD with an unlimited older window. In each dataset and in the ranks, the best result is written in **bold**.

In absolute terms, WSTD improved the accuracies of STEPD in most tested datasets, i.e. the results improved in all sizes of concepts, across all dataset generators, with both abrupt and gradual concept drifts, as well as in several real-world datasets, and using the two base learners, with few exceptions. One that is notable is the pokerhand dataset.

In other words, we claim the performance of WSTD was solid in most situations, with small variations across the different scenarios. Consequently, it was the best ranked method with both

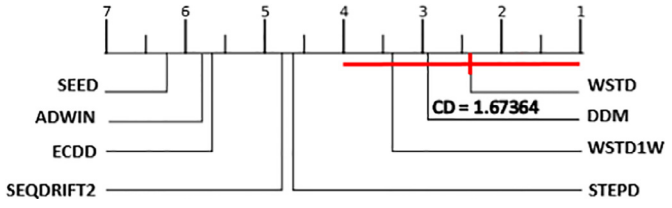

Fig. 1. Accuracy statistical comparison of the methods with Hoeffding Tree using the Bonferroni–Dunn post-hoc test on all tested datasets.

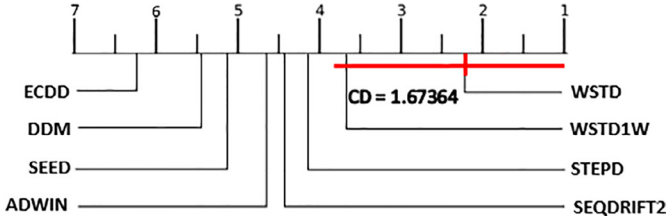

Fig. 2. Accuracy statistical comparison of the methods with Naive Bayes using the Bonferroni–Dunn post-hoc test on all tested datasets.

Table 3
Concept drift identifications of the methods in the abrupt datasets using Hoeffding Tree as base classifier.

DET.	μD	FN	FP	PREC.	Recall	MCC	Dataset	DET.	μD	FN	FP	PREC.	Recall	MCC
ADWIN	45.00	116	275	0.01434	0.03333	0.02156		SeqDr2	**0.00**	116	224	0.01754	0.03333	0.02391
DDM	43.33	117	**99**	0.02941	0.02500	0.02694	Agraw.	STEPD	44.00	60	276	0.17857	0.50000	0.29859
ECDD	32.53	**41**	970	0.07531	**0.65833**	0.22229	20K	WSTD	48.04	74	125	**0.26901**	0.38333	**0.32096**
SEED	50.87	97	295	0.07233	0.19167	0.11746		WSTD1W	47.78	75	125	0.26471	0.37500	0.31490
ADWIN	43.41	79	1314	0.03026	0.34167	0.10113		SeqDr2	**0.00**	67	1282	0.03970	0.44167	0.13191
DDM	N/A	120	**118**	0.00000	0.00000	−0.00020	LED	STEPD	28.05	38	1134	0.06743	0.68333	0.21426
ECDD	22.74	47	414	0.14990	0.60833	0.30172	20K	WSTD	30.00	**31**	239	**0.27134**	**0.74167**	**0.44844**
SEED	29.29	78	2681	0.01542	0.35000	0.07269		WSTD1W	30.00	**31**	242	0.26888	**0.74167**	0.44640
ADWIN	58.33	114	347	0.01700	0.05000	0.02882		SeqDr2	**0.00**	118	279	0.00712	0.01667	0.01059
DDM	67.50	116	114	0.03390	0.03333	0.03342	RBF	STEPD	37.72	**41**	385	0.17026	**0.65833**	0.33456
ECDD	N/A	120	**0**	0.00000	0.00000	0.00000	20K	WSTD	37.79	52	178	0.27642	0.56667	0.39561
SEED	63.88	71	391	0.11136	0.40833	0.21296		WSTD1W	37.39	51	180	**0.27711**	0.57500	**0.39901**
ADWIN	40.25	1	200	0.37304	0.99167	0.60812		SeqDr2	N/A	120	244	0.00000	0.00000	−0.00029
DDM	48.42	**0**	86	0.58252	**1.00000**	0.76318	Sine	STEPD	13.28	1	184	0.39274	0.99167	0.62397
ECDD	**10.25**	1	966	0.10968	0.99167	0.32952	20K	WSTD	17.83	**0**	1	0.99174	**1.00000**	0.99586
SEED	40.00	2	376	0.23887	0.98333	0.48449		WSTD1W	18.25	**0**	**0**	**1.00000**	**1.00000**	**1.00000**
ADWIN	134.84	56	462	0.12167	0.53333	0.25463		SeqDr2	186.52	31	333	0.21090	0.74167	0.39542
DDM	150.43	97	**91**	0.20175	0.19167	0.19658	Agraw.	STEPD	63.37	31	574	0.13424	0.74167	0.31543
ECDD	**48.02**	29	2429	0.03611	0.75833	0.16526	50K	WSTD	56.63	**22**	179	**0.35379**	**0.81667**	**0.53747**
SEED	107.50	48	639	0.10127	0.60000	0.24637		WSTD1W	60.48	78	214	0.16406	0.35000	0.23954
ADWIN	75.33	60	3179	0.01852	0.50000	0.09593		SeqDr2	158.33	24	2254	0.04085	0.80000	0.18057
DDM	147.58	87	**93**	**0.26191**	0.27500	0.26831	LED	STEPD	48.84	25	2073	0.04382	0.79167	0.18605
ECDD	**26.97**	31	1074	0.07653	0.74167	0.23809	50K	WSTD	42.02	**21**	320	0.23628	**0.82500**	**0.44144**
SEED	75.28	48	6684	0.01066	0.60000	0.07955		WSTD1W	46.04	72	392	0.10909	0.40000	0.20878
ADWIN	119.90	15	411	0.20349	0.87500	0.42189		SeqDr2	200.00	**4**	233	**0.33238**	**0.96667**	**0.56678**
DDM	130.00	90	108	0.21739	0.25000	0.23306	RBF	STEPD	56.24	27	789	0.10544	0.77500	0.28574
ECDD	N/A	120	**0**	0.00000	0.00000	0.00000	50K	WSTD	61.73	22	299	0.24685	0.81667	0.44893
SEED	92.61	5	624	0.15562	0.95833	0.38609		WSTD1W	**55.57**	50	290	0.19444	0.58333	0.33670
ADWIN	40.92	**0**	454	0.20906	**1.00000**	0.45716		SeqDr2	200.00	**0**	133	0.47431	**1.00000**	0.68867
DDM	69.67	**0**	131	0.47809	**1.00000**	0.69141	Sine	STEPD	11.93	1	386	0.23564	0.99167	0.48334
ECDD	**10.18**	8	2448	0.04375	0.93333	0.20188	50K	WSTD	17.83	**0**	**5**	**0.96000**	**1.00000**	**0.97979**
SEED	36.47	1	807	0.12851	0.99167	0.35689		WSTD1W	18.36	59	67	0.47656	0.50833	0.49215
ADWIN	162.95	25	838	0.10182	0.79167	0.28386		SeqDr2	195.88	23	607	0.13778	0.80833	0.33368
DDM	230.27	83	**84**	**0.30579**	0.30833	0.30703	Agraw.	STEPD	77.05	25	1114	0.07858	0.79167	0.24934
ECDD	**53.74**	21	4745	0.02044	0.82500	0.12970	100K	WSTD	55.91	**10**	279	0.28278	**0.91667**	**0.50910**
SEED	134.20	20	1283	0.07231	0.83333	0.24540		WSTD1W	77.50	28	392	0.19008	0.76667	0.38170
ADWIN	121.04	43	6241	0.01219	0.64167	0.08824		SeqDr2	196.40	**9**	3055	0.03506	**0.92500**	0.17998
DDM	249.79	73	**72**	**0.39496**	0.39167	0.39328	LED	STEPD	61.05	25	3509	0.02636	0.79167	0.14433
ECDD	**43.56**	33	2240	0.03739	0.72500	0.16453	100K	WSTD	46.92	16	459	0.18473	0.86667	**0.40008**
SEED	133.71	31	13198	0.00670	0.74167	0.07022		WSTD1W	53.43	21	761	0.11512	0.82500	0.30812
ADWIN	139.91	**8**	655	0.14602	**0.93333**	0.36913		SeqDr2	190.99	9	279	0.28462	0.92500	**0.51307**
DDM	229.19	83	140	0.20904	0.30833	0.25384	RBF	STEPD	73.96	19	1550	0.06118	0.84167	0.22683
ECDD	**25.22**	74	**39**	**0.54118**	0.38333	0.45545	100K	WSTD	54.52	16	584	0.15116	0.86667	0.36190
SEED	107.71	11	1112	0.08927	0.90833	0.28470		WSTD1W	56.02	32	665	0.11687	0.73333	0.29269
ADWIN	39.92	**0**	952	0.11194	**1.00000**	0.33452		SeqDr2	200.00	**0**	133	0.47431	**1.00000**	0.68869
DDM	91.42	**0**	103	0.53812	**1.00000**	0.73355	Sine	STEPD	**11.60**	1	686	0.14783	0.99167	0.38283
ECDD	16.05	6	5032	0.02215	0.95000	0.14494	100K	WSTD	16.75	**0**	**8**	**0.93750**	**1.00000**	**0.96824**
SEED	41.25	**0**	1642	0.06810	**1.00000**	0.26090		WSTD1W	17.33	**0**	160	0.42857	**1.00000**	0.65464
ADWIN	85.15	43.08	1277.33	0.11328	0.64097	0.25541		SeqDr2	152.81	43.42	754.67	0.17121	0.63819	0.30942
DDM	132.51	72.17	**103.25**	0.27107	0.39861	0.32503		STEPD	43.92	24.50	1055.00	0.13684	0.79583	0.31211
ECDD	**28.93**	44.25	1696.42	0.09270	0.63125	0.19612	MEAN	WSTD	40.50	**22.00**	223.00	**0.43013**	**0.81667**	**0.56732**
SEED	76.06	34.33	2477.67	0.08920	0.71389	0.23481		WSTD1W	43.18	41.42	290.67	0.30046	0.65486	0.42289

base classifiers. From the other tested detectors, WSTD1W, DDM, and STEPD were the best performing methods in the reported tests whereas ECDD and SEED were the worst.

Despite this, it is clear that WSTD performed better in the abrupt than in the gradual concept drift datasets. Moreover, WSTD achieved the very best results in over 66% and 58% of the abrupt datasets with HT and NB, respectively. Oddly, DDM was comparatively better in the gradual datasets, especially with HT.

It is worth noting a statistical test comparing two fixed sized windows may not be able to detect gradual changes with high confidence because the difference between the error rates of two consecutive concepts can be small. This is the most likely explanation why WSTD is better in abrupt drift scenarios.

It is also evident that WSTD was comparatively less effective in the real-world datasets. Note this fact does *not* mean WSTD was inferior to the other tested methods, but merely that it was not that much superior. These results are probably related to the fact that some of these datasets might not have any concept drifts at all whereas others might have very slow gradual concept drifts, temporal dependencies on the class label, and/or imbalanced data. The aforementioned scenarios might make several methods deliver similar results since none of them was designed for such situations.

When compared to WSTD1W, WSTD consistently delivered higher accuracies in *all* the 50K and 100K instances datasets, maintaining similar results in the 20K instances datasets and in the real-world datasets, with both base learners.

Please cite this article as: R.S.M.d. Barros et al., Wilcoxon Rank Sum Test Drift Detector, Neurocomputing (2017), https://doi.org/10.1016/j.neucom.2017.10.051

Table 4
Concept drift identifications of the methods in the abrupt datasets using Naive Bayes as base classifier.

DET.	μD	FN	FP	PREC.	Recall	MCC	Dataset	DET.	μD	FN	FP	PREC.	Recall	MCC
ADWIN	70.00	117	265	0.01119	0.02500	0.01644		SeqDr2	N/A	120	196	0.00000	0.00000	−0.00026
DDM	70.00	119	103	0.00962	0.00833	0.00877	Agraw.	STEPD	46.67	**51**	208	0.24910	**0.57500**	0.37828
ECDD	**34.26**	52	915	0.06918	0.56667	0.19760	20K	WSTD	47.41	66	**97**	**0.35762**	0.45000	**0.40102**
SEED	57.69	94	251	0.09386	0.21667	0.14235		WSTD1W	47.17	67	**97**	0.35333	0.44167	0.39490
ADWIN	46.76	83	1279	0.02812	0.30833	0.09256		SeqDr2	**0.00**	64	1304	0.04118	0.46667	0.13812
DDM	N/A	120	**117**	0.00000	0.00000	−0.00020	LED	STEPD	28.55	37	1002	0.07650	0.69167	0.22965
ECDD	22.74	47	414	0.14990	0.60833	0.30172	20K	WSTD	29.55	**32**	182	**0.32593**	**0.73333**	**0.48875**
SEED	30.86	85	2562	0.01348	0.29167	0.06191		WSTD1W	29.55	**32**	186	0.32117	**0.73333**	0.48516
ADWIN	64.00	110	323	0.03003	0.08333	0.04971		SeqDr2	**0.00**	119	265	0.00376	0.00833	0.00530
DDM	50.00	117	123	0.02381	0.02500	0.02420	RBF	STEPD	33.42	**41**	353	0.18287	**0.65833**	0.34675
ECDD	N/A	120	**0**	0.00000	0.00000	0.00000	20K	WSTD	37.58	54	159	**0.29333**	0.55000	0.40150
SEED	66.44	61	343	0.14677	0.49167	0.26838		WSTD1W	37.01	53	162	0.29258	0.55833	**0.40401**
ADWIN	40.17	**0**	110	0.52174	**1.00000**	0.72225		SeqDr2	N/A	120	242	0.00000	0.00000	−0.00028
DDM	49.76	38	94	0.46591	0.68333	0.56414	Sine	STEPD	14.33	**0**	162	0.42553	**1.00000**	0.65224
ECDD	**9.83**	2	945	0.11101	0.98333	0.33012	20K	WSTD	18.75	**0**	**3**	**0.97561**	**1.00000**	**0.98773**
SEED	40.00	**0**	207	0.36697	**1.00000**	0.60568		WSTD1W	18.75	**0**	**3**	**0.97561**	**1.00000**	**0.98773**
ADWIN	145.81	58	254	0.19620	0.51667	0.31831		SeqDr2	200.00	**20**	117	**0.46083**	**0.83333**	**0.61966**
DDM	144.00	115	**104**	0.04587	0.04167	0.04365	Agraw.	STEPD	75.47	25	429	0.18130	0.79167	0.37876
ECDD	**58.07**	32	2282	0.03713	0.73333	0.16479	50K	WSTD	81.70	32	106	0.45361	0.73333	0.57671
SEED	119.17	24	244	0.28235	0.80000	0.47521		WSTD1W	96.50	80	139	0.22346	0.33333	0.27285
ADWIN	80.16	59	2987	0.02001	0.50833	0.10057		SeqDr2	157.14	22	2230	0.04210	0.81667	0.18521
DDM	147.27	87	**93**	**0.26191**	0.27500	0.26831	LED	STEPD	45.38	27	1697	0.05196	0.77500	0.20048
ECDD	**26.85**	31	1078	0.07626	0.74167	0.23768	50K	WSTD	41.62	**21**	280	0.26121	**0.82500**	**0.46416**
SEED	78.21	64	6402	0.00867	0.46667	0.06317		WSTD1W	43.27	71	367	0.11779	0.40833	0.21920
ADWIN	124.31	11	370	0.22756	0.90833	0.45457		SeqDr2	200.00	**0**	186	**0.39216**	**1.00000**	**0.62619**
DDM	157.50	104	105	0.13223	0.13333	0.13271	RBF	STEPD	56.70	29	716	0.11276	0.75833	0.29231
ECDD	N/A	120	**0**	0.00000	0.00000	0.00000	50K	WSTD	59.31	19	289	0.25897	0.84167	0.46681
SEED	92.95	8	558	0.16716	0.93333	0.39491		WSTD1W	**52.79**	52	283	0.19373	0.56667	0.33125
ADWIN	41.50	**0**	142	0.45802	**1.00000**	0.67674		SeqDr2	200.00	**0**	128	0.48387	**1.00000**	0.69558
DDM	88.85	33	110	0.44162	0.72500	0.56580	Sine	STEPD	13.95	1	460	0.20553	0.99167	0.45139
ECDD	**9.83**	1	2512	0.04523	0.99167	0.21160	50K	WSTD	18.58	**0**	**4**	**0.96774**	**1.00000**	**0.98374**
SEED	35.50	**0**	193	0.38339	**1.00000**	0.61914		WSTD1W	18.03	59	62	0.49594	0.50833	0.50206
ADWIN	203.07	6	222	0.33929	0.95000	0.56771		SeqDr2	217.09	3	**101**	**0.53670**	0.97500	**0.72337**
DDM	313.33	111	114	0.07317	0.07500	0.07404	Agraw.	STEPD	96.47	18	778	0.11591	0.85000	0.31383
ECDD	**60.34**	31	4661	0.01874	0.74167	0.11773	100K	WSTD	106.47	18	184	0.35664	0.85000	0.55056
SEED	135.00	**0**	270	0.30769	**1.00000**	0.55468		WSTD1W	114.18	29	288	0.24011	0.75833	0.42667
ADWIN	120.13	42	6015	0.01280	0.65000	0.09103		SeqDr2	194.59	**9**	3139	0.03415	**0.92500**	0.17763
DDM	249.79	73	**86**	**0.35338**	0.39167	0.37201	LED	STEPD	54.71	16	3169	0.03178	0.86667	0.16583
ECDD	**42.99**	33	2241	0.03737	0.72500	0.16449	100K	WSTD	46.47	18	402	0.20238	0.85000	**0.41472**
SEED	145.15	21	12391	0.00793	0.82500	0.08063		WSTD1W	53.50	20	691	0.12642	0.83333	0.32453
ADWIN	126.15	3	497	0.19055	0.97500	0.43100		SeqDr2	200.00	**0**	200	**0.37500**	**1.00000**	**0.61235**
DDM	255.95	83	110	0.25170	0.30833	0.27855	RBF	STEPD	75.77	16	1427	0.06793	0.86667	0.24256
ECDD	N/A	120	**0**	0.00000	0.00000	0.00000	100K	WSTD	58.97	23	506	0.16086	0.80833	0.36055
SEED	91.51	1	837	0.12448	0.99167	0.35129		WSTD1W	**58.84**	34	621	0.12164	0.71667	0.29520
ADWIN	40.25	**0**	126	0.48781	**1.00000**	0.69842		SeqDr2	200.00	**0**	122	0.49587	**1.00000**	0.70416
DDM	126.36	32	126	0.41122	0.73333	0.54912	Sine	STEPD	13.42	**0**	893	0.11846	**1.00000**	0.34413
ECDD	**10.00**	2	5168	0.02232	0.98333	0.14803	100K	WSTD	18.17	**0**	**3**	**0.97561**	**1.00000**	**0.98773**
SEED	40.00	**0**	238	0.33520	**1.00000**	0.57894		WSTD1W	18.00	**0**	153	0.43956	**1.00000**	0.66298
ADWIN	91.86	40.75	1049.17	0.21028	0.66042	0.35161		SeqDr2	156.88	39.75	685.83	0.23880	0.66875	0.37392
DDM	150.26	86.00	**107.08**	0.20587	0.28333	0.24009		STEPD	46.24	**21.75**	941.17	0.15163	**0.81875**	0.33302
ECDD	**22.91**	49.25	1684.67	0.04726	0.58958	0.15615	MEAN	WSTD	47.05	23.58	184.58	**0.46579**	0.80347	**0.59033**
SEED	77.71	29.83	2041.33	0.18650	0.75139	0.34969		WSTD1W	48.97	41.42	254.33	0.32511	0.65486	0.44221

Complementing the analysis of the reported results, we used a statistic named F_F and the Bonferroni–Dunn post-hoc test [43] using WSTD as the base method. They were applied twice, once for the results using each base learner. We present the results using graphics where the critical difference (CD) is represented by a bar and methods that are connected to WSTD by the bar are *not* statistically different.

The results of the tests referring to the data at Tables 1 and 2 are summarized in Figs. 1 and 2, respectively. According to Fig. 1, using HT, WSTD was significantly better than all the other methods with the exception of DDM and WSTD1W. On the other hand, when the base learner was NB (Fig. 2), only WSTD1W was *not* statistically different from WSTD.

6.2. Drift identification analysis

A different perspective regarding the performance of the drift detectors can be obtained by analysing the number and position of the concept drifts identified by each method.

Tables 3 and 4 present, for each *abrupt* dataset configuration, the mean distance to the real drift points (μD) in the true positive drift identifications as well as the total number of false negatives (FN) and false positives (FP) of each method considering the 30 repetitions. The numbers of true positive (TP) and true negative (TN) detections were omitted because they can be easily calculated from the other information: $TP = 120 - FN$ and $TN = size \times 30 - 120 - FP$. Again, in each dataset and in the mean results, the best values are written in **bold**.

Please cite this article as: R.S.M.d. Barros et al., Wilcoxon Rank Sum Test Drift Detector, Neurocomputing (2017), https://doi.org/10.1016/j.neucom.2017.10.051

To compute the true positives, we considered the drifts detected within 2% of the concept size after the correct drift position. For instance, in the 20K datasets, the concepts last for 4K instances and, thus, detections occurred up to 80 instances after the exact points were considered true positives.

This analysis considered only the abrupt datasets because we know exactly where the concept drifts are. The gradual drifts datasets have no single change point and, therefore, it is not clear how the detections should be classified as positive or negative.

Regarding the mean distance of the true positives, ECDD, STEPD, and SeqDrift2 (only in 20K datasets) were the best methods in most datasets. However, these results often came at the cost of many false positive detections, hurting their accuracies. On the other hand, the detections of WSTD and WSTD1W were usually fairly close to the best results.

False negatives are related to existing drifts *not* detected by the methods whereas false positives refer to identified drifts where none exists. In both metrics, no method was clearly better than the others in either HT or NB. However, SEED followed by ECDD, ADWIN, and STEPD had the worst false positive results in most datasets with both HT and NB.

Tables 3 and 4 also present the results regarding the evaluation of the methods using *Precision* and *Recall* [44] and the Matthews Correlation Coefficient (MCC) [45], also used by Liu et al. [46]. In all of them, higher values indicate the corresponding methods performed better.

Precision, defined as $TP \,/\, (TP + FP)$, returns the proportion of predicted drifts that are existing drifts, whereas Recall, given by $TP \,/\, (TP + FN)$, represents the proportion of the existing concept drifts that were correctly detected by each method.

The MCC criterion [45], defined in Eq. (2), was included because many other criteria are severely influenced by the imbalance ratio between the numbers of positive and negative samples [46]. It returns values in the $[-1, 1]$ interval and is based on the four values of the confusion matrix: TP, TN, FP, and FN.

$$\frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}} \tag{2}$$

Analysing the results of these three criteria, WSTD was much superior to all the other methods in Precision and MCC with both HT and NB.

In the case of Recall, in most tested datasets, WSTD and STEPD delivered similar results and were much better than all the other tested methods. Observe that WSTD returned the best mean with HT and the second best with NB, with STEPD being second and first, respectively.

WSTD1W was consistently inferior to WSTD in all three metrics and delivered particularly bad results in the 50K instances datasets.

Taking into consideration all the results of our experiments regarding accuracy (Section 6.1) and drift identifications, it is natural to conclude WSTD was the best performing method overall.

7. Conclusion

This article proposed WSTD, a new method to detect concept drifts in data streams using two windows of data, similarly to STEPD. More specifically, WSTD adopts the Wilcoxon rank sum statistical test, instead of the test of equal proportions used in STEPD, and limits the size of the *older* window. An efficient implementation of Wilcoxon's test was also provided.

WSTD was compared to four well-known and two more recent concept drift detectors that also use two windows of data, as well as its previous/alternative implementation (with an unlimited older window as STEPD – WSTD1W), using both Hoeffding Tree (HT) and Naive Bayes (NB) as base learners, several artificial datasets with abrupt and gradual drifts, and also real-world datasets.

In the reported experiments, WSTD was the top-ranked method in accuracy with both base classifiers and its predictive accuracies were among the best in most tested datasets, especially the ones with abrupt concept drifts.

Moreover, according to the Bonferroni–Dunn post-hoc test, WSTD was significantly superior to ADWIN, ECDD, SEED, SeqDrift2, and STEPD with both base learners, and superior to DDM with NB, though the datasets with abrupt concept drifts played an important role in the overall results. In addition, in spite of their different ranks, no significant difference exists between WSTD and WSTD1W.

Regarding the drift identifications, WSTD presented competitive results in the distance to the correct position of the true drifts as well as in the false negative and false positive detections. Moreover, it was also much better than all the other tested concept drift detection methods in the Precision and MCC criteria. In the case of Recall, STEPD was the only method to deliver results that are similar to those of WSTD.

Thus, based on the results of our experiments, we claim WSTD was the best performing drift detection method tested, despite the similar performance of other methods in the gradual and real-world datasets.

Future work includes experimenting with other combinations of statistical tests to detect concept drift in data streams. Also, it should be interesting to include different scenarios in the artificial datasets, with other frequencies of drifts as well as longer transition periods in the gradual concept drifts datasets.

Finally, it is worth explicitly saying we used an empirical approach. In addition, WSTD was implemented in the MOA framework and its source code will soon be freely available.

Acknowledgments

Juan Hidalgo is supported by a postgraduate grant from CAPES. The authors thank Bruno Maciel for his MOA script generator and results extraction tool, which greatly helped speed up the scripts generation and the analysis of the results of the experiments, despite being under development.

References

[1] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation, ACM Comput. Surveys 46 (4) (2014). 44:1–37.
[2] P.M. Gonçalves Jr., S.G.T.C. Santos, R.S.M. Barros, D.C.L. Vieira, A comparative study on concept drift detectors, Expert Syst. Appl. 41 (18) (2014) 8144–8156.
[3] J. Gama, P. Medas, G. Castillo, P. Rodrigues, Learning with drift detection, in: Proceedings of Advances in Artificial Intelligence: SBIA 2004, in: Lecture Notes in Computer Science, 3171, Springer, 2004, pp. 286–295.
[4] K. Nishida, K. Yamauchi, Detecting concept drift using statistical testing, in: Proceedings of Tenth International Conference on Discovery Science (DS'07), in: Lecture Notes in Computer Science, 4755, Springer, 2007, pp. 264–269.
[5] P.M. Gonçalves Jr., R.S.M. Barros, RCD: a recurring concept drift framework, Pattern Recogn. Lett. 34 (9) (2013) 1018–1025.
[6] Y. Lee, L. Wang, K. Ryu, A system architecture for monitoring sensor data stream, in: Proceedings of Seventh IEEE International Conference on Computer and Information Technology, 2007, pp. 1026–1031.
[7] I. Katakis, G. Tsoumakas, I. Vlahavas, Tracking recurring contexts using ensemble classifiers: an application to email filtering, Knowl. Inf. Syst. 22 (2010) 371–391.
[8] I. Žliobaitė, M. Pechenizkiy, J. Gama, An overview of concept drift applications, in: N. Japkowicz, J. Stefanowski (Eds.), Big Data Analysis: New Algorithms for a New Society, Studies in Big Data, 16, Springer, 2016, pp. 91–114.
[9] J.Z. Kolter, M.A. Maloof, Dynamic weighted majority: an ensemble method for drifting concepts, J. Mach. Learn. Res. 8 (2007) 2755–2790.
[10] L.L. Minku, X. Yao, DDD: a new ensemble approach for dealing with concept drift, IEEE Trans. Knowl. Data Eng. 24 (4) (2012) 619–633.
[11] S.G.T.C. Santos, P.M. Gonçalves Jr., G.D.S. Silva, R.S.M. Barros, Speeding up recovery from concept drifts, in: Proceedings of the Machine Learning and Knowledge Discovery in Databases, in: Lecture Notes in Computer Science, 8726, Springer, 2014, pp. 179–194.

[12] R.S.M. Barros, S.G.T.C. Santos, P.M. Gonçalves Jr., A boosting-like online learning ensemble, in: Proceedings of IEEE International Joint Conference on Neural Networks (IJCNN), Vancouver, Canada, 2016, pp. 1871–1878.
[13] A. Bifet, G. Holmes, B. Pfahringer, Leveraging bagging for evolving data streams, in: Proceedings of the Machine Learning and Knowledge Discovery in Databases, in: Lecture Notes in Computer Science, vol. 6321, Springer, 2010, pp. 135–150.
[14] L. Du, Q. Song, L. Zhu, X. Zhu, A selective detector ensemble for concept drift detection, Comput. J. 58 (3) (2014) 457–471.
[15] B.I.F. Maciel, S.G.T.C. Santos, R.S.M. Barros, A lightweight concept drift detection ensemble, in: Proceedings of Twenty-seventh IEEE International Conference on Tools with Artificial Intelligence (ICTAI), Vietri sul Mare, Italy, 2015, pp. 1061–1068.
[16] M. Baena-Garcia, J. Del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavaldà, R. Morales-Bueno, Early drift detection method, in: Proceedings of the International Workshop on Knowledge Discovery from Data Streams, 2006, pp. 77–86.
[17] A. Bifet, R. Gavaldà, Learning from time-changing data with adaptive windowing, in: Proceedings of the Seventh SIAM International Conference on Data Mining (SDM'07), Minneapolis, MN, USA, 2007, pp. 443–448.
[18] S.H. Bach, M.A. Maloof, Paired learners for concept drift, in: Proceedings of the Eighth IEEE International Conference on Data Mining (ICDM'08), Pisa, Italy, 2008, pp. 23–32.
[19] G.J. Ross, N.M. Adams, D.K. Tasoulis, D.J. Hand, Exponentially weighted moving average charts for detecting concept drift, Pattern Recogn. Lett. 33 (2) (2012) 191–198.
[20] R. Pears, S. Sakthithasan, Y. Koh, Detecting concept change in dynamic data streams, Mach. Learn. 97 (3) (2014) 259–293.
[21] D.T.J. Huang, Y.S. Koh, G. Dobbie, R. Pears, Detecting volatility shift in data streams, in: Proceedings of 2014 IEEE International Conference on Data Mining (ICDM), Shenzhen, China, 2014, pp. 863–868.
[22] I. Frías-Blanco, J. del Campo-Ávila, G. Ramos-Jiménez, R. Morales-Bueno, A. Ortiz-Díaz, Y. Caballero-Mota, Online and non-parametric drift detection methods based on hoeffdingâĂŹs bounds, IEEE Trans. Knowl. Data Eng. 27 (3) (2015) 810–823.
[23] A. pesaranghader, H. Viktor, Fast Hoeffding drift detection method for evolving data streams, in: Proceedings of the Machine Learning and Knowledge Discovery in Databases, in: Lecture Notes in Computer Science, 9852, Springer, 2016, pp. 96–111.
[24] F. Wilcoxon, Individual comparisons by ranking methods, Biometr. Bull. 1 (6) (1945) 80–83.
[25] A. Bifet, G. Holmes, R. Kirkby, B. Pfahringer, MOA: massive online analysis, J. Mach. Learn. Res. 11 (2010) 1601–1604.
[26] S. Roberts, Control chart tests based on geometric moving averages, Technometrics 1 (3) (1959) 239–250.
[27] S. Sakthithasan, R. Pears, Y. Koh, One pass concept change detection for data streams, in: Proceedings of the Advances in Knowledge Discov. and Data Mining, in: Lecture Notes in Computer Science, 7819, Springer, 2013, pp. 461–472.
[28] S. Bernstein, The Theory of Probabilities, Gastehizdat Publishing House, Moscow, 1946.
[29] W. Hoeffding, Probability inequalities for sums of bounded random variables, J. Am. Stat. Assoc. 58 (1963) 13–30.
[30] R. Larson, B. Farber, Elementary Statistics: Picturing the World, fourth, Pearson, 2010.
[31] A.G. Bluman, Elementary Statistics: A Step by Step Approach, ninth edn., McGraw-Hill, New York, USA, 2014.
[32] J. Perolat, I. Couso, K. Loquin, O. Strauss, Generalizing the Wilcoxon rank-sum test for interval data, Int. J. Approx. Reason. 56 part A (2015) 108–121.
[33] S.G.T.C. Santos, R.S.M. Barros, P.M. Gonçalves Jr., Optimizing the parameters of drift detection methods using a genetic algorithm, in: Proceedings of the Twenty-seventh IEEE International Conference on Tools with Artificial Intelligence (ICTAI'15), Vietri sul Mare, Italy, 2015, pp. 1077–1084.
[34] A.P. Dawid, Present position and potential developments: some personal views: statistical theory: The prequential approach, J. R. Stat. Soc. Ser. A Gen. 147 (2) (1984) 278–292.
[35] J. Gama, R. Sebastião, P.P. Rodrigues, On evaluating stream learning algorithms, Mach. Learn. 90 (3) (2013) 317–346.
[36] R. Agrawal, T. Imielinski, A.N. Swami, Database mining: a performance perspective, IEEE Trans. Knowl. Data Eng. 5 (6) (1993) 914–925.
[37] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, R. Gavaldà, New ensemble methods for evolving data streams, in: Proceedings of Fifteenth ACM International Conference on Knowledge Discovery and Data Mining (KDD'09), Paris, France, 2009, pp. 139–148.
[38] A. Bifet, J. Read, I. Žliobaitė, B. Pfahringer, G. Holmes, Pitfalls in benchmarking data stream classification and how to avoid them, in: Proceedings of the Machine Learning and Knowledge Discovery in Databases, in: Lecture Notes in Computer Science, 8188, Springer, 2013, pp. 465–479.
[39] A. Frank, A. Asuncion, UCI machine learning repository, 2011, (http://archive.ics.uci.edu/ml/).
[40] D. Ienco, A. Bifet, I. Žliobaitė, B. Pfahringer, Clustering based active learning for evolving data streams, in: Proceedings of the Sixteenth International Conference on Discovery Science (DS'13), in: Lecture Notes in Computer Science, 8140, Springer, 2013, pp. 79–93.
[41] V. Losing, B. Hammer, H. Wersing, KNN classifier with self adjusting memory for heterogeneous concept drift, in: Proceedings of 2016 IEEE International Conference on Data Mining (ICDM), Barcelona, Spain, 2016, pp. 291–300.
[42] I. Katakis, G. Tsoumakas, I. Vlahavas, An ensemble of classifiers for coping with recurring contexts in data streams, in: Proceedings of Eighteenth European Conference on Artificial Intelligence, IOS Press, Patras, Greece, 2008, pp. 763–764.
[43] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[44] T. Fawcett, An introduction to ROC analysis, Pattern Recogn. Lett. 27 (8) (2006) 861–874.
[45] B.W. Matthews, Comparison of the predicted and observed secondary structure of t4 phage lysozyme, Biochim. Biophys. Acta BBA- Prot. Struct. 405 (2) (1975) 442–451.
[46] J. Liu, Q. Miao, Y. Sun, J. Song, Y. Quan, Fast structural ensemble for one-class classification, Pattern Recogn. Lett. 80 (2016) 179–187.

Roberto Souto Maior de Barros received B.Sc. and M.Sc. degrees in Computer Science from Universidade Federal de Pernambuco (UFPE), Brazil, in 1985 and 1988, respectively, and his Ph.D. degree in Computing Science from The University of Glasgow, Scotland (UK), in 1994. From 1985 to 1995, he worked as systems analyst at UFPE and he is a full time Professor (full) and Researcher, also at UFPE, since 1995. His main research areas are software engineering, programming languages, XML, pattern recognition, and machine learning, with special interest in concept drift.

Juan Isidro González Hidalgo received his B.Sc. degree in Computer Engineering from Universidad de Granma, Cuba, in 2012, and his M.Sc. degree in Computer Science from Universidade Federal de Pernambuco (UFPE), Brazil, in 2017. He is now a Ph.D. student at UFPE and his main research areas are pattern recognition and machine learning, with special interest in concept drift and statistics.

Danilo Rafael de Lima Cabral received his B.Sc. degree in Systems Analysis and Development from Universidade Estácio de Sá, Brazil, in 2014, and his M.Sc. degree in Computer Science from Universidade Federal de Pernambuco (UFPE), Brazil, in 2017. He works as IT Software Developer at UFPE since 2014 and he is also a Ph.D. student at UFPE. His main research areas are pattern recognition and machine learning, with special interest in concept drift and statistics.

Please cite this article as: R.S.M.d. Barros et al., Wilcoxon Rank Sum Test Drift Detector, Neurocomputing (2017), https://doi.org/10.1016/j.neucom.2017.10.051