Information Sciences 485 (2019) 376–393

Contents lists available at ScienceDirect

Information Sciences

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

A differential evolution based method for tuning concept drift detectors in data streams

Silas G.T.C. Santos[a,*], Roberto S.M. Barros[a], Paulo M. Gonçalves Jr.[b]

[a] Centro de Informática, Universidade Federal de Pernambuco, Cidade Universitária, Recife 50.740-560, PE, Brazil
[b] Instituto Federal de Educação, Ciência e Tecnologia de Pernambuco, Cidade Universitária, Recife 50.740-540, PE, Brazil

ARTICLE INFO

Article history:
Received 13 March 2018
Revised 14 January 2019
Accepted 11 February 2019
Available online 12 February 2019

Keywords:
Concept drift detectors
Tuning
Differential evolution
Data stream
Online learning

ABSTRACT

Predicting online on data streams, with data flowing continuously, quickly, and in large quantities, is becoming increasingly more important in tackling real-world problems. In such scenarios, data distribution usually evolves over time, a situation known as concept drift. This article presents an empirical method, based on a differential evolution, aimed at guiding users on how to tune concept drift detectors to improve their accuracies in data streams. It also suggests the best detectors, strategies and/or parameterization to use in different data stream scenarios. The most time-consuming part of the method was preprocessed and is based on experiments using eight artificial dataset generators, each of them with five abrupt, fast gradual, and slow gradual concept drift versions (120 dataset versions in all), as well as six real-world datasets, for 11 different drift detectors and two different base learners. The use of the proposed method is illustrated mostly with Drift Detection Method (DDM) together with Naive Bayes, but we compared the performances of all 11 detectors using their default parameter values and the several parameter sets prescribed by the method using both base classifiers—Naive Bayes and Hoeffding Trees. Results indicate that the predictive accuracies of the detectors tuned using the method increased considerably in many situations.

1. Introduction

Streaming environments are characterized by data flowing continuously, quickly and in large quantities, and they thus usually require online learning. Concept drift is a situation frequently described as changes in the data distribution over time, which is common in data stream environments and has received increasing attention in the machine learning and data mining communities.

A well-known categorization of concept drift is based on the speed of change. When the drift from one concept to another is sudden and/or rapid, it is called *abrupt*, and when the transition between two consecutive concepts occurs over a number of instances, it is called *gradual*.

Examples of online learning applications include detecting changes in water temperature or weather, filtering spam in e-mail messages, and monitoring movement data from sensors, among others.

Different approaches have been proposed to adapt classifiers considering concept drifts [23]. Several researchers have proposed ensembles with different weighting functions to compute the resulting classification with internal use of base

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

https://doi.org/10.1016/j.ins.2019.02.031
0020-0255/

classifiers [9,13,29,32,40]. Other algorithms concentrate on concepts that recur [25]. However, a fairly large proportion of the propositions are concept drift detection methods [8,26], lightweight software that normally focuses on identifying whether there is a change in the data distribution. In addition, these concept drift detectors are sometimes used in ensembles [9,13,25,31,32,40].

The literature on concept drift detectors is vast. Its most well-known examples are Drift Detection Method (DDM) [22], Early Drift Detection Method (EDDM) [4], Adaptive Windowing (ADWIN) [11], Statistical Test of Equal Proportions (STEPD) [33], EWMA for Concept Drift Detection (ECDD) [37], and Paired Learners (PL) [3]. More recent methods include Sequential Drift (SeqDrift) [34], Drift Detection Methods based on Hoeffding's Bounds (HDDM) [20], Fast Hoeffding Drift Detection Method (FHDDM) [35], Reactive Drift Detection Method (RDDM) [5,6], Wilcoxon Rank Sum Test Drift Detector (WSTD) [5,7], and Fisher Test Drift Detector (FTDD) [16].

Each detector has its own parameters and their optimal values vary depending on the datasets used, the type of drift these datasets have, and the values of the other parameters, etc.

Evolutionary algorithms [19] are a family of heuristic methods based on ideas of natural selection to solve optimization problems. In other words, evolutionary algorithms are search methods that aim to optimize the values of different variables in a given problem by simulating the survival of the fittest individuals.

Our previous work [39] proposed a specific evolutionary algorithm—based on the Genetic Algorithm (GA) [24]—to optimize the parameter values of concept drift detection methods. The default values chosen for some drift detectors are not always good choices because, in some methods, these values were set to meet specific situations. The aforementioned paper showed how to use the proposed algorithm to assist in finding suitable configurations for specific datasets and presented a simple aggregation strategy (mean results) to optimize the parameters of drift detectors over several datasets.

The work described here extends and improves the previous publication in the following respects:

- It proposes a method aimed at guiding the user on how to optimize the parameters of concept drift detection methods for real (potentially infinite) data streams. It covers several scenarios, depending on the availability (or not) of a data sample of the stream, the amount of time available for the tuning, the existence (or not) of previous knowledge concerning the type of concept drifts in the stream, etc.
- Because it is an optimization problem that in most cases uses real variables, a different evolutionary approach was used to better deal with this characteristic. Thus, the proposed framework is now based on Differential Evolution (DE) [43] concepts.
- In the cases where the parameterization of the drift detectors are optimized based on the results of experiments using artificial data, the method prescribes the use of different aggregation functions for different detectors and scenarios, rather than the simple mean results of the parameter values found by the evolutionary algorithm, previously adopted for all drift detection methods.
- The experiments run to set the parameters of the detection methods using the DE were executed on a much larger and diverse set of artificial datasets. Moreover, this more time-consuming task was carried out (preprocessed) for 11 different concept drift detectors and its results are the basis for our suggestions of the best methods and aggregation functions to use with both Naive Bayes (NB) and Hoeffding Tree (HT) as base learners.
- To some extent, we also consider this work a survey and evaluation of existing detectors, because the optimized methods were also compared against the others and a clear statement is made regarding our choices of best drift detection methods available.
- The code has been modified to (a) optionally include in its initial population one or more parameter sets provided by the user, e.g. the default parameterization of the methods, and (b) detect the drift methods automatically in the framework used—Massive Online Analysis (MOA) [12]—including their specific characteristics (e.g. number of parameters, minimum and maximum values, etc.).

The rest of the article is organized as follows: Section 2 briefly surveys related work, including the drift detectors used in the tests. Section 3 describes DE algorithms, how they work, and how they were used to optimize the parameters of concept drift detectors, also detailing the new features of the implementation. The configuration of the experiments and the datasets used are presented in Section 4. Section 5 introduces the proposed method. The results of the experiments are analyzed in Section 6, including an informal step-by-step application of the method using DDM and some of the results using other methods, both with NB as base classifier. Finally, Section 7 presents our conclusions.

2. Related work

A recurrent difficulty regarding parameterized algorithms (including concept drift detection methods) is finding the best parameter set of values to use, since the value of one parameter may influence the best values for the others. As previously stated [39], it is possible to use evolutionary algorithms to optimize parameter values and, to the best of our knowledge, nobody has ever tried to optimize the parameters of concept drift detectors before, even though some approaches have used this strategy to deal with similar parameterization problems.

Friedrichs and Igel [21] proposed the use of evolutionary strategies to optimize the hyper-parameter for Support Vector Machine (SVM). Their experiments reported significant improvements in most cases and, based on their results, they claimed that evolutionary optimization was generally better than gradient-based methods for SVM model selection.

Another reported use of an evolutionary algorithm in the SVM context [30] attempted to optimize multi-class classification parameters divided into multiple binary sub-problems. The optimizations were performed on each binary SVM and the results were compared to the ones with the parameters set by default.

Soares et al. [42] used a GA to optimize the performance of a neural network ensemble in one of the reported experiments. The subset of models and their respective aggregate functions were the optimization targets and, when compared to other well-known techniques, their experiments reported solid results.

Arce et al. [2] proposed an adaptation of the DE technique to optimize the number of dendrites and the dendrite parameters of a Dendrite Morphological Neural Network (DMNN). Experimental results showed that the DMNN optimized by DE presented a better or equal performance compared to traditional classifiers such as Radial Basis Network (RBN), SVM, and Multilayer Perceptron (MLP).

The suitability of the evolutionary algorithms to solve these problems encouraged us to adapt the generic GA (previously) and DE to optimize the parameter values of concept drift detection methods, rather than merely rely on the default values chosen by their authors.

2.1. Drift detection methods and parameters

DDM [22] detects changes in a distribution by analyzing the probability of error (p) and its standard deviation (s). For each instance i, the predictions of the base classifier ($\hat{y}$) are compared to the correct classes (y) to compute p_i, which will be reduced if $\hat{y} = y$ and increased if $\hat{y} \neq y$. The value of s will be updated by $s_i = \sqrt{p_i \times (1 - p_i)/i}$. In addition to p and s, the minimum values of the probability of error ($p_{\min}$) and standard deviation ($s_{\min}$) are also stored while the instances are arriving and they will be updated whenever $p_i + s_i < p_{\min} + s_{\min}$. Both the concept drift and the warning levels are detected when $p_i + s_i \geq p_{\min} + \alpha \times s_{\min}$, where α is a confidence level, and it should be noted that the warning confidence level (α_w) is lower than the drift confidence level (α_d). DDM has three parameters: the minimum number of instances before the detection of a drift is permitted (n) and the two confidence levels. The default values defined by the authors of DDM for its parameters are $n = 30$, $\alpha_w = 2.0$ and $\alpha_d = 3.0$.

EDDM [4] is similar to DDM but, rather than using the error rate, it monitors the distance between two errors to identify concept drifts. It computes the average distance between two errors (p_i) and its standard deviation (s_i). These values are stored ($p_{\max}$ and $s_{\max}$) when $p_i + 2 \times s_i$ reaches its maximum value. Its authors claim that EDDM is more suitable than DDM for detecting gradual concept drifts, while DDM is better for abrupt concept drifts. EDDM has three parameters, the minimum number of errors (e) before the detection of a drift is permitted and the warning (w) and drift (d) levels. However, its implementation in the MOA framework has a fourth parameter, similar to the n of DDM. The default values provided for them are $n = e = 30$, $w = 0.95$, and $d = 0.9$.

ADWIN [11] uses a sliding window of instances (W) whose size is automatically adjusted: the longer a stream remains in the same distribution, the larger the size of W; when changes occur, W is reduced. Two dynamically adjusted sub-windows are stored, representing older and recent data, respectively. The drifts are detected when the difference in the averages between these sub-windows is higher than a given threshold. ADWIN has two parameters: a confidence level to reduce the window size—$\delta \in (0, 1)$—and the minimum frequency of instances needed for the window size to be reduced—f. The default values of the parameters of ADWIN in its implementation in the MOA framework are $\delta = 0.002$ and $f = 32$.

STEPD [33] is a concept drift detection method that monitors the performance of a base learner over two windows of the data: a recent window, containing the last w examples, and an older window, with all the other examples processed by the base classifier after the last detected drift. The accuracies of these windows are compared using a statistical test of equal proportions, which was implemented using Pearson's chi-square test with Yates' correction for continuity [45]. Drifts and warnings are signaled when significant decreases in the accuracy of the recent window are detected and these can only occur when the number of instances is at least twice the size of the recent window with the most recent examples, i.e. after $2 \times w$ instances. The parameters of STEPD with their respective default values are the window size (w=30) and the significance levels for detecting drifts (α_d=0.003) and warnings (α_w=0.05).

ECDD [37] is an adaptation of Exponentially Weighted Moving Average (EWMA) [36] to be used in data streams with concept drifts. EWMA detects significant changes in the mean of a sequence of random variables, provided the mean and standard deviation of the data are known in advance. In ECDD, however, the mean and standard deviation are not previously known. In the MOA implementation, ECDD has two parameters, the instance weights used to differentiate recent from old instances (λ) and the minimum number of instances before permitting the detection of drifts (n); its third parameter (ARL_0) is now hardcoded with value 400. Since no default values have been defined by the authors for this parameter and this value coincides with the best value found for all configurations in our previous experiments [39], for the purpose of comparison, we ignored this third parameter and chose one of the configurations used by the authors for the other two: λ=0.2 and n=30.

The authors of SeqDrift [34] wrote that it was proposed to improve on some deficiencies of the ADWIN drift detector. SeqDrift uses two sub-windows to represent old and new data. In its newer version, SeqDrift2, an extended version of

Table 1
Default parameters of the detectors with their respective ranges.

Detector	p1 [range]	p2 [range]	p3 [range]	p4 [range]	p5 [range]	p6 [range]
DDM	n=30 [0, MaxInt]	α_w=2.0 [1.0, 4.0]	α_d=3.0 [1.0, 5.0]	N/A	N/A	N/A
EDDM	e=30 [0,MaxInt]	w=0.95 [0.5, 1.0]	d=0.9 [0.5, 1.0]	N/A	N/A	N/A
ADWIN	δ=0.002 [0.0, 1.0]	f=32 [1, MaxInt]	N/A	N/A	N/A	N/A
STEPD	w=30 [0, 1000]	α_d=0.003 [0.001, 0.05]	α_w=0.05 [0.01, 0.2]	N/A	N/A	N/A
ECDD	n=30 [0, MaxInt]	λ=0.20 [0.0, MaxFloat]	N/A	N/A	N/A	N/A
SeqDrift	b=200 [100, 10000]	δ=0.01 [0.0, 1.0]	N/A	N/A	N/A	N/A
HDDM$_A$	d=0.001 [0.0, 1.0]	w=0.005 [0.0, 1.0]	t=1 [0, 1]	N/A	N/A	N/A
HDDM$_W$	d=0.001 [0.0, 1.0]	w=0.005 [0.0, 1.0]	t=1 [0, 1]	λ=0.05 [0.0, 1.0]	N/A	N/A
FHDDM	n=25 [0, MaxInt]	δ=0.000001 [0.0, 1.0]	N/A	N/A	N/A	N/A
RDDM	n=129 [0, MaxInt]	α_w=1.773 [1.0, 4.0]	α_d=2.258 [1.0, 5.0]	max=40000 [1, MaxInt]	min=7000 [1, 20000]	$wLim$=1400 [1, 20000]
WSTD	n=30 [0, 1000]	α_d=0.003 [0.0, 1.0]	α_w=0.05 [0.0, 1.0]	w_2=4,000 [30, 10000]	N/A	N/A

SeqDrift1 [38], the old data is managed using 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 [10] to compare the sample means of both sub-windows, presenting good results compared to other published bounds, especially in distributions with low variance. Its parameters and their default values are the size of the pool (b=200) and the drift level (δ=0.01).

The HDDM authors propose to monitor performance by applying "some probability inequalities that assume only independent, univariate and bounded random variables to obtain theoretical guarantees for the detection of such distributional changes" [20], differently from DDM, EDDM, and ECDD, for example, which assume that measured values are given according to a Bernoulli distribution. HDDM also provides bounds on both false positive and false negative rates, whereas ECDD focuses only on the false positives. Two main approaches were proposed. The authors claim [20] that the first (A_Test, HDDM$_A$) "involves moving averages and is more suitable to detect abrupt changes" and the second (W_Test, HDDM$_W$) "follows a widespread intuitive idea to deal with gradual changes using weighted moving averages". The two versions have three common parameters, the confidence values for drifts (α_D =0.001) and warnings (α_W=0.005), and the direction of the error, which can be one-sided (t=0, only increments), default for HDDM$_W$, or two-sided (t=1, error increments and decrements), default for HDDM$_A$. Finally, HDDM$_W$ has a fourth parameter (λ=0.05) to control how much weight is given to more recent data in comparison to older data.

FHDDM [35] is based on a sliding window and uses Hoeffding's inequality [27] to calculate and compare the maximum probability of the correct predictions previously observed (best) with the most recent probability of correct predictions in order to signal the drifts. Its authors claim that their algorithm presents less detection delay, and fewer false positives and negatives than state-of-the-art detectors. The parameters of FHDDM and their respective default values are the size of the sliding window ($n = 25$) and the probability of error allowed ($\delta = 0.000001$).

The algorithm of RDDM [5,6] was proposed to overcome/alleviate a performance loss problem of DDM when concepts are very large, caused by decreased sensitivity, taking too many instances to detect the changes. RDDM added an explicit mechanism to discard older instances of very long concepts, periodically recalculating the statistics of DDM responsible for detecting the warning and drift levels. In addition, RDDM forces concept drift detections when the number of instances of the warning period reaches a parameterized threshold. The authors of RDDM claim that it delivers higher global accuracy than DDM in most situations, especially in gradual concept drift datasets, by detecting more drifts and detecting them earlier, despite a small increase in false positives and in memory consumption. The default parameters of RDDM are $n = 129$, $\alpha_w = 1.773$, $\alpha_d = 2.258$, $maximum = 40,000$, $minimum = 7,000$, and $warnLimit = 1,400$.

Drawing on STEPD, WSTD [5,7] also detects concept drifts in data streams using two windows of data. More specifically, WSTD detects concept drifts based on an efficient implementation of the Wilcoxon rank sum statistical test [44], rather than the test of equal proportions used in STEPD. The authors claim that WSTD detects fewer false positives than STEPD and is particularly efficient in detecting abrupt drifts. In addition to the three parameters of STEPD with the same default values ($n = 30$, $\alpha_d = 0.003$, $\alpha_w = 0.05$), WSTD limits the size of the *older* window using an extra parameter ($w_2 = 4,000$).

To summarize the information presented in this subsection, Table 1 shows the default parameters of the methods, as well as their ranges.

3. Differential evolution

As already mentioned, DE algorithms [43] search for solutions to multidimensional optimization problems by simulating ideas based on natural selection. Their main features are: creation of an initial population, definition of the objective function, mutation and crossover. This section briefly details each of these steps as well as the changes performed to adapt a generic DE to the context of concept drift detectors and improvements in the implementation with respect to the GA previously implemented [39].

3.1. Population initialization

The first step in a DE is to form an initial population, a set *rh* of *P* randomly selected hypotheses. In addition, some hypotheses may be defined by a specialist. In our problem, *rh* matches the set of parameters of a particular concept drift detector.

In an attempt to make the hypotheses space as widely distributed as possible, some restrictions have been set: the values of each parameter must have a lower bound (*LB*) and an upper bound (*UB*), and the initial random values for the parameters must be evenly distributed in the [*LB, UB*] interval, reducing the risk of generating numbers in the same region and the possibility of convergence to a local minimum.

Algorithm 1 handles this distribution. Note that *rh* is a random hypotheses matrix with *P* rows (hypotheses) and *K*

Algorithm 1: Well-distributed hypotheses generation.

Input: random hypotheses *rh*, lower bound *LB*, upper bound *UB*, generated population size *P*

```
1 i ← LB; p ← 1; size ← (UB − LB)/P;
2 while i < UB do
3 |   rh_p ← Random number between i and i + size;
4 |   i ← i + size;
5 |   p ← p + 1;
6 return h;
```

columns (parameters). In addition, for each column k ($k \in \{1, 2, \ldots, K\}$), different values of *LB* and *UB* are used, which are specific for each parameter. Further, here *P* represents the slots of the initial population that will be generated, i.e. it does not include parameter sets already informed by the user (*uh*), a new feature described in Section 3.6.

3.2. Defining the objective function

The objective function or fitness of an individual refers to his/her ability to live and reproduce in a given environment. The greater this capacity, the fitter/more adapted the individual is in relation to the others.

To simulate this process in our DE algorithm, the different parameter values are the characteristics of each individual. Since the purpose of the DE is to find features that maximize the fitness, the accuracy of the method with the chosen values is its fitness. Note that other options could also be used as the optimization target, such as memory or runtime usage.

3.3. Differential evolution operators

Two of the main factors responsible for the evolution of individuals are the operators known as mutation and crossover. Some of their characteristics are the prevention of convergence in a local minimum and the exchange of gene sequence between different chromosomes.

Using DE algorithm concepts, the mutation process is initially performed. Assume that h is a matrix containing the initial population, which is composed of the randomly generated hypotheses (*rh*—Section 3.1) plus the user-determined hypotheses (*uh*). In addition, consider that m is a matrix, with the same dimensions as h, which will receive the mutant individuals. Thus, for each $p \in \{1, 2, \ldots, P\}$ (where P is the population size), we have:

$$m_p = h_{r1} + F(h_{r2} + h_{r3}), \tag{1}$$

where $r1$, $r2$, and $r3$ are indexes chosen randomly in the interval $\in [1, P]$ and F is a real and constant parameter $\in [0, 2]$ which controls the amplification of the differential variation [43]. It is important to note that $r1$, $r2$, and $r3$ are different from each other and different from the current p.

Once the mutations have been performed (stored in m), the second step is the crossover. For this purpose, consider that c is a matrix that will store individuals after crossover and has the same dimensions as h (P rows and K columns). In this way, for each $p \in \{1, 2, \ldots, P\}$ and $k \in \{1, 2, \ldots, K\}$, the process can be defined as

$$c_{p,k} = \begin{cases} m_{p,k}, & \text{if } rand_1(k) \leq CR \text{ or } k = rand_2(p) \\ h_{p,k}, & \text{if } rand_1(k) > CR \text{ and } k \neq rand_2(p) \end{cases} \tag{2}$$

where *CR* is a real crossover constant $\in [0, 1]$ which has to be determined by the user; $rand_1(k)$ is a real random number defined in the interval $\in [0, 1]$; and $rand_2(p)$ is a randomly chosen index $\in [1, K]$.

The idea behind the conditions in Eq. (2) is to increase population diversity through perturbations inserted into the mutation process. Thus, for each individual present in the population h, at least one of the parameters will be modified. In addition, diversity can be controlled by the *CR* parameter: the higher the value, the greater the likelihood of the parameters being disturbed.

3.4. Selection

The selection stage of the DE algorithm will ensure that only the best individuals pass to the next generation. Thus, after the crossover step, the matrices h and c will be compared. Knowing that an arbitrary line p ($p \in \{1, 2, \ldots, P\}$) of h and c will contain versions of the same individual before and after the crossover, the one who gets the best fitness will be passed on to the next generation. In this way, a comparison in pairs will be performed for each p in order to select the best ones.

3.5. Structure of the differential evolution algorithm

Algorithm 2 presents the basic structure of DE. Firstly, a population is generated with P hypotheses, some with user-

Algorithm 2: Differential Evolution structure.

Input: number of generations G, population size P, user-informed hypotheses uh, differential variation F, crossover constant CR

```
h ← uh ∪ (P - length uh) randomly generated hypotheses;
foreach p in h do
    compute Fitness(p);
for g = 1 to G do
    Create the mutation matrix m using F (Eq. (1));
    Create the crossover matrix c using CR (Eq. (2));
    foreach p in c do
        compute Fitness(p);
    for p = 1 to P do
        h_p ← Fittest(h_p, c_p);
return the hypothesis from h that has the greatest fitness;
```

informed parameter sets (uh) and the rest with random values in the parameters (rh). From the hypotheses created, the objective function is calculated. Thus each element in h contains a hypothesis and its respective result.

3.6. New features

New features have been added in this version of the evolutionary algorithm. The first permits the inclusion of n elements in the initial population h. The parameter values of these elements are informed by the user. This feature is helpful when there is previous knowledge of the method and/or the dataset that will be used. In addition, we deem it a good practice to use this functionality to pass the default parameters of the detectors as proposed by the authors, because it ensures that the DE is going to find values that deliver at least the same accuracies as those obtained with the default values. We emphasize that this functionality is optional and can be ignored by the user when setting up the execution of the DE.

The second improvement is related to the drift methods that can be optimized by the DE and the way they can be inserted and removed from the list of supported methods. In the previous GA implementation [39], these methods and all the information regarding their parameters were hard-coded, so that adding or removing a method meant maintenance and re-compilation of the code. In this new DE version, all the information regarding the detectors are automatically read from the existing methods in the MOA framework. Thus, when a new detector is added to MOA, the DE algorithm recognizes it without any further intervention.

4. Configuration of the experiments

This section describes all the relevant information on the experiments conducted.

We used both Naive Bayes (NB) and Hoeffding Tree (HT) as base learners with all tested drift detectors because of their simplicity, freely available implementations, and widespread use in data stream research experiments.

To evaluate the experiments, we used the Interleaved Test-Then-Train methodology, the basic window version of Prequential [17]. Each instance is first tested and then used for training. This guarantees that every instance is used for both testing and training and the problem of training before testing on any given instance is avoided.

To compute the method's accuracies in the artificial datasets, the experiments were executed 30 times and mean results were computed with 95% confidence intervals.

The experiments were run in the MOA framework, release 2014.11, in a computer with an Intel Core i7 8700 processor, 16GB of DDR4 2666 MHz RAM, an SSD, and the Ubuntu Desktop 18.04 LTS 64 bits operating system.

We selected eight artificial dataset generators and, for each, we created versions with abrupt changes (speed of change=1), fast gradual changes (speed=100) and slow gradual changes (speed=500) as well as five different sizes of concept: 1000, 2000, 5000, 10,000, and 20,000 instances. Specifically, we created 10,000 and 50,000 instances versions, both

combined with nine and four changes, and 100,000 instances versions with four changes, i.e. we used 15 different dataset versions of each generator for a total of 120 artificial datasets. Finally, we also used six real-world datasets in the evaluation of the DE results, which are also described below.

4.1. Artificial datasets

Agrawal generator [1] combines information from people requesting a loan, who are classified as belonging to groups A or B. The attributes—three of them categorical—are salary, commission, age, education level, zip code, the value of the house, the amount of the loan, etc. To perform the classification, the authors propose 10 functions, each with different forms of evaluation.

LED generator [14,40] aims to predict the digit of a seven-segment LED display. It has 24 categorical attributes, 17 of which are irrelevant, and one categorical class with 10 possible values, where each attribute has a 10% probability of being inverted (noise). To simulate concept drifts, the positions of the relevant attributes are inverted.

Mixed generator [4,22] has two Boolean (v, w) and two numeric attributes (x, y). Each instance should be classified as positive or negative. They are positive if at least two of the three following conditions are satisfied: v, w, and $y < 0.5 + 0.3\sin(3\pi x)$. To simulate a concept drift, the labels of these conditions are reversed.

RandomRBF generator [14] uses n centroids with their centers, labels, and weights randomly defined, and a Gaussian distribution to determine the values of m attributes. The datasets were generated using six classes, 40 attributes, and 50 centroids. The concept drifts are simulated by changing the positions of the centroids.

SEA generator [14] has two classes and three randomly generated numeric attributes, with the third used for noise. The numbers are randomly generated in the [0,10] interval. Each instance is class 1 if $f_1 + f_2 \leq \theta$, where f_1 and f_2 are the first two attributes and θ is a threshold used to produce different contexts, its value is 8, 9, 7 or 9.5.

Sine generator [22,37] has two numeric attributes (x, y) and two contexts. In Sine1, an instance is positive if the point is below the curve $y = \sin(x)$. In Sine2, the relevant condition is $y < 0.5 + 0.3\sin(3\pi x)$. In both of them, concept drifts are simulated by reversing the described conditions, i.e. points below the curves become negative or alternate between Sine1 and Sine2.

Stagger generator [41] uses three categorical attributes: size (small, middle or large), color (red, blue or green) and shape (circle, triangle or rectangle). Each example is positive or negative. Drifts are performed alternating between three non-disjoint contexts: the first is positive when size=small and color=red; the second when color=green or shape=circle; and the third when size=middle or size=large.

Waveform generator [14,31] has 40 numerical attributes, 19 of them used for noise. The goal of the problem is to detect the form of the waves generated by combining two of three base waves. To create concept drift, the positions of the attributes, which represent contexts, are reversed.

4.2. Real datasets

Airlines [15] is a binary real-world dataset composed of 539,383 instances. The goal is to discover whether or not a flight is delayed, based on a set of flight information: name of the company, departure time, flight number, duration, airports of origin and destination, and day of the week.

Covertype [14] contains 30 m $\times$ 30 m cells of data from US Forest Service (USFS) Region 2. It has 581,012 instances and 54 attributes, both numeric and categorical, and the goal is to predict the forest cover type. In addition to the original dataset, we also used a version sorted by the *elevation* attribute [28], which induces a natural gradual drift on the class distribution. This occurs because, depending on the elevation, some types of vegetation disappear while others appear.

Electricity [22,29,32,40] has 45,312 instances and eight attributes and presents data collected from the Australian New South Wales Electricity Market. Prices vary every 5 min and depend on market supply and demand. The goal of the problem is to predict whether the price will increase or decrease in relation to a moving average of 24 h.

PokerHand [14,25] represents the problem of identifying the value of a hand (e.g. a pair, two pairs, etc.) of five cards in the Poker game, which is a categorical class with 10 possible values. This dataset has 1,000,000 instances with five categorical and five numeric attributes. Since the cards are not ordered, it is very hard for propositional learners, especially for linear ones. The version used here is the normalized one, where cards are sorted by rank and suit and duplications have been removed, resulting in a dataset with 829,201 examples.

Finally, Sensor Stream [46] contains information collected from 54 sensors deployed in the Intel Berkeley Research Lab (temperature, humidity, light, and sensor voltage). It contains consecutive information recorded over a 2-month period, with one reading every 1–3 min. The sensor ID is the target attribute, which must be identified based on the sensor data and the corresponding recording time. This dataset is made up of 2,219,803 instances, 5 attributes and 54 classes.

5. The method

This section describes the proposed method for tuning concept drift detectors running together with a base classifier in real data streams. It provides a detailed, albeit still simple, step-by-step guide to help the users choose the best detectors

available and to find and/or reuse the best possible parameter values, depending on possible restrictions regarding the existence (or not) of a sample and time for tests, and of prior knowledge of the stream's type of concept drift.

However, before presenting the method per se, we describe the procedure adopted in the experiments that were run to assist in deciding the specific contents of the method. While reading the rest of this section, the reader may find it useful to refer to Section 6, which provides more details on the results of the experiments using NB.

5.1. Procedure for the experiments

The first step in the experiments was to use our Differential Evolution algorithm to optimize the drift detection methods parameter values using the artificial datasets. After some experimentation, the parameters of the DE were set as: population size=30; differential variation=0.5; crossover constant=0.4; and number of generations=100.

These tests were run to find optimized parameters for all 11 drift detectors chosen, i.e. DDM, EDDM, ADWIN, STEPD, ECDD, HDDM$_A$, HDDM$_W$, SeqDrift2, FHDDM, RDDM, and WSTD, in all 120 versions of the artificial datasets, independently for the two base learners—NB and HT.

Then, from all the values obtained for each parameter of each method in this round of tests, several aggregation functions were tried aimed at producing more general parameterization sets for each method. The specific functions used in the experiments and the names we refer to them are as follows:

Mean: This option simply calculates the mean value to aggregate the results. It is probably not the ideal choice because outliers may negatively influence the results in most methods.

Quartiles: This option is similar to *Mean*, but excludes the higher and lower quartiles of data before calculating the mean—it uses the intermediate 50% of the data with the aim of reducing the problem with outliers.

Octiles: This is very similar to Quartiles but it only discards Octiles, keeping 75% of the data.

Weighted: This option weighs the results of the DE in the methods using the improvement in accuracy that each one brings in each dataset, when compared to those with default parameters, with respect to the maximum improvement possible. Given ADE_n, the accuracy result of the DE in a dataset n, and $ADef_n$, the result using the default parameters in the same dataset, the calculated weight of this set of parameters is $100 * (ADE_n - ADef_n)/(100 - ADef_n)$.

Abrupt, Fast and Slow segments: Versions of the previous functions calculated with only the abrupt, fast gradual and slow gradual datasets, respectively, were also tested. Note that the segmented mean, with versions Abrupt Mean, Fast Mean, and Slow Mean, corresponds to the strategy used in our previous work [39].

In the specific case of the six selected real-world datasets, an extra set of parameters was generated for each detector–dataset combination executing the DE on samples containing the first 10% instances of the datasets—an arbitrarily chosen proportion of the data. The rationale was to simulate the availability of a sample from a real data stream and this decision was also the means used to prevent the methods from over-fitting. We refer to these parameter sets as *Sample*.

After calculating the specific aggregated values for all parameters of all the methods using the aforementioned functions in the appropriate collection of values, which also includes the segments, the 11 detection methods were executed, and their accuracy results collected, using the default and each of these calculated parameter sets in all the artificial *and* real-world datasets. In addition, the detectors were also run in the real-world datasets using their respective *Sample* parameter sets.

The following step was to compare these new results to discover the best aggregation functions for each method, and also over all data, as well as the best methods considering their respective best results on (a) the collection of artificial datasets, (b) the three segments (Abrupt, Fast, and Slow), with the datasets grouped by the type of drift they have, and, finally, (c) the real-world datasets.

To carry out this task, we used a statistic called F_F [18], a variation of the Friedman test, and the post-hoc Nemenyi test, with 95% confidence, to find out which of the configurations were statistically superior to the others, and this involved different comparisons. Some of these results are presented at Section 6.

To conclude, it is important to point out that most of this time-consuming procedure needs only be repeated if the user wants to experiment with (a) a different set (possibly larger) of artificial data, (b) other functions for aggregating the several parameter results of the DE, (c) other (new) drift detection methods, or (d) a different base learner.

5.2. Description of the method

We now proceed to describe the method. It was based on the results of the experiments, including the statistical evaluations, and following the procedure just described, that we decided which specific strategies would be the most suitable to attempt in different real-world situations—we called them scenarios. To facilitate references, the scenarios covered by the method are referred to using labels *Sn*, where *n* are sequential numbers.

S1: Sample and time are available

When (a) a sample of the data stream is available for testing the detectors *and* (b) there is enough time for tuning different methods using the DE, running the DE with a few detectors on the available sample should be the chosen option.

Table 2
Mean run-time (in minutes per 1000 instances) the DE consumed to find parameters for the recommended methods with 95% confidence.

Classifier	ADWIN	DDM	ECDD	EDDM	FHDDM	$HDDM_A$	$HDDM_W$	RDDM	SeqDrift	STEPD	WSTD
NB	0.82 ± 0.19	0.81 ± 0.19	0.88 ± 0.20	0.88 ± 0.20	0.88 ± 0.19	0.87 ± 0.20	0.97 ± 0.21	0.89 ± 0.19	0.85 ± 0.20	0.83 ± 0.19	0.89 ± 0.20
HT	1.15 ± 0.25	1.74 ± 0.41	1.41 ± 0.33	2.77 ± 0.69	1.31 ± 0.31	3.29 ± 0.73	6.32 ± 1.65	1.84 ± 0.38	1.11 ± 0.25	2.19 ± 0.49	1.97 ± 0.37

The number of methods obviously depends on the amount of time available: more methods mean more time is needed, but it also increases the chances of choosing the most appropriate detector for predicting the specific data stream. The same applies to the size of the sample: the larger it is, the more time the tests are going to take, but it also increases the chances of finding a good set of parameter values for the detector.

It is worth mentioning that the sample should be large enough to have at least a few concept drifts, in order to make it more likely that the DE will find an effective parameter set. If this is not the case, it may be more sensible to adopt one of the other scenarios.

In most experiments with both artificial and real-world datasets using NB as base classifier, RDDM, DDM, and $HDDM_A$ were the best three methods and are therefore the first recommended choices with NB. Should there be time for tuning one more detector, the recommended method is ECDD.

After these experiments are run, the selected detection method should be the one with the best accuracy in the executions of the DE in the sample, and its chosen parameters should be the ones returned by the DE.

It is important to emphasize that the arrangement to simulate this parameterization option using the 10% samples of the real-world datasets returned the very best accuracy results in most tested scenarios using NB. Moreover, considering the collection of these combinations, this parameterization option was statistically superior to all the others.

Finally, to illustrate the time required for the DE to find good parameters for the recommended methods using the samples, Table 2 shows the mean run-time, in minutes per 1000 instances, considering all the datasets used in the experiments with both base classifiers. For each of these means, the intervals were defined with 95% confidence.

S2: Sample is available but *time is limited*

When (a) a sample of the data stream is available for testing but (b) the time for tuning the methods is limited and insufficient for running the DE, the recommendation is to execute some subset of the methods suggested in scenario *S1* on the available sample with a few parameter sets obtained in the experiments using the artificial datasets.

Assuming there is no prior knowledge of the type of concept drift that exists on the data sample, we recommend testing using two sets of parameters for each of the selected methods. These proposed methods were the ones that achieved the best ranks in the statistical evaluations performed using the artificial datasets and, with the exception of the two RDDM sets, there were few significant differences between them using NB.

On the other hand, if the expert already knows the type of concept drift the data stream contains, an extra set of parameter values should be tested as a third option for each method. Note that, in some cases, it is preferable to use a new method rather than the third parameterization option (e.g. FHDDM with Slow Quartiles rather than the third parameterization for $HDDM_A$).

It is worth mentioning we do *not* recommend discarding any of the more general aforementioned parameter sets. Based on the results of experiments run, if time is critical, we argue that it is preferable to test the three sets of parameters of fewer methods than to run more methods with fewer parameter sets.

As in scenario *S1*, the chosen parameter sets were the ones with the best ranks in the statistical evaluations performed using the respective segments (Abrupt, Fast, and Slow) of the artificial datasets.

Table 3 aggregates all the specific sets of parameter values we chose for the four selected drift detectors in each of the four cases covered in this scenario using NB as base learner. Note that the parameter sets selected in this scenario *S2* also depend on the type of concept drift expected in the stream. Again, the selected drift detector should be the one with the best accuracy in the set of executions run using the sample.

S3: Sample is not *available*

When *no sample* of the data stream is available, the only feasible option is to use the best method–parameterization pair that emerges from the collection of executed experiments.

Using NB, given that the type of concept drift that exists on the data stream is unknown, the chosen drift detector should be RDDM configured with the *Quartiles* parameter set, the configuration that achieved the highest rank in the statistical evaluation of the experiments using the artificial datasets. Similarly, the best method and parameter set combinations resulting from the statistical tests for the cases where the type of drift is known in advance were:

- RDDM with the *Abrupt Quartiles* parameter set, for abrupt datasets;
- ECDD with the *Fast Quartiles* parameters, for datasets with fast gradual concept drifts; and
- RDDM with the *Slow Quartiles* parameter set, for slow gradual concept drift datasets.

Table 3
Best detectors and parameter set combinations in scenario S2 with Naive Bayes.

Detector	p1	p2	p3	p4	p5	p6	Config. name
	No prior knowledge of type of concept drift						
RDDM	n=171	α_w=1.393	α_d=2.078	*max*=448032	*min*=7345	*wLim*=7991	Quartiles
RDDM	n=129	α_w=1.773	α_d=2.258	*max*=40000	*min*=7000	*wLim*=1400	Default
DDM	n=174	α_w=1.631	α_d=2.117	N/A	N/A	N/A	Quartiles
DDM	n=154	α_w=1.672	α_d=2.207	N/A	N/A	N/A	Weighted
HDDM$_A$	d=0.001	w=0.005	t=1	N/A	N/A	N/A	Default
HDDM$_A$	d=0.01322	w=0.392	t=0	N/A	N/A	N/A	Quartiles
ECDD	n=124	λ=0.033	N/A	N/A	N/A	N/A	Quartiles
ECDD	n=30	λ=0.20	N/A	N/A	N/A	N/A	Default
	Abrupt concept drift (extra set)						
RDDM	n=176	α_w=1.266	α_d=1.915	*max*=493138	*min*=8239	*wLim*=8488	Abrupt Quartiles
DDM	n=175	α_w=1.415	α_d=1.920	N/A	N/A	N/A	Abrupt Quartiles
ECDD	n=125	λ=0.056	N/A	N/A	N/A	N/A	Abrupt Quartiles
HDDM$_A$	d=0.01460	w=0.544	t=0	N/A	N/A	N/A	Abrupt Quartiles
	Fast gradual concept drift (extra set)						
ECDD	n=101	λ=0.011	N/A	N/A	N/A	N/A	Fast Quartiles
RDDM	n=148	α_w=1.477	α_d=2.090	*max*=353998	*min*=6372	*wLim*=7171	Fast Quartiles
DDM	n=186	α_w=1.754	α_d=2.148	N/A	N/A	N/A	Fast Quartiles
HDDM$_A$	d=0.00991	w=0.304	t=0	N/A	N/A	N/A	Fast Quartiles
	Slow gradual concept drift (extra set)						
RDDM	n=183	α_w=1.456	α_d=2.434	*max*=494801	*min*=7486	*wLim*=8585	Slow Quartiles
DDM	n=165	α_w=1.734	α_d=2.319	N/A	N/A	N/A	Slow Quartiles
ECDD	n=150	λ=0.042	N/A	N/A	N/A	N/A	Slow Quartiles
FHDDM	n=637	δ=0.012	N/A	N/A	N/A	N/A	Slow Quartiles

It is important to point out that, in most of these scenarios using NB, there were other combinations with similar results to those of the selected methods, with no statistical differences.

Should the user prefer to use one method and parameter set for all cases, the options using NB as base learner are RDDM, with either *Quartiles* or *Default* parameters, or DDM, with either *Quartiles* or *Weighted* parameters.

To conclude, we remind the reader that these recommendations are specific for NB, are based on empirical tests, and depend on the datasets used in the experiments, especially the specific values found for the parameter sets. The corresponding recommendations for HT are presented in Section 6.3.

6. Details on the results of the experiments

This section provides more details on the results of the experiments. Because the number of tests was fairly large, we deemed it inappropriate to include all the results here. We therefore show detailed results for only one of the detectors (DDM) with one base learner (NB). Note that the contents of this section also serve as an informal step-by-step guide on how to apply the procedure described in Section 5.1.

Following the proposed procedure, we initially ran the DE to find optimized parameters for all detectors in all artificial datasets, individually. Next, we used the *Mean, Quartiles, Octiles*, and *Weighted* aggregation functions to produce general parameter sets for each drift method, which were then executed using all these parameters as well as their respective default configurations, and all their accuracy results were collected.

Table 4 presents the general results for DDM using NB—the *Octiles* configuration was omitted to make the data fit on a single page. The best result in each dataset is shown in **bold**.

By merely glancing at the results of Table 4 it is easy to observe that the *Quartiles* and *Weighted* parameter sets produced the best results of DDM, and this fact was later confirmed by the statistical evaluation.

Similarly, the experiment results using the three segmented groups of datasets (Abrupt, Fast, and Slow) have also been placed in separate tables. Table 5 shows the results for DDM with NB using only the abrupt datasets. The tables containing the results of the other segments have been omitted in order to avoid excessive information, and the same is true for some parameter sets of the abrupt segment. Again, the best results are shown in **bold**.

It is worth mentioning that, in the case of DDM, despite being numerically different, all the aggregation functions produced parameters that changed in the same direction. In all of them, α_w and α_d decreased, making DDM more sensitive to changes, and n increased considerably, meaning that more instances will be needed before concept drifts are detected, possibly lowering false alarm detections.

Experiments using the real-world datasets have also been run and their results are presented in Table 6. In this case, the table contains the data of the four best methods. Once again, the best results of each method in each dataset are shown in

Table 4
Accuracy results of DDM + Naive Bayes in the artificial datasets with 95% confidence intervals.

Dat.	Param.	Agrawal	Mixed	SEA	Stagger	Sine	Wave	LED	RBF
Abr.	Default	60.24 ± 0.95	88.59 ± 0.39	84.15 ± 0.47	98.99 ± 0.07	83.47 ± 0.89	78.56 ± 0.55	66.88 ± 0.40	**64.63 ± 0.66**
10k	Mean	63.20 ± 0.52	88.97 ± 0.33	84.51 ± 0.47	99.03 ± 0.08	84.09 ± 0.74	79.07 ± 0.56	67.12 ± 0.41	64.12 ± 0.66
9D	Quartiles	**63.86 ± 0.42**	**89.20 ± 0.38**	**84.63 ± 0.48**	99.06 ± 0.08	**85.11 ± 0.39**	**79.25 ± 0.55**	**67.22 ± 0.37**	63.37 ± 0.69
	Weighted	63.64 ± 0.45	89.15 ± 0.37	84.61 ± 0.50	**99.07 ± 0.08**	85.09 ± 0.38	79.22 ± 0.55	67.21 ± 0.42	63.66 ± 0.67
Abr.	Default	64.34 ± 0.98	89.73 ± 0.24	85.20 ± 0.33	99.35 ± 0.06	85.47 ± 0.38	78.46 ± 0.48	68.78 ± 0.32	**36.62 ± 0.58**
10k	Mean	67.24 ± 1.05	89.97 ± 0.24	85.42 ± 0.31	99.37 ± 0.06	85.59 ± 0.66	78.81 ± 0.45	69.01 ± 0.32	36.41 ± 0.53
4D	Quartiles	**69.80 ± 0.93**	**90.05 ± 0.25**	85.48 ± 0.32	**99.38 ± 0.05**	**85.78 ± 0.78**	**78.97 ± 0.46**	**69.11 ± 0.32**	36.06 ± 0.51
	Weighted	69.38 ± 0.96	90.02 ± 0.27	**85.54 ± 0.34**	**99.38 ± 0.05**	85.65 ± 0.81	78.92 ± 0.48	69.10 ± 0.32	36.10 ± 0.51
Abr.	Default	64.26 ± 0.92	90.75 ± 0.28	86.38 ± 0.32	99.79 ± 0.02	83.74 ± 0.76	79.03 ± 0.29	67.14 ± 1.50	**66.08 ± 0.46**
50k	Mean	66.84 ± 0.47	90.90 ± 0.25	86.87 ± 0.26	99.79 ± 0.02	84.79 ± 0.95	79.47 ± 0.24	71.81 ± 0.25	65.56 ± 0.38
9D	Quartiles	67.42 ± 0.13	90.88 ± 0.34	**87.04 ± 0.24**	**99.80 ± 0.02**	**85.95 ± 0.30**	**79.83 ± 0.23**	**72.01 ± 0.18**	65.21 ± 0.38
	Weighted	**67.47 ± 0.15**	**90.92 ± 0.27**	87.03 ± 0.22	**99.80 ± 0.02**	85.60 ± 0.69	79.78 ± 0.23	**72.01 ± 0.21**	65.31 ± 0.38
Abr.	Default	70.76 ± 0.82	91.06 ± 0.50	87.02 ± 0.21	99.85 ± 0.01	85.09 ± 0.77	79.53 ± 0.16	72.08 ± 0.26	**38.41 ± 0.40**
50k	Mean	72.93 ± 0.41	91.24 ± 0.30	87.31 ± 0.18	99.85 ± 0.01	86.01 ± 0.97	79.73 ± 0.19	72.33 ± 0.35	38.16 ± 0.39
4D	Quartiles	**73.53 ± 0.26**	**91.40 ± 0.12**	87.38 ± 0.16	**99.86 ± 0.01**	**86.28 ± 1.06**	**79.92 ± 0.19**	**72.58 ± 0.14**	38.05 ± 0.35
	Weighted	73.51 ± 0.28	91.18 ± 0.58	**87.39 ± 0.16**	**99.86 ± 0.01**	85.87 ± 1.28	79.86 ± 0.17	72.56 ± 0.14	38.04 ± 0.36
Abr.	Default	72.36 ± 0.66	91.14 ± 0.58	87.40 ± 0.14	**99.92 ± 0.01**	85.00 ± 0.86	79.64 ± 0.17	72.82 ± 0.19	**39.06 ± 0.36**
100k	Mean	73.28 ± 0.59	91.33 ± 0.61	87.60 ± 0.16	**99.92 ± 0.01**	85.89 ± 0.69	79.82 ± 0.13	73.12 ± 0.12	39.00 ± 0.31
4D	Quartiles	74.19 ± 0.11	91.56 ± 0.14	87.74 ± 0.11	**99.92 ± 0.01**	**86.81 ± 0.28**	**80.10 ± 0.14**	**73.17 ± 0.12**	38.82 ± 0.27
	Weighted	**74.22 ± 0.08**	**91.62 ± 0.12**	**87.76 ± 0.11**	**99.92 ± 0.01**	86.66 ± 0.42	80.01 ± 0.14	**73.17 ± 0.12**	38.85 ± 0.31
Fast	Default	59.36 ± 0.91	**86.96 ± 0.30**	84.11 ± 0.49	**96.03 ± 0.24**	83.19 ± 0.62	78.49 ± 0.56	66.64 ± 0.40	**64.70 ± 0.67**
Grad.	Mean	62.68 ± 0.45	86.83 ± 0.31	84.47 ± 0.49	95.05 ± 0.17	83.35 ± 0.66	78.87 ± 0.55	66.87 ± 0.38	64.16 ± 0.68
10k	Quartiles	63.13 ± 0.54	86.62 ± 0.37	**84.59 ± 0.49**	94.81 ± 0.16	83.43 ± 0.31	**79.14 ± 0.55**	**66.90 ± 0.37**	63.25 ± 0.66
9D	Weighted	**63.19 ± 0.45**	86.69 ± 0.37	84.58 ± 0.53	94.95 ± 0.17	**83.50 ± 0.30**	79.08 ± 0.55	66.89 ± 0.38	63.53 ± 0.68
Fast	Default	63.67 ± 0.88	89.34 ± 0.23	85.17 ± 0.33	**98.37 ± 0.08**	85.49 ± 0.31	78.40 ± 0.47	68.66 ± 0.32	**36.59 ± 0.56**
Grad.	Mean	66.02 ± 1.01	**89.35 ± 0.22**	85.48 ± 0.33	98.35 ± 0.07	85.29 ± 0.62	78.70 ± 0.46	68.80 ± 0.33	36.45 ± 0.57
10k	Quartiles	**69.05 ± 0.98**	89.20 ± 0.25	85.48 ± 0.33	98.25 ± 0.08	**85.72 ± 0.46**	**78.92 ± 0.43**	**68.86 ± 0.32**	36.05 ± 0.50
4D	Weighted	68.64 ± 1.14	89.27 ± 0.24	**85.54 ± 0.34**	98.29 ± 0.08	85.40 ± 0.69	78.86 ± 0.45	68.84 ± 0.32	36.21 ± 0.50
Fast	Default	63.84 ± 1.01	90.54 ± 0.54	86.35 ± 0.29	**99.21 ± 0.04**	84.20 ± 0.84	79.02 ± 0.29	66.41 ± 1.46	**66.06 ± 0.46**
Grad.	Mean	66.85 ± 0.33	90.85 ± 0.15	86.88 ± 0.27	99.18 ± 0.04	85.07 ± 0.82	79.54 ± 0.23	71.80 ± 0.21	65.54 ± 0.41
50k	Quartiles	**67.28 ± 0.18**	90.82 ± 0.15	87.03 ± 0.25	99.13 ± 0.04	**85.87 ± 0.21**	**79.82 ± 0.24**	**71.97 ± 0.17**	65.15 ± 0.35
9D	Weighted	67.25 ± 0.17	**90.87 ± 0.15**	**87.06 ± 0.23**	99.14 ± 0.04	85.58 ± 0.67	79.74 ± 0.22	71.91 ± 0.22	65.30 ± 0.37
Fast	Default	70.86 ± 0.78	91.23 ± 0.15	87.03 ± 0.19	**99.63 ± 0.02**	85.69 ± 0.48	79.51 ± 0.15	72.27 ± 0.17	**38.38 ± 0.40**
Grad.	Mean	72.59 ± 0.41	91.28 ± 0.26	87.33 ± 0.19	99.62 ± 0.02	86.03 ± 0.95	79.71 ± 0.18	72.29 ± 0.37	38.19 ± 0.38
50k	Quartiles	73.00 ± 0.31	**91.30 ± 0.13**	87.36 ± 0.16	99.60 ± 0.02	**86.31 ± 1.12**	**79.95 ± 0.18**	**72.56 ± 0.14**	38.03 ± 0.38
4D	Weighted	**73.13 ± 0.31**	91.12 ± 0.48	**87.38 ± 0.16**	99.61 ± 0.02	85.98 ± 1.29	79.89 ± 0.16	72.53 ± 0.15	38.07 ± 0.36
Fast	Default	72.26 ± 0.65	91.49 ± 0.20	87.40 ± 0.15	**99.81 ± 0.01**	84.77 ± 1.02	79.63 ± 0.18	72.81 ± 0.22	**39.08 ± 0.36**
Grad.	Mean	73.66 ± 0.32	**91.51 ± 0.28**	87.64 ± 0.13	99.80 ± 0.01	86.26 ± 0.52	79.88 ± 0.14	73.11 ± 0.12	39.01 ± 0.33
100k	Quartiles	74.13 ± 0.11	91.46 ± 0.31	87.73 ± 0.11	99.78 ± 0.01	**87.05 ± 0.16**	**80.07 ± 0.14**	**73.16 ± 0.12**	38.84 ± 0.27
4D	Weighted	**74.14 ± 0.10**	91.45 ± 0.34	**87.77 ± 0.10**	99.79 ± 0.01	86.94 ± 0.21	80.04 ± 0.14	**73.16 ± 0.11**	38.79 ± 0.28
Slow	Default	58.35 ± 0.69	76.75 ± 0.31	83.76 ± 0.53	86.96 ± 0.30	74.79 ± 0.41	78.06 ± 0.55	63.36 ± 0.40	**64.66 ± 0.66**
Grad.	Mean	60.01 ± 0.41	**76.82 ± 0.31**	84.00 ± 0.51	87.24 ± 0.30	75.18 ± 0.34	78.59 ± 0.52	**63.52 ± 0.40**	64.23 ± 0.67
10k	Quartiles	**60.58 ± 0.28**	76.75 ± 0.32	83.99 ± 0.50	**87.31 ± 0.32**	75.21 ± 0.31	**78.76 ± 0.52**	63.43 ± 0.41	63.39 ± 0.67
9D	Weighted	60.18 ± 0.51	76.77 ± 0.34	**84.03 ± 0.51**	87.25 ± 0.30	**75.24 ± 0.32**	78.71 ± 0.52	63.45 ± 0.41	63.71 ± 0.67
Slow	Default	63.09 ± 0.83	84.86 ± 0.31	85.10 ± 0.35	94.86 ± 0.15	82.00 ± 0.34	78.03 ± 0.45	**66.89 ± 0.36**	**36.56 ± 0.58**
Grad.	Mean	64.89 ± 1.27	84.92 ± 0.27	**85.44 ± 0.35**	**95.10 ± 0.11**	82.30 ± 0.34	78.35 ± 0.43	66.87 ± 0.33	36.45 ± 0.54
10k	Quartiles	**66.29 ± 1.26**	85.03 ± 0.27	85.34 ± 0.33	95.08 ± 0.10	82.53 ± 0.24	**78.59 ± 0.40**	66.74 ± 0.32	36.14 ± 0.51
4D	Weighted	66.03 ± 1.35	**85.06 ± 0.28**	85.41 ± 0.34	95.02 ± 0.10	**82.61 ± 0.21**	78.50 ± 0.44	66.72 ± 0.31	36.19 ± 0.51
Slow	Default	63.94 ± 0.93	88.68 ± 0.13	86.19 ± 0.36	**97.22 ± 0.06**	84.53 ± 0.20	78.85 ± 0.28	69.71 ± 0.90	**66.15 ± 0.48**
Grad.	Mean	66.60 ± 0.22	88.63 ± 0.13	86.63 ± 0.29	**97.22 ± 0.05**	84.63 ± 0.15	79.48 ± 0.22	71.22 ± 0.19	65.63 ± 0.41
50k	Quartiles	**66.81 ± 0.11**	**88.77 ± 0.14**	86.87 ± 0.27	**97.22 ± 0.06**	84.67 ± 0.15	**79.71 ± 0.22**	**71.35 ± 0.16**	65.24 ± 0.39
9D	Weighted	66.80 ± 0.12	**88.77 ± 0.15**	**86.88 ± 0.23**	97.21 ± 0.06	**84.70 ± 0.15**	79.63 ± 0.21	71.30 ± 0.18	65.30 ± 0.41
Slow	Default	70.75 ± 0.77	**90.56 ± 0.12**	87.01 ± 0.21	98.97 ± 0.03	86.14 ± 0.36	79.43 ± 0.17	72.11 ± 0.16	**38.38 ± 0.41**
Grad.	Mean	72.76 ± 0.37	90.49 ± 0.12	87.36 ± 0.18	98.98 ± 0.03	86.52 ± 0.14	79.60 ± 0.17	72.21 ± 0.14	38.20 ± 0.37
50k	Quartiles	73.16 ± 0.25	90.50 ± 0.11	87.35 ± 0.17	98.97 ± 0.04	**86.68 ± 0.13**	**79.79 ± 0.17**	72.19 ± 0.16	38.00 ± 0.37
4D	Weighted	**73.26 ± 0.16**	90.53 ± 0.10	**87.37 ± 0.17**	**98.99 ± 0.04**	86.63 ± 0.14	**79.79 ± 0.17**	**72.22 ± 0.15**	38.11 ± 0.38
Slow	Default	72.11 ± 0.70	**91.27 ± 0.07**	87.42 ± 0.14	99.46 ± 0.02	86.33 ± 0.40	79.58 ± 0.13	72.73 ± 0.22	**39.09 ± 0.33**
Grad.	Mean	73.68 ± 0.29	91.26 ± 0.07	87.66 ± 0.13	99.46 ± 0.02	86.79 ± 0.18	79.84 ± 0.14	73.02 ± 0.11	38.99 ± 0.33
100k	Quartiles	74.03 ± 0.07	91.24 ± 0.07	87.75 ± 0.11	99.48 ± 0.02	**87.06 ± 0.10**	**80.05 ± 0.14**	**73.04 ± 0.12**	38.85 ± 0.28
4D	Weighted	**74.04 ± 0.09**	91.26 ± 0.07	**87.76 ± 0.11**	**99.49 ± 0.02**	87.03 ± 0.11	79.97 ± 0.13	**73.04 ± 0.12**	38.91 ± 0.30

Table 5
Accuracy of DDM + Naive Bayes with general and abrupt parameter sets in artificial datasets containing abrupt drifts with 95% confidence intervals.

Dat.	Param.	Agrawal	Mixed	SEA	Stagger	Sine	Wave	LED	RBF
	Default	60.24 ± 0.95	88.59 ± 0.39	84.15 ± 0.47	98.99 ± 0.07	83.47 ± 0.89	78.56 ± 0.55	66.88 ± 0.40	**64.63 ± 0.66**
	Mean	63.20 ± 0.52	88.97 ± 0.33	84.51 ± 0.47	99.03 ± 0.08	84.09 ± 0.74	79.07 ± 0.56	67.12 ± 0.41	64.12 ± 0.66
Abr.	Ab.Mean	63.65 ± 0.44	89.10 ± 0.40	84.61 ± 0.51	99.07 ± 0.08	84.59 ± 0.61	79.22 ± 0.56	**67.32 ± 0.40**	63.85 ± 0.65
10k	Quartiles	**63.86 ± 0.42**	**89.20 ± 0.38**	**84.63 ± 0.48**	99.06 ± 0.08	**85.11 ± 0.39**	79.25 ± 0.55	67.22 ± 0.37	63.37 ± 0.69
9D	Ab.Quart.	63.80 ± 0.40	88.94 ± 0.47	84.57 ± 0.48	**99.10 ± 0.08**	85.06 ± 0.43	**79.31 ± 0.55**	67.31 ± 0.39	62.95 ± 0.62
	Weighted	63.64 ± 0.45	89.15 ± 0.37	84.61 ± 0.50	99.07 ± 0.08	85.09 ± 0.38	79.22 ± 0.55	67.21 ± 0.42	63.66 ± 0.67
	Ab.Weigh.	63.77 ± 0.47	89.02 ± 0.42	84.45 ± 0.49	99.09 ± 0.09	85.02 ± 0.38	79.20 ± 0.53	67.27 ± 0.43	62.69 ± 0.62
	Default	64.34 ± 0.98	89.73 ± 0.24	85.20 ± 0.33	99.35 ± 0.06	85.47 ± 0.38	78.46 ± 0.48	68.78 ± 0.32	**36.62 ± 0.58**
	Mean	67.24 ± 1.05	89.97 ± 0.24	85.42 ± 0.31	99.37 ± 0.06	85.59 ± 0.66	78.81 ± 0.45	69.01 ± 0.32	36.41 ± 0.53
Abr.	Ab.Mean	68.87 ± 0.98	89.88 ± 0.38	85.53 ± 0.32	99.38 ± 0.06	85.22 ± 0.95	78.94 ± 0.45	69.13 ± 0.32	36.18 ± 0.49
10k	Quartiles	69.80 ± 0.93	**90.05 ± 0.25**	85.48 ± 0.32	99.38 ± 0.05	85.78 ± 0.78	78.97 ± 0.46	69.11 ± 0.32	36.06 ± 0.51
4D	Ab.Quart.	70.36 ± 0.65	90.03 ± 0.25	85.34 ± 0.31	**99.39 ± 0.06**	**86.29 ± 0.27**	**79.06 ± 0.42**	**69.18 ± 0.33**	35.90 ± 0.50
	Weighted	69.38 ± 0.96	90.02 ± 0.27	**85.54 ± 0.34**	99.38 ± 0.05	85.65 ± 0.81	78.92 ± 0.48	69.10 ± 0.32	36.10 ± 0.51
	Ab.Weigh.	**70.44 ± 0.65**	89.90 ± 0.29	85.32 ± 0.31	**99.39 ± 0.06**	86.16 ± 0.32	79.00 ± 0.42	69.17 ± 0.32	35.84 ± 0.49
	Default	64.26 ± 0.92	90.75 ± 0.28	86.38 ± 0.32	99.79 ± 0.02	83.74 ± 0.76	79.03 ± 0.29	67.14 ± 1.50	**66.08 ± 0.46**
	Mean	66.84 ± 0.47	90.90 ± 0.25	86.87 ± 0.26	99.79 ± 0.02	84.79 ± 0.95	79.47 ± 0.24	71.81 ± 0.25	65.56 ± 0.38
Abr.	Ab.Mean	67.15 ± 0.27	90.76 ± 0.37	87.00 ± 0.26	99.80 ± 0.02	84.97 ± 0.98	79.72 ± 0.22	71.97 ± 0.21	65.42 ± 0.41
50k	Quartiles	67.42 ± 0.13	90.88 ± 0.34	**87.04 ± 0.24**	99.80 ± 0.02	85.95 ± 0.30	79.83 ± 0.23	**72.01 ± 0.18**	65.21 ± 0.38
9D	Ab.Quart.	67.36 ± 0.14	90.45 ± 0.59	86.89 ± 0.24	**99.81 ± 0.02**	85.90 ± 0.34	**79.89 ± 0.23**	71.99 ± 0.20	64.99 ± 0.40
	Weighted	**67.47 ± 0.15**	**90.92 ± 0.27**	87.03 ± 0.22	99.80 ± 0.02	85.60 ± 0.69	79.78 ± 0.23	**72.01 ± 0.21**	65.31 ± 0.38
	Ab.Weigh.	67.28 ± 0.17	90.62 ± 0.42	86.79 ± 0.24	99.80 ± 0.02	**86.08 ± 0.21**	79.83 ± 0.22	71.89 ± 0.20	64.81 ± 0.38
	Default	70.76 ± 0.82	91.06 ± 0.50	87.02 ± 0.21	99.85 ± 0.01	85.09 ± 0.77	79.53 ± 0.16	72.08 ± 0.26	**38.41 ± 0.40**
	Mean	72.93 ± 0.41	91.24 ± 0.30	87.31 ± 0.18	99.85 ± 0.01	86.01 ± 0.97	79.73 ± 0.19	72.33 ± 0.35	38.16 ± 0.39
Abr.	Ab.Mean	73.33 ± 0.29	90.97 ± 0.68	87.36 ± 0.17	**99.86 ± 0.01**	84.83 ± 1.56	79.82 ± 0.17	72.55 ± 0.14	38.16 ± 0.37
50k	Quartiles	**73.53 ± 0.26**	**91.40 ± 0.12**	87.38 ± 0.16	**99.86 ± 0.01**	86.28 ± 1.06	79.92 ± 0.19	**72.58 ± 0.14**	38.05 ± 0.35
4D	Ab.Quart.	73.41 ± 0.29	91.19 ± 0.35	87.18 ± 0.17	**99.86 ± 0.01**	86.85 ± 0.33	**79.97 ± 0.16**	72.49 ± 0.19	37.86 ± 0.35
	Weighted	73.51 ± 0.28	91.18 ± 0.58	**87.39 ± 0.16**	**99.86 ± 0.01**	85.87 ± 1.28	79.86 ± 0.17	72.56 ± 0.14	38.04 ± 0.36
	Ab.Weigh.	73.47 ± 0.27	91.28 ± 0.15	87.14 ± 0.19	**99.86 ± 0.01**	**86.93 ± 0.21**	79.95 ± 0.16	72.50 ± 0.18	37.79 ± 0.36
	Default	72.36 ± 0.66	91.14 ± 0.58	87.40 ± 0.14	99.92 ± 0.01	85.00 ± 0.86	79.64 ± 0.17	72.82 ± 0.19	**39.06 ± 0.36**
	Mean	73.28 ± 0.59	91.33 ± 0.61	87.60 ± 0.16	99.92 ± 0.01	85.89 ± 0.69	79.82 ± 0.13	73.12 ± 0.12	39.00 ± 0.31
Abr.	Ab.Mean	73.87 ± 0.28	91.36 ± 0.30	87.72 ± 0.09	99.92 ± 0.01	85.65 ± 0.69	79.98 ± 0.15	73.14 ± 0.12	38.87 ± 0.29
100k	Quartiles	74.19 ± 0.11	91.56 ± 0.14	87.74 ± 0.11	99.92 ± 0.01	86.81 ± 0.28	80.10 ± 0.14	**73.17 ± 0.12**	38.82 ± 0.27
4D	Ab.Quart.	74.16 ± 0.10	91.50 ± 0.15	87.58 ± 0.12	**99.93 ± 0.01**	86.85 ± 0.33	**80.13 ± 0.16**	72.89 ± 0.24	38.57 ± 0.30
	Weighted	**74.22 ± 0.08**	**91.62 ± 0.12**	**87.76 ± 0.11**	99.92 ± 0.01	86.66 ± 0.42	80.01 ± 0.14	**73.17 ± 0.12**	38.85 ± 0.31
	Ab.Weigh.	74.15 ± 0.08	91.33 ± 0.35	87.52 ± 0.14	**99.93 ± 0.01**	**86.97 ± 0.20**	80.11 ± 0.12	72.80 ± 0.28	38.55 ± 0.28

bold and, when they are those of the *Sample* parameter set, the second best result is written in *italics*. The very best result for each dataset appears in italicized ***bold***.

Analyzing the results, we observe that almost all methods improved their accuracies when compared to the corresponding results using default parameters. In most datasets, the parameter set generated using the sample data achieved the best performance. And in many cases when this did not happen, i.e. the sample set did not deliver the best accuracy, the best result was obtained with one of the methods using a segmented version of the parameters, indicating the characteristics of the dataset.

6.1. Accuracy statistical analysis

As mentioned in Section 5.1, a statistic called F_F, based on the nonparametric Friedman test [18], was used to help in deciding the method recommendations. In this test, the null hypothesis states that all methods are statistically equal and, if it is rejected, the test indicates there is a statistical difference in one or more of the methods, but it does not specify in which. To ascertain which methods are statistically superior, a post-hoc test is required. We chose the Nemenyi-test, which compares each method against all the others and uses a critical difference (CD) as a reference.

The statistic and the test were applied several times in different sets of results. We present some of the results using graphics and, in these graphics, the CD is represented by bars and methods that are connected by the bars are *not* statistically different.

The first applications were to determine the best aggregation functions for each detector. We therefore carried out the calculations individually for each drift detection method. In the case of DDM using NB, the results of the tests based on the data presented in Table 4 are summarized in Fig. 1.

Note that, according to Fig. 1, the *Weighted* and the *Quartiles* parameter sets are *not* statistically different and are both significantly better than the other two sets. This was the reason why they were chosen as the suggested parameter sets for DDM using NB as base classifier. Observe that the *Mean* set of parameters was also significantly better than the default parameters of DDM.

Table 6
Accuracy results of the recommended methods in the real-world datasets with Naive Bayes.

Dataset	RDDM	DDM	HDDM$_A$	ECDD	Parameter set	RDDM	DDM	HDDM$_A$	ECDD	Dataset
Airlines	68.53	68.26	***68.64***	67.06	Sample	**87.67**	***87.92***	**87.80**	87.41	**Electricity**
	68.58	67.72	*68.23*	64.73	Default	85.09	82.58	85.58	**87.44**	
	68.58	68.44	65.25	65.14	Mean	83.95	84.25	87.23	86.12	
	68.56	68.45	67.50	**67.10**	Quartiles	84.73	85.10	86.61	85.18	
	68.60	68.49	63.46	65.53	Weighted	84.71	84.94	87.46	86.37	
	68.60	68.30	64.41	65.55	Abr. Mean	84.54	84.60	*87.65*	86.30	
	68.38	68.26	67.21	66.26	Abr. Quartiles	85.10	85.18	86.73	85.33	
	68.43	67.92	62.04	65.50	Abr. Weighted	*86.08*	*85.90*	87.19	86.09	
	68.62	68.30	65.52	64.86	Fast Mean	84.12	84.20	87.51	86.19	
	68.51	68.38	67.56	68.14	Fast Quartiles	84.73	85.01	86.51	83.56	
	68.62	68.39	64.57	64.97	Fast Weighted	84.56	84.56	87.17	85.75	
	68.61	68.55	65.49	66.02	Slow Mean	82.97	84.28	87.60	86.01	
	68.63	68.39	66.91	66.63	Slow Quartiles	83.71	84.53	86.81	85.20	
	68.64	**68.56**	65.03	66.92	Slow Weighted	82.50	84.68	87.25	86.24	
Covertype	**89.99**	***90.19***	89.67	**89.63**	Sample	***81.18***	**81.08**	**80.78**	**79.97**	**Pokerhand**
	86.16	86.61	86.58	*89.26*	Default	77.27	65.85	77.08	*79.80*	
	84.35	84.70	89.48	87.07	Mean	75.90	76.37	79.44	78.62	
	85.23	85.43	88.49	86.01	Quartiles	76.84	77.10	78.36	77.68	
	86.09	85.70	**89.80**	87.23	Weighted	76.43	77.07	80.14	78.68	
	85.05	85.43	89.71	87.12	Abr. Mean	76.51	76.72	79.70	78.66	
	85.51	85.53	88.60	86.57	Abr. Quartiles	77.17	77.38	78.23	78.09	
	87.35	*86.66*	**89.80**	87.43	Abr. Weighted	*77.75*	*78.07*	*80.54*	78.78	
	84.90	84.64	89.00	87.35	Fast Mean	76.11	76.12	79.23	78.88	
	85.97	85.15	88.26	83.50	Fast Quartiles	77.27	77.10	78.12	75.99	
	86.27	84.74	89.43	87.34	Fast Weighted	76.35	76.64	79.83	78.88	
	83.19	84.09	89.11	86.65	Slow Mean	74.94	75.90	79.12	78.03	
	83.92	85.27	88.63	85.87	Slow Quartiles	75.75	76.76	78.44	77.81	
	83.12	85.31	89.49	86.91	Slow Weighted	74.96	76.56	79.51	58.93	
Cov.Sorted	***67.96***	**67.58**	**67.82**	**67.89**	Sample	***92.72***	88.90	**89.88**	**43.03**	**Sensor**
	67.62	65.24	67.31	66.97	Default	88.36	87.49	*88.19*	35.11	
	66.99	67.34	67.17	66.87	Mean	89.15	89.31	87.12	41.86	
	67.50	66.96	67.22	67.28	Quartiles	89.12	89.23	87.30	42.24	
	67.41	67.25	66.62	66.79	Weighted	88.74	89.11	87.33	35.04	
	67.39	67.47	66.56	66.78	Abr. Mean	89.10	**89.43**	87.64	35.04	
	67.59	67.33	67.12	67.14	Abr. Quartiles	89.16	89.28	87.31	42.13	
	67.40	67.20	66.02	66.86	Abr. Weighted	88.49	89.19	85.95	35.04	
	67.24	65.26	67.05	66.85	Fast Mean	89.20	88.31	87.54	35.11	
	67.28	67.16	*67.47*	*67.59*	Fast Quartiles	88.76	88.17	87.26	*42.43*	
	66.91	67.20	66.76	66.84	Fast Weighted	87.45	88.94	87.36	41.92	
	66.78	67.23	67.27	66.80	Slow Mean	88.03	89.26	87.38	35.01	
	66.93	67.34	66.99	67.13	Slow Quartiles	*89.42*	89.09	88.04	42.20	
	66.30	*67.48*	67.17	67.05	Slow Weighted	87.85	89.04	87.16	27.30	

We have likewise used these results to compare all combinations of detectors and configurations to identify the best methods overall, considering their best configurations. A simplified view of the results using NB is presented in Fig. 2. All the best ranks come from parameter sets of the recommended methods and the best eight sets are statistically superior to the best sets of most other methods, the exceptions being HDDM$_W$ and FHDDM. This was the main reason for recommending RDDM, DDM, HDDM$_A$, and ECDD when the base classifier is NB.

Accordingly, we have made similar statistical evaluations with the artificial datasets separated in the three segments, based on the type of concept drift they have. In this round, we evaluated the segmented parameter sets together with the more general ones using each of the segments separately. In all three segments, at least the five best results were parameter sets of RDDM, DDM or HDDM$_A$. Specific details are omitted as they are little different from the ones already presented.

Other statistical comparisons were designed to reveal the most effective aggregation functions considering all detectors. Again, we carried out four rounds, one comparing the general versions of the functions using all the artificial datasets and the other three adding the corresponding segmented versions of the aggregation functions in the comparison using the data of each segment separately. The rank results of these four rounds are presented in Table 7 and the best result in each round

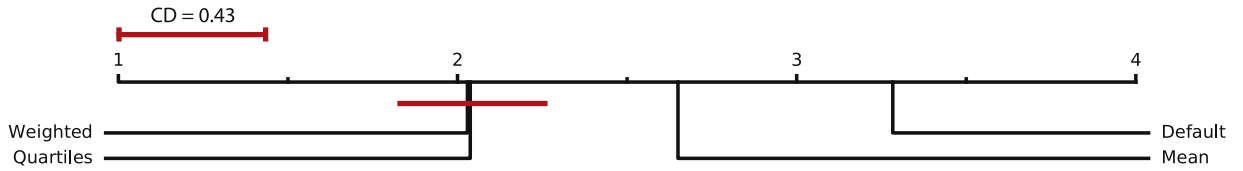

Fig. 1. Comparison of the parameterization sets of DDM + Naive Bayes using the Nemenyi post-hoc test with 95% confidence on artificial datasets.

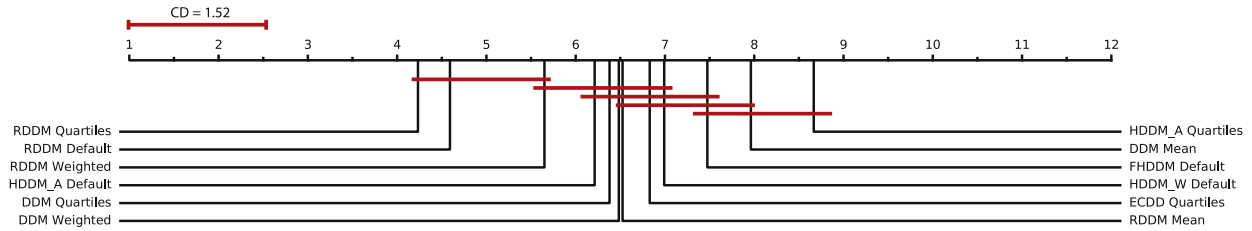

Fig. 2. Simplified results of the best methods + parameterization combinations with Naive Bayes using the Nemenyi post-hoc test with 95% confidence in the artificial datasets.

Table 7
Rank results of the aggregation functions considering the results of all detectors with Naive Bayes.

	Default	Mean	Quartiles	Weighted	Seg. Mean	Seg. Quart.	Seg. Weighted
General	2.11	2.86	**1.97**	3.06	N/A	N/A	N/A
Abrupt	3.39	4.53	**2.79**	4.87	4.42	2.97	5.03
Grad. Fast	3.12	4.67	3.16	5.09	4.32	**2.77**	4.87
Grad. Slow	3.28	4.48	**2.88**	4.98	4.52	3.04	4.83

(line) is presented in **bold**. Note that the last three columns of the table refer to the same functions but different parameter sets of values because they were calculated using different segments of the data.

Based on these results, we argue that *Quartiles* was the most effective aggregation function tested, in all its four tested versions, when NB is the base classifier.

6.2. Accuracy evaluation of the method (NB)

This subsection reuses the results of the experiments with the real-world datasets and NB to further evaluate the performance of the method in each of the scenarios considered.

Table 8 presents the accuracies that would be produced following the application of the method in all real-world datasets using NB as base learner, in each of the scenarios in question, and considering all the possible user decisions regarding knowledge of the type of concept drifts and number of methods to use.

It can be seen in the data of Table 8 that, in scenarios *S1* and *S2*, running more than two methods in the experiments using the *sample* makes no big difference. In most datasets, their results are exactly the same or very similar to those using just two methods. As anticipated, in most cases, the very best results were obtained in the first scenario, when a sample *is* available and there *is* enough time to search for a good parameterization for each method using the DE. Nevertheless, in most datasets, the results of scenarios *S2* and *S3* are often fairly close to those of *S1*.

6.3. Recommendations and comments on the results using HT

This subsection presents recommendations similar to those presented so far but using HT as base learner, instead of NB. It is worth highlighting that, in several situations, the recommendations are similar, reinforcing their applicability.

Initially, for S1 (Section 5), when *sample and time* are available, it is recommended to use the DE in the available sample together with experimentally preselected methods. Again, the methods that obtained the best performances with HT were RDDM, DDM, and HDDM$_A$. For the case in which a fourth method is needed, it also remained ECDD.

For S2, where sample *is* available but time is *limited*, it is recommended to test the preselected (best performing) methods in the available sample, but with predefined parameters. Naturally, as the experiments used another classifier and another round of testing, different parameters were found for the methods, as can be seen in Table 9.

Considering that the type of concept drift is not known, the recommended methods remained the same: RDDM, DDM, HDDM$_A$ and ECDD. On the other hand, when the type of drift is known, as in the case of slow gradual changes, HDDM$_A$ replaced ECDD. For the other two types of changes, the recommended methods remain the same.

Finally, for S3, when a sample is not available, the recommendation is to use the best methods found experimentally. If the type of change is not known, instead of RDDM with the quartiles parameter set, recommended for NB, the suggestion

Table 8
Accuracy results of the method application in the real-world datasets considering different scenarios and user choices with Naive Bayes.

Scenario	Method	Param	Air	Cover	CovSrt	Elect	Poker	Sens
S1	2 methods	Sample	68.26	90.19	67.96	87.92	81.18	92.72
	3 methods	Sample	68.64	90.19	67.96	87.92	81.18	92.72
	4 methods	Sample	68.64	90.19	67.96	87.92	81.18	92.72
S2 No Know.	2 methods	2 Param.	68.58	86.16	67.62	85.10	77.27	89.23
	3 methods	2 Param.	68.58	88.49	67.62	86.61	78.36	89.23
	4 methods	2 Param.	68.58	89.26	67.62	87.44	79.80	89.23
S2 Abrupt	2 methods	3 Param.	68.58	86.16	67.62	85.18	77.38	89.28
	3 methods	3 Param.	68.58	88.60	67.62	86.73	78.36	89.28
	4 methods	3 Param.	68.58	89.26	67.62	87.44	79.80	89.28
S2 Fast Grad.	2 methods	3 Param.	68.58	86.16	67.62	85.10	77.27	89.23
	3 methods	3 Param.	68.58	88.49	67.62	86.61	78.36	89.23
	4 methods	3 Param.	68.58	89.26	67.62	87.44	79.80	89.23
S2 Slow Grad.	2 methods	3 Param.	68.63	86.16	67.62	85.10	77.43	89.42
	3 methods	3 Param.	68.63	88.63	67.62	86.81	78.44	89.42
	4 methods	3 Param.	68.63	89.26	67.62	87.44	79.80	89.42
S3 No Know.	RDDM	Quartiles	68.56	85.23	67.50	84.73	76.84	89.12
S3 Abrupt	RDDM	Abr. Quart.	68.38	85.51	67.59	85.10	77.17	89.16
S3 Fast Grad.	ECDD	Fast Quart.	68.14	83.50	67.59	83.56	75.99	42.43
S3 Slow Grad.	RDDM	Slow Quart.	68.63	83.92	66.93	83.71	75.75	89.42
Best	—	—	68.64	90.19	67.96	87.92	81.18	92.72

Table 9
Best combinations of detectors and parameter sets in scenario S2 with Hoeffding Tree.

Detector	p1	p2	p3	p4	p5	p6	Config. name
	No prior knowledge of type of concept drift						
RDDM	n=129	α_w=1.773	α_d=2.258	*max*=40000	*min*=7000	*wLim*=1400	Default
RDDM	n=157	α_w=1.475	α_d=2.317	*max*=367902	*min*=6791	*wLim*=9016	Quartiles
DDM	n=174	α_w=1.640	α_d=2.184	N/A	N/A	N/A	Quartiles
DDM	n=168	α_w=1.823	α_d=2.287	N/A	N/A	N/A	Weighted
HDDM$_A$	d=0.010	w=0.324	t=0	N/A	N/A	N/A	Quartiles
HDDM$_A$	d=0.001	w=0.005	t=1	N/A	N/A	N/A	Default
ECDD	n=122	λ=0.045	N/A	N/A	N/A	N/A	Quartiles
ECDD	n=30	λ=0.20	N/A	N/A	N/A	N/A	Default
	Abrupt concept drift (extra set)						
RDDM	n=177	α_w=1.312	α_d=2.085	*max*=392923	*min*=5863	*wLim*=9587	Abrupt Quartiles
DDM	n=220	α_w=1.469	α_d=1.977	N/A	N/A	N/A	Abrupt Quartiles
ECDD	n=123	λ=0.043	N/A	N/A	N/A	N/A	Abrupt Quartiles
HDDM$_A$	d=0.009	w=0.452	t=0	N/A	N/A	N/A	Abrupt Quartiles
	Fast gradual concept drift (extra set)						
ECDD	n=132	λ=0.028	N/A	N/A	N/A	N/A	Fast Quartiles
RDDM	n=132	α_w=1.561	α_d=2.271	*max*=373070	*min*=6506	*wLim*=8674	Fast Quartiles
DDM	n=152	α_w=1.732	α_d=2.190	N/A	N/A	N/A	Fast Quartiles
HDDM$_A$	d=0.009	w=0.293	t=0	N/A	N/A	N/A	Fast Quartiles
	Slow gradual concept drift (extra set)						
DDM	n=149	α_w=1.757	α_d=2.453	N/A	N/A	N/A	Slow Quartiles
RDDM	n=163	α_w=1.611	α_d=2.766	*max*=340689	*min*=7982	*wLim*=8687	Slow Quartiles
FHDDM	n=613	δ=0.015	N/A	N/A	N/A	N/A	Slow Quartiles
HDDM$_A$	d=0.014	w=0.256	t=0	N/A	N/A	N/A	Slow Quartiles

for HT is to use the same method but with the default parameter set. If the type of change is known, the suggestions are the same in both abrupt and fast gradual drifts: RDDM and ECDD with abrupt and fast quartiles parameters, respectively. And in the case of slow gradual changes, rather than RDDM with slow quartiles parameters, DDM with the slow quartiles parameter set is recommended.

Comparing the performance of the parameterization sets, several similarities to NB can also be seen in the results of the experiments using HT. Using the general parameters, the best parameterization remains the one defined by quartiles, followed by the default parameters, mean and weighted.

Table 10
Rank results of the aggregation functions considering the results of all detectors with Hoeffding Tree.

	Default	Mean	Quartiles	Weighted	Seg. Mean	Seg. Quart.	Seg. Weighted
General	2.39	2.65	**2.04**	2.93	N/A	N/A	N/A
Abrupt	3.76	4.23	3.04	4.68	4.33	**2.86**	5.08
Grad. Fast	3.68	4.47	3.31	4.91	4.05	**3.09**	4.49
Grad. Slow	3.90	4.20	**3.03**	4.78	4.38	3.19	4.51

To conclude this subsection, in the case of the segmented parameterization sets, only a few differences occurred. In the three cases, the best sets were the segmented quartiles, the general quartiles, and default, with small variations in their relative order. Table 10 presents the results for HT and these can be compared to the ones in Table 7 (NB).

6.4. Critical evaluation of the drift detectors

In this section, we briefly summarize our conclusions on the effectiveness of the concept drift detection methods tested and most of them hold for both base learners tested. Firstly, DDM is still one of the best methods available, despite being the oldest in the group. However, its default parameters were apparently *not* chosen appropriately, especially $n = 30$, as all aggregate functions we applied on the results of the DE returned a much larger value for n.

In the case of EDDM, despite using the same $n = 30$ of DDM, the adoption of $e = 30$ makes it much more robust than DDM in indicating false alarms; in practice, the value of n rarely affects its results. In addition, we failed to confirm its authors' claim [4] that it is better than DDM in the detection of gradual concept drifts, even with default parameters.

Regarding ADWIN and SeqDrift2, despite their conceptual merits and apparently reasonable default parameters, their results were quite modest in our experiments. In the case of SeqDrift2, we also failed to confirm its authors' claim [34] that it presents better results than ADWIN.

Using their respective default parameters, ECDD, FHDDM, STEPD and WSTD also proved to be modest/intermediate methods, together with several others. The optimization of the parameters did *not* make much difference in this regard. However, in some situations, two of these methods, namely ECDD and FHDDM, performed well, leading to their being recommended in at least one scenario.

The results of both versions of HDDM were among the best in many scenarios, especially HDDM_A. Also, their default parameters seem to be well set. Nevertheless, we failed to confirm the claim of their authors [20] that HDDM_A was better for abrupt changes and HDDM_W for gradual changes. In our tests, the results of both versions were mixed with respect to the type of concept drift and HDDM_A was better than HDDM_W in more datasets.

Finally, RDDM was the best method in general. Its performance was very consistent, always being among the best results in the experiments. In terms of parameterization, its default set of values seems to be well defined, being the best option with HT, followed by the quartiles parameter set. With NB, the results were the opposite, with quartiles being the best option, followed by the default configuration. Thus either option is suitable for general use.

To conclude, we remind the reader that all these observations were based on the results of our experiments and depend on the chosen datasets: there is no guarantee that they will hold for every dataset.

7. Conclusion

This article presents a method aimed at choosing a concept drift detection method and appropriate parameters in order to learn from real data streams. The method is based on a Differential Evolution algorithm and on extensive empirical experiments carried out with two base classifiers (Naive Bayes and Hoeffding Tree), 11 different drift detectors, 120 artificial dataset versions, containing abrupt, slow gradual, and fast gradual concept drifts, as well as six real-world datasets. In our experiments, the DE was responsible for optimizing the parameter values of the drift detection methods.

The method presented here sets out to guide the specialist's decisions taking into account different scenarios. Each of the possibilities presented here is based on the results of the aforementioned experimental evaluations. Note that the proposed method can also be used to optimize the parameters of other drift detectors and/or other classifiers, with minor adaptations in the DE.

The results of our experiments indicate that predictive accuracies (used as fitness in this work) increased in most datasets, regardless of the drift detection method used. In most situations, RDDM, DDM and HDDM_A presented the best results, followed by one configuration of ECDD. Thus, these are our recommended concept drift detectors with both base learners tested. Regarding the tested aggregation functions, the experiments have shown that, in general, Quartiles was the most efficient choice.

No time analysis was performed, owing mainly to the large number of experiments. It is well known that evolutionary approaches need long training times, but our prescribed use of the DE is restricted to one execution for each method–dataset combination. Nevertheless, the run-times (in minutes per 1000 instances) of the DE to find parameters for the drift detectors in the datasets used in the experiments were recorded and presented in the text. Note that the most time-consuming use of the DE does not need to be re-executed unless the user wants to experiment with other artificial datasets, aggregation

functions, drift detection methods, and/or base learners. In addition, a balance between accuracy and evaluation time could also be analyzed by the DE, but such analysis is beyond of the scope of this paper.

Future work includes using other aggregation functions with the aim of improving the overall performance of the drift detectors; adapting the DE to optimize the parameters of ensemble classifiers; and modifying the current implementation of the DE to test new heuristics, for example, by using other convergence, selection and mutation criteria, aimed at improving its performance. 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 datasets with gradual concept drifts.

Finally, it is worth pointing out that the source code for the latest version of our DE will eventually be freely available at https://sites.google.com/site/moamethods/, where the previous GA is already available.

Acknowledgment

Silas Santos is a Ph.D. student supported by CNPq postgraduate grant number 141196/2015-7.

References

[1] R. Agrawal, T. Imielinski, A. Swami, Database mining: a performance perspective, IEEE Trans. Knowl. Data Eng. 5 (1993) 914–925.
[2] F. Arce, E. Zamora, H. Sossa, R. Barrón, Differential evolution training algorithm for dendrite morphological neural networks, Appl. Soft Comput. 68 (2018) 303–313.
[3] S.H. Bach, M.A. Maloof, Paired learners for concept drift, in: Eighth IEEE International Conference on Data Mining (ICDM'08), 2008, pp. 23–32. Pisa, Italy
[4] M. Baena-García, J. Del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavaldà, R. Morales-Bueno, Early drift detection method, in: Fourth International Workshop on Knowledge Discovery from Data Streams (IWKDDS'06), 2006, pp. 77–86.
[5] R.S.M. Barros, Advances in data stream mining with concept drift, Centro de Informática, Universidade Federal de Pernambuco, Brazil, 2017 Professorship (Full) Thesis.
[6] R.S.M. Barros, D.R.L. Cabral, P.M. Gonçalves Jr., S.G.T.C. Santos, RDDM: reactive drift detection method, Expert Syst. Appl. 90 (2017) 344–355.
[7] R.S.M. Barros, J.I.G. Hidalgo, D.R.L. Cabral, Wilcoxon rank sum test drift detector, Neurocomputing 275 (2018) 1954–1963.
[8] R.S.M. Barros, S.G.T.C. Santos, A large-scale comparison of concept drift detectors, Inf. Sci. (NY) 451–452 (2018) 348–370.
[9] 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), 2016, pp. 1871–1878. Vancouver, Canada
[10] S. Bernstein, The Theory of Probabilities, Gastehizdat Publishing House, Moscow, 1946.
[11] 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), 2007, pp. 443–448. Minneapolis, MN, USA
[12] A. Bifet, G. Holmes, R. Kirkby, B. Pfahringer, MOA: massive online analysis, J. Mach. Learn. Res. 11 (2010) 1601–1604.
[13] A. Bifet, G. Holmes, B. Pfahringer, Leveraging bagging for evolving data streams, in: Machine Learning and Knowledge Discovery in Databases, in: LNCS, 6321, Springer, 2010, pp. 135–150.
[14] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, R. Gavaldà, New ensemble methods for evolving data streams, in: Proceedings 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'09), 2009, pp. 139–148. Paris, France
[15] A. Bifet, J. Read, I. Žliobaitė, B. Pfahringer, G. Holmes, Pitfalls in benchmarking data stream classification and how to avoid them, in: Mach. Learn. Knowledge Discovery in Datab. (Part I), in: LNCS, 8188, Springer, 2013, pp. 465–479.
[16] D.R.L. Cabral, R.S.M. Barros, Concept drift detection based on Fisher's exact test, Inf. Sci. (NY) 442–443 (2018) 220–234.
[17] A.P. Dawid, Present position and potential developments: some personal views: statistical theory: the prequential approach, J. R. Stat. Soc. Ser. A 147 (1984) 278–292.
[18] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[19] A.E. Eiben, J.E. Smith, Introduction to Evolutionary Computing, second ed., Springer Publishing Company, Incorporated, 2015.
[20] I. Frias-Blanco, J. del Campo-Avila, G. Ramos-Jimenez, R. Morales-Bueno, A. Ortiz-Diaz, Y. Caballero-Mota, Online and non-parametric drift detection methods based on Hoeffding's bounds, IEEE Trans. Knowl. Data Eng. 27 (2015) 810–823.
[21] F. Friedrichs, C. Igel, Evolutionary tuning of multiple SVM parameters, Neurocomputing 64 (2005) 107–117.
[22] J. Gama, P. Medas, G. Castillo, P. Rodrigues, Learning with drift detection, in: Advances in Artificial Intelligence: SBIA 2004, in: LNCS, 3171, Springer, 2004, pp. 286–295.
[23] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation, ACM Comput. Surv. 46 (2014) 1–37.
[24] D. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, 1989.
[25] P.M. Gonçalves Jr., R.S.M. Barros, RCD: a recurring concept drift framework, Pattern Recognit. Lett. 34 (2013) 1018–1025.
[26] 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 (2014) 8144–8156.
[27] W. Hoeffding, Probability inequalities for sums of bounded random variables, J. Am. Stat. Assoc. 58 (1963) 13–30.
[28] D. Ienco, A. Bifet, I. Žliobaite, B. Pfahringer, Clustering based active learning for evolving data streams, in: Discovery Science, in: LNCS, 8140, Springer, 2013, pp. 79–93.
[29] J.Z. Kolter, M.A. Maloof, Dynamic weighted majority: an ensemble method for drifting concepts, J. Mach. Learn. Res. 8 (2007) 2755–2790.
[30] A.C. Lorena, A.C.P.L.F. Carvalho, Evolutionary tuning of SVM parameter values in multiclass problems, Neurocomputing 71 (2008) 3326–3334.
[31] B.I.F. Maciel, S.G.T.C. Santos, R.S.M. Barros, A lightweight concept drift detection ensemble, in: Proceedings of 27th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'15), 2015, pp. 1061–1068. Vietri sul Mare, Italy
[32] L.L. Minku, X. Yao, DDD: a new ensemble approach for dealing with concept drift, IEEE Trans. Knowl. Data Eng. 24 (2012) 619–633.
[33] K. Nishida, K. Yamauchi, Detecting concept drift using statistical testing, in: Proceedings of the 10th International Conference on Discovery Science (DS'07), in: LNCS, 4755, Springer, 2007, pp. 264–269.
[34] R. Pears, S. Sakthithasan, Y.S. Koh, Detecting concept change in dynamic data streams, Mach. Learn. 97 (2014) 259–293.
[35] A. Pesaranghader, H. Viktor, Fast Hoeffding drift detection method for evolving data streams, in: Machine Learning and Knowledge Discovery in Databases, in: LNCS, 9852, Springer, 2016, pp. 96–111.
[36] S. Roberts, Control chart tests based on geometric moving averages, Technometrics 1 (1959) 239–250.
[37] G. Ross, N. Adams, D. Tasoulis, D. Hand, Exponentially weighted moving average charts for detecting concept drift, Pattern Recognit. Lett. 33 (2012) 191–198.
[38] S. Sakthithasan, R. Pears, Y. Koh, One pass concept change detection for data streams, in: Advances in Knowledge Discovery and Data Mining, in: LNCS, 7819, Springer, 2013, pp. 461–472.
[39] 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 27th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'15), 2015, pp. 1077–1084. Vietri sul Mare, Italy
[40] S.G.T.C. Santos, P.M. Gonçalves Jr., G. Silva, R.S.M. Barros, Speeding up recovery from concept drifts, in: Machine Learning and Knowledge Discovery in Databases, in: LNCS, 8726, Springer, 2014, pp. 179–194.

[41] J. Schlimmer, R. Granger, Incremental learning from noisy data, Mach. Learn. 1 (1986) 317–354.
[42] S. Soares, C. Antunes, R. Araújo, Comparison of a genetic algorithm and simulated annealing for automatic neural network ensemble development, Neurocomputing 121 (2013) 498–511.
[43] R. Storn, K. Price, Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (1997) 341–359.
[44] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics Bull. 1 (1945) 80–83.
[45] F. Yates, Contingency table involving small numbers and the χ^2 test, J. R. Stat. Soc. Suppl. 1 (1934) 217–235.
[46] X. Zhu, Stream data mining repository, 2010, Online. URL: http://www.cse.fau.edu/~xqzhu/stream.html.