UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Aplicando ferramentas de análise de séries temporais não lineares e algoritmos de agrupamento estáveis para a detecção de mudanças de conceito em fluxos de dados

Fausto Guzzo da Costa

Tese de Doutorado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

ICMC USP SÃO CARLOS

Fausto Guzzo da Costa

Employing nonlinear time series analysis tools with stable clustering algorithms for detecting concept drift on data streams

Doctoral dissertation submitted to the Instituto de Ciências Matemáticas e de Computação - ICMC-USP, in partial fulfillment of the requirements for the degree of the Doctorate Program in Computer Science and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Rodrigo Fernandes de Mello

USP – São Carlos
October 2017

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados fornecidos pelo(a) autor(a)

Costa, Fausto Guzzo
C837e Employing nonlinear time series analysis tools with stable clustering algorithms for detecting concept drift on data streams / Fausto Guzzo Costa; orientador Rodrigo Fernandes de Mello. -- São Carlos, 2017.
102 p.

Tese (Doutorado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2017.

1. Machine Learning. 2. Data Streams. 3. Concept Drift. 4. Clustering. 5. Nonlinear Time Series. I. Fernandes de Mello, Rodrigo, orient. II. Título.

Fausto Guzzo da Costa

Aplicando ferramentas de análise de séries temporais não lineares e algoritmos de agrupamento estáveis para a detecção de mudanças de conceito em fluxos de dados

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Doutor em Ciências – Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional/Matemática

Orientador: Prof. Dr. Rodrigo Fernandes de Mello

USP – São Carlos
Outubro de 2017

Dedico à minha família e amigos, com gratidão.

"Para ser grande, sê inteiro: nada
Teu exagera ou exclui.

Sê todo em cada coisa. Põe quanto és
No mínimo que fazes.

Assim em cada lago a lua toda
Brilha, porque alta vive."

Ricardo Reis

ACKNOWLEDGEMENTS

The completion of this thesis would not have been possible without the expertise of Prof. Dr. Rodrigo Fernandes de Mello, my thesis advisor. I would like to thank him, who mold each student to go beyond their limits, for the learning opportunities, and for the valuable guidance through various stages of my doctoral. I cannot express my gratitude to him for sharing his knowledge and spending so many hours next to me in order to improve the results of this thesis. His scientific curiosity will be always of great inspiration for me.

I transmit my deep sense of gratitude to my family, in special to my parents, for the continuous support, and the encouragement when times get rough. It was of great comfort and relief to know that you were there when I needed.

I am very much thankful to Prof. Holger Kantz for the opportunity to live next to scientists and feel the culture of a world-recognized research institute.

I also would like to thank the Bio-inspired Computation Laboratory group, specially Paulo Urio, Daniel Cestari, Luís Garcia, Luiz Coletta, Everlandio Fernandes, and Kemilly Dearo, for the daily discussions on topics such as scientific research, technology and politics. To the same extent, I would like to thank the Time Series Analysis (TSA) research group from Max-Plack Institute, particularly Justus Schwabedal, Stephan Bialonski, Phillip Müller, Bennedict Lünsmann, Anna Deluca, Mozhdeh Massah, and Yü. You all supported my scientific development, from my English skills to my scientific criticism, and also my organization at work.

I acknowledge with thanks my friends, who gave me moral and humor support for the completation of this thesis. Special thanks to Edvard for the daily companionship, Christoph for the continuous constructive criticism, and Gustavo for the intellectual encouragement. I am also thankful for the friends that supported me in Germany, whose never asked anything in exchange, Justus, Bennedict, Phillip, Corinna, Alex and Andi. I take this opportunity to record my sincere thanks to Rafael ‘Miyage’, Lucas Pintanel, Rodrigo ‘Genja’ and climbers from CUME, Alex Malheiros and all the family from the Om Mani Padme Hum xamanism.

“Somos todos um.”

“True wisdom comes to each of us when we realize how little we understand about life, ourselves, and the world around us.”

(Socrates)

ABSTRACT

DA COSTA, F. G. **Employing nonlinear time series analysis tools with stable clustering algorithms for detecting concept drift on data streams** 2017. 102 p. Doctoral dissertation (Doctorate Candidate Program in Computer Science and Computational Mathematics) – Institute of Mathematics and Computer Sciences, University of São Paulo, São Carlos – SP, 2017.

Several industrial, scientific and commercial processes produce open-ended sequences of observations which are referred to as *data streams*. We can understand the phenomena responsible for such streams by analyzing data in terms of their inherent recurrences and behavior changes. Recurrences support the inference of more stable models, which are deprecated by behavior changes though. External influences are regarded as the main agent actuacting on the underlying phenomena to produce such modifications along time, such as new investments and market polices impacting on stocks, the human intervention on climate, etc. In the context of Machine Learning, there is a vast research branch interested in investigating the detection of such behavior changes which are also referred to as *concept drifts*. By detecting drifts, one can indicate the best moments to update modeling, therefore improving prediction results, the understanding and eventually the controlling of other influences governing the data stream. There are two main concept drift detection paradigms: the first based on supervised, and the second on unsupervised learning algorithms. The former faces great issues due to the labeling infeasibility when streams are produced at high frequencies and large volumes. The latter lacks in terms of theoretical foundations to provide detection guarantees. In addition, both paradigms do not adequately represent temporal dependencies among data observations. In this context, we introduce a novel approach to detect concept drifts by tackling two deficiencies of both paradigms: i) the instability involved in data modeling, and ii) the lack of time dependency representation. Our unsupervised approach is motivated by Carlsson and Memoli's theoretical framework which ensures a *stability* property for hierarchical clustering algorithms regarding to data permutation. To take full advantage of such framework, we employed Takens' embedding theorem to make data statistically independent after being mapped to phase spaces. Independent data were then grouped using the Permutation-Invariant Single-Linkage Clustering Algorithm (PISL), an adapted version of the agglomerative algorithm Single-Linkage, respecting the stability property proposed by Carlsson and Memoli. Our algorithm outputs dendrograms (seen as data models), which are proven to be equivalent to ultrametric spaces, therefore the detection of concept drifts is possible by comparing consecutive ultrametric spaces using the Gromov-Hausdorff (GH) distance. As result, model divergences are indeed associated to data changes. We performed two main experiments to compare our approach to others from the literature, one considering abrupt and another with gradual changes. Results confirm our approach is capable of detecting concept drifts, both abrupt and gradual ones, however it is more adequate to operate on complicated scenarios. The main contributions of this thesis are: i) the usage of Takens' embedding theorem as tool to provide statistical independence to data streams; ii) the implementation of PISL in conjunction with GH (called PISL–GH); iii) a comparison of detection algorithms in different scenarios; and, finally, iv) an R package (called streamChaos) that provides tools for processing nonlinear data streams as well as other algorithms to detect concept drifts.

Keywords: Machine Learning, Data Streams, Concept Drift, Clustering, Nonlinear Time Series.

RESUMO

DA COSTA, F. G. **Aplicando ferramentas de análise de séries temporais não lineares e algoritmos de agrupamento estáveis para a detecção de mudanças de conceito em fluxos de dados** 2017. 102 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2017.

Diversos processos industriais, científicos e comerciais produzem sequências de observações continuamente, teoricamente infinitas, denominadas *fluxos de dados*. Pela análise das recorrências e das mudanças de comportamento desses fluxos, é possível obter informações sobre o fenômeno que os produziu. A inferência de modelos estáveis para tais fluxos é suportada pelo estudo das recorrências dos dados, enquanto é prejudicada pelas mudanças de comportamento. Essas mudanças são produzidas principalmente por influências externas ainda desconhecidas pelos modelos vigentes, tal como ocorre quando novas estratégias de investimento surgem na bolsa de valores, ou quando há intervenções humanas no clima, etc. No contexto de Aprendizado de Máquina (AM), várias pesquisas têm sido realizadas para investigar essas variações nos fluxos de dados, referidas como *mudanças de conceito*. Sua detecção permite que os modelos possam ser atualizados a fim de apurar a predição, a compreensão e, eventualmente, controlar as influências que governam o fluxo de dados em estudo. Nesse cenário, algoritmos supervisionados sofrem com a limitação para rotular os dados quando esses são gerados em alta frequência e grandes volumes, e algoritmos não supervisionados carecem de fundamentação teórica para prover garantias na detecção de mudanças. Além disso, algoritmos de ambos paradigmas não representam adequadamente as dependências temporais entre observações dos fluxos. Nesse contexto, esta tese de doutorado introduz uma nova metodologia para detectar mudanças de conceito, na qual duas deficiências de ambos paradigmas de AM são confrontados: i) a instabilidade envolvida na modelagem dos dados, e ii) a representação das dependências temporais. Essa metodologia é motivada pelo arcabouço teórico de Carlsson e Memoli, que provê uma propriedade de *estabilidade* para algoritmos de agrupamento hierárquico com relação à permutação dos dados. Para usufruir desse arcabouço, as observações são embutidas pelo teorema de imersão de Takens, transformando-as em independentes. Esses dados são então agrupados pelo algoritmo Single-Linkage Invariante à Permutação (PISL), o qual respeita a propriedade de estabilidade de Carlsson e Memoli. A partir dos dados de entrada, esse algoritmo gera dendrogramas (ou modelos), que são equivalentes a espaços ultramétricos. Modelos sucessivos são comparados pela distância de Gromov-Hausdorff a fim de detectar mudanças de conceito no fluxo. Como resultado, as divergências dos modelos são de fato associadas a mudanças nos dados. Experimentos foram realizados, um considerando mudanças abruptas e o outro mudanças graduais. Os resultados confirmam que a metodologia proposta é capaz de detectar mudanças de conceito, tanto abruptas quanto graduais, no entanto ela é mais adequada para cenários mais complicados. As contribuições principais desta tese são: i) o uso do teorema de imersão de Takens para transformar os dados de entrada em independentes; ii) a implementação do algoritmo PISL em combinação com a distância de Gromov-Hausdorff (chamado PISL–GH); iii) a comparação da metodologia proposta com outras da literatura em diferentes cenários; e, finalmente, iv) a disponibilização de um pacote em R (chamado streamChaos) que provê tanto ferramentas para processar fluxos de dados não lineares quanto diversos algoritmos para detectar mudanças de conceito.

Palavras-chave: Aprendizado de Máquina, Fluxos de Dados, Mudanças de Conceito, Agrupamento, Séries Temporais Não Lineares.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

ACF	Autocorrelation Function
AR	Autoregressive model
ARIMA	Autoregressive Integrated and Moving Average model
ARMA	Autoregressive and Moving Average model
BIRCH	Balanced Iterative Reducing and Clustering using Hierarchies
CF	Clustering Feature
CFT	Temporal Clustering Features
CMC	Core-micro-clusters
EWMA	Exponential-Weighted Moving Averages
FNN	False Nearest Neighbors
GH	Gromov-Hausdorff distance
MA	Moving Average model
MDFT	Multidimensional Fourier Transform
MDR	Missed Detection Rate
MI	Mutual Information
MTD	Mean Time for Detection
MTFA	Mean Time between False Alarms
PE	Permutation Entropy
PISL	Permutation-Invariant Single-Linkage clustering algorithm
PISL–GH	Permutation-Invariant Single-Linkage Clustering Algorithm with the Gromov-Hausdorff distance
RP	Recurrence Plot
RQA	Recurrence Quantification Analysis
RQA–DET	Recurrence Quantification Analysis using Determinism
RQA–LMAX	Recurrence Quantification Analysis using the inverse of the Longest Diagonal Line
RQA–RR	Recurrence Quantification Analysis using Recurrence Rate
SVD	Singular Value Decomposition

LIST OF SYMBOLS

s_t — Unidimensional observation from a time series

S_n — Time series

$\mathbf{x}_t$ — State in a phase space

Γ — Phase space

τ — Time delay for the embedding theorem by Takens (1981)

m — Embedding dimension for the embedding theorem by Takens (1981)

θ — Dendrogram

$\Psi(\theta)$ — Ultrametric space equivalent to dendrogram θ

$d_{\mathscr{G}\mathscr{H}}(\cdot)$ — Gromov-Hausdorff distance

CONTENTS

CHAPTER

1

INTRODUCTION

"Man is condemned to be free; because once thrown into the world, he is responsible for everything he does. It is up to you to give [life] a meaning."

Jean-Paul Sartre

1.1 Context and motivation

Several industrial, scientific and commercial processes produce *data streams*, which are open-ended sequences of observations produced along time. Examples include transactions dispatched by credit card processors, temperature sensing by climate monitoring centers, stock market operations, heart monitoring, etc (AGGARWAL, 2015; GUHA *et al.*, 2003). By analyzing and modeling such data, one can predict, understand and eventually control the phenomena responsible for producing such streams.

Due to modifications on the underlying phenomena, data streams may change their behavior along time, for example when new investments are made in the stock market or when human intervene on climate. Those fluctuations are named *concept drifts*, and they usually point out the most interesting events once they inform us about external variables impacting on the phenomenon. While no drift is detected, we know the phenomenon is stable and already represented by a previous modeling.

The detection of such changes is of main importance in several scenarios, such as in the monitoring of the Sahel, a strip with $500-700$ by $5,400$ kilometers located in the Sub-Saharan Africa, whose area shifts according to variables like temperature, moisture, and absorption of active radiation (BUONTEMPO; BOOTH; MOUFOUMA-OKIA, 2010). Changes in this

phenomenon influence the Sub-Saharan climate, and directly affect the already limited cultivable areas of Africa.

There are several algorithms for detecting concept drift, being most of them dedicated to supervised data (KLINKENBERG; RENZ, 1998; KLINKENBERG; JOACHIMS, 2000; KLINKENBERG, 2004; GAMA *et al.*, 2004; GAMA; FERNANDES; ROCHA, 2006; BAENA-GARCÍA *et al.*, 2006). As consequence, they rely on data labeling to proceed with modeling and classification. However, this supervised learning paradigm faces great issues when data streams are produced at high frequencies and large volumes, what makes unfeasible data labeling. Therefore, researchers have been motivated by the *unsupervised learning* paradigm, once it does not require previous knowledge about data.

Unsupervised learning mainly relies on data clustering algorithms to extract structural patterns from data, by grouping most similar objects through similarity measurements. Such algorithms assume that data changes alter clustering partitions, which are used to detect concept drift (AGGARWAL *et al.*, 2003; ALBERTINI, 2012; VALLIM *et al.*, 2013). Therefore, the partition produced at the time instant t is compared against the previous $t-1$. From this comparison, a divergence measurement is obtained, which is used to indicate a drift. While small perturbations are seen as data instability, significant changes correspond to modifications in the phenomenon responsible for producing the stream. The detection of such changes is the main motivation for the research field of concept drift, which is characterized by abrupt, usually easier to be detected, or by slow and gradual changes (TSYMBAL, 2004).

However, those unsupervised algorithms lack of theoretical foundation to provide guarantees for the concept drift detection. Most of them compare consecutive partitions using some *heuristics*, hence divergences are not necessarily associated to data changes (and, consequently, to the phenomenon), but they may also be produced by the algorithm instability. For instance, algorithms such as STING (WANG; YANG; MUNTZ, 1997) and CluStream (AGGARWAL *et al.*, 2003) can produce different clustering partitions even for the very same data observations due to the K-means parameterization (K-means is known to be unstable because of its random cluster prototype initialization). The main problem in this scenario is that a drift detection may not necessarily indicate data variations resultant from the underlying phenomenon.

In addition to the fact that most of the unsupervised algorithms are not stable, the state-of-the-art clustering algorithms do not adequately represent temporal dependencies among data observations. Many of them even assume data streams were produced by independent and identically probability distributions (GAMA *et al.*, 2004; BAENA-GARCÍA *et al.*, 2006), thus the possible time relationships among observations, which is likely to bring some useful information, are ignored. Others consider that the current observation depends only on the previous one (ALBERTINI, 2012), or assume an exponentially weighted moving average among all the previous observations (AGGARWAL *et al.*, 2003; KRANEN *et al.*, 2011). We believe this characterization of a data stream misses the fact that observations may contain other sort of

dependencies in a more complicated way, which once revealed could better support modeling, prediction and the study of data streams. In fact, this already has been studied by the field of Nonlinear Time Series Analysis (KANTZ; SCHREIBER, 2004) and formalized through Takens (1981)' embedding theorem, in which one data observation may depend on many previous ones at different time lags.

1.2 Hypothesis and objective

In this thesis, we tackle two deficiencies of novel clustering algorithms for detecting concept drifts on data streams: i) the first is related to the modeling instability, and ii) the second, to the lack of representation for time dependencies among data observations. Based on both drawbacks, we define the following hypothesis:

Hypothesis: *By applying Takens (1981) embedding theorem on data streams, we represent time dependencies among observations, and we provide theoretical guarantees for concept drift detection using the stable hierarchical clustering proposed by Carlsson & Mémoli (2010).*

In order to verify this hypothesis, we made use of the stability property proposed by Carlsson & Mémoli (2010), which asserts that a hierarchical clustering algorithm is stable if and only if any permutation of the input data produces the very same model, i.e., the algorithm output is not influenced by the data order. Carlsson & Mémoli (2010) rely on dendrograms to provide their theoretical foundation, which are proven to be equivalent to ultrametric spaces. Once partitions are represented in such space, they can be reliably compared based on Mathematical Topology proofs. Therefore, the Gromov-Hausdorff distance can be employed to compute the divergence between clustering partitions (also referred to as models). Carlsson & Mémoli (2010) also propose a modification on an agglomerative clustering algorithm to make it hold their stability property. In this thesis, we perform the detection of concept drifts on data streams by first employing such algorithm to produce models over time, then comparing them out through the Gromov-Hausdorff distance. By using their framework, we provide theoretical guarantees that divergences among clustering partitions are indeed associated to data changes and, consequently, due to modifications in the generation process.

Such stability property assumes data observations are independent, what brings inconsistency on our proposal, as we intend to consider time dependencies. To tackle this issue, we make use of the Takens (1981)' embedding theorem to reconstruct data stream observations into phase spaces, which is shown to transform data from dependent into independent (PAGLIOSA; MELLO, 2017), and also embeds time dependencies among observations.

Before taking advantage of the embedding theorem, our approach extracts data chunks from the stream using a sliding window along the time domain. Then, this data window $\mathbf{w_t}$ is reconstructed into the phase space Γ_t, which embeds all temporal dependencies into a multi-

dimensional space and make data independent. Then, we cluster the embedded data window by using the Permutation-Invariant Single-Linkage (PISL) Clustering Algorithm, proposed by Carlsson & Mémoli (2010), to produce the model M_t, which is a dendrogram. Afterwards, a second data window $\mathbf{w_{t+1}}$ will pass through the same process, first being embedded as Γ_{t+1} then modeled as M_{t+1}. The two models M_t and M_{t+1} obtained by consecutive windows are then compared using the Gromov-Hausdorff distance to measure their dissimilarity. This divergence measurement is stored, and as new data observations are collected, the window will slide over the stream, and the whole process will take place again, producing more dissimilarity measurements on consecutive windows over time. By analyzing such measurements, our approach provides theoretical guarantees for the concept drift detection on data streams.

This thesis is developed under the following assumptions: i) the underlying data stream is unidimensional, and its observations are numerical and evenly distributed along time; ii) there is some temporal relationship among observations, in order they carry information from previous observations; and iii) there are enough observations into data windows that permits us to properly reconstruct the phase space of the underlying system. Moreover, we planned the methodology by thinking on the theoretical foundations of each step, and also respecting their assumptions.

We compare our approach with state-of-the-art algorithms from the field of Nonlinear Time Series Analysis, such as the Permutation Entropy (PE) (CAO *et al.*, 2004) and the Recurrence Quantification Analysis (RQA) (TRULLA *et al.*, 1996), and another algorithm proposed by our research group, the Multidimensional Fourier Transform (MDFT) (MELLO *et al.*, 2013), which follows a different stability criterion. In order to produce wide results, we decided to perform experiments assessing the detection of *abrupt* and *gradual* changes.

1.3 Organization of the thesis

This doctoral thesis is organized as follows. In Chapter 2, we introduce data streams and the concept drift detection, discussing about the evolution of relevant algorithms. Next, in Chapter 3, we introduce the field of time series analysis, focusing on nonlinear time series and on Takens (1981)' embedding theorem. Still in that chapter, we detail the state-of-the-art algorithms for the detection of concept drifts. Chapter 4 describes the stability hierarchical clustering proposed by Carlsson & Mémoli (2010). In Chapter 5, we detail the experiments performed using the Permutation-Invariant Single-Linkage Clustering Algorithm with the Gromov-Hausdorff distance (PISL–GH), which is used to detect concept drifts based on the sliding windows over data streams. Streams at different dimensions are taken into account, from abrupt to gradual changes. Chapter 6 presents the concluding remarks and future work.

CHAPTER

2

DATA STREAMS AND CONCEPT DRIFT DETECTION

2.1 Initial considerations

In the next sections, we introduce the fields of data streams and concept drift detection, which are considered in the context of this thesis. Lastly, we present the related work in a time evolution manner.

2.2 Data streams

Open-ended sequences of continuously produced data are considered data streams (PAVLIDIS *et al.*, 2011). Typically, it is assumed they are produced in huge volumes and at high frequencies. Some real-world systems produce data streams, such as the ones associated to telecommunication companies, banking systems, stock market, sensor networks, weather satellites, the CERN's Large Hadron Collider, etc (AGGARWAL, 2015; GUHA *et al.*, 2003). Such streams can be continuously studied and analyzed in order to obtain information from the phenomena responsible for generating them and, therefore, model and predict their behavior. In the context of this thesis, data streams are represented as sequences of observations $\mathbf{x}_1, \ldots, \mathbf{x}_i, \ldots, \mathbf{x}_n$, in which each observation is a vector of d real attributes, i.e., $\mathbf{x}_i \in \mathbb{R}^d$.

Given the possible high frequency that data is produced, the algorithms for processing streams experience restrictions on their time complexity, since the algorithm cannot perform demanding operations while processing each data observation, otherwise the stream analysis would be impacted (BAENA-GARCÍA *et al.*, 2006). Another restriction is associated to the infinite nature of streams – since the computer memory is limited, only the most important data or information should be stored. Due to these two restrictions, data stream algorithms should analyze the collected data, store their characteristics or relevant information, and discard them

thereafter. Hence, data observations should be processed in a *single pass*, i.e., in an entirely different manner than the traditional data sets, whose observations are completely stored in memory and processed multiple times, as a batch.

Data streams distinguish from traditional data sets in many aspects, as discussed in Albertini (2012). Among them: i) the number of observations, which is finite in traditional data sets, and infinite in data streams; ii) traditional data sets can be handled in multiple passes, while data streams should be managed in a single pass. Because of this, data streams should be operated in the order data observations are collected; and iii) traditional data sets are commonly labeled by specialists, while this practice is unfeasible when data streams are produced at great volumes and high frequencies (that is why some researchers consider semi-supervised learning in this scenario).

Another characteristic of data streams is the behavior evolution over time. For example, consider the behavior of temperature and precipitation in India during the shift of seasons to monsoon, caused by seasonal reversing wind in the atmospheric circulation. Some parts of India turn from semi-desert into green lands because of the great increase in the volume of rains. Hence, the weather model employed to predict the usual precipitation for those regions has its error considerably increased during the change of seasons (HALPERT; BELL, 1997). Such changes on the stream behavior are referred to as *concept drift*.

In order to process data stream observations, two main approaches are used: sliding windows and the incremental mode. In the first, the algorithms store the collected observations into data windows for further analysis. Such approach can use different strategies for managing the data windows: i) they can have fixed length, being subject to be too short or too long for monitoring concept drifts; ii) they can adapt their sizes according to the duration of a concept; and iii) one can assign weights to data observations according to the time instant they were collected (KLINKENBERG; RENZ, 1998; KLINKENBERG; JOACHIMS, 2000). On the other hand, when using the incremental mode, every data observation is processed as it is received. This continuous nature of data analysis brought more popularity to those algorithms (ALBERTINI, 2012; KRANEN *et al.*, 2011).

In the context of this thesis, we suppose some dependency on the observations composing the data stream. Hence, we define data windows in a similar way to the concept of time series, which will be described in Chapter 3.

2.3 Concept drift

A data stream is composed of a set of concepts, in which each concept commands the data behavior along time (CAO *et al.*, 2006). Therefore, a concept drift is typically seen as a change on the data probability distribution, and they can occur gradually or abruptly. The detection of concept drift is essential for the study of data streams, once they allow to point out

changes on the underlying phenomenon, and, in addition, it permits us to change the models being used to represent data.

Considering the fact that data streams naturally evolve, it can happen that a behavior assumed to be normal at a time instant may be abnormal in another. Hence, the labeling of observations become even more complicated, as a behavior may assume different classes according to the time instant it was collected (CAO *et al.*, 2006). Due to the difficulty associated to the labeling of data stream observations, in this thesis, we do not consider supervised or semi-supervised learning algorithms. We instead take *unsupervised learning algorithms* into account.

In general, the detection of concept drift is performed by one of the following approaches (KLINKENBERG; RENZ, 1998): i) performance measures; ii) analysis of the classification model; or iii) analysis of the data properties. The detection through performance measures consider labeled data to quantify concept drift. For example, in the study by Klinkenberg & Renz (1998), the measures of accuracy, recall and precision are computed for every data window, and when such measures abruptly change, a concept drift is issued. The second approach considers changes in the classification model. For example, let a decision tree built from some data, characterizing a normal behavior (GAMA *et al.*, 2010). On every collected data observation, the decision tree model is modified, and this change is examined. If the change exceeds a threshold, a concept drift is issued. These approaches are related to the supervised and semi-supervised learning fields, since they require that at least part of the data should be labeled. As discussed before, we believe the labeling of data streams is unfeasible given the volume and frequency of data collection, then we will focus on unsupervised learning algorithms.

Finally, the third approach uses data properties for detecting concept drifts, which is performed in many different ways. The simplest is by analyzing the statistical properties from successive data windows, supposing some probability distribution function to characterize data. However, this assumption is hard to accept, since the such distribution may also change along time (GAMA *et al.*, 2004). Another popular way considers the analysis of clustering partitions over time, which is detailed in the next section (once it is considered in this thesis).

Although the unsupervised learning field does not require labeled data, it lacks in terms of theoretical formalization, since the field of Statistical Learning Theory guarantees learning only to supervised algorithms (ABU-MOSTAFA; MAGDON-ISMAIL; LIN, 2012). Hence, it is difficult to ensure that a change in the clustering partitions is due data modifications. However, some other studies introduced important theoretical contributions to guarantee learning on this scenario (KLEINBERG, 2002). Among these studies, Carlsson & Mémoli (2010) provide a framework to ensure the data modeling by means of permutation-invariant clustering algorithms. In their proposal, they define a stability property for hierarchical clustering algorithms, in which a data set or any of its order permutations must produce the same partition. This framework also permits the comparison among partitions obtained for different data sets, and guarantee that the

computed divergence between models is caused by data changes. Thus, using this theoretical framework, we can indeed guarantee when a concept drift happened on the data. Their study will be detailed in Chapter 4, once it is part of the foundation for this thesis.

2.4 Related work

Algorithms for detecting concept drift can be divided in three approaches (KLINKENBERG; RENZ, 1998). In the first, they analyze classification performance measures over time. In the second, the properties from the classification model are periodically analyzed. In the last approach, they analyze data properties. Since the first two approaches are related to supervised and semi-supervised learning and as they are not in the context of this thesis, they will be briefly addressed next.

One of the most influential studies from the first approach is by Klinkenberg & Renz (1998), which adapts a window length by computing the median and standard deviation of the supervised measures of accuracy, recall and precision. Depending on how those statistical information change over time, Klinkenberg & Renz (1998) may issue abrupt or a gradual concept drifts on data streams. In the case of abrupt changes, the window length is reduced to a minimal threshold. In the case of gradual changes, the window length is slightly reduced. If no concept drift is detected, the window length is increased.

The proposal by Gama, Fernandes & Rocha (2006) is one of the most cited studies from the second approach, which performs the detection of concept drift based on the properties of a classification model. In their study, data is modeled through an incremental decision tree algorithm, in which each node summarizes data statistics. After receiving a new data observation, the algorithm searches for the most appropriate tree node to be updated. In this process, other visited nodes also have their statistics updated. The detection of concept drift depends on the magnitude of changes happening on the decision tree.

The third approach of concept drift detection is based on the analysis of data properties (SILVA *et al.*, 2013; AGGARWAL, 2013). The simplest manner is by applying statistical tests between data windows. In the case the test evidences enough differences on windows, an concept drift is issued. However, statistical tests have a few requirements that must be followed, otherwise the result will not be reliable. Most of them assume data should be independently sampled or that they belong to some fixed probability distribution function (such as Gaussian), assumptions that are hard to be ensured on the context of data streams.

The third approach is composed of clustering algorithms, which have presented relevant results, as highlighted in the next sections. The first algorithm we present was not properly designed for data streams, however it has a historical relevance, and its data structure and methodology are important for the next algorithms.

2.4.1 BIRCH

The algorithm Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) was developed by Zhang, Ramakrishnan & Livny (1996) for the field of data mining, aiming at summarizing huge amounts of multidimensional numerical data inputs in a single pass. For this, a data structure called Clustering Feature (CF) was developed, responsible for storing three statistics from a set of observations: the number n of observations it summarizes, and its linear (LS) and squared sums (SS). The algorithm summarizes the input data in such CFs, which can be seen as hyperspheres modeling the data set space (the algorithm is capable of estimating the cluster centers and radii based on the CF statistical information). This data structure respects the important property of additivity, which enables their update with a new observation and even the merge with another CF through a single operation. Those CFs are organized in a structure similar to a B-tree, speeding up the search for nodes.

The BIRCH is separated in two phases: in the online, the incoming data observations are assigned to the CF tree. When a new data observation is received by the algorithm, it searches for the most appropriate CF node in the tree. In this process, all the CFs that the algorithm visited have their statistics updated. In the case of a proper CF is not found, the algorithm creates a new one. In the case a CF exceeds the radii threshold, it is divided in two in order to better represent the data space. Moreover, the algorithm periodically checks the whole tree looking for very similar sibling CFs, merging them out. In summary, during the online phase the tree is dynamically built. In the offline phase, the data model is created based on the CFs information – only the tree leaves are considered as the representation of the collected data, and they are grouped using a traditional clustering algorithm, such as K-means, producing the model. Thus, this algorithm stores only few CFs instead of all data set in memory.

2.4.2 CluStream

By analyzing BIRCH, we observe it has no temporal context, therefore it is not suitable for processing data streams. This deficiency was tackled by Aggarwal *et al.* (2003) during the development of the algorithm CluStream, which assumes that the collected data evolve over time and, therefore, concept drifts are likely to happen. In their study, the data structures also store time information, and are called Temporal Clustering Features (CFT). Such structures maintain the same three statistics the CF stores plus the linear and squared sums of the time instants in which the data observations were collected. With this information, the algorithm computes the centers and radii of the CFTs, besides the means and variances of the time instants the observations arrived. As well as BIRCH, this algorithm also has two phases: online and offline.

In the online phase, CluStream manages a number q of CFTs, which is fixed during the whole execution. Parameter q is set according to the available computer memory. As first step, the algorithm produces an initial model by collecting some data observations from the stream,

and clustering them into q hyperspherical groups using K-means, yielding the first model for the q CFTs. Afterwards, for every new collected observation, the algorithm searches for the most appropriate CFT and updates the statistics of all CFTs visited. In the case no suitable CFT is found, a new one is created. However, as the algorithm preserves a fixed number of CFTs, it has: i) to remove an outlier cluster, or ii) merge two very similar CFTs. Periodically, the algorithm also stores snapshots from the current model. Those snapshots are used in the offline phase, when the user wishes to analyze the stream structure over time.

2.4.3 DenStream

The algorithm DenStream, designed by Cao *et al.* (2006), emerged as an answer to the fixed number of clusters proposed by the CluStream. According to the authors, as a data stream may change over time, the number of clusters also can. Besides this difference, the DenStream produces density models in the offline phase, instead of the hyperspherical ones produced by BIRCH and CluStream. In such study, a slightly different data structure is used, named Core-micro-clusters (CMC), which stores both the spatial (for computing the centers and radii of the data set) and the temporal information (the time instants the data observations were collected). Such temporal information is used for computing the cluster weight, or relevance. The algorithm divides the CMCs in two roles: potential and outliers, in which they can swap as the data stream evolves. The difference between the CMC roles is related with their weights – potential CMCs have greater weight. This algorithm also has two phases: online and offline.

In the online phase, when a data observation is received, the algorithm searches for the closest potential CMC c_p, and its information is updated. In the case this closest potential c_p exceeds a spatial threshold after being updated, the algorithm searches for the closest outlier CMC, referred to as c_o. If the data observation is inside the radius of the CMC c_o, its information is updated. However, if the spatial threshold is again exceeded, a new outlier CMC is created with the information provided by this single data observation. In attempt to preserve an acceptable number of CMCs in memory, the algorithm periodically reduces the weight of them all (working as an aging factor). If a CMC has its weight reduced under a threshold, it is removed. In the offline phase, the potential CMCs are grouped by a traditional density-based clustering algorithm. This algorithm is a modified version of DBSCAN (ESTER *et al.*, 1996), which uses the weight information from the CMCs to produce the model.

2.4.4 ClusTree

The ClusTree, proposed by Kranen *et al.* (2011), is another algorithm from the BIRCH's family and one of the most cited recently. This algorithm has the characteristics of being nonparametric, it manages a variable number of clusters, and it automatically adapts to the frequency of the data stream. Similarly to the other algorithms, ClusTree also uses CFTs as main data structure, which are organized in an R tree. According to the authors, this type of tree is

more effective in the search operation. ClusTree differs from the others in terms of: i) being incremental, i.e., it always maintains an updated model (hence it does not have an offline phase); and ii) it has the characteristic of being anytime, that is, the algorithm produces models with different qualities depending on the data stream frequency.

2.4.5 M-DBScan

The algorithms presented so far are based on the BIRCH and, therefore, are used for reducing data streams into clustering partitions. They may detect concept drifts by monitoring the online phase, that is, when the algorithm receives a new data observation and while managing the CFs. When the CF tree changes, by adding or removing CFs, they can issue a concept drift. This idea was carried out by Vallim *et al.* (2013) on the algorithm M-DBScan, an extension of the DenStream. During the online phase, the algorithm monitors metrics of entropy (SHANNON, 1948) based on the operations on the CF tree. In the case the entropy above a given threshold, a drift is issued. Two characteristics are important in this algorithm: i) the entropy threshold is adaptive; and ii) a concept drift is detected only when several data observations confirm such need, avoiding outliers.

In our point of view, M-DBScan uses a coherent method for detecting concept drift on data streams. However, those detections could be inaccurate as the BIRCH-family algorithms do not consider temporal relations embedded on data observations. They manage data observations as they were produced independently over time, processing them one by one. Moreover, those algorithms have no theoretical guarantees, but relevant empirical results.

2.4.6 GK

Other clustering algorithms process data streams in a single phase, unlike BIRCH and its family. These algorithms are called incremental as they update the clusters as every data observation is received, keeping an up-to-date model. An example is the Growing K-means (GK), proposed by Daszykowski, Walczak & Massart (2002), which is an adaptation of K-means (MACQUEEN *et al.*, 1967). This algorithm initially sets up a space with two random centers. When a data observation is received, the algorithm updates the nearest cluster center following a learning rate. After the modeling of n data observations, a statistical analysis takes place, and new cluster centers can be added or removed. The algorithm keeps this process, gradually reducing its learning rate, up to the point that c clusters are obtained.

2.4.7 SONDE

Another incremental algorithm is the Self-Organizing Novelty Detection (SONDE), proposed by Albertini & Mello (2007). This algorithm uses neurons to hyperspherically group the observations of a data stream. When a new observation is received, the algorithm searches

for the neuron with the greatest activation to represent it. If the activation is above a threshold, the center, radius and similarity degree of such neuron is adapted. Otherwise it creates a new neuron, issuing a concept drift.

These last two algorithms detect concept drifts by analyzing changes in the algorithm model, after a new data observation is received. For example, when the algorithm creates a cluster or a neuron, it may warns about a concept drift. However, as these algorithms are incremental, their models are greatly influenced by the learning parameters chose by the user. Hence, depending on those parameters, they can produce very different models even for the same data stream, consequently detecting concept drift on different time instants. Therefore, those algorithms have no theoretical guarantee about the concept drifts issued. Moreover, there are still other problems, such as the initialization of the algorithm Growing K-means (GK), which sets its initial model based on two randomly selected cluster centers, allowing the generation of different models for the very same data stream.

2.4.8 Final considerations

In this chapter, we discussed about supervised and unsupervised learning algorithms used to detect concept drift on data streams. In particular, the unsupervised ones accomplish the task of summarizing data streams in just one pass, with low computational complexity and using limited memory. However, they face several issues, to mention: i) the temporal relationship among data observations is loosely considered (some algorithms even assume data as independently sampled); and ii) in some cases the initial model is randomly generated. As consequence of these deficiencies, the generated models do not appropriately represent data, since they do not consider temporal relationships and different models can be generated for the very same data stream. As a conclusion, those algorithms have no theoretical guarantee that a detection issued is in fact associated to a data stream change.

CHAPTER

3

TIME SERIES ANALYSIS

"The more we study the major problems of our time, the more we come to realise that they cannot be understood in isolation. They are systemic problems, which means that they are interconnected and interdependent."

Frijot Capra

3.1 Initial considerations

This chapter presents concepts on discrete time series analysis considering linear and nonlinear approaches, which focus is to extract implicit information from time series taking data relationships into account. While the branch of linear time series analysis is mostly based on Statistics, the nonlinear has its roots in the area of Dynamical Systems. The next sections introduce the fundamental concepts associated to both branches as well as their tools. We finish the chapter detailing the main tools in the literature for detecting dynamical changes on nonlinear time series.

3.2 Time series

A time series is a set of observations organized over time, which is produced by some phenomenon of interest, such as meteorological data, stock market indices, chemical processes, etc. (MORETTIN; TOLOI, 2004). In this thesis, we consider only discrete time series, unidimensional and whose observations were generated at a fixed and equidistant interval h, i.e., uniformly sampled. In this sense, a time series with observations $s_1, s_2, \ldots, s_t, \ldots, s_n$ denotes

data obtained at the following time instants $\tau_{0+h}, \tau_{0+2h}, \ldots, \tau_{0+th}, \ldots, \tau_{0+nh}$, in which τ_0 is the time origin and h is the sampling interval. From this, a time series with n observations is here defined as $S_n = \{s_1, s_2, \ldots, s_t, \ldots, s_n\}$.

Time series produced by a real-world phenomenon typically presents some temporal dependency (BOX; JENKINS, 1976; MORETTIN; TOLOI, 2004), so an observation at time t is influenced by previous $t-j, j \in \mathbb{N}$ ones. Considering this assumption, the area of time series analysis provides tools for studying such relationships. Based on this study, one can model time series to comprehend the generation process behind it, i.e., the function that produced the observations, and consequently, understand the underlying phenomenon (MORETTIN; TOLOI, 2004).

3.3 Time series analysis using ARIMA models

In this section, we introduce time series analysis proposed by Box & Jenkins (1976), which allows us to represent the relationships among observations for both stationary and nonstationary time series. According to their methodology, a time series is modeled using three steps. First, it is identified the most adequate model for the time series, using one of the four approaches which are part of the ARIMA models (BOX; JENKINS, 1976). Next, the model parameters are estimated based on the time series relationships. Then, the model quality is assessed in order to carry on improving its suitability. This process may be repeated until an adequate representation is obtained. All those steps also consider the concept of parsimony to produce representative models using few parameters as possible, according to the Occam's Razor principle (ABU-MOSTAFA; MAGDON-ISMAIL; LIN, 2012), and thus avoiding overfitting (LUXBURG; SCHÖLKOPF, 2009).

Box & Jenkins (1976) formalized the ARIMA models, which are the building blocks for their methodology of time series analysis. Next sections present all four ARIMA models, in which the first three are used for linear and stationary time series and, the last, for linear and nonstationary time series (with a linear trend). Those models are defined based on the time series $\tilde{S}_n = \{\tilde{s}_0, \ldots, \tilde{s}_t, \ldots, \tilde{s}_n\}$, which is the original series S_n subtracted from its average μ, i.e., $\tilde{s}_t = s_t - \mu, t = 1, \ldots, t, \ldots, n$. Hence, $\tilde{s}_t$ correspond to the fluctuations of the time series observations around the average.

3.3.1 Autoregressive model

In the Autoregressive model of order p, or $AR(p)$, the value of the current observation $\tilde{s}_t$ is defined using a finite linear system of previous values summed to a random variable a_t, as follows:

$$\tilde{s}_t = \phi_1 \tilde{s}_{t-1} + \phi_2 \tilde{s}_{t-2} + \ldots + \phi_p \tilde{s}_{t-p} + a_t, \tag{3.1}$$

in which the symbols $\phi_1, \phi_2, \ldots, \phi_p$ correspond to the finite set of weights for the past observations $\tilde{s}_{t-1}, \tilde{s}_{t-2}, \ldots, \tilde{s}_{t-p}$.

This model is helpful for adjusting Economics and Geophysics time series, in which observations have some linear time dependency from the previous values (MORETTIN; TOLOI, 2004).

3.3.2 Moving average model

In the Moving Average model of order q, or $MA(q)$, the value of the current observation $\tilde{s}_t$ is defined for a finite linear system of previous random variables, as follows:

$$\tilde{s}_t = a_t - \theta_1 \tilde{a}_{t-1} - \theta_2 \tilde{a}_{t-2} - \ldots - \theta_q \tilde{a}_{t-q}, \tag{3.2}$$

in which $\theta_1, \theta_2, \ldots, \theta_q$ is the finite set of weights for the past random variables $\tilde{a}_{t-1}, \tilde{a}_{t-2}, \ldots, \tilde{a}_{t-q}$.

This model is not as useful as the others when dealing with time series produced by natural phenomena (MORETTIN; TOLOI, 2004). However, they can be used for simulating noise with some time dependency.

3.3.3 Autoregressive and moving average model

The Autoregressive and Moving Average model, or $ARMA(p,q)$, is defined by coupling the models $AR(p)$ and $MA(q)$. Thus, the value of the current observation $\tilde{s}_t$ is defined using a finite linear system of order p from the previous series observations, summed to a finite linear system of order q from the past random variables, as follows:

$$\tilde{s}_t = \phi_1 \tilde{s}_{t-1} + \ldots + \phi_p \tilde{s}_{t-p} + a_t - \theta_1 a_{t-1} - \ldots - \theta_q a_{t-q}. \tag{3.3}$$

This model is useful for adjusting to various real-world time series, since it considers past observation values and other stochastic influences due to the coupling of models $AR(p)$ and $MA(q)$. As discussed in (MORETTIN; TOLOI, 2004), this model supports the analysis of more complicated time series.

3.3.4 Autoregressive integrated and moving average model

The Autoregressive Integrated and Moving Average model, or $ARIMA(p,d,q)$, is used to represent nonstationary time series with some linear trend. It considers a stationary time series W_n, which is resultant of a d-order difference from a nonstationary time series S_n, such that $w_t = \nabla^d s_t$. In summary, after such difference, the model $ARMA(p,q)$ is used for analyzing series W_n. In this scenario, the current observation $\tilde{w}_t$ is defined as follows:

$$\tilde{w}_t = \phi_1 \tilde{w}_{t-1} + \ldots + \phi_p \tilde{w}_{t-p} + a_t - \theta_1 a_{t-1} - \ldots - \theta_q a_{t-q}, \tag{3.4}$$

in which $\tilde{w}_t = \nabla^d \tilde{s}_t$.

This model presents better performance in Economics and Financial studies, in which nonstationary time series become stationary after differentiation (MORETTIN; TOLOI, 2004).

3.4 Nonlinear time series analysis

In this section, we describe methods and tools for analyzing nonlinear time series considered by the field of Dynamical Systems, in order to estimate models. In this field, a current data observation is represented only based on the previous ones [1](ALLIGOOD; SAUER; YORKE, 1997). When models are obtained, one can understand, monitor or even control the underlying phenomenon (OTT; SAUER; YORKE, 1994). Next sections introduce the main concepts.

3.4.1 Essential concepts

A dynamical system is defined as a set of equations that produce a time evolution between states, based on past states (OTT; SAUER; YORKE, 1994). By definition, a state contains enough information to represent current system situation and to evolve it to a next state. Most of the dynamical systems assume that such evolution is performed by a purely deterministic function (or rule, a.k.a. *generation process*), i.e., it only depends on the previous state to obtain the current one without ambiguity. Formally, a state is defined by a variable $\mathbf{x}_t \in \mathbb{R}^m$, and the set of states is referred to as the phase space, denoted by Γ , i.e., $\mathbf{x}_t \in \Gamma, t = 1, \ldots, t, \ldots, n$. These states evolve in the dynamical system following a rule f, whose domain and image are given by Γ, i.e., $f : \Gamma \to \Gamma$. Therefore, function f evolves the state $\mathbf{x}_t$ unambiguously to $\mathbf{x}_{t+1}$, in form $f(\mathbf{x}_t) = \mathbf{x}_{t+1}$.

Given function f, an initial state $\mathbf{x}_0$ of the dynamical system can be sequentially iterated to result in $\mathbf{x}_1$. At a second iteration, given by $f(\mathbf{x}_1)$, or $f(f(\mathbf{x}_0)) = f^2(\mathbf{x}_0)$, one can obtain the next state $\mathbf{x}_2$. Generally speaking, the k-*th* iteration from the initial state $\mathbf{x}_0$ is given by $f^k(\mathbf{x}_0) = \mathbf{x}_k$. For example, let $\mathbf{x}_0 = 1$ be the initial state and $f(\mathbf{x}) = 2\mathbf{x}$ be a function, the first iteration will result in $f(1) = 2$; the second produces $f(f(1)) = f(2) = 4$; the third will result in $f^3(1) = f(4) = 8$; and so on. After k iterations, a set of states $\{\mathbf{x}_0, f(\mathbf{x}_0), f^2(\mathbf{x}_0), \ldots, f^k(\mathbf{x}_0)\}$ is obtained, which is referred to as the trajectory from the initial state $\mathbf{x}_0$. If this trajectory is arranged over time, it can be seen as a time series S_n.

Trajectories may have special states, such as fixed states, that occurs when the function f produces the same previous state $\mathbf{p}$ as the current one, i.e., $f(\mathbf{p}) = \mathbf{p}$. This kind of state is said to be 1-periodic, since after one iteration over $\mathbf{p}$, the same state is obtained. Likewise, fixed points with other periodicity exists. For example, in the case two states $\mathbf{p}_1$ and $\mathbf{p}_2$ sequentially switch between each other after the evolution of a dynamical system, such that $f(\mathbf{p}_1) = \mathbf{p}_2$

[1] In the context of this thesis, we only consider deterministic Dynamical Systems.

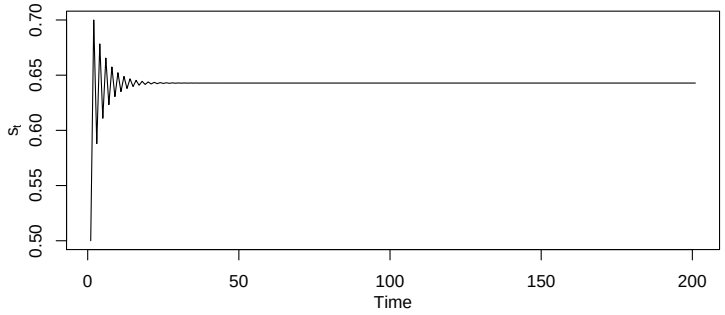
and $f(\mathbf{p}_2) = \mathbf{p}_1$, such states are called 2-periodic. In general, a fixed state of periodicity k, or k-periodic, is produced when $f^k(\mathbf{p}) = \mathbf{p}$. Such fixed states are important in the analysis of time series, since they provide information about *recurrences* of the underlying phenomenon.

After iterating many different initial states on a dynamical system, different trajectories are obtained. These trajectories have relevant information about the system under study, as regions from the phase space may converge over time. Such regions are called *attractors*, which are found while studying fixed points. Consider $\mathbf{p}$ a fixed point of dimension m, i.e., $\mathbf{p} = (p_1, \ldots, p_m) \in \Gamma$, and ε a positive number. The ε-neighborhood of $\mathbf{p}$, called $N_\varepsilon(\mathbf{p})$, defines the subset of states that present a distance less than ε regarding the fixed point $\mathbf{p}$, i.e., $N_\varepsilon(\mathbf{p}) = \{\mathbf{v} \in \Gamma \text{ such that } \|\mathbf{v}-\mathbf{p}\| < \varepsilon\}$. If all points $\mathbf{v} \in N_\varepsilon(\mathbf{p})$ converge to $\mathbf{p}$ over time, i.e., $\lim_{k\to\infty} f^k(\mathbf{v}) = \mathbf{p}$, the fixed point or state $\mathbf{p}$ is considered an attractor, and the set $N_\varepsilon(\mathbf{p})$ is referred to as the basin of attraction (ALLIGOOD; SAUER; YORKE, 1997). These convergence regions provide relevant information about the recurrence of the system.

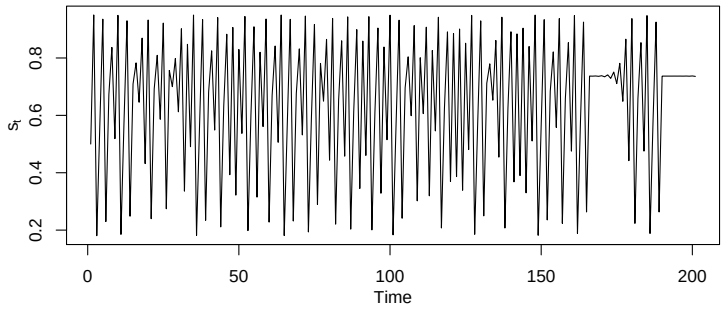
As example of an one-dimensional dynamical system, consider the Logistic Map, whose function is defined as $f(x_t) = r \cdot x_t(1-x_t)$, in which x_t corresponds to the current state and $r \in \mathbb{R}$ is the parameter responsible for different system behaviors. It is worth to mention that those states x_t are denoted as scalars instead of as vectors since it is an one-dimensional dynamical system. Now, when one sets the parameter $r = 2.8$ for a Logistic Map, i.e., $f(x_t) = 2.8 \cdot x_t(1-x_t)$, and the initial state to $x_0 = 0.5$, and iterates it 200 times, a time series as the one plotted in Figure 1a is obtained. We can observe the behavior of this time series converges to a state close to the value of 0.64, which is an attractor (and also a fixed point). Considering another Logistic Map, now with $r = 3.8$, i.e., $f(x_t) = 3.8 \cdot x_t(1-x_t)$, for the same initial state $x_0 = 0.5$ and the same number of iterations, another time series is obtained, as illustrated in Figure 1b. In this case, the time series behavior seems to be completely stochastic, even though the dynamical system is purely deterministic. In order to observe the numerical convergence trends, some states from both trajectories are listed in Table 1.

An important concept in the field of Dynamical Systems is *chaos* (OTT; SAUER; YORKE, 1994). A trajectory is called chaotic if it has a sensible dependency on the initial conditions. For example, consider that a system was initialized with two different states, $\mathbf{x}_0$ and $\mathbf{x}'_0$, such that the distance between them is infinitesimal $\Delta\mathbf{x}_0$, i.e., $\mathbf{x}'_0 = \mathbf{x}_0 + \Delta\mathbf{x}_0$. Let the iteration of such initial states on the dynamical system, and the difference between their trajectories being computed, i.e., $\Delta\mathbf{x}_t = \mathbf{x}'_t - \mathbf{x}_t$. If $\Delta\mathbf{x}_t$ grows exponentially over time, it is said that this system is sensible to initial conditions and, therefore, the attractor is chaotic. As example, consider the Logistic Map presented in Figure 1b, having $r = 3.8$, and two initial states, $x_0 = 0,5$ e $x'_0 = 0,5 + \Delta x_0$, using $\Delta x_0 = 10^{-7}$. Figure 2 shows both trajectories, in which we can observe that after 70 iterations, they develop very different behaviors, characterizing a chaotic attractor.

Based on the presented concepts, next section introduces the technique of reconstructing phase spaces from time series, which is considered of major importance in the analysis of



(a) Logistic Map produced with parameters $r = 2.8$.



(b) Logistic Map produced with parameters $r = 3.8$.

Figure 1 – Illustration of the produced states after the iteration of the Logistic Map, in which each map uses a different value of r. The produced states are presented in the time domain, characterizing them as time series. Both maps were produced using as initial state $x_0 = 0.5$ which was iterated 200 times. Map a uses $r = 2.8$, and has a convergent behavior to a state close to 0.64. On the other hand, Map b uses $r = 3.8$ and produces a behavior which may be mistakenly referred to as stochastic.

Table 1 – Examples of states obtained after the iteration of the Logistic Maps illustrated in Figures 1a and 1b, respectively. Note that the values for the first map converge to a state close to 0.6428, while the second does not.

Index	$r = 2.8$	$r = 3.8$
0	0.5000	0.5000
1	0.7000	0.9500
2	0.5880	0.1805
3	0.6783	0.5620
4	0.6109	0.9353
10	0.6350	0.1850
20	0.6420	0.2396
30	0.6427	0.9020
40	0.6428	0.8249
50	0.6428	0.5371
100	0.6428	0.6562
150	0.6428	0.8094
200	0.6428	0.5462

nonlinear time series (KANTZ; SCHREIBER, 2004). This is the main tool to understand and model the dynamics of a system, i.e., to unfold the time dependencies among data observations.

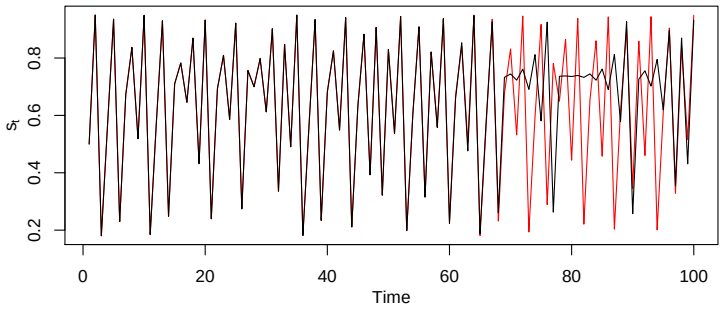


Figure 2 – Two trajectories generated for the same Logistic Map with $r = 3.8$, in which two initial states with very close values were used: 0.5 and 0.5000001. Observe that after 70 iterations, the trajectories are completely different, characterizing the chaotic attractor.

3.4.2 Phase space reconstruction

A sequence of observations obtained from the time domain, also known as time series, can be reconstructed into the domain of states (or phase space), allowing us to analyze its trajectories in order to find periodicities and other information that support the study of a phenomenon. This phase space is reconstructed taking into account the delay embedding theorem by Takens (1981), which unfolds the states $\{\mathbf{x}_0, \ldots, \mathbf{x}_t, \ldots, \mathbf{x}_n\}$ based on an *unidimensional* time series $S_n = \{s_0, \ldots, s_t \ldots, s_n\}$.

For illustration purposes, consider the iterations of a Logistic Map with chaotic behavior as presented in Figure 1b. Consider the trajectory was obtained over time, thus it can be accepted as a pure deterministic time series, denoted as S_n. The phase space of this time series, reconstructed by the embedding theorem, is presented in Figure 3. Even though such attractor is chaotic, and the time series is similar to a stochastic process, one can observe that the phase space has a regular behavior. Through the phase space reconstruction, one can obtain more information about the relationship among observations, what permits data modeling and prediction at greater accuracy (OTT; SAUER; YORKE, 1994; ALLIGOOD; SAUER; YORKE, 1997).

The reconstruction of the phase space Γ is fulfilled by unfolding the states $\mathbf{x}_t$, which are composed of the time series observations s_t. In the previous example, states have two dimensions, that is, two time series observations were composed to create each state, separated by one time unit. In this case, the phase space was reconstructed using the embedding dimension $m = 2$, corresponding to each state dimension, and the time delay $\tau = 1$, corresponding to the time separation between observations. In summary, the association between states and observations is given by $\mathbf{x}_t = (s_t, s_{t-1})$. Table 2 lists some states from this phase space.

Generally speaking, the reconstruction of the phase space needs two parameters: the embedding dimension m, which defines the number of observations used to compose a state or the number of phase space axes; and the time delay τ, which defines the time separation among observations. Having those parameters, one can reconstruct the phase space by employing

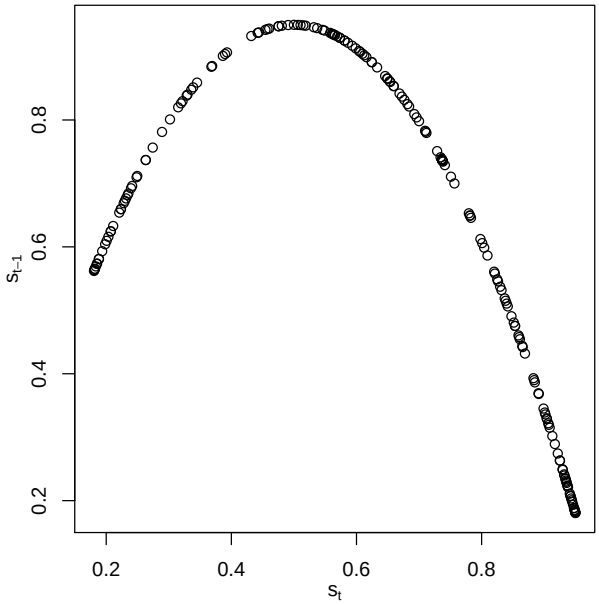


Figure 3 – Phase space reconstructed for the time series obtained using the Logistic Map from Figure 1b. Note that the behavior of this phase space is regular, unlikely the Logistic Map along time. The x-axis corresponds to observations at time instant t and the y-axis to observations at time $t-1$.

Table 2 – Examples of states x_t from the phase space reconstructed for the completely deterministic time series based on the Logistic Map, illustrated in Figure 1b, which has a chaotic behavior.

t	s_t	s_{t-1}
1	0.5000	0.9500
2	0.9500	0.1805
3	0.1805	0.5620
4	0.5620	0.9353
5	0.9353	0.2297

the Takens (1981)' embedding theorem:

$$\mathbf{x}_t = \left(s_t, s_{t-\tau}, s_{t-2\tau}, \ldots, s_{t-(m-1)\tau}\right) \in \mathbb{R}^m, \tag{3.5}$$

in which $\mathbf{x}_t \in \Gamma$ is a state from the reconstructed phase space and $s_t \in S_n$ is an observation from a discrete and unidimensional time series.

In this section, we introduced the embedding theorem by Takens (1981), which is the main tool for reconstructing phase spaces from time series. In the following section, we present approaches for estimating the parameters involved in this theorem: the time delay and the embedding dimensions.

3.4.2.1 Estimating the time delay

The time delay τ is also known as the dimension associated to the time lag among observations, which supports the unfolding of states in the phase space by associating dependent observations. By choosing an arbitrary value for this dimension, the phase space obtained may

be completely useless. If τ is too small, the observations s_t and $s_{t-\tau}$ composed to form a given state may be too similar to one another, producing insignificant states. On the other hand, if τ is greater than necessary, the correlation between observations may be null, producing insignificant states as well. Such consequences are magnified on reconstructed phase spaces from time series generated by real-world phenomena. Therefore, it is necessary to have a method for estimating suitable values for this dimension (OTT; SAUER; YORKE, 1994).

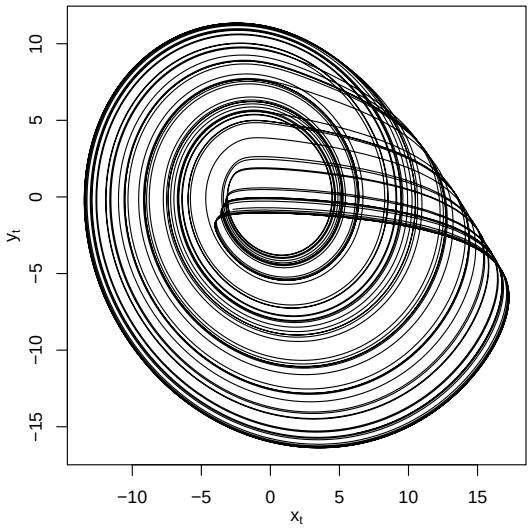
In the past, it was usual to estimate the time delay based on the result of the Autocorrelation Function (ACF) applied on a time series S_n. In that situation, the lag was set as the first coefficient value for which the ACF resulted in zero. However, the ACF admits only linear dependencies, what cannot be *a priori* assumed for any time series. This limitation motivated Fraser & Swinney (1986) to use the Mutual Information (MI) function to estimate the time delay. Such function, originally from the field of Information Theory (SHANNON, 1948), measures the general dependency between variables, and it can be applied to both linear and nonlinear time series. Hence, Fraser & Swinney (1986) estimate the best value for τ by setting it as the first local minimum of the Mutual Information function, while assessing observations at different time displacements.

For illustration purposes, we will reconstruct phase spaces for different time delays based on the 3D attractor by Rössler (1976), defined by the following set of equations:

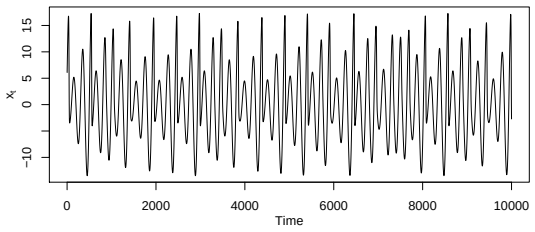
$$\begin{aligned}\dot{x} &= -z - y,\\ \dot{y} &= x + ay,\\ \dot{z} &= b + z(x - c),\end{aligned}$$

in which x, y and z correspond to the position of the current state, $\dot{x}$, $\dot{y}$ and $\dot{z}$ are associated to state displacements at the time instant t, and a, b and c are system parameters. The initial state was set to $(x_0, y_0, z_0) = (10, 0, 0)$ and the parameters set to $a = 0.2$, $b = 0.2$ and $c = 8$. The system was iterated $11,000$ times with the time interval of $\pi/100$, and the first $1,000$ states were discarded to ensure that only the basin of attraction was plotted. The displacements $\dot{x}$, $\dot{y}$ and $\dot{z}$ were organized along the time domain, characterizing them as time series X_n, Y_n and Z_n, respectively. Figure 4 presents: a) the attractor in its bidimensional projection, only considering the x and y dimensions; b) time series X_n; c) time series Y_n; and d) time series Z_n.

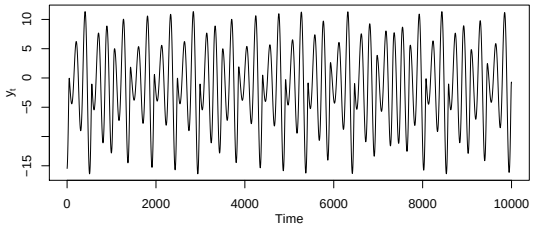
Consider that only the time series X_n is observable, and the other two (Y_n and Z_n) are not for some reason. As such time series is influenced by the other variables (recalling, $\dot{x} = -z - y$), it is possible to reconstruct Rössler (1976)'s attractor, based only on X_n. First, the Mutual Information function is applied on time series X_n, for which the results are shown in Figure 5. According to this plot, the y-axis corresponds to the Mutual Information (MI) value, in bits, considering different time delays (at the x-axis) in interval $[0, 700]$. As one can observe, the greatest MI occurs when the time delay is zero, i.e., when the observation s_t is compared to itself. Following the idea by Fraser & Swinney (1986), the first local minimum represents the best



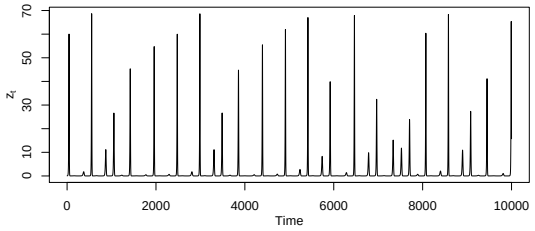
(a) Bidimensional projection of the Rössler (1976)'s attractor.



(b) Plot obtained for the observations produced by equation $\dot{x}$ along the time domain.



(c) Plot obtained for the observations produced by equation $\dot{y}$ along the time domain.



(d) Plot obtained for the observations produced by equation $\dot{z}$ along the time domain.

Figure 4 – Plots for the Rössler (1976)'s attractor: in which (a) is the attractor in its bidimensional projection, only considering the $x-$ and $y-$ axis; in the other plots, the observations produced by each dimension along the time domain: X_n, Y_n and Z_n, respectively.

value for the time delay, therefore, for this case, $\tau = 47$.

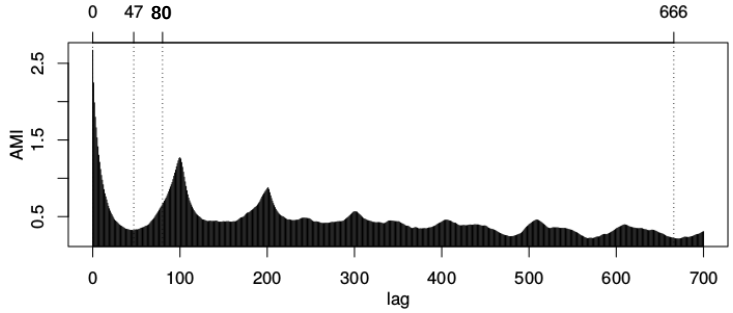


Figure 5 – Mutual Information function applied on the time series X_n for the Rössler (1976)'s attractor, taking different time delays into account. The highlighted positions refer to the following time delays: 0, in which the greatest MI is obtained; 47, which corresponds to the first local minimum and, consequently, to the selected time delay according to Fraser & Swinney (1986); 80, an arbitrary value; and 666, the last local minimum in the studied range $[0, 700]$. These values will be used to exemplify the reconstructed phase space using different time delays, as shown in Figure 6.

The highlighted values from Figure 5, i.e., 0, 47, 80 and 666, are used to exemplify how

the phase space is diversely reconstructed for the time series X_n when different time delays are considered. Firstly, let the time delay be set as $\tau = 0$, which produces a phase space as a straight line (Figure 6a). In that situation, every state is defined as $\mathbf{x}_t = (s_t, s_t)$, providing no information about the system under study. The second phase space, presented in Figure 6b, was reconstructed considering $\tau = 47$, which according to Fraser & Swinney (1986) is the best estimation for the time delay (first minimum). One can observe that the reconstructed phase space is very similar to the original Rössler (1976)'s attractor, previously shown in Figure 4a. The third phase space, illustrated in Figure 6c, considered the arbitrary value of $\tau = 80$ for the time delay, presenting many intersecting trajectories that jeopardize further analysis, as it produces ambiguities on states of the dynamical system. Finally, the phase space illustrated in Figure 6d was reconstructed using $\tau = 666$, i.e., the last local minimum obtained for the Mutual Information function in range $[0, 700]$. Besides the phase space does not contain many intersecting trajectories, they are entangled, what also risks the analysis.

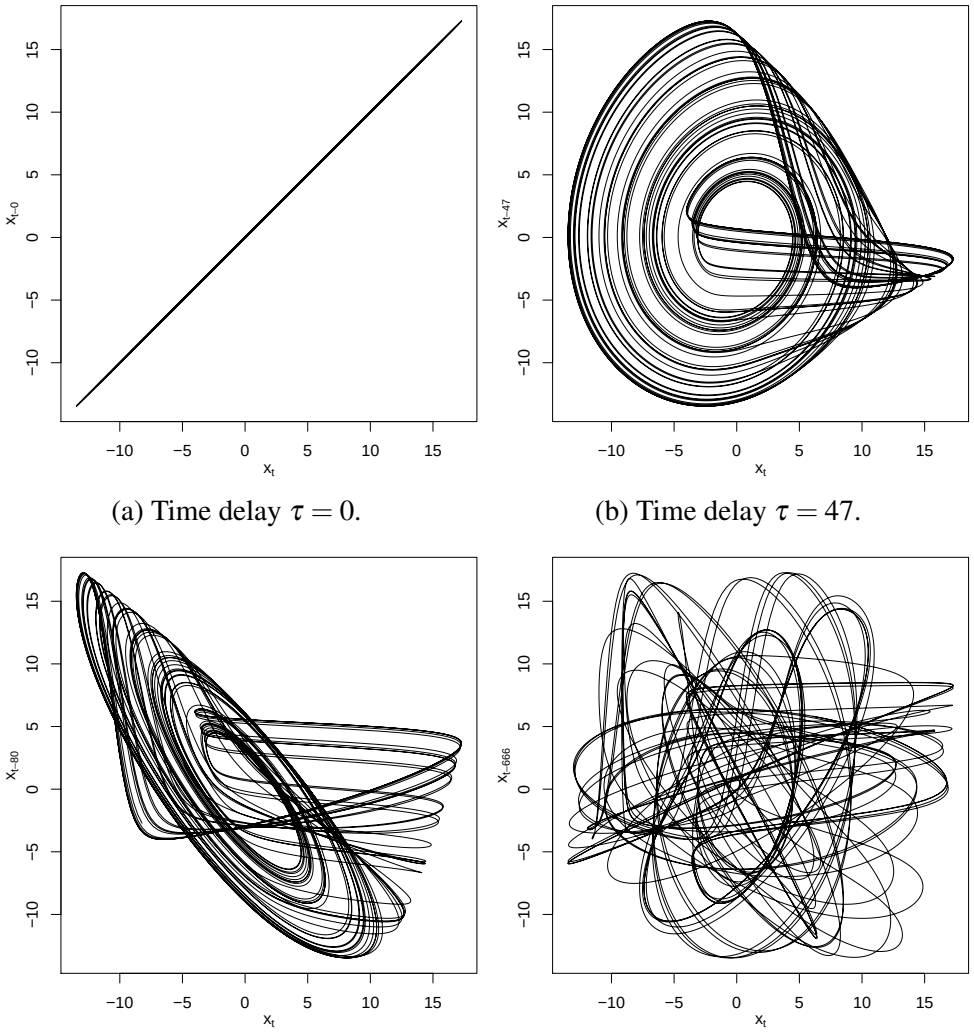
3.4.2.2 Estimating the embedding dimension

While reconstructing a phase space from a time series, it is important to select an appropriate value for the embedding dimension m , in such a way that the obtained trajectories are not entangled after the reconstruction. Figure 7 illustrates two examples of phase spaces, in which the first produces entangled trajectories and the second does not. Such entanglement violates the definition of state, which must have enough information to evolve the dynamical system in an *unambiguous* way. Increasing the embedding dimension, the phase space is untangled and the trajectory intersections tend to disappear (ALLIGOOD; SAUER; YORKE, 1997).

Consider m_E as the smallest but enough value for the embedding dimension that produces a meaningful phase space, i.e., in which the states permit evolving the dynamical system without ambiguity. If the embedding dimension m is set such that $m < m_E$, the reconstructed phase space will contain entangled trajectories. On the other hand, if $m > m_E$, trajectories will already be untangled after $m = m_E$ and unnecessary additional dimensions will be considered. From the mathematical point of view, for any $m \geq m_E$ the obtained attractor will be unfolded (as discussed in (KENNEL; BROWN; ABARBANEL, 1992)), however from the computational point of view, any embedding dimension m greater than m_E will imply unnecessary complexity to the phase space, as well as it will increase the computational time. Therefore, it is crucial to obtain the smallest but enough embedding dimension m.

In order to illustrate the importance of estimating the embedding dimension m, we now use the 3D attractor by Lorenz (1963), defined by the following system of equations:

$$\begin{aligned}
\dot{x} &= \sigma(y - x), \\
\dot{y} &= x(\rho - z) - y, \\
\dot{z} &= xy - \beta z,
\end{aligned}$$



(a) Time delay $\tau = 0$.

(b) Time delay $\tau = 47$.

(c) Time delay $\tau = 80$.

(d) Time delay $\tau = 666$.

Figure 6 – Phase space reconstructed from the time series X_n for the Rössler (1976)'s attractor using different time delays (τ). In (a), the phase space is a straight line, providing no information about the dynamical system under study. In (b), the first local minimum for the Mutual Information is used, which is the best estimation for the time delay according to Fraser & Swinney (1986). This result is the closest to the original attractor (Figure 4a). In (c), the phase space was reconstructed using an arbitrary value, producing few entangled trajectories. Finally, in (d), the phase space has many entangled trajectories, risking any sort of analysis.

in which x, y and z define the current state, $\dot{x}$, $\dot{y}$ and $\dot{z}$ correspond to state displacements at the time instant t, and σ, ρ and β are system parameters. The initial state was set to $(x_0, y_0, z_0) = (0.1, 0.1, 0.1)$ and the parameters set to $\sigma = 10$, $\rho = 28$ and $\beta = \frac{8}{3}$. The system was iterated $11,000$ times and the time interval was set as 10^{-2}. The first $1,000$ states were removed to ensure that only the basin of attraction was used. The displacements $\dot{x}$, $\dot{y}$ and $\dot{z}$ were organized along the time domain, characterizing them as time series X_n, Y_n and Z_n, respectively. Figure 8 shows: a) the 3D attractor; b) observations obtained for time series X_n; c) time series Y_n; and d) time series Z_n.

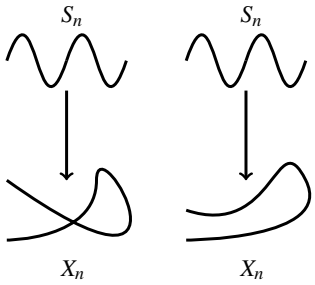
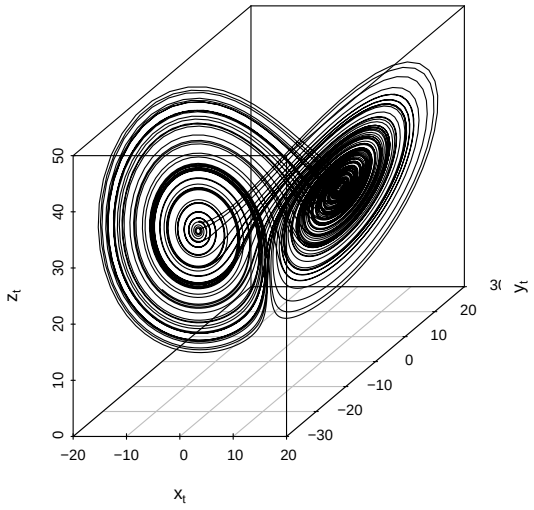
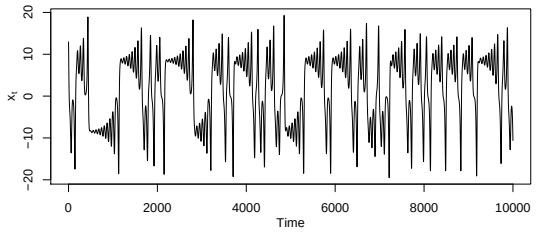


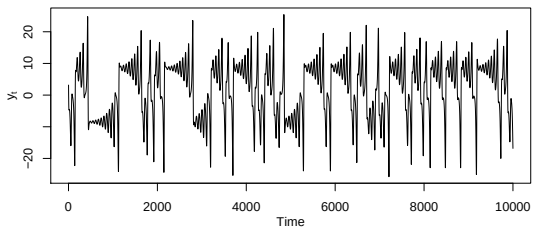
Figure 7 – Example of an unfolded phase space. The objective of the phase space reconstruction is to obtain untangled trajectories, as illustrated on the right (adapted from Ott, Sauer & Yorke (1994)).



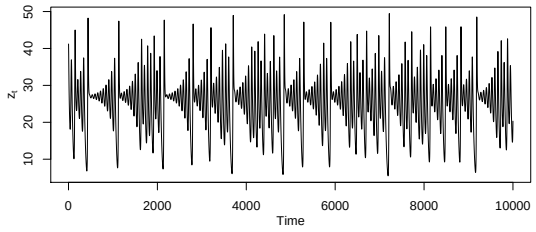
(a) Three-dimensional space obtained for Lorenz (1963)'s attractor.



(b) Observations produced by equation $\dot{x}$ along the time domain.



(c) Observations produced by equation $\dot{y}$ along the time domain.



(d) Observations produced by equation $\dot{z}$ along the time domain.

Figure 8 – Phase space for the Lorenz (1963)'s attractor: in which (a) is the attractor in its three-dimensional space; other plots show each time series: X_n, Y_n and Z_n, respectively.

We will consider the phase space reconstruction using only the time series X_n, which is illustrated in Figure 8b. As first step, Mutual Information is employed to estimate the time delay as described in Section 3.4.2.1. Figure 9 shows the results for the Mutual Information using different time delays, in which the first local minimum $\tau = 16$ is selected as the time delay (FRASER; SWINNEY, 1986).

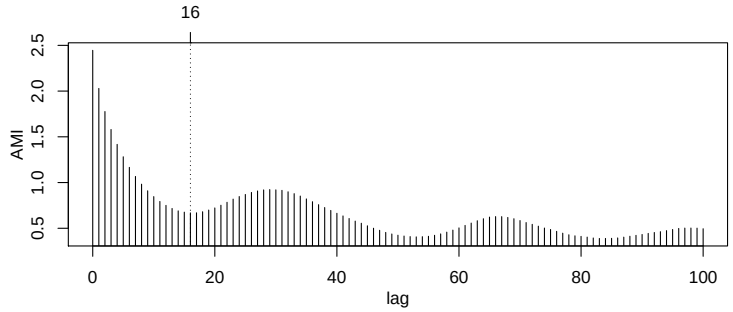
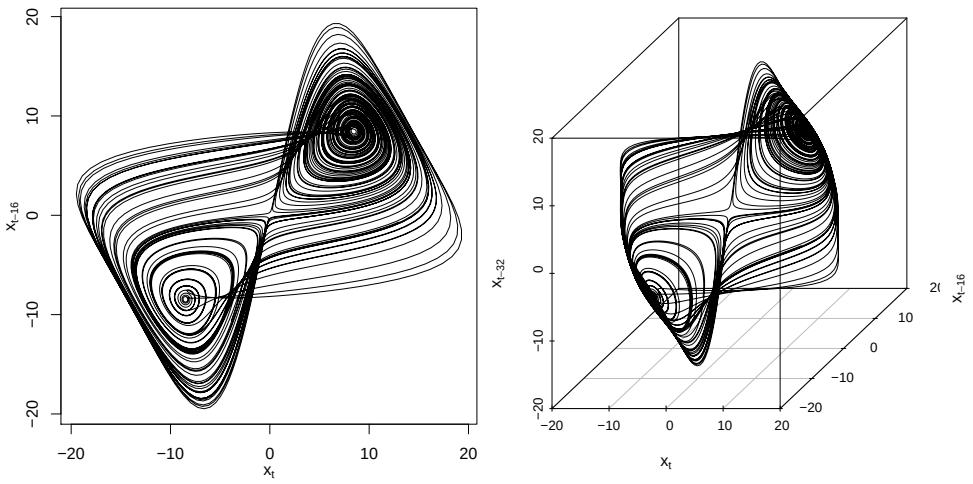


Figure 9 – Mutual Information applied on time series X_n for the Lorenz (1963)'s attractor, given different time delays. The first minimum, 16, is highlighted, which corresponds to the best estimative according to Fraser & Swinney (1986).

Next, two scenarios are used for illustrating differences in the phase spaces reconstructed under different embedding dimensions. The first phase space is reconstructed using $m = 2$, as shown in Figure 10a, which still contains many entangled trajectories, i.e., resulting in ambiguous states. Therefore, this embedding dimension is not enough to unfold the phase space. On the other hand, Figure 10b corresponds to the phase space reconstructed with embedding dimension $m = 3$, which untangles all trajectories and, therefore, is more appropriate.



(a) Embedding dimension $m = 2$. (b) Embedding dimension $m = 3$.

Figure 10 – Phase space reconstructed using time series X_n for the Lorenz (1963)'s attractor, considering different embedding dimensions (parameter m). In (a), the phase space produces a straight line, providing no information about the system under study. Observe the phase space in (a) has a high density of entangled trajectories in the same region, close to coordinate $(10, 10)$. Increasing the embedding dimension to $m = 3$, an unfolded phase space is obtained, as shown in (b).

According to the previous example, any phase space reconstructed using an embedding dimension greater than 3 will also be mathematically suitable for further analysis, however, it will be unnecessarily complex. Using the parsimony principle, we intend to obtain the smallest and sufficient embedding dimension. For this, Kennel, Brown & Abarbanel (1992) proposed the method of False Nearest Neighbors (FNN), which initially reconstructs the phase space with an embedding dimension equals to 1 (that is, the simple usage of the time series observations) and then it computes the nearest neighbors for each produced state. Next, the phase space is again reconstructed, but using an additional dimension (embedding dimension equals to 2), and the nearest neighbors are again computed. Then, the nearest neighbors are compared between consecutive reconstructions. If the nearest neighbors change significantly, they are considered as false neighbors, indicating that a greater embedding dimension is necessary. This process is repeated while computing the rate of false nearest neighbors with respect to the increase of the embedding dimension m.

According to Kennel, Brown & Abarbanel (1992), a good value for the embedding dimension is found when the rate of FNN is below some threshold (typically 10%). Considering the observations from the time series X_n, Figure 11 shows the rate of False Nearest Neighbors for different embedding dimensions. One can observe that when $m = 3$, this rate is smaller than 10%, then it is taken as a good estimation for the embedding dimension.

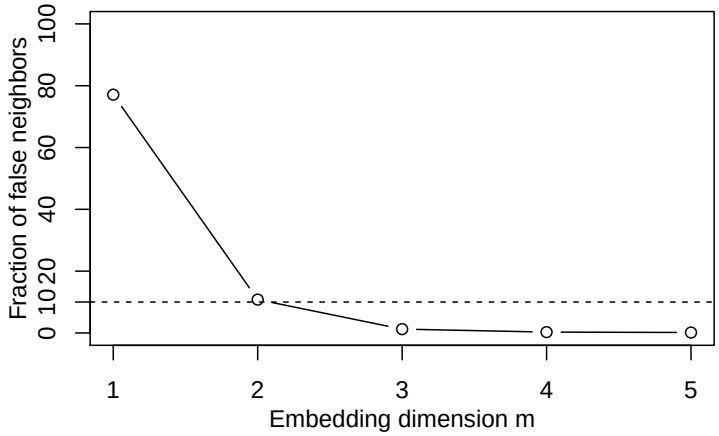


Figure 11 – False Nearest Neighbors applied on time series X_n for the Lorenz (1963)'s attractor. Note that the FNN value is smaller than 10% for $m \geq 3$.

In the previous sections, we presented the essential Dynamical Systems concepts, and we also described the main analysis tool for nonlinear time series which is Takens (1981)' embedding theorem, that permits the reconstruction of phase spaces from discrete and unidimensional time series. In order to obtain meaningful phase spaces, we also presented the methods to estimate both the time delay and the embedding dimension to support the embedding theorem. We suggest the book by Kantz & Schreiber (2004) to complement all the foundation introduced in the previous sections, which has a pragmatic approach on analyzing real-world nonlinear data. Moreover, part of such book is dedicated to advanced techniques on phase space reconstruction.

3.4.3 Detecting dynamical changes in nonlinear time series

3.4.3.1 Common synthetic nonlinear time series

In the previous sections, we discussed three well-known nonlinear dynamical systems. The first was the Logistic Map, an one-dimensional dynamical system typically used for simulating populations of living beings. Its dynamics is defined in Equation 3.6, in which parameter r is commonly set in range $[2.0, 4.0]$. Depending on such parameter, the produced time series is periodic or chaotic.

$$x_{t+1} = r \cdot x_t(1 - x_t) \tag{3.6}$$

The second dynamical system we discussed was discovered by Prof. Edward Lorenz (LORENZ, 1963) during the 1960s, while he was running a simple atmospheric model for weather forecasting. He was trying to duplicate his previous predictions using the same initial conditions and the same code, but the results were completely different after few hundred steps, although they were identical at the beginning (confirming a chaotic behavior). Prof. Lorenz found out that his computer introduced numeric round-offs, which after few steps developed completely different trajectories. Moreover, the Lorenz System exhibited another unexpected behavior, the dynamical system has "regime shifts", by oscillating in between different attractors and abruptly migrating between them (WALLACE; HOBBS, 2006) (one can notice this while observing Figure 8a, in which each regime is one spiral, and the abrupt change happens in the connection of both spirals). Equation 3.7 defines the Lorenz System, for which the most common parameters are $\sigma = 10$, $\rho = 8/3$ and $\beta = 28$.

$$\begin{aligned} \dot{x} &= \sigma(y - x), \\ \dot{y} &= x(\rho - z) - y, \\ \dot{z} &= xy - \beta z \ . \end{aligned} \tag{3.7}$$

The third dynamical system is the Rössler attractor, defined by Rössler (1976) while attempting to provide a simpler representation for the Lorenz System. It develops a single spiral, which makes it easier to be quantitatively analyzed. Equation 3.8 defines the system dynamics, having the most common parameters as $a = 0.1$, $b = 0.1$ and $c = 14.0$.

$$\begin{aligned} \dot{x} &= -z - y, \\ \dot{y} &= x + ay, \\ \dot{z} &= b + z(x - c) \ . \end{aligned} \tag{3.8}$$

Next, we present other two dynamical systems that are typically used in the context of detecting dynamical changes, both based on the later systems. The Transient Logistic Map,

proposed by Trulla *et al.* (1996), is produced by varying the parameter r of a Logistic Map according to a function of time $r(t)$. Typically, the initial value for $r(t)$ is 2.0 and it is iterated up to 4.0, using steps of 10^{-5}. As a consequence, the obtained time series contains $120,000$ observations and it looks as shown in Figure 12. As one can notice, this time series has several different behaviors, namely it begins as 1-periodic up to the value of $r \approx 2.4$, then it evolves to a 2-periodic system up to $r \approx 3.2$, then it changes to a 4-periodic, to an 8-periodic, then to a chaotic behavior, a 6-periodic, chaotic again, a 5-periodic, chaotic, a 3-periodic, and it finally ends as a chaotic system.

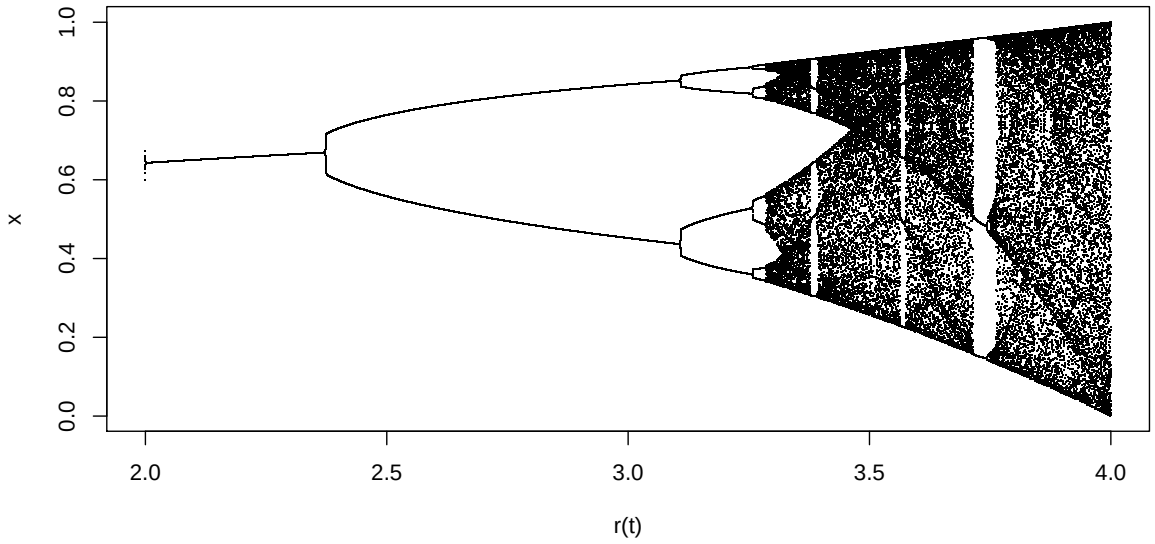


Figure 12 – The Transient Logistic Map, produced by the iteration of parameter $r(t)$ over time in range $[2.0, 4.0]$ (the x-axis). Along its evolution, it assumes eleven different behaviors: first it is 1-periodic, then 2-periodic, 4-periodic, 8-periodic, chaotic, 6-periodic, chaotic, 5-periodic, chaotic, 3-periodic, and it ends as a chaotic system.

The last dynamical system of this section, and also the most complicated one, is the Transient Lorenz System (CAO *et al.*, 2004), which is produced in the same way as the last. This system is iterated while parameter β is increased at every step, thus it is also seen as $\beta(t)$. Such parameter evolves from $\beta(0) = 28.0$ to 268.0 by increasing 0.02 at each step, producing a time series with $120,000$ observations, which is illustrated in Figure 13. Solving Equation 3.7 for the values of $\beta(t)$, we find out there are three periodic windows in this time series, when $99.524 < \beta < 100.795$, $145 < \beta < 166$, and $\beta > 214.4$ (those values are represented as vertical lines in the same figure).

Next sections introduce important techniques used to detect dynamical changes on nonlinear time series.

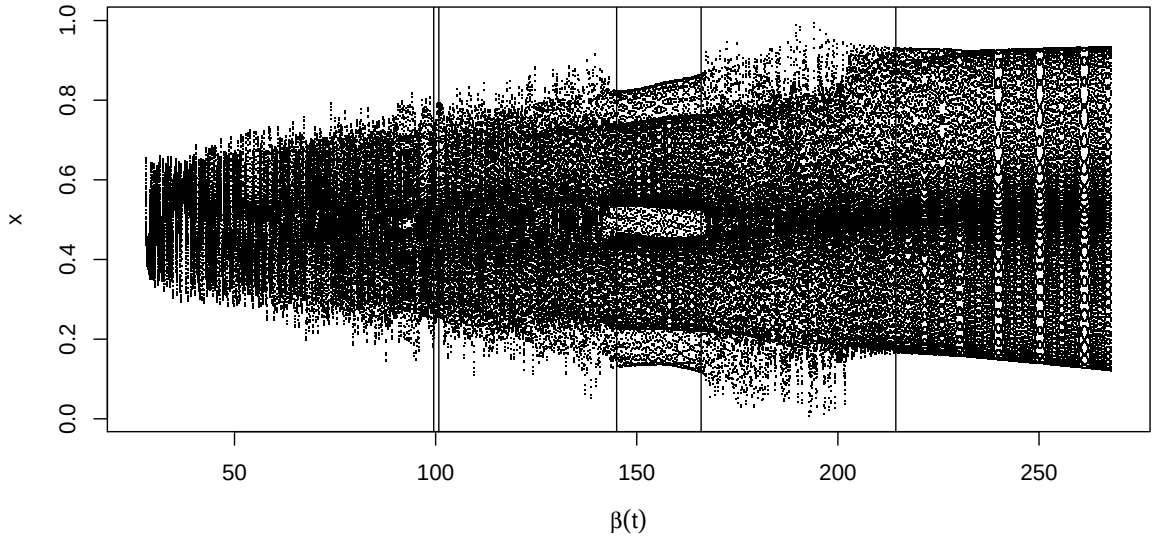


Figure 13 – The Transient Lorenz System, produced by iterating $\beta(t)$ over time, in range $[28.0, 268.0]$, using steps of 0.02 (the x-axis). This dynamical system presents periodic behavior in the following windows: $99.524 < \beta < 100.795$, $145 < \beta < 166$, and $\beta > 214.4$.

3.4.3.2 Lyapunov exponent

Given a dynamical system, the Lyapunov exponent quantifies the sensitivity to initial conditions, i.e., the chaotic behavior. For example, consider two trajectories with adjacent initial conditions on the phase space. When the attractor is chaotic, the trajectories diverge on average at an exponential rate, characterized by the largest Lyapunov exponent (ROSENSTEIN; COLLINS; LUCA, 1993). However, only when the equations describing the dynamical system are available, one can compute the Lyapunov exponent analytically. When dealing with experimental data, the equations of motion are usually unknown, and therefore we need to computationally estimate the Lyapunov Exponent.

The algorithm proposed by Rosenstein, Collins & Luca (1993) is commonly used in such estimation, which begins with the identification of the nearest neighbor of each state in the trajectory $\{\mathbf{x}_j\}$. The nearest neighbor $\mathbf{x}_{\hat{j}}$ is found by searching for the state that minimizes the distance to the particular reference point $\mathbf{x}_j$, as defined as follows:

$$d_j(0) = min_{\mathbf{x}_{\hat{j}}} ||\mathbf{x}_j - \mathbf{x}_{\hat{j}}|| \,, \tag{3.9}$$

in which $d_j(0)$ is the initial distance from the j-th state to its nearest neighbor, and $||\cdot||$ denotes the Euclidean norm.

Next, we iterate the trajectories by increasing the value of i in equation $d_j(i) = ||\mathbf{x}_{j+i} - \mathbf{x}_{\hat{j}+i}||$ and calculating distance $d_j(i)$ at each step. Then, we estimate the largest Lyapunov

exponent at each step i using:

$$\lambda_1(i) = \frac{1}{i\Delta t}\frac{1}{(N-i)}\sum_{j=1}^{M-i}\ln\frac{d_j(i)}{d_j(0)}\ , \tag{3.10}$$

having Δt as the sampling period for the time series, N as the number of states in the phase space, and $d_j(i)$ as the distance between the j-th pair of nearest neighbors after i discrete-time steps. Finally, we can estimate the largest Lyapunov exponent λ_1 by using the method of least-squares to fit an "average" tendency line on $y(i) = \frac{1}{\delta t} < \ln d_j(i) >$, in which $< \cdot >$ denotes the average over all values of j.

The largest Lyapunov exponent can be used to detect dynamical changes on time series by comparing the values of λ_1 over time. Trulla *et al.* (1996) confirmed such analysis while computing the value of λ_1 on a sliding window over a nonlinear, nonstationary and chaotic time series. As they confirmed, when λ_1 is greater than 0, chaotic behavior is observed. Their experiment also corroborate the usefulness of the Lyapunov exponent to identify dynamical changes on time series.

3.4.3.3 Recurrence plot and recurrence quantification analysis

Recurrence Plot (RP) is a method to visualize and thus analyze dynamical systems, introduced by Eckmann, Kamphorst & Ruelle (1987) at the late 1980s. Its initial purpose was to provide visual inspection of the trajectories of the phase space, however nowadays the RP has been applied to several research fields such as Earth sciences, Biology and Engineering (MARWAN *et al.*, 2007).

To better understand the RP, suppose we have a trajectory $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ (that could be obtained after the reconstruction of the phase space from a time series S_t, as shown in Section 3.4.2). The RP is based on a binary recurrence matrix $\mathbf{R}$, whose each cell $\mathbf{R}_{i,j}$ is set to 1 when states $\mathbf{x}_i$ and $\mathbf{x}_j$ are at an ε-neighborhood, otherwise it is set to zero. In other words, let an ε-radius tube around the trajectory $\{\mathbf{x}_i\}$ as shown in Figure 14; for each state $\mathbf{x}_i$, all the states $\mathbf{x}_j$ that are at an ε-neighborhood are considered similar, thus $\mathbf{R}_{i,j} = 1$, otherwise $\mathbf{R}_{i,j} = 0$. Formally, this matrix is expressed as follows:

$$\mathbf{R}_{i,j}(\varepsilon) = \Theta(\varepsilon - ||\mathbf{x}_i - \mathbf{x}_j||), i, j = 1, \ldots, N,$$

in which N is the number of states $\mathbf{x}_i$, ε is a threshold distance, $\Theta(\cdot)$ is the Heaviside function (i.e., $\Theta(x) = 0$, if $x < 0$, and $\Theta(x) = 1$ otherwise) and $||\cdot||$ is a norm (typically the Euclidean norm).

The Recurrence Plot is obtained by plotting the recurrence matrix, using black dots for $\mathbf{R}_{i,j} = 1$, and white dots for $\mathbf{R}_{i,j} = 0$. Figure 15 illustrates the RPs obtained for different systems. Both axes of the RP are associated to time indices, and, by definition, the RP always has a black

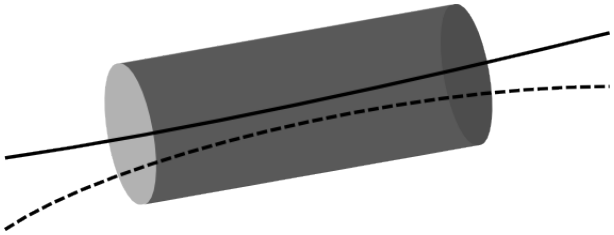


Figure 14 – The Recurrence Plot considers an ε-neighborhood tube around a trajectory (solid line) to estimate which states (from other trajectories – dashed line) are recurrent (adapted from Marwan *et al.* (2007)).

(main) diagonal line due to a state is of course neighbor of itself. Moreover, RPs are symmetrical with respect to the main diagonal.

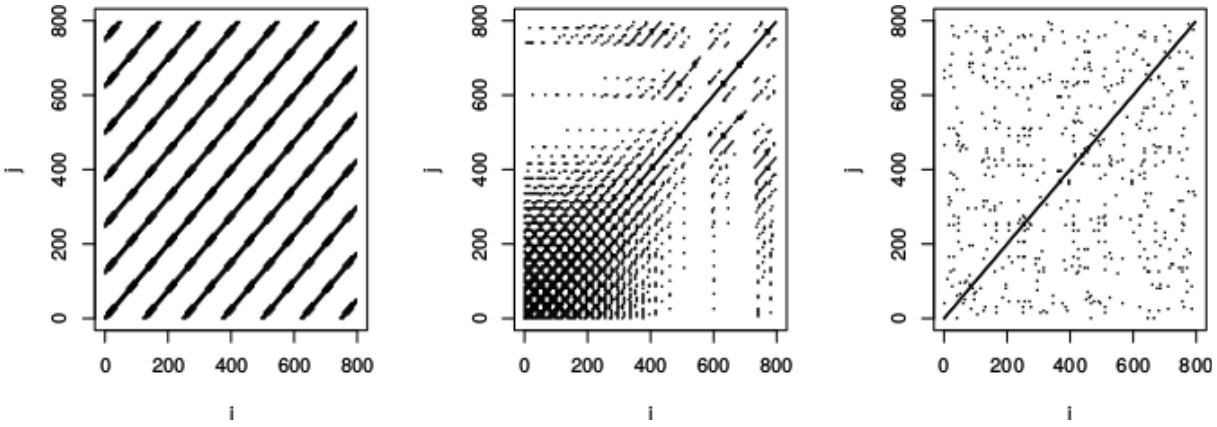


Figure 15 – Examples of Recurrence Plots: on the left side, a periodic motion obtained from a Sine function; at the center, the chaotic Lorenz attractor; on the right side, uniformly distributed points obtained from a Normal distribution.

Analyzing the RPs from Figure 15, one can identify common patterns, as diagonal lines on the periodic motion (left side); diagonal, vertical and horizontal lines on the chaotic motion (center); and uniformly distributed points on the stochastic distribution (right side). This first visual inspection already reveals that the structures found in RPs are associated to the dynamics of the underlying system. In fact, lines are the most important structures of RPs, specially the diagonal ones. This is because lines represent for how long trajectories evolve close to each other, i.e., how long a trajectory stays within an ε-neighborhood tube around another (Figure 14), exhibiting the recurrence among them. Note that RPs are specially convenient for high dimensional systems, as they are good two dimensional projections to analyze the phase space.

In order to compute an RP, an appropriate norm $||\cdot||$ and a suitable threshold ε have to be chosen. While the norm is straightforward to be chosen, usually L_1, L_2 and L_∞, the threshold ε is not trivial to be set, and it is considered the most crucial parameter, mainly because it strongly depends on the dynamical system under study (MARWAN *et al.*, 2007). If the value of ε is set to be too small, few or no neighbors will be found, resulting in no recurrences at all. On the other

hand, when this radius is too large, the RP will identify meaningless recurrences. To properly set ε, several "rules of thumb" were defined, as introduced in Marwan *et al.* (2007), although experiments with different values are suggested. For instance, Serra, Andrzejak *et al.* (2009) suggests to set ε according to a quantile based on the distances among all points in the phase space – they consider ε as a radius enough to have 10% of neighbors around every point.

As the analysis of Recurrence Plots are based on the visual inspection, it induces subjectivity from the examiner. Therefore, several summary measures were defined based on the binary matrix $\mathbf{R}$, named Recurrence Quantification Analysis (RQA) (MARWAN *et al.*, 2007). Among them are the Recurrence Rate (RR – i.e., the percentage of black dots in the matrix), the Determinism (DET – i.e., the percentage of recurrent points forming diagonal lines) and the inverse of the Longest Diagonal Line (LMAX^{-1}). Those summary measures are listed in Equations 3.11 (a-c), in which $\{l\}$ is the set of diagonal lines, $P(l)$ is the probability distribution of diagonal lines, l_{min} is the smallest l and L_{max} is the longest diagonal line l.

$$RR = \frac{1}{N^2} \sum_{i,j=1}^{N} \mathbf{R}_{i,j}, \tag{3.11a}$$

$$DET = \frac{\sum_{l=l_{min}}^{N} lP(l)}{\sum_{i,j=1}^{N} \mathbf{R}_{i,j}}, \tag{3.11b}$$

$$LMAX^{-1} = \frac{1}{L_{max}} \tag{3.11c}$$

The Recurrence Quantification Analysis is used by Trulla *et al.* (1996) for detecting dynamical changes on nonlinear and chaotic time series. Their main motivation on using RQA is that this method permits the use of any kind of data regardless its statistical distribution, nonstationarity and nonlinearity. They adopted sliding windows along the time domain to compute five different summary measures for each of those data chunks. Then, they compared those values from consecutive windows in attempt to detect abrupt dynamical changes. They confirmed the measures RR, DET and LMAX^{-1} are the most reliable for such task.

3.4.3.4 Permutation entropy

In the context of Dynamical Systems, most of the algorithms for detecting dynamical changes employ information on the nearest neighbors in the phase space, which is computationally intensive. On the other hand, Permutation Entropy (PE) uses a different approach, which is conceptually simple and computationally faster (CAO *et al.*, 2004). First, the algorithm maps the states from the phase space into a sequence of symbols. It considers an m-dimensional phase space in which states are defined as $\mathbf{x}_i = (s_i, s_{i+d}, \ldots, s_{i+(m-1)d})$ (produced by the Takens' embedding theorem), having m as the embedding dimension and d as the time delay. The algorithm sorts the m values of every state $\mathbf{x}_i$ in an increasing order, such as $s_{i+(j_1-1)d} \leq s_{i+(j_2-1)d} \leq \ldots \leq s_{i+(j_m-1)d}$.

Afterwards, every state will have its own sorting vector $(j_1, j_2, \ldots, j_m)$, which will be used for mapping it to a symbol between 1 and $m!$.

For example, consider a Rössler Attractor (Section 3.4.3.1) embedded into three dimensions, in which every state will be $\mathbf{x}_i = (s_i, s_{i+\tau}, s_{i+2\tau})$. Getting a sample from the phase space, we will have states such as listed in Table 3.

Table 3 – A sample of the states from the phase space of the Rössler Attractor embedded into three dimensions.

State	$\mathbf{s}_{i,1}$	$\mathbf{s}_{i,2}$	$\mathbf{s}_{i,3}$
$\mathbf{x}_1$	-3.57	-0.06	3.92
$\mathbf{x}_2$	5.33	-1.62	-8.39
$\mathbf{x}_3$	0.60	-7.73	-12.14
$\mathbf{x}_4$	7.64	14.78	8.46

Sorting the m-dimensional values associated to every state $\mathbf{x}_i$ in increasing order, we have the indices for the first state as $\{1,2,3\}$, for the second as $\{3,2,1\}$, for the third as $\{3,2,1\}$ and for the last as $\{1,3,2\}$ (observe indices are the ranking positions of the sorted elements from Table 3). Afterwards, we map those sorted indices into symbols. From this, we create a table with all the possible sorting permutations and give a unique symbol for each, as seen in Table 4.

Table 4 – Symbol mapping for the Permutation Entropy algorithm using an embedding dimension $m = 3$.

Greatest index	...	Lowest index	Symbol
1	2	3	1
1	3	2	2
2	1	3	3
2	3	1	4
3	1	2	5
3	2	1	6

As one can notice, the number of sorting permutations is equal to $m!$. Now we can map the states into symbols, what will provide us: $\mathbf{x}_1 \to 1$, $\mathbf{x}_2 \to 6$, $\mathbf{x}_3 \to 6$ and $\mathbf{x}_4 \to 2$. After such mapping, the algorithm makes a histogram for every possible symbol, producing the probability distributions $P_1, P_2, \ldots, P_{m!}$. With those values in hand, the value of Permutation Entropy is the normalized Shannon entropy as defined in Equation 3.12.

$$H_p(m) = -\frac{1}{\ln(m!)} \sum_{j=1}^{k} P_j \ln P_j. \tag{3.12}$$

In order to detect dynamical changes on time series, Cao *et al.* (2004) performed the PE using sliding windows along the time domain. They found out that PE provides a measure about data regularities. When this measure is small, data is considered regular, and irregular otherwise. To obtain good results with the Permutation Entropy algorithm, they suggest the use of greater values for the embedding dimension m, since it improves the symbolic representation (also

known as overembedding (HEGGER *et al.*, 2000)). They confirmed that for a small embedding dimension, such as $m \leq 4$, poor probability estimators were obtained, and therefore they suggest values such as $m = 5, 6$ or 7 for dynamical systems typically unfolded with 3 dimensions.

3.4.3.5 Multidimensional discrete Fourier transform

The Multidimensional Discrete Fourier Transform (MDFT), proposed by Vallim & Mello (2014), is used for detecting dynamical changes on time series based on the concept of surrogate series. The authors assume that two data windows produced by the same generation process keep similar amplitudes under the same frequencies along time. To contextualize, a surrogate series is obtained by a phase randomization in the frequency domain. Thus, a Fourier Transform (FT) is applied on the original series, then the phase at every frequency is randomized, and finally the inverse FT is applied, producing the surrogate series. By preserving amplitudes associated with frequencies, both the original and the surrogate series have identical statistical measures as the mean and the variance.

The MDFT algorithm works as follows. As first step, the phase space is discretized into a grid of uniformly-length bins and then the number of states per bin is counted. As a consequence, we will have a grid in which every bin represents the density of states contained in such a phase space region. Next, the Multidimensional Discrete Fourier Transform (MDFT) is applied on top of this grid, producing Fourier coefficients. Those coefficients are then used by the Singular Value Decomposition (SVD), which produces eigenvalues λ_i. Those values summarizes the phase space behavior.

Vallim & Mello (2014) use the MDFT algorithm for detecting dynamical changes on time series by using a sliding window along the time domain. At every window, they compute the eigenvalues λ_i^{curr} and compare them against the previous ones λ_i^{prev} using Equation 3.13 (Shannon's Entropy):

$$H(p_1, \ldots, p_m) = -\sum_i p_i \log_2 p_i \ , \tag{3.13}$$

in which probabilities $p_i = \frac{\|\lambda_i^{prev} - \lambda_i^{curr}\|}{max(\lambda_i^{prev}, \lambda_i^{curr})}$ are the differences between the eigenvalues in their order of relevance.

3.5 Final considerations

In this chapter, we introduced time series concepts and how linear and nonlinear time series may be analyzed. Linear time series analysis is mostly based on the field of Statistics and relies on the studies by Box & Jenkins (1976) to proceed with modeling. Nonlinear analysis is based on the field of Dynamical Systems and mainly supported by Takens (1981)' embedding

theorem, which was also detailed. We finished this chapter after describing some of the main literature methods for detecting dynamical changes on nonlinear time series.

CHAPTER

4

STABLE CLUSTERING ALGORITHM

4.1 Initial considerations

In this chapter, we present the theoretical framework proposed by Carlsson & Mémoli (2010), which is used to define the property of stability in terms of data permutation for unsupervised learning algorithms. By allowing any data permutation, this property assumes inputs are produced by some probability distribution from which data is statistically independent. In the next sections, we describe the fundamental concepts considered in such theoretical framework, including metric spaces, ultrametric spaces and equivalence relations. Then,: data permutation invariance we show the general formalization of hierarchical clustering algorithms, which allows the detailing of the two most important parts of this thesis: the Permutation-Invariant Single-Linkage clustering algorithm (PISL), and the Gromov-Hausdorff distance (GH).

4.1.1 Metric spaces

Definition 1. The definition of a *metric space* (X,d) is based on a nonempty set X and a distance function (or metric) $d : X \times X \to \mathbb{R}$. In addition, given $x,y,z \in X$, the following properties must be satisfied:

1. $d(x,y) \geq 0$;
2. $d(x,y) = 0$ if and only if $x = y$;
3. $d(x,y) = d(y,x)$;
4. $d(x,z) \leq d(x,y) + d(y,z)$.

The fourth and last property is referred to as the *triangle inequality*, which considers the sides of a triangle to define the proximity of a pair of elements in X. In summary, each side

of this triangle is always smaller than or equal than the sum of the other two, confirming an element is always as close as, or closer, to another than while traversing through a third point. This property is also taken as a proximity relation among points, i.e., if x is close to y and y is close to z, then x is close to z.

4.1.1.0.1 Example 1

Consider the set of real numbers $\mathbb{R}$ and the distance function $d : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$. Given $a, b \in \mathbb{R}$, the distance between these points is given by $d(a,b) = |a-b|$. Having the four properties of Definition 1 satisfied, tuple $(\mathbb{R}, d)$ defines a metric space.

4.1.1.0.2 Example 2

Consider the set of ordered pairs $\mathbb{R}^2$ and the distance function $d' : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$. Given $(x_1, y_1), (x_2, y_2) \in \mathbb{R}^2$, the Euclidean distance between those pairs will be $d'((x_1, y_1), (x_2, y_2)) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$. Having the four properties of Definition 1 satisfied, tuple $(\mathbb{R}^2, d')$ also defines a metric space.

4.1.2 Ultrametric spaces

Definition 2. An *ultrametric space* is a special type of metric space in which the triangle inequality is replaced by the proximity relation $d(x,z) \leq \max(d(x,y), d(y,z))$, in which $x, y, z \in X$.

This property is called the *strong triangle inequality* as it has a strict upper bound rather than the triangle inequality, and it is also known as the *ultrametric inequality*.

4.1.3 Other notes

In the following definitions, we present the collection of objects used in the formalization by Carlsson & Mémoli (2010).

Definition 3. The collection of all the metric spaces obtained for the set X with n points is given by $\mathscr{X}_n$.

Definition 4. The collection of all the metric spaces obtained for the set X with any number of points is given by $\mathscr{X} = \bigsqcup_{n \geq 1} \mathscr{X}_n$.

Definition 5. The collection of all the ultrametric spaces obtained for the set X with n points is given by $\mathscr{U}_n$.

Definition 6. The collection of all the ultrametric spaces obtained for the set X with any number of points is given by $\mathscr{U} = \bigsqcup_{n \geq 1} \mathscr{U}_n$.

Next, we present the concepts of separability and diameter measures for a metric space.

Definition 7. Let a metric space (X,d), its *separability* is given by $sep(X,d) = \min_{x \neq x'} d(x,x')$, in which $x,x' \in X$.

The separability is the smallest distance between distinct points in the same metric space.

Definition 8. If (X,d) is a metric space, its *diameter* is given by $diam(X,d) = \max_{x,x'} d(x,x')$, in which $x,x' \in X$.

The diameter is the greatest distance between distinct points in the same metric space.

4.1.4 Equivalence relations

Definition 9. Given a set A, a *binary relation* is a subset $S \subset A \times A$. For any pairs $(a,a') \in S$, it is said that a and a' are *related*, which is represented as $a \sim a'$. S is called *equivalence relation* if and only if, for all $a,b,c \in A$ the following properties are hold:

1. Reflexivity: $a \sim a$;
2. Symmetry: if $a \sim b$ then $b \sim a$;
3. Transitivity: if $a \sim b$ and $b \sim c$ then $a \sim c$.

4.1.4.0.1 Example 3

A simple example of equivalence relation is given by the equal signal for the set of real numbers. Consider $a,b,c \in \mathbb{R}$, the equal signal respects the property of reflexivity, i.e., $a = a$; it respects the property of symmetry, i.e., if $a = b$ then $b = a$; and, finally, it respects the property of transitivity, i.e., if $a = b$ and $b = c$ then $a = c$.

Definition 10. The *equivalence class* of a on $\sim$, denoted as $[a]$, is defined by the set of all elements a' in which $[a] = \{a' \in A \text{ such that } a' \sim a\}$.

Definition 11. The *quotient space* $A \setminus \sim$ is the collection of all the equivalence class, i.e., $A \setminus \sim = \{[a], a \in A\}$.

4.1.4.0.2 Example 4

Consider that set $A = \{a,b,c\}$ satisfies the equivalence relations: $a \sim a$, $b \sim b$, $c \sim c$, $b \sim c$ and $c \sim b$. The following sets are equivalents: $[a] = \{a\}$ since $a \sim a$; $[b] = \{b,c\}$, since $b \sim b$ and $b \sim c$; and $[c] = \{b,c\}$, since $c \sim c$ and $c \sim b$. Lastly, the quotient space is obtained by $A \setminus \sim = \{\{a\},\{b,c\}\}$.

4.1.5 *r*-equivalence

Definition 12. Given a metric space (X,d), $x,x' \in X$, the elements are *r-equivalents* (denoted by $x \sim_r x'$), for $r \geq 0$, if and only if there are elements $x_0, x_1, \ldots, x_t \in X$ such that $x_0 = x$, $x_t = x'$ and $d(x_i, x_{i+1}) \leq r$ for $i = 0, \ldots, t-1$.

This way, two elements $x,x' \in X$ are r-equivalents if there is/are zero or more intermediate elements among them so that the maximum distance between consecutive pairs is r.

4.1.5.0.1 Example 5

Consider Figure 16, in which 7 elements are shown in a metric space X using three different distances, $r_1, r_2, r_3 \in \mathbb{R}$, such that $r_1 < r_2 < r_3$. The connected elements represent that the distance between pairs of them is smaller than r_i. Note that elements $x,x' \in X$ are not r_1-equivalents neither r_2-equivalents, but they are r_3-equivalents, i.e., $x \sim_{r_3} x'$, since the distances that connect pairs of points is smaller than r_3.

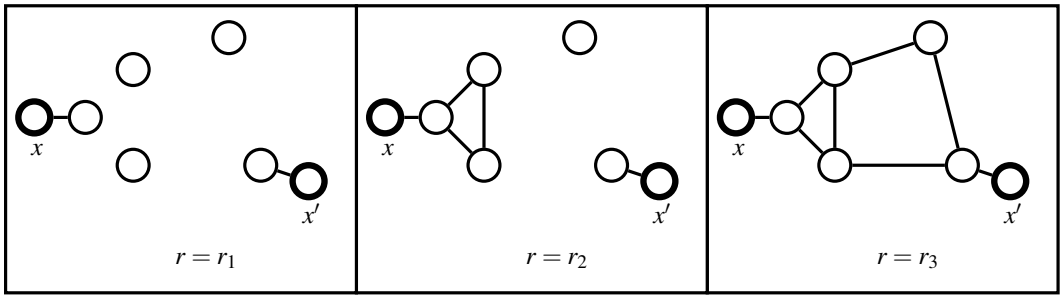


Figure 16 – Illustration of the r-equivalence function. Two points x and x' are r-equivalents if there is/are zero or more intermediary elements between consecutive pairs of points in which their distances are less or equal to r. In this case, x and x' are only r_3-equivalents (adapted from Carlsson & Mémoli (2010)).

4.2 Formalization of hierarchical clustering algorithms

In this section, we set some collection of objects, that will be used to define dendrograms. Then, based on those definitions, the next section defines the concept of an stable clustering algorithm in terms of data permutations, according to the theoretical framework by Carlsson & Mémoli (2010).

4.2.1 Partitions and blocks

Definition 13. If X is a finite set, $\mathscr{C}(X)$ is the collection of all nonempty subsets of X.

Definition 14. If X is a finite set, $\mathscr{P}(X)$ is the collection of all partitions of X.

Definition 15. If $\Pi \in \mathscr{P}(X)$ is a partition of X, each $\mathscr{B} \in \Pi$ is referred to as a *block* of Π.

Definition 16. Given $\Pi, \Pi' \in \mathscr{P}(X)$, Π is said to be *coarser* than Π', or equivalently Π' is a *refinement* of Π, if for each block $\mathscr{B}' \in \Pi'$ there is a block $\mathscr{B} \in \Pi$ such that $\mathscr{B}' \subset \mathscr{B}$.

4.2.2 Dendrograms

Definition 17. A *dendrogram* is defined by the tuple (X, θ) , in which X is a finite set and $\theta : [0, \infty) \to \mathscr{P}(X)$ must hold the following properties:

1. $\theta(0) = \{\{x_0\}, \{x_1\}, \ldots, \{x_n\}\}$ – this property indicates that the initial (bottom) level of the dendrogram produces the greatest refinement as possible, i.e., each element of X is initially in an exclusive cluster;
2. There is some t_0 such that $\theta(t)$ is a unique block for every $t \geq t_0$ – this property indicates that for every value $t > t_0$, dendrograms will be always the same, having only one block containing all elements of X, i.e., $\theta(t) = \{\{x_0, x_1, \ldots, x_n\}\}$;
3. If $r \leq s$, then $\theta(r)$ is a more *refined* dendrogram than $\theta(s)$ – this property ensures a sense of scale for θ and it also suggests the dendrogram partitions are nested;
4. For all r, there is an $\varepsilon > 0$ such that $\theta(r) = \theta(t)$ for $t \in [r, r + \varepsilon]$.

4.2.2.0.1 Example 6

Consider Figure 17, which shows a dendrogram for the set $X = \{x_1, x_2, x_3, x_4\}$. Let θ be the dendrogram, note that $\theta(a) = \{\{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}\}$, $\theta(b) = \{\{x_1, x_2\}, \{x_3\}, \{x_4\}\}$, $\theta(c) = \{\{x_1, x_2\}, \{x_3, x_4\}\}$ and $\theta(t) = \{\{x_1, x_2, x_3, x_4\}\}$ for every $t \geq r_3$.

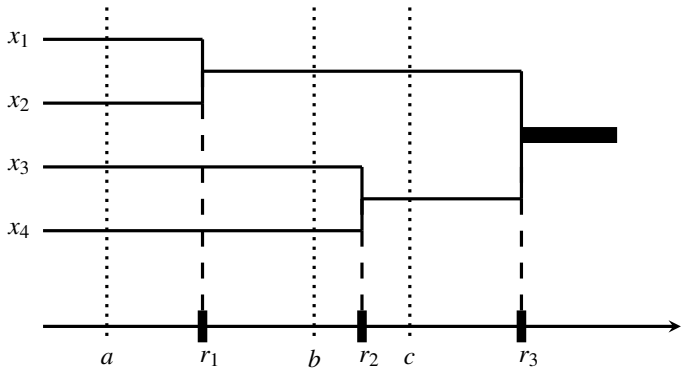


Figure 17 – Illustration of a dendrogram θ in which r_1, r_2 and r_3 are the agglomeration distances. Observe that when $a < r_1$, the dendrogram is $\theta(a) = \{\{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}\}$, at the range $r_1 < b < r_2$, the dendrogram is $\theta(b) = \{\{x_1, x_2\}, \{x_3\}, \{x_4\}\}$, and at the interval $r_2 < c < r_3$, the dendrogram is $\theta(c) = \{\{x_1, x_2\}, \{x_3, x_4\}\}$ (adapted from Carlsson & Mémoli (2010)).

Definition 18. If X is a finite set, $\mathscr{D}(X)$ is a collection of all dendrograms of X.

4.3 Single-linkage clustering algorithm

4.3.0.0.1 The *Single-Linkage* clustering algorithm

Consider the metric space $(X,((d)))$, such that $((d))$ is a distance matrix among elements of X.

1. Define $X_0 = X$, $D_0 = ((d))$ and $\theta(0)$ as the unique/disjoint partitions of X;

2. Search for the smallest distance in D_0, i.e., $\delta_0 = sep(X_0)$, and search for all pairs of elements $\{(x_{i_1},x_{j_1}),(x_{i_2},x_{j_2}),\ldots,(x_{i_k},x_{j_k})\}$ such that elements in X_0 have the pairwise distance δ_0 between each other, i.e., $d(x_{i_\alpha},x_{j_\alpha}) = \delta_0$ for every $\alpha = 1,2,\ldots,n$. Those pairs of elements should be sorted according to their indices, i.e., $i_1 < i_2 < \ldots < i_k$;

3. Select the *first pair* of elements in the list, (x_{i_1},x_{j_1}), and agglomerate it in a new cluster c. Remove the elements x_{i_1} and x_{j_1} from X_0 and add c to the set X_0, i.e., now we have $X_1 = (X_0 \setminus \{x_{i_1},x_{j_1}\}) \cup c$. Define the distance matrix D_1 for $X_1 \times X_1$. Preserve the pairwise distances that were not modified, i.e., $D_1(a,b) = D_0(a,b)$ for all $a,b \neq c$, and change the distances between modified elements (x_{i_1} e x_{j_1}), i.e., $D_1(a,c) = D_1(c,a) = \max(D_0(x_{i_1},a),D_0(x_{j_1},a))$. Lastly, define the dendrogram for δ_0 as follows:

$$\theta(\delta_0) = c \cup \bigcup_{i \neq i_1,j_1} \{x_i\}, \text{such that } c = \{x_{i_1},x_{j_1}\}.$$

4. Repeat the steps until all elements originally in X_0 are agglomerated in a single cluster.

Observe that, in the third step, the algorithm only selects the first pair of elements with the separability of X. Such selection does not consider the situation in which there are multiple pairs at the same smallest distance. As a consequence, when the elements in X are permuted, the produced dendrogram may be different – an undesirable instability for any algorithm. In the next section, we introduce a modification on this algorithm to make it permutation invariant, i.e., for any order of the elements in X, the same dendrogram is produced. This derived algorithm is considered stable according to the stability property defined by Carlsson & Mémoli (2010).

4.4 Permutation-invariant single-linkage clustering algorithm

In this section, we introduce the necessary modifications to make the *Single-Linkage* clustering algorithm which is stable according to permutations in the order of the inputs (CARLSSON; MÉMOLI, 2010). Initially, we define the concepts of linkage functions and then introduce the modified algorithm.

Definition 19. Let (X,d) be a metric space, a *linkage function* $\ell : \mathscr{C}(X) \times \mathscr{C}(X) \to \mathbb{R}^+$ provides a nonnegative measure for each pair of nonempty subsets of X.

Thus, a linkage function defines the distance among subsets rather than among elements, such as the distance metric for a metric space.

Definition 20. Let (X,d) be a metric space, $L(X)$ is a family of linkage functions in X.

Let $\mathscr{B}$ and $\mathscr{B}' \in \mathscr{C}(X)$ be blocks, we define three different traditional linkage functions:

1. Single-Linkage: $\ell^{SL}(\mathscr{B},\mathscr{B}') = \min_{x\in\mathscr{B}} \min_{x'\in\mathscr{B}'} d(x,x')$;
2. Complete-Linkage: $\ell^{CL}(\mathscr{B},\mathscr{B}') = \max_{x\in\mathscr{B}} \max_{x'\in\mathscr{B}'} d(x,x')$;
3. Average-Linkage: $\ell^{AL}(\mathscr{B},\mathscr{B}') = \frac{\sum_{x\in\mathscr{B}}\sum_{x'\in\mathscr{B}'} d(x,x')}{\#\mathscr{B}\cdot\#\mathscr{B}'}$, where # means the cardinality of the object.

Therefore, the linkage function ℓ^{SL} determines the distance between blocks as the distance of their closest elements; function ℓ^{CL} sets the distance between blocks as the distance of their farthest elements; and function ℓ^{AL} defines the distance between blocks as the distance of their average elements.

After these introductory concepts, we present the *Permutation-Invariant Single-Linkage* algorithm, proposed by Carlsson & Mémoli (2010). Such algorithm is invariant to data permutation, therefore it produce models (i.e., clustering partitions) that can be compared in order to quantify data changes or, in the context of this thesis, concept drifts.

4.4.0.0.1 *Permutation-Invariant Single-Linkage* clustering algorithm

1. Fix a linkage function $\ell \in L$. For every distance $R > 0$, consider the equivalence relation $\sim_{\ell,R}$ among blocks $\mathscr{B},\mathscr{B}' \in \Pi$ of partition $\Pi \in \mathscr{P}(X)$. The blocks are R-equivalent with respect to ℓ, i.e., $\mathscr{B} \sim_{\ell,R} \mathscr{B}'$ if and only if there is a sequence of blocks in the same partition, in a way that $\mathscr{B} = \mathscr{B}_1,\ldots,\mathscr{B}_s = \mathscr{B}' \in \Pi$ such that the linkage function between consecutive blocks is smaller than R, i.e., $\ell(\mathscr{B}_k,\mathscr{B}_{k+1}) \leq R$ for $k = 1,\ldots,s-1$;
2. Consider distances $R_1,R_2,\ldots \in [0,\infty)$ and partitions $\Theta_1,\Theta_2,\ldots \in \mathscr{P}(X)$, in which the first partition, i.e., Θ_1 is the initial condition of the algorithm, i.e., $\Theta_1 = \{\{x_1\},\ldots,\{x_n\}\}$. Next, recursively for $i \geq 1$, Θ_{i+1} is the set of partitions of X, such that it is the dendrogram slightly coarser than Θ_i, i.e., $\Theta_{i+1} = \Theta_i \backslash \sim_{\ell,R_i}$, such that:

$$R_i = \min\{\ell(\mathscr{B},\mathscr{B}') + \varepsilon \text{ such that } \mathscr{B},\mathscr{B}' \in \Theta_i, \mathscr{B} \neq \mathscr{B}'\};$$

Observe that two or more blocks may be agglomerated at each iteration of R_i, considering the linkage function provides a value smaller than or equal to R_i, i.e., $\ell(\mathscr{B},\mathscr{B}') \leq R_i$.

Notice this process finishes after a finite number of iterations, due to after a given distance R_k all blocks will be R_k-equivalent regarding to ℓ. Hence, all blocks will belong to partition Θ_k, and this will be the coarsest version of $\mathscr{P}(X)$.

Note that setting a greater value for parameter ε, the algorithm will agglomerate more blocks simultaneously, influencing in the refinement of Θ_i. Thus, the most refined Θ_i is obtained for $\varepsilon = 0$.

3. Define function $\theta^\ell : [0,\infty) \to \mathscr{P}(X)$, having each real value obtained for a partition X regarding to ℓ. This function is a map from the real value r to a partition Θ_i, such that $r \mapsto \theta^\ell(r) = \Theta_{i(r)}$ in which $i(r) = \max\{i | R_i \leq r\}$.

Proposition 21. The following properties come from the proposed algorithm:

1. Θ_{i+1} is coarser than Θ_i, for $i = 1,2,\ldots$;
2. $R_{i+1} \geq R_i$;
3. θ^ℓ is a dendrogram for X.

Note the permutation-invariant algorithm does not specify a linkage function. However, Carlsson & Mémoli (2010) prove that only the *Single-Linkage* function produces stable results. Therefore, we only use this linkage function in the context of this thesis.

4.5 Dendrograms as ultrametric spaces

In order to prove the stability of the permutation-invariant clustering algorithm, Carlsson & Mémoli (2010) use the representation of dendrograms to map ultrametric spaces. Thus, they formulate the algorithm presented in the last section (specifically for the *Single-Linkage* function), so that ultrametric spaces are produced. Next, they prove their proposed algorithm is stable when the Gromov-Hausdorff distance is used, a metric that is capable of computing the divergence among ultrametric spaces.

Theorem 22. If X is a finite set, there is a bijective function $\Psi : \mathscr{D}(X) \to \mathscr{U}(X)$ mapping the collection $\mathscr{D}(X)$ of all produced dendrograms of X to the collection $\mathscr{U}(X)$ of all ultrametric spaces of X, such that for every dendrogram $\theta \in \mathscr{D}(X)$, the ultrametric space $\Psi(\theta)$ of X produces the same hierarchical decomposition that θ, i.e.:

$$\text{for each } r \geq 0, x, x' \in \mathscr{B} \in \theta(r) \iff \Psi(\theta)(x,x') \leq r.$$

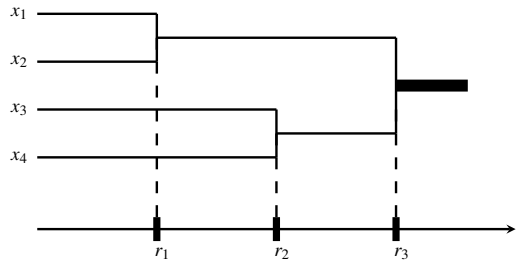
Therefore, for every $r \geq 0$, there is a block $\mathscr{B}$ of the dendrogram $\theta(r)$ that elements x and x' are agglomerated if and only if there is an ultrametric space $\Psi(\theta)$ in which the distance between those elements is smaller than or equal to r. This bijective is also defined by:

$$u_\theta(x,x') = \Psi(\theta)(x,x') = \min\{r \geq 0 | x, x' \text{ belongs to the same block if } \theta(r)\}.$$

The conclusion of this theorem is that, given a finite set X, for each produced dendrogram $\theta \in \mathscr{D}(X)$ there is a corresponding ultrametric space $\Psi(\theta) = u_\theta$.

4.5.0.0.1 Example 7

Consider Figure 18, in which the produced dendrogram θ for set $X = \{x_1, x_2, x_3, x_4\}$ is represented in its form of ultrametric space u_θ. Notice elements x_1 and x_2 are agglomerated in dendrogram θ after r_1, given they have such a distance between each other in the ultrametric space, i.e., $u_\theta(x_1, x_2) = r_1$.



$$((u_\theta)) = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{array} \begin{array}{c} \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \end{array} \\ \begin{pmatrix} 0 & r_1 & r_3 & r_3 \\ r_1 & 0 & r_3 & r_3 \\ r_3 & r_3 & 0 & r_2 \\ r_3 & r_3 & r_2 & 0 \end{pmatrix} \end{array}$$

Figure 18 – Relationship of dendrograms and ultrametric spaces. Observe elements x_1 and x_2 are agglomerated at the point r_1 of the dendrogram, and their distance in the ultrametric space is r_1 (adapted from Carlsson & Mémoli (2010)).

4.6 Formulation of hierarchical algorithms for ultrametric spaces

In this section, we show the formulation of hierarchical algorithms using the result from Theorem 22, in which dendrograms are equivalent to ultrametric spaces.

Definition 23. A method of hierarchical algorithm is defined as a map from a metric space $(X,d) \in \mathscr{X}_n$ to an ultrametric space $(X,u) \in \mathscr{U}_n$:

$$\mathfrak{T} : \mathscr{X} \to \mathscr{U} \text{ such that } \mathscr{X}_n \ni (X,d) \mapsto (X,u) \in \mathscr{U}_n, n \in \mathbb{N}.$$

4.7 Dendrogram comparison

In this section, we show how to compare different dendrograms produced by the permutation-invariant clustering algorithm. First consider the simplest case, in which two different dendrograms (X,α) and (X,β) were produced from the same finite set X, by only changing the metric. According to Theorem 22, these dendrograms are equivalent to ultrametric spaces, such that $u_\alpha = \Psi(\alpha)$ and $u_\beta = \Psi(\beta)$. Thus, we should compute the greatest dissimilarity between these ultrametric spaces, i.e., $\max_{x,x' \in X} |u_\alpha(x,x') - u_\beta(x,x')|$. A natural way to interpret

this dissimilarity is by means of an inequality, $\max_{x,x' \in X} |u_\alpha(x,x') - u_\beta(x,x')| \leq \xi$, i.e., the upper limit is defined by a value ξ. Values $u_\alpha(x,x')$ and $u_\beta(x,x')$ are given by:

$$u_\alpha(x,x') = \min\{r \geq 0 | x,x' \text{ belongs to the same block of } \alpha(r)\}$$

and

$$u_\beta(x,x') = \min\{r \geq 0 | x,x' \text{ belongs to the same block of } \beta(r)\}.$$

4.7.0.0.1 Example 8

Consider dendrograms α (black lines) and β (gray lines) produced from the same set $X = \{x_1, x_2, x_3, x_4\}$, as shown in Figure 19. Let r_i, for $i = 1, \ldots$, be the distances elements were agglomerated in dendrogram α, and r'_i, the distances elements were agglomerated in dendrogram β. The divergence among these dendrograms is given by $\max_i = |r_i - r'_i|$.

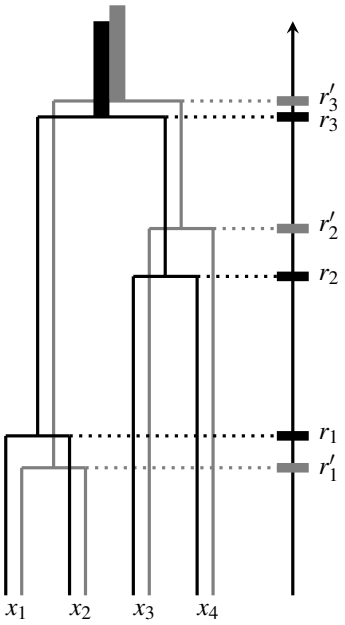


Figure 19 – Divergence between dendrograms according to the Gromov-Hausdorff distance: agglomeration points r_i and r'_i, $i = 1,2,3$ (adapted from Carlsson & Mémoli (2010)).

Now consider the most general case, in which we compare dendrograms (X_1, α) and (X_2, β), produced from two different sets X_1 and X_2, possibly having different cardinalities. In this situation, we consider the mappings $f : X_1 \to X_2$ and $g : X_2 \to X_1$ between sets and compute their *distortions* as follows:

$$dis(f) = \max_{x,x' \in X_1} |u_\alpha(x,x') - u_\beta(f(x), f(x'))|$$

and

$$dis(g) = \max_{x,x' \in X_2} |u_\alpha(g(x), g(x')) - u_\beta(x,x')|.$$

Afterwards, we adapt/select mappings f and g in order to optimize the following objective function $\min_{f,g} \max(dis(f), dis(g))$, which intends to provide the best association for different sets. Then, we compute the Gromov-Hausdorff distance using the *join distortion* between f and g, given by:

$$dis(f,g) = \max_{x \in X, y \in Y} |u_\alpha(x, g(y)) - u_\beta(y, f(x))|.$$

Lastly, the Gromov-Hausdorff distance between (X_1, u_α) and (X_2, u_β) is given by:

$$d_{\mathscr{GH}}(X_1, X_2) = \frac{1}{2} \min_{f,g} \max(dis(f), dis(g), dis(f,g)).$$

All those concepts are employed in the context of this doctoral thesis in order to detect concept drifts under theoretical guarantees.

4.8 Final considerations

In this chapter, we presented the proposal by Carlsson & Mémoli (2010), which is the fundamental basis for this doctoral thesis. We use the permutation invariant hierarchical clustering algorithm, introduced in Section 4.4, on successive windows of the data streams. The obtained dendrograms are then compared by means of the Gromov-Hausdorff distance (Section 4.7), in order to point out divergences among the collected data. However, this stable algorithm cannot be directly applied on data streams, as it requires independent and identically distributed data. Thus, our approach consider a preprocessing step on data windows to transform them into independent states in phase space.

CHAPTER

5

STABLE CLUSTERING TO DETECT CONCEPT DRIFT ON NONLINEAR DATA STREAMS

"Action is movement with intelligence. The world is filled with movement. What the world needs is more conscious movement, more action."

B.K.S. Iyengar

5.1 Initial considerations

In this chapter, we firstly describe our approach for detecting concept drifts in data streams, which considers time dependencies among observations and holds guarantees based on the theoretical foundation provided by Carlsson & Mémoli (2010). Next, we detail two experiments performed to assess our approach in order to point out its advantages and disadvantages while compared to traditional algorithms from the literature.

5.2 Permutation-invariant single-linkage clustering algorithm using the Gromov-Hausdorff distance

In this thesis, we propose an approach for detecting concept drifts in data streams tackling two deficiencies from the unsupervised learning algorithms: their instability for modeling data and their lack of representation of time dependencies among observations. Our approach is based on the proposal by Carlsson & Mémoli (2010), which provides a theoretical framework for *stable*

hierarchical clustering algorithms regarding to data permutation. By taking their proposal into account, we are able to ensure a detected change is indeed associated to data modifications and not to any other reason such as occurs in the literature (see Section 4.7).

In order to accomplish the detection, we decided to use sliding windows along the time domain. As first step, we obtain a data stream window $\mathbf{x_1}$ and embed it using the Takens' embedding theorem (TAKENS, 1981) – the main technique for analyzing nonlinear time series (as detailed in Section 3.4). This step has two objectives. First, by embedding the data we explicitly consider the time dependency among observations, differently from the traditional machine learning algorithms for processing data streams, which only take one past observation into account or an exponential weight over the past observations. Moreover, by embedding the data window we reconstruct the phase space Γ_1, allowing us to analyze nonlinear data. This analysis brings up great advantages over traditional algorithms (that basically consider only linear data), since most of the real-world processes are nonlinear (KANTZ; SCHREIBER, 2004). Second, the embedding theorem transforms the data window from dependent into independent data, which is an required condition for the next step and will be further explained (such characteristic of the embedding theorem motivated other studies in our research group, such as (PAGLIOSA; MELLO, 2017)). As required by the embedding theorem, two parameters must be estimated: the embedding dimension and the time delay, which are both data-driven and therefore should be obtained for each particular data stream under analysis.

In this thesis, we assume data windows have enough data observations to properly reconstruct the phase space of the underlying system. Therefore, we don't go further into discussions on how to define the accurate size of data windows. For more information about this topic, please read (KUGIUMTZIS, 1996).

In the next step, we employ Carlsson & Mémoli (2010)'s theoretical framework on the embedded data, i.e., in the reconstructed phase space Γ_1. First, we cluster Γ_1 using the Permutation-Invariant Single-Linkage (PISL) Clustering Algorithm (detailed in Section 4.4), which is a version of the Single-Linkage algorithm modified to hold the stability property defined by those authors. The algorithm works as follows. Consider a multidimensional data $\{\mathbf{x}_i\}$ independently distributed. The algorithm initially creates a partition for each data observation $\mathbf{x}_i$ at a single cluster. Then, in an iterative manner, the distances among clusters are computed, and all clusters having the same smallest distance among each other are selected and agglomerated, forming new clusters. Indeed, the difference of this modified clustering algorithm to the traditional one is that it agglomerates *all* clusters respecting this minimum pairwise distance, instead of only the *first* two clusters such as the Single-Linkage clustering algorithm. This step is iterated until all data observations are in the same cluster (in a bottom-up approach). At the end, an hierarchical clustering is obtained from the input data, and consequently the dendrogram θ_1.

As a new data window $\mathbf{x_2}$ is obtained from the stream, the steps of embedding and clustering take place, resulting in another reconstructed phase space Γ_2 and dendrogram θ_2,

respectively. As pointed out by Carlsson & Mémoli (2010), by respecting their stability property, the dendrograms θ_i, $i = 1,2,\ldots,n$ produced by the PISL algorithm are equivalent to ultrametric spaces $\Psi(\theta_i)$, $i = 1,2,\ldots,n$. Therefore, we can compute the dissimilarity between consecutive ultrametrics $\Psi(\theta_1)$ and $\Psi(\theta_2)$, by using the Gromov-Hausdorff distance $d_{\mathscr{GH}}(\cdot)$. Such distance computes the greatest dissimilarity between their hierarchical clusterings. First, it selects two mappings f and g that provide the best association for those two ultrametric spaces, as described in Section 4.7. Then, the Gromov-Haussdorf distance is given by the maximum distance between the join distortion of ultrametric spaces.

Although the Gromov-Haussdorf distance provides us theoretical guarantees on the comparison between ultrametric spaces, it is computationally expensive. So, in order to reduce its computational demands, we examined robust and efficient ways to represent multidimensional data that could be used to compute topological invariants, or in other words, ways to summarize the phase space preserving its topology. In the study by Silva & Carlsson (2004), the authors provide a discussion on the topic, and based on such study we designed three different methods to select landmark points in order to reduce the phase space: the random, the kmeans and the maxmin (in which the later was proposed in their study). The first method randomly selects n states from the phase space under study. The second selects n centroids from the phase space based on the K-means clustering algorithm (MACQUEEN *et al.*, 1967). Finally, the maxmin method selects n states that are the most evenly spaced ones, supporting the best coverage of the whole space. To obtain good results, Silva & Carlsson (2004) suggest to choose approximately $n = 5\%$ of the states contained in the phase space.

We use the Gromov-Haussdorf distance to compute the dissimilarity on dendrograms obtained by consecutive data windows as indicative of concept drifts for data streams. This brings up an important advantage for our approach – since most of the concept drift detectors in the field of data streams compare clustering partitions by using some heuristic, empirically supported (as shown in Section 2.4.5), our approach first transforms data windows into ultrametric spaces, that lately are compared by the Gromov-Haussdorf distance (so, it is mathematically supported).

Next, we analyze the obtained distances over consecutive data windows to identify changes. For this, we first smooth the obtained values by applying a Moving Average (MA) with n past values. In this way, we tend to reduce distance fluctuations. Then, we use two Exponentially Weighted Moving Averages (EWMA), one for computing the average and another for estimating the standard deviation of the produced distances between consecutive windows. Our approach triggers a concept drift when the current computed average is greater (or lesser) than the last average plus (or minus) η times the standard deviation. When an alarm is triggered, the estimated average and standard deviation are reseted. All this process of detecting concept drift is performed in a online fashion, that is, after sliding the window on the data stream. The parameters n and η are data-driven and should be estimated for each data stream. The idea behind the average plus/minus η standard deviations come from the Statistical inference (SCHEFLER,

1988). In addition, the use of EWMA imposes some aging effect on the computed distances, so there past divergences bring some historical influences to avoid the useless detection of non-existent drifts.

The algorithm 1 shows all the steps used in this methodology, in which $\mathbf{x}_t$ refers to a data window at time t, Γ_t is the reconstructed phase space for such window, θ_t is the produced dendrogram by the PISL algorithm, and $\Psi(\theta_{t-1})$ is the ultrametric equivalent to such dendrogram. The parameters m, τ and η should be evaluated for every data stream.

Algorithm 1 – Methodology used in this thesis.

while True **do**
  $\mathbf{x}_t \leftarrow$ slideWindow()
  $\Gamma_t \leftarrow$ embedd$(\mathbf{x}_t, m, \tau)$
  $\theta_t \leftarrow$ PISL(Γ_t)
  if exists(θ_{t-1}) **then**
    $\delta_t \leftarrow d_{GH}(\Psi(\theta_{t-1}), \Psi(\theta_t))$
    $\bar{\delta}_t \leftarrow$ updateMeanEWMA(δ_t)
    $s_t \leftarrow$ updateStdevEWMA(δ_t)
    if $(\bar{\delta}_t > \bar{\delta}_{t-1} + \eta \cdot s_t)$ or $(\bar{\delta}_t < \bar{\delta}_{t-1} - \eta \cdot s_t)$ **then**
      triggerAlarm()
    end if
    $\bar{\delta}_{t-1} \leftarrow \bar{\delta}_t$
    $s_{t-1} \leftarrow s_t$
  end if
  $\theta_{t-1} \leftarrow \theta_t$
end while

In summary, the main advantages of our approach over conventional concept drift detection algorithms are the following: i) we explicitly consider the time dependencies among data observations by using Takens' embedding theorem (TAKENS, 1981), making it also possible to analyze nonlinear data; and ii) our approach relies on the theoretical framework provided by (CARLSSON; MÉMOLI, 2010), thus it provide us guarantees on the triggered concept drifts.

5.3 Experiments

In this section, we describe the experiments performed to assess our approach and compare it to traditional algorithms from the literature (see Section 3.4.3). In this manner, we point out and discuss about its advantages and disadvantages to detect concept drift on data streams. We divided the experiments in two sections, in which the first is used to detect *abrupt*, while the second to detect *gradual* changes.

We performed the experiments on synthetic data streams, and added different noise levels to provide interferences found in real-world conditions (KANTZ; SCHREIBER, 2004). Synthetic data was taken into account once they provide the expected points of data change,

simplifying the obtainment of accurate measurements to compare different approaches to detect concept drift in data streams. The measurements considered were: i) the Missed Detection Rate (MDR), i.e., the percentage of true detections that were missed; ii) the Mean Time to Detect (MTD), i.e., the time lag between the dynamical change and its detection; and iii) the Mean Time between False Alarms (MTFA), i.e., the mean time between false detections (BASSEVILLE; NIKIFOROV *et al.*, 1993).

5.3.1 First experiment: detecting abrupt changes

In this first experiment, we synthetically produced the data stream using five different generation processes in a cyclic manner. We used the following dynamical processes: the Lorenz System, the Logistic Map, the Rössler System, the Sine function and the Normal probability distribution (more information about them in Section 3.4.3). We generated the data streams in the following manner: we produced $2,400$ data observations from the first process, i.e., the Lorenz System, then concatenate them to $2,400$ observations from the second process, i.e., the Logistic Map, and so forth, composing the five processes in a cyclic manner (repeating each process eleven times). At the first cycle, we defined a set of parameters for each process, and at every next cycle we slightly changed such parameters, as listed in Table 5. In this manner, we produced different data chunks at each cycle. Moreover, as we concatenated data from one process directly to another, the concept drift is expected to be abrupt.

Table 5 – Dynamical processes used to produce the data stream.

Dynamical process	**Parameters**
Lorenz System	$\sigma = 10, \rho = \{28, 38, 48, 58, 68, 78, 88, 98, 108, 118, 128\}, \beta = 8/3$
Logistic Map	$r = \{3.70, 3.71, 3.72, 3.73, 3.74, 3.75, 3.76, 3.77, 3.78, 3.79, 3.80\}$
Rössler System	$a = 0.15, b = \{0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2\}, c = 10.0$
Sine	frequency equals to $\{1, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0\}$
Normal probability distribution	$\mu = 0, \sigma = \{1, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0\}$

In Figure 20, we illustrate one out of the eleven cycles produced, in which we can notice five different data patterns. We also point out that the observations for each process were normalized in range $[0, 1]$ to avoid any amplitude interference rather than the dynamical behavior of such processes. In total, the data stream was composed of $132,000$ observations and 54 concept drifts.

5.3.1.1 Setup of algorithms

In this experiment we considered the algorithms described in Section 3.4.3: i) the Recurrence Quantification Analysis (RQA) using the measurements of Recurrence Rate (RQA–RR), Determinism (RQA–DET) and the inverse of the Longest Diagonal Line (RQA–LMAX); ii) the Permutation Entropy (PE); iii) the Multidimensional Fourier Transform (MDFT); and iv) our proposal, referred to as Permutation-Invariant Single-Linkage clustering algorithm with the Gromov-Hausdorff distance (PISL–GH).

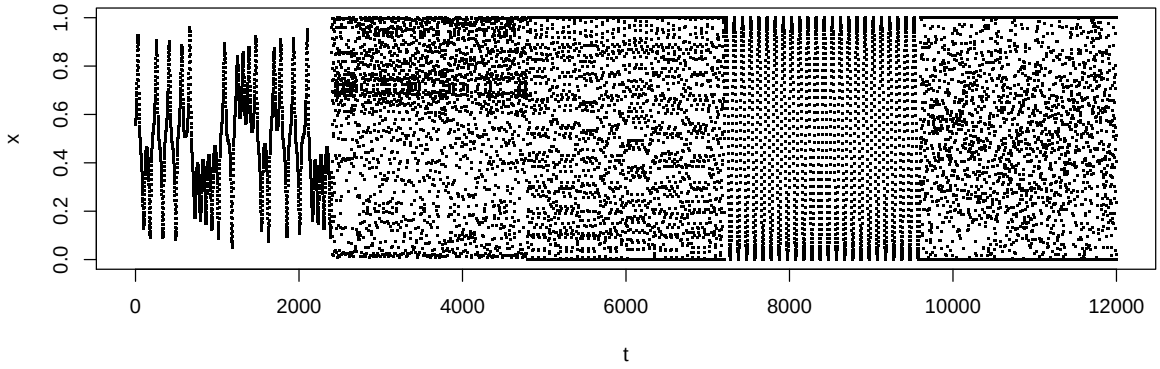


Figure 20 – Sample of the data stream with abrupt changes which was produced using the following dynamical systems: Lorenz System, Logistic Map, Rössler System, Sine function and the Normal probability distribution.

As those algorithms are parameterized according to the target data stream, we had to find the most appropriate parameters for each of them. In order to fulfill this requirement, we performed a Monte Carlo simulation using a sample with the first 3 cycles. In the simulation phase, we considered the permutation of all parameters listed in Table 6. The final results were assessed in the whole data stream using the algorithm parameters found in the simulation phase.

Table 6 – First set of experiments: parameters used in the Monte Carlo simulation.

Algorithm	**Parameter**
PE	$m \in \{5,6,7\}$, $d = \{1,5,10\}$
RQA	$m \in \{2,3\}$, $d = \{1,5,10\}$, radius= $\{0.1, 0.2, \ldots, 1.0\}$
MDFT	$m \in \{2,3\}$, $d = \{1,5,10\}$, nbins= $\{7,9,11,13,15\}$
PISL–GH	$m \in \{2,3\}$, $d = \{1,5,10\}$, landmark method= {maxmin, kmeans, random}

The ranges of parameters were defined as detailed next. As the dynamical processes used to compose the data stream are ideally unfolded using embedding dimension equals to 2 or 3, and time delay equals to 1 (in the case of the maps) or between 5 and 10 (for the Lorenz System and the Rössler Attractor), we ensured good candidates for all algorithms. In the case of the Permutation Entropy, we allowed the embedding dimension to be greater, between 5 and 7, as suggested by Cao *et al.* (2004). For the Recurrence Quantification Analysis, we allowed the radius to be adapted in range $[0.1, 1.0]$ (recall data observations were normalized in interval $[0,1]$, thus a radius of 1.0 would enclose the whole space). In case of the Multidimensional Fourier Transform, we permitted a large range for the number of bins (used to discretize the phase space into n uniform bins per axis). Finally, for the Permutation-Invariant Single-Linkage Clustering Algorithm with the Gromov-Hausdorff distance, we performed three types of selecting landmark points (explained in Section 3.4.3, i.e., random, kmeans and maxmin) to reduce the number of data observations and make the Gromov-Hausdorff distance run faster. We also performed

the experiment using the entire phase space, in order to obtain more precise results besides the additional computational costs.

As all those algorithms are sliding-window oriented, we had to define both the window length and the window overlapping before proceeding with the experiments. Based on the literature (TRULLA *et al.*, 1996; CAO *et al.*, 2004), we defined the window length equals to 800, and an overlapping of 790 observations with the previous window, so every data stream would be processed at every 10 new data observations. Using such setting, all results were comparable one another.

Then, the detection of concept drift worked as follows. For every window $\mathbf{w_t}$, its data was processed by an algorithm A, producing a feature f_t as the main characteristic for the data chunk. Then, after n windows, we obtained features $f_t, t = 1, 2, \ldots, n$ that were used to detect changes. First, features were smoothed through a moving average model, using 1, 5, 10 or 20 previous observations, according to the optimization provided by the Monte Carlo simulation (ROBERT, 2004). Next, we used two Exponential-Weighted Moving Averages (EWMA), one to compute the average value for features (Equation 5.1) and another to estimate their standard deviations (Equation 5.2). It is worth to mention that those EWMAs provided more relevance to the current feature value than for older ones, implying data aging. One should also notice the EWMA computes the statistics in an online fashion, i.e., at every feature produced along time. When the current average value $avg_f(t)$ is greater (or lesser) than the previous $avg_f(t-1)$ plus (or minus) η standard deviations $stdev_f(t-1)$, in form $avg_f(t-1) + \eta \cdot stdev_f(t-1)$, an alarm is triggered to inform about the concept drift.

This approach is based on Statistics inference, more precisely in the area integrated to ensure confidence levels for the Normal probability distribution (SCHEFLER, 1988). In our experiments, we considered $\alpha = \beta = 0.1$ for Equations 5.1 and 5.2, and the value of η was set as either $2, 3, 4, 5$ or 6, what was also optimized using the Monte Carlo simulation.

$$avg_f(t) = (1-\alpha) \cdot avg_f(t-1) + \alpha \cdot f_t \ . \tag{5.1}$$

$$stdev_f(t) = (1-\beta) \cdot stdev(t-1) + \beta \cdot \sqrt{(f_t - avg_f(t))^2} \ . \tag{5.2}$$

As the data stream was synthetically produced, we had supervised information to ensure when the stream behavior changed. In this sense, we could guarantee the exact moments the alarms should be triggered. Based on this supervision, we computed three measurements to asses all the algorithms: the Missed Detection Rate (MDR), the Mean Time for Detection (MTD) and the Mean Time between False Alarms (MTFA) (BASSEVILLE; NIKIFOROV *et al.*, 1993). In order to proceed with such computation, we first labeled every triggered event as either a true or a false alarm. We defined a true alarm as an alarm triggered inside of a radius of 400 data observations of the exact point the concept drift happened (half a window), in this way we do

not penalize an algorithm if it triggers an alarm a bit earlier than expected, and we also limit the time lag an alarm is set. If an alarm is triggered out of such radius, it is considered a false alarm.

Figure 21 illustrates how the feature vector $f_t, t = 1,2,\ldots,n$ is processed in order to obtain the true and false alarms. The upper chart shows two cycles from the data stream used in this experiment, in which one can visualize 10 different data patterns that change along time. The second chart only illustrates the feature vector produced by PISL–GH after processing this data stream, in this case f_t is the Gromov-Hausdorff distance between consecutive dendrograms. Then, we smoothed the feature values, by using a moving average model, obtaining the third chart. Using the two EWMAs as explained before, one for computing the online average and another the online standard deviation for this smoothed feature vector, we identified the alarms triggered by each algorithm (shown as black vertical bars on the fourth chart). Finally, in the last chart, the alarms that are 400 observations close to the expected concept drift are considered true (green vertical bars), while the others are taken as false alarms (red vertical bars). As one can notice, the analyzed algorithm could identify all concept drifts, while few false alarms were triggered.

Based on the true and false alarms, we computed the MDR, MTD and MTFA measurements. MDR corresponds to the rate of missed true alarms, therefore the lesser it is, the better the result is. MTD is the mean time for detecting an expected alarm, which is also better when it assumes smaller values. Finally, MTFA is the mean time between false alarms, which behaves inversely, so the greater its value is, the better the result is. It is worth mentioning those measurements depend on the application requirements, for example, some application domains may give more relevance to avoid false alarms while others to reduce the mean time for the detection (this last may increase the occurrence of false alarms).

We used Monte Carlo to optimize Equation 5.3 for every algorithm A, in which $N(\cdot)$ is a function to normalize data in range $[0,1]$ using the minimal and maximal values, ℓ is the data stream length used to invert the MTFA values so the smaller it is, the better it is, thus behaving in the same way as the other measurements. Finally, $E(A)$ provides the Euclidean distance considering all measurements, thus the smaller this distance is, the better the average results for MTFA, MDR and MTD are.

$$E(A) = \sqrt{N(\frac{\ell}{MTFA})^2 + N(MDR)^2 + N(MTD)^2} \tag{5.3}$$

5.3.1.2 Results

Table 7 shows the parameters found using the Monte Carlo simulation for all algorithms. Notice the proposed algorithm PISL–GH is shown in two rows of the table, one considering the best setup with (w/land) and another without landmark points (wo/land). In this manner, we compare PISL–GH results when some technique to reduce the phase space is used.

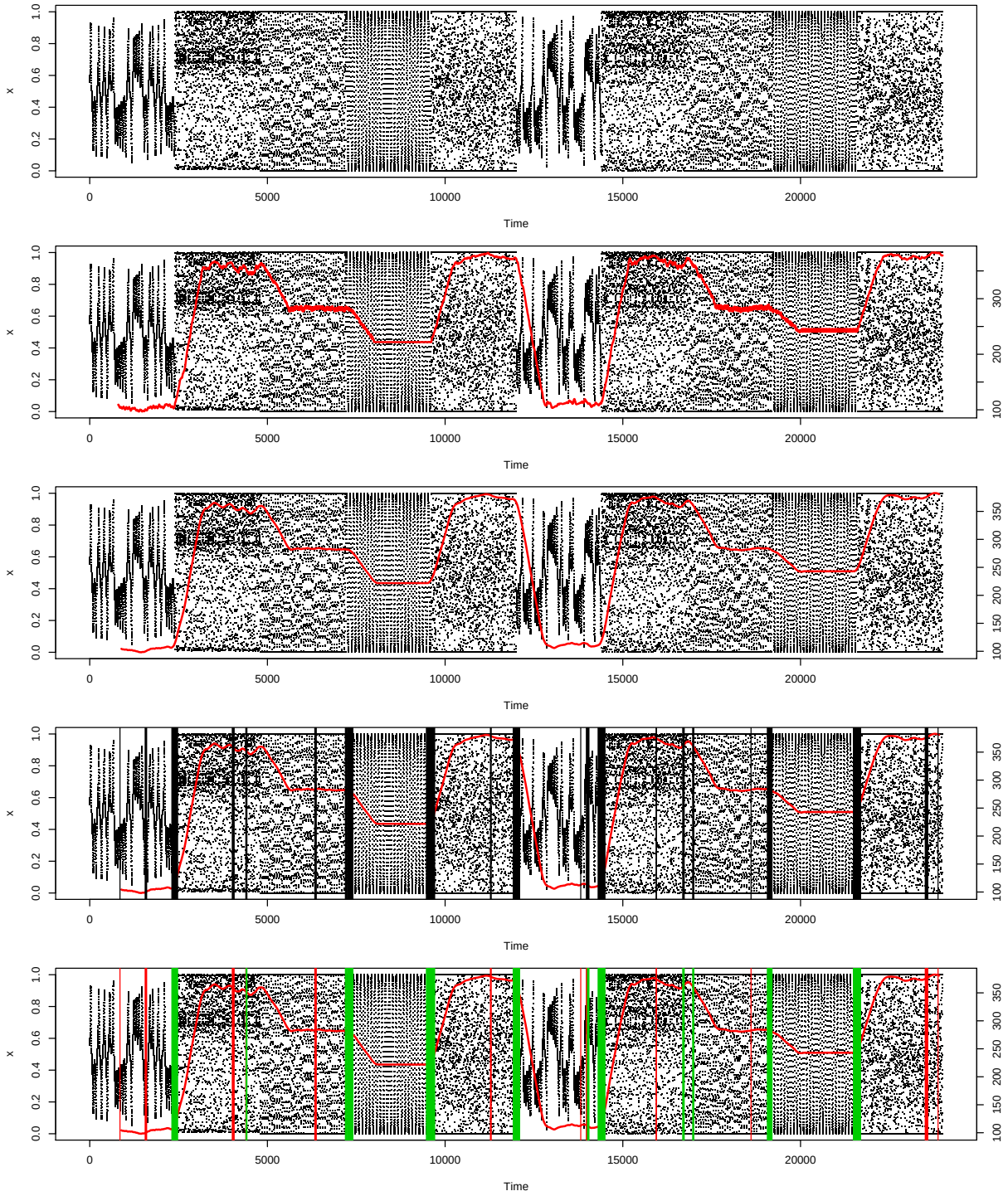


Figure 21 – The process of issuing alarms by an algorithm: in the upper chart, the data stream itself. In the second, the red line is the feature vector produced by an algorithm after processing the data stream. The third chart is the same feature vector after being smoothed by a moving average model. Next, the algorithm issues alarms using the criterion based on two Exponential-Weighted Moving Averages (EWMA). Finally, the last chart shows the true and false alarms.

Table 8 lists the results obtained for each algorithm while dealing with this first experiment, considering the parameters estimated using the Monte Carlo simulation. The measurements Missed Detection Rate (MDR), Mean Time for Detection (MTD), Mean Time between False

Table 7 – Algorithms parameters estimated using the Monte Carlo simulation.

Algorithm	Parameters
PE	$m = 6, d = 1, ma.n = 1, \eta = 4$
MDFT	$m = 2, d = 1, \text{nbins} = 15, ma.n = 5, \eta = 3$
RQA–RR	$m = 2, d = 1, \text{radius} = 1.0, ma.n = 5, \eta = 2$
RQA–DET	$m = 3, d = 10, \text{radius} = 0.4, ma.n = 5, \eta = 2$
RQA–LMAX	$m = 2, d = 1, \text{radius} = 0.9, ma.n = 5, \eta = 5$
PISL–GH (w/land)	$m = 2, d = 10, \text{landmark method=maxmin}, ma.n = 20, \eta = 3$
PISL–GH (wo/land)	$m = 2, d = 10, ma.n = 20, \eta = 2$

Alarms (MTFA) and the Euclidean distance among them are listed. PE obtained the best overall result, with an Euclidean average of 0.785, and moreover, it obtained also the best values for MTD and MTFA. The measurements based on RQA obtained very similar overall results among them, in which the RQA–LMAX obtained the best one, although it did not detected all alarms, it obtained a smaller value of MTD and a greater of MTFA. Following, MDFT obtained as overall result 1.268, detecting almost every alarm but it had an undesirable great MTFA.

With the worst overall result in this experiment, our PISL–GL with landmark points obtained only 2.976 as Euclidean average. It missed 16.7% of the true alarms, similarly to the RQA–LMAX; it also had the longest time lag for detecting true alarms (MTD); and its mean time between false alarms (MTFA) was one of the smallest. On the other hand, without landmark points our approach PISL–GH obtained a much better overall result, surpassing MDFT, RQA–RR and RQA–LMAX. By using landmarks, PISL–GH obviously deals with less data in the phase space, making it run much faster. However, that strongly affects its overall performance. We believe this should be addressed in future work, bringing up the space reduction with low impacts in the detection process.

Table 8 – Results obtained for the first experiment.

Algorithm	MDR	MTD	MTFA	Euclidean
PE	0.148	**25.031**	**14525.556**	**0.785**
MDFT	0.056	114.149	452.276	1.268
RQA–RR	**0.000**	100.602	290.843	1.128
RQA–DET	**0.000**	81.254	336.333	1.065
RQA–LMAX	0.167	64.518	1599.634	1.860
PISL–GH (w/land)	0.167	236.031	459.684	2.976
PISL–GH (wo/land)	0.013	61.525	2571.961	1.108

In this first experiment, we assessed four algorithms (Permutation Entropy, Multidimensional Fourier Transform, Recurrence Quantification Analysis, and our approach, i.e., the Permutation-Invariant Single-Linkage Clustering Algorithm with the Gromov-Hausdorff distance) on a data stream generated by the concatenation of several dynamical systems. By using such concatenation, we consider the concept drifts as abrupt and, therefore, the detection should be easier than the next experiment. In this scenario, the Permutation Entropy obtained the best

overall result among algorithms, followed by RQA–DET and by PISL–GH without landmark points. It is worth to mention that with landmarks our approach was capable of detecting most of the concept drifts, however with a significant delay, and it also triggered several false alarms, providing the worst overall result. Considering no landmark points, our approach obtained the third best result.

5.3.2 Second experiment: detecting gradual changes

In this second stage, we performed experiments on six 120,000-observation data streams produced by a Transient Logistic Map and a Transient Lorenz System, what makes them more interesting than the first experiment due to they keep the same dynamical processes but slowly change their parameterization along time. Table 9 summarizes all data streams considered during experiments.

Table 9 – Data streams used in the second experiment.

Name	Dynamical process	Noise
log-none	Transient Logistic Map	None
log-low	Transient Logistic Map	1% – $\mathcal{N}(0, 0.002)$
log-med	Transient Logistic Map	5% – $\mathcal{N}(0, 0.01)$
lor-none	Transient Lorenz System	None
lor-low	Transient Lorenz System	1% – $\mathcal{N}(0, 0.00175)$
lor-med	Transient Lorenz System	5% – $\mathcal{N}(0, 0.00825)$

Those synthetic data streams were already considered in other studies (TRULLA *et al.*, 1996; CAO *et al.*, 2004), described in Section 3.4.3. Three of them are based on the Transient Logistic Map (using different levels of noise as shown in Table 9), which follows the simple nonlinear dynamical equation $x(t+1) = r(t) \cdot x(t)(1-x(t))$, such that $x(t)$ is a number (typically the population size) in range $[0, 1]$, and $r(t)$ is a parameter that varies over time. Depending on parameter $r(t)$, this simple equation can produce from periodic to chaotic data observations.

As already considered by other studies (TRULLA *et al.*, 1996; CAO *et al.*, 2004), we varied parameter $r(t)$ from 2.8 to 4.0, increasing 10^{-5} per iteration. According to this approach, we created a data stream as illustrated in Figure 22, in which the x-axis refers to the value of $r(t)$ and the y-axis at the left to the observation values. The red line is the largest Lyapunov exponent λ_1, analytically computed for this data stream (as we had the equations to model the dynamics). The y-axis at the right corresponds to the value of λ_1, in which the negative scale is associated to convergent behavior (most seen as periodic data) and the positive to chaotic data. There are 10 behavior changes in this data stream – namely, the segments are 1-periodic, 2-periodic, 4-periodic, 8-periodic, chaotic, 6-periodic, chaotic, 5-periodic, chaotic, 4-periodic and chaotic.

The other three data streams were produced using the Transient Lorenz System (also with different levels of noise as listed in Table 9), which is described in Section 3.4.3. In this case,

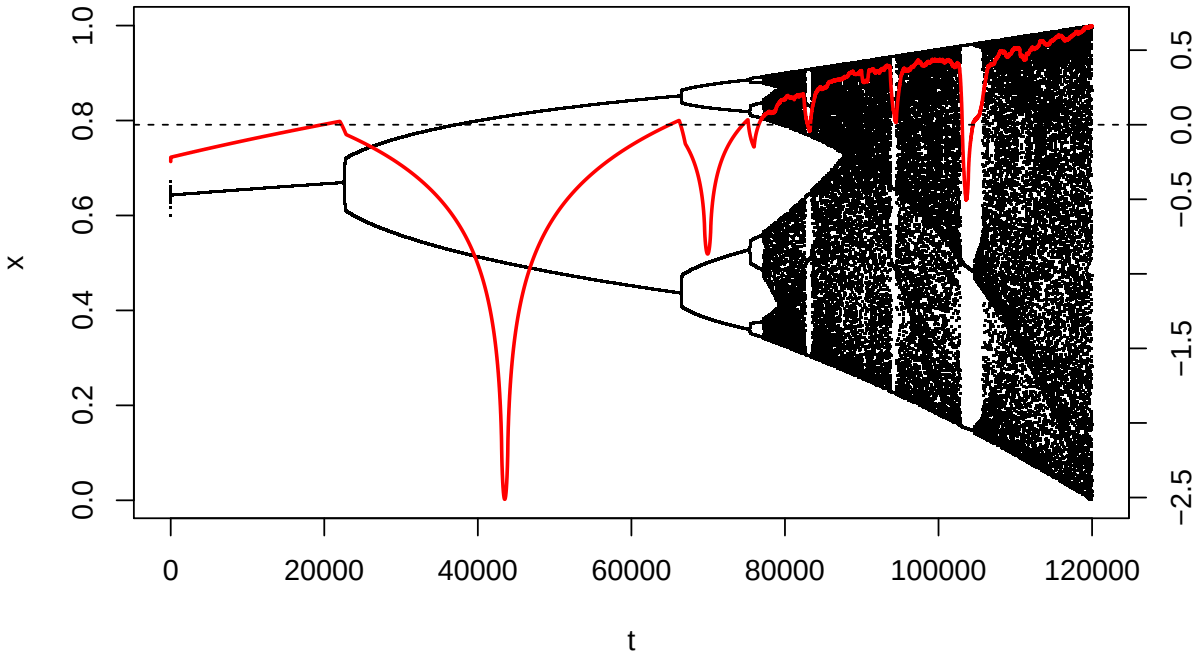


Figure 22 – Transient Logistic Map produced by adapting parameter r along iterations. The x-axis corresponds to parameter $r(t)$ and at the left of the y-axis to the observation values. The red line is the largest Lyapunov exponent λ_1 analytically computed for this data stream. The right y-axis corresponds to the values of λ_1, in which the negative scale is associated to convergent behavior while the positive to chaotic data.

we varied parameter $\rho(t)$ from 28 to 248 increasing 0.002 per iteration, also producing a data stream with $120,000$ observations (Figure 23). The data streams based on the Transient Lorenz System are more complicated than the ones based on the Logistic Map since they are created by the iteration of three ordinary differential equations instead of just one. Thus, we expect to assess the behavior of the algorithms in data streams with different number of dimensions for the phase space.

All data streams had their values normalized between zero and one, and we added noise into four of the six data streams (Table 9) in order to analyze how it could affect the concept drift detection. In this manner, we produced not only pure deterministic data streams, but also streams with stochastic influences generated using the Normal probability distribution with mean $\mu = 0$ and standard deviation σ ($\mathcal{N}(\mu, \sigma)$). In that sense, we added 1% and 5% of relative noise to the data scale, consequently the signal-to-noise ratio was 99% and 95%, respectively.

5.3.2.1 Setup of algorithms

We used the same procedure for finding the best parameters for all algorithms in this second stage of experiments, which was based on the Monte Carlo simulation. In this circumstance, we used a sample of $20,000$ data observations from the data stream without noise and performed the experiments using the following algorithms: Permutation Entropy (PE), Recurrence Quantification Analysis (RQA), Multidimensional Fourier Transform (MDFT) and Permutation-Invariant

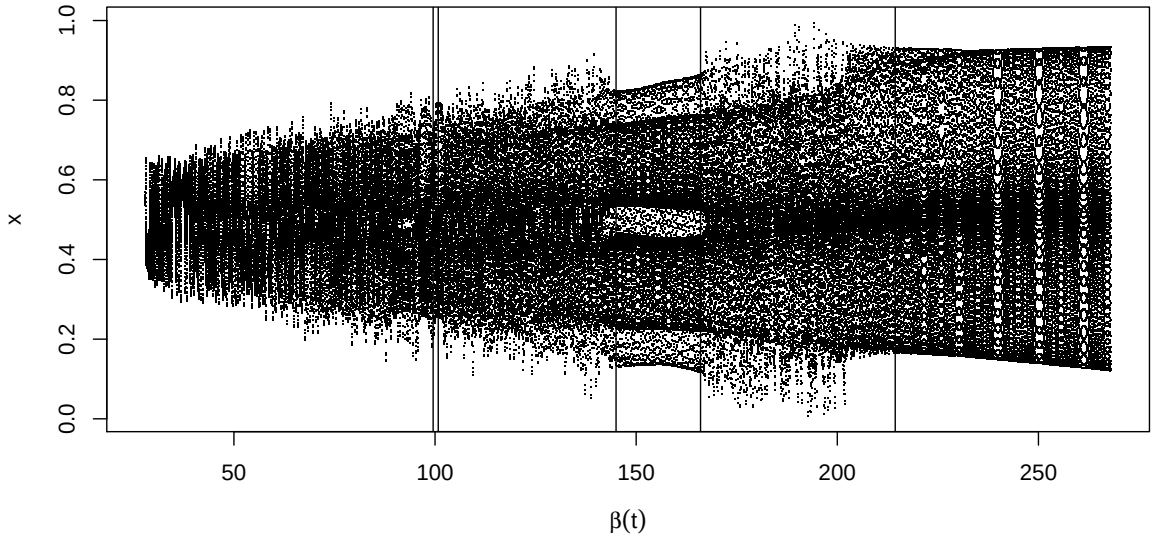


Figure 23 – Transient Lorenz System produced by varying parameter ρ from 28 to 248. The x-axis corresponds to parameter $\rho(t)$ and the y-axis to the observation values.

Single-Linkage Clustering Algorithm with the Gromov-Hausdorff distance (PISL–GH).

We considered the assessment of all parameters listed in Table 10, and we chose their possible values as follows. As the Logistic Map is ideally embedded into 2 dimensions using time delay equals to 1, we allowed the algorithms to change the embedding dimension between 2 and 3, and the time delay was fixed as 1. For the PE, we followed the suggestion by Cao *et al.* (2004) and allowed greater embedding dimensions, varying from 5 to 7. In the RQA, we performed the measurements of Recurrence Rate (RQA–RR), Determinism (RQA–DET) and the inverse of the Longest Diagonal Line (RQA–LMAX) varying the values of radius in range $[0.1, 1.0]$ (recall data streams were normalized in range $[0, 1]$, thus a radius equal to 1.0 would fit all data observations as neighbors). For the MDFT, we allowed the number of bins to vary from 7 to 21. Finally, for our approach, PISL–GH, we permitted the selection of the landmark method to reduce the phase space and consequently the computational cost, from the set random, kmeans and maxmin. For the Transient Lorenz System, we just increased the possible embedding dimensions to 3 or 4 (as it is ideally embedded into 3 dimensions), and fixed the time delay equals to 8 as this dynamical system provides good results using such parameterization.

In the simulation phase, we considered a sample of $20,000$ observations from the data stream, using windows with the same length of 800 observations with an overlapping of 790, thus adding 10 new observations at each next iteration. While processing a window, each algorithm A produced a feature f_t that was initially smoothed by a moving average model, then analyzed by using two Exponential Weighted Moving Averages (EWMA) in order to detect concept drifts. Every time the EWMA used to determine the average was greater (or smaller) than the last mean

Table 10 – Second set of experiments: parameters used in the Monte Carlo simulation.

Data stream	Algorithm	Parameter
Transient Logistic Map	PE	$m \in \{5,6,7\}$, $d = 1$
	RQA	$m \in \{2,3\}$, $d = 1$, radius= $\{0.1, 0.2, \ldots, 1.0\}$
	MDFT	$m \in \{2,3\}$, $d = 1$, nbins= $\{7,9,11,13,15,17,19,21\}$
	PISL–GH	$m \in \{2,3\}$, $d = 1$, landmark method=$\{$maxmin, kmeans, random$\}$
Transient Lorenz System	PE	$m \in \{5,6,7\}$, $d = 8$
	RQA	$m \in \{3,4\}$, $d = 8$, radius= $\{0.1, 0.2, \ldots, 1.0\}$
	MDFT	$m \in \{3,4\}$, $d = 8$, nbins= $\{7,9,11,13,15,17,19,21\}$
	PISL–GH	$m \in \{3,4\}$, $d = 8$, landmark method=$\{$maxmin, kmeans, random$\}$

plus (or minus) η standard deviations, an alarm was triggered. At the end of this simulation, we calculated the measurements of Missed Detection Rate (MDR), Mean Time to Detect (MTD) and Mean Time between False Alarms (MTFA) for every parameter permutation of each algorithm listed in Table 10, and we chose the parameters to obtain the smallest Euclidean distance as possible (Equation 5.3). In this manner, we found the best combination of parameters for each algorithm, and then we proceeded with the experiments on the entire data stream.

5.3.2.2 Results

Table 11 shows the combination of parameters for each algorithm that produced the best results in the Monte Carlo simulation for the Transient Logistic Map. Notice we performed the experiments with the algorithm PISL–GH without landmarks only in few scenarios, such as without the addition of noise. This is due to the time complexity involved in the experiments – while the ones using landmarks took approximately 30 minutes to execute (as the other algorithms from the literature), the experiments without landmarks took one week, and sometimes, even more on a single core of Intel(R) Xeon(R) with 2.00 GHz, with 256GB of RAM.

Table 11 – Parameters estimated using the Monte Carlo simulation on the data stream produced with the Transient Logistic Map.

Algorithm	Parameters	EWMA parameters
PE	$m = 5, d = 1$	$ma.n = 1, \eta = 3$
MDFT	$m = 2, d = 1,$ nbins $= 13$	$ma.n = 10, \eta = 5$
RQA–RR	$m = 3, d = 1,$ radius $= 0.7$	$ma.n = 1, \eta = 2$
RQA–DET	$m = 2, d = 1,$ radius $= 0.5$	$ma.n = 10, \eta = 3$
RQA–LMAX	$m = 3, d = 1,$ radius $= 0.1$	$ma.n = 20, \eta = 6$
PISL–GH	$m = 2, d = 1,$ landmark method $=$ maxmin	$ma.n = 20, \eta = 3$

In Table 12, we show the results of measurements MDR, MTD and MTFA for each algorithm on the whole Transient Logistic Map. Observe we have assessed those algorithms on three data streams based on this same dynamical process, all having the same window length and parameters, however under different noise levels. First we analyzed the pure deterministic stream, then we added 1% of noise following a Normal probability distribution with mean 0

and $\sigma = 0.002$, and at last with 5% of noise, using Normal distribution with a greater standard deviation given by $\sigma = 0.01$.

First we discuss the data stream without noise. In this scenario, the lowest Missed Detection Rate (MDR) of 20% was produced by our proposal PISL–GH with landmarks, followed by the PE and RQA–RR with 30%, and our PISL–GH without landmarks with 40%. It is worth to mention that PISL–GH without landmarks was probably worse because the Monte Carlo simulation was not performed for that setting due to the time consumed in such operation. The other algorithms obtained greater MDR, showing that they could not trigger the correct number of true alarms. The lowest value for the Mean Time to Detect (MTD) was obtained by the MDFT with 36.667 observations, followed by the RQA–LMAX with 104.000 and RQA–DET with 121.250. Our approach obtained 154.296 using landmarks and 125.595 without them. The greatest Mean Time between False Alarms (MTFA) was obtained again by the MDFT with 5671.905, followed by the RQA–LMAX with 2017.288 and the PE with 1602.432. Our approach obtained 508.590 using landmarks and 1228.144 without them. In summary, using the Euclidean distance of those (normalized) measurements, the best score was obtained by the MDFT with 0.734, followed by the PE with 0.866 and by our approach without landmarks, with 0.986. Using landmarks, we obtained the score of 1.160, being still far from the worst result of 1.786 for the RQA–RR.

In the scenario with the 1%-noise data stream, the best MDR of 10% was obtained by our approach PISL–GH with landmarks, followed by the PE with 20% and RQA–RR with 30%. The others obtained a value greater than 60%. The lowest value of MTD was obtained by the RQA–LMAX with 109.000 observations, and our approach obtained 148.222. The greatest MTFA was obtained again by the MDFT with 2290.577, and our approach produced the worst results with 420.530. In summary, the best algorithm on this task was the RQA–LMAX, followed by our approach.

In the third data stream, the 5%-noise Transient Logistic Map, the best MDR of 30% was performed by the PE, RQA–RR and PISL–GH, followed by the RQA–LMAX with 40%. The lowest MTD was assessed by the MDFT with 100.250, followed by RQA–DET with 128.095 and RQA–LMAX with 193.500. The greatest value of MTFA was obtained by RQA–LMAX with 3216.757, followed by RQA–DET with 2290.769 and MDFT with 1253.789. In summary, the best algorithm was the RQA–LMAX, followed by MDFT. Our approach obtained the second worst result.

Figure 24 illustrates the evolution of the Euclidean distance along the measurements obtained by each algorithm applied on the noisy data streams, remembering the lower values are preferred. The x-axis is the level of noise added to the data stream, the y-axis is the Euclidean distance, in which the smaller values are better, and the algorithms are identified by the line style and color at the legend. In this scenario, our approach obtained relatively good results when there is no noise or low level of noise. The algorithm that obtained the best result is the MDFT

Table 12 – Results obtained for the data stream produced with the Transient Logistic Map.

Noise level	Algorithm	MDR	MTD	MTFA	Euclidean
0,00%	PE	0.300	150.095	1602.432	0.866
	MDFT	0.700	**36.667**	**5671.905**	**0.734**
	RQA–RR	0.300	237.313	256.660	1.786
	RQA–DET	0.800	121.250	1280.860	1.773
	RQA–LMAX	0.600	104.000	2017.288	1.038
	PISL–GH (w/land)	**0.200**	154.296	508.590	1.160
	PISL–GH (wo/land)	0.400	125.595	1228.144	0.986
1,00%	PE	0.200	224.646	383.754	1.591
	MDFT	0.700	152.778	**2290.577**	1.359
	RQA–RR	0.300	272.294	239.618	2.067
	RQA–DET	0.600	171.000	1253.895	1.487
	RQA–LMAX	0.600	**109.000**	2164.000	**1.015**
	PISL–GH (w/land)	**0.100**	148.222	420.530	1.145
5,00%	PE	**0.300**	226.651	382.830	1.666
	MDFT	0.600	**100.250**	1253.789	1.235
	RQA–RR	**0.300**	260.785	201.506	1.986
	RQA–DET	0.700	128.095	2290.769	1.267
	RQA–LMAX	0.400	193.500	**3216.757**	**0.828**
	PISL–GH (w/land)	**0.300**	258.823	470.365	1.874

when considering no noise, and the RQA–LMAX when noise is added.

Next, we discuss the results obtained by the algorithms on the data streams produced by the Transient Lorenz System. As first step, we performed the Monte Carlo simulation and obtained the parameters shown in Table 14. Those parameters were used by the algorithms for processing the whole data stream.

Table 13 – Parameters estimated using the Monte Carlo simulation on the data streams produced with the Transient Lorenz Map.

Algorithm	Parameters	EWMA parameters
PE	$m = 5, d = 8$	$ma.n = 10, \eta = 3$
MDFT	$m = 3, d = 8, \text{nbins} = 13$	$ma.n = 1, \eta = 3$
RQA–RR	$m = 3, d = 8, \text{radius} = 0.4$	$ma.n = 1, \eta = 2$
RQA–DET	$m = 4, d = 8, \text{radius} = 0.5$	$ma.n = 20, \eta = 2$
RQA–LMAX	$m = 3, d = 8, \text{radius} = 0.1$	$ma.n = 1, \eta = 6$
PISL–GH	$m = 3, d = 8, \text{landmark method} = \text{random}$	$ma.n = 5, \eta = 3$

Table 14 shows MDR, MTD, MTFA, and their Euclidean distance while running all algorithms on the three data streams – without noise, 1%-noisy and 5%-noisy. In the first data stream, without noise, both algorithms RQA–DET and PISL–GH detected all the concept drifts, thus providing MDR equals to 0%. On the other hand, the RQA–LMAX could not detect a single concept drift. Considering the measurement MTD, the lowest value of 205.961 was obtained by RQA–RR, followed by MDFT with 212.777 and PISL–GH with 213.988. The highest MTFA was obtained by RQA–LMAX with 2160.666, followed by PE with 620.208 and MDFT with

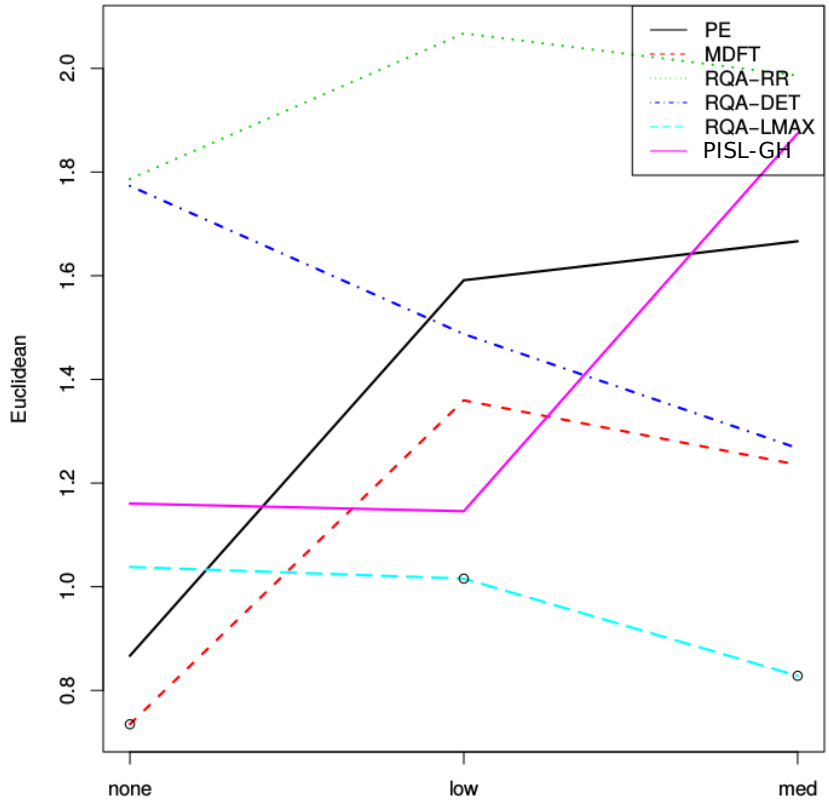


Figure 24 – Transient Logistic Map: score of each algorithm based on the Euclidean distance computed using MDR, MTD and MTFA taking data streams with different noise levels. The x-axis is the data stream noise level and the y-axis corresponds to the Euclidean distance, in which smaller values are better.

318.764. Analyzing the Euclidean distance, our approach PISL–GH obtained the lowest and best result, with 0.925, followed by MDFT with 0.960 and RQA–RR with 1.153. Our approach without landmarks obtained an even better result of 0.655.

Now considering the data stream with 1% of noise, the lowest value of MDR was obtained again by our approach PISL–GH, as it detected all the concept drifts, followed by MDFT, RQA–RR and RQA–DET with 25%. RQA–LMAX again could not detect a single concept drift. In terms of MTD, the lowest value of 162.500 was obtained by MDFT, followed by RQA–DET with 187.214 and RQA–RR with 204.205. Considering the measurement MTFA, the highest value was produced by the RQA–LMAX with 1641.972, followed by PE with 692.326 and PISL–GH with 426.405. The lowest Euclidean distance among measurements was again obtained by our approach with 0.797, followed by MDFT and PE.

On the data stream with 5% of noise, the lowest value of MDR was obtained by MDFT, RQA–RR and PISL–GH with 0%. Again RQA–LMAX could not detect a single concept drift. Unexpectedly PE also could not detect any concept drift either. The lowest value of MTD was obtained again by PISL–GH with 120.089, followed by RQA–RR and RQA–DET with values of 251.458 and 259.000, respectively. The highest value of MTFA was obtained by RQA–

LMAX with 968.483, followed by PE with 810.069 and PISL–GH with 306.324. Considering the Euclidean distance, our approach obtained again the lowest value of 0.799, followed by MDFT with 1.109 and RQA–RR with 1.220.

Table 14 – Results obtained for the data stream produced with the Transient Lorenz System.

Noise level	Algorithm	MDR	MTD	MTFA	Euclidean
0,00%	PE	0.250	344.167	620.208	1.254
	MDFT	0.250	212.777	318.764	0.960
	RQA–RR	0.250	**205.961**	89.593	1.153
	RQA–DET	**0.000**	249.624	98.282	1.202
	RQA–LMAX	1.000	∞	**2160.666**	2.000
	PISL–GH (w/land)	**0.000**	213.988	290.634	**0.925**
	PISL–GH (wo/land)	0.200	133.75	536.81	0.655
1,00%	PE	0.500	266.667	692.326	1.025
	MDFT	0.250	**162.500**	290.027	0.897
	RQA–RR	0.250	204.205	89.729	1.149
	RQA–DET	0.250	187.214	103.226	1.103
	RQA–LMAX	1.000	∞	**1641.972**	2.062
	PISL–GH (w/land)	**0.000**	207.709	426.405	**0.797**
5,00%	PE	1.000	∞	810.069	2.424
	MDFT	**0.000**	267.083	266.532	1.109
	RQA–RR	**0.000**	251.458	86.536	1.220
	RQA–DET	0.250	259.000	108.595	1.287
	RQA–LMAX	1.000	∞	**968.483**	2.330
	PISL–GH (w/land)	**0.000**	**120.089**	306.324	**0.799**

In Figure 25, we show the algorithm results on data streams, considering only the Euclidean distance among measurements. Our approach, PISL–GH, obtained the best result on every scenario, followed by MDFT, which achieved the second overall result on every data stream. The Permutation Entropy assessed relatively good results when no noise or low level of noise were considered, however it obtained the worst result when greater levels of noise were added.

In this second experiment, we assessed four algorithms on two sets of data streams, one produced based on the Transient Logistic Map and another, even more complicated, based on the Transient Lorenz System. Since the data streams were generated by fixed dynamical systems, but changing their parameters along time, the concept drifts are considered gradual, thus they are harder to be detected than the streams used in the first experiment.

In the scenario produced by the Transient Logistic Map, our approach obtained relatively good results, but not enough to outperform MDFT and RQA–LMAX, which obtained the best results. In the second scenario, using the Transient Lorenz System, our approach obtained the best overall result, obtaining the lowest value for the three data streams, both with and without noise. PISL–GH also obtained the lowest value of MDR with 0% in every data stream, showing the approach was capable to identify every concept drift. It also obtained good scores for both MTD and MTFA while compared to the other algorithms, providing the best (or close to it) value

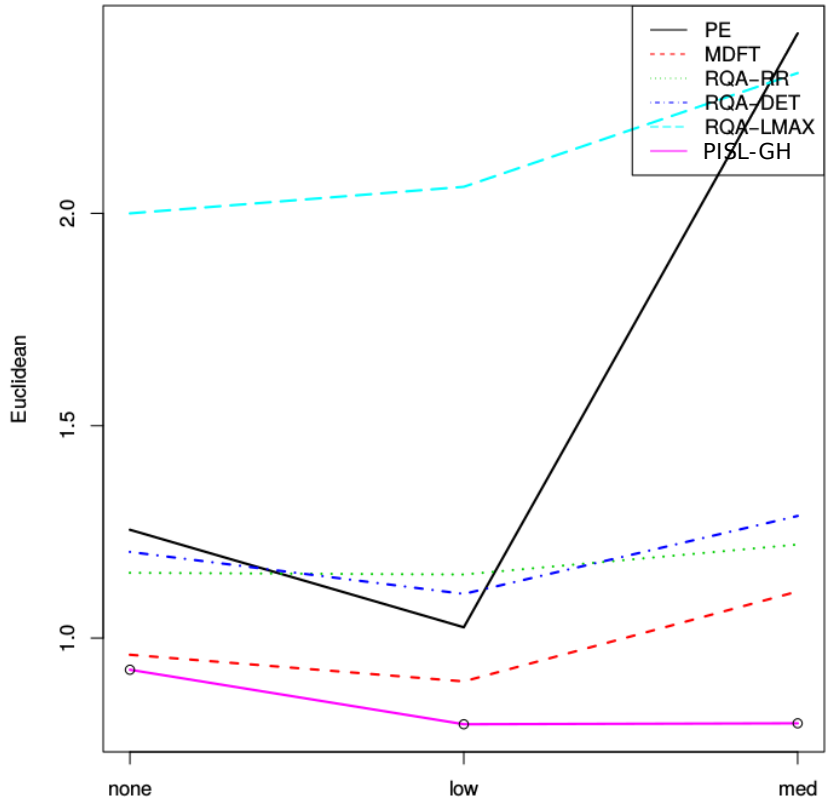


Figure 25 – Transient Lorenz System: score of each algorithm based on the Euclidean distance computed on MDR, MTD and MTFA taking data streams with different noise levels. The x-axis is the data stream noise level and the y-axis corresponds to the Euclidean distance, in which smaller values are better.

for MTD. The second best result was provided by MDFT, also proposed by our research group. The algorithms PE and RQA unexpectedly obtained poor results, specially when dealing with great amount of noise.

5.4 Final considerations

In this chapter, we detailed our approach for detecting concept drifts on data streams, which is the Permutation-Invariant Single-Linkage clustering algorithm with the Gromov-Hausdorff distance (PISL–GH), and assessed it by performing two experiments, one with abrupt and another with gradual changes. When detecting abrupt changes, our approach obtained the third best overall result considering no landmarks, following the algorithms Permutation Entropy and Recurrence Quantification Analysis. However, when considering landmarks, our approach obtained the worst result. On the other hand, in the second set of experiments, involving the detection of gradual drifts, our approach produced much better results. In fact, this latter scenario is considered to be the most relevant and difficult to be analyzed (CAO *et al.*, 2004).

On the two-dimensional data stream, produced using the Logistic Map, the Recurrence

Quantification Analysis obtained again the best results, followed by the Multidimensional Fourier Transform. The PISL–GH obtained a median overall result. On the most complicated scenario based on the three-dimensional data stream from the Lorenz System, our approach obtained the best overall result, considering all noise levels, followed by the MDFT (also designed by our research group (VALLIM; MELLO, 2014)). In this last scenario, the algorithms Permutation Entropy and Recurrence Quantification Analysis provided poor results.

CHAPTER

6

CONCLUSIONS

"I teach you the Overman. Man is something that shall be overcome. What have you done to overcome him? ... The time has come for man to set himself a goal. The time has come to plant the seed to his highest hope."

Friedrich Nietzsche

In this thesis, we tackled two deficiencies of current concept drift detection algorithms while operating on data streams: i) their modeling instability, and ii) their lack of time dependency representation. Motivated by Carlsson & Mémoli (2010)'s theoretical framework, we proposed an unsupervised learning approach to ensure model divergences are indeed associated to data changes. At a first step, this approach samples observations from a data stream using a sliding window along time. Next, it maps every extracted data sample into the phase space using Takens' embedding theorem (TAKENS, 1981) to make data statistically independent and, as consequence, allow the usage of Carlsson and Memoli's framework. Thirdly, we cluster the data samples in the phase space using the Permutation-Invariant Single-Linkage Clustering Algorithm (PISL), designed to respect the stability property proposed by Carlsson & Mémoli (2010). PISL outputs a dendrogram for every sample, which is proven to be equivalent to an ultrametric space. Therefore, we are able to compare two ultrametric spaces through the Gromov-Hausdorff (GH) distance, and then detect concept drifts. As part of the contributions of this thesis, our approach distinguishes from traditional algorithms, specially from the field of Machine Learning (ALBERTINI; MELLO, 2007; VALLIM *et al.*, 2013; GAMA *et al.*, 2010), which normally consider a weak or no dependency among observations, and are driven to linear data.

We compared our approach against others by performing two experiments, one with abrupt, and another with gradual changes. In the first, we built a data stream by concatenating several different dynamical systems, producing abrupt changes. In the second, we generated six data streams, three of them based on the Transient Logistic Map (TRULLA *et al.*, 1996), a two-dimensional nonlinear data with gradual changes, and the other three streams following the Transient Lorenz System (CAO *et al.*, 2004), which comprises a more complicated three-dimensional nonlinear system providing gradual changes. Two different levels of noise, generated with Normal distributions, were added to the streams in this second set of experiments to simulate external influences in the detection process.

In the first experiment, our approach obtained a comparative poor result. Even though it was capable of detecting almost every concept drift, the other algorithms outperformed it in terms of the time delays and false alarms. When considering the data streams with gradual changes and noise added, our approach provided better results than the other algorithms, given it obtained the best result for every stream based on the Lorenz System, the most complicated data streams assessed (CAO *et al.*, 2004). Several algorithms could not even detect concept drifts while tackling the Lorenz streams. Results confirm our approach is capable of detecting concept drifts, both abrupt and gradual ones, however it is more adequate to address more complicated scenarios, since the traditional algorithms provided better results on the simpler ones.

The extent of the experiments motivated the development of the R-package streamChaos, which processes data streams in order to detect concept drift using either PISL–GH or any of the other algorithms assessed in this thesis. We included a set of several nonlinear data stream generators, such as the Logistic Map, the Lorenz Systems, and their transient versions. This package supports the reproduction of experiments by any user, as well as it can be extended for further analysis in other scenarios and with other algorithms. Additionally, as such package was developed based on stream (HAHSLER; BOLANOS; FORREST, 2015), it allows the connection to real-world data streams, for example, financial data and twitter messages.

In summary, the main contributions of this thesis are: i) the usage of Takens' embedding theorem as tool to provide statistical independence to data streams; ii) the implementation of PISL in conjunction with GH (called PISL–GH); iii) a comparison of detection algorithms in different scenarios; and, finally, iv) an R package (called streamChaos) that provides tools for processing nonlinear data streams as well as other algorithms to detect concept drifts, which is, in the period of the write of this thesis, in a web repository [1].

It is worth to mention that several fields of science were studied and assessed while developing this thesis, including: the theoretical foundations of Machine Learning, linear and specially nonlinear time series analysis. This is most certainly part of the indirect contributions of this thesis. The author spent one year of his doctorate at the Max-Planck Institute for the Physics of Complex Systems, Dresden, Germany, under the supervision of Prof. Holger Kantz,

[1] https://github.com/faustogc/streamChaos

the head of the Nonlinear Dynamics and Time Series Analysis research group, and author of several relevant scientific contributions (KANTZ; SCHREIBER, 2004). The main research field of that group involves the processing and modeling of real-world data from **global warming**. During such period, the author had the opportunity to work with other researchers in the same institute, experience their culture, and participate of several study groups together with other people.

As future work, we plan to publish both the additional results reported in this thesis and the package streamChaos. As well, we intend to perform an experiment on real-world data stream, to assess the methodology when dealing with real and noisy data. Moreover, some current limitations of our methodology may be studied in further projects, such as the following:

- One of its main limitations is the restriction to use only unidimensional data stream, since the Taken's embedding theorem was initially developed for this class of data. However, there are other techniques that are able to embed multivariate data, as shown in (DEYLE; SUGIHARA, 2011);

- In our experiments, we assessed the algorithms on data streams with only up to 95% of signal-to-noise ratio, using uncorrelated noise. Thus, it would be compelling to assess lesser levels of signal-to-noise ratio, perhaps using other classes of noise. The article of (CASDAGLI *et al.*, 1991) may be of great use to properly reconstructing the state space in the presence of noise;

- Another limitation of our methodology is that data parameters are fixed all over the data stream, such as the embedding dimension, the time delay, and the size of data windows. It would be smarter to change those values over time, in order to fit the best parameters for the current data stream.

Finally, the following contributions were published:

1. **2017 Expert Systems with Applications (submitted / JCR 2.981)**, da Costa, F.G.; Duarte, F.S.; Vallim, R.M.; de Mello, R.F., *Multidimensional Surrogate Stability to Detect Data Stream Concept Drift*.

2. **2016 International Conference on Nonlinear Science and Complexity**, da Costa, F.G.; de Mello, R.F., *Detecting Dynamical Changes in Data Streams*. São José dos Campos, Brazil.

3. **2016 Expert Systems with Applications (JCR 2.981)**, da Costa, F.G.; Rios, R.A.; de Mello, R.F., *Using Dynamical Systems Tools to Detect Concept Drift in Data Streams*.

4. **2015 International Symposium on Recurrence Plots**, da Costa, F.G.; Rios, R.A.; de Mello, R.F., *Applying dynamical system tools to detect concept drift on data streams*. Grenoble, France.

5. **2014 Brazilian Conference on Intelligent Systems**, da Costa, F.G.; de Mello, R.F., *A Stable and Online Approach to Detect Concept Drift in Data Streams*.

REFERENCES

ABU-MOSTAFA, Y. S.; MAGDON-ISMAIL, M.; LIN, H.-T. **Learning from data**. [S.l.]: AMLBook New York, NY, USA:, 2012. v. 4. Citations on pages 33 and 40.

AGGARWAL, C. C. **A Survey of Stream Clustering Algorithms.** 2013. Citation on page 34.

____. **Data mining: the textbook**. [S.l.]: Springer, 2015. Citations on pages 27 and 31.

AGGARWAL, C. C.; HAN, J.; WANG, J.; YU, P. S. A framework for clustering evolving data streams. In: VLDB ENDOWMENT. **Proceedings of the 29th international conference on Very large data bases-Volume 29**. [S.l.], 2003. p. 81–92. Citations on pages 28 and 35.

ALBERTINI, M. K. **Adaptação de viés indutivo de algoritmos de agrupamento de fluxo de dados**. Tese (Doutorado) — Universidade de São Paulo, 2012. Citations on pages 28 and 32.

ALBERTINI, M. K.; MELLO, R. F. de. A self-organizing neural network for detecting novelties. In: ACM. **Proceedings of the 2007 ACM symposium on Applied computing**. [S.l.], 2007. p. 462–466. Citations on pages 37 and 95.

ALLIGOOD, K. T.; SAUER, T. D.; YORKE, J. A. **Chaos: An Introduction to Dynamical Systems**. [S.l.]: Springer, 1997. ISBN 0-387-94677-2. Citations on pages 42, 43, 45, and 49.

BAENA-GARCÍA, M.; CAMPO-ÁVILA, J. del; FIDALGO, R.; BIFET, A.; GAVALDÀ, R.; MORALES-BUENO, R. Early drift detection method. **ECML PKDD 2006 Workshop on Knowledge Discovery from Data Streams**, 2006. Citations on pages 28 and 31.

BASSEVILLE, M.; NIKIFOROV, I. V. *et al.* **Detection of abrupt changes: theory and application**. [S.l.]: Prentice Hall Englewood Cliffs, 1993. v. 104. Citations on pages 79 and 81.

BOX, G. E.; JENKINS, G. M. **Time series analysis: forecasting and control**. [S.l.]: John Wiley & Sons, 1976. Citations on pages 40 and 61.

BUONTEMPO, C.; BOOTH, B.; MOUFOUMA-OKIA, W. Sahelian climate: past, current, projections. **Met Office Hadley Centre, Devon, UK**, 2010. Citation on page 27.

CAO, F.; ESTER, M.; QIAN, W.; ZHOU, A. Density-based clustering over an evolving data stream with noise. In: **Proceedings of the 2006 SIAM International Conference on Data Mining**. [S.l.: s.n.], 2006. p. 328–339. Citations on pages 32, 33, and 36.

CAO, Y.; TUNG, W.-w.; GAO, J.; PROTOPOPESCU, V. A.; HIVELY, L. M. Detecting dynamical changes in time series using the permutation entropy. **Physical Review E**, APS, v. 70, n. 4, p. 046217, 2004. Citations on pages 30, 55, 59, 60, 80, 81, 85, 87, 93, and 96.

CARLSSON, G.; MÉMOLI, F. Characterization, stability and convergence of hierarchical clustering methods. **The Journal of Machine Learning Research**, MIT Press, v. 99, p. 1425–1470, 2010. Citations on pages 29, 30, 33, 63, 64, 66, 67, 68, 69, 70, 71, 72, 73, 75, 76, 77, 78, and 95.

CASDAGLI, M.; EUBANK, S.; FARMER, J. D.; GIBSON, J. State space reconstruction in the presence of noise. **Physica D: Nonlinear Phenomena**, Elsevier, v. 51, n. 1-3, p. 52–98, 1991. Citation on page 97.

DASZYKOWSKI, M.; WALCZAK, B.; MASSART, D. L. On the optimal partitioning of data with k-means, growing k-means, neural gas, and growing neural gas. **Journal of chemical information and computer sciences**, ACS Publications, v. 42, n. 6, p. 1378–1389, 2002. Citation on page 37.

DEYLE, E. R.; SUGIHARA, G. Generalized theorems for nonlinear state space reconstruction. **PLoS One**, Public Library of Science, v. 6, n. 3, p. e18295, 2011. Citation on page 97.

ECKMANN, J.-P.; KAMPHORST, S. O.; RUELLE, D. Recurrence plots of dynamical systems. **EPL (Europhysics Letters)**, IOP Publishing, v. 4, n. 9, p. 973, 1987. Citation on page 57.

ESTER, M.; KRIEGEL, H.-P.; SANDER, J.; XU, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In: **Kdd**. [S.l.: s.n.], 1996. v. 96, p. 226–231. Citation on page 36.

FRASER, A. M.; SWINNEY, H. L. Independent coordinates for strange attractors from mutual information. **Physical review A**, APS, v. 33, n. 2, p. 1134, 1986. Citations on pages 47, 48, 49, 50, and 52.

GAMA, J.; FERNANDES, R.; ROCHA, R. Decision trees for mining data streams. **Intelligent Data Analysis**, IOS Press, v. 10, n. 1, p. 23–45, 2006. Citations on pages 28 and 34.

GAMA, J.; MEDAS, P.; CASTILLO, G.; RODRIGUES, P. Learning with drift detection. In: **Advances in Artificial Intelligence–SBIA 2004**. [S.l.]: Springer, 2004. p. 286–295. Citations on pages 28 and 33.

GAMA, J.; RODRIGUES, P. P.; SPINOSA, E. J.; CARVALHO, A. C. P. L. F. de. **Knowledge discovery from data streams**. [S.l.]: Citeseer, 2010. Citations on pages 33 and 95.

GUHA, S.; MEYERSON, A.; MISHRA, N.; MOTWANI, R.; O'CALLAGHAN, L. Clustering data streams: Theory and practice. **IEEE transactions on knowledge and data engineering**, IEEE, v. 15, n. 3, p. 515–528, 2003. Citations on pages 27 and 31.

HAHSLER, M.; BOLANOS, M.; FORREST, J. Introduction to stream: An extensible framework for data stream clustering research with r. **Journal of Statistical Software**, 2015. Citation on page 96.

HALPERT, M. S.; BELL, G. D. Climate assessment for 1996. **Bulletin of the American Meteorological Society**, v. 78, n. 5, p. 1038–1038, 1997. Citation on page 32.

HEGGER, R.; KANTZ, H.; MATASSINI, L.; SCHREIBER, T. Coping with nonstationarity by overembedding. **Physical review letters**, APS, v. 84, n. 18, p. 4092, 2000. Citation on page 61.

KANTZ, H.; SCHREIBER, T. **Nonlinear time series analysis**. [S.l.]: Cambridge university press, 2004. v. 7. Citations on pages 29, 44, 53, 76, 78, and 97.

KENNEL, M. B.; BROWN, R.; ABARBANEL, H. D. Determining embedding dimension for phase-space reconstruction using a geometrical construction. **Physical review A**, APS, v. 45, n. 6, p. 3403, 1992. Citations on pages 49 and 53.

KLEINBERG, J. An impossibility theorem for clustering. In: **NIPS**. [S.l.: s.n.], 2002. v. 15, p. 463–470. Citation on page 33.

KLINKENBERG, R. Learning drifting concepts: Example selection vs. example weighting. **Intelligent Data Analysis**, IOS Press, v. 8, n. 3, p. 281–300, 2004. Citation on page 28.

KLINKENBERG, R.; JOACHIMS, T. Detecting concept drift with support vector machines. In: **ICML**. [S.l.: s.n.], 2000. p. 487–494. Citations on pages 28 and 32.

KLINKENBERG, R.; RENZ, I. Adaptive information filtering: Learning drifting concepts. In: CITESEER. **Proc. of AAAI-98/ICML-98 workshop Learning for Text Categorization**. [S.l.], 1998. p. 33–40. Citations on pages 28, 32, 33, and 34.

KRANEN, P.; ASSENT, I.; BALDAUF, C.; SEIDL, T. The clustree: indexing micro-clusters for anytime stream mining. **Knowledge and Information Systems**, Springer, v. 29, n. 2, p. 249–272, 2011. Citations on pages 28, 32, and 36.

KUGIUMTZIS, D. State space reconstruction parameters in the analysis of chaotic time series—the role of the time window length. **Physica D: Nonlinear Phenomena**, Elsevier, v. 95, n. 1, p. 13–28, 1996. Citation on page 76.

LORENZ, E. N. Deterministic nonperiodic flow. **Journal of the atmospheric sciences**, v. 20, n. 2, p. 130–141, 1963. Citations on pages 15, 49, 51, 52, 53, and 54.

LUXBURG, U. V.; SCHÖLKOPF, B. Statistical learning theory: models, concepts, and results. **Elsevier**, 2009. Citation on page 40.

MACQUEEN, J. *et al.* Some methods for classification and analysis of multivariate observations. In: CALIFORNIA, USA. **Proceedings of the fifth Berkeley symposium on mathematical statistics and probability**. [S.l.], 1967. v. 1, n. 281-297, p. 14. Citations on pages 37 and 77.

MARWAN, N.; ROMANO, M. C.; THIEL, M.; KURTHS, J. Recurrence plots for the analysis of complex systems. **Physics Reports**, Elsevier, v. 438, n. 5, p. 237–329, 2007. Citations on pages 57, 58, and 59.

MELLO, R. de; COSTA, F. da; DUARTE, F.; VALLIM, R. Multidimensional surrogate stability to detect data stream concept drift. **Expert Systems with Applications**, Elsevier, 2013. Citation on page 30.

MORETTIN, P. A.; TOLOI, C. M. C. **Análise de Séries Temporais**. São Paulo: Editora Edgard Blücher Ltda., 2004. 535 p. Citations on pages 39, 40, 41, and 42.

OTT, E.; SAUER, T.; YORKE, J. A. **Coping with chaos. Analysis of chaotic data and the exploitation of chaotic systems**. [S.l.]: John Wiley & Sons, 1994. v. 1. Citations on pages 42, 43, 45, 47, and 51.

PAGLIOSA, L. de C.; MELLO, R. F. de. Applying a kernel function on time-dependent data to provide supervised-learning guarantees. **Expert Systems with Applications**, Elsevier, v. 71, p. 216–229, 2017. Citations on pages 29 and 76.

PAVLIDIS, N. G.; TASOULIS, D. K.; ADAMS, N. M.; HAND, D. J. λ-perceptron: An adaptive classifier for data streams. **Pattern Recognition**, Elsevier, v. 44, n. 1, p. 78–96, 2011. Citation on page 31.

ROBERT, C. P. **Monte carlo methods**. [S.l.]: Wiley Online Library, 2004. Citation on page 81.

ROSENSTEIN, M. T.; COLLINS, J. J.; LUCA, C. J. D. A practical method for calculating largest lyapunov exponents from small data sets. **Physica D**, v. 65, p. 117–134, 1993. Citation on page 56.

RÖSSLER, O. E. An equation for continuous chaos. **Physics Letters A**, Elsevier, v. 57, n. 5, p. 397–398, 1976. Citations on pages 15, 47, 48, 49, 50, and 54.

SCHEFLER, W. C. **Statistics: concepts and applications**. [S.l.]: Benjamin-Cummings Publishing Co., Inc., 1988. Citations on pages 78 and 81.

SERRA, X.; ANDRZEJAK, R. G. *et al.* Cross recurrence quantification for cover song identification. **New Journal of Physics**, IOP Publishing, v. 11, n. 9, p. 093017, 2009. Citation on page 59.

SHANNON, C. A mathematical theory of communication. **Bell System Technical Journal**, v. 27, p. 379–423, 623–656, July, October 1948. Citations on pages 37 and 47.

SILVA, J. A.; FARIA, E. R.; BARROS, R. C.; HRUSCHKA, E. R.; CARVALHO, A. C. de; GAMA, J. Data stream clustering: A survey. **ACM Computing Surveys (CSUR)**, ACM, v. 46, n. 1, p. 13, 2013. Citation on page 34.

SILVA, V. D.; CARLSSON, G. Topological estimation using witness complexes. **Proc. Sympos. Point-Based Graphics**, p. 157–166, 2004. Citation on page 77.

TAKENS, F. Detecting strange attractors in turbulence. In: **Dynamical systems and turbulence, Warwick 1980**. [S.l.]: Springer, 1981. p. 366–381. Citations on pages 21, 29, 30, 45, 46, 53, 61, 76, 78, and 95.

TRULLA, L.; GIULIANI, A.; ZBILUT, J.; WEBBER, C. Recurrence quantification analysis of the logistic equation with transients. **Physics Letters A**, Elsevier, v. 223, n. 4, p. 255–260, 1996. Citations on pages 30, 55, 57, 59, 81, 85, and 96.

TSYMBAL, A. The problem of concept drift: definitions and related work. **Computer Science Department, Trinity College Dublin**, Citeseer, v. 106, 2004. Citation on page 28.

VALLIM, R. M.; FILHO, J. A. A.; MELLO, R. F. de; CARVALHO, A. C. de. Online behavior change detection in computer games. **Expert Systems with Applications**, Elsevier, 2013. Citations on pages 28, 37, and 95.

VALLIM, R. M.; MELLO, R. F. de. Proposal of a new stability concept to detect changes in unsupervised data streams. **Expert Systems with Applications**, Elsevier, 2014. Citations on pages 61 and 94.

WALLACE, J. M.; HOBBS, P. V. **Atmospheric science: an introductory survey**. [S.l.]: Academic press, 2006. v. 92. Citation on page 54.

WANG, W.; YANG, J.; MUNTZ, R. Sting: A statistical information grid approach to spatial data mining. In: **VLDB**. [S.l.: s.n.], 1997. v. 97, p. 186–195. Citation on page 28.

ZHANG, T.; RAMAKRISHNAN, R.; LIVNY, M. Birch: an efficient data clustering method for very large databases. In: ACM. **ACM SIGMOD Record**. [S.l.], 1996. v. 25, n. 2, p. 103–114. Citation on page 35.