Expert Systems With Applications 60 (2016) 39–50

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Review

Using dynamical systems tools to detect concept drift in data streams

F.G. da Costa[a,*], R.A. Rios[b], R.F. de Mello[a]

[a] Institute of Mathematics and Computer Science, Universidade de São Paulo, São Carlos, 13566-590, Brazil
[b] Department of Computer Science, Universidade Federal da Bahia, Salvador, 40170-110, Brazil

ARTICLE INFO

Article history:
Received 28 September 2015
Revised 19 April 2016
Accepted 20 April 2016
Available online 21 April 2016

MSC:
68Q32
68P20
68Wxx
65P20

Keywords:
Concept drift
Data streams
Dynamical systems
Chaos
Unsupervised learning

ABSTRACT

Real-world data streams may change their behaviors along time, what is referred to as concept drift. By detecting those changes, researchers obtain relevant information about the phenomena that produced such streams (e.g. temperatures in a region, bacteria population, disease occurrence, etc.). Many concept drift detection algorithms consider supervised or semi-supervised approaches which tend to be unfeasible when data is collected at high frequencies, due to the difficulties involved in labeling. Complementarily, current studies usually assume data as statistically independent and identically distributed, disregarding any temporal relationship among observations and, consequently, risking the quality of data modeling. In order to tackle both aspects, we employ dynamical system modeling to represent the temporal relationships among data observations and how they modify along time in attempt to detect concept drift. This approach considers Taken's immersion theorem to unfold consecutive windows of data observations into the phase space in attempt to represent and compare time dependencies. From this perspective, we proposed four new concept drift detection algorithms based on the unsupervised machine learning paradigm. The first algorithm builds dendrograms of consecutive phase spaces (every phase space represents the time relationships for the observations contained in a particular data window) and compare them out by using the Gromov–Hausdorff distance, providing enough guarantees to detect concept drifts. The second algorithm employs the Cross Recurrence Plot and the Recurrence Quantification Analysis to detect relevant changes in consecutive phase spaces and warn about relevant data modifications. We also preprocess data windows by considering the Empirical Mode Decomposition method and Mutual Information in attempt to take only the deterministic stream behavior into account. All algorithms were implemented as plugins for the Massive Online Analysis (MOA) software and then compared to well-known algorithms from literature. Results confirm the proposed algorithms were capable of detecting most of the behavior changes, creating few false alarms.

1. Introduction

Different application domains, such as climatology, network monitoring, health, and stock-market analysis, are characterized by the continuous production of large amounts of data in the form of streams, i.e., an open-ended sequence of observations whose behavior changes over time (Guha, Mishra, Motwani, & O'Callaghan, 2000). The application of traditional data mining approaches to model and extract information from such applications is prohibitive due the lack of resources to process and fully store data in main memory or even in secondary devices for later analysis (Babcock, Babu, Datar, Motwani, & Widom, 2002; Guha et al., 2000; O'callaghan, Meyerson, Motwani, Mishra, & Guha, 2002). This issue has motivated several studies to design algorithms to induce learning models by analyzing a data stream as its observations are collected. Such algorithms read observations only once and create a data summary to represent all relevant information, modeling the data stream at the current time instant (Aggarwal, Han, Wang, & Yu, 2003). Afterwards, such observations are discarded, consuming the least possible amount of main memory (Babcock et al., 2002; Guha et al., 2000). As new observations are collected, learning models are updated.

The comparison between past and current learning models makes possible to point out changes in data behavior along time. While small perturbations are seen as data instability, significant changes correspond to modifications in the phenomenon which produces the stream. The detection of these changes is the main motivation for concept drift area. Concept drift is the term used to define modifications in data streams along time, which are characterized by abrupt, usually easier to be detected, or by small changes (Tsymbal, 2004).

* Corresponding author.
E-mail addresses: fausto@icmc.usp.br, fausto.guzzo@gmail.com (F.G. da Costa), ricardoar@ufba.br (R.A. Rios), mello@icmc.usp.br (R.F. de Mello).

http://dx.doi.org/10.1016/j.eswa.2016.04.026
0957-4174/

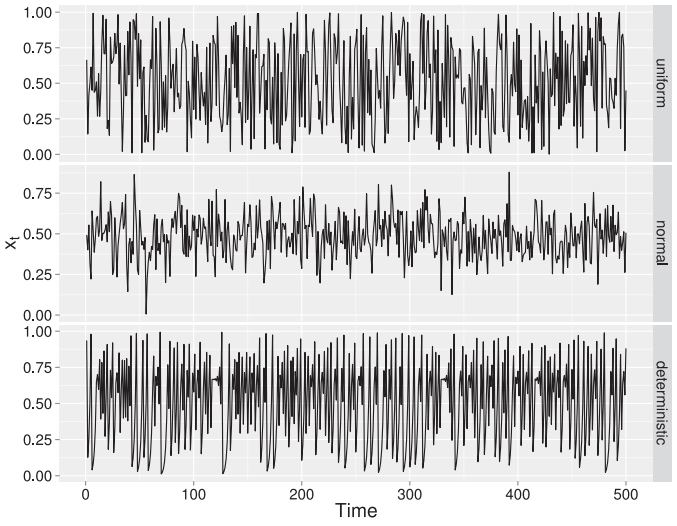

Fig. 1. Synthetic data streams produced using a Uniform distribution, a Normal distribution and a deterministic process.

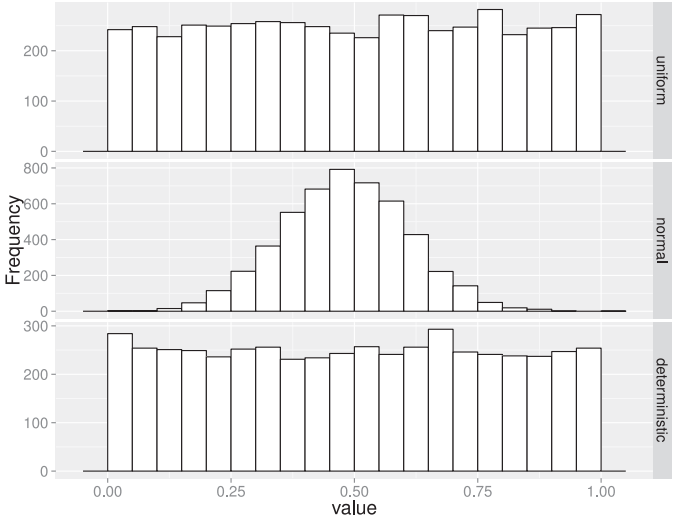

Fig. 2. Histograms produced using the data streams illustrated in Fig. 1. Observe it is possible to distinguish the Uniform from the Normal distribution, but it is not possible to separate it from the deterministic process.

Most of current studies assume that a data stream concept is composed of a set of statistically independent observations, which are typically modeled using linear and stationary processes such as the Normal and the Uniform distributions (Gama, Medas, Castillo, & Rodrigues, 2004; Klinkenberg & Joachims, 2000). Those studies rely on the estimation of probability distributions for consecutive windows of data, then they compare such estimations to point out changes (Bifet & Gavalda, 2007). We believe this characterization of a data stream concept misses the fact that data are produced along time and observations may contain dependencies which, once revealed, could better support the modeling of the stream behavior. As an example, consider three data stream concepts produced using different processes: (i) a Uniform distribution; (ii) a Normal distribution; and (iii) the deterministic process defined in Eq. 1

$$s_n = \frac{\arccos(-x_n)}{\pi}, \text{ in which } x_n = 1 - 2x_{n-1}^2. \quad (1)$$

Fig. 1 illustrates the concepts produced by all three processes, while Fig. 2 shows the correspondent histograms used to estimate probability distributions. By analyzing the histograms, one can observe the Uniform and the Normal processes indeed characterize

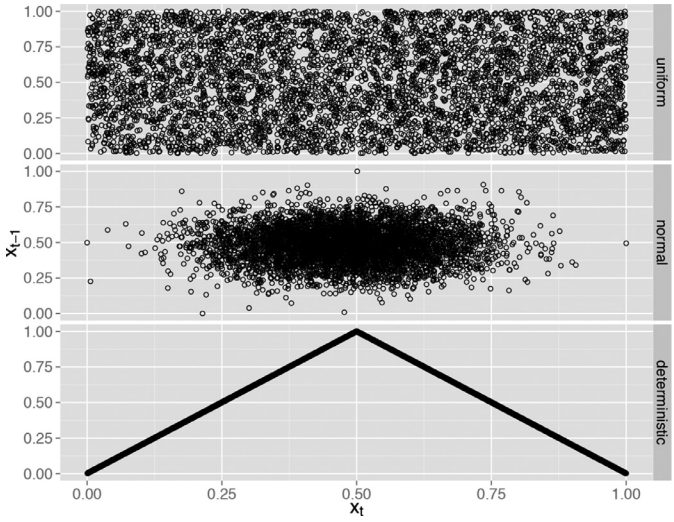

Fig. 3. Phase spaces reconstructed using the data streams illustrated in Fig. 1. Observe that all phase spaces are very different from each other, what allows us to point out three different concepts.

different concepts, however the deterministic process is quite similar to the Uniform one, making difficult to separate them out. In fact, this is among the most commonly considered approaches to tackle the problem of concept drift detection (Bifet & Gavalda, 2007).

Aiming at overcoming this issue, we present a new approach that considers dynamical system tools. As a first step, we apply Takens' immersion theorem (Takens, 1981) to reconstruct data stream windows into a multidimensional space called phase space, making possible to observe differences as shown in Fig. 3. As one can observe, the phase space for the Uniform process spreads points all over the range, the Normal one accumulates more points around the center, while the deterministic process produces a characteristic function, commonly referred to as attractor (Alligood, Sauer, & Yorke, 1997). Thus, in this paper we consider phase spaces to model concepts and compare them out.

In addition to consider data observations as statistically independent, many of the studies employ either supervised or semi-supervised machine learning algorithms to detect concept drift (Aggarwal et al., 2003; Gama et al., 2004), i.e., they assume that the observations (or part of them) are labeled by specialists. We believe this assumption is hard to employ in many practical scenarios, once data can be produced at high frequencies or there is lack of information about the phenomenon from which the stream is generated. As a result, we decided to take only the unsupervised learning paradigm into consideration while developing this study.

We tackle this problem of concept drift detection in a different manner. First of all, we reconstruct data streams into a multidimensional space, also referred to as phase space (Alligood et al., 1997), in order to better understand the temporal relationships among observations. Then, we compare how points are distributed in such phase space using two different approaches: (i) the Permutation-Invariant Clustering Algorithm proposed by Carlsson and Mémoli (2010) and (ii) the Cross-Recurrence Plot (CRP) in conjunction with the Recurrence Quantification Analysis (RQA) measure Q_{max}, proposed by Serra, Serra, and Andrzejak (2009).

The first approach considers the Gromov–Hausdorff distance to compare ultrametric spaces, providing theoretical guarantees of representing the dissimilarities between models. The second approach allows the comparison between the dynamics of phase spaces, i.e., how the trajectories formed by two consecutive data

windows of a stream move around those spaces, allowing to point out how similar two dynamical systems are along time. Moreover, we also employed the EMD-MI method proposed by Rios and Mello (2016), which is based on the Empirical Mode Decomposition (EMD) and Mutual Information (MI), to extract the deterministic component contained in data streams in attempt to detect concept drift based only on the recurrent patterns. This decomposition step was taken into account due to the improvements it provides for concept drift detection as shown in (da Costa & de Mello, 2014).

We implemented four algorithms based on the two approaches we mentioned. The first takes the original data stream as input, reconstructs phase spaces and employs the Permutation-Invariant Clustering Algorithm in conjunction with the Gromov–Hausdorff distance. The second works similarly but employs the CRP and RQA to quantify changes in phase spaces and map them to concept drifts. The third is only a variation of the first, in which the original data stream is previously decomposed using the EMD-MI method (Rios & Mello, 2016) and only its deterministic component is used as input. Similarly, the fourth algorithm is a variation of the second, in which the original data stream is also decomposed and only its deterministic part is taken into account.

All four algorithms were implemented as plugins for the Massive Online Analysis (MOA) software (Bifet, Holmes, Kirkby, & Pfahringer, 2010; Bifet, Read, Pfahringer, Holmes, & Žliobaitė, 2013) and then compared to three traditional algorithms from literature: ADWIN (Bifet & Gavalda, 2007), CUSUM (Page, 1954), and Page-Hinkley (Page, 1954). Nine synthetically produced data streams, considering four different deterministic concepts (Logistic map, Lorenz system, Sine function and Rössler) mixed with four different stochastic ones (Uniform distribution, Normal distribution, Autoregressive model and Autoregressive model with Integration). Thus, the concepts used to compose data streams are defined according to a sum of different levels of deterministic and stochastic behaviors, making possible to represent more complex scenarios. Synthetic data was used due to we know what to expect from it, allowing us to validate our proposal.

Three metrics were used to assess the proposed algorithms: Missed Detection Rate (MDR), Mean Time to Detection (MTD) and Mean Time between False Alarms (MTFA) (Basseville, Nikiforov et al., 1993). Results confirm the proposed algorithms were capable of detecting most of the behavior changes, creating few false alarms.

This paper is organized as follows: the next section introduces data streams, concept drift and dynamical systems, as well describe the main issues and our approach; Section 3 presents the three algorithms from literature as well as the four new algorithms we proposed; Section 4 details the experimental setup; and Section 5 report the results and discusses them; finally, we draw conclusions and present future work in Section 6.

2. The dynamical system bias towards concept drift

Studies on concept drift detection can be classified in two different approaches: i) based on the data distribution; and ii) based on data clustering. The first approach models a data stream using probability distributions in order to point out divergences and, finally, identifies concept drift. The second one uses clustering models and considers different measures, such as Entropy, to detect changes according to modifications in data partitions.

The analysis of data distributions is usually performed assuming streams are composed of independent and identically distributed observations, processing them by chunks or windows (Baena-García et al., 2006; Gama et al., 2004; Klinkenberg & Joachims, 2000). This processing involves the estimation of distribution parameters (e.g. the mean and the standard deviation for the Normal distribution), which are used to compare data windows. When estimated parameters of consecutive windows present enough modification, a concept drift is issued. In this situation, a concept is defined by considering the parameters of a predefined (a priori) distribution on data. In our point of view, this misses the fact that data present temporal dependencies, what causes a huge impact in modeling as illustrated in Section 1.

On the other hand, studies related to the second approach consider the temporal relationship among data observations. However, clustering approaches lack in theoretical foundation to ensure confidence levels when detecting concept drift (Albertini & de Mello, 2007; Cao, Ester, Qian, & Zhou, 2006; Vallim, Andrade Filho, De Mello, & De Carvalho, 2013). Aiming at overcoming this issue, there are significant theoretical advances on stability for data clustering which could be applied in the context of data streams (Carlsson & Mémoli, 2010; Vallim & De Mello, 2014).

In this paper, we claim that concept drift detection must be characterized by temporal relationships, therefore it can be improved by analyzing observation dependencies in multidimensional phase spaces (Alligood et al., 1997; Kantz and Schreiber, 2004; Whitney, 1936). To better explain the phase space, consider the time series produced using a Logistic map (Fig. 4a). Now consider this time series as data stream collected along time. In this scenario, the reconstruction of a data stream into the phase space is accomplished by applying Takens' immersion theorem (Takens, 1981) which unfolds a time series $X = \{x_0, x_1, \ldots, x_n\}$ and maps it into states $\mathbf{x}_i$ in the multidimensional phase space as follows $\mathbf{x}_i = (x_i, x_{i-\tau}, \ldots, x_{i-(m-1)\cdot\tau})$, having τ as the time delay dimension and m as the embedding dimension. In phase space, every state corresponds to time series observations that depend on each other, i.e., in this situation observations $x_i, x_{i-\tau}, \ldots, x_{i-(m-1)\cdot\tau}$ are related. When all the states in phase space are reconstructed, the time series behavior is fully unfolded and it can be modeled, studied and better understood.

Aiming at reconstructing a data stream into the phase space, it is necessary to estimate the time delay and embedding dimensions. To estimate the time delay dimension τ, the pairwise correlations of τ-delayed observations may be analyzed. This analysis can be conducted by means of the Auto-Mutual Information, as proposed by Fraser and Swinney (1986), and selecting the first minima.

The embedding dimension m is traditionally estimated by using the False Nearest Neighbors (FNN) method (Kennel, Brown, & Abarbanel, 1992). FNN suggests to adopt the value for m which first provides the fraction of false nearest neighbors lower than 10%. The first value is considered to avoid unnecessary dimensions in phase space. If one sets the embedding dimension as greater than the optimal m_{min}, there is no mathematical issue, however the phase space will be more complex than necessary. As consequence, the processing time will be more demanding. On the other hand, if $m < m_{min}$, the reconstructed phase space will not unfold all temporal dependencies and will most probably be inappropriate.

When this reconstruction is successful, we obtain a well-formed attractor as illustrated in Fig. 4b, otherwise we will have a point cloud, as illustrated in Fig. 4bc, due to incorrect dimension estimations.

Therefore, the phase space is an important mathematical representation to study and understand the time relations among data observations, allowing to select adequate models to represent the stream. In this paper, data streams are taken as time series and they are, initially, segmented into consecutive windows. Then, every window is reconstructed in phase space considering two different ways: i) directly using the window observations; or ii) after decomposing the data stream window.

The decomposition step is performed using the EMD-MI method, proposed by Rios and Mello (2016); Rios, Parrott, Lange, and de Mello (2015), which employs the Empirical Mode

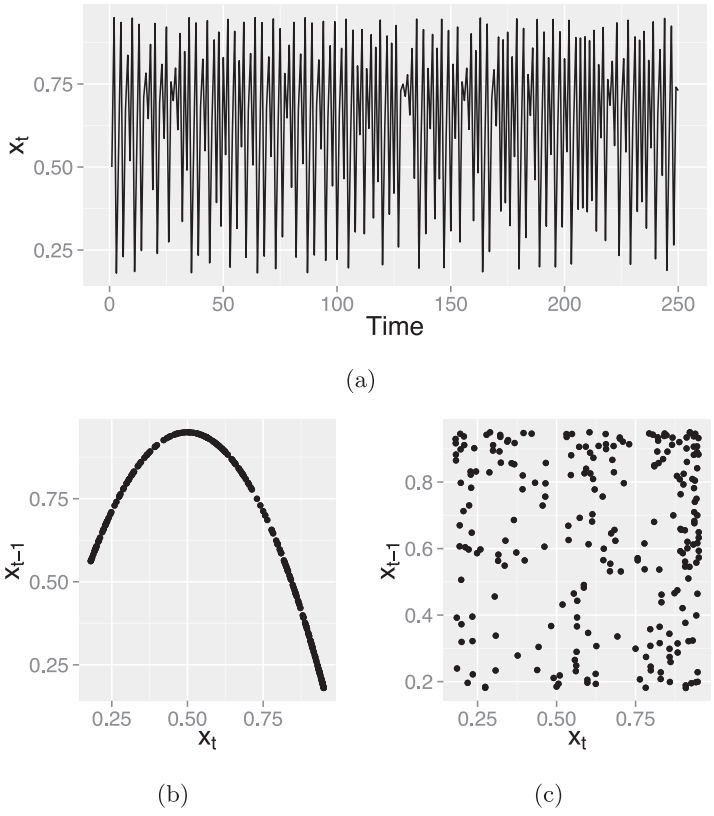

Fig. 4. (a) A time series produced using the Logistic Map, and its reconstructed phase spaces: (b) well reconstructed, with $m = 2$ and $\tau = 1$; and (c) poorly reconstructed , with $m = 2$ and $\tau = 10$.

Decomposition (EMD) in conjunction of Mutual Information (MI) into a time series to separate it into two time series – one containing the deterministic bias and another the stochastic bias. As confirmed by previous results (da Costa & de Mello, 2014), the deterministic component obtained after this decomposition is useful to detect concept drift in data streams. Those results motivated us to apply our approach either directly on stream windows or on deterministic components extracted from such data windows.

In our scenario, as previously stated, we consider a data stream window as a time series. After buffering a stream window, we conducted experiments with and without the EMD-MI to separate the deterministic component present in data (we will discuss on results with and without such decomposition step). Then, we apply Takens' immersion theorem to reconstruct either the raw stream window or its deterministic component into the phase space. As next step, we compare this phase space against the one obtained for the previous data window. When significant changes occur, we point out a concept drift.

2.1. Proposed approach

The modeling of data streams is nowadays one of the greatest challenges in the machine learning area, due to its open-ended nature and the possibility of streams to be produced specially at high frequencies. As main consequence, algorithms must process such information using data windows to estimate models and support the understanding of the underlying phenomenon responsible for producing the stream. In this context, a major concern is how to point out relevant changes that may occur in data streams along time, what motivates the area of concept drift detection. In fact, those changes should warn systems and users about real modifications in real-world phenomenon.

In our point of view, most of studies miss the fact that concept drift detection should consider possible data dependencies along time and that data is not necessarily linear and stationary (Gama et al., 2004; Klinkenberg & Joachims, 2000). This has motivated us to tackle the concept drift detection problem using dynamical system tools. Our approach is based on the modeling of consecutive data stream windows using Takens's immersion theorem, which is responsible for producing a multidimensional space called phase space (Takens, 1981). This space is used to represent time relationships among data observations.

In our approach, a data window is unfolded in the phase space, which is taken as the model for the current data stream concept. Consecutive models are then compared in order to identify concept drifts. There are two main advantages of our approach: i) the immersion in phase space allows to represent time dependencies among data; ii) Takens' immersion theorem supports the modeling of nonlinear and non-stationary processes (rather than simply linear and stationary ones as typically proposed in literature (Gama et al., 2004; Klinkenberg & Joachims, 2000)).

We proceed with the modeling, i.e., the reconstruction in phase space, of a data stream window in two different ways: i) by decomposing it using the EMD-MI method (Rios & Mello, 2016; Rios et al., 2015), or ii) without any decomposition process. The decomposition process separates the deterministic from the stochastic component present in data. Then, only the deterministic component is considered to be reconstructed in phase space and compared along time, once it expresses the time dependencies among observations. In this manner, we assume the data stochasticity as noise or simply uninteresting (what may depend on the target scenario). As a complementary strategy, we also support the data modeling without this decomposition stage, however, in that circumstance, all information contained in the data window will be

mapped to the phase space, including the stochastic component which may compromise our models.

After obtaining models for consecutive data windows, we compare them using two different approaches: i) the stable clustering method, in conjunction with the Gromov-Hausdorff metric, as proposed by Carlsson and Mémoli (2010); and ii) the Cross-Recurrence Plot (CRP) together with the measures provided by the Recurrence Quantification Analysis (RQA) (Trulla, Giuliani, Zbilut, & Webber, 1996).

The first comparison approach is based on the study by Carlsson and Mémoli (2010), who assessed the stability of clustering algorithms in terms of changes in the input set. According to authors, clustering algorithms should produce the same results when the order of the input set is changed, i.e., they should be permutation invariant. In the lack of stable algorithms, they proposed a modification in the Single-Linkage clustering algorithm (SL) to make it permutation invariant. This new SL version, here referred to as SSL (Stable Single-Linkage algorithm), always produces the same dendrogram when the order of input data is modified.

As main result, Carlsson and Mémoli (2010) show that dendrograms produced by SSL could be compared using the Gromov-Hausdorff distance to point out relevant changes in input data. Therefore, one can use this approach to compare two clustering models to identify data modifications. However, this approach is only applicable on statistically independent data, once it is designed to produce the same clustering model independently of the input order (by changing the order we break any possible time relationship among observations). This is a problem in our scenario, once we consider data streams as containing some dependency among observations. However, there is a solution for this issue. When we reconstruct a data window in phase space, all time dependencies among observations are unfolded into states, so every state only depends on itself (Alligood et al., 1997), making possible to apply Carlsson and Mémoli (2010) approach directly on phase spaces.

It is important to note Carlsson and Mémoli (2010) provided a mathematical framework for measuring the differences between two sets of data, however they did not implemented nor evaluated their algorithm empirically.

In the second approach, we compare consecutive data windows using the Cross-Recurrence Plot (CRP) and the Recurrence Quantification Analysis (RQA). CRP allows to compare the similarities between phase spaces. It builds a binary matrix in which row indices correspond to states in phase space 1 and column indices are states in phase space 2, so when the states of different phase spaces evolve similarly along time, cells of this matrix are marked down with 1s, otherwise 0s. Afterwards, RQA is computed on top of this binary matrix to provide recurrence measures. In this paper, we only consider the measure Q_{max} proposed by Serra et al. (2009), which summarizes the longest diagonal line in such a matrix. The line length corresponds to the longest time period that both phase spaces evolved through similar trajectories. When $Q_{max} \approx 0$, no similarity is found, what is mapped as concept drift. When $Q_{max} \approx n$ (having n as the number of states in phase space), both phase spaces evolve quite similarly along time, indicating no concept drift.

Serra et al. (2009) used this technique for measuring the similarities between a set of static time series, which were observed from music signals. In our studies, we are using such technique into a new context, of detecting concept drift on data streams.

Our intention when considering Carlsson and Mémoli (2010) results is to provide stability guarantees when detecting concept drift. Complementarily, the use of CRP and RQA is much more connected to Takens' immersion theorem and the area of dynamical systems, because they allow to compare two attractors in phase spaces, pointing out similarities and dissimilarities along time. In fact our objective is to tackle the concept drift problem in two different and complementary points of view. Both views consider unsupervised learning, so there is no need for data labeling to proceed with concept drift detection, what we believe to be more adequate for high-frequency data streams. Such type of stream produces too much data along time, making labeling unfeasible.

3. Algorithms

In this section, we describe all the seven algorithms evaluated in this study. First, we present the three well-known algorithms from literature to monitor and detect changes in sequential data. Afterwards, we present the four algorithms based on dynamical system tools which were proposed in this work.

3.1. Algorithms from literature

3.1.1. Cumulative sum

CUSUM is one of the most well-known algorithms for monitoring and detecting changes (Page, 1954). Differently from other methods, which compute the mean and the variance of the most recent data window, CUSUM employs the cumulative sum of observations x_i, $i = 1, \ldots, t$. For each observation x_t, the algorithm computes S_t as defined in Eq. 2. If S_t exceeds some threshold h, a detection is issued. Its computational complexity is $\mathcal{O}(n)$ for processing n new observations

$$\begin{aligned} S_0 &= 0, \\ S_t &= \max(0, S_{t-1} + x_t) \end{aligned} \tag{2}$$

3.1.2. Page-Hinkley test

Several monitoring and change detection applications employ the Page-Hinkley Test (PHT). This statistical test detects modifications in the mean of a signal, considering it was produced by a Gaussian distribution. For every observation x_t, the PHT algorithm calculates a cumulative variable m_t as shown in Eq. (3). This variable computes the differences between observation x_i and mean $\bar{x}_t$. Also, at every iteration, this algorithm computes M_t, which is the smallest value for m_i (having $i = 1, 2, \ldots, t$), and the difference between m_t and M_t. If such difference is greater than a user-defined threshold ϵ_{PHT}, a change is detected. If so, values of m_t and M_t are set to zero. Its computational complexity is $\mathcal{O}(n)$ for processing n new observations

$$\begin{aligned} m_t &= \sum_{i=1}^{t} (x_i - \bar{x}_t - \delta_{PHT}) \\ \bar{x}_t &= \frac{1}{N} \sum_{i=1} t x_i \\ M_t &= \min(m_i), i = 1, 2, \ldots, t \end{aligned} \tag{3}$$

3.1.3. Adaptive windowing

ADWIN (Bifet & Gavalda, 2007) is an algorithm which maintains a window W with the n most recent observations of the data stream under analysis. After collecting a next observation x_t, window W is subdivided in two, W_1 with length n_1 and W_2 with length n_2, and their means are computed, i.e., μ_{W_1} and μ_{W_2}. If the difference between these means is greater than a user-defined threshold ϵ, a change is detected and window W_1 is dropped from W. Its computational complexity is $\mathcal{O}(\log n_W)$, in which n_W is the size of the window.

3.2. Proposed algorithms

In this paper, we propose four new algorithms for detecting concept drift in data streams. The first one, called Stable Cluster-

ing Concept Drift Detection (SCCDD), is based on the framework proposed by Carlsson and Mémoli (2010) for computing the divergence between hierarchical clusterings. The second, called Cross Recurrence Concept Drift Detection (CRCDD), is mainly motivated by the results obtained by Serra et al. (2009) when computing the similarity between cover songs. These two algorithms are based on comparing the phase spaces produced by successive windows of data stream observations.

The other two algorithms, SCCDDd (Stable Clustering Concept Drift Detection with Decomposition) and CRCDDd (Cross Recurrence Concept Drift Detection with Decomposition), are variations of the previous ones. These approaches preprocess every data stream window using the EMD-MI method proposed by Rios and Mello (2016); Rios et al. (2015), in order to extract the deterministic component contained in data. In this context, EMD-MI is used as a filter to remove stochastic influences, what we believe could improve the reconstruction of phase spaces. Details about the proposed algorithms are presented next.

3.2.1. SCCDD – Stable clustering concept drift detection

This first proposed algorithm maintains a window with the most recent data stream observations. Such window is considered as a time series $\mathbf{W}_t$. As the first step, this series is reconstructed in the phase space by using Taken's immersion theorem (Takens, 1981), enabling one to represent the dynamics of such time series. After estimating the embedding (m) and the time delay τ dimensions (more details in Section 2), a state in phase space is reconstructed as follows: $\mathbf{w}_i = (w_i, w_{i+\tau}, \ldots, w_{i+(m-1)\tau}), i = 0, \ldots, N$.

As next step, phase space $\mathbf{W}_t$ is clustered by the stable clustering algorithm proposed by Carlsson and Mémoli (2010). This clustering algorithm produces a tree-based structure called proximity dendrogram – in which the root corresponds to a partition with a single group, leafs represent clusters of single points, and the intermediate nodes are inner subtrees. The distance between one cluster and another is known. The algorithm works as follows: i) first, the pairwise distances among all points are computed; then, ii) pairs of points are selected according to their minimum distances; next, iii) the selected points are clustered together. The algorithm iterates back to step (i) until a single cluster is found. Note that at step (iii) every point that has the minimum distance is clustered at the same time, preventing that the order of the points alter the produced dendrogram, as detailed in Algorithm 1. The dendrogram obtained after this clustering strategy is represented as $\Theta_{\mathbf{W}_t}$.

Algorithm 1: Stable Single-Linkage algorithm.

```
Data: A vector of points W = {W_t}, W_t ∈ R^m
Data: A distance ϵ ∈ R
Result: A vector of distances D = {d_i}, d_i ∈ R
D ← ∅;
while size(W) > 1 do
    M ← dist(W) ;    /* creates a matrix of distances */
    d ← min(M) +ϵ ;    /* selects the minimum distance
    between points */
    add d to D;
    Q ← which(M ≤ d) ;    /* select matrix cells which
    have distance less than d */
    foreach point q_{i,j} in Q do
        c ← mean(W_i, W_j) ;    /* create the new cluster */
        remove the point W_i from W;
        remove the point W_j from W;
        add the point c to W;
    end
end
```

Then, the algorithm waits for new data stream observations. Let us call this new window as $\mathbf{W}_{t+1}$. By applying the same process on $\mathbf{W}_{t+1}$ such as on $\mathbf{W}_t$, we will have a new dendrogram, referred to as $\Theta_{\mathbf{W}_{t+1}}$. These two dendrograms are then compared by the Gromov-Hausdorff distance. Let $\mathbf{D}_1$ be the distance between clusters on the dendrogram $\Theta_{\mathbf{W}_t}$, and $\mathbf{D}_2$ of the dendrograms $\Theta_{\mathbf{W}_{t+1}}$, i.e. $\mathbf{D}_1$ and $\mathbf{D}_2$ are vectors of distances. The Gromov-Hausdorff distance between these vectors is $\alpha_t = \max(\mathbf{D}_1) - \max(\mathbf{D}_2)$. This value is then stored in a time series $\alpha = \{\alpha_0, \ldots, \alpha_t\}$ and the concept drift is issued after analyzing such series.

Before presenting the remaining proposed methods, it is important to provide more details about the framework proposed by Carlsson and Mémoli (2010), which is used in this work to produce dendrograms and compare them to detect concept drift.

Despite of the great number of studies devoted to the formalization of classification algorithms, such as the Statistical Learning Theory (SLT) (Vapnik & Chervonenkis, 1971), few other studies are concerned with the theoretical foundations of clustering methods. Knowing this gap, Carlsson and Mémoli (2010) focused on the formalization of a stable hierarchical clustering (HC) algorithm. Such algorithm models a finite metric space (X, d) into a dendrogram, which represents a hierarchical decomposition of X – i.e., a nested family of partitions.

Carlsson and Mémoli (2010) focused their study on the traditional agglomerative HC algorithm called Single-Linkage (Carlsson & Mémoli, 2010). Such algorithm merges a pair of data points at each step, until all points belong to the same cluster, thus producing a dendrogram. The selection of pairs to be merged is based on the minimum distance of points. The authors stressed that the traditional algorithm does not specifically explain how to deal with when more than a pair of points has the same minimum distance. However, implementations commonly take the first pair and merge them together, leading to an undesirable feature – different dendrograms are produced depending on the order of points.

Then, Carlsson and Mémoli (2010) developed a variation of the Single-Linkage algorithm, in which all pairs of data points with the same minimum distance (or even with under some small variation ϵ) are merged at the same step. In this way, the ordering of the points does not influence anymore when building the dendrogram. Such algorithm is referred to as Permutation Invariant Single-Linkage or, in this paper, as Stable Single-Linkage algorithm (SSL).

In order to formalize the proposed algorithm, Carlsson and Mémoli (2010) used the well-known representation of dendrogram as ultrametrics. In such way, they could use the Gromov-Hausdorff distance to compute the dissimilarity between ultrametrics, i.e., two dendrograms, and point out divergences.

As this algorithm should create a dendrogram for each window by the Permutation Invariant Single-Linkage algorithm, its computational complexity is $\mathcal{O}(n_W^2)$, in which n_W is the size of the window.

3.2.2. CRCDD – Cross recurrence concept drift detection

In the second proposed algorithm, two data windows are maintained – the two most recent ones, represented by $\mathbf{W}_t$ and $\mathbf{W}_{t+1}$. As first step, the phase spaces are reconstructed using the same dimensions for both windows, producing $\mathbf{W}_t$ and $\mathbf{W}_{t+1}$, respectively. These phase spaces are then processed by the Cross Recurrence Plot (CRP) method, which produces a matrix of recurrence **CR** (for more information about Recurrence Plots, read the paper Marwan, Romano, Thiel, & Kurths (2007)). Essentially, this binary matrix represents the pairwise similarity of states contained in the phase spaces of both data windows. Each cell $\mathbf{CR}_{i,j}$ informs whether state $\mathbf{W}_{t,i}$ and $\mathbf{W}_{t+1,j}$ are neighbors, what depends on a radius ε_i around both. If state $\mathbf{W}_{t,i}$ contains $\mathbf{W}_{t+1,j}$ as a ε-neighbor, then cell $\mathbf{CR}_{i,j}$ is set to 1, otherwise $\mathbf{CR}_{i,j} = 0$.

One can plot matrix **CR** (using black and white dots) to interpret recurrent patterns and understand how those two phase spaces evolve along time. To support this human interpretation, several measures were developed to produce quantitative information (Zbilut, Giuliani, & Webber, 1998; Zbilut & Webber, 1992). In this work, we use the Q_{max} measure (Serra et al., 2009) for computing the similarity between phase spaces. Q_{max} computes the longest diagonal in matrix **CR**, even considering curved diagonal lines with some disruption.

Serra et al. (2009) applied CRP and RQA on audio signals in order to detect cover songs. They constructed CRP matrices for pair of songs, then computed the Q_{max} measure to quantify the similarity of trajectories in phase spaces. The Q_{max} is based on the RQA measure max diagonal line L_{max}, which is obtained after building the cumulative matrix L for $i = 2, \ldots, N_x$ and $j = 2, \ldots, N_y$

$$L_{i,j} = \begin{cases} L_{i-1,j-1} + 1 & \text{if } R_{i,j} = 1, \\ 0 & \text{if } R_{i,j} = 0, \end{cases} \tag{4}$$

having the first row and column set equal to zero, i.e., cells $L_{1,j} = 0,\ \forall j$ and $L_{i,1} = 0,\ \forall i$. After obtaining this cumulative matrix L, the L_{max} measure is found in form: $L_{max} = \max(L_{i,j}),\ \forall i, j$.

As cover songs present differences in their tempo, produced by musicians to impose their own signatures, the CRP matrix usually contain curved diagonal lines. Therefore, Serra et al. (2009) formalized a new measure to quantify the length of curved traces, which is an adaptation of Eq. (4). First, matrix S is built and the following cells are set as $S_{1,j} = S_{2,j} = S_{i,1} = S_{i,2} = 0$ for $i = 1, \ldots, N_x$ and $j = 1, \ldots, N_y$. Then, this new measure recursively computes

$$S_{i,j} = \begin{cases} \max(S_{i-1,j-1}, S_{i-2,j-1}, S_{i-1,j-2}) + 1 & \text{if } R_{i,j} = 1, \\ 0 & \text{if } R_{i,j} = 0, \end{cases} \tag{5}$$

for $i = 3, \ldots, N_x$ and $j = 3, \ldots, N_y$. Finally, S_{max} is defined as $S_{max} = \max(S_{i,j})$.

Besides tempo differences, musicians may also change or even skip some chords when performing cover songs, introducing short disruptions in diagonal lines. Serra et al. (2009) then extended Eq. (5) by computing the cumulative matrix Q. As first step, the following cells are set as $Q_{1,j} = Q_{2,j} = Q_{i,1} = Q_{i,2} = 0$ for $i = 1, \ldots, N_x$ and $j = 1, \ldots, N_y$, and then they recursively compute

$$Q_{i,j} = \begin{cases} \max(Q_{i-1,j-1}, Q_{i-2,j-1}, & \\ \quad Q_{i-1,j-2}) + 1 & \text{if } R_{i,j} = 1, \\ \max(0, Q_{i-1,j-1} - \gamma(R_{i-1,j-1}), & \\ \quad Q_{i-2,j-1} - \gamma(R_{i-2,j-1}), & \text{if } R_{i,j} = 0, \\ \quad Q_{i-1,j-2} - \gamma(R_{i-1,j-2})) & \end{cases}$$

for $i = 3, \ldots, N_x$ and $j = 3, \ldots, N_y$, in which the following penalty function for disrupted diagonal lines is used:

$$\gamma(z) = \begin{cases} \gamma_0, & \text{if } z = 1, \\ \gamma_e, & \text{if } z = 0. \end{cases}$$

Finally, $Q_{max} = \max(Q_{i,j})$ for $i = 1, \ldots, N_x$ and $j = 1, \ldots, N_y$. This new measure is used by Serra et al. (2009) to analyze the similarity between songs, pointing out when they are covers. In this paper we use Q_{max} measure to quantify the similarity between the phase spaces found for consecutive data stream windows.

As this algorithm should compute the matrix of recurrence for every window, its computational complexity is $\mathcal{O}(n_W^2)$, in which n_W is the size of the window.

3.2.3. SCCDDd and CRCDDd – Stable clustering concept drift detection with decomposition and cross recurrence concept drift detection with decomposition

In order to add some robustness on scenarios with noisy data streams, algorithms SCCDD and CRCDD were extended by adding a preprocessing stage responsible for decomposing data window into deterministic and stochastic components. This decomposition is performed by using the EMD-MI method proposed by Rios and Mello (2016); Rios et al. (2015). The SCCDDd algorithm takes the deterministic component of a data stream window to produce dendrograms, while CRCDDd constructs the CRP matrix only using this same deterministic influence. The motivation for using the deterministic component is because it contains temporal dependencies among data observations which may allow to point out relevant stream modifications.

Those two algorithm extensions rely on the Empirical Mode Decomposition (EMD) and on the Mutual Information (MI) methods, which are introduced next.

3.3. Empirical mode decomposition and mutual information

Data streams produced by real-world phenomena may contain some level of noise, which could be introduced by the collection equipment or by another unknown source. In this scenario, Rios and Mello (2016); Rios et al. (2015) considered the usage of the Empirical Mode Decomposition (EMD) (Huang et al., 1998) to separate the stochastic from the deterministic components contained in time series. First, they employ EMD on a time series $X = \{x_1, \ldots, x_N\}$ to produce k monocomponents (they are also time series with the same length as X, i.e., $|X| = N$) $h_1, \ldots, h_k$, which are referred to as Intrinsic Mode Functions (IMFs), plus a residual series $R = \{r_1, \ldots, r_N\}$. By adding the observations at the same time indices for all monocomponents and residue, one can obtain the original time series as follows:

$$x_j = \sum_{i=1}^{i=k} h_{i,j} + r_j,\ j = 1, \ldots, N.$$

After this decomposition, Rios and Mello (2016); Rios et al. (2015) attempts to separate the stochastic from the deterministic IMFs, by comparing the phases (produced by the Fourier Transform) of consecutive IMFs in terms of their Mutual Information (MI) (Darbellay, Vajda et al., 1999). All MI for successive IMFs are stored in a vector v, in which $v_p = MI(\Phi(h_p), \Phi(h_{p+1})),\ p = 1, \ldots, k-1$ and having $\Phi(.)$ as the function responsible for extracting the phases of a series using the Fourier coefficients. Next, Rios and Mello (2016); Rios et al. (2015) analyze vector v to find the greatest peak that defines the cutoff point l, i.e., the position that separates the stochastic from the deterministic IMFs. Finally, the first l IMFs are summed up to form the stochastic time series $S_j = \sum_{i=1}^{i=l} h_{i,j}$ and the remaining IMFs plus the residual series are summed up to form the deterministic series $D_j = \sum_{i=l+1}^{i=k} h_{i,j} + R_j$. The computational complexity of the EMD-MI method is $\mathcal{O}(n \cdot \log(n))$ (Bernhard, Darbellay et al., 1999; Wang, Yeh, Young, Hu, & Lo, 2014).

4. Design of experiments

In this section, we describe the experiments conducted to evaluate the seven algorithms considered in this work. First of all it is worth mentioning that we took into account the concept drift detection on data streams containing nonlinear and chaotic behaviors. Each data stream was generated containing four deterministic concepts according to the sine function, the Lorenz System, the Rössler attractor and the Logistic map. They were parameterized as defined in Table 1. Every concept contains 5000 observations and was normalized in range [0, 1]. Such concepts were connected by using the sigmoid function defined in Eq. (6), in which t is the instant time, s is the smoothing parameter (set to 0.01 in the experiments) and t_c is the time of change. Then, we produced

Table 1
The deterministic concepts and the parameterization used to produce the synthetic data streams.

Data stream generator	Parameters
Sine function	Amplitude equals to 1, period equals to 100, phase equals to 0
Lorenz system	$\sigma = 10, \beta = 8/3, \rho = -8/3$
Rössler attractor	$A = 0.15, b = 0.2, c = 10$
Logistic map	$R = 3.8, x_0 = 0.5$

Table 2
The stochastic concepts and the parameterization used to produce noise for the synthetic data streams.

Noise generator	Parameters
Normal distribution	mean = 0, standard deviation = 0.1
Uniform distribution	min = 0, max = 1
AR(1)	$\alpha = 0.7$
ARIMA(1, 1, 0)	$\alpha = 0.7$

Table 3
The synthetic data streams and their characteristics.

Data stream	Type of noise	Balance factor
ds1	None	0.0
ds2	Normal distribution	0.1
ds3	Normal distribution	0.2
ds4	Uniform distribution	0.1
ds5	Uniform distribution	0.2
ds6	AR(1) model	0.1
ds7	AR(1) model	0.2
ds8	ARIMA(1, 1, 0) model	0.1
ds9	ARIMA(1, 1, 0) model	0.2

Table 4
The algorithms and their parameters used in experiments: WL – window length; WO – window overlapping; m – embedding dimension; τ – time delay dimension; NV – number of variances; and ϵ – defines a neighborhood for the hierarchical clustering algorithm proposed by (Carlsson & Mémoli, 2010).

Algorithm	Parameters
CUSUM	MinNumInstances = 30; $\delta = 0.005$; $\lambda = 50$
PHT	MinNumInstances = 30; $\delta = 0.005$; $\lambda = 50$; $\alpha = 1$
ADWIN	$\delta = 0.002$
CM	WL = 1000; WO = 0%; $\epsilon = 0.005$; $m = 5$; $\tau = 3$; NV= 2;
CRQ	WL = 1000; WO = 0%; $m = 5$; $\tau = 3$; NV= 2;
EMD-MI+CM	WL = 1000; WO = 0%; $\epsilon = 0.005$; $m = 5$; $\tau = 3$; NV= 2;
EMD-MI+CRQ	WL = 1000; WO = 0%; $m = 5$; $\tau = 3$; NV= 2;

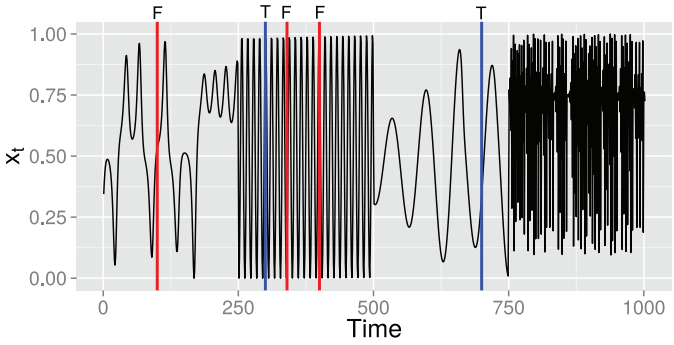

Fig. 5. Example of alarms detected for a data stream blue lines (with "T" caption on top) indicate true alarms while red lines (with "F" caption on top), false alarms. Note that the fourth concept was not detected, so it counts as a true alarm that was missed. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

data streams with 20, 000 observations, with four concepts changing smoothly over time. The sigmoid function we employed was also considered in related work (Bifet & Kirkby, 2009; Pereira & de Mello, 2014)

$$S(t, s, t_c) = \frac{1}{1 + e^{-s(t-t_c)}} \tag{6}$$

In our experiments, we analyzed synthetic data streams because they provide a controlled scenario in which the expected behavior is *a priori* known, i.e., we know the correct answer and, consequently, we can better compare all concept drift algorithms.

In order to make the data streams more realistic, we added different noise types at different levels. Four types of noise were generated using: the Normal distribution, the Uniform distribution, the AR and the ARIMA models. The parameters used to produce noise are listed in Table 2. The motivation for using those types of noise is that the Normal and the Uniform distributions produce data without any time dependency, whereas the AR and ARIMA models generate noises with some dependency. Therefore, we can evaluate more possibilities by considering those models.

Noises were then added to data streams in the following way: first, noisy time series with the same length of data streams were produced using the four models presented in Table 2. Afterwards, noisy series were normalized in range [0, 1] and then multiplied by a balance factor α, having $0 \geq \alpha \geq 1$. Complementarily, data streams were multiplied by $1 - \alpha$, then both were added. In this manner, balance factor α was used to define the amount of stochasticity (noise) contained in every data stream. Experiments considered two values for the balance factor $\alpha = \{0.1, 0.2\}$ to produce nine data streams: eight under noisy influences and one noise-free stream, as illustrated in Table 3.

Next, the produced data streams were processed by the seven algorithms described in Section 3. The parameters used by them are listed in Table 4. Parameters WL, WO, NV and ϵ were estimated using the Monte Carlo method. Parameters m and τ were estimated using the False Nearest Neighbors (Kennel et al., 1992) and the Auto-Mutual Information (Fraser & Swinney, 1986) techniques, respectively. Parameters MinNumInstances, α, δ and λ kept stable during the Monte Carlo simulation due to they did not affected results for the evaluated data streams. An extended discussion of the proposed algorithms' parameterization is fulfilled on latter part of Section 5.1.

4.1. Evaluation of results

Results were assessed using three metrics related to true and false alarms. As every synthetic data stream was composed by four consecutive concepts, three true alarms were expected. An example of the detection of alarms in a data stream is shown in Fig. 5. The blue vertical lines show true alarms, as they were detected after the occurrence of a concept drift. On the other hand, the red vertical lines show false alarms, as they were identified when no drift happened since last alarm. Moreover, in the last concept, a true alarm was missed, i.e., no alarm was triggered. On top of those alarms we compute three metrics: i) Missed Detection Rate (MDR), which calculates the number of true alarms that were missed (this is as percentage, in which the lesser it is, the better it is); ii) Mean Time to Detection (MTD), which computes the average time to detect true alarms (the lesser is better); iii) Mean Time between False Alarms (MTFA), which represents the average time between the detection of false alarms (the greater is better) (Basseville et al., 1993).

The four proposed algorithms were developed using the R Statistical Software (R Core Team, 2015) and experiments were executed on the Massive Online Analysis (MOA) software (Bifet et al., 2010) with the help of the Concept Drift-MOA (CD-MOA) framework (Bifet et al., 2013). MOA was considered because it is well-known and commonly used in literature as a tool for conducting

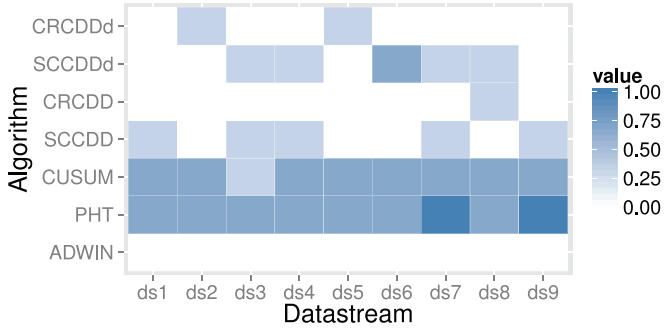

Fig. 6. Missed Detection Rate (MDR) – the less the value is, the better is the result. Note that ADWIN has detected every true alarm, followed by both CRCDD and CRCDDd, which missed one or two true alarms. The algorithms SCCDD and SCCDDd lost one true alarm in half of the data streams under evaluation. Algorithms CUSUM and PHT missed between 2/3 to 3/3 of true alarms, indicating they were not capable of identifying when the concept drift happened.

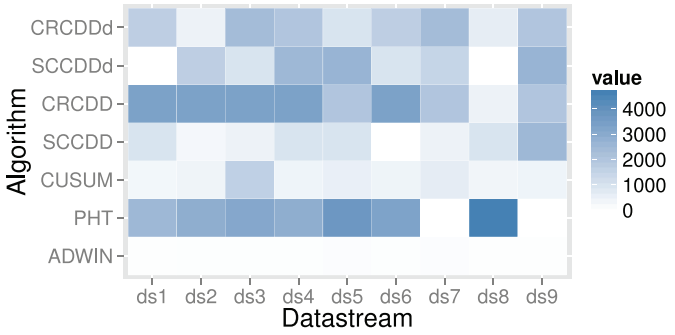

Fig. 7. Mean Time to Detection (MTD) – the less the value is, the better is the result. Again, ADWIN provided the best result, detecting true alarms very quickly, followed by CUSUM. Next, SCCDD obtained satisfactory results. Both proposed algorithms with preprocessing stage did not provide good results. Algorithms CRCDD and PHT provided the worst results.

experiments on data streams (Bifet et al., 2013; Elwell & Polikar, 2011).

5. Results

In this section, we present all experimental results, according to the scenarios described in Section 4. Some of the results are plotted as heat maps to improve visualization, in which the darker the color is, the greater the value is. The heat map shown in Fig. 6 illustrates the results of the metric Missed Detection Rate (MDR). To recall, as all data streams used have four consecutive concepts, three alarms were expected for MDR. In this case, the white color represents 0% of missed alarms, i.e. the best value, and the blue represents 100% of missed alarms. The *x*-axis corresponds data streams and *y* to the algorithms. We observe that ADWIN and the four proposed algorithms produced the best results, for which the values of MDR were between 0% and 33%. ADWIN obtained 0% of missed detection in all data streams. PHT and CUSUM missed 2/3 of the alarms in almost all data streams. SCCDD and SCCDDd missed between 0 and 1/3 of the alarms. Finally, CRCDD and CRCDDd missed almost no alarm.

The heat map in Fig. 7 shows the results in terms of the metric Mean Time to Detection (MTD), in which light colors correspond to best results (they mean short time delays to detect true alarms). To complement the illustration of results, we decided to use boxplots for MTD as showed in Fig. 8, supporting the observation of delay distribution per algorithm. We see that ADWIN provided the best results – it detected concept drifts with time delays between 43 and 110 observations. CUSUM detected concept drifts around 380

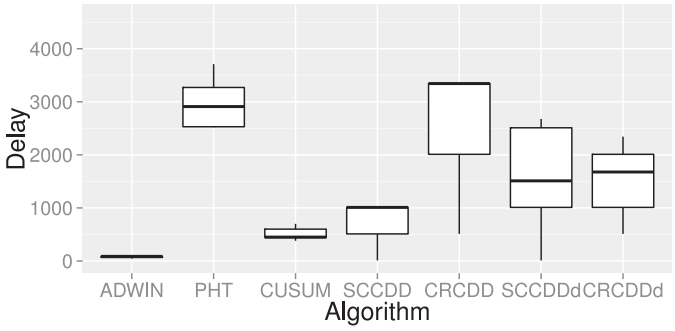

Fig. 8. Boxplot of Mean Time to Detection (MTD) for each algorithm on all data streams. The best result was obtained by ADWIN, followed by CUSUM and SCCDD.

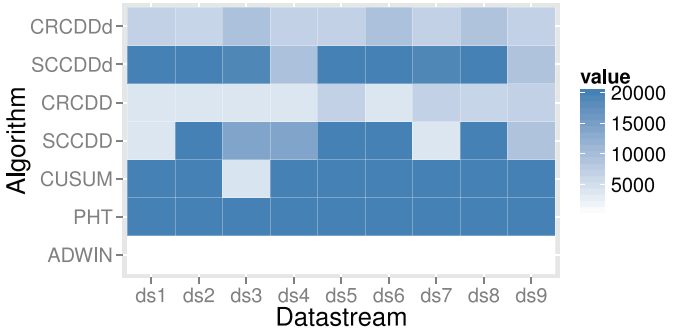

Fig. 9. Mean Time between False Alarms (MTFA) – the greater the value is, the better is the result. According to this metric, ADWIN provided the worst results at all, indicating it issues alarms constantly, true as well as false alarms (it is too way sensitive to data variations, what is negative). PHT and CUSUM detected few false alarms, as shown in Fig. 6. SCCDDd detected few false alarms, followed by SCCDD. Both algorithms based on CRCDD provided more incidences of false alarms.

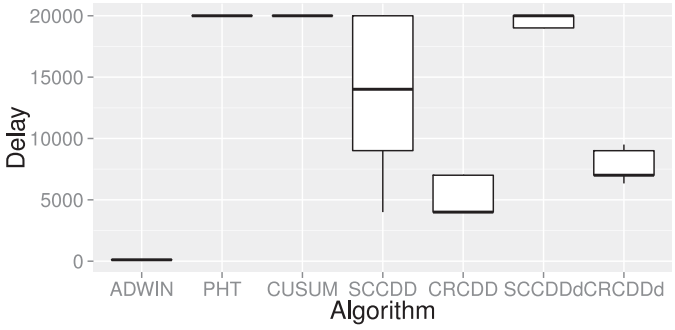

Fig. 10. Boxplot of Mean Time between False Alarms (MTFA) for each algorithm on all data streams. The best result were obtained by PHT, CUSUM and SCCDDd. The worst result was produced by ADWIN.

and 700 observations later, exceptionally, in one case (data stream 3, i.e., ds3), it detected the drift only after 1, 630 observations. Algorithm SCCDD detected concept drifts with an average time delay of 880 observations and the algorithm SCCDDd detected with the average of 1454. CRCDD identified concept drifts with the average time delay of 2584 while CRCDDd with the average of 1, 584. Finally, PHT detected concept drifts with time delays between 2530 and 3710, exceptionally in two cases (ds7 and ds9) in which it detected instantaneously.

The third and last heat map, shown in Fig. 9, is related to the metric Mean Time between False Alarms (MTFA), in which darker colors mean best results (they are associated to large time delays between the detection of false alarms). For better understanding, we also used boxplots in this situation as shown in Fig. 10. ADWIN obtained the worst result at all, detecting false alarms very often, almost at every 110 observations, confirming it is very sen-

Table 5
Average of MDR, $MTFA^{-1}$ and MTD for all evaluated algorithms – normalized according to the lengths of data streams. Note ADWIN, PHT and CUSUM provided very unbalanced results. On the other hand, the proposed algorithms provided more equalized results, except CRCDD.

	MDR	$MTFA^{-1}$	MTD
ADWIN	0.000	1.000	0.000
PHT	1.000	0.000	0.989
CUSUM	0.850	0.002	0.213
SCCDD	0.250	0.006	0.319
CRCDD	0.050	0.018	1.000
SCCDDd	0.300	0.001	0.548
CRCDDd	0.100	0.009	0.600

Table 6
Index obtained after providing the same importance level for each one of the metrics shown in Table 5. SCCDD provided the best result at all, followed by CRCDDd and SCCDDd. Remember small values mean better results.

	Index
ADWIN	1.000
PHT	1.406
CUSUM	0.876
SCCDD	0.405
CRCDD	1.001
SCCDDd	0.625
CRCDDd	0.609

sitive to data variations. PHT and CUSUM provided the best results, followed by SCCDDd, which detected almost no false alarm. Next, SCCDD provided a false alarm at every 13, 892 observations in average. CRCDDd detected one false alarms at every 7708 observations in average, and CRCDDd one false alarm every 5261 observations.

Table 5 summarizes the average results of MDR, $MTFA^{-1}$ (the inverse of MTFA, once the greater MTFA is, the worst it is, making it easier to compare to the other measures) and MTD for each one of the algorithms. Values were normalized in range [0, 1] according to the lengths of data streams (number of observations) to improve comparison. Small values mean better results (after inversing MTFA).

Now, consider an application that provides the same importance level (or weight) for all the metrics we considered, thus we obtain the index shown in Table 6. The algorithms with better results are SCCDD, followed by CRCDDd and SCCDDd. The other algorithms provided much worse results, given small values are better. Comparing the proposed algorithms, SCCDD had better performance than CRCDDd, but its preprocessed version did not offered any improvement. On the other hand, the preprocessed version of CRCDDd was better than CRCDD.

5.1. Discussion of results

Before the discussion of results, it is worth to remember it is difficult to classify the "best algorithm" since all the metrics we considered are based on the target application. According to the results, ADWIN provided the best results for MDR and MTD, but also provided the worst result for MTFA. It conducted an "always-detection" approach. Consequently, ADWIN detected all the true alarms very quickly, but produced numerous false alarms (in a rate of 45 times more than the second worst algorithm, in term of the metric MTFA). On the other hand, PHT missed 2/3 of the true alarms for almost all data streams (and missed 100% in two of them). It obtained the best result for MTFA, since this algorithm operated as a "never-detection" approach. CUSUM detected true alarms quickly, but it missed 2/3 of the true alarms in almost all data streams (and missed 1/3 in one of them). Also, it generated few false alarms.

The reason the traditional algorithms ADWIN, PHT and CUSUM had performed those results, showing they are not good candidates for detecting concept drifts on those data streams, is that they trigger alarms when there are changes in the mean or the standard deviation of the collected data. Such statistics have significance for characterizing data which have linear behavior, but not when data has nonlinear and chaotic behavior, as were tested in this work.

The proposed algorithm SCCDD detected all true alarms for 4 data streams and missed 1/3 of those alarms in 5. Most of the detections happened with a time delay between 0 to 1000 observations, i.e., inside the window that had the concept changed or at window later, in average (as we considered 1000 as window length for this algorithm). Moreover, few false alarms were generated. The algorithm CRCDD detected all true alarms in 8 data streams and missed 1/3 of true alarms in one of them. These detections were under a time delay of approximately 3000 observations, i.e., after processing three windows of data. MTFA was greater than the one obtained with SCCDD. The algorithm SCCDDd detected all true alarms in 4 data streams, missed 1/3 in other 4 and 2/3 in one. The delay to detect true alarms was longer than SCCDD, and MTFA was similar. The algorithm CRCDDd obtained better results than CRCDD, because the delay to detect true alarms was shorter and the time delay between false alarms was longer. MDR was similar for both.

After this analysis, we conclude the algorithms SCCDD and SCCDDd are promising, but they had shown long delays for detecting concept drifts. Because of this, we performed a second experiment changing the length of the data window for both algorithms, considering the same synthetic data streams. We thus evaluated these algorithms for windows varying as 250, 500, 750 and 1000, always with an overlapping of 0%. The values obtained for MDR, MTD and MTFA are shown in Table 7.

From the results presented in Table 7, we observe the smaller the window length is, the best is the result for MTD and MDR, what is indeed expected once our algorithms analyze data windows to take decisions. On the other hand, the worst is the result for MTFA when the window length is reduced for our algorithms.

Table 7
Results of the second experiment: changing the length of the data window for the algorithms SCCDD and SCCDDd.

Algorithm	Window length	Metrics (mean ± sd)
SCCDD	250	MDR: 0 ± 0 MTD: 562 ± 208 MTFA: 2063 ± 405
SCCDD	500	MDR: 0.08 ± 0.15 MTD: 1062 ± 662 MTFA: 4555 ± 435
SCCDD	750	MDR: 0.125 ± 0.1725 MTD: 1681 ± 993 MTFA: 8912 ± 2552
SCCDD	1000	MDR: 0.1667 ± 0.1781 MTD: 864 ± 758 MTFA: 15128 ± 6079
SCCDDd	250	MDR: 0 ± 0 MTD: 395 ± 243 MTFA: 2083 ± 392
SCCDDd	500	MDR: 0.04 ± 0.11 MTD: 1260 ± 503 MTFA: 5461 ± 1166
SCCDDd	750	MDR: 0.08 ± 015 MTD: 1744 ± 1268 MTFA: 10208 ± 6084
SCCDDd	1000	MDR: 0.25 ± 0.23 MTD: 1635 ± 954 MTFA: 17066 ± 4841

In summary, by reducing the window length we better detect the true alarms, however unnecessarily issuing false alarms.

From this analysis we conclude our algorithms can be parameterized according to the target application. If the application domain requires the detection of concept drifts with shorter time delays, regardless the number of false alarms, one should consider smaller data windows. On the other hand, if the priority is to reduce the number of false alarms, we recommend larger windows.

6. Conclusion and future works

Real-world data streams may change their behaviors over time, what is referred to as concept drift. By detecting those changes one can obtain important information about the phenomenon which produced the data stream. Most of the concept drift detection algorithms from literature consider that a concept is associated to the modification of linear and stationary processes. As first contribution, in this work we also allow a concept to be produced by nonlinear and non-stationary processes. Furthermore, we proposed four concept drift detection algorithms based on dynamical systems, which consider how data evolves along time. Consequently the temporal dependencies of data observations are taken into account, what, in our point of view, is very relevant once streams are the consequence of the collection of data produced by different phenomena along time (such as temperature and rainfall measurements).

We compared our four algorithms against other three from the literature, considering three metrics: Missed Detection Rate (MDR), Mean Time to Detection (MTD) and Mean Time between False Alarms (MTFA). Those algorithms applied on several synthetic data streams under different deterministic and stochastic (used to represent noise) influences. Synthetic data was produced once we could know when exactly every concept changes. Results confirm that the algorithms from literature proceed: i) either by an always detection approach, creating several false alarms; or ii) by a never detection approach, avoiding alarms as much as possible. On the other hand, our algorithms detected most of the concept drifts in data streams containing complex behaviors, creating few false alarms. We considered those streams are complicated enough as they contain data produced by deterministic as well as stochastic sources.

According to experiments, our algorithm SCCDD obtained the best results according to the index detailed in Section 5, once it is capable of detecting drifts even with the addition of noise. The second best algorithm was CRCDDd, which provided best results than CRCDD, confirming this last one needs the decomposition stage to reveal the deterministic influences of data streams. From that, we conclude the employment of dynamical system tools can indeed support the detection of concept drift in data streams. To respect the principle of fairness, as weakness point our algorithms were implemented to model data windows and compare them to the past ones in order to identify concept drifts. In fact we are working to improve them in order to model the instantaneous collection of data observations as they become available, making them incremental as necessary to process real-world data streams with chaotic influences.

Acknowledgments

This paper is based upon projects sponsored by CAPES (Coordination for the Improvement of Higher Level Personnel), CNPq (National Council for Scientific and Technological Development) and FAPESP (São Paulo Research Foundation), Brazil, under grants CAPES DS-5230736/D, CNPq 206926/2014-6, 441583/2014-8, 303051/2014-0 and FAPESP 2014/13323-5. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of CAPES, CNPq and FAPESP.

References

Aggarwal, C. C., Han, J., Wang, J., & Yu, P. S. (2003). A framework for clustering evolving data streams. In *Proceedings of the 29th international conference on very large data bases-volume 29* (pp. 81–92). VLDB Endowment.

Albertini, M. K., & de Mello, R. F. (2007). A self-organizing neural network for detecting novelties. In *Proceedings of the 2007 ACM symposium on applied computing* (pp. 462–466). ACM.

Alligood, K. T., Sauer, T. D., & Yorke, J. A. (1997). *Chaos: An introduction to dynamical systems*. Springer.

Babcock, B., Babu, S., Datar, M., Motwani, R., & Widom, J. (2002). Models and issues in data stream systems. In *Proceedings of the twenty-first ACM sigmod-sigact-sigart symposium on principles of database systems* (pp. 1–16). ACM.

Baena-García, M., del Campo-Ávila, J., Fidalgo, R., Bifet, A., Gavaldà, R., & Morales-Bueno, R. (2006). Early drift detection method, 77–86.

Basseville, M., Nikiforov, I. V., et al. (1993). *Detection of abrupt changes: theory and application*: vol. 104. Prentice Hall Englewood Cliffs.

Bernhard, H.-P., Darbellay, G., et al. (1999). Performance analysis of the mutual information function for nonlinear and linear signal processing. In *Acoustics, speech, and signal processing, 1999. proceedings., 1999 IEEE international conference on: vol. 3* (pp. 1297–1300). IEEE.

Bifet, A., & Gavalda, R. (2007). Learning from time-changing data with adaptive windowing.. In *SDM: vol. 7* (p. 2007). SIAM.

Bifet, A., Holmes, G., Kirkby, R., & Pfahringer, B. (2010). Moa: Massive online analysis. *The Journal of Machine Learning Research, 11*, 1601–1604.

Bifet, A., & Kirkby, R. (2009). Data stream mining a practical approach.

Bifet, A., Read, J., Pfahringer, B., Holmes, G., & Žliobaitė, I. (2013). Cd-moa: Change detection framework for massive online analysis. In *Advances in intelligent data analysis XII* (pp. 92–103). Springer.

Cao, F., Ester, M., Qian, W., & Zhou, A. (2006). Density-based clustering over an evolving data stream with noise.. In *SDM: vol. 6* (pp. 328–339). SIAM.

Carlsson, G., & Mémoli, F. (2010). Characterization, stability and convergence of hierarchical clustering methods. *The Journal of Machine Learning Research, 99*, 1425–1470.

da Costa, F. G., & de Mello, R. F. (2014). A stable and online approach to detect concept drift in data streams. In *2014 Brazilian Conference on Intelligent Systems (BRACIS)* (pp. 330–335).

Darbellay, G. A., & Vajda, I. (1999). Estimation of the information by an adaptive partitioning of the observation space. *IEEE Transactions on Information Theory, 45*(4), 1315–1321.

Elwell, R., & Polikar, R. (2011). Incremental learning of concept drift in nonstationary environments. *Neural Networks, IEEE Transactions on, 22*(10), 1517–1531.

Fraser, A. M., & Swinney, H. L. (1986). Independent coordinates for strange attractors from mutual information. *Physical Review A, 33*(2), 1134.

Gama, J., Medas, P., Castillo, G., & Rodrigues, P. (2004). Learning with drift detection. In *Advances in artificial intelligence–SBIA 2004* (pp. 286–295). Springer.

Guha, S., Mishra, N., Motwani, R., & O'Callaghan, L. (2000). Clustering data streams. In *Foundations of computer science, 2000. proceedings. 41st annual symposium on* (pp. 359–366). IEEE.

Huang, N. E., Shen, Z., Long, S. R., Wu, M. C., Shih, H. H., Zheng, Q., et al. (1998). The empirical mode decomposition and the hilbert spectrum for nonlinear and non-stationary time series analysis. In *Proceedings of the royal society of London A: Mathematical, physical and engineering sciences: vol. 454* (pp. 903–995). The Royal Society.

Kantz, H., & Schreiber, T. (2004). *Nonlinear time series analysis*: vol. 7. Cambridge University Press.

Kennel, M. B., Brown, R., & Abarbanel, H. D. (1992). Determining embedding dimension for phase-space reconstruction using a geometrical construction. *Physical Review A, 45*(6), 3403.

Klinkenberg, R., & Joachims, T. (2000). Detecting concept drift with support vector machines.. In *ICML* (pp. 487–494).

Marwan, N., Romano, M. C., Thiel, M., & Kurths, J. (2007). Recurrence plots for the analysis of complex systems. *Physics Reports, 438*(5), 237–329.

O'callaghan, L., Meyerson, A., Motwani, R., Mishra, N., & Guha, S. (2002). Streaming-data algorithms for high-quality clustering. In *2013 IEEE 29th international conference on data engineering (ICDE)* (p. 0685). IEEE Computer Society.

Page, E. (1954). Continuous inspection schemes. *Biometrika, 41*, 100–115.

Pereira, C. M., & de Mello, R. F. (2014). Ts-stream: clustering time series on data streams. *Journal of Intelligent Information Systems, 42*(3), 531–566.

R Core Team (2015). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing.

Rios, R. A., & Mello, R. F. (2016). Applying empirical mode decomposition and mutual information to separate stochastic and deterministic influences embedded in signals. *Signal Processing, 118*, 159–176.

Rios, R. A., Parrott, L., Lange, H., & de Mello, R. F. (2015). Estimating determinism rates to detect patterns in geospatial datasets. *Remote Sensing of Environment, 156*, 11–20.

Serra, J., Serra, X., & Andrzejak, R. G. (2009). Cross recurrence quantification for cover song identification. *New Journal of Physics, 11*(9), 093017.

Takens, F. (1981). *Detecting strange attractors in turbulence*. Springer.

Trulla, L., Giuliani, A., Zbilut, J. P., & Webber, C. L. (1996). Recurrence quantification analysis of the logistic equation with transients. *Physics Letters A, 223*(4), 255–260.

Tsymbal, A. (2004). *The problem of concept drift: definitions and related work*. Computer Science Department, Trinity College Dublin.106

Vallim, R. M., Andrade Filho, J. A., De Mello, R. F., & De Carvalho, A. C. (2013). Online behavior change detection in computer games. *Expert Systems with Applications, 40*(16), 6258–6265.

Vallim, R. M., & De Mello, R. F. (2014). Proposal of a new stability concept to detect changes in unsupervised data streams. *Expert Systems with Applications, 41*(16), 7350–7360.

Vapnik, V. N., & Chervonenkis, A. Y. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & Its Applications, 16*(2), 264–280.

Wang, Y.-H., Yeh, C.-H., Young, H.-W. V., Hu, K., & Lo, M.-T. (2014). On the computational complexity of the empirical mode decomposition algorithm. *Physica A: Statistical Mechanics and its Applications, 400*, 159–167.

Whitney, H. (1936). Differentiable manifolds. *Annals of Mathematics, 37*, 645–680.

Zbilut, J. P., Giuliani, A., & Webber, C. L. (1998). Detecting deterministic signals in exceptionally noisy environments using cross-recurrence quantification. *Physics Letters A, 246*(1), 122–128.

Zbilut, J. P., & Webber, C. L. (1992). Embeddings and delays as derived from quantification of recurrence plots. *Physics letters A, 171*(3), 199–203.