INPE – National Institute for Space Research
São José dos Campos – SP – Brazil – May 16-20, 2016

DETECTING DYNAMICAL CHANGES IN DATA STREAMS

Fausto Guzzo da Costa[1] *and Rodrigo Fernandes de Mello*[2]

[1]Universidade de São Paulo, São Carlos, Brazil, fausto@icmc.usp.br
[2]Universidade de São Paulo, São Carlos, Brazil, mello@icmc.usp.br

Abstract: Nature processes are typically nonstationary, many of them exhibit chaos. In some situations, the changes in their behavior represent the most interesting event. A natural way of analyzing such data is by the study of phase space trajectories over time. In this study, we employ concepts from Data Streams, a subarea of Machine Learning, for detecting such dynamical changes. The proposed method is capable of dealing with data continuously produced at high rates. Experiments with synthetic confirm our approach detects dynamical changes on nonlinear and nonstationary data with noise added.

keywords: Nonlinear Dynamics and Complex Systems, Chaos and Global Nonlinear Dynamics, Time Series Analysis, Data Streams.

1. INTRODUCTION

Several processes found in nature have the characteristics of being nonlinear and nonstationary. They are very common, for example, when observing natural or cultural phenomena, such as recordings from living beings [3]. Many of them can also exhibit chaos, producing time series that are complicated – even when the processes are deterministic, due to sensitive dependence on initial conditions.

Since almost every traditional nonlinear method for time series analysis assume some stationarity, detecting nonstationarity is a crucial task [6]. Furthermore, in some situations the nonstationarity is the most interesting event. In several fields, such as physical, medical, engineering and economic sciences the detection of changes in the process behaviors is very important [2].

The basic tool often used for nonlinear time series analysis is the time delay embedding [3], which permits the reconstruction of the processes phase space. Such representation, instead of time or frequency domain approaches, is the fingerprint of the field. Let $s_N = \{s_1, s_2, \ldots, s_n\}$ be a time series, i.e. a sequence of scalar measurements with size N. The time delay reconstruction in m dimensions is given by:

$$\mathbf{x}_n = (s_n, s_{n-\tau}, s_{n-2\tau}, \ldots, s_{n-(m-2)\tau}, s_{n-(m-1)\tau}), \quad (1)$$

where $\mathbf{x}_n$ is a state on the phase space, specified by a vector $\mathbf{x} \in \mathbb{R}^m$. The dimension m is referred to as embedding dimension, and τ as the delay time.

Most of the methods for detecting dynamical changes in time series perform their task by examining the reconstructed phase space. Commonly, a sliding window is handled over the time series, reconstructing the phase space and applying an underlying method for every window. Thus, each window is represented by a single number. This number is then analyzed considering the sequence of numbers already produced by the method over the last windows, and when a change is detected, it is stated that a dynamical change happened in the data. Examples of methods are the largest Lyapunov Exponent estimate [5], Recurrence Plots [4] and Permutation Entropy [2].

In recent years, we have seen the uprise of processes that generate huge amounts of online data, such as sensor networks. Potentially, these data can be produced at rapid rate and can grow indefinitely. For these kind of data, referred as *data streams*, new algorithms were developed for dealing with the new challenges, as the need of online processing and limited storage.

In this paper we propose a method for detecting dynamical changes in data streams. As data streams are potentially unbounded, we limit our storage by processing every data point as it is received, dropping it afterwards. This is a different method from the traditional literature, as we do not use sliding windows. We maintain a model of the actual phase space, updating it as new data points arrive. First, when a data point is received, we reconstruct its (phase space) state. This reconstruction is performed in a online fashion, which is also a novel in this work. Next, we use a algorithm from the data stream field called DenStream [2] for clustering every reconstructed state, composing the model of the actual phase space. This algorithm also considers that states that composes the model fades over time. Analyzing the behavior of this algorithm, as proposed in [7], we are able to detect when the model changes, therefore detecting a dynamical change in the data stream.

This paper is organized as follows: in the next section the traditional methods from the literature are described; in Section 3 the proposed method is detailed; in Section 4 the experiment is described and its results are discussed; and in the last section, conclusions are presented.

2. RELATED WORK

2.1. Lyapunov exponent

One of the most important methods for detecting dynamical changes in time series is by estimating the largest Lyapunov exponent for each window. The Lyapunov exponents can be thought as a ellipsoid of m dimensions. Choosing two random trajectories produced by the same dynamical system but with subtle different initial conditions, each dimension of the ellipsoid represents the distance between each dimension of the states $\mathbf{x}_i$ and $\mathbf{y}_i$ of different trajectories. As the time elapses, i.e. i increases, the ellipsoid change its size. If the dynamical system in study exhibit chaos, the largest Lyapunov exponent (λ_1) will, on average, exponentially increase. Commonly only the largest Lyapunov exponent is estimated, as it provides the most important information about the process in study [5].

Traditionally the largest Lyapunov exponent is estimated on a sliding window over a time series, producing positive values which indicates chaos and negative values otherwise. By the analysis of λ_1 over time, a dynamical change can be detected. The biggest disadvantage of this method is that the algorithms are computationally expensive.

2.2. Recurrence Quantification Analysis

Another important method is based on the Recurrence Plot, introduced in the late 1980's. Recurrence Plots are a powerful analysis tool which can be employed in nonstationary, nonlinear and chaotic time series [4]. By the construction of a binary recurrence matrix which indicates which states from the phase space are neighbors, a specialist is able to visually identify patterns in the data. This method is particularly interesting for high-dimensional data.

In order to reduce the subjectivity caused by the visual interpretation of the Recurrence Plots, it is preferred to use measures of complexity known as Recurrence Quantification Analysis. This measures are computed based on the textures contained in the recurrence matrix, such as counting the number of black dots with regard to white dots (called Recurrence Rate), or counting the largest largest diagonal line.

2.3. Permutation Entropy

Both methods previously described (and most of the methods from the field), quantify the dynamical changes based on aspects of the nearest neighbors in phase space. Thereby those methods are computationally expensive [2]. A more recent method, called Permutation Entropy, uses a different approach: it maps each state of the phase space into a symbol, and then calculates the entropy of such sequence of symbols. It is reported that such method is up to 100 times faster than approaches used to estimate the Lyapunov exponent [2].

3. THE PROPOSAL

In order to detect dynamical changes in data streams, we propose the use of a method for clustering data streams widely used in the literature, called DenStream [1]. As main features, the DenStream processes each received data point *only once*; it handles noise and outliers; and the most important feature for this work: it assumes that data streams change their behavior over time. For this, the algorithm manages a set of structures called *micro-clusters*, which are responsible for summarizing the received data points. These structures can be viewed as hyperspheres in a multidimensional space, capable of absorbing new data data points by changing their information. Next we define a micro-cluster.

Definition of micro-cluster: let a group of close points $p_1, \ldots, p_n$ with time stamps $t_1, \ldots, t_n$ being summarized by a micro-cluster c_p. The weight of this micro-cluster is a function of the data points time stamps: $w = \sum_{i=1}^{n} f(t - t_i)$, where t is the actual time. The center of this micro-cluster is defined as $c = \frac{\sum_{j=1}^{n} f(t-t_i)p_j}{w}$ and the radius as $r = \frac{\sum_{j=1}^{n} f(t-t_i)dist(p_j,c)}{w}$. In all these equations, the function $f(\cdot)$ is a fading function $f(t) = 2^{-\lambda \cdot t}$, where $\lambda > 0$ (this function is frequently used in temporal applications [1]). As this algorithm accepts only "dense" micro-clusters, two restrictions are made: for the weight, $w \geq \mu$, and for the radius, $r \leq \epsilon$, i.e. the micro-clusters has minimum weight and maximum radius.

Based on the micro-cluster information (center, radius and weight), the algorithm differentiates them in two types: potential micro-clusters and outlier micro-clusters. Both have the same structure as described before, but they are managed differently by the algorithm. When a micro-cluster has its weight w greater than a threshold $\beta \cdot \mu$ it is considered a potential micro-cluster; otherwise it is considered a outlier micro-cluster. The idea is that frequently updated hyperspheres are considered as potential.

When a new data point arrives, the algorithm chooses the most appropriate potential micro-cluster for merging. Then, this micro-cluster has its weight, center and radius updated. If the new radius exceeds a threshold value, i.e. the variance in this hypersphere is considered too large, the merge is not allowed, and therefore the algorithm chooses the closest outlier micro-cluster for merging. This new possible micro-cluster has its information updated. If the new radius is small enough, but the weight has increased more than a threshold, this outlier micro-cluster change its role to a potential one. On the other hand, if the merge has failed again, a new outlier micro-cluster is created only for this new arrived point. Note that a micro-cluster can change its role over time.

Additionally, the DenStream periodically promotes the administration of micro-clusters, by fading them all: this way, some are removed and some potential micro-clusters are updated to outliers.

We use the algorithm DenStream for detecting dynamical changes in a data stream produced at a fast rate. For this, the whole proposed method should work in a online way, i.e. it should process every data point received by the stream. The proposed method is divided in three steps: i) the online reconstruction of phase space; ii) the clustering; and iii) the detection of dynamical changes. Every obtained data point is processed by the three steps.

When a data point s_t is received from the data stream, the *first step* is to reconstruct the state vector $\mathbf{x}_t$. For this, it is necessary to use data points from the past, namely the $s_{t-\tau}, s_{t-2\tau}, \ldots, s_{t-(m-1)\tau}$. Subsequently the reconstruction of the state vector $\mathbf{x}_t = (s_t, s_{t-\tau}, s_{t-2\tau}, \ldots, s_{t-(m-1)\tau})$, the data point s_t should also be stored for further state vector's reconstructions. A buffer of size $(m-1)\cdot\tau$ is enough.

On the *second step*, the reconstructed state vector $\mathbf{x}_t$ is then clustered by the algorithm DenStream [1], which manages the merging, creation and removal of micro-clusters (potential and outliers). This step represents the space phase of the data stream considering a fading function, i.e. micro-clusters suffers from aging, slowly decreasing their importance to the algorithm up to the point they are removed. In addition, new micro-clusters can be created, representing new behaviors of the data. We use such changing on the micro-clusters for detecting dynamical changes.

The *third step* is performed by analyzing six rates gathered from the algorithm DenStream: the merging, creation and removal from the potential and outlier micro-clusters, as proposed in the work of [7]. Analyzing these rates it is expected to detect when a dynamical change has happened in the underlying data stream. For example, when the rate of merging is large, it is likely that no change is happening. On the other hand, if the rates of creation and removal of micro-clusters are increasing, it is likely that a dynamical change is happening.

4. EXPERIMENT AND RESULTS

We performed a experiment on a nonlinear and nonstationary data stream generated by sine functions, shown in Figure 1. The stream has $2,000$ data points in which 5 sines are generated, with 400 data points each, varying from 0 to 1. On the fifth sine, we modified its values to its surrogate, i.e. the statistics was preserved (as the first and second moments), but the time-dependence was broken. Thus in the fifth part of the data stream a dynamical change happens, and it is expected the method to detect it. We also added 5% of noise produced by a Normal distribution.

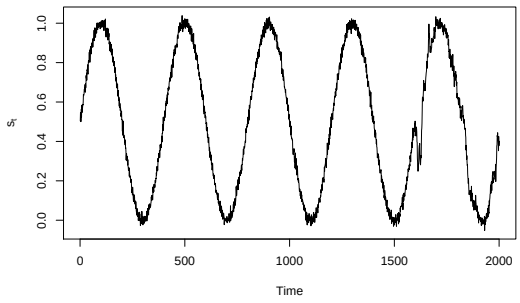

Figure 1 – The data stream used for the experiment, composed of 5 sines, in which the last one is a surrogate (i.e. the first and second moments are identical, but the time-dependence is broken.)

In Figure 2, we show the results of the proposed method. In the top-most plot, it is shown the received data stream; in the center plot, it is presented the rate of creation, merging and removal of potential micro-clusters (PMC); and in the bottom one, the rate of creation, merging and removal of outlier micro-clusters (OMC). In all plots, we indicate with a vertical bold line the end of the second sine. From the beginning of the data stream to this point we consider the algorithm was learning the data pattern. After this point, one can notice that all the rates (presented on the center and bottom graphics) are mostly constant, with the data points merging on PMCs while all the other rates are mostly zero. When a change happens, indicated by a vertical dashed line, the rate of merging on PMCs falls abruptly, and the creation of OMCs (and consequently the merging of OMCs) increases substantially.

The proposed method were capable of learning the "normal" behavior of the data stream, and consequently detect the dynamical change of it. The experiment were performed in a online fashion, i.e. no use of batch processing or even sliding windows. For this experiment, it was tested a frequency up to 400 Hz for the incoming data stream, which have not affected the performance of the method.

The great disadvantage of the proposed method is the number of parameters to set. In total it is necessary to define 4 parameters, namely the λ, for the exponential fading function; ϵ, the minimum radius threshold for a micro-cluster; μ, the minimum weight threshold for a micro-cluster; and β, the factor which divides a micro-cluster from being potential

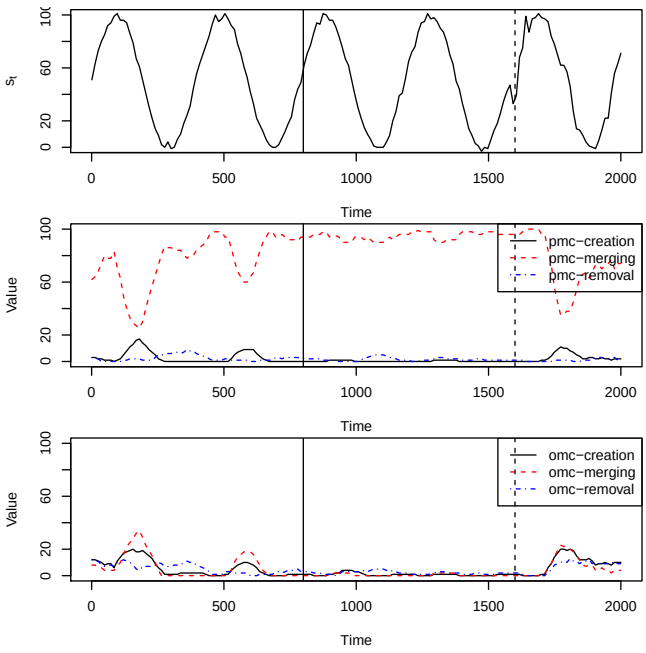

Figure 2 – Results from the experiment. The top graphic shows the received data stream. The center graphic presents the rate of potential micro-clusters that were created, merged and removed. The bottom graphic shows the rate of creation, merging and removal of outlier micro-clusters. From the time 0 to the bold vertical line it is the time the algorithm was learning the data behavior. From the bold vertical line to the dashed vertical line, the behavior does not change. And after this, a dynamical change happened.

or outlier. Besides this, it is also necessary to set the embedding dimension m and the delay time τ for reconstructing the phase space (which can be estimated).

5. CONCLUSIONS

In this paper we presented a method for detecting dynamical changes in data continuously produced at high rate, known as data streams. We performed a experiment in a online environment, in which every data point received from the stream was processed by the method. As a result the method was capable of successfully perform the detection. As far as our knowledge, there is no literature on using algorithms from the data streams field for detecting dynamical changes. Moreover, it is also a novel to reconstruct the phase space in a online fashion. We think the proposed method can be used for applications such as sensor networks for detecting changes on the underlying phenomena.

As future work, we plan to analyze the method when more complicated data streams are used, for example ones exhibiting chaos. Besides this, we intend to develop an automatic parameter estimate for the method, based on a "normal" behavior set from the data stream.

ACKNOWLEDGMENTS

We thank the support provided by CAPES DS-5230736/D, CNPq 206926/2014-6, 441583/2014-8, 303051/2014-0 and FAPESP 2014/13323-5. Findings and opinions do not necessarily reflect the view of CAPES, CNPq and FAPESP.

References

[1] Feng Cao, Martin Ester, Weining Qian, and Aoying Zhou. Density-based clustering over an evolving data stream with noise. In *SDM*, volume 6, pages 328–339. SIAM, 2006.

[2] Yinhe Cao, Wen-wen Tung, JB Gao, Vladimir A Protopopescu, and Lee M Hively. Detecting dynamical changes in time series using the permutation entropy. *Physical Review E*, 70(4):046217, 2004.

[3] Holger Kantz and Thomas Schreiber. *Nonlinear time series analysis*, volume 7. Cambridge university press, 2004.

[4] Norbert Marwan, M Carmen Romano, Marco Thiel, and Jürgen Kurths. Recurrence plots for the analysis of complex systems. *Physics reports*, 438(5):237–329, 2007.

[5] Michael T Rosenstein, James J Collins, and Carlo J De Luca. A practical method for calculating largest lyapunov exponents from small data sets. *Physica D: Nonlinear Phenomena*, 65(1):117–134, 1993.

[6] Thomas Schreiber. Detecting and analyzing nonstationarity in a time series using nonlinear cross predictions. *Physical Review Letters*, 78(5):843, 1997.

[7] Rosane MM Vallim, José A Andrade Filho, Rodrigo F De Mello, and André CPLF De Carvalho. Online behavior change detection in computer games. *Expert Systems with Applications*, 40(16):6258–6265, 2013.