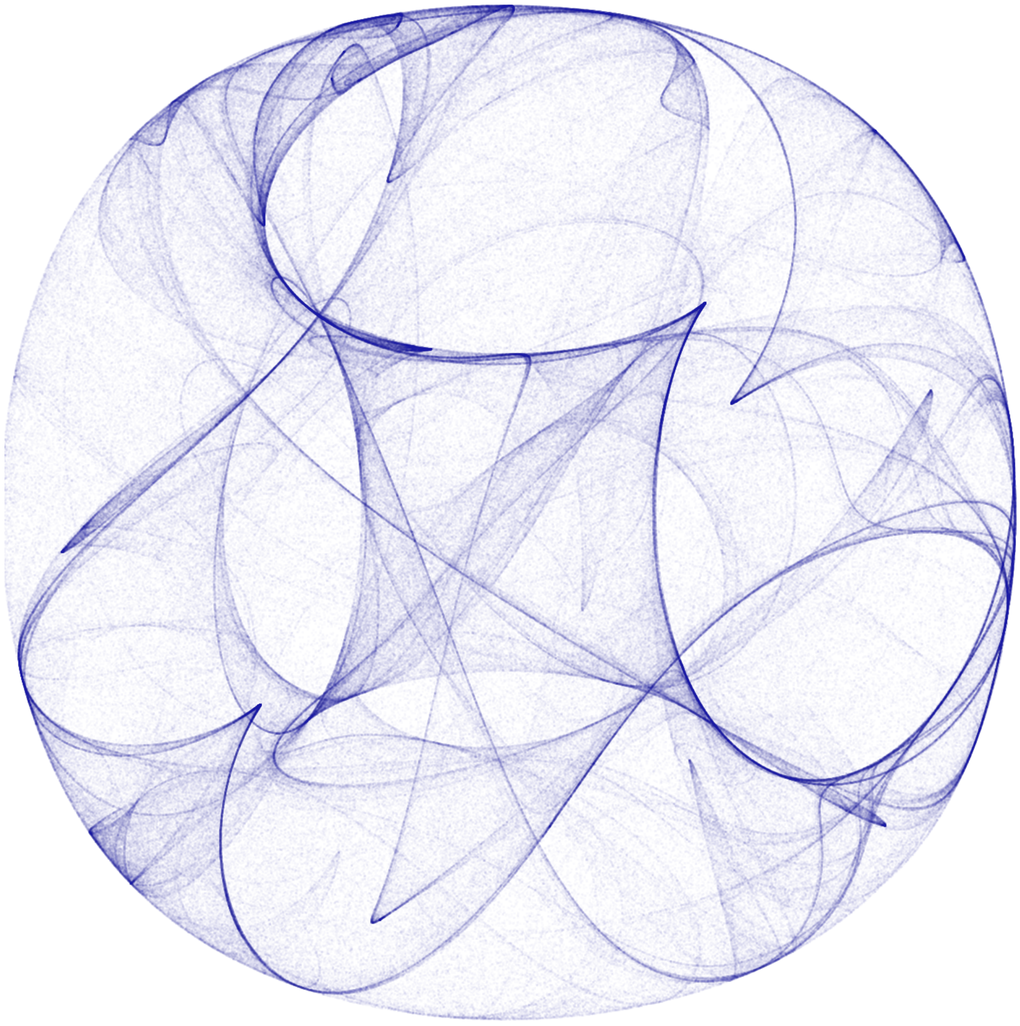
EXPLORING CHAOTIC TIME SERIES AND PHASE SPACES

From Dynamical Systems to Visual Analytics



LUCAS DE CARVALHO PAGLIOSA

The work in this thesis has been carried out as a double-degree PhD in a cooperation between the Scientific Visualization and Computer Graphics (SVCG) research group from the University of Groningen (RuG) and the Bio-inspired Computation (BIOCOM) research group from the University of Sao Paulo (USP)

Cover: Clifford attractor with 1 million states.

Exploring chaotic time series and phase spaces
From Dynamical Systems to Visual Analytics

Lucas de Carvalho Pagliosa
PhD Thesis

ISBN 978-94-034-2245-9 (printed version)
ISBN 978-94-034-2244-2 (electronic version)

Exploring chaotic time series and phase spaces

From Dynamical Systems to Visual Analytics

PhD thesis

to obtain the degree of PhD at the
University of Groningen
on the authority of the
Rector Magnificus Prof. C. Wijmenga
and in accordance with
the decision by the College of Deans.
and
to obtain the degree of PhD at the
University of São Paulo
on the authority of the
Director Prof. M. Oliveira

Double PhD degree

This thesis will be defended in public on

Monday 16 March 2020 at 11.00 hours

by

Lucas de Carvalho Pagliosa

born on 20 December 1991
in Campo Grande, Brazil

“Two things are infinite: the universe and human stupidity; and I’m not sure about the universe.”
— Albert Einstein

ABSTRACT

Technology advances have allowed and inspired the study of data produced along time from applications such as health treatment, biology, sentiment analysis, and entertainment. Those types of data, typically referred to as time series or data streams, have motivated several studies mainly in the area of Machine Learning and Statistics to infer models for performing prediction and classification. However, several studies either employ batch-driven strategies to address temporal data or do not consider chaotic observations, thus missing recurrent patterns and other temporal dependencies especially in real-world data. In that scenario, we consider Dynamical Systems and Chaos Theory tools to improve data-stream modeling and forecasting by investigating time-series phase spaces, reconstructed according to Takens' embedding theorem.

This theorem relies on two essential embedding parameters, known as embedding dimension m and time delay τ, which are complex to be estimated for real-world scenarios. Such difficulty derives from inconsistencies related to phase space partitioning, computation of probabilities, the curse of dimensionality, and noise. Moreover, an optimal phase space may be represented by attractors with different structures for different systems, which also aggregates to the problem.

Our research confirmed those issues, especially for entropy. Although we verified that a well-reconstructed phase space can be described in terms of low entropy of phase states, the inverse is not necessarily true: a set of phase states that presents low levels of entropy does not necessarily describe an optimal phase space. As a consequence, we learned that defining a set of features to describe an optimal phase space is not a trivial task.

As alternative, this Ph.D. proposed a new approach to estimate embedding parameters using an artificial neural network training on an overestimated phase space. Then, without the need of explicitly defining any phase-space features, we let the network filter non-relevant dimensions and learn those features implicitly, whatever they are. After training iterations, we infer m and τ from the skeletal architecture of the neural network. As we show, this method was consistent with benchmarks datasets, and robust in regarding different random initializations of neurons weights and chosen parameters.

After obtaining embedding parameters and reconstructing the phase space, we show how we can model time-series recurrences more effectively in a wider scope, thereby enabling a deeper analysis of the underlying data.

SAMENVATTING

Technologische vooruitgangen hebben de studie van tijdsafhankelijke data mogelijk gemaakt in toepassingen zoals gezondheidszorg, biologie, sentimentanalyse, en entertainment. Dit type data, ook bekend als tijdseries of *data streams*, hebben geleid tot verschillende studies vooral op het gebied van *machine learning* en statistiek om modellen te infereren voor predictie en classificatie. Niettemin de meerderheid van deze studies gebruiken *batch-driven* strategieën voor tijdsafhankelijke data-analyse of, anders, ze benaderen chaotische observaties niet; dit mist recurrente patronen en andere tijdsafhankelijkheden in vooral reële data. In deze gevallen gebruikt men instrumenten van dynamische systemen en chaostheorie om het modelleren en voorspellen van *data streams* door de fase-ruimte van deze *time series* te analyseren volgens het theorem van Takens.

Dit theorem maakt gebruik van twee essentiële parameters – de *embedding* dimensie m en tijdsvertraging τ, die zijn moeilijk te schatten voor reële data. Deze uitdagingen stammen uit inconsistenties betreffend het partitioneren van de fase-ruimte, kansberekening, de zogenaamde *curse of dimensionality*, en ruis. Verder kan een optimale fase-ruimte gerepresenteerd worden door attractoren met verschillende structuren voor verschillende systemen, wat het probleem nog complexer maakt.

Ons onderzoek heeft deze problemen bevestigd, met name wat de entropie betreft. Hoewel we hebben geverifieerd dat een goede reconstructie van de fase-ruimte beschreven kan worden in termen van een lage entropie van de fase-ruimte, het omgekeerde is niet noodzakelijk waar: Fase-ruimtes met lage entropieniveau's zijn niet noodzakelijk optimaal. De consequentie is dat het definiëren van parameters die optimale fase-ruimtes beschrijven is verre van simpel.

Als een alternatief, ons werk stelt een nieuwe benadering voor voor het schatten van *embedding* parameters met gebruik van een kunstmatig neuraal netwerk of een overgeschatte fase-ruimte. Dit stelt ons in staat om het netwerk niet-relevante dimensies te laten filteren en de nodige paramet ers te laten leren, welke dan ook, zonder een expliciete definitie van fase-ruimte parameters. Na *training*, we schatten m en τ vanuit de skeletarchitectuur van het netwerk. We laten zien dat deze methode consistent is met *benchmark* datasets en ook robuust ten opzichte van willekeurige initialisatie van de neurongewichten en andere parameters.

Na het schatten van de *embedding* parameters en reconstructie van de fase-ruimte we laten zien hoe wij tijdsserie-recurrenties effectief kunnen modelleren voor een groot bereik van gevallen, wat verder een diepere analyse van de onderliggende data mogelijk maakt.

RESUMO

Avanços tecnológicos permitiram e inspiraram o estudo de dados produzidos ao longo do tempo a partir de aplicativos como tratamento de saúde, biologia, análise de sentimentos e entretenimento. Esses tipos de dados, geralmente chamados de séries temporais ou fluxos de dados, motivaram vários estudos principalmente na área de Aprendizado de Máquina e Estatística a inferir modelos para realização de previsões e classificações. No entanto, vários estudos empregam estratégias orientadas por lotes para tratar dados temporais ou não consideram observações caóticas, perdendo assim padrões recorrentes e outras dependências temporais especialmente em dados do mundo real. Nesse cenário, consideramos as ferramentas de Sistemas Dinâmicos e Teoria do Caos para melhorar a modelagem e previsão do fluxo de dados investigando os espaços fase das séries temporais, reconstruídos de acordo com o teorema de mergulho de Takens.

Esse teorema baseia-se em dois parâmetros essenciais de mergulho, conhecidos como dimensão de mergulho m e tempo de atraso τ, que são complexos de serem estimados para cenários do mundo real. Essa dificuldade deriva de inconsistências relacionadas ao particionamento do espaço fase, ao cálculo de probabilidades, à maldição da dimensionalidade e à ruídos. Além disso, um espaço fase ideal pode ser representado por atratores com estruturas diferentes para sistemas diferentes, o que também se agrega ao problema.

Nossa pesquisa confirmou esses problemas especialmente para entropia e, embora tenhamos verificado que um espaço fase bem reconstruído pode ser descrito em termos de baixa entropia de seus estados, o inverso não é necessariamente verdadeiro: um conjunto de estados do espaço fase que apresenta baixos níveis de entropia não descreve necessariamente um espaço fase ideal. Como conseqüência, aprendemos que definir um conjunto de recursos para descrever um espaço fase ideal não é uma tarefa trivial.

Como alternativa, este doutorado propôs uma nova abordagem para estimar parâmetros de mergulho a partir do treinamento de uma rede neural artificial em um espaço fase superestimado. Então, sem a necessidade de definir explicitamente quaisquer características de espaço fase, deixamos a rede filtrar dimensões não relevantes e aprender essas caractereísticas implicitamente, sejam elas quais forem. Após o treinamento das iterações, inferimos m e τ a partir da arquitetura esquelética da rede neural. Como mostramos, esse método mostrou-se consistente com conjuntos de dados conhecidos, e robusto em relação a diferentes inicializações aleatórias de pesos de neurônios e parâmetros da rede.

Após obter os parâmetros de mergulho e reconstruir o espaço fase, podemos modelar as recorrências de séries temporais com mais eficiência em um escopo mais amplo, prosseguindo para uma análise mais profunda dos dados.

PUBLICATIONS

This thesis is the result of the following publications:

- L. de Carvalho Pagliosa, R. F. de Mello (2017) Applying a Kernel Function on Time-Dependent Data to Provide Supervised-Learning Guarantees. *Expert Systems with Applications* vol. 71, pp. 261-229 (Chapter 5).

- L. de Carvalho Pagliosa, R. F. de Mello (2018) Semi-supervised time series classification on positive and unlabeled problems using cross-recurrence quantification analysis. *Pattern Recognition* vol. 80, pp. 53-63 (Chapter 6).

- L. de Carvalho Pagliosa, A. Telea (2019) RadViz++: Improvements on Radial-Based Visualizations. *Informatics* vol. 6, nr. 2, 16 (Chapter 8)

- L. de Carvalho Pagliosa, R. F. de Mello (2019) On Theoretical Guarantees to Ensure Concept Drift Detection on Data Streams – Submitted (Chapter 7)

- L. de Carvalho Pagliosa, A. Telea, R. F. de Mello (2019) Estimating Embedding Parameters using Neural Networks – Submitted (Chapter 9).

CONTENTS

1 INTRODUCTION

1.1 CONTEXT AND MOTIVATION

Technology advances have allowed and inspired the study of data produced from domains such as health treatment, biology, sentiment analysis, entertainment, the financial markets and many more (Tucker, 1999; Robledo and Moyano, 2007). Typically, such data is modeled as data collections, or datasets, consisting of a large number of observations (also called samples), each of which captures the phenomenon of interest by one or more measurements of its properties along so-called dimensions, variables, or attributes. In this context, researchers from several areas of science such as Data Mining (Ester et al., 1996; Hodge and Austin, 2004), Natural Language Processing (Indurkhya and Damerau, 2010), and Information Visualization (Ward et al., 2010; Telea, 2014; Munzner, 2014) have proposed different approaches within their research scope and concepts (and some times uniting efforts) to analyze large data collections to extract actionable conclusions. In addition to the difficulty of extracting information from large, multidimensional and multivariate data, there are cases where data changes along *time*. Such datasets characterize typically more complex scenarios referred to as *time-series* or *data-streams* analysis (Farmer and Sidorowich, 1987; Kantz and Schreiber, 2004; Muthukrishnan, 2005). When dealing with such scenarios, in addition to batch-driven studies such as classification and searching for patterns, clusters, and outliers, *forecasting* is usually the most important task, typically performed in the context of Machine Learning and Dynamical Systems (Hitzl, 1981; Tucker, 1999; Robledo and Moyano, 2007; de Mello, 2011; Vallim and De Mello, 2014; da Costa et al., 2017).

When analyzing time series, it is worth to recall that the variable *time* has as much importance as the raw values of observations themselves, so that the *data order* is crucial for analysis. Thus, instead of employing traditional batch-driven approaches, *e.g.*, by directly applying some regression function along raw data (Waibel et al., 1990; Postolache et al., 1999; de Mello, 2011) or using data visualization methods to discover patterns in the time series (Wong and Bergeron, 1997; Ward et al., 2010), it is mandatory to also consider temporal recurrences (trends, cycles, and trajectories) while modeling, which usually leads to better forecasting results. In this sense, researchers generally tackle time series by assuming they have either a deterministic or a stochastic bias. Nevertheless, due to diverse reasons (inherent signal noise, acquisition problems, restricted float number representation, or even the nature of the analyzed phenomenon itself), it is common to find series composed of both deterministic and stochastic behaviors in conjunction, a well-known example being the Sunspot dataset (Andrews and Herzberg, 1985). Therefore, methods have been proposed to decompose time series into both stochastic and deterministic components (Graben, 2001; Ishii et al., 2011; Rios, 2013), and, consequently, focus on studying subsequent aspects of linearity and stationarity, as shown in Figure 1.1.

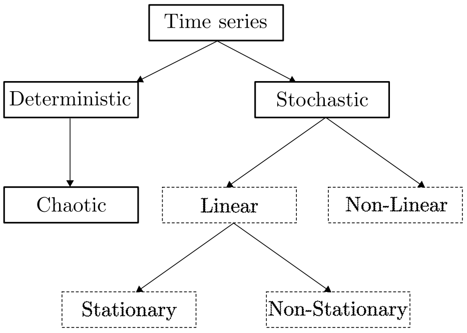


Figure 1.1: The first step in time-series analysis usually consists of verifying whether the series has a deterministic or a stochastic bias. This process is mainly based on measuring the number of recurrences the series has, what can be inferred from the series itself or through its phase space. Chaos, on the other hand, is mainly detected using phase-space measurements. The solid-line boxes represent phase-space-based steps. Dashed-line boxes represent the out-of-scope analysis usually computed directly on the time series. Despite important, those are not covered in this thesis as we predominantly deal with deterministic series.

When dealing with a predominantly stochastic time series, one common approach is to use statistical-based tools such as the ARIMA models (Box and Jenkins, 2015) to describe time-series components, which include random behavior (*e.g.*, Normal and Uniform distributions). As the main advantage, this strategy permits each type of component to be modeled using the most adequate tool available for it (Graben, 2001; Rios and de Mello, 2013). On the other hand, for predominantly deterministic series, especially those derived from natural phenomena, physicists (Kennel et al., 1992) typically rely on Dynamical Systems and Chaos Theory [1] to

1 A chaotic system has strong sensitiveness to initial conditions, so that it tends to evolve to completely different orbits (Alligood et al., 1996; Ott, 2002; Kantz

map the series into a multidimensional space referred to as *phase space* (Takens, 1981). In this space, the dynamics of the studied phenomenon are (hopefully) bound by a so-called *attractor*: this is a lower-dimensional manifold that depicts how the series changes over any given interval of time. The main advantage of using phase-space representations is that they factor out the importance of the time variable, thereby making the analysis simpler (Pagliosa and de Mello, 2017).

Regarding this transformation, also known as the *kernel function*, three main methods were proposed to reconstruct the phase space from a time series:

- the method of derivatives (Packard et al., 1980);
- the method of delays or Takens' theorem (Whitney, 1936; Takens, 1981);
- a method based on singular value decomposition (Broomhead and King, 1986).

Despite there is no formal evidence on which of the above three methods is the most appropriate, Ravindra and Hagedorn (1998) suggest that the Takens' embedding theorem leads to more consistent results when analyzing nonlinear time series. Indeed, this is the most used method in the literature of Dynamical Systems for phase-space reconstruction (Alligood et al., 1996; Kantz and Schreiber, 2004). Such theorem defined that, given a time series T_i formed by observations of a single variable $i \in [1, d]$ from the d-dimensional system $\mathcal{S}^d$ (representing the underlying phenomenon under analysis), the dynamics of $\mathcal{S}^d$ could be reconstructed into an m-dimensional phase space, if points on that space, typically referred to as *phase states*, were formed by m observations time-shifted τ units along T_i. We describe this process with more details in Chapter 2.

Nevertheless, the method of delays also has some important limitations, as follows. First, Takens' theorem stated nothing about the embedding pair (m, τ), only that a "sufficient" phase space can be properly unfolded when the embedding dimension m is greater or equal to $2d+1$. In practical scenarios, however, this information is not helpful since most time series are derived from experimental data: nothing is known about the phenomenon of origin and the dimension d. Furthermore, despite being a simple and effective approach to reconstruct phase spaces from time series, the method of delays is very sensitive to the choice of the parameters m and τ. Different values of these parameters lead to completely different reconstructions and, consequently, conclusions about the time series.

and Schreiber, 2004; Boccaletti and Bragard, 2008), typically giving the wrong impression of randomness.

To alleviate this, several approaches were proposed to estimate the time delay τ and the embedding dimension m, each of them presenting benefits and drawbacks for different scenarios. In this scenario, even with their limitations including sensitiveness to noise and lack of consistency, False Nearest Neighbors (Kennel et al., 1992) and Auto-Mutual Information (Fraser and Swinney, 1986) (details in Chapter 4) are the most employed methods to estimate of m and τ, respectively (see Chapter 4 for details on the related work). All in all, the above leads, in our view, to a **gap in the literature: there is no method robust enough to estimate embedding parameters for general time series**.

Once the phase space is sufficiently unfolded, one can take advantage of the system dynamics to assess crucial assets such as: i) understand and visualize low-dimensional attractors (Section 2.3.2); ii) measure the amount of the time-series determinism (Marwan et al., 2007; Serrà et al., 2009; Marwan and Webber, 2015); iii) identify and model chaos, *i.e.*, measure how initial conditions impact on next series observations; iv) forecast time series recursively (Farmer and Sidorowich, 1987; Myers et al., 1992; Farmer and Sidorowich, 1987; Meng and Peng, 2007; de Mello and Yang, 2009; Bhardwaj et al., 2010); and v) propose ways to interfere and control the underlying phenomenon (Boccaletti and Bragard, 2008).

1.2 OBJECTIVE, HYPOTHESIS AND RESEARCH QUESTIONS

Given the periodic behavior of real-world phenomena (Andrews and Herzberg, 1985; Tucker, 1999), the main motivation behind this research is to explore and understand phase spaces for improving time-series modeling. However, as the phase space firstly needs to be reconstructed and, based on the current gap in the literature (Fraser and Swinney, 1986; Kennel et al., 1992) outlined in the previous section, **the main objective of this thesis is to improve the estimation of the embedding dimension m and the time delay τ.**

We established the above objective as follows. Based on studies on the dynamics of well-known chaotic systems, we noticed that after increasing m and τ up to a certain limit (and sometimes this limit can be the minimum pair, as shown next), the optimal embedding usually presents the most well-structured attractor. Indeed, such behavior is expected for deterministic time series defined by maps or partial differential equations, as one state leads to exactly a single other in the future. For two-dimensional phase spaces, for instance, such relationship is easier to be tracked using cobweb plots (Alligood et al., 1996), where the route of trajectories is illustrated by lines connecting consecutive phase states.

Figure 1.2 depicts the cobweb plot for the Logistic map, whose optimal phase space can be unfolded with $m = 2$ and $\tau = 1$ (details on Section 3.2.2).

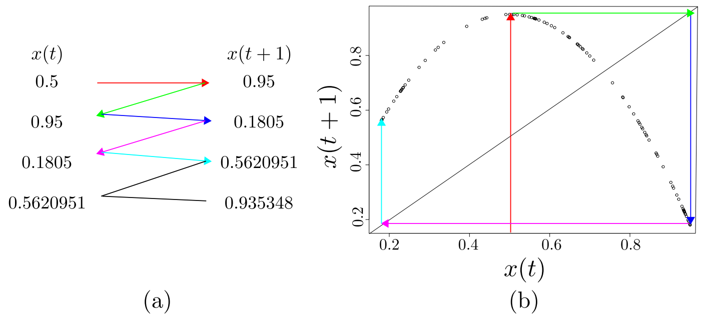


Figure 1.2: **(a)** Each row represents one phase state of the Logistic map phase space, reconstructed using the optimal embedding pair $(m = 2, \tau = 1)$. **(b)** The cobweb plot uses the diagonal line $x = y$ to guide the drawing of trajectories.

In cobweb plots, the dynamics of the system can be observed after connecting the first $m - 1$ dimensions of a state with its latter one, and then going back to the first $m - 1$ dimensions on the next state. Then, the trajectories are drawn by executing this process iteratively for all states. In Figure 1.2, this is represented by projecting each dimension ($x(t)$ and $x(t + 1)$) into the diagonal line $x = y$. As it can be noticed, the created line tends to hit (if a sufficient small open ball around each state is considered) a single state during the course of trajectories. Thus, even that such unique correspondence does not occur for generic datasets, it is expected well-reconstructed phase spaces to have minimum levels of ambiguity. Moreover, after increasing embedding parameters too excessively, the attractor starts to fade, eventually losing its structure. Figure 1.3 illustrates this process, also known as irrelevance, after setting $\tau = 10$. Conversely, too small values of embedding pairs (m, τ) lead to redundant phase spaces usually characterized by hyper-diagonal attractors. In those cases, the embedding does not have enough information to unfold the system dynamics. Based on both extremes, it was noticed that a sufficient phase space should present some balance between the expansion and contraction of phase states (Rosenstein et al., 1994). In this sense, the concept of entropy (Hammer et al., 2000; Han et al., 2012) seemed a good measurement to describe these behaviors. This enables us to outline the central hypothesis explored in this thesis:

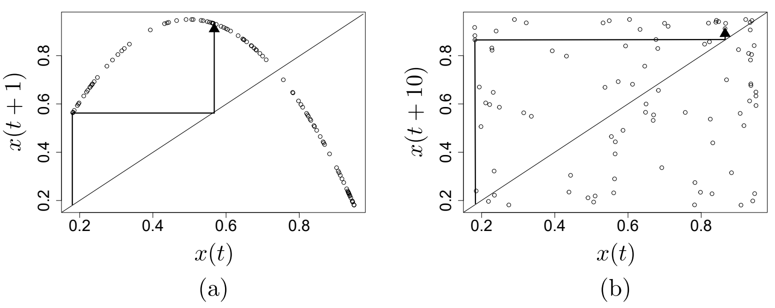


Figure 1.3: **(a)** The cobweb plot applied over the optimal $(m = 2, \tau = 1)$ and **(b)** overestimated $(m = 2, \tau = 10)$ phase spaces for the Logistic map. As one can notice, orbits (arrowed line) intersect many more states in the overestimated space, reinforcing that entropy can be used as guideline to validate phase-space reconstructions.

Hyphotesis. "The reduction of entropy, measured in function of the trade-off between irrelevance and redundancy of phase states, is a sufficient criterion to estimate the time delay and the embedding dimension required to reconstruct the phase space from a univariate time series, therefore supporting the analysis and prediction of deterministic and chaotic phenomena."

However, as we observed in the course of our work (see Section 5.7 for details), entropy by itself showed to be insufficient to derive such conclusions. As such, the above hypothesis was proved wrong. Although a negative result, we believe that this insight is an important and useful contribution to the research on Dynamical Systems.

As part of the methodology to investigate the above-mentioned hypothesis, several Dynamical Systems concepts and methods referring to Chaos Theory and phase spaces were analyzed. This eventually led to different research questions (RQ), as outlined next:

RQ1. Does the optimal phase space have indeed low levels of entropy?

RQ2. Is it better to use phase-space rather than time-series modeling?

RQ3. How to ensure learning in concept-drift scenarios?

RQ4. How to correlate time-series and phase-space attributes?

RQ5. Can neural networks estimate Takens' embedding parameters?

To answer these research questions, we have first designed and employed analysis methods based on Machine Learning (ML). However, as outlined by the case study proposed by Anscombe (1973) (Figure 1.4), which shows different datasets having identical statistical measurements, traditional ML approaches may not be sufficient to discriminate the data initially, and a first analysis based on different approaches might be required. As alternative, visualizing the data lies among the most considered options (Wong and Bergeron, 1997). Therefore, we decided to also consider Information Visualization metaphors to highlight insights about the system dynamics and compare multiple embeddings at the same time. Thus, we implemented our ML methods in the R programming language (R Development Core Team, 2008), chosen due to its simple algebraic manipulation, chaos-related packages (Hegger et al., 1999; Antonio, 2013; Garcia and Sawitzki, 2015), and compactness; and our visualization tools were designed in JavaScript using D3 (Bostock et al., 2011), due to the its easy plotting and interaction support. This allowed us to develop a visual analytics approach where different underlying techniques (from Dynamical Systems, Machine Learning, and Visualization) were tightly combined to address our research questions.

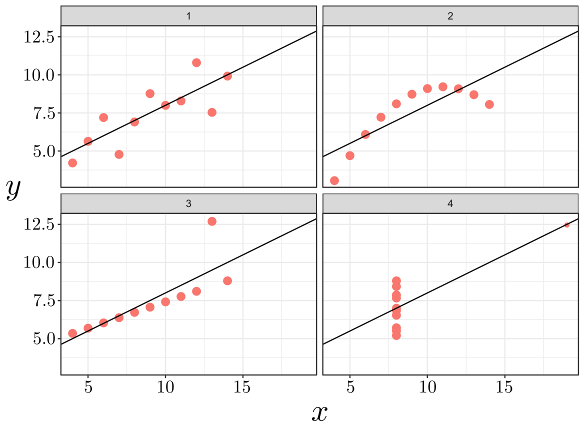


Figure 1.4: The Anscombe quartet shows the importance of data visualization. All datasets share several identical statistical measurements, such as variance and mean (for x and y axis), linear regression line (in black), and coefficient of determination (R squared) (McClave, 2006). However, the datasets are different.

1.3 THESIS STRUCTURE

The remaining of this thesis is organized as follows.

Chapter 2 presents the nomenclature and main concepts used throughout this manuscript. This material serves to formally define the context of our work, as well as important terms and techniques subsequently referred to in the next chapters.

Chapter 3 details the chaotic time series typically used as benchmarks in the dynamic systems literature, as well as their most accepted phase spaces. Although there are more types of time series of interest (and used as study object in Dynamical Systems), we mainly focused on datasets whose generating rule is known, so it is possible to compare and validate the reconstructed phase spaces.

Chapter 4 describes the state of the art to estimate embedding parameters. Given the central scope of this thesis in Dynamical Systems, this chapter focused on the related work concerning mainly our hypothesis, *i.e.*, phase-space reconstruction. Complementary, the related work covering topics from Machine Learning, Statistical Learning Theory, and Information Visualization, important when addressing the refined research questions (RQ1 to RQ5), is discussed as needed in the corresponding chapters.

Chapter 5 details our initial study to correlate optimal phase spaces with their entropy (RQ1). As entropy is a sensitive measurement difficult to be computed in practice, we then relied on the dependence of phase states (a measure proportional to their entropy (Myers et al., 1992)) to validate such a relationship. Although we empirically show that optimal phase spaces indeed present the highest independence among their states, we could not find any relation to deterministically estimate the optimal embedding given its entropy levels.

Chapter 6 effectively shows how phase-space methods can improve analysis of time series when compared to raw (based on the time series itself) data, therefore tackling RQ3. The study was performed on the classification of positive and unlabeled data in a semi-supervised scenario. Of course, this does not mean that the phase space will always lead to better results for general cases, but rather that Dynamical-Systems methods worth being considered when analyzing time series.

Chapter 7 presents a set of conditions that a concept drift algorithm should respect to ensure learning while parsing time series (RQ3). Although this line of research seems to be orthogonal to this Ph.D. hypothesis, it is related to it since reconstructing the phase space is one of the required steps in our proposed methodology. Moreover, this chapter is a consequence of our first study, in which we assumed data was provided under a controlled environment with a fixed distribution, so the Statistical Learning Theory framework could be used to tackle RQ1.

Chapter 8 describes a novel visualization tool to simultaneously explore the attributes and dimensions of multidimensional datasets.

We have improved the related work of radial-based visualizations in terms of exploration, scalability and decreasing of ambiguities. Despite designed to deal with general types of data, this visual metaphor could have been used to correlate time-series and phase-space attributes (RQ4). Nonetheless, due to the lack of time, this analysis has been left for future work.

Chapter 9 assesses our final proposal for estimating the embedding parameters. After verifying the difficulty in correlating optimal phase states with states-based measurements, we decided to rely on an artificial neural network (RQ5) to automatically learn optimal features, whatever they are. As we have shown, despite results depend on a certain level of interpretability, we describe a robust and deterministic method to estimate embedding parameters. We validated our approach against different scenarios of noise, input parameters, and benchmark datasets.

Chapter 10 summarizes the work conducted during our research. We reflect upon our attempts to prove the key hypothesis and discuss the implications of our main conclusion, namely that our current insights showed our hypothesis has been disproved. Finally, we summarize our contributions and suggest directions for future work.

2 FUNDAMENTALS

2.1 INITIAL CONSIDERATIONS

This chapter introduces and describes relevant concepts related to time series (Section 2.2), Dynamical Systems (Section 2.3), and embeddings (Section 2.4). Next, we combine all theory to show how to reconstruct phase spaces from time series in practice (Section 2.5), to finally proceed to important analysis based on phase-states features (Section 2.6).

As noted in Chapter 1, related work encompasses, apart from the fundamental concepts and results related to time series (our main focus), also work in Statistical Learning Theory, Machine Learning, and Information Visualization. Since this second type of related work pertains specifically to the techniques addressing individual research questions, we introduce and describe it only when needed along Chapter 5 to Chapter 9[1].

2.2 TIME SERIES

A univariate[2] time series T_i is the sequence of n observations

$$T_i = \{x(0), x(1), x(2), \cdots, x(n-1)\},\ \ x_k \subset \mathbb{R}, \tag{2.1}$$

that models the evolution of some variable i (*e.g.*, wind speed, relative humidity of the air), representing a feature of some phenomenon of interest (*e.g.*, weather) during an interval of time. In practical scenarios, T_i is formed after collecting impulses from or solving mathematical equations describing (Butcher, 1996) the phenomenon at sampling rate t_s, which defines the time elapsed between two consecutive observations. Moreover, sampling rates t_s can be kept constant (Figure 2.1) or change along time, depending on the target application.

Along this manuscript, the subscripted index (such as i in T_i) is also appended to the time-series corresponding features, such as

1 For some of the topics included in this chapter, different definitions were used in the published articles. Thus, although we have tried our best to standardize the nomenclature in this thesis, the reader may find some divergences when comparing the following chapters with their corresponding articles.

2 In the context of this manuscript, we approach unidimensional time series only. However, the studies developed in this thesis and in the produced articles can be extended to multiple dimensions as performed in (Serrà et al., 2009), without loss of generality.

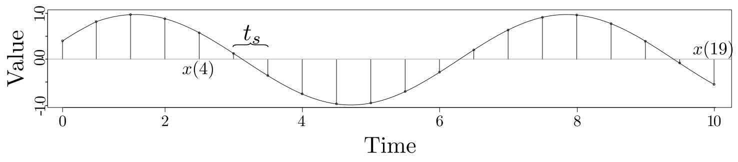


Figure 2.1: Example of time series with $n = 20$ observations, each sampled after $t_s = 0.5$ seconds.

the number of observations, average, or variance. Further, aggregated indexes will denote variations of the same series. For instance, T_i and T_j represent two different series, while T_{ij}, for $j \in [1, s]$, denotes one of the s modifications of T_i. In the last case, the features of modified series remain identical to the original by default (*i.e.*, number of observations, average, etc.) unless explicitly stated differently. Such notations become useful when comparing phase spaces (Section 2.3.3) and dealing with surrogate data (Theiler et al., 1992).

In addition to the sampling rate t_s, other features such as the length of the series, the initial observation $x(0)$ (especially for chaotic data), and the amount of noise (Graben, 2001) also need to be considered when analyzing a time series. These additional features help quantifying various statistical properties of interest, and are useful to identify when different series might still represent the same phenomenon of interest. Figure 2.2 illustrates the idea on observations from one of the variables of the Lorenz system (Tucker, 1999). As it can be seen by this figure, a time series from the Lorenz system can be represented in different ways. Then, the robustness of some model can be tested against variations of the same series. Nonetheless, we expect the series to be large (at least 1000 observations) and "clean" enough to preserve the nature of the measured variable. According to our point of view, this is not too much to ask for, as no relevant models can really be inferred from too small-noisy datasets. In other words, the series must have sufficient information to unfold the dynamics of the generating rule (see Section 2.3 for details).

Apart from the above, additional notations include the time delay $\tau \in \mathbb{I}^+$, representing the number of observations to be shifted from the current timestamp t, such that $x(t \pm \tau) \in T_i$; and the leap time $\rho \in \mathbb{I}^+$, a moment in the future to be forecasted as a *single* observation.

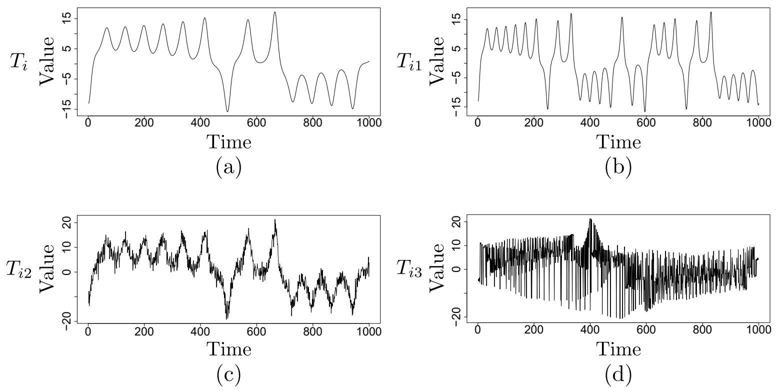


Figure 2.2: Different representations of the Lorenz system. **(a)** T_i is the "original" time series, generated using $t_s = 0.01$. **(b)** T_{i1} shows a variation using $t_s = 0.02$; **(c)** T_{i2} has an additional noise following a Normal distribution $\mathcal{N}(0, 4)$, with zero mean and standard deviation equal to 2. **(d)** T_{i3} is a surrogate generated by the iAAFT method (Schreiber and Schmitz, 1996), which attempts to preserve the linear structure and the amplitude distribution.

2.3 DYNAMICAL SYSTEMS

A dynamical system $\mathcal{S}^d = \{\boldsymbol{p}_0, \cdots, \boldsymbol{p}_\infty\}$ is a set of d-dimensional states (also known as points) $\boldsymbol{p}_t = [p_{t,1}, p_{t,2}, \cdots, p_{t,d}]$[3] that, driven by a generating (also called governing) rule $R(\cdot)$, models the behavior of some phenomenon as a function of state trajectories so that

$$R : \mathcal{S}^d \to \mathcal{S}^d, \tag{2.2}$$

where d corresponds to the number of degrees of freedom the system has, *i.e.*, the number variables required to describe $R(\cdot)$. Further, although $\mathcal{S}^d$ can consist of an infinity of states, in practical terms, a dynamical system is usually represented by the finite set $\mathcal{S} = \{\boldsymbol{p}_0, \boldsymbol{p}_1, \cdots \boldsymbol{p}_N\} \subset \mathcal{S}^d$ of N states.

2.3.1 *Types Of Dynamical Systems*

Dynamical systems can be classified in function of their generating rule, as described below.

3 When referring to time-series and phase-spaces attributes, indices will start at 0. For other contexts, such as indices denoting dimensions or position in arrays, we start counting from 1.

First, the generating rule is classified either as *discrete* or *continuous*, as follows.

Discrete rules: Also known as maps, discrete rules are functions of the form $F = \{f_1, f_2, \cdots, f_d\}$ that explicitly relate states based on past values (defining their trajectories) so that $\boldsymbol{p}_{t+1} = F(\boldsymbol{p}_t)$ and

$$\mathcal{S}^d = \{F(\boldsymbol{p}_0), F^2(\boldsymbol{p}_0), \cdots, F^\infty(\boldsymbol{p}_0)\}. \tag{2.3}$$

In the above, $F^2(\boldsymbol{p}_t) = F(F(\boldsymbol{p}_t))$, and similarly for higher composition orders.

Continuous rules: Also called fluxes, continuous rules are modeled by a set of differential equations

$$\boldsymbol{p}_{t+1} = \partial \boldsymbol{p}_t, \tag{2.4}$$

that describe how $\mathcal{S}^d$ varies in the limit. In such scenarios, Equation 2.4 is typically approximated in the form of Equation 2.3 based on discrete methods (Butcher, 1996) solved using the sampling rate t_s. Summarizing the above, no matter whether the rule is a discrete map or a continuous flux, d different time series, as described by Equation 2.1, can be generated to represent each dimension of the underlying system.

Separately, $\mathcal{S}^d$ can be either *deterministic* or *stochastic*, based on the nature of the generating rule, as follows.

Non-deterministic (stochastic) dynamical systems are used to model unknown influences by means of random or conditional parameters. An example of such system is the two-dimensional random walk

$$\boldsymbol{p}_{t+1} = \sum_{t=0}^{n} Z(\boldsymbol{p}_t), \tag{2.5}$$

where $Z(\boldsymbol{p}_t)$ is a Markov chain (Meyn and Tweedie, 2009) over each state based on the probability density function $P([\boldsymbol{p}_0, \cdots, \boldsymbol{p}_t])$. Among other applications, the random walk, depicted in Figure 2.3(a), is a simplified model that mimics the Brownian motion used in physics to represent the random motion of fluid molecules (Einstein, 1956). Random walks are also present in several other disciplines such as economics, chemistry, computing, and biology. In this scenario, when one analyzes each dimension of Equation 2.5 as a time series, the most common approach is to use statistical tools (Box and Jenkins, 2015) to support modeling and prediction.

Deterministic dynamical systems, on the other hand, have a well-defined generating rule $R(\cdot)$ that produces a single and unique state in the future, given a starting moment. Nonetheless, deterministic systems may present chaotic behaviors. A well-known example is the Lorenz system, designed to model atmospheric data to support weather forecasting (Tucker, 1999) in the form

$$\partial \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \sigma(y-x) \\ x(\rho - z) - y \\ xy - \beta z \end{bmatrix}. \tag{2.6}$$

In this context, parameters σ, β, ρ are adjusted to simulate different environmental conditions. A chaotic system is typically observed using $\rho = 28$, $\sigma = 10$ and $\beta = 8/3$. In this case, approaches like non-linear regression (Bates and Watts, 1988) to forecast observations may lead to poor results when applied directly over dimensions (*i.e.*, time series), as small disturbances tend to evolve to completely different trajectories. Phase-space methods aim to improve modeling by considering phase states and their orbits instead (see Chapter 4).

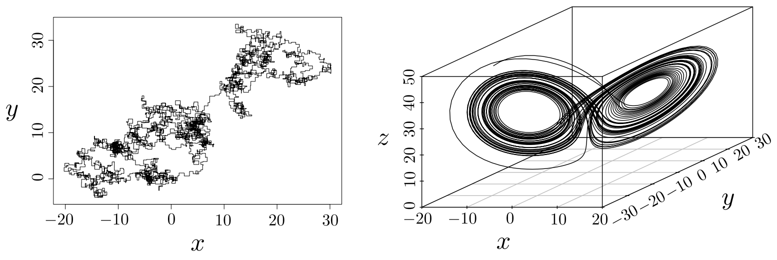


Figure 2.3: Example of dynamical systems. **(a)** Stochastic random walk. **(b)** The Lorenz system was created using $\boldsymbol{p}_0 = \{-13, -14, 47\}$, $t_s = 0.01$, $n = 5001$, $\sigma = 10$, $\beta = 8/3$, and $\rho = 28$. Parameters σ, β, ρ were set with values known to produce a chaotic behavior.

Lastly, it is worth to say that although systems usually present a mixture of both deterministic and stochastic observations, this thesis focused on exploring predominantly deterministic time series[4] due to the typical chaotic/cyclical behavior of natural phenomena (Andrews and Herzberg, 1985).

4 We add artificial noise in most of our experiments when dealing with deterministic generating rules.

2.3.2 *Orbits And Attractors*

Let F be a function that represents either a map or a flux after solving the describing differential equations (Equation 2.4). Given a state $\boldsymbol{p}_t \in \mathcal{S} \subset \mathcal{S}^d$, its k-trajectory or *k-orbit* is the set of states $\{\boldsymbol{p}_t, F(\boldsymbol{p}_t), F^2(\boldsymbol{p}_t), \dots F^k(\boldsymbol{p})\}$ that defines the temporal evolution of $\boldsymbol{p}_t$ to $\boldsymbol{p}_{t+k}$. A state $\boldsymbol{p}_t$ is called *fixed* if $F(\boldsymbol{p}_t) = \boldsymbol{p}_t$, and k-periodic when $F^k(\boldsymbol{p}_t) = \boldsymbol{p}_t$. A fixed state is also *stable* or *unstable* if its nearest states are attracted or repelled to it during the course of their orbits, respectively. Moreover, due to the required notion of distance to measure nearest neighbors, states are assumed to lie in some metric space, such as the Euclidean space $\mathbb{E}^d$, which implies $\mathcal{S}^d \subseteq \mathbb{E}^d$. Thus, the state $\boldsymbol{p}_{t'}$ is a neighbor of $\boldsymbol{p}_t$ if it lies in the interior of the open ball $B(\boldsymbol{p}_t, \varepsilon)$, centered in $\boldsymbol{p}_t$ and with radius ε, in form

$$B(\boldsymbol{p}_t, \varepsilon) = \{\boldsymbol{p}_{t'} \in \mathbb{E}^d : \|\boldsymbol{p}_t - \boldsymbol{p}_{t'}\|_2 < \varepsilon\}, \tag{2.7}$$

where $\|\cdot\|_2$ is the Euclidean norm. In this context, if $\lim_{k\to\infty} F^k(\boldsymbol{p}_t') = \boldsymbol{p}_t$, then $\boldsymbol{p}_t$ is a sink or an *attractor*. On the other hand, if states of the image $F(B(\boldsymbol{p}_t, \varepsilon))$ become more distant from $\boldsymbol{p}_t$ than when they were in $B(\boldsymbol{p}_t, \varepsilon)$, *i.e.*, they are repelled from $\boldsymbol{p}_t$ along their orbits, then such point is called a *source* state. The *basin of attraction* is the region formed by the smallest, but sufficient radius ε, such as neighbors of $\boldsymbol{p}_t$ are attracted to it. Moreover, fixed points can behave differently across dimensions, such that saddles may be formed (for $\mathcal{S}^{d>1}$), as illustrated in Figure 2.4.

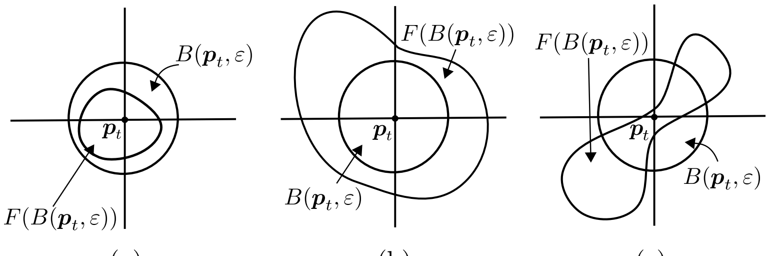

$B(\boldsymbol{p}_t, \varepsilon)$
$\boldsymbol{p}_t$
$F(B(\boldsymbol{p}_t, \varepsilon))$
(a)
$F(B(\boldsymbol{p}_t, \varepsilon))$
$\boldsymbol{p}_t$
$B(\boldsymbol{p}_t, \varepsilon)$
(b)
$F(B(\boldsymbol{p}_t, \varepsilon))$
$\boldsymbol{p}_t$
$B(\boldsymbol{p}_t, \varepsilon)$
(c)

Figure 2.4: Different types of attractors. From left to right: $\boldsymbol{p}_t$ is **(a)** an attractor point or sink; **(b)** a repelling point or source; or **(c)** a saddle point. Adapted from Alligood et al. (1996).

Based on the above concepts, one may realize that it is not uncommon to find multiple and different types of attractors that, together, define the dynamics of a system. Such orbits sometimes evolve into nonlinear trajectories that are useful to visualize (for low dimensions) and measure important features on the space (Section 2.6). For instance, two-dimensional attractors can be depicted by cobweb plots, while isolated circuits are called limit circles (Alligood et al., 1996). Moreover, periodicities of high-dimensional systems may form d-dimensional tori. On the other hand, more complex structures like fractals (Section 2.6.1) and manifolds (Section 2.4) are known as *strange attractors* (Mandelbrot, 1977; Alligood et al., 1996; Lee, 2003), as it is the case of the famous Lorenz system (Equation 2.6). In the latter case, any initial point $\boldsymbol{p}_0$ will, eventually, converge and be bounded to the trajectories of the attractor, as illustrated in Figure 2.5.

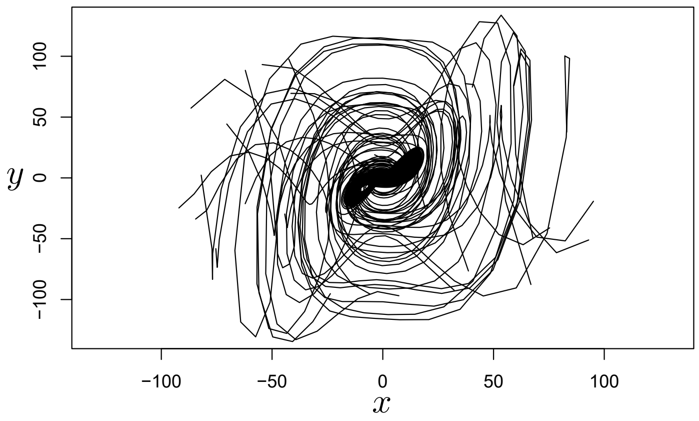


Figure 2.5: Example of different orbits of the Lorenz system using 20 random initial points $\boldsymbol{p}_0$. As it is noticed, all trajectories eventually converge to the attractor, never leaving afterwards. In order to simplify the visualization, only a two-dimensional system is shown.

Despite different types of generating rules can lead to previously mentioned attractors, strange attractors are typically found in chaotic systems due to their sensitiveness to the initial conditions. In such scenarios, two almost identical states $\|\boldsymbol{p}_{t'} - \boldsymbol{p}_t\| \leq \varepsilon \to 0^+$ tend to evolve to completely different orbits (even though remaining restricted to the form of the attractor) as time elapses, eventually getting close to each other again after a certain number of iterations. Such factor makes those systems especially hard to predict, as minimum errors/fluctuations in data sampling and modeling (even in the limited capacity of float number representation) may be enough to change orbit trajectories.

2.3.3 *Phase Space*

A deterministic generating rule $R(\cdot)$ maps the state $\boldsymbol{p}_t \in \mathcal{S} \subset \mathcal{S}^d$ to, ideally, a single state $\boldsymbol{p}_{t+1}$ in the future. Conversely, unpredictable factors such as random processes can disrupt such mapping for stochastic systems. Therefore, for deterministic data, the analysis of this rule offers, as main advantage, a more consistent approach to: i) identify patterns, cycles and trends; ii) forecast observations; and iii) correlate systems. The quality of such analyses directly depends on the number of states in $\mathcal{S}$ and how they are rearranged in the space. For the case when $\mathcal{S}$ is characterized by a sufficient set of states representing all possible dynamics of $\mathcal{S}^d$, then $\mathcal{S}$ is called the *phase space* of $\mathcal{S}^d$. When such a phase space is extracted from the time series, the variable *time* has no longer influence on the system (Pagliosa and de Mello, 2017). Therefore, the phase space can be used to interpret how the analyzed phenomenon behaves along *any given period of time*, thereby considerably simplifying the analysis and, in particular, the prediction. Note that if $\mathcal{S}$ is a phase space, then it may be represented by a finite set of states with potentially lower dimension than d, as the dynamics of the system may converge to its attractor.

Although Figure 2.1(b) already exemplifies the phase space of the Lorenz system, let us reinforce this concept using another, simpler, example given by the nonlinear motion of the pendulum

$$\frac{\partial^2\theta}{\partial t^2} + \omega \sin\theta = 0, \tag{2.8}$$

where θ defines the angle between the pendulum rod and the vertical line, $\omega = \frac{g}{L}$ is the motion frequency, g is the gravitational acceleration, and L gives the pendulum length. Such a system can be expressed in terms of the angle $x = \theta$ and the angular velocity $y = \frac{\partial\theta}{\partial t}$, as the other parameters remain constant. Thus, one can use these two variables to reconstruct the phase space according to the relation

$$\frac{\partial}{\partial t}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} y \\ -\omega \sin x \end{bmatrix}, \tag{2.9}$$

which represents all possible combinations between angle and angular velocity that a pendulum may have.

Despite the visualization of phase states (in the form of two/three-dimensional trajectories) is usually enough to analyze patterns and behaviors for well-defined structures, a vector field (the gradients of Equation 2.9) represents a more intuitive visual depiction to track the *dynamics* of the system when analyzing dense spaces, as illustrated in Figure 2.6. From this figure, it is observed the existence of sink and source states that can be useful to predict cycles and future observations (Boccaletti and Bragard, 2008). Additionally, more refined approaches such as feature detection methods (Post et al., 2003), geometric methods (McLoughlin et al., 2010), and texture-based methods (Laramee et al., 2004) can be applied to highlight deeper insights.

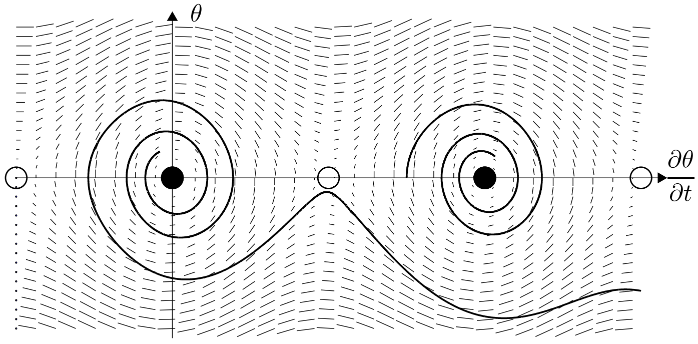


Figure 2.6: Vector field of the phase space of the simple pendulum, given by Equation 2.9. Small dashes represent the gradients of states. For simplicity, the arrows were removed, but the overall orientation of states is depicted by solid lines. Using this picture, one can interpret, for instance, that with greater velocities the pendulum has enough energy to rotate, while preserving an oscillating pattern at lower speeds, eventually converging to an equilibrium point when its velocity reaches zero.

Finally, it is worth to say that phase-space analysis is also possible when the generating rule is unknown, as shown in more detail in Section 2.5.

2.4 IMMERSION AND EMBEDDING

This section presents the basic concepts regarding topology and diffeomorphism, which later inspired Takens to propose his embedding theorem to reconstruct the phase space from univariate time series (Takens, 1981).

Topological space: A topological space is a set of open sets that, following axioms based on set theory, characterizes one of the most general structures of mathematical spaces. As an open set is an abstract concept that generalizes the notion of an open interval in $\mathbb{R}$, a topological space is mainly defined by a set of points and their relation with their neighborhoods. Thus, other spaces such as the metric and normed spaces are specializations of the topological space, since additional constraints and measures are defined (Mendelson, 1975). Further, more complex spaces where the concept of distance and direction may be needed in order to perform deeper analyses. Normally, this space is the Euclidean space $\mathbb{E}^d$ (or equivalently the real space $\mathbb{R}^d$), where the notion of neighborhood is given as in Equation 2.7.

Manifolds: The correspondence between topological and Euclidean spaces may be given through manifolds. More precisely, a d-manifold $\mathcal{M}$ is a topological space such that each of its open sets $p \in \mathcal{M}$ can be mapped to a point $\boldsymbol{p} \in \mathbb{E}^d$ and vice-versa, *i.e.*, $p \approx \boldsymbol{p}$, without loosing any topological property (neighborhood relationships). Under those circumstances, a d-manifold is said to be locally homeomorphic to $\mathbb{E}^d$ (Mendelson, 1975; Lee, 2003; LaValle, 2006). By this definition, examples of unidimensional manifolds consist of open intervals in $\mathbb{E}$ and circles in $\mathbb{E}^2$, respectively, while surfaces such as planes, spheres, and tori in $\mathbb{E}^3$ are representations of two-dimensional manifolds.

Differentiable manifolds: A differentiable manifold is a manifold locally defined by a set of C^k differential equations that provide additional information to the abstract topological space $\mathcal{M}$. With these functions, one can unambiguously define directional derivatives and tangent spaces to perform infinitesimal calculus and deform manifolds. Some of those deformations, which are in the form $F : \mathcal{M} \to \mathcal{N}$, receive special attention depending on the properties they preserve. For instance, if $T_{\boldsymbol{p}}\mathcal{M}$ is the tangent space (Lee, 2003) on the point $\boldsymbol{p}$ in the manifold $\mathcal{M}$, then an *immersion* is a function whose derivative $\partial_{\boldsymbol{p}}F$ (partial of F with respect to $\boldsymbol{p}$) is everywhere injective

$$\partial_{\boldsymbol{p}}F : T_{\boldsymbol{p}}\mathcal{M} \to T_{F(\boldsymbol{p})}\mathcal{N}, \tag{2.10}$$

which guarantees the resulting image ($\mathcal{N}$) has well-defined derivatives in all its domain ($\mathcal{M}$). However, the image of an immersion is not necessarily a manifold. Figure 2.7 illustrates the possible scenarios involving an immersion. An *embedding*, on the other hand, is a transformation that, besides being an immersion, is an injective function itself that also creates a diffeomorphism[5] between $\mathcal{M}$ and $\mathcal{N}$. Therefore, in contrast to immersions, the image of an embedding is always a manifold, as illustrated in Figure 2.8. Moreover, if the manifold is *compact*[6] (as is the case of most attractors), then every injective immersion is an embedding.

The motivation behind these deformations is to transform the topological properties of a manifold into a more intuitive, and easier to process, representation such as a surface in the Euclidean space. One of the most famous examples to illustrate this concept

5 A diffeomorphism is an invertible function that maps one differentiable manifold to another, such that both the function and its inverse are smooth.

6 A manifold is compact if it is finite and has limit points.

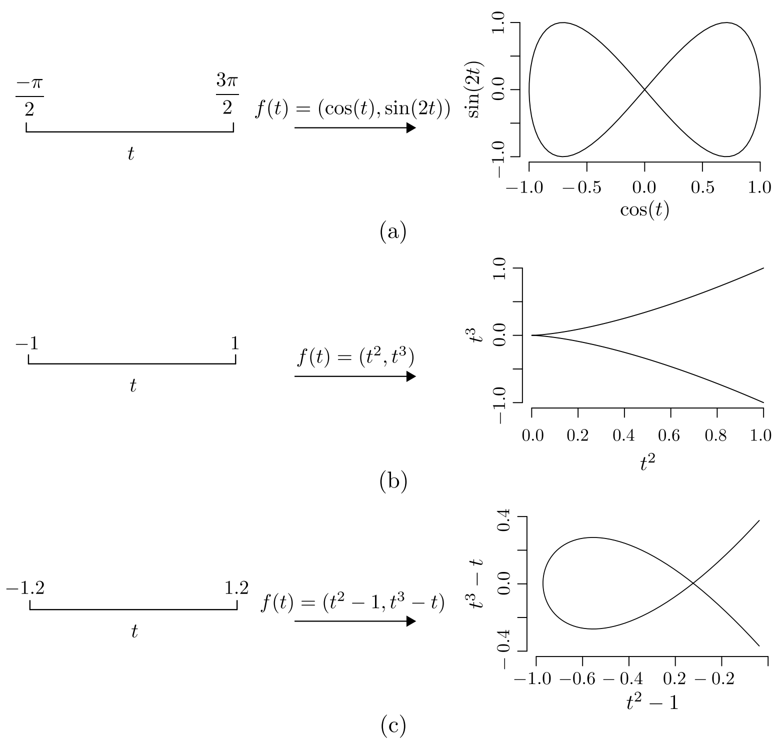


Figure 2.7: Example of immersions. **(a)** The eight-shaped closed curve is an immersion of the open set $(\frac{-\pi}{2}, \frac{3\pi}{2})$ into $\mathbb{E}^2$. **(b)** The cuspidal cubic (middle) is not an immersion, as the partial of $f(t)$ is not injective in 0. **(c)** The nodal cubic (bottom) is an immersion: $f'(t) = (2t, 3t^2 - 1) = (0, 0)$ has no solution in t. All images are non manifolds. Adapted from Tu (2010).

is the immersion of the Klein bottle, a 2-manifold whose topology can be described by an identification (LaValle, 2006) in the form of a square. In this representation, points near edges should remain together so that the orientation of similar arrows match. Thus, despite the topological space has enough information to describe how points behave, the immersion of the Klein bottle into $\mathbb{E}^3$ (Figure 2.9) turns it much easier to understand and study, even if the resulted image is not a manifold. Similarly, one can embed the Klein bottle into $\mathbb{E}^4$ to remove the observed self-intersections.

Finding a space in which a manifold can be embedded is not a trivial task in most cases. In this context, Whitney (1936) proposed a theorem saying that $\mathbb{E}^{2d+1}$ is a sufficient space to embed a d-manifold, since no two points from a d-dimensional manifold could be mapped to the same point in the $(2d + 1)$-dimensional space (Ghomi and Greene, 2011; Forstnerič, 2011). It is worth to re-

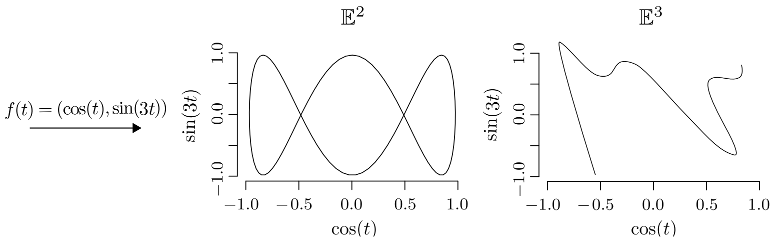


Figure 2.8: The function is not an embedding in $\mathbb{R}^2$, but it is in $\mathbb{R}^3$. In this example, $t \in (\frac{-\pi}{2}, \frac{3\pi}{2})$. Adapted from Tu (2010).

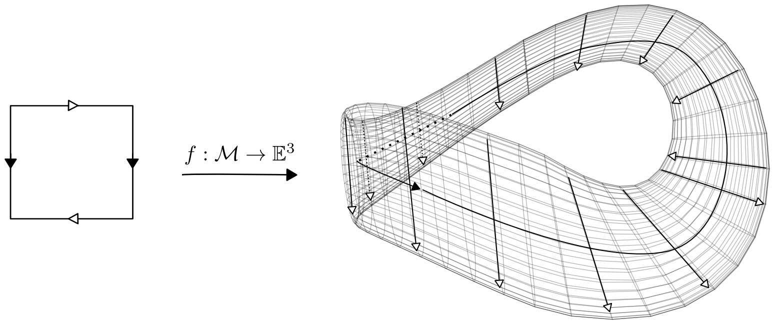


Figure 2.9: Identification of the topological space (left) and the resulting image of the immersion of the Klein bottle into $\mathbb{E}^3$ (right). Points in this topology should be close to each other such that the orientation of similar arrows are equal.

inforce, however, that this theorem elaborates a sufficient, but not necessary condition, such that lower dimensions may be enough to embed a manifold, as it is the case of the Klein bottle. According to Whitney's theorem, such 2-dimensional manifold can be embedded in $\mathbb{E}^5$, but $\mathbb{E}^4$ is already enough.

Extending this study, Takens (1981) proposed his own embedding theorem, described next. Let $\mathcal{M}$ be a compact manifold of dimension d[7]. For pairs (φ, y), where $\varphi : \mathcal{M} \to \mathcal{M}$ is a diffeomorphism and $y : \mathcal{M} \to \mathbb{R}$ is a smooth function, it is a generic property that the map $\Phi : \mathcal{M} \to \mathbb{E}^{2d+1}$, in the form

$$\Phi_{(\varphi,y)}(\boldsymbol{p}) = (y(\boldsymbol{p}), y \circ \varphi(\boldsymbol{p}), \cdots, y \circ \varphi^{2d}(\boldsymbol{p})), \tag{2.11}$$

is an embedding. In other words, the main contribution of Takens' theorrem was to show that a single quantity of the manifold $\mathcal{M}$ is enough to embed it in $\mathbb{E}^{2d+1}$. However, like Whitney, Takens' theorem elaborates a *sufficient*, but not a *necessary* condition. As matter of fact, the space $\mathbb{E}^{2d+1}$ is usually an overestimation, and

7 The dimension here is the Euclidean space in which the manifold lies, not the dimension it is homeomorphic to.

finding a lower, simpler **embedding dimension**, from now on referred to as m, is desirable in order to decrease the computational costs involved in modeling and prediction, especially when dealing with large volumes of continuously collected data, also known as data streams (Muthukrishnan, 2005). For instance, a sufficient space to embed the 2-manifold, 3-dimensional Lorenz system, according to Takens is $\mathbb{E}^7$, but $\mathbb{E}^{m=3}$ is already enough to unfold the Lorenz attractor dynamics.

2.5 RECONSTRUCTING PHASE SPACES

The previous sections have introduced the mathematical support on dynamical systems and immersions. Next, this section combines those concepts and describes how a time series can be embedded in practice.

As previously discussed, the process of finding a finite set $\mathcal{S} \subset \mathcal{S}^d$ resembling the dynamics of $\mathcal{S}^d$ is known as *unfolding* or *reconstructing* the phase space of $\mathcal{S}^d$. If one knows $R(\cdot)$, the reconstruction becomes quite straightforward after generating enough states of the respective map or discretized flux. However, this process becomes more difficult when the generating rule is *unknown*, as it is the case of real-world data sampled from some arbitrary time-dependent phenomenon. Additionally, an even more problematic issue is the lack of information on the data: humans tend to model phenomena in terms of the variables they observe and know, which usually tend to be an insufficient and inaccurate representation of the underlying phenomenon. Separately, data measurements may in practice be corrupted or have missing values, forcing the analyst to disregard them. Summarizing, one may face several scenarios in which only a small number of dimensions is available for analysis. In the limit, we consider the case where just a single dimension $i \in [1, d]$ of $\mathcal{S}^d$ is considered[8].

A dynamical system $\mathcal{S}^d$, especially when modeling natural phenomena, usually presents recurrent patterns and observations. In addition, it is expected that variables composing such a system do not only impact themselves, but directly or indirectly affect other variables along time. Such correlation can indeed be noticed in the Lorenz system (Equation 2.6) and in the simple pendulum map (Equation 2.9). Further, if one represents the ith ($i \in [1, d]$) component of all phase states as the time series T_i, it is reasonable to expect that such observations have, even that implicitly, informa-

8 There exist methods that analyze the impact of using more than one time series to reconstruct the phase space (Cao et al., 1998). However, this matter is out of the scope of this thesis.

tion related to other variables of $R(\cdot)$[9]. In order to take advantage of this relation, one can rely on Takens' embedding theorem (Takens, 1981) to embed a d-dimensional manifold $\mathcal{M}$ into $\mathbb{E}^{2d+1}$ according to Equation 2.11, where $y(\cdot)$ is interpreted as a direct map to access the observations of T_i. Thus, according to Takens, a time series T_i can be embedded into a space that is diffeomorphic to $\mathcal{S}^d$ or, more precisely, to its phase space $\mathcal{S}$. In this situation, the phase space will be represented by the $N_i \times (2d+1)$ *trajectory matrix*, denoted from now on to as $\mathbf{\Phi}_i$, in form

$$\mathbf{\Phi}_i = \begin{bmatrix} y(\boldsymbol{p}_0) & y \circ \varphi(\boldsymbol{p}_0) & y \circ \varphi^2(\boldsymbol{p}_0) & \cdots & y \circ \varphi^{2d}(\boldsymbol{p}_0) \\ y(\boldsymbol{p}_1) & y \circ \varphi(\boldsymbol{p}_1) & y \circ \varphi^2(\boldsymbol{p}_1) & \cdots & y \circ \varphi^{2d}(\boldsymbol{p}_1) \\ y(\boldsymbol{p}_2) & y \circ \varphi(\boldsymbol{p}_2) & y \circ \varphi^2(\boldsymbol{p}_2) & \cdots & y \circ \varphi^{2d}(\boldsymbol{p}_2) \\ y(\boldsymbol{p}_3) & y \circ \varphi(\boldsymbol{p}_3) & y \circ \varphi^2(\boldsymbol{p}_3) & \cdots & y \circ \varphi^{2d}(\boldsymbol{p}_3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y(\boldsymbol{p}_{N_i-1}) & y \circ \varphi(\boldsymbol{p}_{N_i-1}) & y \circ \varphi^2(\boldsymbol{p}_{N_i-1}) & \cdots & y \circ \varphi^{2d}(\boldsymbol{p}_{N_i-1})) \end{bmatrix}, \tag{2.12}$$

so that, mathematically

$$T_i \to \mathbf{\Phi}_i \approx \mathcal{S} \subset \mathcal{S}^d. \tag{2.13}$$

In this context, Takens also proposed a convenient diffeomorphic function φ in the form

$$\varphi^\tau(\boldsymbol{p}) : \boldsymbol{p} \to \tau\boldsymbol{p}, \quad \tau \in \mathbb{I}^+, \tag{2.14}$$

later commonly known as *the method of delays* due to its time displacement characteristics. Assuming the manifold is discretized as a non-uniform grid (see Figure 2.10), such a direction could be, for instance, the dimension i, so that φ shifts the point $\boldsymbol{p} \in \mathcal{M}$ τ units to the right. Finally, the function $y(\cdot)$ merely maps the component $p_{t,i}$ from the phase state $\boldsymbol{p}_t$ to $x(t) \in T_i$, as illustrated in Figure 2.11.

Therefore, given the time series $T_i = \{x(0), \cdots, x(n_i - 1)\}$, the phase space $\mathbf{\Phi}_i$ with N_i states can be reconstructed according to Equation 2.12. More precisely, the method of delays reconstructs each phase space as

$$\boldsymbol{\phi}_i(t) = [x(t), x(t+\tau), x(t+2\tau), \cdots, x(t+2d\tau)], \tag{2.15}$$

where τ is the **time delay**, as defined in Section 2.2. Moreover, it is worth to note that, because $\mathbf{\Phi}_i \approx \mathcal{S}$, *i.e.*, the reconstructed

9 As consequence, it is worth to mention that the quality of the reconstructed phase space depends on the amount of influence T_i brings about other variables.

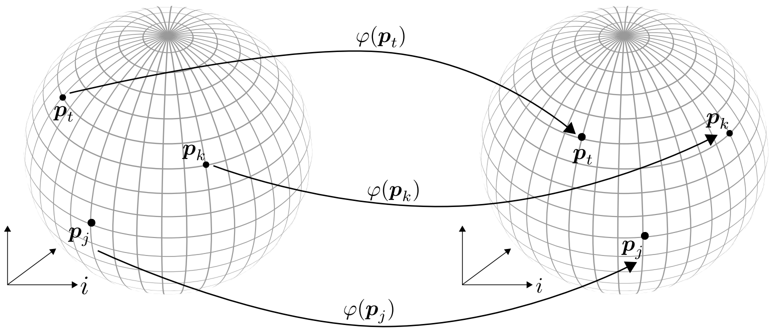


Figure 2.10: Example of diffeomorphism between manifolds. The rotation of a manifold is a transformation whose image is diffeomorphic to the original manifold (sphere). The states $\boldsymbol{p}_t$, $\boldsymbol{p}_k$, and $\boldsymbol{p}_j$ are mapped to $\tau = 4$ units in the direction of the rotation (in this case, to the right). The sphere has been discretized to facilitate understanding.

phase space is diffeomorphic to $\mathcal{S}$, it is known that Equation 2.15 locally preserves neighboring properties of phase states. According to our notation, this relation is expressed as

$$\boldsymbol{\Phi}_i(t) \approx \boldsymbol{p}_t. \tag{2.16}$$

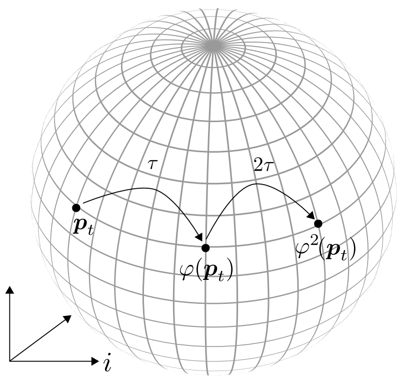
Other strategies to reconstruct the phase space are either modifications of Equation 2.15 (Broomhead and King, 1986) or based on different diffeomorphism functions. For instance, Packard et al. (1980) proposed the method of derivative coordinates, where each row in Equation 2.12 is defined as

$$\varphi^\tau(\boldsymbol{p}) : \boldsymbol{p} \rightarrow \frac{\partial^\tau F(\boldsymbol{p})}{\partial i^\tau}, \quad \tau \in \mathbb{I}^+. \tag{2.17}$$

This method creates phase states similarly to the method of delays but uses infinitesimal time delays, which is impractical when the generating rule is unknown. Nevertheless, one can assume the time series is structured as $T_i = \{x(-h_i), \cdots, x(0), \cdots, x(h_i)\}$, where $h_i = (n_i - 1)/2$, and approximate Equation 2.17 by finite differences (Canuto and Tabacco, 2008) as

$$\boldsymbol{\phi}_i(t) = \left[x(t), \frac{x(t+\tau) - x(t)}{\tau}, \frac{x(t+\tau) - 2x(t) + x(t-\tau)}{\tau^2}, \cdots \right]. \tag{2.18}$$

Figure 2.12 illustrates both methods for the Lorenz system. While there is no clear evidence about which of these methods is the most appropriate, Ravindra and Hagedorn (1998) elaborate



$$\begin{aligned}
\boldsymbol{p}_t &= [p_{t,1}, p_{t,2}, p_{t,3}] \\
\varphi(\boldsymbol{p}_t) &= [p_{t+\tau,1}, p_{t+\tau,2}, p_{t+\tau,3}] \\
\varphi^2(\boldsymbol{p}_t) &= [p_{t+2\tau,1}, p_{t+2\tau,2}, p_{t+2\tau,3}]
\end{aligned}$$

$$\begin{aligned}
y \circ \boldsymbol{p}_t &= x(t) \\
y \circ \varphi(\boldsymbol{p}_t) &= x(t+\tau) \\
y \circ \varphi^2(\boldsymbol{p}_t) &= x(t+2\tau)
\end{aligned}$$

Figure 2.11: Diffeomorphism according to the method of delays. The method of delays allows the reconstruction of the phase space using a single dimension i, represented as the time series T_i. In the proposed coordinate system, i represents the first dimension/component.

that the method of delays produces better results when analyzing nonlinear time series. In fact, the method of delays is the most used approach in the literature (Stark, 1999; Yap and Rozell, 2011; Yap et al., 2014). It is worth to say that the method of delays, as originally proposed (Equation 2.15), assumes an uniform τ. Nonetheless, there are articles that investigate the usage of multiple time delays (Breedon and Packard, 1992; Manabe and Chakraborty, 2007).

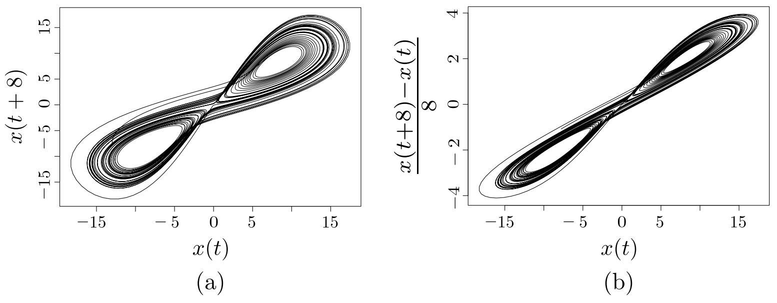


Figure 2.12: Despite the different results, the reconstructed phase space using either the method of delays **(a)** and derivatives **(b)** preserve topological properties and, most importantly, the dynamics of the original Lorenz system (Figure 2.3(b)). The reconstruction was performed using $m = 3$ and $\tau = 8$ in both methods. For simplicity, only the first two dimensions are visualized.

2.6 PHASE SPACE FEATURES

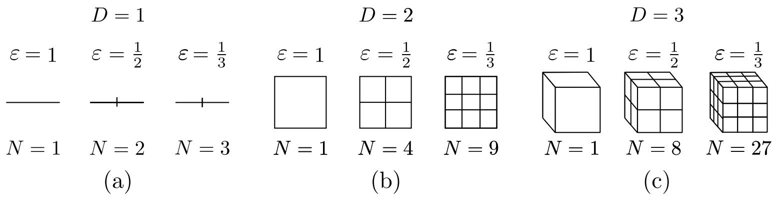
Based on reconstructed phase space, several methods were proposed to identify and predict chaotic time series (Farmer and Sidorowich, 1987; Andrievskii and Fradkov, 2003; Boccaletti and Bragard, 2008; de Mello and Yang, 2009). Next, concepts such as correlation dimensions and Lyapunov exponents, important to these two types of analysis, are described.

2.6.1 *Fractal Dimension*

Following the Gestalt principles (Chang et al., 2007), a geometrical object (shape) can be described in term of its patterns and how these are arranged in space. Further, a pattern can be defined as a feature that repetitively occurs at ε spatial units of measure. For instance, patterns can be defined in function of length, area, or volume, in one, two, and three-dimensional spaces, respectively. Therefore, if a D-dimensional object presents N patterns, as illustrated in Figure 2.13, one can notice the relation of proportionality ($\propto$)

$$N \propto \varepsilon^D \therefore D \approx \log N / \log \varepsilon. \tag{2.19}$$

With the above, a fractal can be defined as an object whose



$D = 1$
$D = 2$
$D = 3$
$\varepsilon = 1$
$\varepsilon = \frac{1}{2}$
$\varepsilon = \frac{1}{3}$
$\varepsilon = 1$
$\varepsilon = \frac{1}{2}$
$\varepsilon = \frac{1}{3}$
$\varepsilon = 1$
$\varepsilon = \frac{1}{2}$
$\varepsilon = \frac{1}{3}$
$N = 1$
$N = 2$
$N = 3$
$N = 1$
$N = 4$
$N = 9$
$N = 1$
$N = 8$
$N = 27$
(a)
(b)
(c)

Figure 2.13: Relation between dimension and geometry for one **(a)**, two **(b)** and **(c)** three-dimensional objects. The number of patterns N, the scaling factor ε and the dimension of the object D are correlated by Equation 2.19.

patterns are given in function of the object itself (Mandelbrot, 1977). If patterns are perfect replicas occurring at every scale ε, the fractal follows a self-similar pattern, as it is the case of the Koch snowflake (Figure 2.14). Differently from other shapes, fractals usually do not have a uniform relation between ε and N, so that D, also called the *fractal dimension*, is often a real number. Despite not being unique, this quantity turns to be an important space descriptor, as it abstracts the complexity of a shape.

Use for dynamical systems: In the scope of dynamical systems, one can compute the fractal dimension D based on the orbits of the

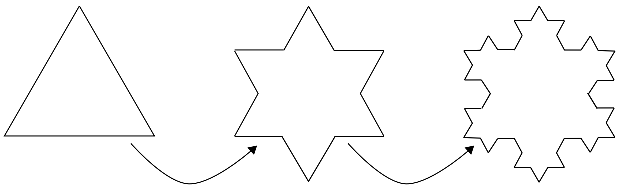


Figure 2.14: The Koch snowflake is a fractal that replicates itself 4 times per iteration (from left to right: 1, 4, 16 triangles). Each pattern occurs after dividing each edge in 3 equally-length pieces. Thus, $N = 4$, $\varepsilon = 1/3$ and $D = 1.2619$.

phase states to quantify the structure of $\mathcal{S}^d$. Among other applications, the fractal dimension becomes useful to distinguish deterministic/chaotic from stochastic systems, as explained next. Suppose we have a dynamical system with a d'-dimensional attractor. The fractal dimension of such an attractor is less or equal to d', *i.e.*, $D \leq d'$ (due to the finite data, noise, and no self-similar patterns). If the attractor is embedded or lies in some higher dimension, the fractal dimension stays the same. Therefore, the fractal dimension of a reconstructed phase space is *invariant to the embedding dimension m*. If the generating rule is guided by a stochastic process, *e.g.*, a Normal distribution, the space has greater probability to be equally filled by states, leading the fractal dimension to always be equal to the embedding dimension. In summary, the following relation is noticed:

1. if the fractal dimension D does not vary with the embedding dimension m, then m is greater than the dimension of the attractor, and T_i has more probability to be deterministic (chaotic or not);

2. if the fractal dimension D is equal to the embedding dimension m for different values of m, either the space does not unfold the dynamics of the attractor or T_i is stochastic (non-deterministic).

Unfortunately, the fractal dimension D cannot be exactly computed in practice. Because fractals may change patterns as $\varepsilon \to 0$, one would need an infinity amount of data ($N \to \infty$) to find the true value of D. Thus, several approaches were proposed to *estimate* the fractal dimension of an object. These include the box-counting dimension D_0; information dimension D_1; *correlation dimension* D_2 and D_L *Lyapunov dimension* D_L (Clark, 1990; Theiler, 1990; Ding et al., 1993; Alligood et al., 1996).

While all these dimensions have relative advantages, the correlation dimension D_2 is one of the most used methods in the literature (Kantz and Schreiber, 2004). Nonetheless, there is evidence that D_2 and D_L are numerically close to each other and less prone to be affected by noise for small datasets (Otani and Jones, 1997). As consequence, we next detail the D_2 method, followed by the D_L method (Section 2.6.3).

2.6.2 *Correlation Dimension*

The correlation dimension of the system $\mathcal{S}^d$ is based on the correlation integral

$$\hat{C}(\varepsilon) = \int\int \theta(\varepsilon - \|\boldsymbol{p}_t - \boldsymbol{p}_{t'}\|_2) dH(\boldsymbol{p}_t) dH(\boldsymbol{p}_{t'}), \tag{2.20}$$

where ε is an open-ball radius, $\|\cdot\|_2$ is the Euclidean norm, and $H(\cdot)$ is the invariant distribution of $\mathcal{S}^d$. As we have $\boldsymbol{\Phi}_i \approx \mathcal{S} \in \mathcal{S}^d$, the correlation sum, proposed by Grassberger and Procaccia (1983), aims to estimate the correlation integral for the discrete set of N phase states, embedded using parameters (m, τ), as

$$C(m, \varepsilon) = \sum_{t<t'}^{N-1} \theta(\varepsilon - \|\boldsymbol{p}_t - \boldsymbol{p}_{t'}\|_2) \frac{2}{N * (N-1)}, \tag{2.21}$$

where θ is the Heaviside step function

$$\theta(x) = \begin{cases} 0, & \text{if } x < 0, \\ 1, & \text{otherwise.} \end{cases} \tag{2.22}$$

The correlation dimension D_2 is then estimated from the fractal integral (Equation 2.19) as

$$\lim_{\varepsilon \to 0} \lim_{N \to \infty} C(m, \varepsilon) \approx \varepsilon^{D_2} \therefore D_2 \approx \lim_{\varepsilon \to 0} \lim_{N \to \infty} \frac{\log C(m, \varepsilon)}{\log \varepsilon}. \tag{2.23}$$

However, as the phase space is represented by a finite set of states, one cannot infinitely decrease the radius ε, also referred to as scaling factor in this case. Thus, the most common approach is to assume the fractal to have self-similar patterns and extrapolate it as $\varepsilon \to 0$. Among the alternatives, this estimation is obtained by computing the slope of the regression line that best fits the points in the plot $\log C(m, \varepsilon)$ versus $\log \varepsilon$, as shown in Figure 2.15. In addition, as the correlation dimension should not vary for chaotic attractors (as it is the case for the Lorenz system) when $m \geq D$, a better estimate can be achieved by taking the average slope for multiple embeddings. For instance, this approach estimates $D_2 = 2.04$ for the Lorenz system, where the true fractal dimension is known to be 2.05 (Grassberger and Procaccia, 1983). Despite the good result for this particular case, estimating the correlation dimension is not trivial for general systems, as the algorithm is sensitive to several parameters including the number of observations N, the scaling factor ε, and the interval from where the slope is computed.

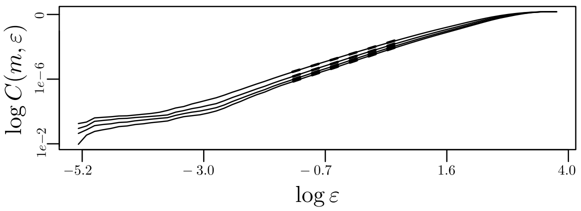


Figure 2.15: The correlation dimension of the Lorenz system ($\tau = 8$) can be estimated as the average slope of the log-log plot between the correlation sum versus the scaling factor ε, over different embedding dimensions (from top to bottom: $m = [3, 6]$). The interval used to compute the slope is bounded by bolder lines.

2.6.3 *Lyapunov Exponents*

Let $\boldsymbol{J}_t$ be the $d \times d$-Jacobian matrix in the form

$$\boldsymbol{J}_t = \begin{bmatrix} \dfrac{\partial f_1}{\partial p_{t,1}} & \cdots & \dfrac{\partial f_1}{\partial p_{t,d}} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial f_d}{\partial p_{t,1}} & \cdots & \dfrac{\partial f_d}{\partial p_{t,d}} \end{bmatrix}, \tag{2.24}$$

where $F = \{f_1, f_2, \cdots, f_d\}$ is a d-dimensional function applied to the point $\boldsymbol{p}_t$. Among several applications, the Jacobian matrix can be used as a transformation to linearly approximate states through Taylor expansion (Canuto and Tabacco, 2008) as

$$F(\boldsymbol{p}_t + \boldsymbol{h}) \approx F(\boldsymbol{p}_t) + \boldsymbol{J}_t \boldsymbol{h}, \tag{2.25}$$

where $\boldsymbol{h} \to \boldsymbol{0}$ is an infinitesimal displacement vector. Moreover, if $(\lambda_i, \boldsymbol{v}_i)$, $i \in [1, d]$ are eigenpairs of the Jacobian $\boldsymbol{J}_t$, then the eigenvalues λ_i are the coefficients of the linear combination $\boldsymbol{J}_t = \lambda_1 \boldsymbol{v}_1 + \cdots + \lambda_2 \boldsymbol{v}_2 \cdots + \lambda_d \boldsymbol{v}_d$ that quantifies the rate of variation $\boldsymbol{p}_t$ has in each dimension. In other words, the Jacobian can be used as an approximation of the derivatives of F.

Use for dynamical systems: In the scope of dynamical systems, F represents the map or discretized flux of the generating rule

$R(\cdot)$ describing the dynamics of $\mathcal{S}^d$, and $\boldsymbol{p}_t$ is a phase state on that system. In that sense, if $\boldsymbol{p}_t$ and another state are separated $\boldsymbol{h}$ units of measurement after k iterations of their orbits, so that $\boldsymbol{\delta}(k) = \|[\delta(k)_1, \cdots, \delta(k)_d]\|_2 = \|\boldsymbol{h}\|_2$ is the norm of such distance (Figure 2.16), the divergent rate for each dimension $i \in [1, d]$ can be described by the corresponding eigenvalue λ_i of $\boldsymbol{J}_t^k$, defined from the chain rule (Alligood et al., 1996) as

$$\boldsymbol{J}_t^k = \boldsymbol{J}_{t+k-1}\boldsymbol{J}_{t+k-2}\cdots\boldsymbol{J}_t. \tag{2.26}$$

For the case of chaotic time series, however, close trajectories are known to exponentially diverge from each other after k iterations (Figure 2.16), such that

$$\lim_{k\to\infty} \delta(k)_i \approx \delta(0)_i \exp(\lambda_i k), \tag{2.27}$$

is the divergence rate for dimension i. As consequence, the average variation between states along the dimension i, per iteration, is roughly approximated as $\lambda_i^{1/k}$.

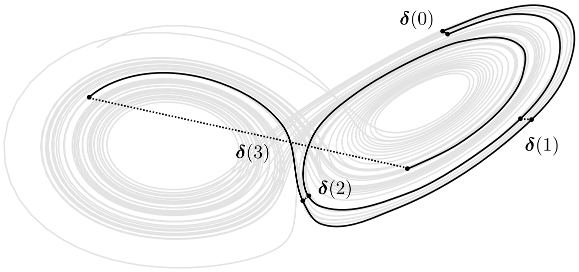


Figure 2.16: Divergence of initially close orbits in chaotic systems. State trajectories tend to diverge exponentially in chaotic systems, so that two states, initially close, evolve to completely different orbits after k iterations. In this example, $k = 3$.

The relation among chaotic trajectories has motivated the definition of the ith local Lyapunov exponent as

$$L_i = \lim_{k\to\infty} \frac{1}{k} \log \lambda_i, \tag{2.28}$$

in attempt to quantify the amount (and in which direction) the dynamical system is varying. Kaplan and Yorke (1979) proposed to rank local Lyapunov exponents, in the form of $L_1 \geq L_2 \geq \cdots L_d$, and use them to estimate the Lyapunov dimension D_L as

$$D_L = j + \frac{\sum_{i=1}^{j} \lambda_i}{\lambda_j + 1}, \tag{2.29}$$

where j is the largest integer so that $\lambda_1 + \lambda_2 + \cdots + \lambda_j \geq 0$.

Besides estimating the fractal dimension, Lyapunov exponents are used to measure a system sensitivity to initial conditions, *i.e.*, chaos (Alligood et al., 1996). More precisely, if one computes the (global) Lyapunov exponent as the most representative dispersion of $\boldsymbol{J}_t^k$ (some authors use the trace of $\boldsymbol{J}_t^k$, *i.e.*, $\sum_{i=1}^{d} \lambda_i$, instead), we define

$$\lambda = \max(L_i), \tag{2.30}$$

and then the system either tends to converge to attractor points ($\lambda < 0$), be conservative ($\lambda = 0$), or present unstable and chaotic behavior ($\lambda > 0$) after k iterations. In this context, λ is also used to define the *prediction horizon*, *i.e.*, the maximum number of future observations one can forecast within a reliable margin of confidence, as

$$H = -\frac{\log E}{\lambda}, \tag{2.31}$$

given a training error E. To exemplify this concept, assume that an algorithm models a time series achieving training error $E = 0.001$ for a $\lambda = 0.692$ system. Despite the small training error, by Equation 2.31, only $H = \frac{-\log 0.001}{0.692} = 9.98 \approx 10$ observations can be predicted with enough confidence. Without knowing that, one may wrongly assumed that the algorithm overfitted, being uncapable of generalization (Vapnik, 1998; Luxburg and Schölkopf, 2011). Actually, the algorithm has likely *learned* the data well, but its *forecasting* capabilities are limited by the chaotic nature of the analyzed system. Thus, small divergences in the representation of initial conditions lead orbits to exponentially diverge after k iterations, especially if one uses recursive forecasting to respect the butterfly effect (Brock et al., 1992).

Unfortunately, the Lyapunov dimension is unfeasible to compute in practical scenarios when generating rules are unknown and/or when the Jacobian matrix is not computable. In order to overcome this issue, one needs to first reconstruct the phase space from a time series $T_i \in [1, d]$ and compute Equation 2.27 for several reference states. Similarly to the correlation dimension, one can extrapolate $k \to \infty$ when dealing with finite datasets by taking the slope of a linear region on the $\mathcal{S}_i(k)$ versus k plot, where $\mathcal{S}(k) = \lambda k$.

2.7 FINAL CONSIDERATIONS

This chapter introduced the main concepts associated with this thesis, starting with the definition of time series to later cover important dynamical-systems characteristics such as trajectories, orbits, and attractors. As highlighted, a fundamental concept and tool for the analysis of such systems is the *phase space*, for which several reconstruction methods have been outlined. By unfolding the dynamical behavior in such space, one can reveal important features about the underlying phenomenon and, therefore, reach a better understanding of the series and its properties. In addition, due to the map between phase states and time-series observations, one can take advantage of modeling phase-space trajectories to forecast chaotic time series.

However, characterizing dynamical systems by analyzing their phase-space representations is challenging. First and foremost, it is not evident how to reconstruct a phase space *in practice*, given a sampled time-dependent phenomenon. Secondly, this reconstruction is subject to various parameters and quality metrics, and it is not evident how to compute, or even define, which is the optimal reconstruction (apart from relatively simple dynamical systems having well-known generating rules). Therefore, the concepts described in this chapter were important to either understand the related work and ii) to formulate the proposed research questions in Chapter 1

Based on the concepts introduced here, we next exemplify the challenges involved in interpreting dynamical systems by introducing more complex examples (Chapter 3). Further on, we describe techniques for estimating embedding parameters in Chapter 4.

3 DATASETS

3.1 INITIAL CONSIDERATIONS

In the previous chapter, we illustrated the concepts and challenges of attractors, phase spaces, and their features by using two quite simple dynamical systems – the Lorenz system and the mechanical pendulum. The Lorenz system has become the typical example in studies on phase-space reconstruction and dynamical systems, mainly due to its easy-to-understand visual structure (Rosenstein et al., 1994; Tucker, 1999; Manabe and Chakraborty, 2007).

As also mentioned in Chapter 2, a major challenge in validating a phase-space reconstruction and, as consequence, its embedding parameters m and τ, is by having the *ground truth* to assess it. That is, for a given dynamical system in general, we can compute a multitude of embeddings. How do we know when an embedding is good enough and which one is the best (if any)? Answering this question is of key importance for our work, as in the next chapters we move to explore and propose methods for computing such optimal embeddings.

To approach this question, we will consider as ground-truth sets of well-known dynamical systems and datasets. By "well known", we mean the following:

- the generating rule $R(\cdot)$ is known, hence embedding parameters can be obtained by trial-and-error after comparing the original and reconstructed phase spaces;
- a specialist in the area of the respective dynamical system defined the phase space that best represents the dynamics of the studied phenomenon; or
- the embedding dimensions and corresponding embedding parameters (m, τ) were estimated according to state-of-the-art methods which are well documented and accepted in the related literature. Those methods are described next in Chapter 4.

Following Chapter 2, we consider two main classes of systems. Section 3.2 presents systems of discrete maps and function types. Section 3.3, on the other hand, describes continuous systems (fluxes).

For each system, we compute a dataset that captures its dynamics. Those datasets were used in our experiments described from Chapter 5 onwards. For completeness and replication purposes, we outline below practical considerations involved in computing such datasets.

Parameters: For each of the systems discussed next, we report its default parameters and constants (Equations 3.1–3.5). Those were chosen to simulate some chaotic behavior according to the literature. This led to structures to form in the phase space, such as fractals (Mandelbrot, 1977) and manifolds (Lee, 2003), as discussed in Chapter 2. Nonetheless, one could use any different set of parameters to simulate a different scenario, without loss of generality.

Presentation: For presentation purposes, components from the first dimension (usually represented as the variable x in most cases) are used to represent the time series. Therefore, $T_i = x(t), \forall t \in [0, n_i - 1]$.

Implementation: All partial equations were solved using the sampling time $t_s = 0.01$ in the R language (Hegger et al., 1999; Antonio, 2013; Garcia and Sawitzki, 2015). All time series were analyzed using a sufficient length (time duration) to preserve the dynamics of the unfolded phase space. Concretely, at least $n = 1000$ observations were used in most cases. However, to facilitate the understanding and avoid clutter, we use fewer observations (*e.g.*, 100) to illustrate some time series in this chapter.

3.2 DISCRETE MAPS AND FUNCTION-BASED SYSTEMS

3.2.1 *Sinusoidal Function*

Given a right-triangle angle, the sinusoidal function represents the ratio between the length of the opposite side of that angle and the hypotenuse, in form

$$x(t) = A(t)\sin(2\pi t/n) + \theta) + \mathcal{U}(a, b), \tag{3.1}$$

where $A(t)$ is the amplitude along time (represented by n samples, $t = 0, \ldots, n-1$), θ is the sinusoidal phase, and $\mathcal{U}(a, b)$ introduces noise following a uniform probability distribution in range $[a, b]$. Due to its periodical and conservative behavior, this function is typically considered to model time-recurrent phenomena, such as sound and light waves, sunlight intensity, and day duration (Bracewell, 1978). In this situation, stochastic components might be added to represent spurious influences along data collection, which is common in real-world scenarios, such as the lack of sensor precision or any other unexpected fluctuations.

The phase space for the sinusoidal function follows an ellipse shape when properly unfolded with $(m = 2, \tau = [1, 20])$. The range of acceptable time delays varies according to the sampling time, and its mainly responsible for the radius of the ellipsoidal attractor, so that greater values are required to overcome lower signal-to-noise ratios (Pagliosa and de Mello, 2017). Figure 3.1 illustrates the phase space reconstructed with $(m = 2, \tau = 1)$ for the default parameters $A(t) = 1, \forall t$, $\theta = 0$, $\mathcal{U}(0, 0), n = 1000$.

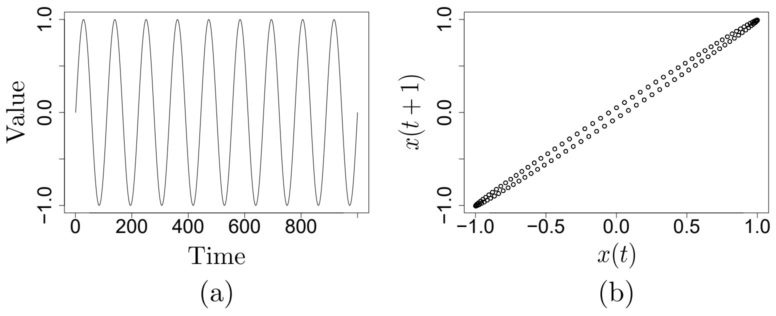


Figure 3.1: Time series **(a)** and phase space **(b)** of the sinusoidal function. Adapted from Pagliosa and de Mello (2017).

3.2.2 *Logistic Map*

The Logistic map

$$x(t+1) = rx(t)(1 - x(t)), \tag{3.2}$$

models the growth rate of populations like bacteria or humans (Robledo and Moyano, 2007). First, the population grows as there are more resources available than consumed. After a while, the population rate tends to decrease given struggles for resources and diseases (if we are modeling humans, for instance) or treatments (if we are modeling bacterias, instead). This trade varies according to the population grow rate r, and, eventually, some balance is found before the beginning of another cycle.

Figure 3.2 illustrates the series (parameters are $x(0) = 0.5$ and $r = 3.8$, $n = 100$) and its phase space, reconstructed with $(m = 2, \tau = 1)$.

3.2.3 *Hénon Map*

The Hénon map (Hitzl, 1981), whose generating rule is given by

$$\begin{aligned} x(n+1) &= 1 - ax(n)^2 + y(n), \\ y(n+1) &= bx(n), \end{aligned} \tag{3.3}$$

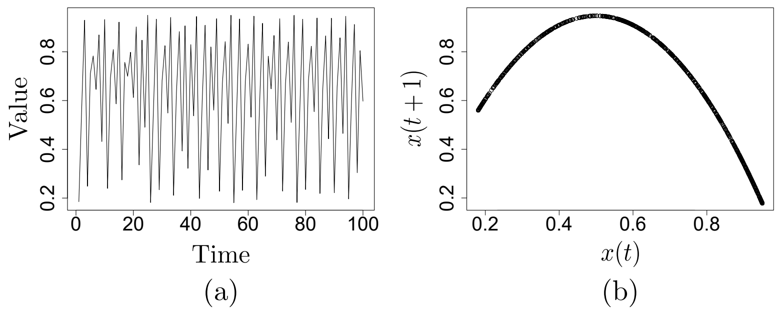


Figure 3.2: Time series **(a)** and phase space **(b)** of the Logistic map. Adapted from Pagliosa and de Mello (2017).

represents the intersection of a periodic orbit of the Lorenz phase space with a certain lower-dimensional subspace, transversal to the flow of the system, also known as the Poincaré section (Diacu, 1999). This function can either approach a set of attractor points or diverge to infinity, based on the starting point. Figure 3.3 shows $n = 100$ observations generated using $x(0) = -0.0064$, $y(0) = -0.4735$, $a = 1.4$, and $b = 0.3$; the phase space was generated using $(m = 4, \tau = 1)$.

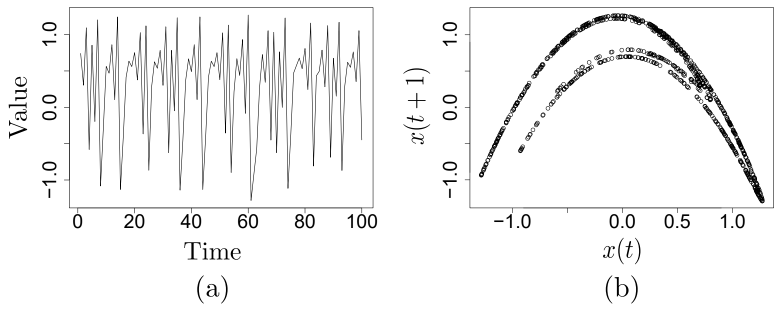


Figure 3.3: Time series **(a)** and phase space **(b)** of the Hénon map. Adapted from Pagliosa and de Mello (2017).

3.2.4 *Ikeda Map*

The Ikeda map, defined as

$$\begin{aligned} z(n+1) &= a + bz(n)e^{ig}, \\ g &= k/(1 + z(n)^2) + c, \end{aligned} \tag{3.4}$$

was proposed as a model of light going around across a nonlinear optical resonator (Ikeda, 1979), in which i is the imaginary unit;

$z(n)$ is the electric field inside the resonator at the n-th step of rotation; a indicates the laser light applied from the outside; c is the linear phase across the resonator; and, finally, b is the chaotic parameter (when $b \leq 1$ there is resonator loss; when $b = 1$ this map becomes conservative). We initialize $z(0)$ and $z(1)$ with random values and used $a = 0.85$, $b = 0.9$, $c = 7.7$, $k = 0.4$. The series with $n = 100$ samples and its phase space, embedded with $(m = 2, \tau = 1)$, are illustrated in Figure 3.4.

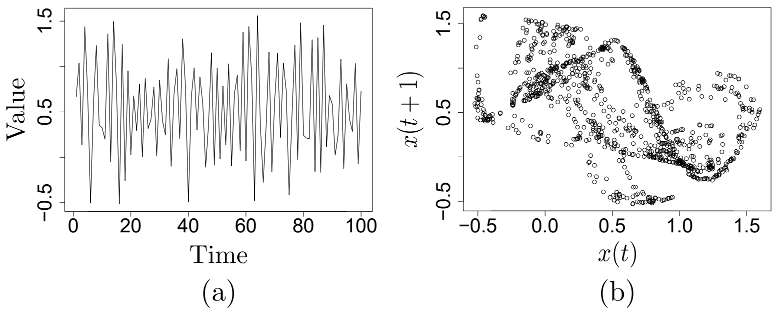


Figure 3.4: Time series **(a)** and phase space **(b)** of the Ikeda map. Adapted from Pagliosa and de Mello (2017).

3.2.5 *Sunspot Dataset*

The real-world Sunspot dataset (Andrews and Herzberg, 1985) is produced from the number of monthly average of sunspots collected from 1749 until 2013. This time series measures the number of sunspots s and groups of sunspots g observed on the Sun surface by n different laboratories around the world. Each laboratory has a relevance to determine the sunspot occurrence, which is based on the facility location and instrumentation capabilities. On top of this time series, previous studies discovered a cyclical solar behavior (known as the solar cycle) of approximately 11 years, close to a sinusoidal signal with noise added. This type of study is relevant since it allows to model, predict, and detect changes on the Sun surface which produce side effects on the space as well as on the Earth surface and atmosphere. In addition, as this series follows sinusoidal-like characteristics (sum of sinusoidal trends with noise), the attractor is expected to have several concentric sharped-elliptic-like trajectories. Indeed, this structured is acquired after reconstructing the phase space using the same parameters for the sinusoidal function, *i.e.*, $(m = 2, \tau = 1)$. Nonetheless, greater values of τ are also accepted.

Also, it is worth to mention that the original series has a lot of fluctuations that jeopardize the visualization of the attractor, as expected from real-world observations. Using the raw data, the phase space yields several concentric ellipses overlapping as approaching the origin, leading to cluttered points that jeopardize the tracking of trajectories. Therefore, Figure 3.5 shows a spline curve fit to 100 equally-spaced observations (10%) of the original series, and the respective phase space. This shorter and smoother version still reassembles the original series, but it makes easier to follow the dynamics of the elliptical phase space.

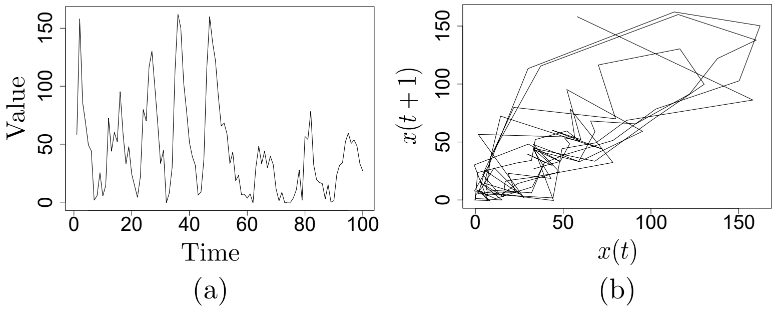


Figure 3.5: Time series **(a)** and phase space **(b)** of the Sunspot dataset. Lines connecting consecutive states depict phase-space trajectories. Adapted from Pagliosa and de Mello (2017).

3.3 CONTINUOUS SYSTEMS

3.3.1 *Lorenz System*

The Lorenz system, defined in Equation 2.6, models atmospheric data to support weather forecasting (Tucker, 1999). This model uses the heat flow between upper and lower surfaces of the observed phenomena, having σ as the ratio of momentum and thermal diffusivity, also known as the Prandtl number; β determines whether the heat transfer is primarily in the form of conduction or convection, also known as the Rayleigh number; and ρ is a geometric factor. We used $x(0) = -13$, $y(0) = -14$, $z(0) = 47$, and $\sigma = 10$, $\beta = 8/3$, $\rho = 28$. Its series and phase space with $(m = 3, \tau = 8)$ are shown in Figure 3.6.

3.3.2 *Rössler System*

The Rössler system, whose generating rule is given by

$$\partial \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -y - z \\ x + \alpha y \\ \beta + \rho(x - z) \end{bmatrix}, \tag{3.5}$$

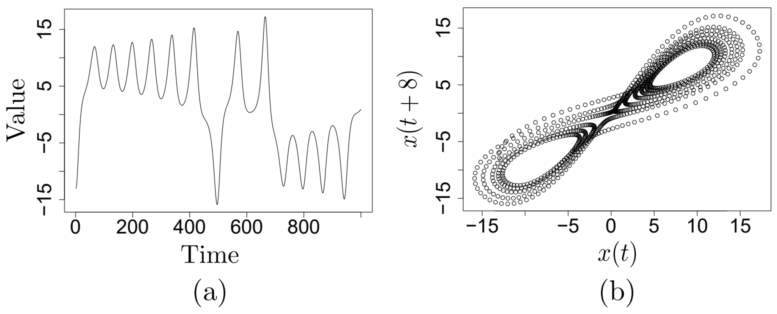


Figure 3.6: Time series **(a)** and phase space **(b)** of the Lorenz system (only two dimensions are shown). Adapted from Pagliosa and de Mello (2017).

was originally designed to behave like the Lorenz system, but providing an easier qualitative analysis (Rössler, 1976). Thus, it shares the same optimal embedding parameters with Lorenz. Typically, chaotic parameters are: $x(0) = -2$, $y(0) = -10$, $z(0) = 0.2$, and $\alpha = 0.2$, $\beta = 0.2$, $\rho = 5.7$. Figure 3.7 depicts the series as well as the resulted phase space.

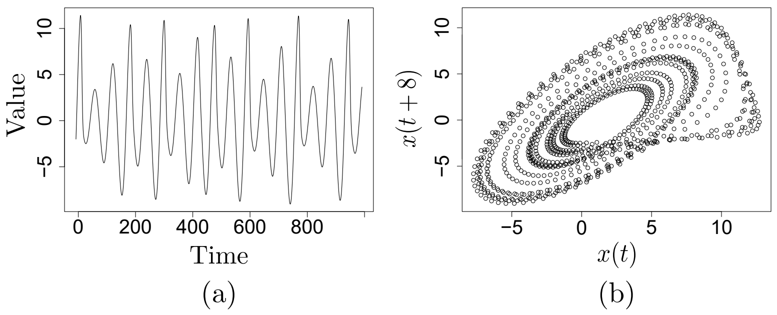


Figure 3.7: Time series **(a)** and phase space **(b)** of the Rössler system (only two dimensions are shown). Adapted from Pagliosa and de Mello (2017).

3.4 FINAL CONSIDERATIONS

This chapter introduced the generating rules, series and phase spaces used along the following chapters. The aim of selecting such systems was to create a consistent set of datasets that describes the behavior of dynamical systems for which ground truth is available in terms of their optimal embedding parameters (m, τ). This way, we can next validate the estimation of other methods.

Apart from those dynamical systems, many more exist and have been studied in the literature (Farzad et al., 2006; Korsch et al., 2008). We refrained from including such additional systems in our work, since they are less pervasive in benchmarks commonly used in most analyses (Alligood et al., 1996; Tucker, 1999; Ott, 2002; Kantz and Schreiber, 2004; Robledo and Moyano, 2007). As such, the comparison with those additional systems *could* have be done, without loss of generality.

4 RECONSTRUCTING PHASES SPACES

4.1 INITIAL CONSIDERATIONS

As introduced in Chapter 2, reconstructing the phase space from a time series is a key step in dynamical systems analysis. For this purpose, the embedding theorem proposed by Takens (1981), also known as method of delays, is the most employed approach in the literature (Packard et al., 1980). This chapter further discusses methods for performing the phase-space reconstruction that are associated with Takens' theorem.

Before describing actual methods to compute embedding parameters (m, τ), let us overview the general considerations that underlie the computation of such embeddings. We outline three points of interest: the use of a single *vs* multiple time delays τ; the effects of overestimating the embedding dimension m; and the impact for the estimation if m and τ are correlated or not.

Single *vs* multiple time delay: Methods for computing phase-space optimal embeddings of time series rely on the fact that a time series T_i, whose observations are the components of a single dimension i of $\mathcal{S}^d$, is directly or indirectly influenced by other non-measured variables $j \in [1, d], \forall j \neq i$. While studying natural phenomena, however, such influences among variables may happen at different timestamps and intervals, leading to nonlinear recurrences and chaos (Kantz and Schreiber, 2004). Thus, one could use multiple time delays $\tau_1, \tau_2, \cdots$ to reconstruct the attractor of S that resembles the features of $\mathcal{S}^d$, as first investigated by Breedon and Packard (1992) and more recently by Manabe and Chakraborty (2007). However, this Ph.D. (as most studies) is mainly concerned in estimating a single τ, which may be interpreted as the *most common or maximal time delay*. Thus, implications of using more than one time delay in phase-space reconstruction is out of the scope of this thesis.

Overestimated embedding dimension: In order to reconstruct the phase space $\mathcal{S} \subset \mathcal{S}^d$, an adequate pair of embedding parameters m and τ is required to unfold the system dynamics. In that sense, given that most real-world phenomena present low-dimensional phase spaces (as shown in Chapter 3), underestimated embeddings are difficult to occur in practice, at least in terms of m. Conversely, estimations leading to an overestimated embedding dimension will not impact the quality of the analysis (as the phase space will be already unfolded), but will increase the *computational effort* needed for modeling, which is undesirable for real-time applications on data streams. Thus, a minimal embedding pair (m, τ) is highly desired. Complementary, an overestimated time delay may generate misleading conclusions, especially when dealing with maps as it is the case of the Hénon and Logistic time series. For instance, a second-order polynomial regression (Björck, 1996) applied over the phase space of Figure 1.3(a) gives

$$x(t+1) = -3.8x(t)^2 + 3.8x(t) - 1.49236 \times 10^{-7}, \tag{4.1}$$

which is enough to get an approximation of the actual generating rule built with $r = 3.8$ (Equation 3.2). With the generating rule available, simpler and more reliable analyses can be performed than when using the raw time series itself. Nonetheless, the attractor structure is lost in Figure 1.3(b), which resembles a phase space from a stochastic process (Alligood et al., 1996). In such cases, no simple regression model is capable of providing the generating rule.

Relation of the two parameters: Regarding embedding parameters, despite Takens proved that m and τ are independent for infinite-length free-noise time series, there is no consent about the true relation among such parameters for finite-noisy observations (Martinerie et al., 1992; Kim et al., 1999; Ma and Han, 2006; Cai et al., 2008, etc.). Some researchers, for instance, focused on finding one of those parameters at a time, assuming they are uncorrelated. Conversely, others believe that these parameters are bounded by the time delay window $t_w = (m-1)\tau$ (a number representing the total time spanned by the components of each embedded point), thus making m and t_w independent instead.

Next, we discuss methods that estimate these parameters assuming that they are uncorrelated (Section 4.2) or dependent (Section 4.3). Lastly, Section 4.4 concludes the chapter with our final considerations.

4.2 ASSUMING INDEPENDENCE OF EMBEDDING PARAMETERS

Methods to estimate the optimal time delay τ (Section 4.2.1) can be mainly grouped in two categories: the ones that use series correlation (Fraser and Swinney, 1986; Albano et al., 1987, 1991; Ma and Han, 2006, etc.), and the ones based on the phase-space expansion (Kember and Fowler, 1993; Rosenstein et al., 1994; Chen et al., 2016, etc.). Despite their differences, such methods do not attempt to estimate m in the process, usually performing computations on the time series itself (which, roughly speaking, is the same as using an embedding dimension $m = 1$) or on some predefined m. For instance, when the generating rule $R(\cdot)$ is given, some authors decide to use the minimal embedding dimension that satisfies Takens' theorem ($m = 2d + 1$).

When estimating the embedding dimension m (Section 4.2.2), on the other hand, authors typically use $\tau = 1$ or the time delay found by one of the previously-mentioned methods (detailed in Section 4.2.1). Also, those algorithms usually take decisions based on phase-states distances. Therefore, the phase state is always assumed to be endowed in the Euclidean space, unless stated differently.

4.2.1 *Estimating The Time Delay*

Several methods exist for estimating the time delay τ: Autocorrelation Function (Section 4.2.1.1), Auto-Mutual Information (Section 4.2.1.2), high-order correlations (Section 4.2.1.3), Singular Value Fraction (Section 4.2.1.4), Average Displacement (Section 4.2.1.5), Multiple Autocorrelation Function (Section 4.2.1.6), and Dimension Derivation (Section 4.2.1.7). These are described next.

4.2.1.1 *Autocorrelation Function*

Let $S = \{x(t), x(t+1), \cdots\}$ and $Q = \{x(t+\tau), x(t+1+\tau), \cdots\}$ denote, respectively, the time series T_i and the same series τ-units shifted along time, namely $T_{i,+\tau}$. The Autocorrelation Function (ACF) quantifies the amount of linear independence in the form

$$R_{xx}(\tau) = \frac{E[(S-\mu)(Q-\mu)]}{\sigma^2}, \tag{4.2}$$

where $E[\cdot]$ is the expected value, μ is the mean, and σ^2 is the variance of T_i. In the scope of phase-space reconstruction, ACF was used to estimate the value of τ that better unfolded the attractor, since authors believed that the more independent states were from each other (up to a certain limit), the greater it is the probability of describing a deterministic rule and form a representative structure. Then, several approaches were proposed on $R_{xx}(\cdot)$ to measure this limit. For instance, Albano et al. (1987); Abarbanel et al. (1993) chose the first zero-crossing $R_{xx}(\cdot)$ to estimate τ, while King et al. (1987) suggested the first inflection point of $R_{xx}(\cdot)$ instead. Later, Albano et al. (1988) achieved more robust results by choosing the time delay that causes $R_{xx}(\cdot)$ to first drop to a certain fraction of its initial value. However, despite that autocorrelation methods provide a good initial estimation for the time delay in some scenarios, such approaches showed to be inconsistent for general systems, probably due to the low correlation among linear dependencies on the time series and the nonlinear structures of the underlying dynamical system (Fraser and Swinney, 1986; Martinerie et al., 1992).

4.2.1.2 *Auto-Mutual Information*

Fraser and Swinney (1986), already knowing the problems involving the linear dependencies of ACF (even earlier than the publication of corresponding articles), proposed a method to measure the general dependencies among signals based on the Auto-Mutual Information (AMI)

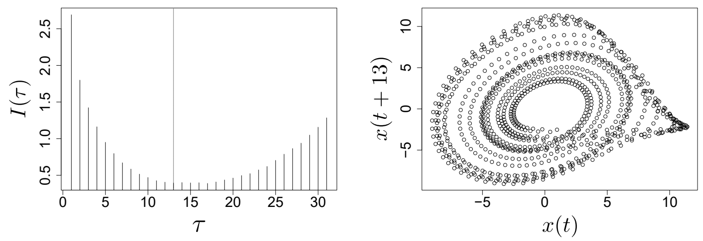
$$I(\tau) = \int_S \int_Q p_{SQ}(s,q) \log_2 \left(\frac{p_{SQ}(s,q)}{p_S(s) p_Q(q)} \right) dxdy, \tag{4.3}$$

where p_S and p_Q are marginal probability densities from the continuous variables S e Q (representing T_i and $T_{i,+\tau}$), respectively, and p_{SQ} is the joint probability density function. Given a measurement $s \in S$, the mutual information is the number of bits of $q \in Q$, on average, that can be predicted. Thus, p_S and p_Q can be estimated by respectively partitioning S and Q into bins and counting occurrences on them. Similarly, p_{SQ} is usually computed by recursively partitioning the plane SQ until each cell is uniformly distributed according to some statistical criteria. As observed later by Rosenstein et al. (1994), the authors used the auto-mutual information in attempt to measure the shift from redundance to irrelevance on time series. As the irrelevance error is more difficult to compute, Fraser and Swinney (1986) associated the first local minimum of AMI to the optimal τ (less redundant time delay).

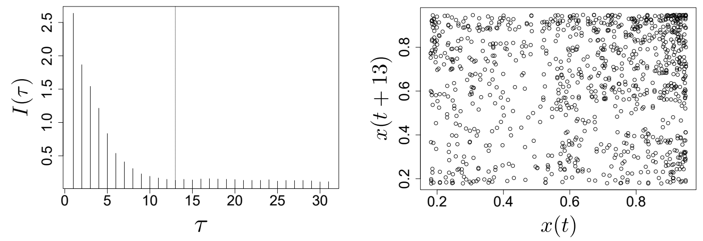
Liebert and Schuster (1989) later showed that such minima coincide with those found using the correlation integral (Theiler, 1987; Wong et al., 2005). However, Martinerie et al. (1992) empirically observed that neither ACF (using the previously described measurements) nor AMI were consistent to find the optimal value of t_w (and, as consequence, a bound for m and τ), as illustrated in Figure 4.1.

4.2.1.3 *High-Order Correlation*

Albano et al. (1991) noticed that time series deriving from multivariate systems should carry multivariate cumulants and moments. By investigating the relations of high-order correlations on the columns of the trajectory matrix (formed by states of the phase space, as described in Equation 2.12), they observed that a number of correlation functions, although not consistently the same ones, have extrema occurring at the same time, later referred to as t_c.



(a)



(b)

Figure 4.1: Although a good estimation for the Rössler system is obtained **(a)**, the first minimum of AMI (left) overestimated the time delay for the Logistic map **(b)**, leading to a poor reconstruction of the attractor (right). Coincidentally, $\tau = 13$ was found for both attractors (solid line).

As there are no previous reason for such coincidence, the authors empirically defined t_c as the time delay window $t_w = (m-1)\tau$, which defines important attractor features. However, no patterns were found to deterministically correlate t_c with t_w (and τ).

4.2.1.4 *Singular Value Fraction*

Kember and Fowler (1993) proposed the Singular Value Fraction (SVF) method, which estimates the time delay when the attractor is mostly expanded in all dimensions. As the expansion of an attractor can be described by the spreading rate of its phase states, *i.e.*, in function of the singular values $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m \geq 0$ of the trajectory matrix (Equation 2.12), they attempt to find to the (ideal) case when all eigenvalues are equal. Thus, given the function

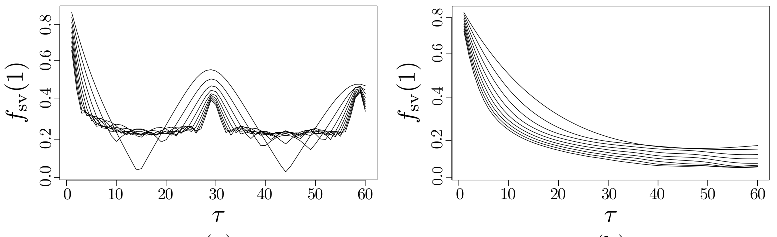
$$F_{\text{sv}}(k) = \frac{\sum_{j=1}^{k} \lambda_j^2}{\sum_{j=1}^{m} \lambda_j^2}, \tag{4.4}$$

SVF is defined as

$$f_{\text{sv}}(k) = \frac{mF_{\text{sv}}(k) - k}{m-k}, \tag{4.5}$$

so that $0 \leq f_{\text{sv}}(k) \leq 1$ and $f_{\text{sv}}(m) = 1$. By this definition, Equation 4.5 only approaches zero when $F_{\text{sv}} = k/m$, meaning $\lambda_j^2 = c, \forall i \in [1, m]$, where c is a constant. For general attractors, however, hardly ever $f_{\text{sv}} = 0$, and, consequently, τ was defined as the first timestamp SVF reaches a local minimum. Moreover, the authors empirically observed that $f_{\text{sv}}(1)$, *i.e.*, the average of eigenvalues in function of the most spread dimension, led to better estimations of the time delay. From our point of view, such approach was an early attempt to overcome the well-known "curse of dimensionality" (Chen, 2009).

One interesting point about this method is that the plot of $f_{\text{sv}}(1)$ versus τ gives a simple (but important) idea about the time-series recurrences and the interval of its periodicities, as illustrated in Figure 4.2(a). As it can be seen, this figure shows that the Rössler system has cyclical behavior each 29 units of time, providing empirical evidence that the first peak of SVF could be used to estimate the time delay window. Despite consistent results for different dimensions (including noisy data), the same strategy failed for the Lorenz system, as shown in Figure 4.2(b). As a counterargument to this limitation, the authors were able to enhance results for the Lorenz series by powering the observations in the form $T_i^{\alpha=2} = \{x(t)^2, x(t+1)^2, \cdots\}$, where α is the manifold genus (number of voids or holes). However, this approach may be difficult to be reproduced when there is no *a priori* knowledge about the system.



(a) (b)

Figure 4.2: **(a)** The SVF method found $\tau = 10$ for the Rössler system when embedded using $m = 3$, which is a good estimate for the time delay. Also, a constant time-delay-window length was found for different dimensions, empirically reinforcing the belief that m and τ are independent to each other. **(b)** On the other hand, SVF does not provide any useful information for the Lorenz system: there is neither a minimum nor a peak. From outer to inner curves, $m = 2, \cdots, 10$ in both plots.

Similarly, Chen et al. (2016) proposed Singular Entropy (SE), a modified version of SVF. They based their algorithm on the same principle of equality as SVF, but used the ratio of the entropy of eigenvalues instead

$$E(k) = \sum_{j=1}^{k} \Delta E_j, \tag{4.6}$$

where

$$\Delta E_j = -P_j \log P_j, \quad P_j = \frac{\lambda_j}{\sum_{j=1}^{m} \lambda}. \tag{4.7}$$

Given Equation 4.6 was based on Shannon's entropy (Hammer et al., 2000) of the singular values, the major difference between SE and SVF is the sign of their results, and, therefore, the concavity of their plots. Thus, similar results given by SVF, such as τ and t_w, were also obtained while analyzing inverse features of SE, *i.e.*, the minimum of SVF becomes the maximum of SE and vice-versa, as illustrated in Figure 4.3.

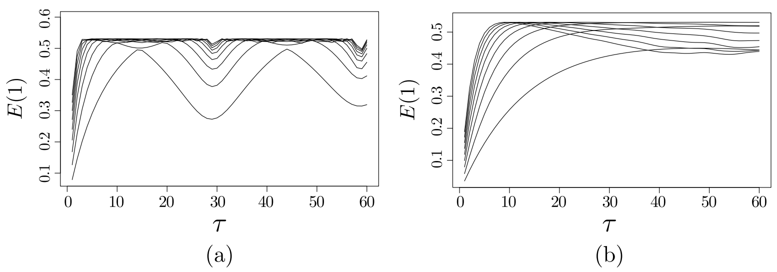


Figure 4.3: SE leads to similar conclusions as the SVF method for the Rössler **(a)** and Lorenz **(b)** systems. From outer to inner curves, $m = 10, \cdots, 2$.

4.2.1.5 *Average Displacement*

An interesting discussion about irrelevance and redundance, as well as the errors associated with both structural conditions, were conducted by Rosenstein et al. (1994). Despite the difficulties in measuring the irrelevance error (there are no guarantees the reconstruction will become less space-filling as the lag increases beyond the optimal time delay), the authors assumed the irrelevance error to be lower than the redundance error, therefore focusing on minimizing the latter while searching for a proper time delay. In this scenario, they inversely measure the relation between the redundance error and the attractor expansion in form of the Average Displacement (AD) of phase states, as function of the time delay

$$S_m(\tau) = \frac{1}{N_i} \sum_{t=0}^{N_i-1} \sqrt{\sum_{j=1}^{m-1} (x(t+j\tau) - x(t))^2}. \tag{4.8}$$

As illustrated in Figure 4.4, it was observed that AD increases until it reaches a plateau on the plot $S_m(\tau)$ versus τ, which indicates the attractor is sufficiently expanded. Thus, the authors empirically observed that a good estimation for τ was given when the slope of such relation decreased by less than 40% of its initial value. Additionally, the authors also noticed that this criterion corroborated with the assumption that τ and m are correlated within the time delay window: increasing the values of m contributes to the early formations of the plateau (and early estimations for τ). Indeed, the pairs (m, τ) found for the Lorenz system (shown in the figure) were $\{(14, 4), (8, 5), (5, 8), (3, 14)\}$. Despite those contributions, some authors argued that the dependency on the slope of the AD method may consider non-ignorable errors (Ma and Han, 2006).

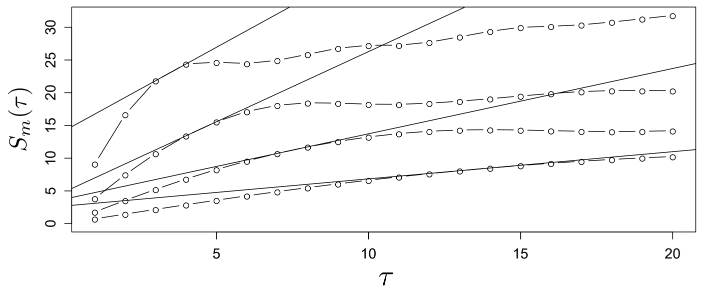


Figure 4.4: Results of the average displacement for the Lorenz system. Diagonal lines indicate the first time delay in which the slope of the curve decreases by less than 40% of its initial value. From outer to inner curves: $m = 14, 8, 5, 3$.

4.2.1.6 *Multiple Autocorrelation Function*

Rosenstein et al. (1994) also investigated the relation between AD and ACF. They squared Equation 4.8 as follows

$$S_m^2(\tau) = \frac{1}{N_i} \sum_{t=0}^{N_i-1} \sum_{j=1}^{m-1} (x(t+j\tau) - x(t))^2, \tag{4.9}$$

so that $S_m^2(\tau)$ could be interpreted as a scaled version of $S_m(\tau)$ as $S_m^2(\tau) = f(\tau)S_m(\tau)$. Moreover, as ACF (Equation 4.2) from a finite set can be approximated by

$$R_{xx}(\tau) \approx \frac{1}{n_i - \tau} \sum_{t=0}^{n_i-\tau} x(t)x(t+\tau), \tag{4.10}$$

it can be shown that, under some assumptions

$$S_m^2(\tau) \approx c - 2R_{xx}^m(\tau), \tag{4.11}$$

where c is a constant and $R_{xx}^m(\tau) = \sum_{j=1}^{m-1} R_{xx}(j\tau)$. After this definition, Rosenstein et al. (1994) noticed that $R_{xx}(\cdot)$ has the same shape as S_m^2 when $m = 2$, indicating the easier-to-compute ACF should be used in the place of S_m^2 for two-dimensional systems. However, as most systems are at least three-dimensional and S_m^2 tends to become more sensitive to variations as m is increased, the authors concluded that the Multiple Autocorrelation Function (MACF), as later referred to S_m^2, should lead to the poor estimation of attractors for high-dimensional systems.

4.2.1.7 *Dimension Derivation*

More recently, Tamma and Lachman Khubchandani (2016) proposed a method to find both embedding parameters m and τ from a time series. However, as the method to find the embedding dimension m was very similar to FNN (Kennel et al., 1992), the main contribution of their article was the estimation of τ, which is based on the attractor expansion and the pointwise correlation dimension, defined as

$$D_p(t,\tau) = \lim_{\epsilon \to 0} \frac{\log(P(t,\tau,\epsilon))}{\log(\epsilon)} \tag{4.12}$$

where $P(t,\tau,\epsilon)$ is the probability of two phase states, reconstructed using a predefined m and the chosen τ, to be closer than the open ball centered in $\boldsymbol{\phi}_i(t)$ with radius ϵ, similar to (Equation 2.21). Since $D_p(t,\tau)$ should be invariant to any chosen state $\boldsymbol{\phi}_i(t)$ (Ott, 2002), a well-reconstructed phase space should present zero Dimension Deviation (DD) among their pointwise correlations, *i.e.*

$$f(\tau) = \frac{1}{N_i} \sum_{t=0}^{N_i-1} (D_p(t,\tau) - \overline{D}_p(t,\tau))^2 = 0, \tag{4.13}$$

where $\overline{D}_p(t,\tau) = \sum_{t=0}^{N_i-1} D_p(t,\tau)/N_i$.

Due to noise and floating point fluctuations, however, Equation 4.13 would hardly be equal to zero. Hence, the authors defined the first minimum of $f(\tau)$ (over a predefined range of time delays) as the optimal time delay. However, results showed that neither this method nor the choice of the first minimum are robust to estimate τ for general attractors, as illustrated in Figure 4.5. There, the fourth minimum would led to the closest optimal estimation for the Rössler system. Lastly, the authors claimed that a representative set of states could be used instead of all phase states in order to improve performance.

To check this out, we tested the DD algorithm using (roughly) 1000 phase states and 200 (representative) centroids. Following our implementation, we found a smoother curve for the plot $f(\tau)$ versus τ. This actually hindered the estimation for the time delay, as depicted in Figure 4.6.

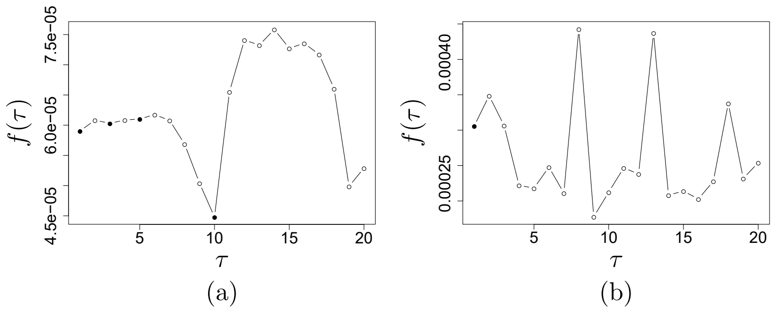


Figure 4.5: Results of the DD method for the Rössler system **(a)** and Logistic map **(b)** over the range of $\tau \in [1, 20]$. Filled circles represent the number of minima until the closest estimation for the optimal time delay.

4.2.2 *Estimating The Embedding Dimension*

Methods to estimate the embedding dimension m comprise False Nearest Neighbors (Section 4.2.2.1), Gamma test (Section 4.2.2.2), and others based on the fractal dimension (Section 4.2.2.3), as follows.

4.2.2.1 *False Nearest Neighbors*

Kennel et al. (1992) proposed False Nearest Neighbors (FNN) to estimate the optimal embedding dimension m. By using the time delay found after AMI (Section 4.2.1.2), the method reconstructs the attractor using different dimensions $m \in [m_{\min}, m_{\max}]$ and computes, for each of them, the index set of the k-nearest neighbors $N_m(\boldsymbol{\phi}_i(t), k)$ for each phase state $\boldsymbol{\phi}_i(t) \in \mathcal{S}^m$ (the number of phase states is kept constant for different embeddings). The opti-

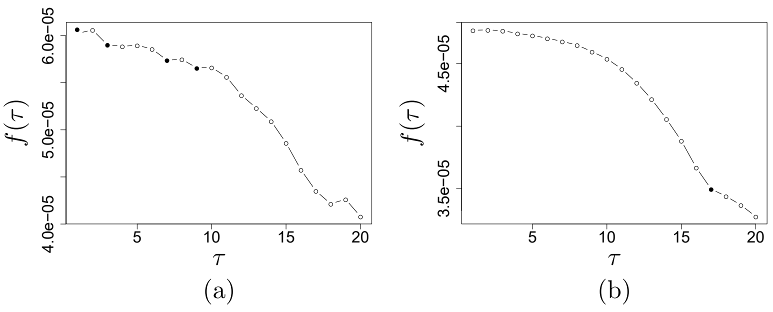


Figure 4.6: The DD method applied to the Lorenz system using $m = 3$. **(a)** 200 centroids led to an acceptable time delay $\tau = 10$, but only when the fourth minimum of $f(\tau)$ is used. **(b)** The first minimum yielded $\tau = 17$ when all phase states are considered. As in Figure 4.5, filled circles represent the number of minima until the closest estimation for the optimal time delay.

mal dimension m is defined as the one in which a fraction of the nearest neighbors in the attractor remains constant as the dimension increases, *i.e.*, $N_m(\boldsymbol{\phi}_i(t), k) = N_{m-1}(\boldsymbol{\phi}_i(t), k)$. This assumption could be relaxed, however, by analyzing how many neighbors (usually 30%) remained close to each other according to a threshold r_{tol}, as m is increased.

In other words, if the kth nearest neighbor of the phase state $\boldsymbol{\phi}_i(t)$ is $\boldsymbol{\phi}_i(t')$, then the Euclidean distance between them is given by

$$R^2_m(t,\tau,k) = \sum_{j=1}^{m}(x(t+j\tau) - x(t'+j\tau))^2. \tag{4.14}$$

After adding one more dimension, we get

$$R^2_{m+1}(t,\tau,k) = R^2_m(t,\tau,k) + (x(t+(m+1)\tau) - x(t'+(m+1)\tau))^2. \tag{4.15}$$

If the distance between such states become greater than r_{tol} after increasing one dimension, *i.e.*

$$\begin{gathered}\left(\frac{R^2_{m+1}(t,\tau,k) - R^2_m(t,\tau,k)}{R^2_{m+1}(t,\tau,k)}\right)^{1/2} = \\ \frac{x(t+(m+1)\tau) - x(t'+(m+1)\tau)}{R^2_{m+1}(t,\tau,k)} > r_{\text{tol}},\end{gathered} \tag{4.16}$$

then the states are considered false neighbors of each other, as illustrated in Figure 4.7.

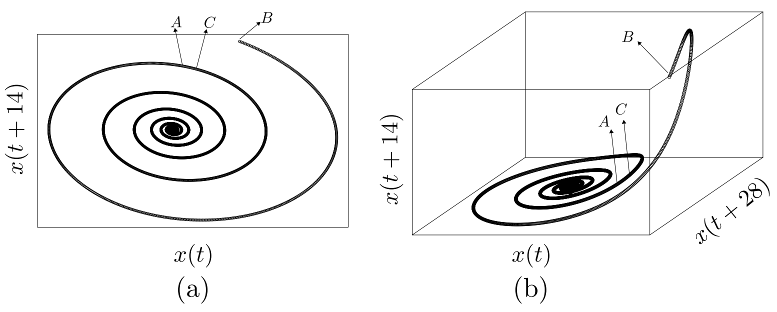


Figure 4.7: Example of false nearest neighbors. Representation of 100 states of the Rössler system. For the sake of simplicity, three states are depicted by letters A, B, C. In the example, A and C remain neighbors from each other after expanding the space in one dimension. State B is labeled as a false neighbor.

Kennel et al. (1992) stated that as long as there is some variation in the attractor, the phase space is not completely unfolded and, thus, the number of false neighbors is still too large. According to their experiments, $r_{\text{tol}} \geq 10$ and $k = 1$ were enough to correctly identify false neighbors[1]. A problem raised by the authors is the lack of effectiveness of FNN in the presence of noise. A simple example was performed using a white noise following the Normal distribution $\mathcal{N}(0, 1^2)$, which is known to have a high-dimensional (but finite) attractor. In this scenario, FNN tends to find greater values of m as the number of states increases, *i.e.*, $N = \infty \rightarrow m = \infty$. This occurs mainly due to the curse of dimensionality (Chen, 2009), in which all states tend to become dissimilar to each other as the dimension increases, never dropping the rate of false neighbors. As observed, this is a contrast to commonly found low-dimensional attractors, where more states in the phase space should contribute to a better formation of its structure (and to the confirmation of its dimension).

Although this method does not achieve consistent results for general time series (which usually contain noisy observations), due to seminal contributions and simplicity, FNN still remains as one of the most used methods to estimate the embedding dimension.

1 It is not uncommon, however, to find algorithms that use $r_{\text{tol}} \geq 20$.

4.2.2.2 *Gamma Test*

Otani and Jones (1997) proposed to use the Gamma test (or Γ test) (Stefánsson et al., 1997) to find the optimal embedding dimension m. Let

$$\boldsymbol{y} = f(\boldsymbol{X}) + r \Rightarrow [y_0, y_1, \cdots, y_N] = f([\boldsymbol{x}_0, \boldsymbol{x}_1, \cdots, \boldsymbol{x}_N]) + r, \quad (4.17)$$

be a function applied over m-dimensional phase states $\boldsymbol{\phi}_i(t), \forall t \in [0, N_i - 1]$, such that every pair $(\boldsymbol{x}_t, y_t)$ is defined as

$$([x(t), \cdots, x(t + (m-2))\tau], x(t + (m-1)\tau)). \quad (4.18)$$

The adjust parameter r can be seen as the amount of noise in the system or the lack of determinism of f. Therefore, if one assumes a controlled level of noise, the variance of r, namely σ_r^2, estimates the quality of the reconstructed phase space. Given the statistic

$$\gamma = \frac{1}{2N_i} \sum_{t=0}^{N_i-1} (y_{N_m(y_t,1)} - y_t)^2, \quad (4.19)$$

where $N_m(y_t, 1)$ is the index of the nearest neighbors of y_t, one can show that $\gamma \to \sigma_r^2$ as $N \to \infty$. Let $\boldsymbol{N}^x = N_m(\boldsymbol{x}_t, k)$ and $\boldsymbol{N}^y = N_m(y_t, k)$ be the index vectors of the k-nearest neighbors of $\boldsymbol{x}_t$ and y_t, respectively. If $\|\cdot\|_2$ is the Euclidean norm, then

$$\Delta(k) = \frac{1}{k} \sum_{h=1}^{k} \frac{1}{N_i - 1} \sum_{t=0}^{N_i-1} \|\boldsymbol{x}_{\boldsymbol{N}_h^x} - \boldsymbol{x}_t\|_2^2, \quad (4.20)$$

computes the mean square distance of the $(h \leq k)$-nearest neighbors and

$$\Gamma(k) = \frac{1}{k} \sum_{h=1}^{k} \frac{1}{2N_i} \sum_{t=0}^{N_i-1} \|y_{\boldsymbol{N}_h^y} - y_t\|_2^2, \quad (4.21)$$

is an estimation of γ for the $(h \leq k)$-nearest neighbors. Based on both quantities, the Γ-test algorithm estimates $\gamma \to \sigma_r^2$ by means of the linear correlation of $\Delta(k)$ and $\Gamma(k)$. In a perfect deterministic scenario, where $\sigma_r^2 = 0$, $\boldsymbol{x}_t$ should lead to exactly one y_t, and $\boldsymbol{N}^x = \boldsymbol{N}^y$. Therefore, the ideal plot of $\Delta(k)$ versus $\Gamma(k)$ should form a straight diagonal line for different values of k. Due to noise and fluctuations, this plot becomes irregular in practice. After a linear regression, the line $y = mx + \overline{\Gamma}$ that best fits the plot is an estimation of Equation 4.17, so that $\overline{\Gamma} \approx \sigma_r^2$.

Otani and Jones (1997) noted that the phase space should be described by non-deterministic states for small values of m, yielding greater values for σ_r^2. For a sufficient m, the attractor is unfolded and should present the most deterministic structure (lower σ_r^2). For greater values of m, neighbors suffer interferences from different orbits, again increasing σ_r^2. Therefore, the optimal embedding was defined as the dimension when $\overline{\Gamma}$ reaches its first minimum.

As the authors also observed, the less $\overline{\Gamma}$ approaches to zero, the more non-deterministic are the state changes, and there is no guarantee in reconstructing the attractor accurately. This may happen if the signal-to-noise ratio is low or when the choice of the time delay is poor. With this in mind, rather than using an elsewhere estimated time delay, one can propose an extended algorithm to find (m, τ) using an acceptable trade-off between the smaller values of $\overline{\Gamma}$ and m, assuming $\overline{\Gamma}$ initially decreases as m is increased.

4.2.2.3 *Methods Based On The Fractal Dimension*

As already explained at various points, the reconstruction of attractors using the method of delays relies on the mathematical framework proposed by Takens (1981). His theorem states that a sufficient reconstruction is achieved by using at least $m \geq 2d+1$ dimensions for a d-dimensional system. For common phenomena, however, m is (usually) considerably smaller, *i.e.*, $m \leq 2d+1$. In addition, as stated by Otani and Jones (1997), the effective dimensionality of some dissipative system is that of its attractor, which may be lower than the number of variables specialists use to describe it. Therefore, if the dimension of the attractor is $d' \leq d$, then m is at least d' (Farmer and Sidorowich, 1987). Based on both facts, the relation

$$d' \leq m \leq 2d' + 1, \tag{4.22}$$

can be used to estimate or validate the embedding dimension m. However, in order to apply Equation 4.22, one needs to rely on two assumptions: i) embedding the phase space using $m \leq 2d+1$ is enough to unfold the dynamics of the underlying system; and ii) the attractor dimension d' is well estimated by fractal-based methods (see Section 2.6.1). Thus, methods based on this approach may be inconsistent for, and not applicable to, general systems.

4.3 ASSUMING DEPENDENCE TO ESTIMATE THE EMBEDDING PARAMETERS

A separate class of algorithms does not use previous estimations or any other information about the underlying dynamical system. Based on the assumption τ and m are correlated by the time delay window $t_w = (m-1)\tau$, some authors believe that important system features (*e.g.*, correlation dimension) can be revealed using different combinations of (τ, m). Therefore, as some authors strive on defining such window, others focus on directly finding the simpler, but sufficient, set of parameters m and τ that provides a good reconstruction. Therefore, most algorithms rely on Monte Carlo simulations (Landau and Binder, 2005; Rubinstein and Kroese, 2007) to find both parameters. This makes such methods quite computationally expensive.

Methods that estimate m and τ assuming some dependence between them include: Wavering Product (Section 4.3.1), Fill Factor (Section 4.3.2), C-C method (Section 4.3.3), Entropy Ratio (Section 4.3.4), Non-Biased MACF, Gamma test (Section 4.3.5), and neural networks (Section 4.3.6), as follows.

4.3.1 *Wavering Product*

The Wavering Product (WP) was one of the first methods designed to find the embedding dimension m and the time delay τ at the same time. Similarly to FNN, this method uses an expansion-based approach relying on the fact that a well-reconstructed embedding should have consistent topological structure, such as neighborhood relationship. Differently from FNN, Liebert et al. (1991) used the distances between the old $\mathcal{S}^m$ and new $\mathcal{S}^{m+1}$ embeddings in their method, as explained next.

In order to keep consistency with the previous presented nomenclature, let

$$R^2_m(t, \tau, k, m+1) \tag{4.23}$$

be the Euclidean distance from the phase state $\boldsymbol{\phi}_i(t) \in \mathcal{S}^m$ to its kth nearest neighbor, but measured in the embedding $\boldsymbol{\Phi}_i \in \mathcal{S}^{m+1}$, having the inverse applied to $R^2_{m+1}(t, \tau, k, m)$. In this context, the authors defined the ratio

$$Q_1(t, \tau, k, m) = \frac{R^2_{m+1}(t, \tau, k, m)}{R^2_{m+1}(t, \tau, k, m+1)}, \tag{4.24}$$

so that $Q_1(t, \tau, k, m) = 1$ when the distance of $\boldsymbol{\phi}_i(t)$ to its kth nearest neighbor remains equal in both embeddings, and $Q_1(t, \tau, k, m) > 1$ otherwise. Nonetheless, even when $\boldsymbol{\phi}_i(t)$ has the same set of p-nearest neighbors for greater dimensions, Equation 4.24 becomes sensitive to the order of how neighbors are rearranged. Such an order may oscillate from one dimension to another as well as due to the presence of noise (hence the name *wavering*). In order to mitigate this, the authors applied

$$P_i(m, \tau) = \left(\prod_{k=1}^{p} Q_1(t, \tau, k, m) \right)^{1/p} \tag{4.25}$$

over all p-nearest neighbors. However, as P_i does not tend to 1 even for sufficiently-known embeddings, the authors introduced a second ratio

$$Q_2(t, \tau, k, m) = \frac{R_m^2(t, \tau, k, m)}{R_m^2(t, \tau, k, m+1)}, \tag{4.26}$$

analogous to Q_1, but calculated using the old embedding. Thus, the wavering product, defined as

$$W_i(m, \tau) = \left(\prod_{k=1}^{p} Q_1(t, \tau, k, m) Q_2(t, \tau, k, m) \right)^{1/2p}, \tag{4.27}$$

should be approximately equal to 1 for sufficient reconstructions, as states in such attractor should have the same set of neighbors in both old and new embeddings.

Liebert et al. (1991) defined $\overline{W}(m, \tau) = \log \langle W_i(m, \tau) \rangle_s$ as the WP average deviation over a sufficient number of s reference points, such as 10% of the whole dataset. Finally, they applied $\overline{W}(m, \tau)/\tau$ versus τ over a range of embedding dimensions, and according to their experiments, the first minimum of each curve provided a good estimation for τ. In addition, as m was increased, curves tended to assume a constant value, such that the first dimension before the constant curves should be used as an estimation for m (and the minimum of this curve should define τ). Although curves converge to constant values for greater time delays (allowing the definition of upper bounds for τ), no further patterns were found to correlate curves to the embedding dimension m.

4.3.2 *Fill Factor*

Buzug and Pfister (1992a) proposed two methods to find the embedding parameters for a time series: Fill Factor (FF) and Local Integral Deformation (LID). While both rely on the expansion of states, the former is based on the volume of the attractor, whereas the latter on the trajectories evolution[2].

Given a candidate pair (m, τ), the FF method selects $m+1$ phase states $\boldsymbol{\phi}_i(t_{r_1}), \boldsymbol{\phi}_i(t_{r_2}), \cdots, \boldsymbol{\phi}_i(t_{r_{m+1}})$, where the subscript r_j is a randomly selected index $j \in [1, N_i]$. Then, defining a pivot state, namely $\boldsymbol{\phi}_i(t_{r_1})$, the m-dimensional vector

$$\boldsymbol{d_j}(t_{r_1}) = \boldsymbol{\phi}_i(t_{r_j}) - \boldsymbol{\phi}_i(t_{r_1}), \; j \in [1, m+1], \tag{4.28}$$

2 LID is just cited in here, as FF is more robust and well-known. In addition, LID is more complicated to be implemented and computationally expensive.

can be seen as one edge of the hyperparallelepiped described by the $m \times m$ matrix $\boldsymbol{M}_{m,r_1}(\tau) = (\boldsymbol{d_1}(t_{r_1}), \boldsymbol{d_2}(t_{r_1}), \cdots, \boldsymbol{d_m}(t_{r_1}))$, whose volume is given by the determinant

$$V_{m,r_1}(\tau) = |\det[\boldsymbol{M}_{m,r_1}(\tau)]| \,. \tag{4.29}$$

To compute an average volume over s parallelepipeds from the reconstructed attractor (the authors used $s = 2\%$ of states), the quantity

$$F_m(\tau) = \frac{\frac{1}{s}\sum_{k=1}^{s} V_{m,r_s}(\tau)}{(\max(T_i) - \min(T_i))^m} \tag{4.30}$$

is used, so that the logarithmic fraction of volume

$$f_m(\tau) = \log_{10} F_m(\tau) \tag{4.31}$$

is defined as the fill factor. By analyzing the plot $f_m(\tau)$ versus τ, results empirically led the authors to chose the first non-constant line as m, and the maximum of FF in that dimension as the best time delay. Despite that Buzug and Pfister (1992a) achieved good estimations for the Duffing system (Korsch et al., 2008) using FF, they noticed this method is not robust when applied to attractors with more than one unstable focus, as the Lorenz system (see result in Figure 4.8). Based on this last figure, it is clear that neither patterns nor effective maximum were found to proper estimate the embedding dimension and the time delay. As those attractors are not uncommon, FF becomes, at least as proposed, infeasible to be applied for general systems. Additional drawbacks of FF and LID are found in (Rosenstein et al., 1994).

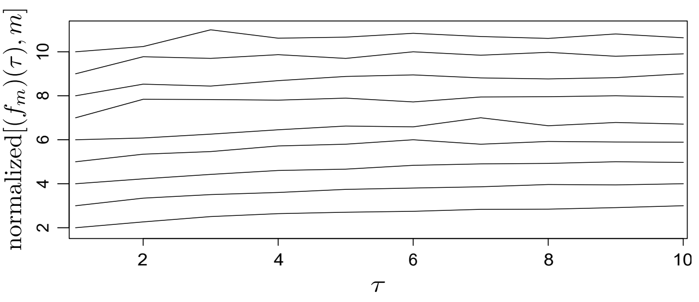


Figure 4.8: Fill Factor applied on the Lorenz system. No patterns were generated by our implementation testing this method, which raises concerns about its reproducibility in all cases. Due to the difference of magnitudes while assessing FF results for different dimensions m, values were normalized in range $[m, m+1]$.

4.3.3 *C–C Method*

Kim et al. (1999) proposed the C–C method to find the optimal time delay τ and the time delay window t_w. Nonetheless, both parameters can also be used to estimate the embedding dimension as well, since $t_w = (m-1)\tau$. The authors were inspired by the BDS statistic (Brock et al., 1992), which can be used to test the null hypothesis that a time series is independently and identically distributed (i.i.d.). Briefly speaking, Brock et al. (1992) noticed that if a n_i-length time series T_i was generated from a completely stochastic process, then the reconstructed phase space $\mathbf{\Phi}_i$ respects the following power rule

$$C(m, \tau, r, n_i) = C^m(1, \tau, r, n_i), \;\; r > 0, \tag{4.32}$$

where $C(m, \tau, r, n_i)$ is the correlation dimension (some extra arguments were put in evidence, when compared to Equation 2.23) of $\mathbf{\Phi}_i$ embedded with (m, τ), measured with decreasing radius ϵ. Therefore, the C–C method quantifies

$$S(m, \tau, \epsilon, n_i) = C(m, \tau, \epsilon, n_i) - C^m(1, \tau, \epsilon, n_i), \tag{4.33}$$

which yields zero for infinite i.i.d. time series. However, for finite-length observations collected from natural phenomena, $S(m, \tau, r, n_i) \neq 0$ due to nonlinear correlations. Therefore, the C–C method is concerned to measure dependencies on a time series. In order to eliminate spurious temporal correlations, the approach subdivided T_i in τ disjoint sub-series: $[\{x(t), x(t+\tau), \cdots\}, \{x(t+1), x(t+1+\tau), \cdots\}]$, and computes

$$S(m, \tau, r, n_i) = \frac{1}{\tau}\sum_{s=1}^{\tau}[C_s(m, \tau, r, \frac{n_i}{\tau}) - C_s^m(1, \tau, r, \frac{n_i}{\tau})], \tag{4.34}$$

where C_s is the correlation integral for the sth sub-series. Then, if T_i is i.i.d., any fixed values of m and τ respect the relation

$$S(m, \tau, \epsilon) = \frac{1}{\tau}\sum_{s=1}^{\tau}[C_s(m, \tau, \epsilon) - C_s^m(1, \tau, \epsilon)] = 0, \tag{4.35}$$

for all ϵ.

On the other hand, for general time series, Kim et al. (1999) defined that the best representation of independence among a finite set of observations is either given by the first zero-crossing of $S(m, \tau, r)$ or when $S(m, \tau, r)$ shows the least variation of ϵ, in the form

$$\Delta S(m, \tau) = \max\{S(m, \tau, \epsilon)\} - \min\{S(m, \tau, \epsilon)\}. \tag{4.36}$$

In addition, the zero-crossing of $S(m, \tau, \epsilon)$ should be nearly the same for all m and ϵ, as well as the minimum of $\Delta S(m, \tau)$ for all values of m. With this in mind and after defining representative ranges for $m \in [m_{\min}, m_{\max}]$ and $\epsilon \in [\epsilon_{\min}, \epsilon_{\max}]$[3], the averages

$$\overline{S}(\tau) = \frac{\sum_{m=m_{\min}}^{m_{\max}} \sum_{\epsilon=\epsilon_{\min}}^{\epsilon_{\max}} S(m, \tau, \epsilon)}{(m_{\max} - m_{\min})(\epsilon_{\max} - \epsilon_{\min})}, \tag{4.37}$$

and

$$\Delta\overline{S}(\tau) = \frac{\sum_{m=m_{\min}}^{m_{\max}} \Delta S(m, \tau)}{m_{\max} - m_{\min}}, \tag{4.38}$$

were used to estimate τ and t_w, respectively. As t_w should describe the interval for optimal independence on series observations, and τ should be the first locally optimal time, the authors defined τ as the first zero-crossing of $\overline{S}(\tau)$ or the first local minimum of $\Delta\overline{S}(\tau)$, and t_w as the time instant when both measures are close to zero, *i.e.*, the minimum of the quantity

$$S_{\text{cor}} = \Delta\overline{S}(\tau) + |\overline{S}(\tau)|. \tag{4.39}$$

Despite the contribution on proposing a different point of view to find τ and t_w, the results presented by Kim et al. (1999) overestimated τ and m. For instance, they reported that the C−C method estimated $m = 8$ and $\tau = 18$ for the Lorenz system, which is known under the presented circumstances to have a representative attractor with $m = 3$ and $d = 8$. Later, Cai et al. (2008) listed more drawbacks of the C−C method and proposed improvements which led to a new strategy called C−C−1. However, there is, to our knowledge, no evidence that any of the above two methods can handle general purpose systems.

4.3.4 *Entropy Ratio*

Gautama et al. (2003) proposed Entropy Ratio (ER), a method based on minimizing the ratio between the entropy of the phase spaces from the original series and its surrogates[4]. The authors realized that a deterministic attractor should have a well-formed structure and, therefore, low entropy. In this scenario, they use the Kozachenko-Leonenko entropy to measure the amount of disorder in the phase space

$$H(\mathbf{\Phi}_i, m, \tau) = \sum_{t=0}^{N_i-1} \ln(N_i R_m^2(t, \tau, 1)) + \ln 2 + 0.5572, \tag{4.40}$$

3 In that article, ϵ was defined in function of the standard deviation of the time series.

4 Due to its relation to entropies, the concept of the ER method was the closest to our hypothesis.

where $R_m^2(t, \tau, 1)$ is the distance of the tth phase space to its nearest neighbors (Equation 4.14).

In attempt to overcome the curse of dimensionality, the method chooses the pair (m, τ) that minimizes the entropy ratio after superimposing the Minimum Description Length (Rissanen, 1978)

$$R_{\text{ent}}(m, \tau) = I(m, \tau) \left(1 + \frac{m \ln N_i}{N_i}\right), \tag{4.41}$$

where

$$I(m, \tau) = \frac{H(\mathbf{\Phi}_i, m, \tau)}{\langle H(\mathbf{\Phi}_{ij}, m, \tau)\rangle_s}, \tag{4.42}$$

and $\langle . \rangle_s$ is the mean of all the s surrogates of T_i, referred to as $T_{ij}, j \in [1, s]$.

Initially creating surrogates based on random permutation of T_i, the authors observed that the ER usually led to a time delay equals to one. To improve this result, they applied the iterative Amplitude Adjusted Fourier Transform (Schreiber and Schmitz, 1996) instead, as this method attempts to preserve both the distribution and the spectrum of the original series, as illustrated in Figure 4.9.

However, according to our own implementation of the ER method, the entropy ratio is still prone to estimate $\tau = 1$ even when using the iAAFT.

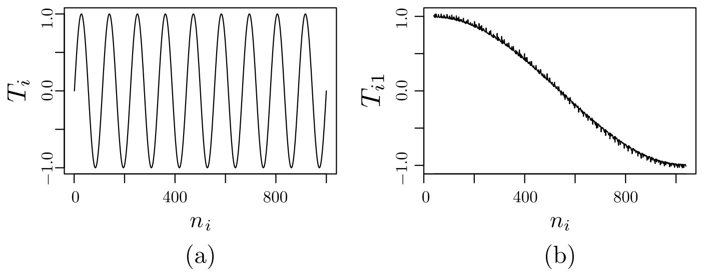


Figure 4.9: ER creates surrogates using the iAATF method, which iteratively creates a new series **(b)** in attempt to preserve the same distribution and spectrum from the original one **(a)**. The example shows the sinusoidal function (Section 3.2.1).

4.3.5 *Non-Biased MACF And Gamma Test*

Although the Γ test (Section 4.2.2.2) was adapted by Otani and Jones (1997) to estimate the optimal embedding dimension m, Ma and Han (2006) noticed that it also could be used to validate τ. In this sense, based on the fact that AD (Section 4.2.1.5) can be applied to reveal correlations on higher-order systems, overcoming restrictions on ACF-based methods[5], the authors initially used it to have a "guesstimation" on the time delay. Nonetheless, the authors later noticed that AD may struggle with inconsistencies while computing the slope of the plateau. To overcome this issue, they proposed the Non-Biased Multiple Autocorrelation Function (NB-MACF)

$$C^m_{xx}(\tau) = R^m_{xx}(\tau) - (m-1)\mu_i^2, \tag{4.43}$$

where μ_i is the mean value of the time series T_i. The algorithm then was divided in three steps based on the interval $m \in [m_{\min}, m_{\max}]$. Firstly, for each dimension, τ is estimated using the NB-MACF. Secondly, the Γ test is used to estimate the corresponding embedding dimension m for the previously found time delay. Finally, the optimal embedding pair is defined as the first one to reach the minimum value of Equation 4.43. By using this approach, Ma and Han (2006) achieved good results for the Hénon map and the Lorenz system. However, this method still needs further validation on other systems.

4.3.6 *Neural Networks*

Taking advantage of Multilayer Perceptron (MLP) (Delashmit and Manry, 2005; Haykin, 2009), Karunasinghe and Liong (2006) proposed to train an MLP network in order to find the best model to predict chaotic time series. Due to chaos, they tried to predict an observation $\rho = 1, 3, 5$ step(s) in the future, so that $\hat{x}(t)$ denotes the predicted value for $x(t)$, given a chosen leap-time ρ. In this context, they implemented s feed-forward, back-propagation sigmoidal MLPs for each leap time and selected the best pair resulting in the smallest prediction error, as illustrated in Figure 4.10. Denoting by μ_i the mean value of the time series T_i with n_i observations, they used the Normalized Root-Mean-Square Error (NRMSQ)

$$\sqrt{\sum_{t=0}^{n_i-1} (x(t) - \hat{x}(t))^2 / \sum_{t=0}^{n_i-1} (x(t) - \mu_i)^2} \tag{4.44}$$

as loss function in order to train the network. Additionally, they reported the Mean-Absolute Error (MAE)

$$\frac{\sum_{t=0}^{n_i-1} |x(t) - \hat{x}(t)|}{n_i} \tag{4.45}$$

5 Although ACF measures linear dependencies between $x(t)$ and $x(t+\tau)$, nothing is inferred on $x(t)$ and $x(t+2\tau)$.

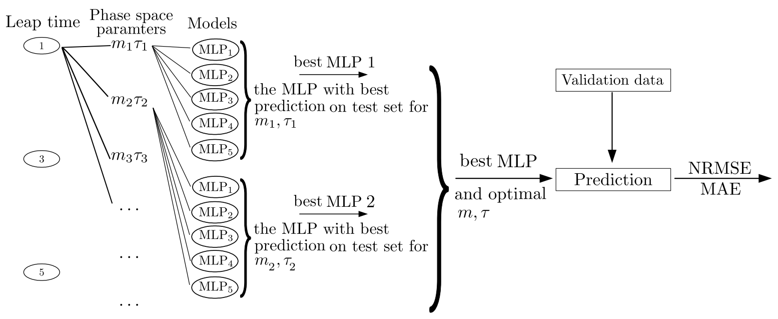


Figure 4.10: The pipeline proposed by Karunasinghe and Liong (2006). Several neural networks were used to find the best set of embedding parameters. Figure adapted from their article.

in their experiments to rely on both global (NRMSE) and local (MAE) measurements. They search space considered varies the range of embedding dimension $m \in [1, 10]$ with combination of fixed values of $\tau = \{1, 3, 6, 9\}$.

As reported by its authors, this method found $(m = 7, \tau = 6)$ for free-noised observations from the Lorenz system, which yielded to better predictions, (according to the NRMSE and MAE errors) when compared to other methods (Farmer and Sidorowich, 1987; Abarbanel et al., 1993). While sufficient, the embedding dimension is known to be overestimated. In addition, three other (usual) drawbacks of neural networks must be mentioned: i) long computational time required to train MLPs (they varied τ over a small set of values in attempt to overcome this issue and used $s = 5$); ii) costly and/or delicate hyperparameter tunning, including the number of epochs, error threshold, and adaptive learning rate. In order to mitigate such issues, after some trial-and-error procedure, the MLPs were defined with the following architecture: m units at the input layer, 100 units at the hidden layer, and 1 unit at the output layer; and (iii) no guarantees that forecasting results are due to the embedding parameters or to other reasons. For instance, the output could vary depending on the random initialization of the layers weights.

Later, Bhardwaj et al. (2010) proposed a similar approach for training a neural network, but using Hidden Markov Models (Meyn and Tweedie, 2009) instead. More recently, Manabe and Chakraborty (2007) estimated m and τ directly from the neural-network architecture after training interactions. Due to its importance and relatively good results (compared to other estimation methods), such article has inspired us when tackling RQ5 (more details in Chapter 9).

4.4 FINAL CONSIDERATIONS

This chapter provided a detailed and, to our knowledge, comprehensive overview of existing methods designed to estimate the optimal embedding dimension m and time delay τ to reconstruct phase spaces based on Takens (1981), either assuming they are independent (Section 4.2) or not (Section 4.3) on each other. For completeness, we should mention that a number of additional methods, not mentioned here, exist in the literature. We did not cover those as they are variants of methods already discussed here; or methods which present less validation, or had otherwise lower impact, than the methods discussed here. For the interested reader, we direct further reading to Rosenstein et al. (1994), who discuss additional approaches to estimate τ; and to Otani and Jones (1997), who mention additional strategies to compute m. A summary of algorithms to find both parameters (either simultaneously or not) is also found in (Buzug and Pfister, 1992b; Ma and Han, 2006).

From this overview, we can draw several conclusions, as follows.
Validation difficulty: It is difficult to define the set of attributes (*e.g.*, homogeneity and distribution of phase states) that generally describes the optimal phase space, especially since this phase space can manifest itself in notorious different ways (as shown in Chapter 3). The computation of metrics based on entropy, correlation dimension or attractor expansion struggle with several inconsistencies related to the partition of the space, computation of probabilities, curse of dimensionality, and the presence of noise. These aspects reinforce the difficulty in defining a set of properties to look at when validating phase spaces. Those issues are the main factors for the lack of robustness in state-of-the-art methods that, despite contributions and good estimations for some datasets, are not adequate to be use in *general* scenarios. Nonetheless, FNN and AMI are still among the most common methods to estimate m and τ, respectively. These difficulties are one of the main reason why we chose a small set of systems, which are well-understood in the literature, and for which consensus exists on optimal embeddings (described in Chapter 3) to compare our work (described in the next chapters) against.

Complexity: This chapter has overviewed different methods to estimate m and τ. All methods are based on extensive sets of heuristics and involve numerous parameters. Some are also computationally expensive. Hence, the quest is still open for designing simple to understand (and use), robust, and fast estimation methods. These requirements are among the main drivers that underlie our work presented in the next chapters.

5 APPLYING A KERNEL FUNCTION ON TIME-DEPENDENT DATA TO PROVIDE SUPERVISED-LEARNING GUARANTEES

5.1 INITIAL CONSIDERATIONS

In Chapter 1, we introduced our hypothesis and subsequently refined it into five research questions (RQ1–RQ5). Chapters 2 to 4 have introduced the fundamentals of dynamical systems, benchmark datasets used for evaluating such systems and their optimal embeddings, and techniques for computing such embeddings, respectively.

In this chapter, we start exploring our first research question:

RQ1. "Does the optimal phase space have indeed low levels of entropy?"

To tackle RQ1, we initially considered to use (and combine) different types of entropies, measured in function of states in the phase space, to propose an optimization problem that once solved would lead to the optimal embedding (Section 5.7). However, as stated in Chapter 1 and further refined in Chapter 4, computations in the phase space are not trivial in practical scenarios, as they depend on several parameters and conditions that may drastically vary according to the attractor, such as how the space is partitioned and the considered open-ball radius for the computation of nearest neighbors, among others.

In order to bypass those limitations, we relied on the correlation the entropy has with the independence among phase states (Myers et al., 1992) – a high-entropy configuration refers to independent states (in phase space), whereas a low-entropy configuration is associated with less independent states – to tackle the problem from a different perspective. However, measuring independence of states is itself complicated. Therefore, instead of estimating the independence of phase states explicitly (which should be great for *optimal embeddings*), we based on the Statistical Learning Theory (SLT) (Vapnik, 1998; Luxburg and Schölkopf, 2011), a mathematical framework that supports learning with supervised classifiers, to measure it intrinsically. Within SLT, a highly-independent set of samples should realize better learning than a highly-dependent set of samples. So, at this point, we propose to estimate independence by looking at the accuracy of a classifier trained and tested on the respective set of samples.

To this end, we test different embedding pairs and chose the one that leads to the best forecasting accuracy under the prediction horizon, computed using the Lyapunov exponents (Section 2.6.3) of its attractor. Next, we provide both theoretical and experimental results based on a cross-validation strategy as evidence of generalization, which allows us to assume that input data is independent. This is an important finding, since SLT can only be used if several assumptions about the input data hold, among which data independence is required (as detailed in Section 5.2).

The structure of this chapter is organized as follows. In Section 5.2, we introduce the Statistical Learning Theory (SLT). Section 5.3 connects SLT with our domain of interest, *i.e.*, dynamical systems. Section 5.4 discusses the issue of data independency. Section 5.5 details on how to forecast time series. Section 8.5 reports experiments that demonstrate how our proposal works. Section 5.7 defines entropy and presents multiple ways to model and compute it, and comments on the correlation of entropy values with (near) optimal embeddings. Finally, Section 5.8 concludes this chapter.

5.2 STATISTICAL LEARNING THEORY

The Statistical Learning Theory (SLT) provides the theoretical foundation to enable machine learning in supervised scenarios (Luxburg and Schölkopf, 2011). This framework considers an input space $\mathcal{X}$ and an output space $\mathcal{Y}$ in which every $x_i \in \mathcal{X}$ corresponds to a data sample (or feature vector) and $y_i \in \mathcal{X}$ is its expected class or label. For simplicity, we next consider that $x_i \in \mathbb{R}$, $y_i \in \mathbb{R}$. SLT assumes a joint probability distribution $P(\mathcal{X} \times \mathcal{Y})$ for all possible combinations of input examples in $\mathcal{X}$ and classes in $\mathcal{Y}$. In this context, *learning* is defined as the process of finding a classifier $f : \mathcal{X} \to \mathcal{Y}$ that provides the minimum risk (error) as possible. The best classifier corresponds to the function that best represents the joint probability distribution $P(\mathcal{X} \times \mathcal{Y})$.

From this perspective, supervised learning requires the definition of a loss function to measure the risk of a classifier $f \in \mathcal{F}$, in which $\mathcal{F}$ is the algorithm bias, *i.e.*, the set of all functions used by the supervised learning algorithm to represent every possible classifier. For example, the bias of Naïve Bayes (Raschka, 2014) is composed of linear and orthogonal hyperplanes, while the Multilayer Perceptron (Haykin, 2009) may have a more complex bias consisting of hyperplanes not necessarily orthogonal to the feature-space dimensions.

The two most common functions used in supervised learning are the $0-1$ loss function and the squared-error function, defined respectively as

$$l(x_i, y_i, f(x_i)) = \begin{cases} 1 & \text{iff } f(x_i) \neq y_i, \\ 0 & \text{otherwise,} \end{cases} \tag{5.1}$$

and

$$l(x_i, y_i, f(x_i)) = (y_i - f(x_i))^2. \tag{5.2}$$

Then, learning algorithms typically adapt the classifier by changing its parameters to minimize such losses, which in practice often corresponds to shifting the decision hyperplanes following the gradient of the loss function (Bergstra et al., 2011).

However, can one guarantee that an algorithm that performs perfectly on training data, *i.e.*, for which the training error is equal to zero, performs similarly on *unseen* examples? SLT aims to answer this question. To this end, Vapnik (1998) proves several properties to ensure the generalization of supervised learning algorithms. These, however, only hold if the following assumptions are respected:

A1. examples must be independent from each other and sampled in an identical manner;

A2. no assumption is made about the joint probability distribution $P(\mathcal{X} \times \mathcal{Y})$, otherwise one could simply estimate its parameters;

A3. labels $y \in \mathcal{Y}$ can assume nondeterministic values due to noise and class overlapping;

A4. the joint probability distribution is fixed over time;

A5. the joint probability distribution is unknown at training time; it must be estimated using training examples.

Assumptions A1 and A4 are mandatory given Vapnik decided to use the Law of Large Numbers to build up the Empirical Risk Minimization Principle (ERMP), which ensures that

$$P(|R(f) - R_{\text{emp}}(f)| > \epsilon) \to 0, \quad n \to \infty, \tag{5.3}$$

where $|\cdot|$ is the absolute-value norm. The remaining assumption A5 gives general purpose to his analysis. As result, Equation 5.3 states that the empirical risk $R_{\text{emp}}(f) \in [0, 1]$ of a classifier converges, probabilistically, to the real risk $R(f) \in [0, 1]$ (also known as expected risk or, simply, risk) as the sample size n goes to infinity. In this context, the empirical risk corresponds to the training error (computed using the training set), defined as

$$R_{\text{emp}}(f) = \frac{1}{n}\sum_{i=1}^{n} l(x_i, y_i, f(x_i)), \tag{5.4}$$

while the (expected) risk is defined as

$$R(f) = E(l(\mathcal{X}, \mathcal{Y}, f(\mathcal{X}))), \tag{5.5}$$

with $E(\cdot)$ as the expected value.

The motivation for Equation 5.3 came from the definition of generalization as $|R(f) - R_{\text{emp}}(f)|$, which means a classifier f is expected to provide a similar risk (error) over training data *and* unseen examples, otherwise it overfits and, therefore, it does not model the target problem. Note that a classifier with good generalization is not necessarily the one that provides a low risk but the one in which the empirical risk (training error) is a good estimator for the expected risk (over all possible inputs, including unseen ones). Finally, learning only happens when f generalizes and its empirical risk $R_{\text{emp}}(f)$ is small enough according to some problem-specific criterion.

Vapnik observed the need to define an upper limit for such probability in order to ensure learning for *any* supervised algorithm. He proved the following bound to be valid

$$P(\sup_{f\in\mathcal{F}} |R(f) - R_{\text{emp}}(f)| > \epsilon) \leq 2\mathcal{N}(\mathcal{F}, 2n)e^{-n\epsilon^2/4}, \tag{5.6}$$

which considers the worst-to-best convergence scenario as the sample size $n \to \infty$, if and only if the algorithm bias $\mathcal{F}$ is characterized by a polynomial shattering coefficient $\mathcal{N}(\mathcal{F}, 2n)$. This coefficient, or function, must grow polynomially, otherwise learning is not guaranteed according to ERMP (Equation 5.3). This is one of the most important formal steps for machine learning, as it ensures the learning conditions for supervised algorithms.

Following the above theoretical foundation, Equation 5.6 can be rewritten as

$$P(\sup_{f\in\mathcal{F}} |R(f) - R_{\text{emp}}(f)| > \epsilon) \leq \delta, \tag{5.7}$$

where $\delta = 2\mathcal{N}(\mathcal{F}, 2n)e^{-n\epsilon^2/4}$, thereby defining the probability P for which both risks do not diverge more than ϵ measurement units, *i.e.*, the empirical risk approximates the expected one. Solving the above for ϵ yields

$$\epsilon = \sqrt{\frac{4}{n}\left(\log \mathcal{N}(\mathcal{F}, 2n) - \log \delta\right)}. \tag{5.8}$$

This gives the divergence between the empirical and the expected risks as

$$R(f) \leq R_{\text{emp}}(f) + \epsilon, \tag{5.9}$$

which is also referred to as the Generalization Bound. The main goal of any supervised learning algorithm is to obtain the smallest combination of empirical risk and ϵ, so that the expected risk will also be small and one will obtain the optimal classification performance.

5.3 CONNECTING SLT AND DYNAMICAL SYSTEMS

According to Bayes' theorem (Raschka, 2014), every data sample $x \in \mathcal{X}$ must have meaningful attribute values to estimate the output class $y \in \mathcal{Y}$. Hence, input attributes must present some dependency level on the expected outcome (Bishop, 2006). In our context, such dependence is revealed after applying a kernel function (*e.g.*, Takens' embedding theorem) on the time series to unfold the phase space. In this setting, states intrinsically represent the temporal dependencies of observations, *but they do not depend on each other*[1]. More precisely, if the phase space is reconstructed using the embedding pair (m, τ), then for each state $\boldsymbol{\phi}_i(t), \forall t \in [0, N_i] - 1$, the pair $(\{x(t), \cdots, x(t + (m-2)\tau)\} = \boldsymbol{x}_t \in \mathcal{X}, x(t + (m-1)\tau) = y_t \in \mathcal{Y})$ (Equation 4.18) contains such a dependency. For illustration purposes, Table 1 this dependency for the Logistic map phase space.

Table 1: Time-series observations organized in a tabular form after applying the kernel function. For simplicity, a phase space given by the embedding parameters $m = 2$ and $\tau = 1$ is considered.

State	$\mathcal{X}$	$\mathcal{Y}$
$(x(t), x(t+1))$	$x(t)$	$x(t+1)$
$(x(t+1), x(t+2))$	$x(t+1)$	$x(t+2)$
$(x(t+2), x(t+3))$	$x(t+2)$	$x(t+3)$
$(x(t+3), x(t+4))$	$x(t+3)$	$x(t+4)$
$(x(t+4), x(t+5))$	$x(t+4)$	$x(t+5)$
$\vdots$	$\vdots$	$\vdots$

The important consequence of the above is that one can uniformly sample phase states in any order to infer some classification

1 We remind the reader that deterministic systems define a unique state in the future.

or regression model, a process which can be seen as a selection of points under the same probability distribution (Carlsson and Mémoli, 2013). Moreover, if we have enough states to represent the attractor, we also ensure they are identically distributed into its structure, as illustrated in Figure 5.1).

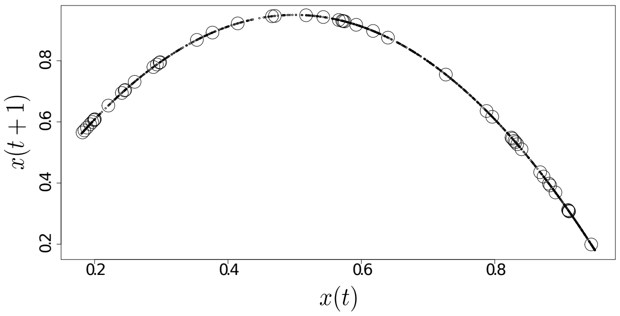


Figure 5.1: Uniform sampling of states in the Logistic map phase space. Circles illustrate regions respective states (in the center of each circle) were sampled from. Adapted from (Pagliosa and de Mello, 2017).

In the above context, assumptions A1 and A4 of SLT can be satisfied after applying such a kernel function, once phase states are now seen as i.i.d. data inputs. Assumptions A2, A3, and A5 are straightforwardly fulfilled. If we assume we are dealing with samples coming from a single phenomenon, the joint probability distribution is time-independent, which satisfies A4. Chapter 7 addresss separately the case of concept drift scenarios (Gama et al., 2014) in which time-series samples may derive from different phenomena (with distinct joint probability functions).

Since we now have satisfied all prerequisites needed to treat our problem in a machine learning setting, we can perform *regressions* on the reconstructed phase space. For this, we use a simple Distance-Weighted Nearest Neighbors (DWNN) supervised learning algorithm, which builds a classifier $f : \mathcal{X} \to \mathcal{Y}$ in form

$$f(\boldsymbol{x}_{\text{new}}) = \frac{\sum_{t=0}^{N_i-1} w_t y_t}{\sum_{t=0}^{N_i-1} w_t}, \tag{5.10}$$

in which w_t is the Gaussian-based distance between the new state $\boldsymbol{x}_{\text{new}}$ and the known sample $\boldsymbol{x}_t$, defined as

$$w_t = e^{-|\boldsymbol{x}_{\text{new}} - \boldsymbol{x}_t|_2^2 / 2\sigma^2}, \quad \forall t, \tag{5.11}$$

where σ is the dispersion of the radial-basis activation function. This function produces values in $[0, 1]$, having 1 when $\boldsymbol{x}_t$ is at the same space location as $\boldsymbol{x}_{\text{new}}$ and 0 when they are very far from each other, *i.e.*, σ is too small to include $\boldsymbol{x}_t$ in the neighborhood of $\boldsymbol{x}_{\text{new}}$.

We next analyze the behavior of DWNN with respect to the value of σ. By setting $\sigma \to 0^+$, we have the smallest open ball (Mendelson, 1975) around the new state that contains no other neighbor but itself, so Equation 5.11 yields 0 for all examples. Consequently, we have that

$$f(\boldsymbol{x}_{\text{new}}) = \frac{\sum_{t=0}^{N_i-1} w_t y_t}{\sum_{t=0}^{N_i-1} w_t} = \frac{0}{0}, \tag{5.12}$$

which characterizes an undefined situation. Hence, there is no learning guarantee in this scenario. This is the situation in which the classifier learns to handle only the training samples (Luxburg and Schölkopf, 2011), or, in other words, it *memorizes* outputs and only reports correct label when $\boldsymbol{x}_{\text{new}} = \boldsymbol{x}_t$, for all t. This classifier would never generalize to further unseen data, which represents a typical case of overfitting.

On the other hand, when $\sigma \to +\infty$, Equation 5.11 tends to 1 for all samples, we have

$$f(\boldsymbol{x}_{\text{new}}) = \frac{\sum_{t=0}^{N_i-1} w_t y_t}{\sum_{t=0}^{N_i-1} w_t} = \frac{\sum_{t=0}^{N_i-1} 1 y_t}{\sum_{t=0}^{N_i-1} 1} \approx \mu_{y_t \in \mathcal{Y}}, \tag{5.13}$$

so our classifier f always outputs the average class referred to as $\mu_{y_t \in \mathcal{Y}}$. This is the situation where the classifier underfits data, leading to poor regressions. For the sake of illustration, given a binary classifier, this is equivalent to flipping a coin in attempt to hit its output.

We conclude there is a need for a balance between $\mathcal{F} = \mathcal{F}_{\text{all}}$ and $|\mathcal{F}| = 1$ to model a target problem adequately (both extremes are illustrated in Figure 5.2). This trade-off is also known as Bias-Variance Dilemma in Machine Learning literature (Geman et al., 1992; Luxburg and Schölkopf, 2011). For DWNN, the bias is mainly described by the value of σ. Finding a good stationary point between under and overfitting should be in agreement with the Statistical Learning Theory (Luxburg and Schölkopf, 2011). Separately, we should set σ large enough to contain sufficient neighbor states for every unseen sample, so that the classifier is capable of extrapolating from these to the unseen points.

5.4 ON THE KERNEL FUNCTION TO DEAL WITH DATA DEPENDENCIES

Takens' embedding theorem is a dynamical system tool to reconstruct data from the temporal space into the phase space. If this reconstruction is adequate, we assume states in the phase

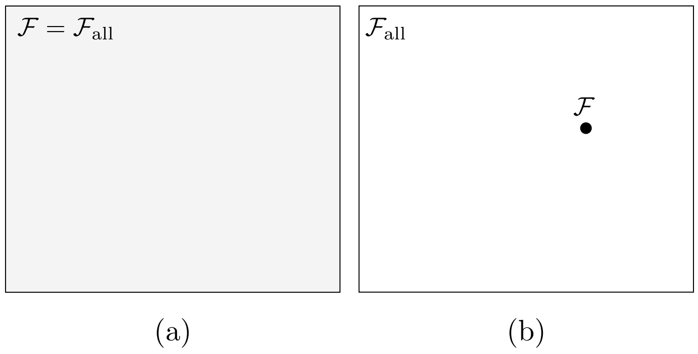

$\mathcal{F} = \mathcal{F}_{\text{all}}$
$\mathcal{F}_{\text{all}}$
$\mathcal{F}$
(a)
(b)

Figure 5.2: DWNN biases: **(a)** the space contains every possible classifier in $\mathcal{F}_{\text{all}}$ as $\sigma \to 0^+$. ii) **(b)** the space contains only one classifier when $\sigma \to +\infty$.

space to be independent on each other, as they can be sampled in an independent-and-identically-distributed (i.i.d.) manner (see Figure 5.1). Then, to evaluate the independence of points in the phase space, we relied on the SLT framework as it ensures learning only when training-data samples are independent from each other.

To this end, we proceed as follows. We conducted several experiments to produce an extensive set of phase spaces, using Monte Carlo simulation (Landau and Binder, 2005) for the embedding parameters, which were evaluated in terms of their ability to build regressive functions (those behave in the same manner as discrete-based label classifiers). For each embedding, we have selected random states in the phase space (with replacement to keep data distribution unchanged, as discussed in (Luxburg and Schölkopf, 2011)) to build learning models using the DWNN supervised algorithm. We chose this method as it holds all properties of SLT and, therefore, it is guaranteed to learn when σ does not overfit (when training samples are represented) nor underfit (when all training samples are considered simultaneously). In addition, DWNN allows to change its bias by adapting σ, and measure its influence in terms of training and test errors to confirm the occurrence of both cases.

We define the *ideal independent phase space* as the embedding for which DWNN learns the most while respecting the Generalization Bound (Equation 5.9). In other words, we have chosen the set of kernel function parameters – namely the embedded dimension m, time delay τ, and radius σ – that minimized both the empirical risk (training error) and ϵ to ensure $R_{\text{emp}}(f)$ is a good estimator of $R(f)$. After defining such embedding, we verify if this space is indeed capable of generalizing for unseen data in practical scenarios, which allows us to confirm whether over or underfitting have happened. To prove that, we use a ghost sample as supported by the symmetrization lemma (Vapnik, 1998)

$$\begin{aligned} &P(\sup_{f \in \mathcal{F}} |R(f) - R_{\text{emp}}(f)| > \epsilon) \leq \\ &2P(\sup_{f \in \mathcal{F}} |R_{\text{emp}}(f) - R'_{\text{emp}}(f)| > \epsilon/2) \leq \delta \end{aligned} \tag{5.14}$$

having $R'_{\text{emp}}(f)$ as the empirical risk computed for the ghost sample.

This lemma states that if two independent samples have empirical risks that do not diverge more than ϵ, then it is expected that the empirical risk is also a good estimator for the expected risk. To employ the symmetrization lemma, we use the recently-computed optimal parameters to embed two independent, randomly-sampled subseries from T_i. Moreover, in order to be in accordance with ERMP, we simulate the growing of the input space (*i.e.*, $n \to \infty$) by equally increasing the length of both sets, starting with $10 \times H$ (H is the prediction horizon (de Mello and Yang, 2009) estimated using the Lyapunov exponent as defined in Equation 2.31) up to the total number of $n_i/2$ observations. Finally, we compute using DWNN, for each round, the respective empirical risks for both series, representing $R_{\text{emp}}(f)$ and $R'_{\text{emp}}(f)$, respectively.

Thus, we validate the choice of the optimal embedding by taking the difference between both empirical risks. If this difference is small, then a small ϵ satisfies Equation 5.14 and, based on the Generalization Bound (Equation 5.9), the expected risk $R(f)$ is also small. Besides, if Equation 5.14 is held for most cases, the shattering coefficient $\mathcal{N}(\mathcal{F}, 2n)$ grows polynomially along n, which improves consistency for the learning algorithm (Vapnik, 1998). This aspect ensures learning, providing empirical evidence that ideal phase spaces contain independent points. This is a very important step towards supervised learning when it comes to time-dependent data.

In our experiments, we show that when the ideal (*i.e.*, the best setting for m and τ) phase space is found after applying Takens' embedding theorem as kernel function, the supervised learning algorithm models unseen data with significant low errors. Those experiments considered time series produced by synthetic systems. Consequently, we know their generation rule (*i.e.*, the equation $R(\cdot)$ used to generate data) and the most common setting for the embedding dimension m and the time delay τ to reconstruct phase spaces (the ideal kernel function parametrization). Therefore, we can compare other possible reconstructions against the mostly used ones and measure their generalization capabilities (prediction of unseen observations). In this manner, the analysis of synthetic data allows to take conclusions based on well-known expectations. In addition, we also considered the Sunspots dataset (Andrews and Herzberg, 1985) to illustrate how our approach addresses a real-world scenario.

5.5 CONCRETE EXAMPLE: PREDICTING TIME SERIES

As discussed earlier in Section 5.2, the risk of a classifier f for any supervised algorithm is defined using a loss function $l(x_i, y_i, f(x_i))$, in such a way that the best classifier achieves the smallest expected risk $R(f)$. Moreover, such a loss function should be defined accordingly to the problem goal, space, and restrictions. In the context of time-series modeling, the $0-1$ loss function (Equation 5.1) and the squared-error one (Equation 5.2) may be inadequate to compute the error (the difference between two series) since time-shifted observations can still preserve the nature of the series. To improve comparison, we used instead the Mean Distance from the Diagonal Line (MDDL) (Rios, 2013). MDDL uses Dynamic Time Warping (DTW) to ideally match expected-to-forecasted observations, which produces a path in the two-dimensional space having each series along one axis (Figure 5.3(a)). Next, MDDL calculates the average distance from each path to the diagonal line (Figure 5.3(b)), providing a final distance estimation.

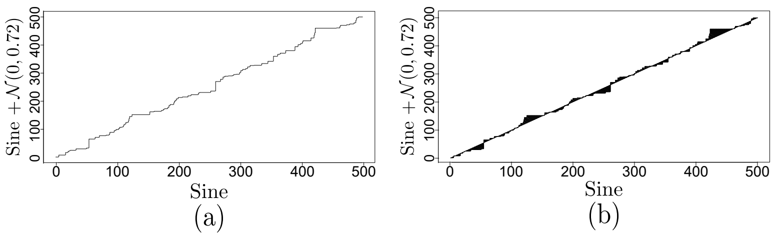


Figure 5.3: **(a)** DTW warping path between series. **(b)** MDDL computes the cumulative distances from the warping path to the diagonal line (shaded). Adapted from (Pagliosa and de Mello, 2017).

An advantage of MDDL is that it summarizes the distance between series considering time displacements (such as time-series trends), improving upon both the traditional Euclidean distance and DTW. For instance, Figure 5.4 illustrates a sinusoidal function with a stochastic process defined by a Normal probability distribution $\mathcal{N}(0, 0.72^2)$ (with 0 mean and 0.72 standard deviation[2]) added to it, and two possible forecasted series: i) the mean-valued series,

2 Usually, the Normal distribution is represented by $\mathcal{N}(\mu, \sigma^2)$. However, most of our routines in R simulating data distribution received as input the standard deviation. Thus, we decided to report, from now on, Normal distributions highlighting the standard deviation, *i.e.*, explicitly showing it in the form σ^2.

and ii) the noise-free series. When considering the DTW distance, the mean-valued series would be taken as more similar (normalized distances to (i) and (ii) are 0.31159 and 0.30894, respectively), which implies DTW does not take into account trend components (very important when analyzing time series). In contrast, MDDL would indicate the sinusoidal series as the most adequate solution: distances are 5.84667 and 170.65778 for (i) and (ii), respectively. It is fair to mention that DTW-D (ratio between the DTW and the Euclidean distance) could also have been used in this scenario.

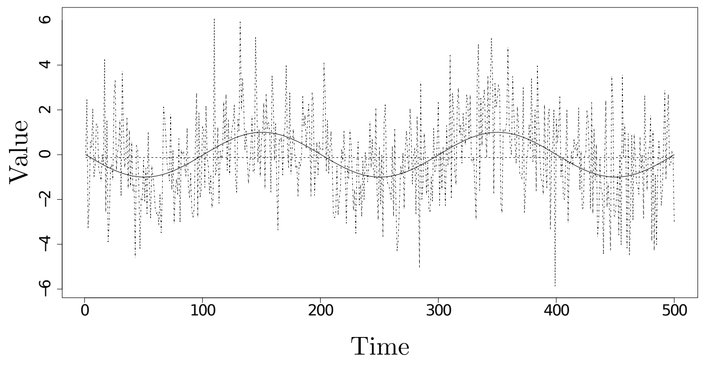


Figure 5.4: Two forecasted series are compared against the expected sinusoidal series with noise added to it (represented by the dashed line with sinusoidal form): the noise-free series (solid line) and the mean-valued one (straight-dashed line). Adapted from (Pagliosa and de Mello, 2017).

We then consider MDDL as loss function to propose the k-Fold Time Cross Validation (kFTCV), a novel approach to validate time-series classification and forecast time series. Given a time series T_i and its phase space $\mathbf{\Phi}_i$, the algorithm splits T_i into k subseries as $T_i = \{S_1, S_2, \cdots, S_k\}$ (we assume $k = 10$ in our experiments), then it uses one of those subseries S_j as expected data (test samples) and the remaining observations to infer a classifier (training samples), as illustrated in Figure 5.5.

To predict new observations and compare against one of the folds, we perform the following steps: i) we use DWNN to build up a regression function with 90% (because $k = 10$) of the points in phase space; ii) we predict the first instance for the remaining 10% of points (S_j); iii) we employ the newly predicted value to recursively forecast next observations (note that the error will accumulate along predictions); iv) we repeat step (iii) h times. Due the chaotic behavior, h assumes the prediction horizon $h = H$ for each series, or any constant $h \leq H$. Finally, the forecasted series is compared to the expected one using MDDL, and the average MDDL value over all folds is used as empirical risk.

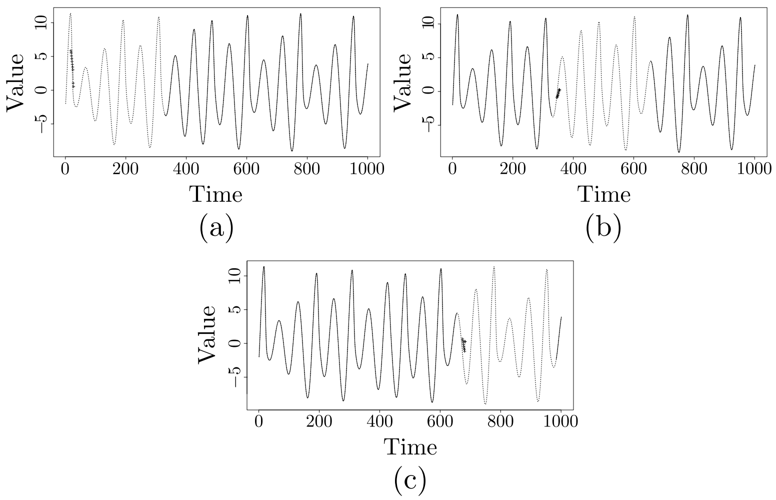


Figure 5.5: Execution of 3-Fold Time Cross Validation on our system. For each fold (dashed line in **(a)**–**(c)**), the algorithm trains with the remaining series (solid line) to recursively predict H observations (crossed line). Notice that by increasing the number of folds, the learning confidence also increases. Adapted from (Pagliosa and de Mello, 2017).

The best set of embedding parameters is defined as the one whose respective phase space provides the smallest risk. In other words, we select the embedding dimension m and time delay τ (*i.e.*, the kernel function parametrizations) that best unfold the phase space and transform time-series observations into i.i.d. instances. Lastly, we confirm the true risk to be also small in terms of the symmetrization lemma (Equation 5.14).

5.6 EXPERIMENTS

The Distance-Weighted Nearest Neighbors (DWNN) algorithm, a variation of the k-nearest neighbors, was used to perform experiments because it holds, for an adequate σ, the principles established by the Statistical Learning Theory (Luxburg and Schölkopf, 2011). Two different types of experiments were conducted. The first one assessed different phase-space reconstructions (kernel function parametrizations) for synthetic and real-world data, and then we proceeded by selecting the space producing the best classifier, *i.e.*, with the greatest generalization capacity given by $|R(f) - R_{\text{emp}}(f)| \approx 0$. In the second one, we used the recently above-found kernel parametrization to quantify the accumulated error and evaluate the generalization capacity in terms of recurrent forecasting.

Our assumption is that if:

i) time-series dependencies are intrinsically defined in terms of every data sample (phase state);

ii) data samples can be uniformly selected to compose a training set in order to infer a classification or regression model;

iii) we obtain good enough results, measured by cross-validation for training and test sets,

then we can conclude that the kernel function produced an immersion in a multidimensional space which is capable of translating observations into independent-and identically-distributed (i.i.d.) examples. As a consequence, a supervised learning algorithm, such as DWNN, is capable of modeling uniformly-sampled states from the phase space and, consequently, predict unseen observations (which are associated to class labels).

This section is structured as follows. First, we describe the setup of the entire experiment, including algorithmic choices and parameter settings (Section 5.6.1). Next, we describe the first set of experiments, aimed at selecting the optimal phase-space embedding for a given time series, from both synthetic and real-world data (Section 5.6.2). Finally, we use this optimal embedding to show how the learned model can be used to generalize, *i.e.*, predict, or forecast, time series data (Section 5.6.3).

5.6.1 *Experimental Setup*

To avoid underfitting and overfitting, we set σ as the minimal pairwise distance containing an average percentage of nearby states in state space. In order to define the best percentage, 20 values equally spaced from 0.001 to 0.2 were processed by a Monte Carlo simulation. In addition, to break eventual time dependencies, we randomized the training examples and applied the 10-Fold Time Cross Validation (Section 5.5) to compute the empirical risk. We also made sure every σ was sufficiently large to include at least one neighbor, avoiding a memory-based, overfitting, classifier.

We penalized high embedding dimensions and time delays to simplify (in the parsimony sense of Occam's razor (Rasmussen and Ghahramani, 2001)) as much as possible the phase space. We then selected classifiers (regression functions) by assessing the risk computed by MDDL as described in Section 5.5. We look for the best embedding while varying m and τ in range $[2, 15]$ and $[1, 15]$, respectively. However, we only changed the embedding parameters

m and τ when this risk was smaller than the current (best) risk according to a threshold (here defined as 5×10^{-2} when increasing space dimensions and 1×10^{-2} when increasing time delays). In order to estimate a fair upper limit for the expected risk, we defined ϵ as the 20% quantile of all absolute differences of two training sets (as we increased the input length n) in an attempt to hold Equation 5.14 and, therefore, obtain a polynomial shattering coefficient $\mathcal{N}(\mathcal{F}, 2n)$.

We also defined a constant for the number of forecasted observations h, which is justified by: i) the fact that Lyapunov exponent estimators (Kantz, 1994; Rosenstein et al., 1993) might overestimate the prediction horizon; ii) when $h \leq H$, we have a higher confidence in forecasting (less accumulated errors); iii) to avoid $h > H$, and consequently lead to poor results, we defined $h = 10$ during the training stage (to estimate the best classifier), which is expected not to surpass H for most cases.

The final design decision regards the selection of the Relevant Neighbors ($\mathcal{R}$), which we explain next. When learning, DWNN requires the definition of pointwise neighborhoods in order to estimate a class for unseen examples. This process involves the weighted mean for all neighbors under the influence of a Gaussian activation function (Equation 5.11) with standard deviation σ. Thus, even after narrowing σ, distant neighbors may still have more influence that they should have. For instance, suppose a state having two nearest neighbors equally far away; DWNN will assign to this state the mean class of both neighbors, no matter how far they are. This is inadequate in our scenario in which labels are real values. In such situations, we have experimentally concluded that such a point with no relevant neighbors should be discarded.

Our experiments suggest that $\mathcal{R}$ can be smaller for conservative, deterministic, and dense embeddings, whereas it should be greater for spread-out and noisy phase spaces. Although we have achieved good results, we reinforce that each state (or group of states) should have its (or their) own parameter(s) $\mathcal{R}$, considering a phase space can be formed by local structures. We have relied our selection for $\mathcal{R}$ on the fact that the Gaussian activation function can be interpreted as a Normal probability distribution, thus, by setting $\mathcal{R} = 1.5$ (*i.e.*, setting the distance for relative neighbors to be as far as $1.5 \times \sigma$), the closest states are considered relevant if and only if they are at most 1.5 standard deviations from the mean (the mean is defined by the position of the unseen sample under analysis). According to the z-table (Schefler, 1988), this setting takes an area of 86% of data into account (Figure 5.6). Interestingly, a similar discussion appears when selecting the so-called perplexity parameter when reducing data dimensionality using the t-SNE projection algorithm (van der Maaten and Hinton, 2008). Globally put, the selection of a "good" perplexity value (which should consider only a given range of nearest neighbors for a high-dimensional point) is conceptually similar to our discussion of selecting a "good" σ value for our classification purposes.

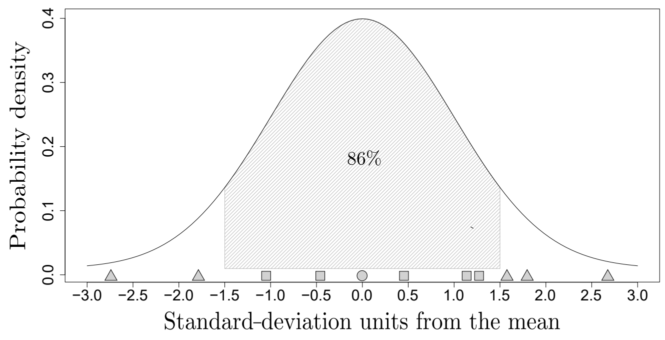


Figure 5.6: Even after mitigating the influence of far states by decreasing σ in Equation 5.11, distant neighbors (triangles) can still jeopardize the classification of unseen examples (circle) when using DWNN. In this form, only relevant neighbors (squares) should be taken into account. Adapted from (Pagliosa and de Mello, 2017).

5.6.2 *Assessing Phase-Space Reconstruction*

In the first set of experiments, we assessed different phase-space reconstructions for synthetic and real-world data. Next, we proceeded with the selection of the space $\mathbf{\Phi}_i$ that produces the most generalizing classifier. Before showing our results, it is worth to remind that the discrete nature of maps like Logistic, Ikeda and Hénon imposes an ideal phase-space reconstruction. Consequently, the embedding dimension m and time delay τ are well-known. On the other side, the continuous nature of systems such as Lorenz and Rössler determines different kernel function parametrizations depending on the rate of data sampling. For these systems, we use a default sampling rate of $t_s = 0.01$.

5.6.2.1 *Synthetic Time Series*

Table 2 summarizes all relevant results from kFTCV. Columns describe, from left to right: the time series T_i; embedding dimension m; time delay τ; σ used in DWNN; relevant neighbors $\mathcal{R}$; empirical risk $R_{\text{emp}}(f)$; divergence parameter ϵ; and the upper limit for the expected risk $R(f)$ (namely $R(f)^*$), computed as described in Equation 5.9 for the best-learning phase space $\mathbf{\Phi}_i$.

Table 2: Using the Statistical Learning Theory to find the best embedding parameters for the kernel function (Takens' immersion theorem).

T_i	m	τ	σ	$\mathcal{R}$	$R_{\text{emp}}(f)$	ϵ	$R(f)^*$
Logistic	2	1	0.00318	1.96	0.13719	0.02734	0.16
Ikeda	3	1	0.13277	1.3	0.60625	0.03432	0.64
Hénon	4	1	0.07924	1.5	0.36391	0.04017	0.4
Lorenz	3	8	0.81707	4	1.30174	0.02440	1.32
Rössler	3	8	0.26946	1.5	0.44782	0.03017	0.47
Sinusiodal	2	1	0.00001	0.5	0	0.04143	0.04
Sunspots	2	4	1.52	1.5	1.15081	0.03577	1.18

According to Table 2, we observe that the parameters found after the phase-space assessments match the commonly used ones (ground-truth parameters) in almost all cases. For clarity, let us mention that researchers typically use $m = 2$ for the Ikeda and Hénon maps only for visualization purposes; while the real embedding dimensions for them are $m = 3$ and $m = 4$, respectively.

Moreover, it is worth mentioning that if we had proceeded without recursive forecasting, different embeddings would have been found. This is due to the fact one-step prediction creates a bias that forces the algorithm to (usually) estimate $\tau = 1$ (which is indeed enough to predict a next but not a recursive sequence of observations).

5.6.2.2 *Synthetic Time Series With Noise Added*

In an attempt to correlate noise components with the time delay, we have added signals produced by two Normal distributions $\mathcal{N}(0, 0.05^2)$ and $\mathcal{N}(0, 0.1^2)$ to the data produced using the sinusoidal function, whose embeddings with $m = 2$ and $\tau = 1$ are illustrated in Figure 5.7. As expected, noise wrongly correlates phase-space trajectories, leading to poor learning results. This situation is overcome when time delay τ is increased, stretching the phase space and deviating embedding points. Nonetheless, the embedding dimension m is also affected, as more dimensions are necessary to stabilize false nearest neighbors (Kennel et al., 1992). As a conclusion, we state that supervised algorithms tend to find more complex embeddings when dealing with noisy observations.

Figure 5.8 illustrates the phase space with the smallest empirical risk for each case, having the pair $(m = 2, \tau = 9)$ for both cases in Figure 5.8. This scenario illustrates how even a small amount

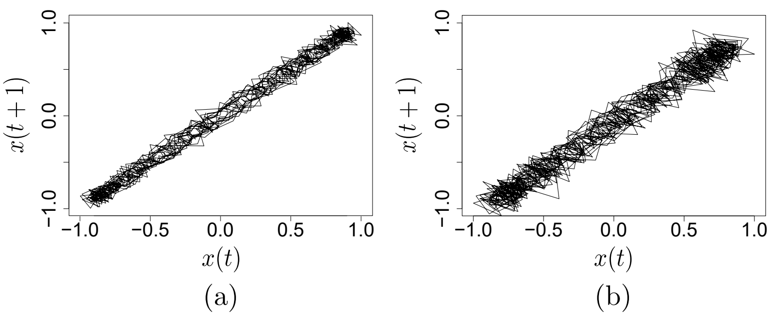


Figure 5.7: Embedding of sinusoidal series with $\mathcal{N}(0, 0.05^2)$ **(a)** and $\mathcal{N}(0, 0.1^2)$ **(b)** added to it. Forced correlated (seen as misplaced) points misleads DWNN classification. Adapted from (Pagliosa and de Mello, 2017).

of noise significantly changes the ideal embedding ($m = 2, \tau = 1$) (de Mello and Yang, 2009).

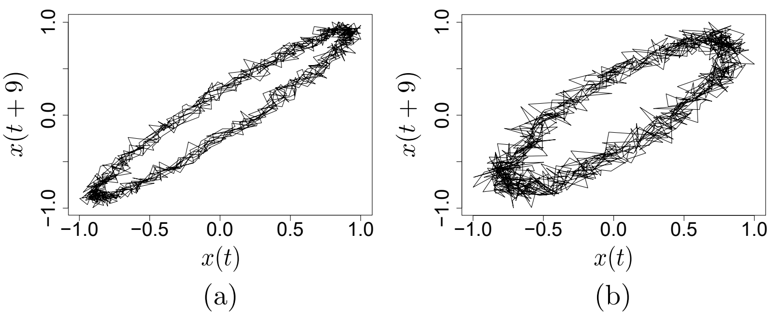


Figure 5.8: In order to learn, DWNN demands a more complex phase space when analyzing noisy data. For the sinusoidal series with $\mathcal{N}(0, 0.05^2)$ and $\mathcal{N}(0, 0.1^2)$ noise added, the best found phase space-reconstruction parameters were $(m = 2, \tau = 9)$ in both cases. Adapted from (Pagliosa and de Mello, 2017).

5.6.2.3 *Real-World Data*

Let us now highlight the importance of the number of relevant neighbors ($\mathcal{R}$) and how this parameter influences our analysis. For real-world scenarios, the area under the Gaussian activation function (Equation 5.11), used to define the relevant neighbors, may vary depending on the phase-space behavior. Different aspects must be considered: i) if phase-space points are affected by noise, decreasing $\mathcal{R}$ may be the best strategy so that only closest points are considered by DWNN; however, ii) if the phase space is formed by spread points, $\mathcal{R}$ should be large enough to involve distant neighbors.

In addition, when predominately deterministic, dense, and continuous phase spaces are analyzed, more points could be taken as neighbors to build the classifier. Take the Logistic map as example. Its phase space is very smooth, forming a well-structured attractor. This structure allows us to set DWNN to consider more neighbors in such a space. In fact, better results were observed when we set $\mathcal{R} = 1.96\sigma$ (*i.e.*, covering 95% of neighbors). On the other hand, we defined $\mathcal{R} = 4$ for the Rössler system (see Figure 5.9). Although this space has a well-defined structure, some regions must take more neighbors into account to correctly classify points due to the local point density. Therefore, we conclude that $\mathcal{R}$ should be defined *locally* on every point or on attractor regions.

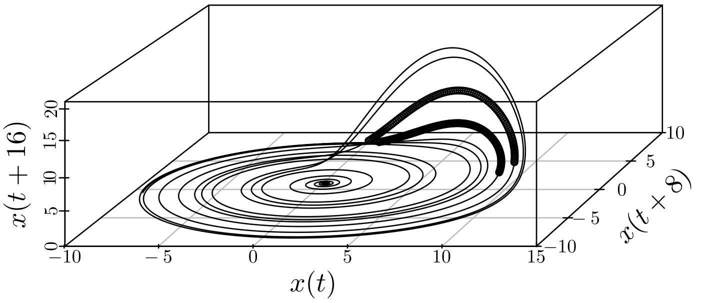


Figure 5.9: Rössler three-dimensional phase space. Despite its deterministic structure, regions in which points are located more apart from each other (darker-thicker lines) force greater values for $\mathcal{R}$, so distant neighbors can be correctly involved during classification. Therefore, different values of $\mathcal{R}$ should be assign to different points, depending on their locations and attractor neighborhoods. Adapted from (Pagliosa and de Mello, 2017).

As we know, the Sunspot dataset was produced by a natural phenomenon, so we expect that data-measurement errors and unexpected behavior may occur. In addition, we also know this data behaves similarly to a sinusoidal function with noise added to it. As a consequence, we assumed $\mathcal{R} = 1.5$ to form pointwise neighborhoods, which allows us to find embedding dimension $m = 2$ and time delay $\tau = 4$. We believe that this is a good approximation considering that this dataset contains sinusoidal influences (the sinusoidal function is typically embedded using $(m = 2, \tau = 1)$. As previously discussed, real-world scenarios lead to more complex phase spaces (a greater value for τ, in this case), typically due to the presence of noise.

5.6.3 *Evaluating The Generalization Capacity When Forecasting*

We now measure the generalization capacity of the found embeddings (obtained as described in Section 5.6.2). This allows us to assess whether learning has indeed happened as desired. For this, we computed MDDL for different training samples with 60%, 70%, 80% and 90% of the total number (n_i) of observations from T_i. For the sake of brevity, we only illustrate the last scenario, *i.e.*, training with 90% of data, as the other cases yielded similar results. Figure 5.10 shows the MDDL average result for all training set sizes.

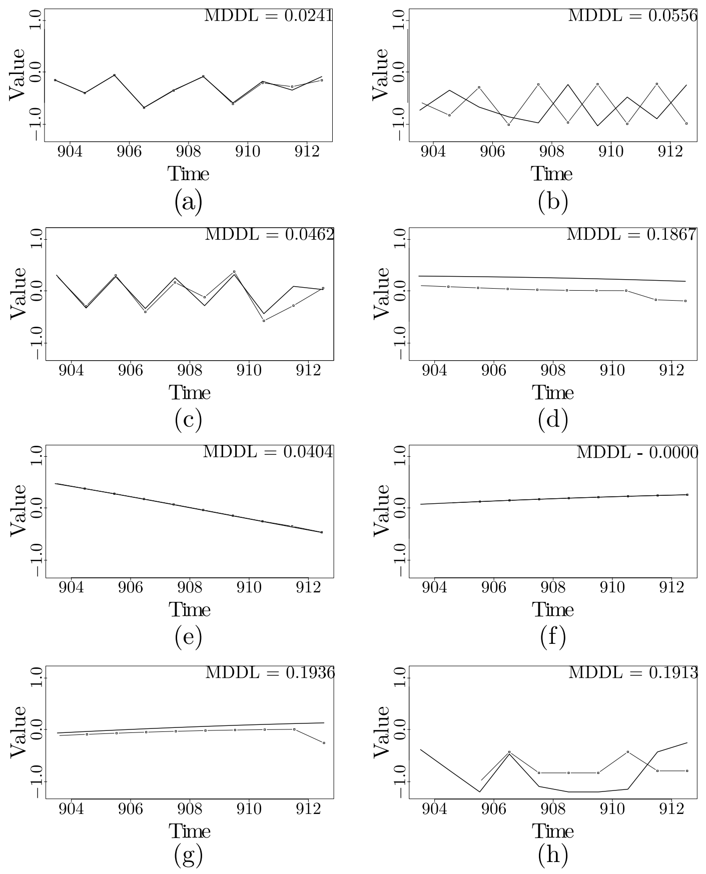


Figure 5.10: Forecasted (dotted lines) and expected (solid lines) observations. From **(a)** to **(h)**: Logistic, Ikeda, Hénon, Lorenz, Rössler, sinusoidal, sinusoidal with $\mathcal{N}(0, 0.05^2)$ noise added, Sunspot. In all plots, observation values were normalized to $[-1, 1]$. Adapted from (Pagliosa and de Mello, 2017).

As expected, all results (estimators of the real risk) were smaller than the upper limit bound given by Table 2. This strongly indicates all phase spaces provided a good generalization capacity for the supervised learning algorithm and, therefore, DWNN was capable of learning. Also, it is worth mentioning that results became worse as time elapses due to the error accumulated during the recursive forecasting (something expected and measurable using the Lyapunov exponent). However, when testing with other values for parameters m, τ, σ and $\mathcal{R}$, we observed generalization was jeopardized. This indicates that the method we proposed earlier in Section 5.6.2 to find optimal parameter values is indeed reliable.

5.7 ENTROPIES AND PROBABILITIES

For completeness, let us turn back to the initial point mentioned in Section 5.1, *i.e.*, directly estimating the entropy. As it was stated, doing this is complicated. In this section, we detail on this matter and show some of the difficulties in deterministically correlating such measurement with optimal embeddings.

Entropy can be measured in several ways. Among the most common methods, Shannon's entropy (Hammer et al., 2000) is defined as

$$E_{\text{sh}} = -\sum_{j}^{n} P_j \log P_j, \tag{5.15}$$

where P_j is a probability function. In the context of phase-space reconstruction, Equation 5.15 can be translated to

$$E_{\text{sh}}(\boldsymbol{\phi}_i) = -\sum_{t=0}^{N_i-1} P(\boldsymbol{\phi}_i(t)) \log P(\boldsymbol{\phi}_i(t)), \tag{5.16}$$

where $P(\boldsymbol{\phi}_i(t))$ is a probability function over each phase state $\boldsymbol{\phi}_i(t)$. Such probability can also be measured in different ways. Among alternatives, we initially estimate $P(\boldsymbol{\phi}_i(t))$ throughout the complex network given by the graph $G(V, E)$ (Newman, 2003), where V is the set of vertices representing each state, such that $v_t = \boldsymbol{\phi}(t)_i$, and $e(v_t, v_n) \in E$ is the set of edges where v_t is connected to v_n only if $\boldsymbol{\phi}(t)_i$ is a predecessor of $\boldsymbol{\phi}(k)_i$, *i.e.*, when

$$|x(t + t_w) - x(k)| \leq \epsilon, \tag{5.17}$$

where ϵ is a radius factor, and $t_w = (m - 1)\tau$ is the time delay window, respectively. From a different perspective, Equation 5.17 tries to keep track on the intersections of states trajectories with the phase-space hyperdiagonal, as illustrated in Figure 1.2. Moreover, if phase states are represented in a tabular form as in Table 1, then Equation 5.17 will hold when the last column (the mth component) of $\boldsymbol{\phi}(t)_i$ matches the first component of $\boldsymbol{\phi}(n)_i$ according to ϵ. Then, the probability $P(\boldsymbol{\phi}(t))$ is set as a function of the number of edges, *i.e.*, the valency or degree of v_t, namely $\mathcal{V}_t$. Thus, one can naively define $P(\boldsymbol{\phi}(t)) = 1/\mathcal{V}_t$, but other options may be considered (Buhmann and Buhmann, 2003; Delashmit and Manry, 2005).

Alternatively, one can describe the uncertainty (and, as consequence, the level of determinism) in the phase space, embedded with the pair (m, τ), in terms of Von Neumann's entropy (Han et al., 2012), described as

$$E_{\text{vn}}(\boldsymbol{\phi}_i) = -\sum_{j}^{m} \lambda_j \log \lambda_j, \tag{5.18}$$

where λ_j is one of the eigenvalues of $\boldsymbol{\phi}_i$. Moreover, Equation 5.18 can also be defined in function of the probability of the eingenvaules, as presented in Equation 4.7. In both cases, the entropy will be higher when the attractor fully expands in all directions, and lower when an a subset of dimensions mainly describes the phase space.

Lastly, a third possible way to compute the entropy in the phase space is to partition it into m-dimensional cells and count the number of occurrences (states) inside of them. In that form, the partition usually follows some criterion. For instance, the space can be recursively and equally divided into quadtrees or octrees (Sperber, 2017) according to a minimal density of states n (if a cell has more states than n, it is split again), or using other geometrical property (Mount, 2010). A relation between the number of empty and non-empty cells can be used to formulate a probability function. Using the same example that the space was split using the density of states, Equation 5.15 could be computed for different values of n. This approach goes in the same direction of the correlation dimension D2 (Equation 2.23), but presented from a different perspective.

The last entropy modeling approach above also motivated us to consider an entropy definition based on the conditional probability of Naive Bayes (Raschka, 2014). Given a uniform partition of the space $\boldsymbol{\Phi} = [\mathcal{X}, \mathcal{Y}]$, for instance, where $\mathcal{X}$ and $\mathcal{Y}$ are the output and input subspaces (Equation 4.18 and Table 1), respectively, then n^m m-dimensional cells $c_{i,j\in[1,n]}$ can be created, so that the following relation is expected to be maximal for optimal phase spaces

$$P(\mathcal{Y}|\mathcal{X}) = \frac{P(\mathcal{X}|\mathcal{Y})P(\mathcal{Y})}{P(\mathcal{X})} = \frac{P(\mathcal{Y},\mathcal{X})}{P(\mathcal{X})} = \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{n_{c_{i,j}}}{n_{c_i}}, \tag{5.19}$$

where n_{c_i} represents the total of occurrences in the ith subspace of $\mathcal{X}$, and $c_{i,j}$ is number of occurrences in the intersection of such space with the jth subspace of $\mathcal{Y}$ (joint probability). Figure 5.11 illustrates an example of such computation for the cell $c_{1,3}$ in two scenarios. Although such approach is expected to work in theory, as the output space should be strongly dependent on the input space for deterministic embeddings, *i.e.*, the conditional probability of $\mathcal{Y}$ to happen given $\mathcal{X}$ should be high, unfortunately our experiments led to unsatisfactory results: no consistent patterns were observed for general systems.

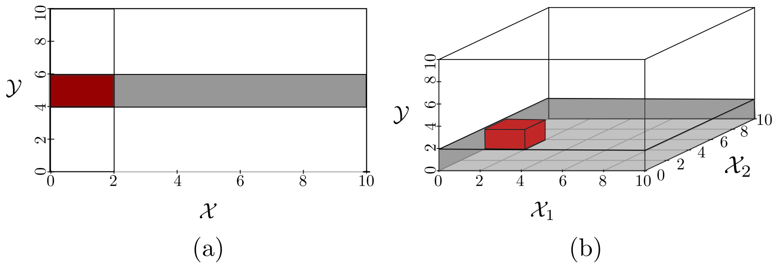


Figure 5.11: Conditional probability computation on the phase space for 2D **(a)** and 3D **(b)**. The cell $c_{1,3}$ is highlighted in red for both cases, whereas the ith subspace of $\mathcal{X}$ is represented in gray. For the 3-dimensional plot, $\mathcal{X} = [\mathcal{X}_1, \mathcal{X}_2]$.

Summarizing, estimating the entropy is dependent on some pre-requisites, such as the radius ϵ in Equation 5.17 and how the space is partitioned. This makes computing entropy difficult. Moreover, all entropies referenced in this section struggle from two additional problems: they are not unique descriptors and they suffer from the curse of dimensionality (Chen, 2009). In order to mitigate the last drawback, the Singular Entropy (SE) method (*cf* Section 4.2.1.4) relies on the first eigenvalue (Equation 4.7).

We have next tried to use those entropies to estimate the optimal embedding. Firstly, we aimed for a phase space where the number of intersections in the cobweb plot was minimal, so that

$$H_{\text{sh}}(\boldsymbol{\phi}) = \text{minimize}\left(E_{\text{sh}}(\boldsymbol{\phi})\right). \tag{5.20}$$

However, usually the hyperdiagonal-based attractor is found for this optimization problem, since this structure has the maximum determinism among its states (Rosenstein et al., 1994).

Apart from the above, Von Neumann's entropy could be used in an alternative way to find the most spread attractor in form

$$H_{\text{vn}}(\boldsymbol{\phi}) = \text{maximize}\left(E_{\text{vn}}(\boldsymbol{\phi}) - \beta\|\boldsymbol{v}\|_2\right), \tag{5.21}$$

where $\boldsymbol{v} = [\alpha_1, \alpha_2, \cdots, \alpha_m]$ is a vector indicating which dimensions should be used, *i.e.*

$$\alpha_i = \begin{cases} 1 & \text{dimension } i \text{ composes the phase space,} \\ 0 & \text{otherwise.} \end{cases} \tag{5.22}$$

Here, $\boldsymbol{v}$ is a penalty factor to decrease the importance of higher dimensions in Equation 5.21. However, this process is complex to implement and compute.

Thirdly, we considered to combine Equation 5.20 to Equation 5.21 in order to find some balance between under (redudance) and overestimated (irrelevance) phase spaces, in the form

$$H(\boldsymbol{\phi}) = \text{maximize}\left(H_{\text{vn}}(\boldsymbol{\phi}) - C \times H_{\text{sh}}(\boldsymbol{\phi})\right), \tag{5.23}$$

where $C \in [0, 1]$ is a constant weight that defines the trade-off between both quantities.

Although a solution is guaranteed to exist for Equation 5.23 (both entropies have concave characteristics (Boyd and Vandenberghe, 2004)), they are not ensured to converge to benchmark embeddings.

To test whether RQ1 holds, we computed the above entropies for well-known embeddings of the described in Chapter 3. The hypothesis was that, when optimal embeddings (as known by the literature for specific datasets) are used, then the computed entropies would be smaller when compared to other embeddings. The results of this experiment, however, proved inconsistent: we did not observe the expected correlation between minimal entropy and optimal embedding (again, the latter as given by the literature) for the analyzed datasets (Chapter 3). A similar behavior was reported for the SE algorithm (Section 4.2.1.4), where inconsistencies were found for attractors with genus more than one.

However, the fact that RQ1 was disproved in the above form does not mean that the reconstructed embeddings, estimated with these measurements, are not *good enough* to unfold phase spaces in some cases. Regressions on non-ideal phase spaces can still lead to adequate models and forecasting, if the estimated embedding is not too much far from the optimal. However, we have always aimed for an estimation as closest as possible to the generating rule $R(\cdot)$, such that overestimated embeddings (mainly in terms of τ) are not acceptable as valid. Chapter 9 will present a different way for estimating optimal embedding parameters, using deep learning, which has led to consistent results for all tested datasets.

As a final (and side) note, based on Equation 5.21, we have also considered to apply the Least Absolute Shrinkage and Selection

Operator (LASSO) (Tibshirani, 1996) to estimate the embedding pair, which models the problem as

$$\boldsymbol{b} = \arg\min_{\boldsymbol{b}\in\mathbb{R}^m} \frac{1}{N_i}||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{b}||_2^2 + C \times |\boldsymbol{y}|_1, \tag{5.24}$$

where $\boldsymbol{X}$ and $\boldsymbol{y}$ are matrix representations of $\mathcal{X}$ and $\mathcal{Y}$, respectively. The goal is to find a vector $\boldsymbol{b}$ of m coefficients

$$y_j = b_0 + \boldsymbol{X}_j^T\boldsymbol{b} = b_0 + x_{j,1}b_1 + x_{j,2}b_2 + \cdots + x_{j,m}b_m + \epsilon_j, \forall j \in [0, N_i - 1], \tag{5.25}$$

so that the sum of the squares of the differences between the dependent and exploratory variables is minimized. In order to use Equation 5.24, firstly $\boldsymbol{X}$ is standardized and $\boldsymbol{y}$ centered. Later, states are overembedded so the penalty factor will shrink some coefficients in $\boldsymbol{b}$ to zero. As consequence, we filter the most representative dimensions out by the absolute values of $\boldsymbol{b}$, and estimate τ according to the lags among non-zero coefficients. However, LASSO performs a linear regression, which leads to the fact that estimations are not consistent for nonlinear systems. As a refinement of this idea, we also considered to apply a kernel function to map nonlinear phase spaces into linear attractors. The idea was to find a map that would lead states into a linear space, preserving as much as possible their topological properties, perform LASSO on this space, and then perform an inverse mapping into the found coefficients of $\boldsymbol{b}$ to have an estimation with respect to the original space. Whether this approach leads to consistent results is, however, still subject to research.

5.8 FINAL CONSIDERATIONS

The goal of this chapter was to show that states are organized in an independent-and-identically manner (i.i.d.) in the optimal phase space, so that low levels of entropy are observed. To find this optimal space, we used the Statistical Learning Theory (SLT). Simply put, we construct a series of regressors, for different values of the embedding parameters m and τ, and choose as optimal embedding the one for which the constructed regressor generalizes best to unseen test data.

As a consequence, instead of relaxing the assumption of data independence in the context of time-series modeling as performed in other studies (Faria et al., 2013; Tan et al., 2011; Al-Khateeb et al., 2012), the usage of a kernel function to reconstruct time series into phase spaces is also important to ensure supervised learning in those scenarios. To the extent of our knowledge, there is no previous work that tackles this goal.

To prove the above statement, we show that forecasted series generated by building regression functions on phase spaces produce small empirical risks and ϵ, which restricts the expected risk to small upper limits and should be enough to theoretically prove learning. However, we also show in practical experiments that good forecasting results were generated, giving us strong evidences that learning occurred. According to SLT, however, learning can only be *proven* if examples are independent and identically sampled (given other assumptions are already held – as is the case). This allows us to conclude that our kernel function (Takens' embedding theorem), under the best parametrization, unfolds data observations into i.i.d. examples.

To confirm that points in ideal phase spaces are independent and identically distributed, we conducted experiments by producing an extensive set of phase spaces, using Monte Carlo simulation for the embedding parameters, which were evaluated in terms of their ability to build regressive functions. Training to predict the label of one single unseen observation at a time instant led to wrong phase-space parameters and poor prediction results. To find more reliable spaces, we therefore recursively predicted $h \leq H$ observations (given H is the prediction horizon). This latter approach results in optimal phase spaces, which match the most commonly used embedding parameters from the literature (Ikeda, 1979; Hitzl, 1981; Rössler, 1976). We also noticed that comparing forecasted and expected series using the Euclidean distance could lead to wrong results (a subject previously discussed in (Rios, 2013)), as mean-valued observations could be classified as more similar to the expected series than when comparing noisy observations. To overcome such scenarios and also consider data trends, MDDL was used as loss function.

In addition, to provide a fair training using different time instants of the series, we selected the best phase space that produced the lowest accumulative according to our k-Fold Time Cross Validation, whose steps can be summarized as: i) forecast h observations; ii) use MDDL as loss function; iii) proceed with a k-cross folding.

As a main contribution, this chapter provides theoretical and empirical evidences that phase spaces transform time-series observations into i.i.d. examples. Moreover, **the answer to (RQ1) is, partially, positive: as low levels of entropy correlates to higher levels of independence among phase states, optimal phase spaces can (consequently) be described in terms of such measurements.** However, it is still difficult to come up with a single definition (and way of computing) of entropy of an embedding, and we could not conclude that optimal embeddings (for the cases where such information was known) generate minimal entropy values over all possible embeddings.

Another consequence of this work is that supervised learning can be performed and ensured for time-dependent data. With this knowledge one can, for instance, employ supervised-learning algorithms to tackle classification or regression tasks in time series and have learning guarantees according to SLT. Other classifiers can also be used instead of DWNN. As main drawback, our approach requires at least thousands of observations to proceed with the phase-space reconstruction and obtain a meaningful attractor. However, this should not be a problem in practice, when dealing with real-world applications which already have access to large (big data) collections of measurements.

There are several directions of possible future work based on the results presented in this chapter: i) adapting the joint probability distribution so one can rely on learning guarantees to proceed with concept drift detection on data streams; ii) an adaptive strategy to define values for σ and/or $\mathcal{R}$ according to different attractor points or regions, by exploiting information on how points are locally spread over the phase space (Rios, 2013); iii) designing a more robust approach to penalize higher embedding dimensions as well as time delays to avoid unnecessarily complex phase spaces; and, finally, iv) using the proposed approach to forecast data from other real-world datasets from climate, biology, medicine, and other domains.

6 SEMI-SUPERVISED TIME-SERIES CLASSIFICATION ON POSITIVE AND UNLABELED PROBLEMS USING CROSS-RECURRENCE QUANTIFICATION ANALYSIS

6.1 INITIAL CONSIDERATIONS

Chapter 5 showed how optimal phase spaces can be characterized and computed using techniques and methods from Statistical Learning Theory and Machine Learning. The good results obtained in this process motivated us to extend our investigation on Dynamical Systems, which led to our second research question:

RQ2. "Is it better to use phase-space rather than time-series modeling?"

After measuring the forecasting accuracy to tackle RQ1, we next decided to validate phase-space analysis on *classification* scenarios. More precisely, we focused on the problem of Positive and Unlabeled (PU) data in dealing with semi-supervised learning. In this case, few labeled examples P from a single class of interest are available to proceed with the classification of unseen instances U, according to their similarities with the known class.

In the scope of time series, most of the current studies propose to address this topic using a self-training approach based on similarity measurements on the time domain, such as the Euclidean Distance (ED) or the Dynamic Time Warping-Delta (DTW-D), to provide features for the self-training classification stage, which is typically performed with the 1-Nearest Neighbor (1-NN) algorithm (Wei and Keogh, 2006; Ratanamahatana and Wanichsan, 2008; Chen et al., 2013). Self-training is employed to accumulate knowledge and, consequently, improve the classification of new instances. Despite the relevant contributions of time-domain measurements, we claim that such approaches do not consider temporal recurrences commonly found in natural phenomena (*e.g.*, population growth, climate studies) and are more sensitive to local noise and fluctuations, as already mentioned in Chapter 5.

To exemplify and reinforce such drawbacks of time-domain measurements, consider the analysis of a cyclical phenomenon, whose behavior is described by a sinusoidal signal (Equation 3.1, repeated below for ease of reading)

$$x(t) = A(t)\sin(2\pi t/n) + \theta) + \mathcal{U}(a, b), \tag{6.1}$$

where $A(t)$ is the amplitude along time, sampled at moments $t = 0, \ldots, n-1$; θ changes the sinusoidal phase; and $\mathcal{U}(a, b)$ adds noise to the samples following a Uniform probability distribution in range $[a, b]$.

Now consider that we create three examples of time series with the same length $n = 200$. The first series is a free-noise sinusoidal function with $A(t) = 1\,\forall t$, $\theta = 0$ and $\mathcal{U}(0, 0)$. The second is a dissipative sine whose observations were produced using $A(t) = \frac{n-t}{n}$, $\theta = \pi/2$ and $\mathcal{U}(-0.1, 0.1)$. The third series represents a random noise following a Uniform probability distribution $\mathcal{U}(-0.5, 0.5)$. All series are illustrated in Figure 6.1. Note that, although we made a couple of changes in the second signal, it remains sinusoidal-like. This simulates a real-world scenario in which we have two signals collected from the same phenomenon representing distinct behaviors at different time instants. The first series corresponds to the time interval in which the phenomenon is conservative. After some interaction (coupling) with another system, the signal begins to lose power eventually converging to zero, such in the case of the damped harmonic oscillator (Alligood et al., 1996). This leads us to the second signal.

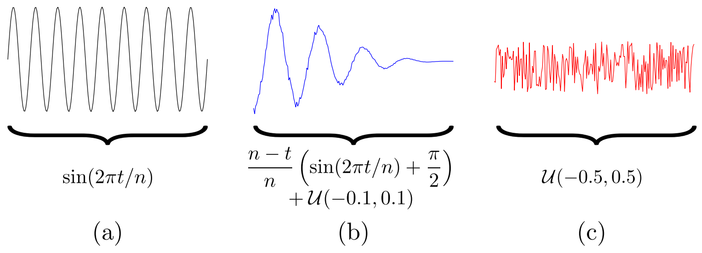


Figure 6.1: Examples of two series produced by variations of the sinusoidal function **(a,b)** and another series generated using a uniform distribution **(c)**. Assuming the first signal as the initial known positive example, time-domain measurements (see Section 6.3) may consider the conservative series **(a)** as more similar to the uniform distribution **(c)** than to the dissipative one **(b)**, thereby misleading a classifier. Adapted from (Pagliosa and de Mello, 2018).

Assume some specialist told us that the first series belongs to the positive set P, which was already studied in the context of our application domain (*i.e.*, cyclical phenomena), and that the other two series compose the unlabeled dataset U. Next, assume, for the sake of example, that we use a self-training strategy to label the most similar time series in U to P, using a 1-NN algorithm (other learning algorithms can be used as well). Although we know that the dissipative sine should be classified as a positive instance, time-domain measurements provide us the undesired result that the random noise should be added to the positive set instead (more details in Table 3). As discussed further in this chapter, besides not comparing recurrences (a feature that should be considered when dealing with natural phenomena), time-domain measurements are more sensitive to local differences enhanced by noise and mean-valued observations, so they can mislead classification. In a self-training scenario, this can lead to inconsistent and undesired results.

The above issues have motivated us to investigate the use of phase-space representations as an alternative to time-series representations for building classifiers of temporal data. In detail, we propose the use of the Maximum Diagonal Line of the Cross-Recurrence Quantification Analysis (MDL-CRQA), applied on phase spaces (Takens, 1981), as similarity measurement for classification. By comparing phase spaces rather than the series themselves, we can assess how their trajectories change along time (Marwan and Webber, 2015), including their periodicities and temporal cycles, as well as decreasing noise influences.

The remaining of this chapter is organized as follows. Section 6.2 shows the related work of time-series semi-supervised learning. Different methods typically used to compare time series are described in Section 6.3. Our approach is given in Section 6.4. We perform experiments comparing time-domain and phase-space domain in Section 8.5, to later discuss our results in Section 6.5.4 and finally draw conclusions.

6.2 RELATED WORK FOR SEMI-SUPERVISED LEARNING IN TIME SERIES

Despite the proposal of semi-supervised techniques like self-training (Li and Zhou, 2005), generative models (Baluja, 1999), co-training (Blum and Mitchell, 1998), density-based (Bennett and Demiriz, 1998), graph-based (Blum and Chawla, 2001), outlier detection (Janssens et al., 2009), and their extensions/modifications to tackle specific scenarios (Nigam et al., 2000; Chapelle and Zien, 2005; Zhu et al., 2009; Chapelle et al., 2010; Daneshpazhouh and Sami, 2014; Wang et al., 2016; Sheikhpour et al., 2017; Pereira and da Silva Torres, 2018; Wu and Prasad, 2018, etc.), to the extent of our knowledge, few studies addressed semi-supervised classification for time-series analysis in the literature.

The first study related to semi-supervised time series was proposed by Wei and Keogh (2006). By starting with a single positive instance s representing the positive set P, their self-learning method classifies a new positive instance belonging to the unlabeled dataset U as the most similar series in U to P, and the process continues until some stopping criterion is met. Despite their seminal contribution to the area, their approach has a couple of problems: i) they used the Euclidean Distance (ED) to compute the 1-NN algorithm, which is known to be less accurate than the Dynamic Time Warping (DTW) method in the presence of time-displacements (Ratanamahatana and Keogh, 2004); and ii) their stopping criterion (later referred to as Minsofar) was confirmed to be inadequate in several scenarios (Ratanamahatana and Wanichsan, 2008). Based on these observations, Ratanamahatana and Wanichsan (2008) proposed the Stop Criterion Confidence (SCC), which despite improving upon Minsofar, is not yet ideal, as it yields early termination for multiple datasets. Separately, Chen et al. (2013) used DTW-D (ratio of DTW by ED) to compare similarities between time series, improving the results reported by Wei and Keogh (2006).

Alternatively, Nguyen et al. (2011) relied on the method proposed by Wei and Keogh (2006) to classify a positive initial (training) set to later run k-means on the unlabeled dataset. Afterwards, the method applies PCA on both labeled and unlabeled sets to finally classify clusters based on their similarities provided by eigenpairs. Zhong (2004) uses self-training with Hidden Markov Models (HMM) to summarize time-series information. The algorithm first initializes the number of states using parameter k, from k-means, then maximizes the likelihood estimation for the HMM using positive labeled examples. Unlabeled instances are set as positive when their accumulated transition probability (similarity) is high for the trained positive model.

Out of the scope of self-training approaches, few other techniques exist for this problem. Marussy and Buza (2013) proposed a cluster-and-label multi-class algorithm which computes a minimum spanning forest (using DTW) among all instances, so that each tree has one labeled instance as root. Each tree starts with one labeled instance, proceeding with the addition of further nodes. At the end, series belonging to a tree are labeled according to the label of its root.

In short, we found that the majority of studies tackling semi-supervised time-series classification on PU problems have used the 1-NN algorithm with a self-training approach, including the seminal research proposed by Wei and Keogh (2006). Therefore, we also decided to follow this approach in order to support a fair comparison of results. A detailed analysis of the sensitivity of the 1-

NN classifier and a comparison thereof against other classification methods are out of the scope of our work.

More importantly, we also noticed that all related methods use *time-domain measurements*, such as ED and DTW, to measure similarities among time series. As already outlined above, this strategy may be not the best approach in many situations, especially when time series present strong cyclical patterns and trends. Hence, in contrast to existing research, we propose a novel self-training approach to tackle semi-supervised PU time-series classification using MDL-CRQA as similarity measurement, applied on the series *phase spaces* rather than on the series themselves. This allows us to assess and compare time-series recurrences more fairly, as we describe in Section 6.4. Our approach can also be extended to use other classification algorithms rather than 1-NN, without loss of generality.

6.3 TIME-DOMAIN SIMILARITY MEASUREMENTS

Current self-training methods use time-domain measurements to find the most similar instance to be labeled as a positive example throughout iterations. For instance, given two unidimensional time series T_i, T_j with n_i, n_j observations each, the Euclidean Distance (ED) computes the similarity between them as

$$\mathrm{ED}(T_i, T_j) = \sqrt{\sum_{t=0}^{n_i-1} (x_i(t) - x_j(t))^2}. \tag{6.2}$$

Despite simple, ED is not suitable to compare time-displaced series, additionally requiring both series to have the same length $n_i = n_j$ (although this constraint can be relaxed via interpolation approaches (Ratanamahatana and Keogh, 2004)).

Dynamic Time Warping (DTW) was proposed to address the comparison of time-displaced series, by finding the best match between shifted observations along time by computing

$$\mathrm{DTW}(T_i, T_j) = \sqrt{\sum_{t=0}^{n_i-1} p_{i,j}(t)^2}, \tag{6.3}$$

where $p_{i,j}(t) = x_i(t+\alpha(t)) - x_j(t+\beta(t))$ corresponds to the shortest warping path, with $\alpha(t), \beta(t) \in \mathbb{Z}$.

However, there are scenarios in which ED and DTW lead to incorrect results (Chen et al., 2013). To mitigate this drawback, a combination of both approaches was proposed, referred to as Dynamic Time Warping-Delta (DTW-D), computed by

$$\text{DTW-D}(T_i, T_j) = \frac{\mathrm{DTW}(T_i, T_j)}{\mathrm{ED}(T_i, T_j)}, \tag{6.4}$$

when compared to DTW and ED, DTW-D measurements improve classification results in several contexts.

Finally, Mean Distance from the Diagonal Line (MDDL) (Rios and de Mello, 2013) is another method to measure time-series similarities, defined as

$$\text{MDDL}(T_i, T_j) = \sum_{t=0}^{n_i-1} (p_{i,j}(t) - d_{i,j}(t))^2, \tag{6.5}$$

where $d_{i,j}(t)$ indicates the diagonal line (perfect match) in the space found by DTW, as illustrated in Figure 5.3(b). MDDL is adequate to compare time-series trends and disregard mean-valued time series, similarly to DTW-D.

6.4 SEMI-SUPERVISED TIME-SERIES CLASSIFICATION USING CRQA

Let two phase spaces $\mathbf{\Phi}_i$ and $\mathbf{\Phi}_j$ be properly reconstructed after applying Takens' embedding theorem on time series T_i and T_j, respectively, as discussed in Chapter 5. A Cross Recurrence Plot (CRP) between these two phase spaces yields the matrix $\boldsymbol{R}$ having as entries the values

$$R_{a,b} = \begin{cases} 1, & \text{if } \boldsymbol{\phi}_i(a) \text{ is a neighbor of } \boldsymbol{\phi}_j(b) \text{ according to an open ball} \\ & \text{centered at } \boldsymbol{\phi}_i(a) \text{ with radius } \varepsilon, \\ 0, & \text{otherwise,} \end{cases} \tag{6.6}$$

which indicates when (and where in the attractor) states $\boldsymbol{\phi}_i(a) \in \mathbf{\Phi}_i$ and $\boldsymbol{\phi}_j(b) \in \mathbf{\Phi}_j$ are close enough to each other. The CRP matrix $\boldsymbol{R}$ can also be used to measure for how long two phase spaces remain similar to each other. For instance, horizontal or vertical traces suggest trajectories on the given state are bound by an attractor in one space, while changing "normally" in the other. Similarly, sparse and small areas in $\boldsymbol{R}$ can indicate phase spaces do not share trajectories. However, such interpretations are delicate and require specialized domain knowledge. To avoid such complications, the Cross Recurrence Quantification Analysis (CRQA) was designed to extract a set of predefined measurements based on the CRP, which can be automatically used to reveal important features such as patterns and statistical distributions between phase spaces. Lastly, it is worth to say that, when computing the CRP, the number of states N_i and N_j may vary, but the dimension m must be the same for both embeddings being compared.

Based on Serrà et al. (2009), we consider the Maximal Diagonal Line (MDL) to indicate for how long the trajectories of two different phase spaces remain similar (close) to each other. In order to compute MDL, we start by filling the first column and row of $\boldsymbol{R}$ with zeros ($\boldsymbol{R}[*, \mathbf{0}] = \boldsymbol{R}[\mathbf{0}, *] = \mathbf{0}$) and define

$$R_{a,b} = \Theta(\varepsilon_a^i - \|\boldsymbol{\phi}_i(a) - \boldsymbol{\phi}_j(b)\|_2)\ \Theta(\varepsilon_b^j - \|\boldsymbol{\phi}_j(b) - \boldsymbol{\phi}_i(a)\|_2), \quad (6.7)$$

for $a = 1, \ldots, N_i - 1$, $b = 1, \ldots, N_j = 1$, where ε_a^i and ε_b^j are open ball radii for $\boldsymbol{\phi}_i(a)$ and $\boldsymbol{\phi}_j(b)$, respectively, and $\Theta(\cdot)$ is a Heaviside step function given by

$$\Theta(v) = \begin{cases} 0, & \text{if } v < 0, \\ 1, & \text{if } v \geq 0. \end{cases} \quad (6.8)$$

The radius ε_a^i is found by firstly computing the Euclidean distances from state $\boldsymbol{\phi}_i(a) \in \boldsymbol{\phi}_i$ to every other state $\boldsymbol{\phi}_j \in \boldsymbol{\phi}_y$. Then, we sort all those distances out in increasing order and set $\varepsilon_i^a = \varepsilon$, having ε as big enough to include the kth-nearest neighbor from $\boldsymbol{\phi}_i(a)$. In practice, we set k to 1% of the number of states in phase space. This way, we ensure that every state in $\boldsymbol{\phi}_i$ will always have the same number of neighbors in the other phase space $\boldsymbol{\phi}_j$ during the similarity analysis. The finding of the radius ε_j^b proceeds analogously.

The orbits of similar phase spaces may suffer from noise or small fluctuations. Thus, a perfect diagonal line may not occur in most scenarios. To model this, we can relax the concept of similarity by also allowing some "bumps" while computing the maximum diagonal in Equation 6.8. Thus, our next step consists of running a Dynamic Programming algorithm to fill the matrix $\boldsymbol{Q}$ which accumulates and penalizes recurrence similarities stored in $\boldsymbol{R}$. The matrix $\boldsymbol{Q}$ is defined by its entries

$$Q_{a,b} = \begin{cases} \max\{Q_{a-1,b-1}, Q_{a-2,b-1}, Q_{a-1,b-2}\} + 1, & \text{if } R_{a,b} = 1, \\ \max\{0, & \\ \qquad Q_{a-1,b-1} - \gamma(R_{a-1,b-1}), & \\ \qquad Q_{a-2,b-1} - \gamma(R_{a-2,b-1}), & \\ \qquad Q_{a-1,b-2} - \gamma(R_{a-1,b-2})\}, & \text{otherwise.} \end{cases} \quad (6.9)$$

We then use the Maximal Diagonal Line, defined as $\max(\boldsymbol{Q})$ to compare two phase spaces. To compute $\boldsymbol{Q}$, we initially set its two first columns and rows to zero, and use the auxiliary function

$$\gamma(z) = \begin{cases} \gamma_o & \text{if } z = 1, \\ \gamma_e & \text{if } z = 0, \end{cases} \quad (6.10)$$

with $\gamma_o = 5$, $\gamma_e = 0.5$ as disruption penalties[1]. As we are interested in providing a *dissimilarity* measure, we consider the inverse of MDL as $1/\max(\boldsymbol{Q})$. Note that $\max(\boldsymbol{Q})$ is never zero.

We claim that measuring similarity in phase space (rather than in the time domain) leads to better classification results in the context of semi-supervised PU learning. In order to support our claim, consider the three time series in Figure 6.1. In this situation, although the first two time series were produced by the same generating rule (sinusoidal function), local differences enhanced by different parametrizations lead time-domain similarities such as ED, DTW, DTW-D and MDDL to wrongly classify unlabeled instances.

Consider now performing comparisons in phase space rather than in the time domain. If we know or discover that the positive class contains sinusoidal-based time series, we could reconstruct the positive phase space using Takens' embedding theorem and analyze the similarity of this space against the other unfolded phase spaces (from the unlabeled dataset) using the same embedding parameters. More precisely, we use MDL from CRQA of two spaces, here referred to as MDL-CRQA. Figure 6.2 shows the phase spaces for the series in Figure 6.1 after reconstructing them using $m = 2$ and $\tau = 1$.

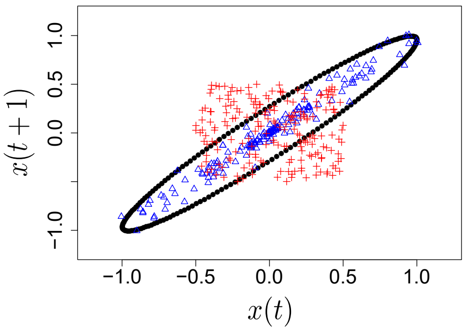


Figure 6.2: Phase spaces obtained for the time series illustrated in Figure 6.1. Dark circles, blue triangles and red crosses represent phase-space states of the first, second and third time series, respectively. Adapted from (Pagliosa and de Mello, 2018).

Table 3 lists the dissimilarities of the above time series when using time-domain dissimilarity methods (ED, DTW, DTW-D, and MDDL) as well as the phase-space-based MDL-CRQA. The results confirm that MDL-CRQA supports a better classification than local time-based measurements for this example.

1 Disruption penalties are heuristic weights used by Serrà et al. (2009) to improve the measurement of the longest diagonal line such as in Edit distance (Ristad et al., 1998).

Table 3: Comparing time series by using different similarity measures. We show the minimum, mean and maximum dissimilarities for each series (after adding random noise to it) *vs* the target series over 30 experiments. We show, in bold, the most similar "unknown" series that would be classified as the next positive instance.

Measure	Series	Minimum	Mean	Maximum
ED	Sine 1 / Sine 2	0.037	0.037	0.037
	Sine 1 / $\mathcal{U}$	**0.034**	**0.033**	**0.035**
DTW	Sine 1 / Sine 2	0.266	0.262	0.269
	Sine 1 / $\mathcal{U}$	**0.234**	**0.227**	**0.241**
DTW-D	Sine 1 / Sine 2	14.248	14.036	14.391
	Sine 1 / $\mathcal{U}$	**13.712**	**13.199**	**14.331**
MDDL	Sine 1 / Sine 2	64.301	35.508	83.686
	Sine 1 / $\mathcal{U}$	**22.396**	**10.137**	**53.695**
MDL-CRQA	Sine 1 / Sine 2	**0.026**	**0.020**	**0.032**
	Sine 1 / $\mathcal{U}$	0.158	0.111	0.200

The idea of above experiment is not to induce that time-domain dissimilarities perform incorrectly in *all* scenarios and should be discarded. Conversely, we just want to show that it might not be difficult to find examples for which all these dissimilarities would lead to wrong classification results. Additionally, we reinforce why phase-space measurements should be included in time-series analysis.

In summary, our semi-supervised classification method requires three initial settings: i) one initial positive example; ii) the embedding dimension m; and iii) the time delay τ (both m and τ are used to represent the phase space for the positive class). Given those, we compute the embedding parameters for the single starting positive example to unfold the phase space of the positive class according to Takens' embedding theorem. Next, we unfold any new instance phase space using the same embedding parameters, and compare it with the positive example using MDL-CRQA as similarity function. As proposed by Wei and Keogh (2006), we use the 1-NN algorithm for classification, *i.e.*, to specify the unlabeled instance with the greatest probability to belong to the positive class. As already mentioned, other classification algorithms can also be easily used.

6.5 EXPERIMENTS

In order to get additional insights on the behavior and performance of our proposed phase-space dissimilarity, we performed three sets of experiments: i) the first with synthetic time series to validate our method; ii) the second with real-world time series; and iii) and a third, also involving real-world data, that simulates a more difficult scenario when non-positive time series have similar features both in time and phase domains with the positive set.

Each experiment considers three dynamical systems. When such systems are represented by generating rules $R(\cdot)$ in the form of unidimensional signals/functions, such as the sinusoidal function (Equation 6.1), we basically take n observations from $R(\cdot)$; when dealing with multidimensional fluxes/maps, such as the Lorenz system, we take n observations from the first dimension (any other dimension could have been used equally well) to represent trajectories in the phase space (Kantz and Schreiber, 2004). For all experiments, we use $n = 10^5$ samples.

One of the functions in Chapter 3 is chosen to represent the phenomenon under the positive class. A few other functions are defined to compose the unlabeled dataset U. To create the training and test datasets, we divide the positive and unlabeled series into 32 and 50 sub-series, respectively, with 200 observations each.

To test the classification performance in the presence of noisy data, we generated time series using $\mathcal{N}(0, 1^2)$, *i.e.*, a normal probability distribution with mean 0 and standard deviation 1, added to 2/3 of all positive instances. In addition, we also included in the unlabeled set time series representing the mean-valued series from the positive series, as well series deriving from Normal distributions $\mathcal{N}(0, 0.05^2)$ and $\mathcal{N}(0, 0.1^2)$. Summarizing, our experiments use 32 positive instances and 232 unlabeled series (200 plus 32 mean-valued series from the positive set), as illustrated in Figure 6.3.

We used 50% of positive and 90% of unlabeled instances for training, leaving the remaining time series for testing. Only a single positive example was used to initiate the self-learning algorithm; the remaining positive instances were added to the unlabeled dataset. As the stopping criterion is an open problem in the PU literature, we decided to employ the method proposed in (Chen et al., 2013) to train our classifier until all positive instances (belonging to the unlabeled dataset) were labeled. We then used the labeled training series to classify the test observations using the 1-NN algorithm. As the classifier can be influenced by the choice of the selected positive instance, we ran it using different values for s multiple times. As final results, the mean precision, recall and F1-score performances over all experiments are reported.

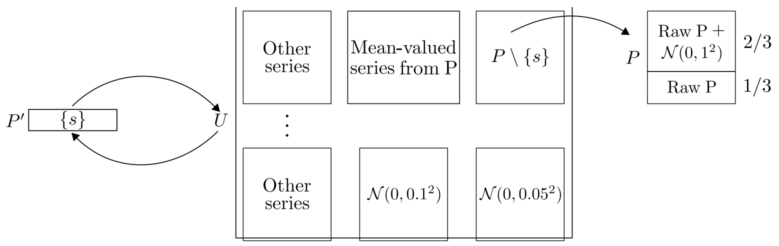


Figure 6.3: The set of unlabeled instances U is composed of: i) positive instances P but one randomly selected series s; ii) series from other systems; iii) two random series representing noise and iv) constant mean-valued series from $P \setminus \{s\}$. The self-learning algorithm continues until all positive instances are correctly classified, *i.e.*, when $P \subseteq P'$. Adapted from (Pagliosa and de Mello, 2018).

6.5.1 *Case Study 1: Synthetic Data*

We start our experiments by analyzing synthetic time series in order to validate our method. For such time series, their generating rules $R(\cdot)$ are well known. Hence, the embedding parameters to reconstruct their phase spaces are also known. In this context, we chose the Logistic map (Equation 3.2), the Hénon map (Equation 3.3), and the Lorenz system (Equation 2.6) to compose the synthetic experiments.

Among all possibilities, we defined the Lorenz system to compose the positive class, randomly choosing one series from it and leaving all remaining series to form the unlabeled set. We used the well-known embedding dimension $m = 3$ and time delay $\tau = 8$ for reconstructing the phase space associated with the Lorenz system. Classification performances are shown in Table 4. As one can notice, time-domain measurements are more sensitive to local disturbances, such as, but not limited to, noisy observations and mean-valued series. Therefore, by comparing time-series trajectories and recurrences along a wider period of time, MDL-CRQA becomes a global measurement that suffers less from those fluctuations, achieving better classification results.

6.5.2 *Case Study 2: Real-World Data*

In this experiment, we consider the real-world Sunspot dataset (Andrews and Herzberg, 1985) to belong to the positive set. As this series follows sinusoidal-like trajectories (but with significant noise), we chose the embedding pair $(m = 2, \tau = 8)$ to define the positive class phase space. The unlabeled set was formed by series deriv-

Table 4: MDL-CRQA supports better classification results for the Case Study 1: our method correctly classified 100% of the positive instances.

Dissimilarity	Precision	Recall	F1-score
ED	0.410	**1.000**	0.581
DTW	0.410	**1.000**	0.581
DTW-D	0.842	**1.000**	0.914
MDDL	0.444	**1.000**	0.615
MDL-CRQA	**1.000**	**1.000**	**1.000**

ing from the Rössler system and the Ikeda map (as well as the mean-valued and noise series for the positive class).

The performance results are listed in Table 5. Similarly to the first experiment, MDL-CRQA yielded the best classification performance, achieving almost 20% more precision than DTW-D. The best explanation for such results is again the presence of noise, which usually misleads classification when time-domain dissimilarities such as ED and DTW are used.

Table 5: MDL-CRQA supports better classification results for the Case Study 2.

ED	0.410	**1.000**	0.581
DTW	0.410	0.992	0.581
DTW-D	0.787	0.898	0.820
MDDL	0.432	**1.000**	0.603
MDL-CRQA	**0.962**	**1.000**	**0.979**

6.5.3 *Case Study 3: Recurrent Time Series*

In the last experiment, we analyze how our method behaves when non-positive time series have similar (up to a certain limit) recurrences of the positive instances. In other words, we simulate the case where some unlabeled time series have similar phase spaces to the positive instance, but should not be taken as positive due to small variations. This case is more challenging than the first two described so far, where the distinction between the classes is sharper.

In order to construct this scenario, we used the same series from previous experiment, *i.e.*, the Sunspot dataset as the positive class, and series deriving from the Rössler system and the Ikeda map defined the unlabeled set, respectively (as well as the mean-valued and noise series). We also included a sinusoidal function with parameters $A(t) = 1$, $\theta = 0$ and $\mathcal{U}(0, 0)$ and noise $\mathcal{N}(0, 0.05^2)$ added to it. This last addition creates a non-positive phase space (the sine phase space) whose dynamics partially mimics the Sunspot phenomenon, as illustrated in Figure 3.5. Nevertheless, as observed in this figure and in Figure 3.1, the Sunspot and the sine time series model different phenomena. Consequently, sine instances should not be classified as positive.

Although similarities between phase spaces may lead MDL-CRQA to wrongly classify some positive instances, this measurement still provides the best classification results, as shown in Table 6. Therefore, we empirically conclude that our method is robust enough to classify PU time series even when non-positive time series share some common patterns and recurrences with the positive examples.

Table 6: Case Study 3: Although positive and unlabeled series (especially the ones generated from the sine function) present similar trends and recurrences, MDL-CRQA still supports better classification when compared to time-domain measurements.

Dissimilarity	Precision	Recall	F1-score
ED	0.410	**1.000**	0.581
DTW	0.381	0.875	0.530
DTW-D	0.628	**1.000**	0.760
MDDL	0.444	**1.000**	0.615
MDL-CRQA	**0.849**	**1.000**	**0.917**

Even for unlabeled series with sinusoidal behavior, MDL-CRQA was capable of separating those series from the Sunspot ones. This happened since the Sunspost series are not a perfect sinusoidal function and, in addition, it contains noise. If our classifier had not achieved good results, we could also consider to overembed the positive class to obtain a more representative phase space (Kantz and Schreiber, 2004), *i.e.*, increase the embedding dimension m in order to unfold more complex data (such as data containing noise).

By adding extra dimensions to the phase space (up to a certain limit), one can analyze the details of more complex dynamical system trajectories (Alligood et al., 1996) and improve the separation of Sunspot versus the sine series. Although the Sunspot resembles sinusoidal, it is not as much sinusoidal as the sine function itself. Therefore, the trend is that those phase spaces become more dissimilar to each other as we increase their embedding dimensions.

6.5.4 *Discussion*

According to our experiments, we confirmed that MDL-CRQA supports better classification results for both synthetic and real-world time series whose data present recurrent observations. In order to mitigate the influences of the starting positive instance in the self-learning algorithm, we performed each experiment several times varying the starting example, and reported the mean performances achieved at each iteration as the final result. Nonetheless, we noticed the results barely vary when different positive examples were used.

As reported in Section 6.5.1, our method correctly classified 100% of the positive instances in the first experiment (synthetic data with added noise). When dealing with the real-world Sunspot dataset (Section 6.5.2), although making some errors, MDL-CRQA still achieved the best classification results when compared to time-domain measurements, even when non-positive time series share common recurrences with the positive set (Section 6.5.3).

In addition, we also have tested our method with datasets from the UCR collection (Chen et al., 2015), which contain several time series commonly used as benchmark. Training and testing files are already defined for each dataset in this collection. In this context, we think two relevant aspects are worth to be mentioned. First, the majority of those datasets were not designed to simulate PU problems, bringing relevant issues when defining the positive class. As consequence, the proper embedding parameters to unfold the positive phase space were also unknown. In order to overcome those issues, we naively defined all instances under the class label 1 as positive (we observed this class usually has fewer observations) and left all remaining classes as being part of the unlabeled dataset; and, to create the phase space for positive instances, we relied on current estimation methods (Kennel et al., 1992; Fraser and Swinney, 1986). As in the previous experiments, we assume all positive instances were unlabeled except one which was used to start the self-training process. For these datasets, our results of precision and recall did not surpass 0.5 on average. However, we noticed that none of the time-domain measurements achieved good results, to mention, they did not surpass 0.5 in terms of F1-score on average. While DTW achieved better classification performances for some datasets, DTW-D, MDDL, and even ED measurements provided better results for others.

Although the results for the UCR datasets are far less positive than the ones for the three types of datasets discussed in the previous sections, we report them here for two reasons: i) to confirm one of the main motivations of our study here, namely that time-domain measurements can lead to inconsistent results; ii) to highlight the importance of having a proper phase-space embedding (we cannot get good dissimilarities if this embedding is not found).

Limitations of our method include a higher computational effort when compared to time-domain methods[2], since MDL-RQA needs to compare phase states ($\mathcal{O}(N_i)^2$ steps) while time-domain measurements perform computations only using the time series itself ($\mathcal{O}(n)$ and $\mathcal{O}(n^2)$ steps for ED and DTW). Thus, despite the computation of the Maximal Diagonal Line (MDL) demands extra processing time specifically when studying high dimensional phase spaces, we believe this is not prohibitive in several practical scenarios due the increase in the use of cluster computing, optimization packages and parallel programming. As future work, one could use more than one simultaneous measurement to enhance classification results. As we stated, the unlabeled instances may contain any time series outside the positive class. Therefore, even MDL-CRQA may wrongly classify certain datasets. In this situation, one could use a co-training technique to learn from the time domain using ED, DTW, DTW-D or MDDL and from the phase domain using MDL-CRQA. Finally, the setting the stopping criterion is a fundamental, but open, question in PU problems, which still requires further research.

6.6 FINAL CONSIDERATIONS

The PU scenario is a well-known problem in semi-supervised classification (Wei and Keogh, 2006; Ratanamahatana and Wanichsan, 2008; Chen et al., 2013). Despite relevant contributions, current methods tackling PU problems using measurements such as ED, DTW and DTW-D do not compare temporal recurrences. This feature may be of great importance when comparing time series, especially when observations repeat themselves as is the case of many real-world scenarios, some of them studied along the manuscript such as population growth, meteorological data and sunspot activity.

In this chapter, we investigated the comparison of time series by using a dissimilarity measure, called MDL-CRQA, defined on their phase spaces, computed by using suitable embedding parameters. Our proposal, measured over two attractors in the same phase space (where all possible scenarios are unfolded, and therefore recurrences are easily modeled) is a feasible approach to measure the amount of recurrence one time series has with another. This approach attempts to mitigate local problems caused by mislead-

2 For example, the Case Studies took around 30 minutes while running on a 40-core Xeon processor at 2.8Ghz.

ing noise, chaotic events and eventually different observations produced along the collection of the phenomenon of interest.

In order to apply our method, we require two parameters from the user: the embedding dimension m and time delay τ for reconstructing the series under the positive class according to Takens' embedding theorem (Takens, 1981). These parameters can be either known (for a given problem domain) or else estimated using the methods discussed in Chapter 5.

Experimental results confirm MDL-CRQA improves classification results for PU time series when compared against the mostly used time-domain similarity measurements. **This answers our research question 2 (RQ2) positively: yes, phase-space methods do lead to better models of time series, when properly unfolded.** However, the answer needs to be nuanced: we have only shown that phase-space methods are superior to time-domain modeling for a *subclass* of problems, namely PU scenarios; and even for those, there exist datasets for which both phase-space models and time-domain models perform poorly. Lastly, refining our initial phase-space model, *e.g.*, by using different distance measures or better classifiers, is an open and interesting direction for future work.

7 ON THEORETICAL GUARANTEES TO ENSURE CONCEPT DRIFT DETECTION ON DATA STREAMS

7.1 INITIAL CONSIDERATIONS

Chapter 6 showed that phase-space modeling is an effective tool, as opposed to time-domain modeling, for applications such as semi-supervised learning. However, the datasets and use-cases considered in such a chapter used measurements drawn from a *single* phenomenon. We now focus on how to extend the learning paradigm for time-dependent data derived from *multiple* phenomena, a problem mentioned as important also in Chapter 5. More precisely, in Chapter 5, we relied on the SLT framework (Section 5.2) to perform regression in the phase space, assuming that input data was given by a fixed distribution. However, such a constraint is not always met when dealing with continuously collected observations (data streams) in the context of concept-drift detection. As such, in this chapter, we focus our work to answer the next question:

RQ3. "How to ensure learning in concept-drift scenarios?"

Formally, data streams are open-ended sequences of uni or multidimensional observations rather than batch-driven datasets (Dua and Karra Taniskidou, 2019)[1]. These observations are generated by processes modeled as stochastic and/or deterministic dynamical systems (Section 2.3) which simulate several phenomena at different timestamps such as temperatures at a given world region, flood sensing, or motor and cognitive development (Agarwal, 1995; Metzger, 1997; Rios et al., 2015). Those processes, or their parameters, may change along time due to some other phenomenon interacting and/or acting on them, *e.g.*, the effect of a medicine on blood pressure (Andrievskii and Fradkov, 2003). Such data behavior changes are referred to as *Concept Drift* (CD), pointing out decisive instants that some system or phenomenon should be studied in order to comprehend anomalous behaviors.

Concept-drift algorithms compare features from current to next observations to detect relevant changes (Gama et al., 2014). Such

1 In this chapter, we consider unidimensional streams only; however our work can be extended to multiple dimensions, without loss of generality, following (Serrà et al., 2009).

features are usually modeled by classification performance (Gama et al., 2004b; Baena-García et al., 2006; Bifet et al., 2009) or statistical measurements (Gama et al., 2014; Page, 1954; Bifet et al., 2009). Although classification methods generally lead to more robust comparisons, they require class labels to perform supervised learning, which may not always be available. Conversely, statistical methods have the advantage of requiring no label, but they cannot distinguish more complex processes from each other, especially when dealing with non-stationary or chaotic phenomena (da Costa et al., 2017).

More importantly, neither classification nor statistical methods provide *learning guarantees* to support CD detection, even when results use performance measures such as accuracy, Mean Time Between False Alarms, and Mean Time for Detection (da Costa et al., 2016). Such measures cannot be trustworthy when the algorithm poorly generalizes (under or overfits). In other words, either the CD algorithm may randomly issue drifts and still provide adequate performance according to the considered metrics, or it may overfit in order to provide the best possible result.

Instead of considering specific measurements on particular scenarios, in this chapter we propose a general and formal approach to perform CD detection relying on Statistical Learning Theory (SLT) (Vapnik, 1998). As consequence, our strategy provides the necessary probabilistic foundation to ensure reported drifts are not by chance.

We start by introducting the notations and terminology related to Concept Drift (Section 7.2). Next, we adapt and map SLT requirements (already introduced in Section 5.2) to the context of CD algorithms (Section 7.3). This provides us a theoretical framework for comparing actual CD algorithms. We next use this framework to analyze and compare several state-of-the-art algorithms (Section 7.4). This analysis shows us, interestingly, that no CD algorithm, from the set of analyzed ones, complies perfectly with SLT. Finally, Section 7.5 concludes this chapter.

7.2 CONCEPT-DRIFT DETECTION

Let a data stream $\mathcal{D}$ be defined as the sequence of observations

$$\mathcal{D} = \{x(0), x(1), x(2), \cdots, x(\infty)\}, \quad x(k) \in \mathbb{R}, \tag{7.1}$$

describing the behavior of some phenomenon along time. Differently from a time series (Equation 2.1), a data stream defines a continuous flow of incoming data, whose observations are derived from (potentially) multiple Joint Probability Distributions (JPDs). Thus, a time series T_i can be seen as the jth window W_j of $\mathcal{D}$, such that $\bigcup_{j=0}^{t\to\infty} W_j = \mathcal{D}$ and W_t represents the current window (see Figure 7.1). In this context, despite the fact that $T_i = W_j$, differentiating time series from data streams is necessary: whereas the time-series subindex defines the phenomenon of interest (or, from another perspective, the variable/dimension from the phenomenon), the window subindex shows the "location" of T_i in $\mathcal{D}$. Additionally, although the configuration of windows may vary from application to application, it is common to assume a fixed length n for every window without the overlapping of observations, so that

$$W_j = \{x(jn), x(jn+1), \cdots, x(jn+n-1)\}. \tag{7.2}$$

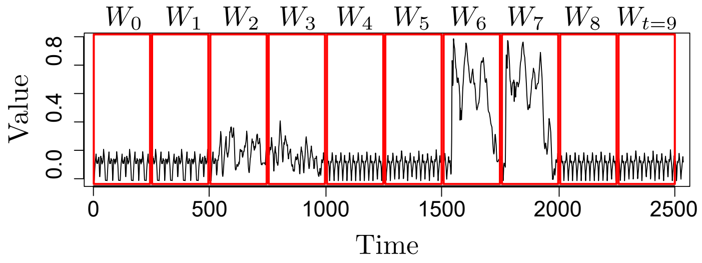


Figure 7.1: A data stream divided in 10 windows (red boxes) with no overlapping, each containing $n = 250$ observations. In this example, the data stream contains (up to the current moment) four different phenomena, namely: $T_1 = \{W_0, W_1\}$, $T_2 = \{W_2, W_3\}$, $T_3 = \{W_4, W_5, W_8, W_9\}$, $T_4 = \{W_6, W_7\}$.

If s denotes the initial window W_s (initially set to zero) describing some phenomenon, a CD algorithm induces the indicator function

$$g_{t-1} : \phi(f_{[s,t)}) \to [0,1], \tag{7.3}$$

which basically classifies whether the incoming window W_t continues to represent the same phenomenon or not. For brevity, the index of g is next omitted unless necessary to track its current time (as in Section 7.4).

Next, let the function ϕ model the extraction of a vector of features $\boldsymbol{v}_j = \phi(f_j)$ from the model

$$f_j : \mathcal{X}_j \to \mathcal{Y}_j,\ \forall j \in [s,t], \tag{7.4}$$

where $\mathcal{X}_j$ and $\mathcal{Y}_j$ are the input and class spaces of window W_j, respectively, derived either after applying dynamical-system reconstructions (Section 5.3) or by using statistical measurements (Gama et al., 2014; Page, 1954; Bifet et al., 2009). Such features can be simply the result of f_j itself so that ϕ is the identity function (*e.g.*, if the model is based on the average, variance, or entropy of a window) or the result of more complex filters and feature extractors (*e.g.*, when f_j is represented by a Neural Network (Haykin, 2009), features can be given by unit weights or values of activation functions).

Formally, we define $\boldsymbol{v}_t$ as the features obtained for the current window W_t, and $\boldsymbol{v}_{[s,t)}$ as the set of vectors including the same features but from past windows $W_{[s,t)}$. In this context, a CD algorithm, here responsible for inducing the function g, reports a drift whenever $\boldsymbol{v}_t$ significantly differs from $\boldsymbol{v}_{[s,t)}$ by more than an acceptable threshold λ. If the divergence between features is however small, g understands that $\boldsymbol{v}_t$ and $\boldsymbol{v}_{[s,t)}$ are from the same phenomenon. Thus, the model g (Equation 7.3) is updated such that $\boldsymbol{v}_{[s,t]} = \boldsymbol{v}_{[s,t)} \cup \boldsymbol{v}_t$. Lastly, t is incremented to represent a new window.

From the above, we see that drift detection depends on the divergence computed on consecutive windows. Note, however, that $\boldsymbol{v}_{[s,t)}$ is much greater than $\boldsymbol{v}_t$. Thus, g must either perform aggregations or apply kernel functions to be make sure $\boldsymbol{v}_{[s,t)}$ has the same number of features than $\boldsymbol{v}_t$ in order to proceed with a fair comparison. In this context, the former strategy is commonly employed in the form

$$g(\boldsymbol{v}_t) = \begin{cases} 1, & \text{if } \|\boldsymbol{v}_t - (\mu_{\boldsymbol{v}_{[s,t)}} + \eta\sigma_{\boldsymbol{v}_{[s,t)}})\|_2 > \lambda \text{ or} \\ & \|\boldsymbol{v}_t - (\mu_{\boldsymbol{v}_{[s,t)}} - \eta\sigma_{\boldsymbol{v}_{[s,t)}})\|_2 > \lambda, \\ 0, & \text{in case of no drift,} \end{cases} \tag{7.5}$$

where $\mu_{\boldsymbol{v}_{[s,t)}} = 1/w \sum_{j=s}^{t-1} \boldsymbol{v}_j$ and $\sigma_{\boldsymbol{v}_{[s,t)}} = \sqrt{\sum_{j=s}^{t-1} \frac{(\boldsymbol{v}_j - \mu_{\boldsymbol{v}_{[s,t)}})^2}{w-1}}$ are the average and standard deviation of past features, respectively; $w = t - s - 1$ is the current number of windows describing the same phenomenon; $\eta \in \mathbb{R}_+$ controls the sensitiveness of detection; and $\|\cdot\|_2$ is the Euclidean norm.

The greater the value of η is in Equation 7.5, the smaller is the number of reported drifts. Conversely, the lower the η, the easier it is for the algorithm to detect false drifts. Figure 7.2 exemplifies this trade-off for different values of η. In this example, the data stream $\mathcal{D}$ contains (until the current time $t = 3000$) three sinusoidal waves. Hence, two drifts should be reported: the first at $x(1000)$ owing the slightly changing in the wave frequency and amplitude; and the second at $x(2000)$, after a more drastic changing in these parameters. As it can be seen, different outcomes might be derived according to the sensitiveness of g to small/large variations.

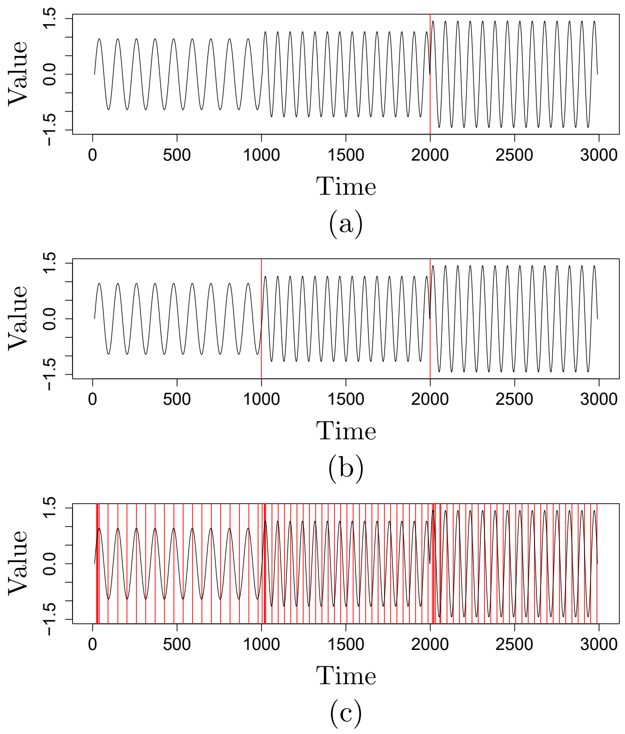


Figure 7.2: Alarms are depicted by red vertical lines. **(a)** The value of η is too large, such that drifts derived from small changes are not captured. **(b)** A better choice of η has led to the optimal model g. **(c)** As η is decreased, the CD algorithm becomes too sensitive, resulting in alerts of false drifts.

7.3 ENSURING LEARNING IN CONCEPT-DRIFT SCENARIOS

Our goal is to elaborate the necessary conditions a CD algorithm should satisfy to ensure drift detections are the *direct result* of actual changes in the phenomenon under analysis. It is worth to make clear that we do not intend to propose any new CD algorithm. Rather, we want to understand under which conditions an existing CD algorithm works as intended, and analyze existing algorithms in the light of these conditions. This allows us to determine when a CD algorithm performs correctly, in which case we can next safely use existing performance measures to validate the quality of reported drifts.

To set up a theoretical framework for understanding how to ensure learning for CD algorithms, we use, again, the Statistical

Learning Theory (SLT) (Vapnik, 1998). Recalling from Section 5.2, the assumptions of SLT are:

A1. examples must be independent from each other and sampled in an identical manner;

A2. no assumption is made about the Joint Probability Distribution (JPD), otherwise one could simply estimate its parameters;

A3. labels can assume non-deterministic values due to noise and class overlapping;

A4. the JPD is fixed, *i.e.*, it cannot change along time; and, finally,

A5. data distribution is still unknown at the time of training, thus it must be estimated using data examples.

Finally, it is worth to mention that the algorithm bias $\mathcal{F}$ must follow the Bias-Variance Dilemma (BVD) (Geman et al., 1992; Luxburg and Schölkopf, 2011; de Mello and Moacir, 2018). Thus, a balanced complexity of the function class is recommended to achieve the best-as-possible risk minimization (Geman et al., 1992). Assumptions A2, A3 and A5 are straightforwardly fulfilled in most real-world scenarios. However, assumptions A1 and A4 are more difficult to ensure, especially in the CD scenario, in which observations are *time dependent*, and *different phenomena* (with distinct JPDs) are expected to happen.

7.3.1 *Adapting The SLT To CD Scenarios*

Our proposal starts by adapting the general concepts of SLT, described in Section 5.2, to ensure learning bounds in the context of CD detection. We remind that the class set $\mathcal{Y}_j$, typically assumed when inferring $f_j : \mathcal{X}_j \rightarrow \mathcal{Y}_j$ on each window W_i, is mostly often not available. This is due to the difficulty for a human specialist to continuously label observations collected over time, especially for high-frequency streams.

From the above, we conclude that class labels must be somehow extracted on the fly from the data stream itself. Two possible strategies can been used for this: (i) if f_j is the result of a regression performed on the phase space $\mathbf{\Phi}_i$ of window W_j, then each input $\boldsymbol{x}_k \in \mathcal{X}_j$ is a tuple composed of the first $(m-1)$ components of $\mathbf{\Phi}_i(k)$, while the respective class label $y_k \in \mathcal{Y}_j$ is the last component of such state, as already shown in Table 1; and (ii) when the class information is merely the result of a measurable function

$m(\boldsymbol{x}_k)$, such as the average, variance, kurtosis, or similar, then observations themselves are the input data, such that $W_j = \mathcal{X}_j$, and the output is simply given as $m(\boldsymbol{x}_k) = y_k \in \mathcal{Y}_i$ (Bifet et al., 2009).

In the next step, function ϕ extracts the vector of features $\boldsymbol{v}_j$ from the inferred model f_j, such that $\boldsymbol{v}_j = \phi(f_j)$. Then, the indicator function g is responsible for mapping every feature vector into a binary space, indicating whether a drift has happened given the current data window or not (Equation 7.3). If no drift is detected, then g is expected to be updated based on the new features, thereby ensuring model adaptation. Thus, the set of features continuously approximates the true set of features corresponding to the analyzed phenomenon as time passes, allowing us to elaborate the following connection to the ERMP (Equation 5.3)

$$P(\|\boldsymbol{v}_{[s,+\infty]} - \boldsymbol{v}_{[s,t)}\|_2 \geq \epsilon) \to 0, \quad n \to \infty. \tag{7.6}$$

In other words, if we assume the difference between the true and empirical risks $|R(f) - R_{\text{emp}}(f)|$ decreases as the sample size increases, then it is fair to expect that, simultaneously, the features extracted along time also converge to the true features over the entire data population. Ideally, we should use a window length large enough to contain all observations from the analyzed phenomenon (de Mello et al., 2019). However, this becomes a great challenge as: (i) we do not have access to all observations from a phenomenon (we cannot, among others, see what the future will deliver), and (ii) several drifts are expected to happen in early windows. Thus, we decided to adapt the Symmetrization lemma (Equation 5.14), rewritten next for clarity as

$$\begin{aligned} &P(\sup_{f \in \mathcal{F}} |R(f) - R_{\text{emp}}(f)| > \epsilon) \leq \\ &2P(\sup_{f \in \mathcal{F}} |R_{\text{emp}}(f) - R'_{\text{emp}}(f)| > \epsilon/2) \leq \delta \end{aligned} \tag{7.7}$$

to represent learning in terms of windows features in the form

$$P\left(\sup_{f_j \in \mathcal{F}} \|\boldsymbol{v}_{[s,+\infty]} - \boldsymbol{v}_{[s,t)}\|_2 \geq \epsilon\right) \leq 2P\left(\sup_{f_j \in \mathcal{F}} \|\boldsymbol{v}_t - \boldsymbol{v}_{[s,t)}\|_2 \geq \frac{\epsilon}{2}\right) \leq \delta. \tag{7.8}$$

As $\boldsymbol{v}_{[s,t)}$ represents an aggregation of all measurements for past windows, the sample sizes of $\boldsymbol{v}_t$ and $\boldsymbol{v}_{[s,t)}$ are the same. Therefore, if the difference $\|\boldsymbol{v}_t - \boldsymbol{v}_{[s,t)}\|$ is held low as new windows are processed, we have probabilistic support that g is actually learning from data.

7.3.2 *Satisfying SLT Assumptions*

In order to use on Equation 7.8, however, we must satisfy SLT assumptions A1 and A4 listed in Section 5.2. Moreover, for practical reasons, we also need to ensure such equation is consistent by choosing a CD algorithm whose complexity is moderate, according to the Bias-Variance Dilemma (BVD) (Vapnik, 1998).

Firstly, we draw attention to the fact that drifts will happen only between windows, not among observations. According to our approach, the algorithm responsible for inferring f_j, using window W_j, is expected to deal with A1, while model g faces the challenge A4. Moreover, models f_j should employ some strategy to map observations into a different space, ensuring data becomes i.i.d. For instance, the Fourier transform (Bracewell, 1978) could map windows into the frequency space, or the Takens' embedding theorem (Takens, 1981) could reconstruct observations into phase spaces (Chapter 5). Following the research path of this thesis, we believe that the latter is better as it allows a more diversified analysis (Section 2.6). Complementarily, model g assumes that each data window may come from distinct but fixed/unique probability distributions, so when this indicator function reports a drift, any previous model should be discarded, allowing a fresh start to analyze a next coming distribution while still ensuring learning guarantees.

Regarding under/overfitting, one should choose functions f_j and g whose bias complexity is considered moderate according to the BVD (Luxburg and Schölkopf, 2011). When f_j is based on statistical measures, usually the search space consists of a single function, making f_j more prone to underfitting. Furthermore, such a model is only effective to test particular hypotheses, *e.g.*, when data is statistically stationary (which we claim it is unlikely to happen when dealing with real, nonlinear, and/or chaotic datasets (Kantz and Schreiber, 2004)). Alternatively, when f_j is inferred based on Dynamical System approaches, the model usually relies on the distances among phase states and their neighbors inside the open-ball radius ε (Equation 2.7). In this context, small values of ε typically overfit as f_j basically memorizes each state. Conversely, excessively large radii make the model to learn from the attractor average, leading to underfitting (de Mello and Moacir, 2018). Thus, a balanced-complexity model should be based on a fair and adaptive percentage of distances among states, *e.g.*, ε can be defined in terms of the k-nearest neighbors or as some quantile over the maximum distance of states (see Section 5.3). Regarding the indicator function g, the comparison between windows should follow some strategy as the one defined in Equation 7.5, otherwise simpler functions would lead to underfitting and more complex indicators to overfitting.

In summary, the requirements to use SLT in CD scenarios are:

R1. the indicator function g should be updated based on past data, so that the underlying phenomenon is better represented;

R2. the model f_j must receive i.i.d. data, something to be ensured by a pre-processing step (*e.g.*, Fourier transform or phase-space reconstruction);

R3. the function g should compare features from the same JPD. When a different phenomenon is detected, a reset of g is necessary;

R4. the algorithm bias from both g and f_j should moderate, following the BVD.

7.4 ANALYZING STATE OF ART IN CD ALGORITHMS

As discussed in Section 7.2, a CD algorithm has two components: the first extracts features from data windows, using function f_j; and the second compares those features using some indicator function g. Following this structure, we present state-of-the-art CD algorithms and highlight how they approach requirements R1–R4. In this discussion, we do not cover CD classification methods (Klinkenberg and Joachims, 2000; Mena-Torres and Aguilar-Ruiz, 2014; Loo and Marsono, 2015; Jedrzejowicz and Jedrzejowicz, 2015; Krawczyk and Woźniak, 2015; Angel et al., 2016), given they rely on *explicit* labeling information provided by external specialists. Also, we do not cover algorithms that only address the optimization of processing costs (Hulten et al., 2001; Gama et al., 2004b), as they take a far different and more empirical perspective on CD detection comparesd to our more formal approach.

Several algorithms have been proposed to identify data sampled from changing phenomena (Gama et al., 2014). We discuss next several well-known algorithms in this collection.

Cumulative Sum: The Cumulative Sum (CUSUM) (Page, 1954) algorithm reports a drift whenever an incoming observation is significantly different from the sum of past data. Thus, knowing that g_s is initially set to zero, a drift occurs when

$$g_t = \max(0, g_{t-1} + x(t)) \geq \lambda, \tag{7.9}$$

in which λ is an acceptable threshold and $x(t)$ consists of a single observation, so that $W_j = x(j)$ (window length $n = 1$). In this scenario, $f_j : x(j) \to x(j)$ and $\phi(f_j) = x(j)$ corresponds to the identity function while g_t is a model directly correlated to the average of such a phenomenon. A drift is reported when g_t results in a value larger than the threshold λ, and $g_{s=t}$ resets the analysis for a new phenomenon (satisfying R3). If negative values are considered, $\min(\cdot)$ is used instead of $\max(\cdot)$ in Equation 7.9 and drifts are triggered when g_t is smaller than λ. In summary, CUSUM respects R1 as g is updated as new data arrives. However, f_j infers a model based on the time-series observations, not satisfying R2. Lastly, R4 is not satisfied as f_j overfits (memorizing the current observation) and g underfits (too restrictive bias) data since a single cumulative linear model may not be enough to represent more complex behavior.

Page-Hinkley Test: The Page-Hinkley Test (PHT), also proposed by Page (1954), is a variation of CUSUM (using the same window configuration) in the sense it assesses data changes in terms of standard-deviation measurements rather than its averages. Thus, given the average estimation $\mu_t = 1/(t-s)\sum_{j=s}^{t} x(j)$, where interval $[s,t]$ representing the evolution of some phenomenon from the start (s) to the current window (t), PHT reports a drift whenever

$$g_t = |m_t - M_t| > \lambda, \tag{7.10}$$

where $m_t = \sum_{k=s}^{t}(x(k) - \mu_k)$, $M_t = \min(m_{[s,t]})$ and $|\cdot|$ is the absolute-value norm. In other words, a drift occurs whenever the difference between the cumulative standard deviation is λ units greater than the minimum standard deviation observed up to the current moment. Similarly to CUSUM, g is updated as new windows are processed and a reset occurs when a drift is issued, so that both R1 and R3 are satisfied. However, since m_t is computed over a time-dependent sequence of observations, R2 is not respected. In addition, despite PHT is slightly more complex than CUSUM, it is still prone of overfitting (failing R4).

Adaptive Sliding Window: The Adaptive Sliding Window (ADWIN) method, proposed by Bifet et al. (2009), also comprises an extension of CUSUM, but applied to different window configurations. The data stream $\mathcal{D}$ is divided into two adaptive windows $W_{[s,k]} = \{x(s), \cdots, x(k)\}$ and $W_{[k+1,t]} = \{x(k+1), \cdots, x(t)\}$. In this context, ADWIN reports a drift whenever

$$g_k = |\mu_{W_{[s,k]}} - \mu_{W_{[s+1,t]}}| > \lambda, \quad \forall\, k \in [s,t), \tag{7.11}$$

where $\mu_{W_{[a,b]}}$ is the average of $W_{[a,b]}$. As soon as a drift is issued, then $s = t$ in order to reset the past model and represent a new phenomenon (respecting R3). However, the algorithm just compares averages between consecutive windows, taking no advantage from past data to update g (thus, R1 is not satisfied). Further, as f_j is inferred directly from data stream observations, R2 is not respected either. Lastly, despite the search space of g is larger than the ones considered by CUSUM and PHT (more windows are taken into account), the usage of an average model $f_j : W_{[a,b]} \to \mu_{W_{[a,b]}}$ and the fact that g is too simplistic, make the whole algorithm still prone to underfit.

Unidimensional Fourier Transform: Vallim and De Mello (2014) proposed the Unidimensional Fourier Transform (1DFT) to infer the model $f_j : \boldsymbol{C}_j \to \boldsymbol{C}_j$, in which $\boldsymbol{C}_j = [c_{j,1}, \cdots, c_{j,(n-1)/2}]$ is the vector of Fourier coefficients (Bracewell, 1978) on W_j, defined as

$$c_{j,k} = \frac{\epsilon}{n} \sum_{c=0}^{n-1} x(jn+k) e^{-ik2\pi \frac{c}{n-1}}, \quad \epsilon = \begin{cases} 1, & j = 0, \\ 2, & j > 0, \end{cases} \tag{7.12}$$

where i is the imaginary unit and $0 \leq j \leq (n-1)/2$. In this context, $\phi(f_j)$ is the identify function and g reports a drift when

$$g_t = \|\boldsymbol{C}_{t-1} - \boldsymbol{C}_t\|_2 > \lambda, \tag{7.13}$$

from which we conclude R1 is not satisfied as g simply compares two consecutive windows, so that nothing is learned from past data. R3 is automatically respected since g requires no reset. Moreover, the method fulfills R2, as Fourier coefficients are independent from each other. Lastly, this method is less prone to underfitting as the Fourier coefficients better represent the data than averages and standard deviations. However, despite improving R4, this requirement is only partially fulfilled as f_j still memorizes data and g might be ambiguous, since completely different sets of coefficients may lead to similar Euclidean distances.

Cross Recurrence Concept Drift Detection: da Costa et al. (2016) proposed the Cross Recurrence Concept-Drift Detection (CRCDD) algorithm, which compares two phase spaces using the Cross-Recurrence Analysis (Marwan et al., 2007; Marwan and Webber, 2015). Initially, the embedding pair (m, τ) for the first window W_s is estimated using the FNN (Section 4.2.2.1) and AMI (Section 4.2.1.2). Such a pair is then assumed for all remaining windows until a drift is reported. Next, the method maps each window to the phase space to quantify the difference between consecutive embeddings using the Maximum Diagonal Length (MDL) (Section 6.4), *i.e.*, the diagonal with maximum length represented by consecutive values equal to 1 in $\boldsymbol{R}$ (Equation 6.7). Assuming the current window W_t is represented by time series T_i, the inferred model $f_t : \boldsymbol{\Phi}_i \to \boldsymbol{\Phi}_i$ respects R2, as states are i.i.d. in the phase space (Chapter 5). Moreover, $\phi(f_j)$ is the identify function and g has the form

$$g_t = \max(\boldsymbol{Q}_t) > \lambda, \tag{7.14}$$

where $\boldsymbol{Q}_t$ (details in Equation 6.9) is the penalized CRP comparing the phase spaces from W_t and W_{t-1}.

As a result, R1 is not fulfilled as no knowledge is accumulated from past observations (R3 is automatically satisfied). As the matrix $\boldsymbol{R}$ is computed using an open ball with radius set to the average of the maximum distances from all $\log N_i$-nearest neighbors of each phase state, R4 is respected for g, as the algorithm bias is adapted to the size of the input data. However, R4 is not satisfied for the memory function f_j.

Multidimensional Fourier Transform: Finally, the authors of CRCDD also proposed the Multidimensional Fourier Transform (MDFT) (da Costa et al., 2017) to compare the Fourier coefficients from phase spaces. Their method uniformly partitions each axis of an m-dimensional phase space into n bins forming a grid, so that n^m cells are created. Then, the Fast Fourier Transform is computed along each grid dimension, yielding the multidimensional complex coefficients for each data window. Singular Value Decomposition (SVD) is then applied on the coefficients of each window to obtain the eigenvalues, which provides information about data variances along each space dimension. Eigenvalues from the previous window $\{\lambda_{t-1,1}, \cdots, \lambda_{t-1,m}\}$ and the ones obtained for the current window $\{\lambda_{t,1}, \cdots, \lambda_{t,m}\}$ are then compared by

$$\lambda_c = \frac{|\lambda_{t-1,c} - \lambda_{t,c}|}{\max(\lambda_{t-1,c}, \lambda_{t,c})}, \tag{7.15}$$

which measures the relative distortions for each dimension on both phase spaces. From that, the Von Neumann's entropy (Han et al., 2012)

$$E_{\text{vn}}(t) = -\sum_{c=1}^{m} \lambda_c \log \lambda_c, \tag{7.16}$$

is computed and used as criterion to identify drifts along time, so that

$$g_t = E_{\text{vn}}(t) > \lambda. \tag{7.17}$$

In summary, the algorithm is based on the model $f_j : \boldsymbol{\Phi}_i \to \boldsymbol{C}_j$, such that the feature vector $\phi(f_j) = E_{\text{vn}}(j)$ is given to g alert drifts based on past entropy values (see Equation 7.17). R1 is satisfied as knowledge is accumulated from past observations. However, R3 is not fulfilled given that no reset is considered when g detects drifts. As the input data of f_j is ensured to be i.i.d., R2 is fulfilled. Lastly, R4 is respected as f_j has enough information to represent each window.

More CD algorithms exist in the literature (Gama et al., 2004a; Baena-García et al., 2006; Wang et al., 2013; Bifet et al., 2010, etc.). Detailing and analyzing all of them here is out of our scope. Rather, our analysis outlined above showed how a relevant set of well-known CD algorithms can be compared against SLT criteria. The results of this comparison are summarized in Table 7 in which: (i) column "Update" shows if the indicator function g is updated according to past data (R1); column "IID" means some space transformation is performed to ensure f_j is inferred from identically and independently distributed data (R2); column "Fixed JPD" informs whether the algorithm resets g whenever a drift is detected (R3); column "BVD(f_j, g)" depicts if the spaces of admissible functions (a.k.a. algorithm bias) of both g and f_j are in accordance with the Bias-Variance Dilemma (R4). The interested researcher can extend this table by considering additional algorithms.

Table 7: Comparison of CD algorithms *vs* requirements R1–R4.

Method	Update (R1)	IID (R2)	Fixed JPD (R3)	BVD(f_j, g) (R4)
CUSUM	Yes	No	Yes	(No, No)
PHT	Yes	No	Yes	(No, No)
ADWIN	No	No	Yes	(No, No)
UDFT	No	Yes	Yes	(No, No)
CRCDD	No	Yes	Yes	(Yes, No)
MDFT	Yes	Yes	No	(Yes, Yes)

As summarized in Table 7, CRCDD and MDFT have the strongest learning guarantees, meaning that their drifts are most likely to be the result of actual changes in data behavior rather than by chance. However, no single algorithm fulfills *all* criteria. Hence, formally, we cannot state, for any of the analyzed algorithms, that they will detect actual drifts in a hard sense of the word.

7.5 FINAL CONSIDERATIONS

This chapter proposes a methodology to overcome the complexity involved in labeling data streams and the lack of theoretical learning guarantees in Concept-Drift (CD) scenarios, therefore **answering research question 3 (RQ3).** More precisely, a CD algorithm can rely on the SLT framework to ensure learning when it meets the following requirements:

R1. given that features from f_j are extracted using function ϕ, then the indicator function g must compare past against current features and, in case no drift is issued, it should be updated to improve the representation of the current phenomenon;

R2. window observations should be reconstructed into another space in order to ensure data independence and allow i.i.d. sampling. Among alternatives, we suggest to map them into phase spaces, using dynamical system tools, to automatically define spaces $\mathcal{X}_j$ and $\mathcal{Y}_j$. As discussed in Section 2.6, several features derived from such space enhance the quality of chaotic series (data streams);

R3. if a drift is confirmed, then the model g should be reset to start the analysis of a new phenomenon based on fixed JPD;

R4. the biases of both g and f_j should respect the BVD to avoid under/overfitting.

We analyzed several state-of-the-art CD algorithms against these requirements. Strikingly, none of them fulfilled them *all*, which means that all such algorithms are prone to some extent to detect drifts which are not actually existing in the data. Relatively speaking, the MDFT and CRCDD algorithms provide the strongest learning guarantees among the analyzed ones. Therefore, they are most likely to detect actual changes in data behavior rather than issue drifts by chance.

We expect this analysis to be helpful to other researchers who intend to design new CD algorithms or evaluate the existent ones. As future work, we envisage proposing new measures to evaluate the performance of CD algorithms taking all four requirements into account. This will arguably increase the quality of such evaluations and comparisons beyond what is provided by currently used metrics such as Mean Time Between False Alarms and Mean Time for Detection.

8 RADIAL VISUALIZATIONS FOR HIGH-DIMENSIONAL DATA

8.1 INITIAL CONSIDERATIONS

The previous chapters of this thesis have shown that time series can be represented and analyzed both in the *time domain* and, alternatively, in *phase space*. As discussed in detail, the phase-space domain has several advantages, ranging from the ability to reason about high-level features such as orbits and attractors, to more technical points such as the ability to construct accurate classifiers and regressors. However, one main challenge of using phase spaces is that they are both *abstract* and *high-dimensional*. Hence, practitioners may have significant trouble in understanding data represented in such spaces. This brings us to formulating our research question:

RQ4. "How to correlate time-series and phase-space attributes?"

In this chapter, we take a different approach to answer our current research question, as compared to previous chapters. Rather than using the machinery provided by automated analysis (such as classifiers) or reasoning about Dynamical Systems on a theoretical level, we now turn to the problem of enabling users to actually *see* their data at hand. Designing techniques and tools to visualize high dimensional data is of growing interest to many communities in data science and Machine Learning. Overall, such tools do not replace, but complement, automated analyses and theoretical examination.

High-dimensional data visualization is an active area of research (Van Leeuwen and Jewitt, 2000), with many types of techniques being offered. However, no such technique can successfully present both data *dimensions* (especially when these are many) and highlight similarity patterns between data *observations* equally well.

In our quest to push the state of the art in high-dimensional data visualization, we chose as starting visual metaphor the so-called *radial layout* (Bertini et al., 2005), which is well known and accepted in practice, simple to implement, and addresses the visualization of both instances and dimensions simultaneously. However, such layout is far from optimal, struggling with ambiguity and scalability problems (discussed along this chapter). Then, based on several requirements to which radial-based visualizations should comply, we proposed technical and algorithmic improvements to satisfy them. As consequence, we developed a novel improved visualization solution, from which we validate it on several real-world high-dimensional datasets, not directly related to time-series analysis. As consequence, we created a generic visualization metaphor that can be applied to any dataset consisting of a set of observations with several dimensions (measurements) per observation. Nonetheless, we demonstrate that this visualization could be employed in the context of Dynamical Systems.

The structure of this chapter is as follows. Section 8.2 introduces high-dimensional data visualization and visual analytics, and outlines the place of radial visualization techniques and their requirements. Section 8.3 elaborates on the above presenting the state-of-the-art in radial visualizations, including their strengths and limitations. Section 8.4 details our visualization, called RadViz++. Section 8.5 demonstrates RadViz++ on several real-world datasets. Section 8.6 discusses our proposed technique. Section 8.7 shows another visualization to explore time-series embeddings. Finally, Section 8.8 concludes this chapter.

8.2 BACKGROUND ON VISUAL ANALYTICS

Methods to study multidimensional datasets are a core topic in Visual Analytics (Van Leeuwen and Jewitt, 2000). Analyses supported by such methods can be divided into three classes: (i) data-to-data; (ii) data-to-variable; and (iii) variable-to-variable. The first type of analysis generally consists of Dimensionality Reduction (DR) methods that project data into a low-dimensional space to visually search for clusters and patterns (Nonato and Aupetit, 2018). While aiming to preserve data-to-data relationships, DR methods by themselves do not explain the *variable* space or, *e.g.*, which variables impact the projection the most – doing this requires additional visual metaphors (Silva et al., 2015; Coimbra et al., 2016; Pagliosa et al., 2016). On the other hand, methods like Parallel Coordinate Plots (Inselberg, 2009) and Scatterplot Matrices (Telea, 2014) help to perform data-to-variable analyses, but problems such as visual clutter (excess and overlapping of components) and limited usability tend to occur when tens of variables or more are analyzed, hindering data-to-data correlation. Lastly, histograms and box-plot-based metaphors (McGill et al., 1978) can show distributions and similarities of variables, but are also limited for high-dimensional data as a large visual space is required to fairly compare several variables.

Overall, most high-dimensional visualization methods are mainly designed to tackle one (two, at most) type of analysis. Conversely, the Radial Visualization (RadViz), originally proposed by Hoffman et al. (1997), is one of the most popular techniques (Bertini et al., 2005) that perform all three types (i–iii) simultaneously. In this metaphor, each variable is mapped as an anchor along the circle such that data instances (represented as 2D points) are pulled towards them according to their respective variable values. In this context, while data information can be extracted by analyzing the formation of clusters and outliers inside the circle (data-to-data), those patterns can be explained by the proximity of data points to the anchors (data-to-variable). In addition, variables are correlated according to their distance or the order they appear in the circle (variable-to-variable).

Despite benefits, however, RadViz-class methods can also lead to misconceptions and clutter when different instances are mapped into the same visual location. These so-called *ambiguities* (Bertini et al., 2005; Rubio-Sanchez et al., 2015), the dependency to anchors positioning, and the limited space in the circle contribute to a generally lower ability to separate same-data-sample clusters than *e.g.*, DR methods (Nonato and Aupetit, 2018). In this context, methods in the literature (Section 8.3.2) proposed to optimize how anchors are ordered in the circle, but solutions are still restricted for a relatively small number (few tens) of variables. In summary, we identify the following possible improvements for RadViz-class visualizations:

R1. be scalable in both the number of variables and instances;

R2. decrease and/or explain visual ambiguities they create in data-to-variable analyses;

R3. show unambiguously variable relations to support variable-to-variable analyses;

R4. separate data clusters well to support data-to-data analyses.

Based on those requirements we propose RadViz++, a novel RadViz-class technique to support tasks (i-iiii) while better satisfying R1-R4. We order variables along the circle following the hierarchical clustering based on variable correlations, and draw clusters compactly using an icicle-plot metaphor (Kruskal and Landwehr, 1983). Scalability is addressed by allowing users to interactively aggregate and/or filter out variables while exploring how this changes data-to-data insights. We add histograms over each icicle-plot cell to show its respective variable distribution. Besides showing this, one can select histograms bins to filter data based on ranges of multiple variables. Conversely, we use a brushing-and-linking metaphor to select data points and explain them by their respective variable bins, thereby decreasing ambiguity issues. We use an edge-bundling technique (Holten, 2006) to show strongly correlated variable anchors, thereby clarifying variable-to variable relations. Finally, we allow smoothly animating between the RadViz scatterplot and a classical DR scatterplot to let users link cluster (best shown by the latter) by variables that explain them (best shown by the former).

8.3 RELATED WORK

We firstly describe the fundamental concepts and problems of RadViz-class visualizations. Next, we present how state-of-the-art methods tackled those issues, and where they can be improved.

8.3.1 *Concepts And Background*

Following the nomenclature of RadViz Deluxe (Cheng et al., 2017), consider a multidimensional dataset, represented in matricial form as

$$\boldsymbol{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,n} & x_{2,2} & \vdots & \vdots \\ \ddots & \ddots & \cdots & \vdots \\ x_{m,1} & x_{m,2} & \cdots & x_{m,n} \end{bmatrix}, \tag{8.1}$$

where m and n are the number of instances (also called samples or observations) and variables (also called attributes, dimensions, or features), respectively. In this context, a RadViz-class visualization (Nováková and Štěpánková, 2011) maps the variables $V_1, \cdots, V_n$ (class of each column in $\boldsymbol{X}$) to so-called *anchors* $v_1, \cdots, v_n$ on the circle boundary (with radius r) as

$$v_j = \left(r \cos \frac{(j-1)2\pi}{n}, r \sin \frac{(j-1)2\pi}{n} \right), \tag{8.2}$$

so that instances D_i (rows of X) are represented by points P_i according to

$$P_i = \sum_{j}^{n} \frac{x_{i,j}}{\sum_j^n x_{i,j}} v_j. \tag{8.3}$$

In this context, P_i is "pulled" towards the anchors v_j proportionally to the its positive value x_{ij} (Figure 8.1). If we use the same logic, P_i should be repelled by the same force if negative values were allowed. Yet, this would be misleading, since repelling a point from an anchor v_j inevitably pushes it to some other anchor along the circle boundary, opposite of v_j. Separately, normalization of Equation 8.3 is needed to ensure all points are mapped inside the circle, which is not guaranteed when negative V_j values are involved. Therefore, negative values are usually handled by either normalizing V_j to $[0, 1]$ or taking their absolute values. However, properly normalizing is hard as the proportionality of variables over instances can be lost.

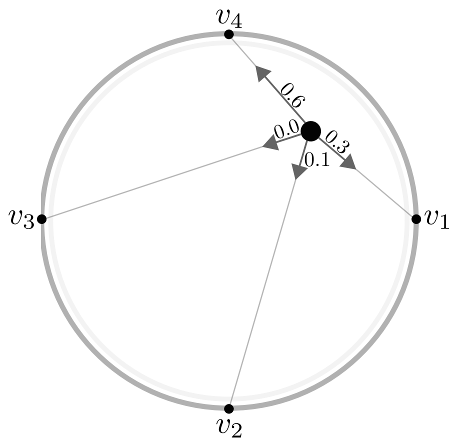


Figure 8.1: An instance is pulled towards the anchors proportionally to its normalized variable values. Adapted from Pagliosa and Telea (2019).

Nonetheless, visual ambiguities are still a problem even after the above steps, see instances D_1, D_2, and D_3 in Table 8, for instance. As we can see, normalizing the variables of those three instances (to remove negative values) maps D_1 and D_2 to the same point. Similarly, D_2 and D_3 get overlapped if absolute variable values are used. Thus, the most common form of ambiguity in RadViz-class methods occurs when points are "pushed" to the circle center (even when only positive values are used), either because instances have equal variable values (D_4) or when a subset of anchors is placed so that their forces cancel each other (D_5, D_6). To alleviate this, several methods try to optimize anchor placement and how points are attracted to them (Section 8.3.2). Yet, inconsistencies will eventually occur, especially when the number of variables increases. This is due to the inherent limits of the circular space along which anchors are placed. Due to these limits, we need ways to *disambiguate* different instances that get mapped at similar locations.

Table 8: Different inconsistencies that can occur in RadViz. Instances D_1, D_2, D_3 get mapped to the same point after procedures to avoid negative numbers. On the other hand, instances D_4, D_5, D_6 show the typical sensitivity to anchor positioning in RadViz designs.

Instance	V_1	V_2	V_3	V_4
D_1	0	-20	-60	-60
D_2	20	40	80	80
D_3	2	4	8	8
D_4	0	0	0	0
D_5	20	1	20	1
D_6	100	5	100	5

8.3.2 *Related Methods*

As cyclic ordering is an NP-complete problem (Ankerst et al., 1998), several heuristics were suggested to optimize circular anchor placement to decrease ambiguities. For example, to separate different instances that get overlapped in a classical RadViz plot, Nováková and Štěpánková (2009) propose a 3D RadViz design where instances are drawn into the xy plane via Equations 8.2 and 8.3, while their norms are mapped to the z axis. This addresses R2, as instances like D5 and D6 (Table 8) can be distinguished by their heights while viewing the 3D layout from different viewpoints. Yet, this does not tackle scalability (R1) nor support deeper variable-to-variable analysis (R3). Moreover, finding a suitable 3D viewpoint can be hard, as even with the z map clutter might be formed in that dimension.

The Mean Shift (MS) method (Zhou et al., 2015) partitions each variable into several new variables according to its probability distribution function. The procedure repeats for each variable as follows. First, the distribution of V_j is discretized into a histogram of p bins, whose density values are interpreted as 1D points. These p points are clustered by a Gaussian-based technique that maps points to the centroid of their neighbors (all points inside the kernel bandwidth). After all bins converge to a centroid, V_j is partitioned into new variables according to each centroid interval. Moreover, V_j is removed from the visualization and the new variables added. Finally, all variables are placed along the circle to optimize the Dunn index (Dunn, 1974), exhaustively calculated for all possible combinations of anchor positions. This method can be seen as an extension of Vectorized RadViz (VRV) (Sharko et al., 2008), proposed to analyze categorical data. Both MS and VRV aim to decrease ambiguities (R2), higfight interval-basis similarities among variables (R3), and aim to better cluster the data (R4). Yet, both methods have an even lower dimension-scalability (R1) than classical RadViz as they need to accommodate more variables (from the partitions) in the visual space.

Ono et al. (2015) propose Concentric RadViz (CRV), an interactive tool for multitask classification. Variables are clustered according to their tasks into concentric circles following (Di Caro et al., 2010). Sigmoid normalization is applied to ensure all points remain inside the circle, even when nested anchors are aligned. Users can rotate anchors in any direction and at any level to analyze the formation of patterns and correlate instances over multiple tasks. CRV can also be seen as an extension of Star Coordinates (SC) (Kandogan, 2000), where users can rotate and scale anchor positions at will, starting from an initial equally-distributed anchor placement along a single circle. Both CRV and SC handle well datasets with a few tens of variables. For more variables, the interactive search for a good anchor placement becomes hard as there is no visual cue to guide users during this search (limited R1 support). Moreover, both methods eventually lead to clutter even when multiple circles are used (problems with R1 and R2). However, structures (clusters) are potentially better represented after interactions (R4). Finally, R3 is partially addressed as users can correlate variables not only by their distances but also by how they are aligned in the circular hierarchy.

Also an extension of SC, iStar (Zanabria et al., 2016) is an interactive tool that, besides allowing traditional scale/rotate operations of the variable axes, also supports the union and separation of axis anchors at will, readjusting data points in real time. To support R1 for large numbers of variables, these can be clustered automatically by the k-means algorithm (Grira et al., 2004) based on their variance, bidimensional PCA coordinates, or centroids of classes (when class labels are present). Next, given the matrix M of variable-pairwise similarities, a graph is created where nodes are anchors and an edge connecting two anchors has its length defined by the pairwise similarities M_{ij}. The ordering of anchors around the circle is then given by the optimal closing path connecting all nodes, computed using a Genetic Algorithm (Wang et al., 2007). The distance between adjacent anchors is given by their edge length (similarity). Given their design, iStar axes are related to biplot axes, well known in information visualization (Gower and Hand, 1995; Greenacre, 2010; Gower et al., 2011). iStar supports R1 very well, showing dataset examples of hundreds up to thousands of instances and variables. Variable-to-variable analyses are also well supported by the proposed clustering (R3). However, setting the number of k clusters in k-means is no trivial task – this works well only if the user has *beforehand* a good idea of how many groups-of-variables he/she would like to simplify the data into, which similarity metric to use for the variables, and if the variables are indeed distributed this way in the data. Similarly, despite that iStar allows users to freely joint and split variable groups, there is no visual cue to *guide* this process, besides the formation of point-group structures in the plot *after* the respective user action was done. iStar does not tackle R2 as there is no visual metaphor provided to explain ambiguous points. Finally, iStar can achieve quite good cluster separation, as demonstrated on many datasets (R4). However, this requires careful user intervention in terms of selecting k, as well as manual anchor arrangement, grouping, and filtering.

From a different perspective, Rubio-Sánchez et al. (2017) proposed to use the user-defined anchor positions from SC to minimize $\left\|\boldsymbol{P}\boldsymbol{A}^T - \boldsymbol{X}\right\|_F^2$, where $\boldsymbol{A}$ is the $n \times 2$ matrix composed of 2D anchor vectors, $\boldsymbol{P}$ is the $m \times 2$ matrix containing the 2D coordinates of the scatterplot points, and $\| \cdot \|_F^2$ denotes the Frobenius norm. The authors also apply a kernel function to A to make its columns mutually orthonormal, which provides "a more faithful representation of the data since it avoids introducing distortions, and enhances preserving relative distances between samples". The above minimization improves R4 and partially fulfills R2, as there are no metaphors to explain data-to-variable analysis ambiguities. Finally, the method does not extend variable-to-variable analysis (R3) with new solutions, nor does it explicitly address dealing with large numbers of variables (R1).

Recently, the RadViz Deluxe (RVD) (Cheng et al., 2017) method aims to improve the quality of all analysis types (i) to (iii). RVD proposes different methods to reduce errors of the low-dimensional representation, namely variable-to-variable, data-to-variable and data-to-data errors, in this order, as follows. First, anchor placement along the circle is computed by an approximate Hamilton Cycle solution (Bollobás et al., 1987), so that distances between adjacent anchors reflect their pairwise correlations. Secondly, the data-to-variable error is decreased by a series of iterative geometrical operations. Finally, the data-to-data error is reduced by a spring system similar to (Tejada et al., 2003), where an instance D_i is attracted (respectively repelled) to D_j if their distance in nD space is smaller (respectively greater) than in the 2D visual space. Despite improvements regarding R2 and R4, RVD still lacks solutions for R1 (scalability) and R3 (variable-to-variable analysis). Moreover, RVD reduces errors following a fixed pipeline. Hence, it is likely that after changing the visualization to decrease one error (*e.g.*, data-to-data), other errors increase (*e.g.*, data-to-variable and variable-to-variable). Finally, let us recall that a main proposal of RadViz is to explain the projected data *and* their variables. Consider Figure 8.2(a), generated by RadViz. Here, anchors correctly describe (explain) data points. For instance, the black outlier point, close to anchor v_1, has variation only in variable V_1. This explanation is partially lost by RVD corrections, as data points are not strictly represented by anchors anymore. Consider Figure 8.2(b), generated by RVD. Point clusters are indeed better separated now. However, anchors cannot be used to reliably explain the points. For instance, the black outlier moved towards the center, which could give the wrong impression that it may also have positive values in V_2, V_3 or V_4. Figure 8.2(c) shows the difference between the first two figures.

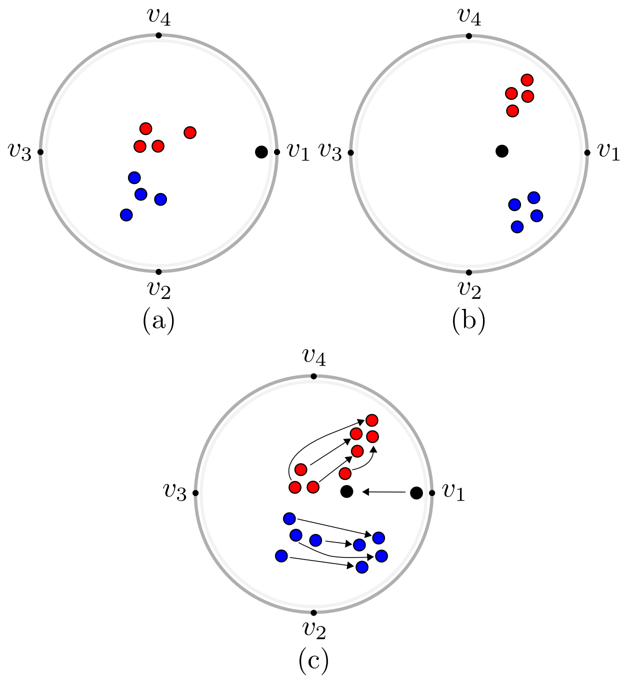


Figure 8.2: **(a)** RadViz representation of a simple dataset showing clusters (red and blue) and one outlier (black). **(b)** RadViz Deluxe layout of the same data showing better cluster separation but poorer explanation of the outlier. **(c)** Differences highlighted between **(a)** and **(b)**. Adapted from Pagliosa and Telea (2019).

8.4 RADVIZ++ PROPOSAL

To address the requirements listed in Section 8.2 and to alleviate the observed limitations of current methods, we propose RadViz++, a novel radial-based visualization for high-dimensional data. RadViz++ allows users to interactively aggregate, separate, and filter variables, and see in real time how this impacts the layout on a data-to-data, variable-to-variable and data-to-variable basis. We next introduce and explain the features of RadViz++ and outline how they address R1–R4 and also improve upon related RadViz-class methods. We use as running example the well-known Segmentation dataset (Joia et al., 2011; Martins et al., 2014; Dua and Karra Taniskidou, 2019), which has $m = 2100$ instances, $n = 18$ variables, and 6 instance classes. For conciseness, the variable names are next referred to as $V_1, V_2, \cdots, V_{18}$. Instances are randomly-chosen 3×3 pixel blocks from seven manually-segmented outdoor images. Variables are statistical image attributes, such as color mean, standard deviation, and horizontal/vertical contrast, often used in image classification. The class attribute denotes the image type. Visual analysis tools use this dataset to discover how specific sets of variables and/or variable ranges can explain the similarity of groups of points (Tung et al., 2005; Coimbra et al., 2016). In turn, this can help designing better feature-engineering-based classifiers for such data.

8.4.1 *Anchor Placement*

As a baseline, we show the results of the original RadViz method on the Segmentation dataset (Figure 8.3(a)). Here, anchors v_i are placed anticlockwise along the circle in the order their variables V_i appear in the dataset. Here and next, scatterplot points are color-coded on their class label. As visible, no clear cluster separation can be seen. Yet, we know that such a separation does exist (Tung et al., 2005; Joia et al., 2011; Martins et al., 2014; Coimbra et al., 2016). To see this separation, a better approach is to order anchors based on the similarity of their variables. Among many ways to compute this similarity, known in the time-series literature, *e.g.*, AMI (Fraser and Swinney, 1986), DTW (Berndt and Clifford, 1994; Ratanamahatana and Keogh, 2004; Müller, 2007), ARIMA (Box and Jenkins, 2015), we choose the Pearson correlation coefficient (Benesty et al., 2009), similarly to RVD, given its simplicity. Hence, the similarity of V_i with V_j is given by

$$\rho(V_i, V_j) = \frac{\mathrm{Cov}(V_i, V_j)}{\sqrt{(\mathrm{Var}(V_i) \times \mathrm{Var}(V_j)}}. \tag{8.4}$$

To obtain a similarity metric, we normalize ρ to $[1, 0]$. To place anchors, we next compute the all-variable-pairs distance matrix $A_{ij} = \rho(V_i, V_j), 1 \leq i \leq n, 1 \leq j \leq n$, and next cluster the variables V_i via average-linkage Agglomerative Hierarchical Clustering (AHC) (Rokach and Maimon, 2005). Figure 8.4(a) shows the cluster dendrogram produced for our running dataset. In contrast to

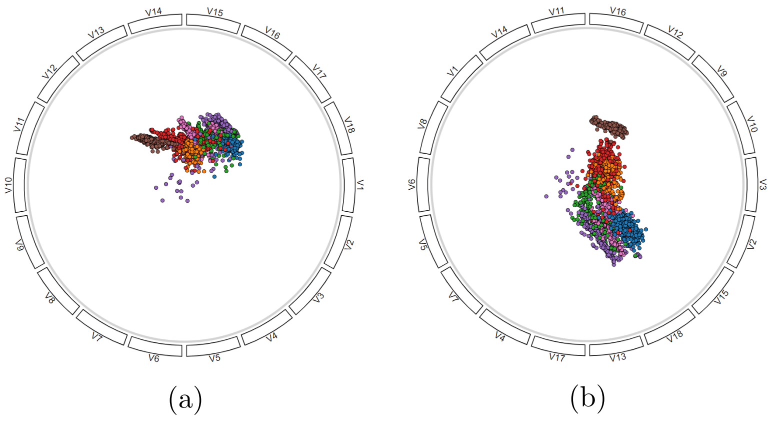


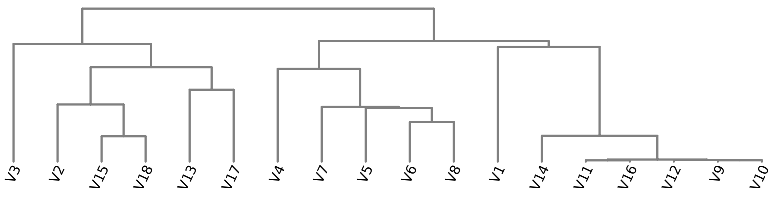
Figure 8.3: **(a)** RadViz with no variable ordering. **(b)** in RadViz++, anchors are rearranged in the circle according to their correlation coefficient. In our implementation, anchors are depicted by cells with the corresponding variable names above them, and points are colored based on their classes. Adapted from Pagliosa and Telea (2019).

RVD, we now arrange anchors around the circle in the order that leaves appear in the dendrogram (Figure 8.3(b)). As visible, clusters already get better separated than in the original RadViz layout (Figure 8.3(a)). In addition, we show in Section 8.4.4.1 how we can use the hierarchy to address scalability for many variables (R1), as well as to better cluster separation and explanation (R4).

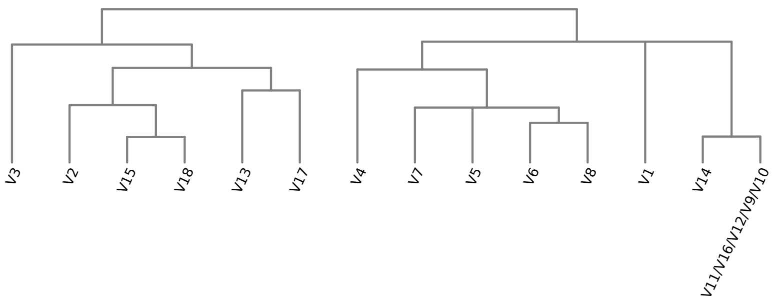
Despite this approach now leads to two clusters instead of one, it is still not enough to achieve an optimal representation. Regarding the distance among anchors, it is worth to mention that we *accept* the fact that, as dimensionality increases, it becomes more difficult to place all variables well separated in the circle according to their similarities. Therefore (and also in contrast to RadViz Deluxe), we make all anchors equally sized so neighbors have the same distance among themselves so that we can fit more variables in the same amount of visual space without clutter.

8.4.2 *Variable-To-Variable Analysis*

Atop of the hierarchical variable placement described in Section 8.4.1, we propose two visual metaphors to help variable-to-variable analysis in different levels, as follows.



(a)



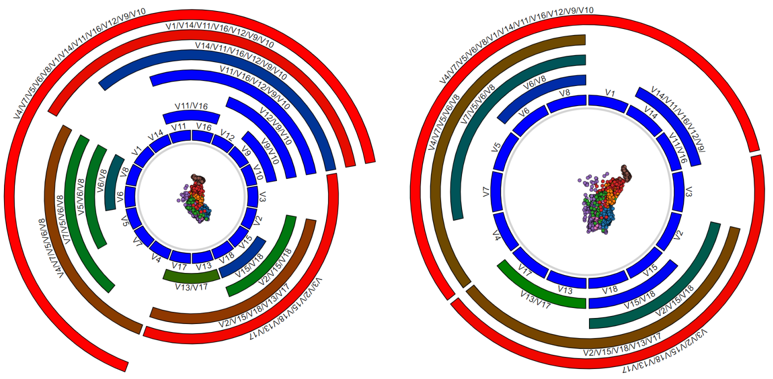
(b)

Figure 8.4: **(a)** Dendrogram built from variable correlation (Section 8.4.1). **(b)** Simplified dendrogram (Section 8.4.2.1). Adapted from Pagliosa and Telea (2019).

8.4.2.1 *Variable Hierarchy*

We draw the variable dendrogram using a circular icicle-plot metaphor where all leaves are aligned at the same level. A similar layout for hierarchical data was used by Holten (2006) for displaying different data types (software containment) and in a different context (program comprehension). As a key difference, icicle-plot cells in our case are groups of similar (correlated) variables, and not data instances. Cell colors indicate variable similarity using a blue-to-green-to-red (similar-to-dissimilar) ordered colormap. Labels atop cells show the variables these aggregate.

In this context, depicting the full dendrogram produced by AHC typically demands too much space in the visual plot, since each binary clustering event creates a new level. Figure 8.5(a) shows the resulting icicle plot for the dendrogram in Figure 8.4(a). Hence, we simplify the dendrogram by aggregating variables (by summing their respective values) having parents that are more similar than $\delta = 10\%$ of the root-cluster diameter. A similar approach was used in a different context by Carlsson and Mémoli (2013). Figure 8.4(b) shows the simplified dendrogram for our running example dataset. Thus, increasing this value yields a simpler dendrogram which needs less visual space, but shows less details on how variables relate to each other. Conversely, decreasing this value yields more details on the similarity values of variable pairs, but requires more visual space. Figure 8.5(b) shows the visualization of this simplified dendrogram.



(a) (b)

Figure 8.5: **(a)** Circular icicle plot showing the full dendrogram ($\delta = 0\%$). **(b)** Plot of the simplified dendrogram ($\delta = 10\%$) leading to a more compact layout. We acknowledge the difficulty to read labels, but they are not important for the given example (and others in the same format). Still, we keep them for consistency. Adapted from Pagliosa and Telea (2019).

8.4.2.2 *Similarity Disambiguation*

The icicle plot described above addresses the task of finding *groups* of similar variables, as children of the same node in the plot. However, the plot does not (easily) support the task of finding how similar a group of variables is to *other groups*. To see this, one needs to carefully study the entire icicle-plot hierarchy, including comparing the colors of multiple nodes. To support this task, we adapt the Hierarchical Edge Bundling (HEB) (Holten, 2006) technique as follows. We consider a graph G where each node is an anchor v_i, and each edge is the similarity $\rho(V_i, V_j)$ between variables V_i and V_j. We then construct the HEB bundling of G, using as hierarchy the one given by the (simplified) dendrogram, to draw it such each edge has ρ encoded into its opacity. Figure 8.6 shows the result. The less correlated two variables are, the more transparent and closer to the circle center will be its bundled edge. Conversely, strongly correlated variables will have dark (opaque) and far-from-center bundled edges. Bundles thus show groups of variables which are similar to each other.

Bundling serves an additional disambiguation task. As explained in Section 8.4.1, for a sufficiently large variable count n, it becomes hard, and in the limit impossible, to assign positions for the anchors v_i along a circle so that their distances *accurately* reflect high-dimensional similarities of the variables V_i, no matter which anchor placement strategy we use. This is the well-known distance preservation problem in dimensionality reduction when going from n dimensions to a single one. Moreover, the *circular* nature of RadViz designs will place variables which are at opposite ends in the (simplified) cluster tree (V_3 and (V_{11}, V_{16}) in Figure 8.4(b)) next to each other along the circle (Figure 8.6). The same happens for variables V_4 and V_{17}. Without any other visual cue, one may think that these are very similar variables. The HEB bundles solve this: as no dark bundle connects those cells in Figure 8.6, their respective variables *are not* similar.

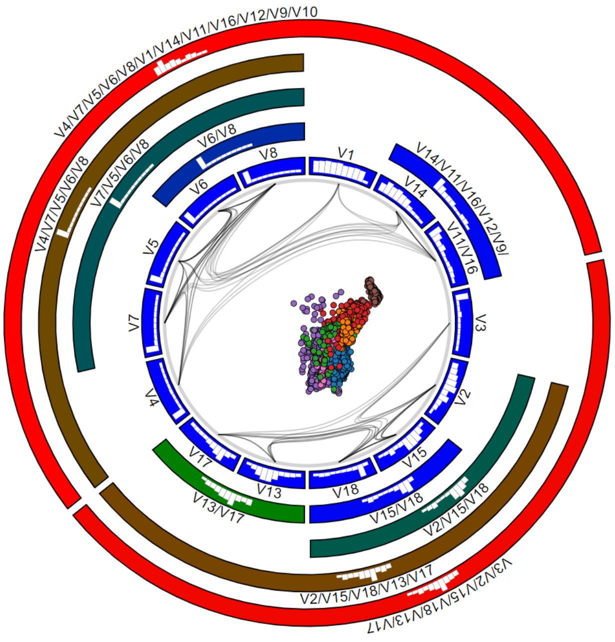


Figure 8.6: HEB bundles and variable histograms in RadViz++. Adapted from Pagliosa and Telea (2019).

8.4.3 *Analyzing Variable Values*

The mechanisms discussed so far show us which variables are similar, but they do not explain in detail why. Moreover, one is often interested in explaining the similarity of instances not only in terms of entire variables, but *ranges* of values thereof. To support such tasks, we plot histograms over each icicle-plot cell to show the respective variable distributions. By default, we use $h_{def} = 10$

histogram bins. However, icicle-plot cells can have widely different sizes, depending on the dendrogram clustering and total number of variables. For over a few tens of variables, some cells become too small to display 10-bar histograms. Varying the visual width of a histogram bar on the cell size is not a good idea, as it makes comparing histograms in different-width cells hard. Hence, we fix the width of a histogram bar to w_h, set in practice to 5 pixels, and use $h = \min(h_{def}, w_c/w_h)$ histogram bins for a cell of width w_c. This way, smaller cells will display fewer-bin histograms (see *e.g.*, Figure 8.6).

Besides seeing the value distributions of each variable V_i, histograms have two other uses. First, they allow comparing different variables. For instance, in Figure 8.6, we see that V_5, V_6, V_7, and V_8 are strongly correlated (since linked by bundles and children of a grandparent node colored dark-green), *and* they also have very similar distributions, with mostly small values. In contrast, nodes V_{15} and V_{18} show a similar correlation (same dark blue color), but quite different distributions. Secondly, histogram bars can be interactively clicked to select points whose values belong to the selected bins. By doing this, the user can either explore which variable *ranges* are responsible for certain patterns in the scatterplot, as well as to de-clutter scatterplot areas where multiple points are plotted atop each other.

8.4.4 *Scalability And Level-of-Detail*

We address both these issues by aggregating and filtering variables and data points, as follows.

8.4.4.1 *Aggregating Variables*

The key purpose of the icicle plot is to show how the data can be explained in terms of *groups* of similar variables. In the case when the user decides that all child variables of a parent node in this plot can be seen as a single one, displaying all of them makes the visualization unnecessarily verbose. Clicking such a parent node aggregates all its children variables, replacing them with the centroid value of the respective AHC cluster, and regenerates the visualization. Figure 8.7(a) shows this after we aggregate variables $V_2, V_{13}, V_{15}, V_{17}, V_{18}$ (large brown cluster, Figure 8.6 bottom); variables V_6, V_8 (Figure 8.6, top-left blue cluster); and variables $V_9, V_{10}, V_{11}, V_{12}, V_{14}, V_{16}$ (Figure 8.6, top-right blue cluster). The former aggregation, however, leads to more overlap in the scatterplot – hence, this simplification level may be too strong to allow us to correctly interpret the data. To fix this, we do one step backward by clicking on the large brown cluster in Figure 8.7(a) to split it into its direct children. The result (Figure 8.7(b)) shows a very similar scatterplot to the original, unaggregated, one (Figure 8.6). This plot is obtained by using only *nine* variables (either original ones or aggregations) as compared to the original 18. Hence, we obtain a 50% dimensionality reduction with little loss of the data structure.

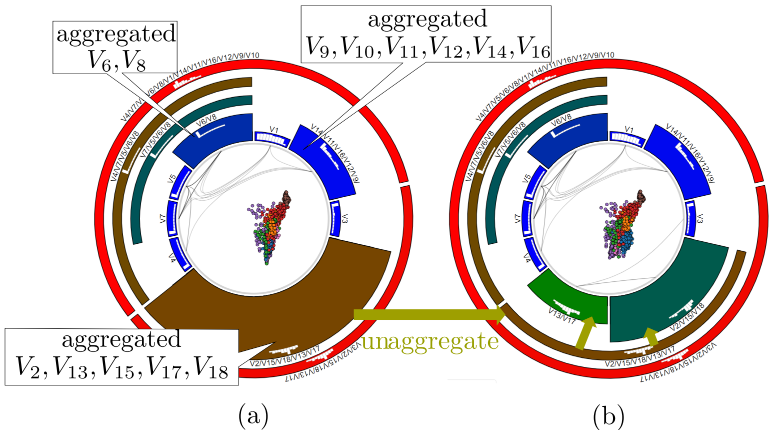


Figure 8.7: **(a)** Aggregation of several variables. **(b)** Refining the aggregation for the bottom (brown) cluster. Adapted from Pagliosa and Telea (2019).

8.4.4.2 *Variable Filtering*

While useful, variable aggregation has the problem that it actually *synthesizes* new variables from existing ones. This is not always desirable, *e.g.*, when certain variables do not logically make sense to be averaged together. Conversely, there are cases when we want to completely eliminate, or *filter* away, certain variables, *e.g.*, which we recognize as not useful for the analysis. By clicking icicle-plot cells the user can also filter away desired variables, after which the remaining space is reallocated to the remaining variables. Figure 8.8 illustrates this. First, we decide that only variables in the colored cells should remain after filtering (Figure 8.8(a)). We filter away all other variables, keeping only 11 of the original 18, and obtain the layout in Figure 8.8(b). The colors of the remaining cells change to reflect the range of similarities present in the recomputed dendrogram after filtering.

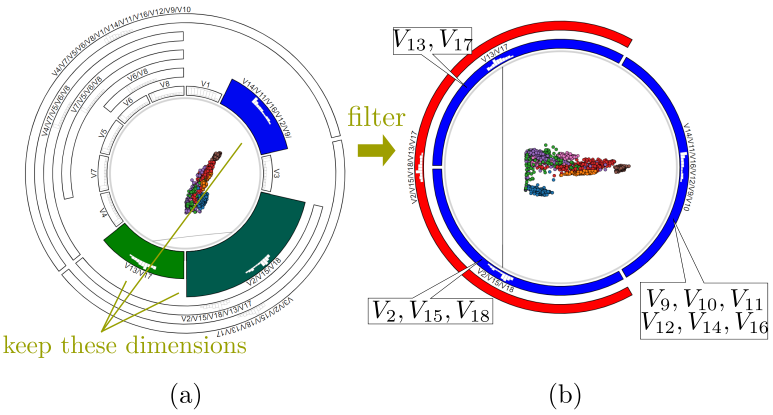


Figure 8.8: **(a)** Variables to filter away (white). **(b)** RadViz++ after variable filtering (using 11 of the original 18 variables). Adapted from Pagliosa and Telea (2019).

8.4.5 *Data-To-Data And Data-To-Variable Analysis*

As mentioned in Section 8.2, while RadViz-class methods are desirable when one wants to explore both instances and variables simultaneously, other dimensionality reduction (DR) methods exist. State-of-the-art methods, like the Local Affine Multidimensional Projection (LAMP) (Joia et al., 2011) and the t-Distributed Stochastic Neighbor Embedding (t-SNE) (van der Maaten and Hinton, 2008), achieve in general a (much) better similar-point cluster segregation, which is an important data-to-data analysis task (Nonato and Aupetit, 2018). However, such methods do not provide ways to explain how *variables* determine such clusters.

We combine the strengths of the RadViz metaphor (seeing both instances and variables, explaining instances by variables) and DR projections (better cluster segregation) by displaying in the inner circle scatterplots created by any such DR methods, instead of the force-based RadViz one. To explain projected groups, we next allow users to smoothly *animate* the DR scatterplot towards the RadViz scatterplot and vice-versa. This way, one can visually focus on a point group, clearly shown in the DR scatterplot, then see where the group goes in the RadViz scatterplot (following the animation), and finally use RadViz++ mechanisms to explain the points. Figure 8.9 shows several frames from the animation between RadViz and LAMP scatterplots for the Segmentation dataset. Using animation to link different displays of the same data, in particular merging insights obtained from different types of DR projections (Kruiger et al., 2017), but also for other data types such as 3D data volumes (Hurter et al., 2014), trail sets (Hurter et al., 2011), and 2D images (Brosz et al., 2013), has been proven to be very effective. As demonstrated in all these works, animation is superior to using (two) spatially-distinct views linked by classical brushing-and-selection for the task of relating elements (groups of data points) shown in the two views. The topic is further discussed in (Hurter, 2015). Key to this are the facts that users (1) can focus on a single view in the animation case, rather than having to continuously switch looking at two views; and (2) can spot structures of interest, *e.g.*, forming or splitting groups of points, that appear at *any* moment during the animation, but are not visible in the end views. Moreover, using a single view increases the visual scalability of the method, *i.e.*, allows it to show larger datasets in the same screen space.

Differently from RVD, we do not show a *static* interpolation of two projections (in its case, RadViz and a spring-based system) to explain a better-clustered plot in terms of the anchors, as this may misguide the user, as already discussed in Section 8.3.2. Therefore, our combination of a DR projection with a RadViz explanation is to our knowledge novel. However, when the LAMP plot is shown (at the end of the animation), users may be confused by the visual presence of the anchors, and aim to interpret the positioning of points in LAMP in terms of the anchors, which would be incorrect. To alleviate this, we add a gray background behind the anchors in the icicle plot. When the background is visible (gray), it tells we are in LAMP mode, so the anchor positions should not be considered (Figure 8.9 right); when it is invisible (white), it tells we are in RadViz mode, so anchors explain the scatterplot point positions (Figure 8.9 left). During the animation, the background color linearly changes between its two end colors, indicating that we have a transitional state. As alternative we considered to make anchors transparent in LAMP mode. However, this did not allow us to explain point groups by variable values.

Figure 8.10(a) shows the result of LAMP in RadViz++ for the Segmentation dataset. Compared to the RadViz force-based layout (Figure 8.8 and earlier), we now see a much better cluster separation. Animating this view towards the RadViz layout (Figure 8.8) allows us to explain these clusters in terms of the data variables, as discussed so far.

DR projections can also benefit from variable filtering (Section 8.4.4.2). Figure 8.10(b) shows LAMP applied to the variables selected after the filtering done in Figure 8.8. We see the same cluster separation as when using LAMP on all 18 variables (Figure 8.10(a)). We obtain a DR projection having roughly the same clustering quality as the original one, but with about half (11) of the original 18 variables.

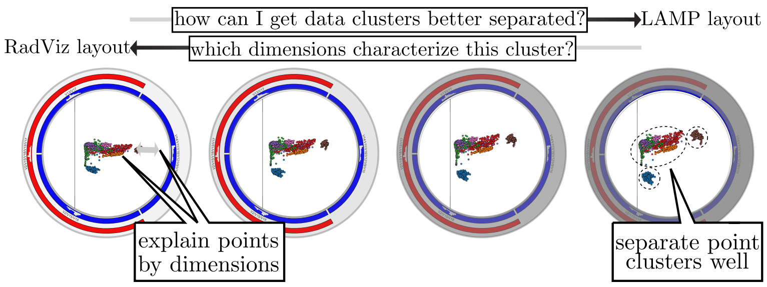
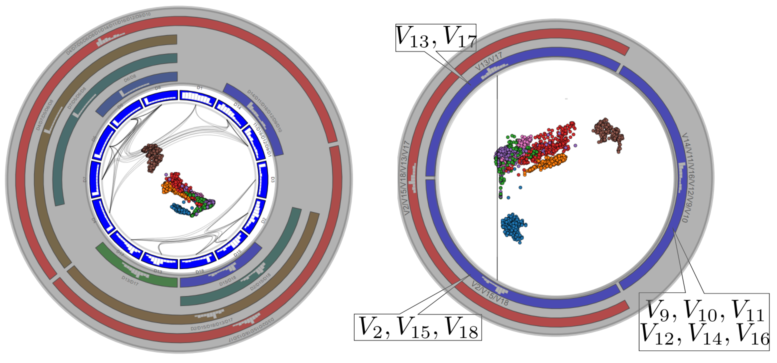


Figure 8.9: Animation of RadViz scatterplot **(left)** towards the LAMP scatterplot **(right)** for the Segmentation dataset. Interpolation factors are $0.2, 0.4, 0.6, 0.8$. While LAMP plots offer better cluster segregation, RadViz plots explain the points better in terms of their variables. Note how the icicle-plot background opacity changes to indicate the RadViz *vs* LAMP mode of the scatterplot. The lack of details (*e.g.*, it is difficult to see labels and histograms) is not prohibitive to understand the figure: the objective is to show how clusters are better represented as the animation goes from RadViz to Lamp layout. Adapted from Pagliosa and Telea (2019).

While force-based point positioning and variable-range filtering (Section 8.4.3), implicitly explain *all* scatterplot points by variables and their ranges, one often wants to explain a *specific* group of points. We support this by a brushing-and-linking tool that links brushed and/or selected points (in the scatterplot) to their histogram bins (in the circular icicle plot) where their values reside. We show the linking by drawing lines between points and bins. To reduce visual clutter, we use again bundling to group these lines. Brushing-and-linking tool is bidirectional, as we can also select bins and show all points having values therein, as shown in Figure 8.11. We selected here two clusters in the LAMP scatterplot of the Segmentation dataset. For each cluster, bundles show how its points can be explained by specific ranges (bins) of the three variable-sets used in the analysis.

8.5 EXPERIMENTS

We next illustrate the working and added-value of RadViz++ with experiments on three different datasets. First, we validate our method using a synthetic dataset, for which the ground truth is known (Section 8.5.1). Next, we compare RadViz++ with other high-dimensional visualization methods and show that we can reach the same conclusions (Section 8.5.2). Finally, we present



(a) (b)

Figure 8.10: **(a)** LAMP scatterplot for the Segmentation dataset. **(b)** LAMP after the variable filtering shown in Figure 8.8, leading to a better clustering, but using only 11 of the 18 variables. Adapted from Pagliosa and Telea (2019).

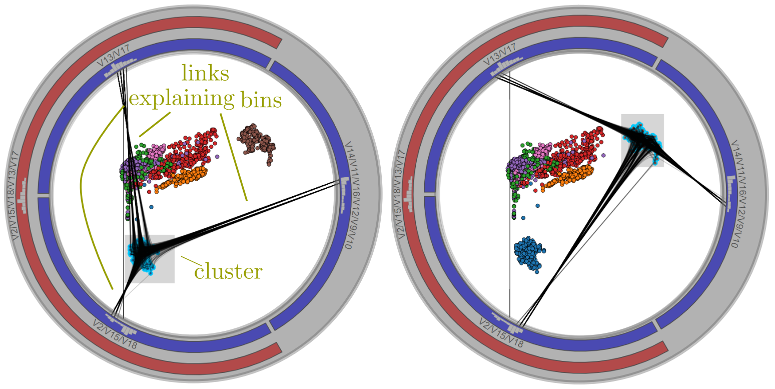
the analysis and obtained insights from a complex dataset (Section 8.5.3).

8.5.1 *Validation On Synthetic Data*

We use the dataset described in (Pagliosa et al., 2016) to validate our method. In their article, the authors proposed several visual metaphors (different from ours) to explain the projected data by their variables. The dataset has $m = 350$ instances, $n = 3$ variables, and $|C| = 2^n - 1$ clusters. Each cluster $c \in C$ contains instances having variation in only a subset of the n variables, while the rest is set to zero. In this sense, clusters $c_1, \cdots, c_7$ contain instances with variation in the variables $\{V_1\}, \{V_2\}, \{V_3\}, \{V_1, V_2\}, \{V_1, V_3\}, \{V_2, V_3\}$, and $\{V_1, V_2, V_3\}$. Data variation in each cluster c follows a different Dormal distribution $\mathcal{N}(\mu_c, \sigma_c^2)$ centered at μ_c and with standard deviation σ_c. The dataset was created with $(\mu_1, \cdots, \mu_7) = (0, 5, 7, 30, 40, 30, 20)$ and $\sigma_1, \cdots, \sigma_7 = 0.5$.

The authors visualized this dataset using LAMP (Figure 8.12(a)). The LAMP projection is binned on a uniform 2D grid based on user settings, where a clustering algorithm takes place. For each found cluster, histograms show the variance of the variables of the contained data points. Briefly put, the method shows clusters in the data and also which variables are (mostly) responsible for their formation.

We next use RadViz++ to find and explain clusters in this dataset (Figure 8.12(b)). For visual inspection, we color scatter-



(a) (b)

Figure 8.11: Brush-and-link explanation of the **(a)** blue and **(b)** brown clusters in LAMP mode. Despite groups of points cannot be correlated to anchors in the LAMP scatterplot, it still valid to explain them in terms of variable ranges. Adapted from Pagliosa and Telea (2019).

plot points by their respective cluster IDs. Here and next, these IDs are not used as variables in RadViz++. In the result, we see that the scatterplot contains 7 distinct point clusters $c_1, \cdots, c_7$. The *positions* of these clusters with respect to the 3 variables directly provide the needed explanations, without requiring more complex interaction, linked-views, comparison of bars heights in different histograms, or data gridding as in (Pagliosa et al., 2016). Equally importantly, the explanations of clusters in terms of variables in our case are the same as those provided by Pagliosa et al. (2016). However, in RadViz++ instances whose variation occurs only in one variable, *i.e.*, those in clusters c_1, c_2, c_3, are mapped to the same 2D locations, due to limitations of the RadViz force-based placement scheme (see Section 8.3.1). To decrease this visual ambiguity, we use the brushing-and-linking tool (Figure 8.13) to select each such point-like cluster in the scatterplot. Since we see edges going from a cluster to *multiple* bins in at least one variable, this explicitly shows that there are *multiple* points mapped to the same scatterplot location. A tooltip could inform details about the selected points for further analysis.

8.5.2 *Wisconsin Breast Cancer*

This dataset is commonly used as benchmark in Visualization and Machine Learning (see the extensive reference list in (Dua and Karra Taniskidou, 2019)). It has $m = 699$ instances (patient tissue

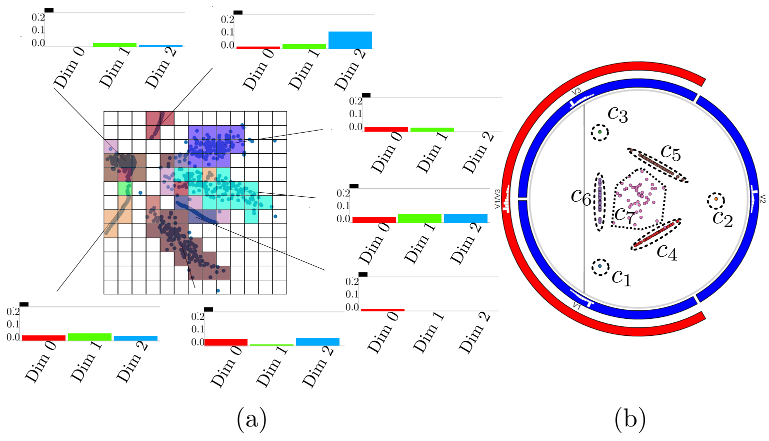


Figure 8.12: **(a)** Attribute-based analysis of 7 Gaussian clusters dataset (Pagliosa et al., 2016). The variable "Dim i" maps to V_{i+1} in our notation. **(b)** RadViz++ leads to the same conclusions with a cleaner and simpler layout. Adapted from Pagliosa and Telea (2019).

samples), $n = 9$ variables (microscopic tissue data), and 2 labels (cancer or lack thereof). The aim is to find which variables or ranges of variables that help to predict the class labels, much as for the Segmentation dataset. We again compare RadViz++ with (Pagliosa et al., 2016) to verify whether we can achieve the same conclusions.

In their article, Pagliosa et al. (2016) concluded that both clusters (for the two existing labels) mainly differ because of the *variance* of specific variables. This is shown by the box plots in Figure 8.14. The bottom (orange) cluster, corresponding to malignant instances, is described by a high variance in almost all variables. The top cluster (benign instances) has a low variance in all variables except *Clump Thickness*. In addition, one can also conclude that *Mitosis* is the least discriminant variable between the two clusters, as it has quite low variance in both.

We next use RadViz++ for this dataset (Figure 8.15(a)). The edge bundles show directly that the *Mitosis* anchor is the only one that has no edge to other anchors, which indicates that that variable has the lowest correlation with all others. As we saw, this is confirmed in Figure 8.14. Conversely, the most opaque edge connects the variables *Uniformity of Cell Size* (*UofCSize*) and *Uniformity of Cell Shape* (*UofCShape*). Also, their high correlation is depicted by their blue-parent node in the icicle plot. Note that the visualization in Figure 8.14 cannot show this insight. Besides these extremes, Figure 8.15(a) shows no other significant clusters or correlation differences. This tells that the remaining variables have

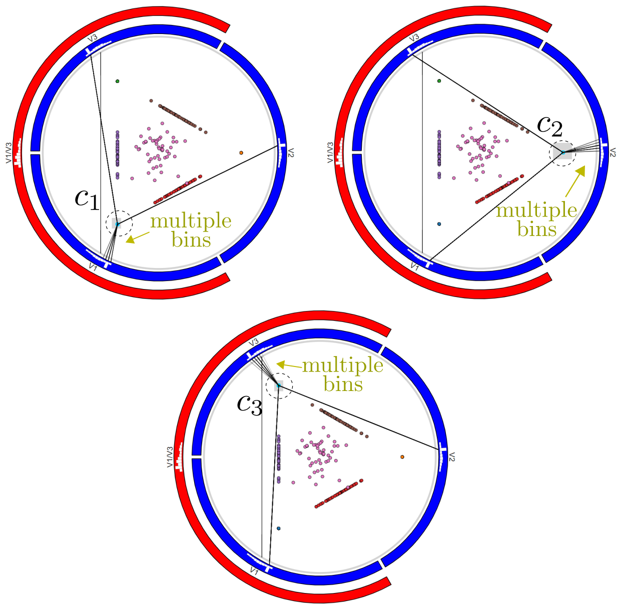


Figure 8.13: The brush-and-link tool helps explain clusters whose points overlap in the scatterplot, thereby decreasing ambiguity problems. For each selected cluster c_i, bundles show that its points have *multiple* values in at least one variable bins. Adapted from Pagliosa and Telea (2019).

similar correlation coefficients. In this case, it is not a good option to analyze this dataset using the force-based scatterplot metaphor as proposed by RadViz, since this will map all instances close to the circle center, as we indeed see in Figure 8.15(a).

To find which variables discriminate between the two clusters, and why, we use the LAMP scatterplot in RadViz++ (Figure 8.15(b)). As expected, this scatterplot separates clusters. We now use brushing-and-linking to explain these in terms of variables. We first select points in the benign (blue) cluster (Figure 8.16(a)), then in the malignant (orange) cluster (Figure 8.16(b)), and compare the two views to find similarities and differences as follows. First, we see that edges from the benign cluster (Figure 8.16(a)) go to *multiple* bins of the same variable in both cases, except for variable *Mitosis*, where edges go mainly to the lowest-value bin. Hence, *Mitosis* has a much lower variance for benign instances than the other variables (confirmed in Figure 8.14). Secondly, we see that bundles for the benign cluster (Figure 8.16(a)) are more concentrated than bundles for the malignant one (Figure 8.16(b)). Hence, variables have a higher variance for the latter than the former instances (again, confirmed by the box plots in Figure 8.14.

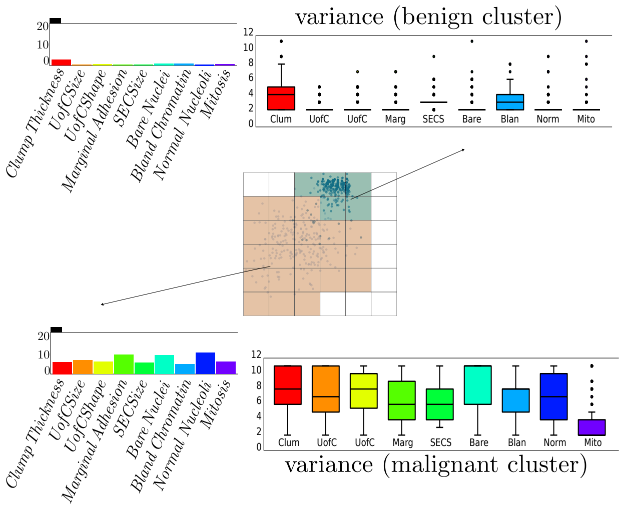


Figure 8.14: Breast Cancer dataset analysis performed by Pagliosa et al. (2016). The variance of the involved variables is the main discriminative factor between the two clusters. All variables contribute quite similarly to discrimination, except *Mitosis*, which has a low overall variance. Adapted from Pagliosa and Telea (2019).

Thirdly, we see that bundles go mainly to the low-side bins of their respective histograms in Figure 8.16(a), while bundles in Figure 8.16(b) go more uniformly to all bins, and sometimes more to high-side bins in their respective histograms. The figure illustrates this for the variable *UofCShape*, but the same is visible for most other variables. Hence, benign instances have overall lower variable values than malignant ones. This finding also matches Figure 8.14.

We conclude that RadViz++ can lead to the same insights as (Pagliosa et al., 2016). However, RadViz++ requires no multiple linked views, data gridding, or other user settings present in the latter, which should make it easier to use. Moreover, RadViz++ allows a *fine grained* linking of variables, and their ranges (bins) to user-specified sets of points in the scatterplot. The technique in (Pagliosa et al., 2016) cannot do this – it only shows aggregated box-plot statistics for entire classes.

8.5.3 *Corel Dataset*

Finally, we test our method using the Corel dataset (Martins et al., 2014), composed of $m = 1000$ images, $n = 150$ SIFT descriptors $(V_1, \cdots, V_{150})$ and 10 class labels. As for the other datasets, vi-

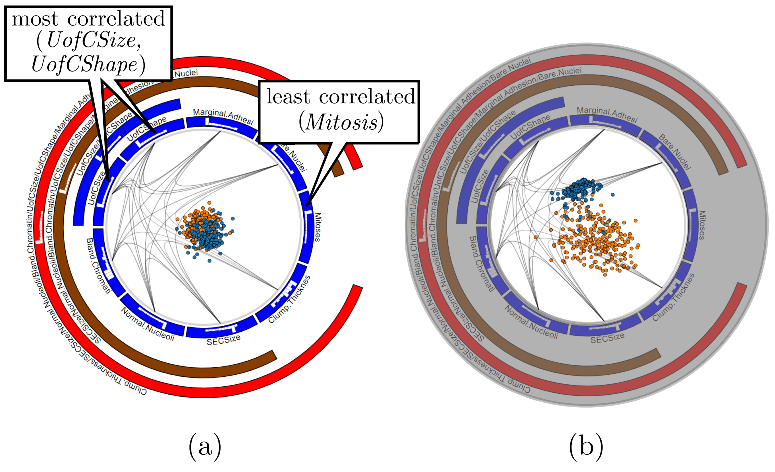
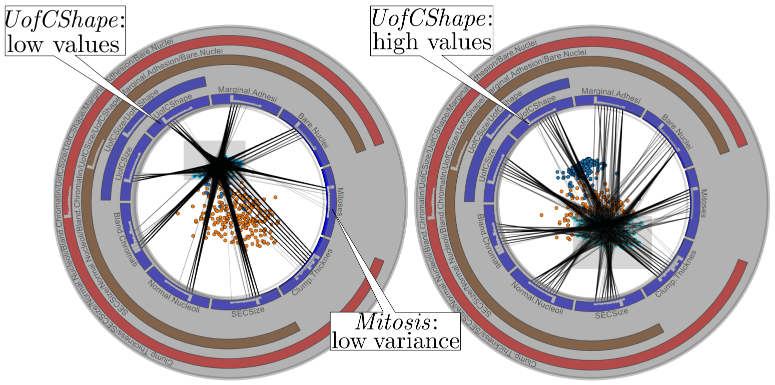


Figure 8.15: Breast Cancer dataset analyzed using **(a)** RadViz++ with force-based and **(b)** LAMP projection. Adapted from Pagliosa and Telea (2019).

sual exploration aims to find correlations of variables (or their properties, such as ranges or variance) with the respective image classes, to further help classifier engineering. This is a much more challenging dataset as the previous ones, not only because of the larger number of classes, but because of its higher dimensionality. In particular, methods such as RadViz, RadViz Deluxe, or the other methods discussed in the related work cannot easily handle 150 variables.

Figure 8.17 shows RadViz++ visualization for this dataset, which lets us draw several insights. First, we see that same-class clusters get formed, although not well separated. However, we also see that the 150 variables get partitioned quite clearly into 9 groups, each indicated by a set of mutually bundled edges. This suggests that we could strongly reduce the dimensionality of the data, by variable aggregation and/or filtering, and thereby possibly achieve a better cluster separation and, thus, explanation.

Following the above observations, we next proceed to aggregate/filter variables. First, we aggregate variable-groups having a medium-range correlation, by selecting their respective nodes, marked green in the icicle plot. Figure 8.18(a) shows the start of the selection process, where four such groups are highlighted by the corresponding green-hue nodes in the icicle plot. After a few extra aggregation operations, we obtain the simplification shown in Figure 8.18(b). The ten groups of variables present in the figure fairly describe the underlying data, as each variable group describes well one of the 10 classes, seen as "pulling" the points of the respective



(a) (b)

Figure 8.16: Breast Cancer dataset, explaining the **(a)** benign and **(b)** malignant clusters by variables in LAMP mode. Adapted from Pagliosa and Telea (2019).

class towards its anchor. To see if we can improve class separation, we add a few more variable-groups to the selected ones (yellow "+" signs in Figure 8.18(c)). However, this addition does not improve the class separation – compare the scatterplot in this image with the earlier one in Figure 8.18(b). Hence, we revert this step, going back to the variable-groups shown in Figure 8.18(b). Finally, we create a new layout using only the selected variables (Figure 8.18(d)). We can see now how each class is strongly "pulled" towards a single anchor, corresponding to the variable-set that describes it best. Of course, the cluster separation is not perfect – there is still a number of points in the center of the scatterplot, which require most of the selected variables to be described. Finding such points is actually useful, as these are difficult classification cases.

We can next use interactive variable selection to verify how each of the 10 variable-groups we ended with (Figure 8.18(d)) indeed explain the data clusters. For this, we deselect all these variable-groups and next select (activate) them one-by-one. Figure 8.19(a)–c show three such selection steps. We can now see quite well how each variable-set is responsible for explaining a separate cluster, as points having the respective cluster color get clearly "pulled" towards the respective selected anchor. Indeed, if the variable-sets we created did not explain data clusters well, then activating them would pull points having mixed colors (of many different classes) towards the respective anchors. Finally, we consider further aggregating (simplifying) the variable-set we obtained so far. For this, we aggregate the two variable-sets which are children of the red parent node in the icicle plot in Figure 8.18(d). Figure 8.19(d) shows the result. Even if the red color of the parent node had not been

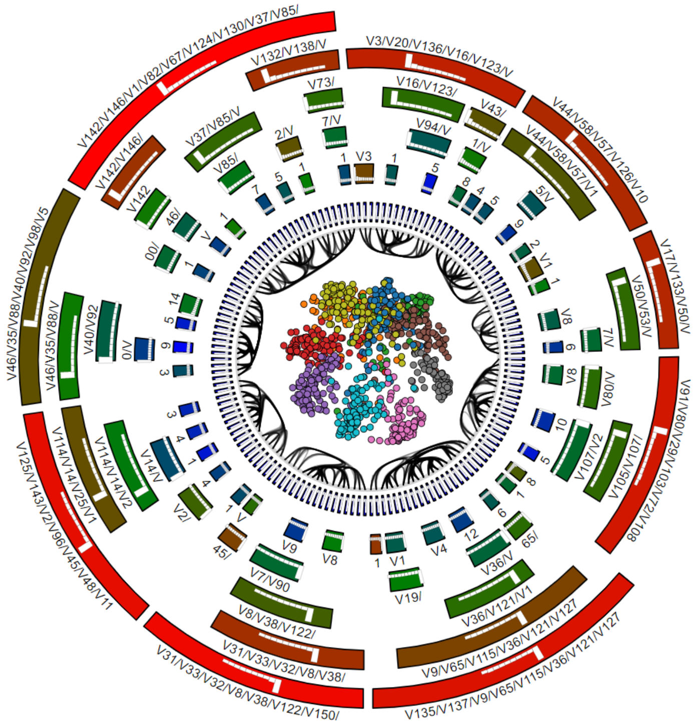


Figure 8.17: Corel dataset visualized using RadViz++. Adapted from Pagliosa and Telea (2019).

a sufficiently strong warning that the respective variable-sets are very dissimilar (uncorrelated), we can see in the scatterplot in Figure 8.19(d) that the top-right green and orange clusters, which were quite well separated before aggregation (Figure 8.18(d)), now get mixed up under the aggregated set of variables. Hence, we revert this aggregation and end the exploration with the 10 variable-sets shown in Figure 8.19(d) as being the best ones for explaining the 10 clusters in the dataset.

8.6 DISCUSSION

We next discuss our proposal vs. the requirements R1–R4:
R1: Our method is as scalable as all other scatterplot-based visualization techniques in the number of *instances*, as every one is mapped to a 2D point. *Variable*-wise, we argue that our method scales far better than all existing RadViz-class techniques due to the hierarchical variable aggregation and variable filtering. Two aspects are related to this point, as follows. First, even when hierarchical variable aggregation is not used, we can display

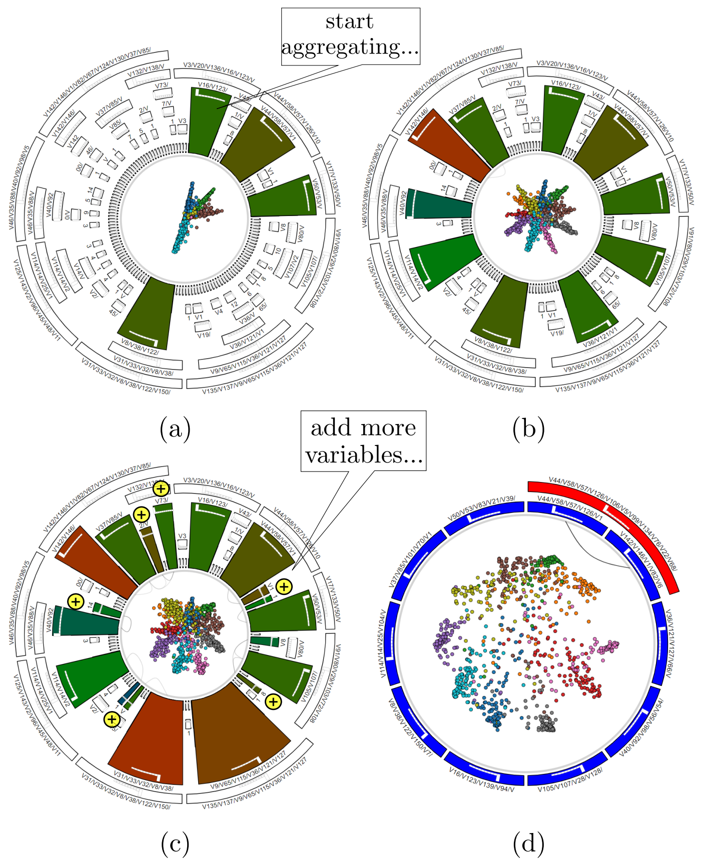


Figure 8.18: Finding the most descriptive variables for 10 clusters in the Corel dataset. Detailed description in the text. Adapted from Pagliosa and Telea (2019).

up to roughly thousand variables along the plot circumference, since each variable requires only a circle sector of a few pixels width to be visible and distinct from its neighbors. This same scalability has been demonstrated earlier by visual designs using the same radial icicle plot, see *e.g.*, (Holten, 2006; Hoogendorp et al., 2009; Reniers et al., 2014) for applications visualizing thousands of elements from software hierarchies. Secondly, as explained in Section 8.4.2.1, we simplify the hierarchy produced by agglomerative clustering based on a user-defined similarity factor δ (preset to 10% of the root-cluster diameter). As explained there, this factor controls the number of levels the simplified hierarchy will show, so users can "flatten" arbitrarily large hierarchies in this way up to the desired level of detail. Also, it is important to note that we do not need to display, nor even compute, the *full* variable

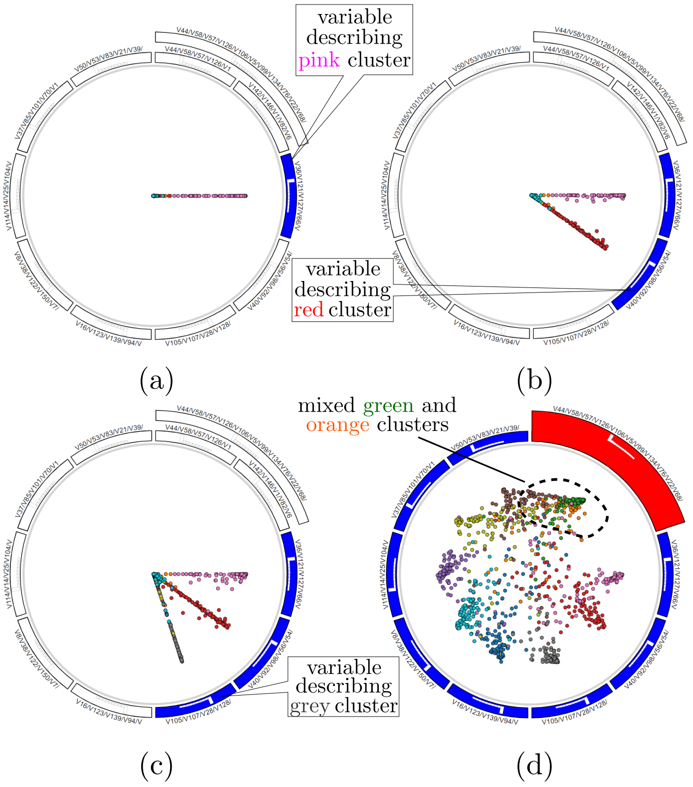


Figure 8.19: Verifying the explanatory power of each variable-set after selecting its respective anchor **(a–c)**. Further aggregating these variables reduces cluster separation **(d)**, so should be avoided. Adapted from Pagliosa and Telea (2019).

hierarchy: if, during the bottom-up clustering process, we decide that we reach a point where the dissimilarity of variable-groups (roots of hierarchy subtrees computed so far) is larger than what the user can tolerate, then we can simply stop clustering and only use the hierarchy levels computed so far. This explicitly limits the maximum number of levels (concentric rings in the icicle plot) that will be present in RadViz++. Finally, users can always *locally* refine the level-of-detail by choosing to aggregate certain groups of variables (hierarchy subtrees) but show other ones in full detail. The same techniques have been successfully used to visualize hierarchies of tens of levels and thousands of leaf nodes, as mentioned earlier (Hoogendorp et al., 2009; Reniers et al., 2014). In the same time, the hierarchy allows a flexible variable-placement along the RadViz circle where similar variables are placed close to each other.

R2: We decrease ambiguities of data-to-variable analyses by histogram bins and brushing-and-linking that show which variables (and their ranges) correspond to a user-specified given subset (cluster) of scatterplot points. Separately, we decrease such ambiguities by variable filtering and aggregation, which allocates more visual space to explain fewer variables – thus, more space per variable. We also bundle bin-to-cluster links to further decrease visual clutter and associate data points to variable ranges (bins) easier.

R3: Besides the aforementioned hierarchy-based anchor placement and dendrogram of clustered variables, we use hierarchical edge bundles (HEB) to explicitly show groups of similar variables. HEB is spatially compact, intuitive, and also explains anchor-placement ambiguities which are inherent to the RadViz circular layout.

R4: To better separate point clusters, we allow exploring data by two different dimensionality-reduction methods. At one extreme, the RadViz projection explains well instances in terms of variables, but may not separate point clusters well. At the other extreme, the LAMP projection achieves the opposite. Users can fuse insights provided by the two projections by *e.g.*, selecting clusters of interest (in LAMP) and animate them back-and-forth to the RadViz projection, which explains them in terms of variables (or conversely).

Limitations: While scalable, simple to implement, and working generically for any quantitative high-dimensional dataset, our proposal also has several limitations. First, even when doing variable aggregation and filtering, a certain amount of visual overlap of different-value instances will occur into the scatterplot, due to inherent limitations of the RadViz placement (Equation 8.3). While other placement methods may improve upon this, *e.g.*, RadViz Deluxe (Cheng et al., 2017), we chose to do this via a radically different way, namely using a different DR method (LAMP) and animation to link it with the anchor placement. Whether our approach is better than RadViz Deluxe in terms of ease of use and accuracy of the obtained insights is an open problem requiring further evaluations. Secondly, our approach cannot yet handle categorical data; also, handling negative data values is subject to limitations present, to our knowledge, in all other RadViz-class methods. Extending our hierarchical anchor placement based *e.g.*, on similarity metrics defined on categorical data (Broeksema et al., 2013) is an interesting possibility yet to be explored.

8.7 VISUALIZING EMBEDDINGS

Alternatively, we later tried to tackle RQ4 with a different visualization approach. By simultaneously correlating several phase spaces as function of embedding parameters and entropy (also investigating R1), the goal was to find out how similar to each other different embeddings are, and what makes them different.

First, we predefine a range of possible values for $m \in [m_{\text{min}}, m_{\text{max}}]$ and $\tau \in [\tau_{\text{min}}, \tau_{\text{max}}]$ for the embedding parameters. Next, we create embeddings with all combinations of (m, τ). Further on, we aim to create a plot, like the one shown in the inner ring of RadViz++, where we can visually *compare* embeddings. For this, we apply dimensionality reduction, considering every embedding as a data point. In contrast to the usage of LAMP for dimensionality reduction, presented in the earlier sections, we now explore the usage of another dimensionality reduction algorithm, namely t-SNE (Section 8.4.5). This is motivated by t-SNE better capability to separate clusters of similar obsevations. The main drawback of t-SNE, namely its low speed, is less relevant in this scenario (as opposed to the scenarios discussed in the earlier sections), since we now aim to create a *single* plot rather than interactively explore a sequence of plots.

In order to work, t-SNE must receive as input a similarity matrix containing the similarities among all computed phase states, Using the Cross Recurrence Plot (CRP) matrix $\boldsymbol{R}$ (Equation 6.7), defined as

$$R_{a,b} = \Theta(\varepsilon_a^i - \|\boldsymbol{\phi}_i(a) - \boldsymbol{\phi}_j(b)\|_2)\ \Theta(\varepsilon_b^j - \|\boldsymbol{\phi}_j(b) - \boldsymbol{\phi}_i(a)\|_2), \quad (8.5)$$

is not suitable for this task, as it compares states with the same number of components (dimensions). In other words, CRP accepts different number of states N_i, N_j, but the embedding dimension m for both spaces must be the same.

The Joint Recurrence Plot (JRP), defined by a matrix having as entries

$$R_{a,b} = \Theta(\varepsilon_a^i - \|\boldsymbol{\phi}_i(a) - \boldsymbol{\phi}_i(b)\|_2)\ \Theta(\varepsilon_b^j - \|\boldsymbol{\phi}_j(b) - \boldsymbol{\phi}_j(a)\|_2), \quad (8.6)$$

tries to overcome this limitation. In our case, JRP has an inverse behavior when compared to the CRP: The number of dimensions m_i, m_j may be different, but the number of states $N_i = N_j$ must be the same. This is relatively easier to accomplish by *e.g.*, reducing the number of states in both phase spaces to $\min(N_i, N_j)$, assuming that this amount is sufficient to unfold the dynamics of both systems. Still, this approach is not useful to distinguish the same time series embedded with different pairs (m, τ). This happens because the relation among their orbits remains roughly the same. For instance, three different embeddings of the Lorenz system will have almost identical JRP matrices. As consequence, the similarities among them (*e.g.*, taking the Maximum Diagonal Line (Section 6.4)) will not tell much about their phase-spaces differences, which is, as explained, what we want to visualize

As a third alternative, we propose the Average Neighboring Preservation (ANP), explained next. Let $N(\boldsymbol{\phi}_i(a), k)$ and $N(\boldsymbol{\phi}_j(b), k)$ be the set of indices of the k-nearest neighbors of $\boldsymbol{\phi}_i(a)$ and $\boldsymbol{\phi}_j(b)$, respectively. We take the intersection and union of both sets to compute the Jaccard index between the two states as

$$J_{a,b} = \frac{|N(\boldsymbol{\phi}_i(a), k) \cap N(\boldsymbol{\phi}_j(b), k)|}{|N(\boldsymbol{\phi}_i(a), k) \cup N(\boldsymbol{\phi}_j(b), k)|}, \tag{8.7}$$

where $|\cdot|$ is the set cardinality. This yields a similarity matrix $\boldsymbol{J}$ based on the neighboring preservation of both embeddings. Next, we use the average of $\boldsymbol{J}$ to represent the similarity of the two embeddings. Note that this approach is invariant to rotations and transactions of the phase spaces, which is not the case for CRP and JRP. Thus, only the surroundings of states (within their own attractor) are important, and not their distances with states from another system.

The t-SNE projection of the set of phase spaces yields a scatterplot in which a point represents a phase space given by a particular (m, τ) combination. To further see how m and τ impact the embeddings and their properties, we depict each point in the scatterplot by a so-called *tri-ring*. This is a glyph consisting of two concentric circles. We use the inner circle to depict Von Neumann's entropy E_{vn} based on the first dimension (as proposed by the SE method), coded by luminance. The ring area between the inner and outer circle is divided vertically into two half-rings. We next color-code the values of m, respectively τ, onto the left, respectively right, half-rings, using a grey-red-blue colormap.

Figure 8.20 shows an example of our visualization, called Projection of Embedding (PoE), for the Logistic map using a range of $m, \tau \in [2, 15]$). Several types of information can be derived from this plot. For instance, *clusters* of phase-space embeddings are formed based on the time delay – indeed, points within such a cluster share similar colors for their right ring-halves, so, they have very similar τ values. Secondly, within a cluster, the m value explains the spread of points from the projection center towards its periphery – this is visible in the luminance gradient of the left ring-halves that is bright on points close to the projection center (low m values) to dark on points far away from the projection center (high m values). Thirdly, we see how clusters "curl" close to their far-from-projection-center ends. This may indicate that the respective phase spaces have reached a plateau of dissimilarity, and now turn back to become similar to lower dimensions. Such behavior can suggest the attractor was first completely unfold, then lost its structure, and now it is starting to be unfolded again, which could be used to estimate the time delay window t_w.

Separately from the above, if we look at the color of the inner rings in Figure 8.20, we see how these vary from roughly bright in the projection center (low entropy values) to dark at the projection periphery (high entropy values). This shows that the entropy grows proportionally with the embedding dimension. A similar tree-like structure was found for the Hénon system. Note also the optimal embedding indicated by an arrow in Figure 8.20. Interestingly, this is not located to the projection center, but to one of its extremes. The fact that we see other points having similar low entropy (bright inner circle) values *far away* from this optimal embedding in the projection tells us something very important, namely that phase spaces which are *quite different* (points far in the projection) have very similar *low-entropy* values. This is an additional argument in favor of the study in Section 5.7 that shows that low entropy is not one-to-one correlated with an optimal embedding.

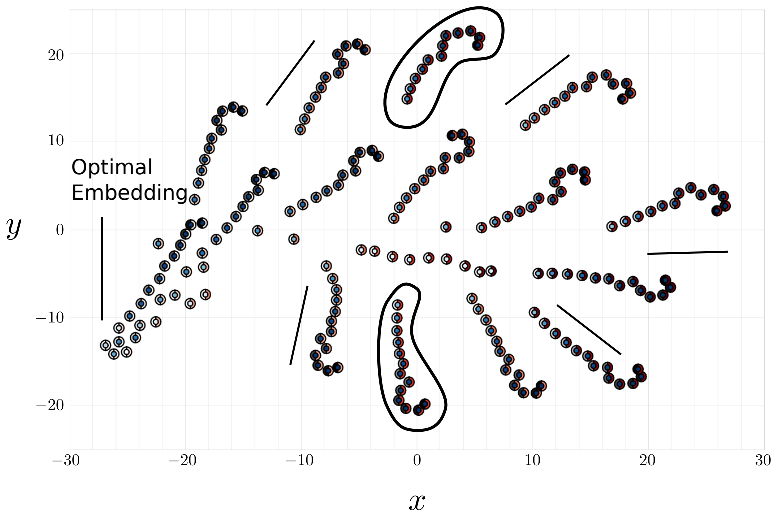
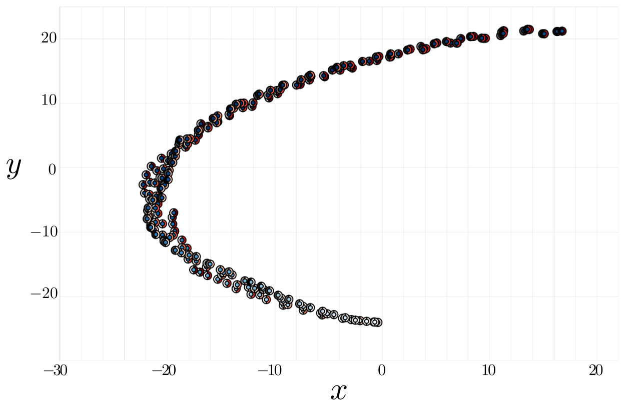
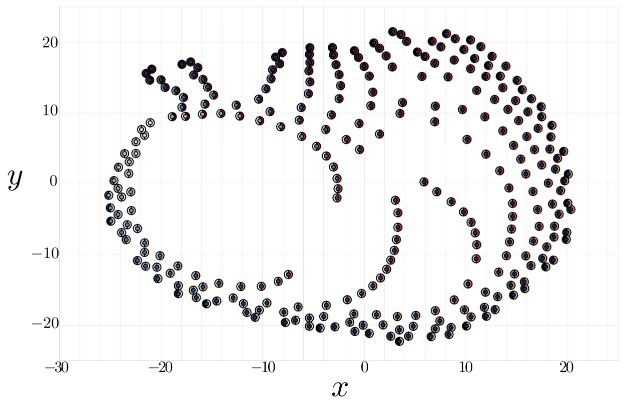


Figure 8.20: PoE for the Logistic map. Clusters (two of which are marked by closed curves) are formed mainly based on similar τ values. Points spread away from the center as the embedding dimension m increases. The optimal phase-space embedding $(m = 2, \tau = 1)$ is marked at the left.

However, quite different patterns were found for other time series. For instance, Figure 8.21(a) shows the PoE plot for the Lorenz system, which resembles a crescent-like structure. From this plot, not much information could be extracted. In particular, we do not see the structuring of phase spaces in clusters that are explained by the values of m and τ. A similar unstructured plot was obtained for the Rössler dataset. However, by increasing the range of acceptable time delays (we remember that the the SE estimated a time delay window $t_w = 29$ for the Rössler system), we achieved another PoE result, as shown in Figure 8.21(b).



(a)



(b)

Figure 8.21: PoE for the **(a)** Lorenz and **(b)** Rössler systems. In contrast to PoE results for the Logistic map, far less structure in the set of phase spaces is visible.

The less structured results for PoE plots in Figure 8.21 can have several, not mutually exclusive, explanations. First, the difference between trajectories in the phase spaces obtained for the respective Lorenz and Rössler systems may, indeed, be far smaller than between trajectories in the phase spaces of the Logistic system. This denotes, albeit implicitly, the difficulty of finding general characteristics that define a good embedding for any system. Secondly, it is known that t-SNE results are quite sensitive to the setting of its perplexity parameter (Wattenberg et al., 2016). This means that it *could* be possible to obtain more insightful plots for the Lorenz and Rössler systems by using other perplexity values.

Concluding this section, we found out that it is possible, indeed, to visualize how phase spaces change with the embedding parameters; low entropy values do not necessarily characterize embeddings which are very similar to the optimal embedding; and significant structure in the set of phase space does exist, but it cannot be easily and reliably captured for all systems.

8.8 FINAL CONSIDERATIONS

We have presented RadViz++, a set of techniques for interactive exploration of high-dimensional data using a RadViz-type metaphor. We designed our techniques to alleviate several types of problems present in existing RadViz-class methods, as follows. We increase variable scalability by using a variable-clustering technique and simplified variable-hierarchy visualization, which allows us to easily handle over a hundred variables. We reduce ambiguities of the RadViz circular layout, and also summarize variable similarities by using a hierarchical edge-bundling approach. We explain data clusters in terms of variables and variable-ranges by linking the former with histogram bins representing the latter. Finally, we reduce visual clutter to better analyze data clusters by integrating a separate dimensionality-reduction method, good at cluster segregation, and linking its explanation with the RadViz metaphor via animation. We show that our approach can lead to the same insights on two different datasets as when using existing visualization methods, but with less effort, and demonstrate scalability on a third dataset.

Returning back to our research question RQ4, we recognize that we have **answered it partially**, as we tackled a wider (but related) question: *how to correlate observations with their attribute values?* The key reason for taking this more general (time-series-independent) approach was to design a visualization solution that is comparable with existing results in the Visual Analytics literature. As we found no specific visualization techniques (at least, not in the context of Dynamical Systems and/or time series) in the radial class, the only way to validate our proposal was to compare it with other techniques and on generic datasets.

On the positive side, the evaluation of RadViz++ was done on large, complex, high-dimensional real-world datasets consisting of hundreds of dimensions and thousands of observations. Within this context, RadViz++ has shown to fulfill the visual-analysis requirements identified for radial-class visualizations better than other visualization techniques in the same class. After such validation, we conclude that RadViz++ could be used to tackle R4 directly, further exploring the relations between time-series and phase-space features.

Additionally, we investigate how to explore different phase spaces simultaneously using t-SNE. The idea was to visualize interesting patterns related to how phase spaces are similar to each other as function of the embedding parameters, and how entropy correlates with different phase spaces. However, a clear pattern was not found in this analysis. From the current results, it is not possible to say if the observed correlations and structures in the set of phase spaces would hold for any (or at least, a large class of) Dynamical Systems. We acknowledge that this topic is under-explored, and future research is needed to consolidate and refine insights in this direction.

9 ESTIMATING EMBEDDING PARAMETERS USING NEURAL NETWORKS

9.1 INITIAL CONSIDERATIONS

As outlined at numerous points in this Ph.D. thesis, the estimation of an optimal embedding, captured by the embedding dimension m and time delay τ parameters, is of paramount importance for all subsequent applications that use a phase-space representation to deal with dynamical systems, *e.g.*, classification, regression, or visual exploration. Estimating m and τ is challenging. Chapter 4 discusses many methods to this end, none of which is ideal. Chapter 5 explores this topic even further, showing the correlation between optimal embeddings and low entropy, but does not put forward a definitive solution to the estimation problem.

In this chapter, we examine a different set of mechanisms, based on deep learning, for addressing the optimal embedding estimation problem. That is, we explore the research question:

RQ5. "Can neural networks estimate Takens' embedding parameters?"

Before we proceed with detailing our deep-learning approach to embedding estimation, we should first argue *why* deep learning should be considered in this context. First of all, let us state the arguments against it: despite the effectiveness of neural networks in time-series forecasting (Chakraborty et al., 1992; Han and Wang, 2013; Firmino et al., 2014), such approaches should only be considered as a last resource, when *there is no deterministic way* to tackle a given problem. The main reason for this is that there is no detailed knowledge of what is precisely happening inside a neural network while it is learning. Although some researches have tried to derive and show information about the learning process using visualization methods (Zeiler and Fergus, 2014; Rauber et al., 2017), it is still very hard to know what has led the neural networks to converge to an acceptable risk and, hence, whether the network has learned properly the task at hand.

Nonetheless, this context represents exactly our case: we do not know which is the best set of attributes to define the phase space – apart from Takens' theorem which outlines maximal bounds for the embedding parameters, but not which are *optimal* values for these. Moreover, such optimal parameters vary in ways we do not know how to model across different phenomena. However, deep learning was designed precisely with the aim of capturing patterns and similarities which do exist in the data but which are hard to describe by a set of explicit rules. Hence, we argue that using deep learning to capture optimal embedding parameters is a valid investigation proposal.

In this chapter, we propose a deep learning approach to estimating the embedding parameters with the following contributions with respect to existing state-of-the-art approaches:

R1. few user-defined parameters and settings involved in the embedding estimation process;

R2. low sensitivity and complexity in performing searches on the space of parameters;

R3. robust validation against ground-truth datasets.

The structure of this chapter is as follows. Section 9.2 discusses the terms and concepts behind deep learning relevant to our context, as well as related work from the perspective of requirements R1–R3 on the usage of neural networks for phase-space reconstruction. Section 9.3 introduces our deep-learning method. Section 9.4 demonstrates the proposed method for the estimation of embedding parameters for several dynamical systems for which ground-truth embedding is known. Section 9.5 concludes this chapter.

9.2 REVIEW OF THE RELATED WORK

Despite the importance of Takens' embedding theorem (Section 2.4), the respective work does not provide any additional information on how to estimate the embedding parameters, only that a *sufficient* dimension m should be at least twice larger than d to properly unfold the phase space (although this is usually an overestimation). Several methods were proposed to estimate m and τ under the assumption they are independent or bounded to the time-delay window $t_w = (m-1)\tau$. For a broader related work overview, refer to Chapter 4.

Early methods (Albano et al., 1987, 1988; Abarbanel et al., 1993) used ACFs (Section 4.2.1.1) to estimate τ, which have limited modeling abilities given that only linear functions are used. Fraser and Swinney (1986) tried to overcome these issues by using the first local minimum of the nonlinear AMI function over different time delays (Section 4.2.1.2). This simple approach respects R1 and R2 as it scarcely contains any parameters. Nonetheless, Martinerie et al. (1992) empirically observed that neither ACF nor AMI were consistent to estimate the time-delay window t_w (and, as consequence, a bound for τ), violating R3.

Kennel et al. (1992) proposed the FNN method (Section 4.2.2.1) to estimate the optimal embedding dimension m. By using the time delay τ estimated using AMI, FNN reconstructs a time series using different dimensions while computing the index set of the k-nearest neighbors (Mucherino et al., 2009) for each phase state. The best value for m is defined as the one for which the fraction of nearest neighbors remains constant as the dimension increases. In spite of being simple and requiring an acceptable number of parameters (thus, satisfying R1), this method is very sensitive to the choice of τ and noise, counterposing R2 and R3.

Rosenstein et al. (1994) employed the AD measure (Section 4.2.1.5) to gauge the inverse relation between the redundancy error and the attractor expansion as a function of the time delay. They observed that AD increases until it reaches a plateau, indicating the attractor is sufficiently expanded. However, non-negligible errors are typically introduced while analyzing general systems (Ma and Han, 2006), which goes in disagreement with R3; similarly to FNN, this method involves Monte Carlo simulations (Rubinstein and Kroese, 2007) while scanning the space of parameters, failing R2.

The expansion of an attractor can be also described in terms of the spreading rate of its phase states, *i.e.*, as function of its singular values. In this context, Kember and Fowler (1993) proposed SVF (Section 4.2.1.4) to estimate the time delay when the attractor is maximally spread out, which ideally should happen when all eigenvalues are equal. As this is unlikely to occur for real-world scenarios, the time delay τ yielding the minimum SVF was defined as the most adequate to represent the phase space. In summary, this method is simple and demands no parameters to compute, which satisfies R1 and R2. However, despite SVF shows consistent results for different dimensions, as recently reinforced by a modified version (Chen et al., 2016), it may not properly work for attractors with genus (number of voids in the manifold) greater than 1, thus it does not fully meet R3.

Gautama et al. (2003) realized that a deterministic attractor should have a well-formed structure and, therefore, low entropy. Thus, they proposed ER (Section 4.3.4), a method based on minimizing the ratio between the entropy from the phase spaces of the original series and a set of surrogates, providing a function similar to the Minimum Description Length (Rissanen, 1978). In this scenario, R2 is not held as the method needs to reconstruct the phase space for all parameter combinations in order to assess the minimum ER. In addition, no consistency was achieved for such approach either (failing R3). As also discussed in Chapter 5, entropy is not a unique descriptor and might not be the best feature to characterize phase spaces.

In spite of several studies involving the prediction of time series through the usage of neural network models (Chakraborty et al., 1992; Karunasinghe and Liong, 2006; Bhardwaj et al., 2010; Han and Wang, 2013), to the extent of our knowledge, only two of these approaches attempted to estimate m and τ, as follows.

The first approach (Karunasinghe and Liong, 2006) (Section 4.3.6) selected the set of embedding parameters over a densely-sampled range of values based on forecasting accuracies, which violates R2. In addition, their results were overestimated for ground-truth datasets, failing R3.

The second approach (Manabe and Chakraborty, 2007) proposed a more consistent strategy for estimating m and τ without the need of exhaustive comparisons. They start using FNN and AMI to set the maximum embedding bounds (MEB), *i.e.*, the greatest values for m and τ respectively, referred from now on as $m_{\max}$ and $\tau_{\max}$. The phase state is then reconstructed in the Provisional Embedding Vector (PEV) form, as follows

$$\boldsymbol{\phi}(t) = [x(t), x(t+1), x(t+2), \cdots x(t+(m-1)\tau)], \tag{9.1}$$

so that $|\text{PEV}| = (m-1)\tau + 1$ components are taken into account. Note that this approach is different from the Standard Embedding Vector (SEV) presented in Equation 2.15, rewritten below for clarity

$$\boldsymbol{\phi}(t) = [x(t), x(t+\tau), x(t+2\tau), \cdots, x(t+(m-1)\tau)]. \tag{9.2}$$

It is worth to mention that the subindex i on each phase state (and, as consequence, on the phase space $\boldsymbol{\Phi}_i$), typically employed so far in this thesis to refer to the time series T_i, was omitted in the two equations above. This is because more variables are required to explain the neural network architecture, so that variable i will have a different meaning in the remainder of this chapter. This should not cause any confusion, as this chapter assumes that observations (samples) are derived from a single phenomenon with no concept drift or influences of surrogate data.

Next, the PEV vector is used as input layer to be propagated to a hidden layer (the number of neurons were not detailed by the authors), and next to a single output neuron to forecast ρ steps ahead, in the form

$$f_{\text{NN}}(\boldsymbol{\phi}(t)) = x(t+(m-1)\tau+\rho), \tag{9.3}$$

where $f_{\text{NN}} : R^{PEV} \to R^1$ is the neural-network predictive mapping. After the network converges, embedding parameters were estimated directly from the network architecture. Moreover, the authors performed learning with forgetting, hidden unit clarification, selective learning and pruning heuristics during training in attempt to provide a final skeletal network, so m and τ were computed based on the most relevant (largest absolute magnitudes) weights connecting input-to-hidden layers. Nonetheless, this approach requires various thresholds and parameter settings that increase the modeling complexity and chances of overfitting (violating R1 and R2). Finally, no test was performed on more complex datasets such as the Lorenz and Rössler systems, so that R3 was not fully covered.

From our point of view, the main contribution of Manabe and Chakraborty (2007) (referred to from now on as MC) was to infer m and τ without explicitly defining any phase-space features. Hence, phase-space inconsistencies such as expansion rate, noise, genus, and redundancy are disregarded in their approach. On the other hand, the method is complex, as it requires expensive Monte Carlo simulations for determining parameter values and, finally, it yields to different results for multiple runs on the same input due to the random weight initialization.

9.3 PROPOSED METHOD

In this section, we introduce our estimation method and compare its differences and improvements against MC, which is the most similar study found in the literature. Firstly, we describe our network architecture and its settings (Section 9.3.1) to next discuss how m and τ can be confidently inferred from this proposal (Section 9.3.2).

9.3.1 *Network Architecture And Settings*

Our model is based on a fully-connected three-layer neural network trained using the backpropagation algorithm (Figure 9.1). The triple (N, L, M)[1] represents the number of input, hidden, and output neurons, respectively. Similarly to MC, our architecture is based on PEV to forecast a single observation so that M is always set equals to one. In contrast to MC, however, we restrict our input layer to $N = |\text{PEV}| - 1$ neurons and force the last PEV observation to define the class label to be used by the output layer, such that always $\rho = 1$ (Equation 9.3) in our architecture. Despite a small detail, such restriction is important to avoid overfitting with respect to the butterfly effect (Brock et al., 1992). In those cases, a recursive forecasting should be used as discussed in Chapter 5.

In addition, we explicitly set $L = \log(N) + 1$ to probabilistically ensure the algorithm search space (a.k.a. bias) is in parsimony with

1 The variable N, commonly used so far to indicate the number of states in the phase space, as a different meaning in this chapter.

the Bias-Variance Dilemma (Section 5.2). In other words, by logarithmically increasing L, we simultaneously avoid underfitting (the search space gets bigger and more functions can be used to fit data) and overfitting (it grows in a moderate pace based on the number of input neurons, which holds the model complexity).

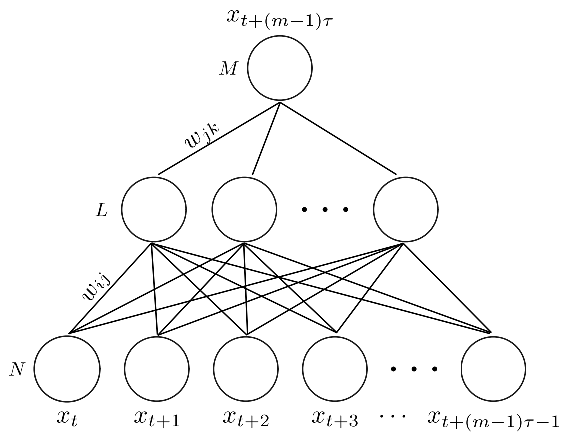


Figure 9.1: Architecture of our three-layer neural network. Terms w_{ij} and w_{jk} represent input-to-hidden and hidden-to-output weights, respectively.

Moreover, our architecture includes learning with forgetting by using the following cost function

$$C = \min_{\boldsymbol{\phi}(t)\in\boldsymbol{\Phi}} \left(\sum_t E(\boldsymbol{\phi}(t)) + \lambda \sum_{e_{ij}\in K} |w_{ij}| \right), \tag{9.4}$$

in which $E(\boldsymbol{\phi}(t))$ is the error function given the input state $\boldsymbol{\phi}(t)$, w_{ij} is the weight for edge e_{ij}, and K is the set of all $(N \times L)$ input-to-hidden network edges. Parameter λ sets the trade-off between weight minimization and accuracy performance. In such circumstance, the MC method chooses λ based on the Relative Normalized Score (RNS) and Monte Carlo simulations. However, our experiments suggest this step is not necessary, since a strong forgetting threshold $\lambda = 10^{-3}$ is enough to deliver relevant results (Section 8.5).

Moreover, our model simplifies MC as it does not depend on hidden unit clarification, selective forgetting or pruning heuristics. By removing such elements, the training stage became faster and more robust as it required smaller search spaces while being less prone to overfitting. Training was performed until cost C (Equation 9.4) reached a predefined threshold $C_{\max}$ or a maximum number of epochs g (in our experiments, those parameters were set as 0.001 and 500, respectively).

Lastly, we normalized our data in range $[0, 1]$, such that our network weights were randomly initialized using 10% of this range. In other words, rather than taking the typical weight range $[-1, 1]$, as we suppose MC did as there is no additional information on this matter, we considered just the interval $[-0.1, 0.1]$ to bring solutions closer to the quasi-convex region of the squared-error surface as analyzed in (de Mello and Moacir, 2018) (this makes even more sense given our data normalization). In practice, this initialization strategy was confirmed to provide better accuracy results than the most typical range of $[-1, 1]$. Table 9 lists our settings and compare them against MC, including the momentum rate α and the step size η, both employed by the gradient descent method.

Table 9: Network settings (n/a refers to missing information).

Method / Parameter	MC	Ours
Number of input neurons N	$\lvert\text{PEV}\rvert$	$\lvert\text{PEV}\rvert - 1$
Number of hidden neurons L	n/a	$\log N + 1$
Number of output neurons M	1	1
Step size η	0.1	0.1
Momentum rate α	0.2	0.2
Forgetting parameter λ	set by RNS	0.001
Number of epochs g	50000	500
Maximal error tolerance $C_{\max}$	n/a	0.001
Interval of random weights	n/a	[-0.1, 0.1]

Our only free parameter is then the size N of the input layer that, in contrast to MC, it was defined in advance using FNN and AMI estimations. In that sense, the MC approach will work well only when FNN and AMI overestimate the embedding parameters. However, in case those methods underestimate m and τ, the Provisional Embedding Vector (PEV) may be too short, resulting in a poor architecture and in not enough information to learn about the underlying phenomenon. As an alternative, we propose to use smaller-to-medium values for MEB to analyze both how the dataset and the network behave for different embeddings (see Section 9.4.3).

9.3.2 *Visual Inspection Of Embedding Parameters*

From a local perspective, each of the N input neurons of our neural-network architecture corresponds to an observation of the PEV.

From a global point of view, however, each input neuron can be seen as a *dimension* of a representative basis. As our neural network is fully connected, we measured the relevance of each dimension $i \in [0, N]$ in terms of the sum $I_i = \sum_{j=1}^{L} w_{ij}$, in which w_{ij} is the weight associated with the connections between the i^{th} input neuron to the j^{th} hidden neuron. Such relevance can be depicted by a bar chart, in which the length of the i^{th} bar maps the magnitude I_i of a given input neuron i (Figure 9.2).

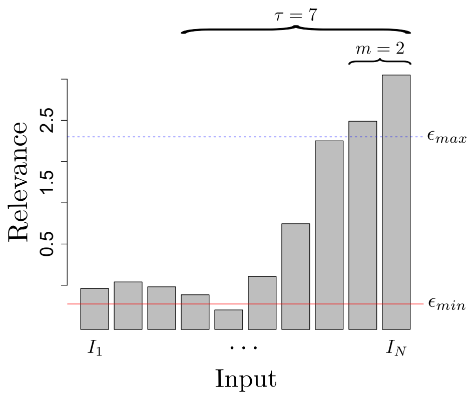


Figure 9.2: Bar chart representing the relevance of input dimensions. Each bar corresponds to the sum I_i of connection weights w_{ij}, given the input neuron i, to every hidden neuron identified with j. The dashed-blue and solid red lines illustrate the thresholds ϵ_{max} and ϵ_{min} respectively, both used to determine the embedding parameters, which were set as $(m, \tau) = (2, 7)$ in this example. For simplicity, indexes $I_1, \cdots, I_N$ are not shown in future plots.

We next use this bar chart to select the embedding parameters m and τ inferred from our network model, as follows. Firstly, we consider all dimensions (at least two) whose relevance exceeds a quantile measure of $\epsilon_{\text{max}} = 80\%$ over all $I_{1 \leq i \leq N}$, as relevant enough to represent parameter m. Secondly, distance $|j - k|$ results in the time delay τ which corresponds to the lag between the most and the least relevant terms I_j and I_k, respectively. Notice I_k is not simply associated with the smallest value, but with the least relevant dimension that lies above a minimum threshold of $\epsilon_{\text{min}} = 10\%$ of I_j. If no delay is found, we set $\tau = 1$ as, in practice, it is the smallest possible value for the time delay. It is worth to mention that, despite ϵ_{max} and ϵ_{min} are free parameters that should be modified based on the search space, we suggest to set them as 80% and 10% respectively, based on experimental analysis.

In summary, our method contributes to the related work in the sense that it does not require any specific definition on phase-space features (such as rate of expansion, number of false neighbors, entropy, etc.) to estimate the optimal m and τ. As we propose, the neural network learns those properties while training on forecasting errors, and we select the embedding parameters based on the final architecture. On the other hand, despite our proposal shares similarities with MC, we simplified the training process, improved the network architecture and settings, proposed a different approach to estimate m and τ from such architecture, and performed more complex experiments regarding variations on the search space.

9.4 EXPERIMENTS

We performed experiments to assess our method in light of the requirements R1–R3 (for more details, see Section 9.1). Next, we introduce the datasets used in the evaluation process (Section 9.4.1), while we discuss the obtained results and aspects of our proposal from Section 9.4.2 to Section 9.4.5.

9.4.1 *Datasets*

In attempt to validate our method, we considered four benchmark datasets, namely Logistic map (Section 3.2.2), Hénon map (Section 3.2.3), Lorenz system (Section 3.3.1), and Rösler system (Section 3.3.2). The systems were generated using a sampling rate of 0.01 (we assumed a sampling rate that preserves the dynamics of the trajectories). Those datasets were chosen because their respective generating rules $R(\cdot)$ are known, and their expected attractors can be fairly compared from the perspective of our approach. Additionally, we consider the Sunspot dataset (Section 3.2.5) to support an empirical analysis based on real-world data. Although there is no ground truth for this last dataset, there is strong evidence that its attractor follows an ellipsoid structure as discussed by Pagliosa and de Mello (2017). Finally, a discussion about how our method behaves while analyzing stochastic data, following a Normal distribution, is also performed.

We do not replicate the datails about used datasets (already described in Chapter 3) but simply show their expected embedding parameters and the values predicted by existing methods (whenever available) in Table 10. The embedding dimensions and time delays were defined as single or multiple possible values according to the extensive analysis provided in the related work (Rössler, 1976; Tucker, 1999; Robledo and Moyano, 2007, etc.). It is also important to mention that the results obtained with FNN were only properly estimated after using the ground-truth values for the time delay, depicting a clear limitation. Whenever τ was computed using

AMI, as usually performed in conjunction with FNN, the expected embedding dimension m was hardly ever found.

Table 10: Comparison of embedding parameters (m, τ). From left to right: datasets tested, ground truth (expected values according to the generating rule), results given by existing methods and ours. As one may notice, some methods either only estimate m or τ, whereas ER, MC, and ours estimate both. Terminology n/a denotes datasets without a known ground truth (column 2) or which were not handled by MC (column 8).

Series	Expected	AMI	FNN	AD	SVF	ER	MC	Ours
	(m, τ)	(τ)	(m)	(τ)	(τ)	(m, τ)	(m, τ)	(m, τ)
Logistic	(2–3, 1)	13	2	3	1	(2, 1)	n/a	(2, 1)
Hénon	(2–4, 1)	12	3	3	1	(3, 1)	(2–3, 1–5)	(2,1)
Lorenz	(2–3, 5–12)	17	2	14	55	(5, 1)	n/a	(3, 8–12)
Rössler	(3, 5–12)	13	2	11	10	(5, 1)	n/a	$(3, 5)$
Sunspot	n/a	6	3	10	59	(2, 1)	(2–4, 1–7)	(2, 1)

All time series were composed of $1,000$ observations. We used a 5-resampling validation criterion in all experiments, always taking 75% of data for training and the remaining 25% for testing.

9.4.2 *Logistic And Hénon: Consistency Along Resamplings*

One of the drawbacks of neural networks is the output of different results owing to the random weight initialization. While this aspect is less important for pure classification or regression tasks, it becomes crucial when information is extracted from the network architecture, as in our case (Section 9.3.2).

We have tested that our approach yields consistent results for different datasets and different initializations, reinforcing that a stable pattern is being learned. Figure 9.3(a–e) show the relevances while running the network for five resamplings on the Logistic map. In this circumstance, we considered the search space provided by $(m_{\max} = 5, \tau_{\max} = 3)$.

Figure 9.3(f) shows the average of the five resamplings. Here and next, box plots (McGill et al., 1978) are drawn on each bar to indicate the variance of relevances along resamplings. Very similar results were obtained for the other datasets (not included to avoid redundancy). By analyzing Figure 9.3 while using the threshold procedure outlined in Section 9.3.2, we observe that all resamplings suggest, with high confidence as made evident by narrow box plots, an embedding dimension $m = 2$ and time delay $\tau = 1$, matching the ground truth as desired (Table 10).

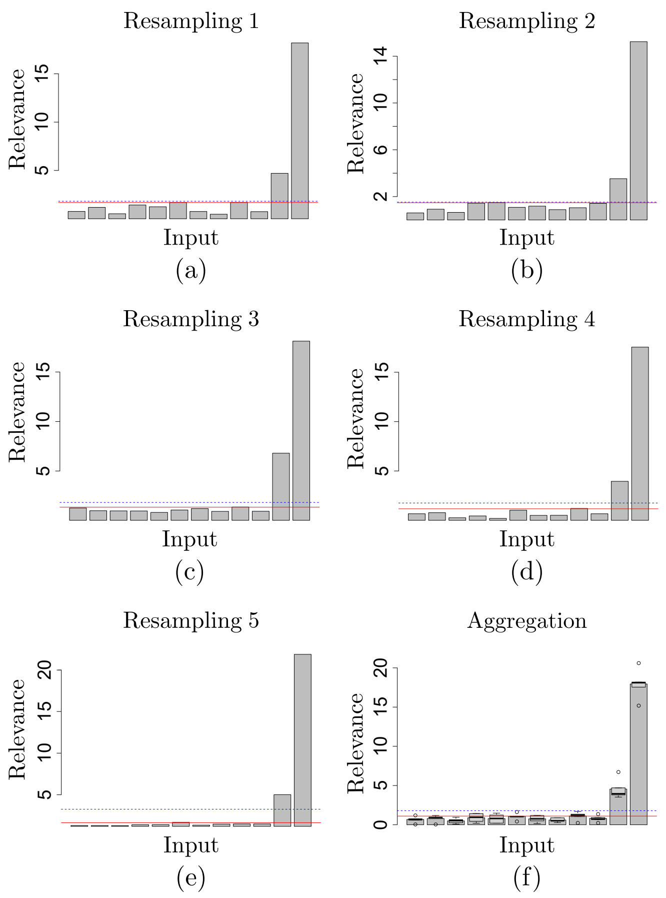


Figure 9.3: **(a–e)** Results of five resamplings for the Logistic map. **(f)** Aggregation of the five resamplings. The maximum embedding dimensions, or MEB, were set as $(m_{\max} = 5, \tau_{\max} = 3)$.

In order to reinforce the robustness of our method with respect to network initialization, we performed three experiments with the Hénon map, using three different random strategies, as outlined in Section 9.4.1. In all situations, we defined the search space using $(m_{\max} = 4, \tau_{\max} = 4)$. Figure 9.4 shows the plots of aggregated importance for these experiments. One may notice very similar relevances and almost the same embedding parameters, regardless the initialization.

9.4.3 *Lorenz: Consistency Along The Search Space*

The Lorenz series is produced from a nonlinear system which is more complex than the Logistic and the Hénon maps. This dataset

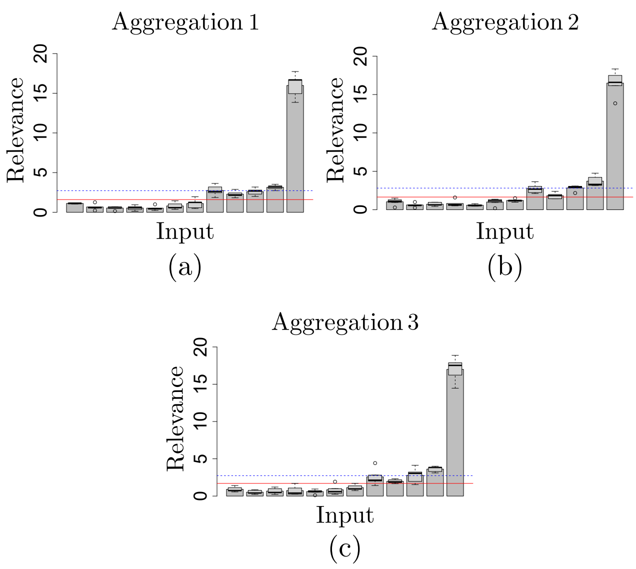


Figure 9.4: Results for the Hénon map under three different initializations. Respective estimatives from **(a–c)**: $(2, 4)$, $(3, 4)$, $(3, 4)$.

is used to study the robustness of our method with respect to variations in the search space. In this sense, as the search space in our case is represented by the number of input neurons, we ran our network under different MEB parametrizations $(m_{\max}, \tau_{\max})$ and analyzed how predictions (m, τ) varied under such conditions. All other network settings remained the same as discussed in Table 9.

The experiment results, shown in Figure 9.5, reinforce that excessively small MEB values may create a network whose architecture is not big enough to capture the system dynamics (Figure 9.5(a)). Conversely, similar values of embedding parameters can be estimated when smaller-to-medium values of MEB are used (Figure 9.5(b-d)). On the other hand, by excessively increasing the search space, it is more difficult to find a clear set of parameters (m, τ) as the model captures more disturbances especially in nonlinear systems such as Lorenz. In such cases, in attempt to obtain a highly confident estimation for (m, τ), one needs to increase the threshold $\epsilon_{\max}$ from our model, as illustrated in Figure 9.5(e,f), where we have increased the upper threshold $\epsilon_{\max}$ to 90% and 65%, respectively.

The experiment also suggests that the range of MEB is an important parameter, but no crucial for the estimation. After applying our method for smaller-to-greater values of MEB, we can see that the network architecture led to similar patterns especially in middle-range values (Figure 9.5(b-d)). This goes in accordance to the Bias-Variance Dilemma, which states that one should choose an algorithm bias that is not too restricted (prone to underfitting) nor too relaxed (where complex functions will tend to overfit/memorize the data). For general systems, we suggest at first to use typical (according to the related work) values of MEB that lead to $|\text{PEV}| - 1 = [12, 30]$ input neurons.

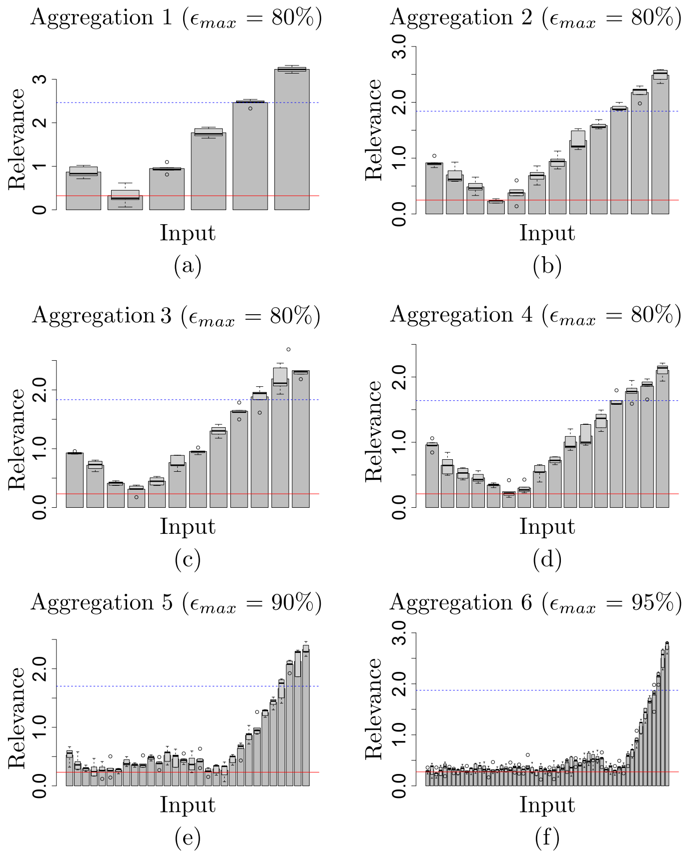


Figure 9.5: Robustness of the estimation of embedding parameters as function of the initial search space $(m_{\max}, \tau_{\max})$. From **(a–f)**, MEB are: $(3, 3)$, $(5, 3)$, $(7, 2)$, $(3, 8)$, $(6, 6)$, $(8, 8)$. Respective estimated parameters: $(2,)$, $(3, 8)$, $(3, 9)$, $(3, 11)$, $(3, 11)$, $(3, 13)$.

In addition, as the network was trained using a different number of inputs (maximum embedding bounds) and its architecture still led to similar outputs of m and τ, this experiment suggests that even using different embeddings, the neural network is robust enough to converge to the Lorenz dynamics (Figure 2.3(b)). This goes in accordance the claiming that m and τ are bounded by the time delay window t_w, and that several tuples (m, τ) can be used to unfold the attractor.

9.4.4 *Rössler: Forecasting Accuracy*

Besides comparing the estimated embedding parameters with known ground truth, a different way of assessing the performance of the proposed neural network is by *predicting* data. We conducted such strategy using the Rössler dataset, another well-known benchmark in the context of Dynamical Systems (Rössler, 1976). Starting with an initial search space set in form $(m_{\max} = 4, \tau_{\max} = 5)$, we obtained the embedding parameters $(m = 3, \tau = 6)$ as shown in Figure 9.6(a). We refer to Section 9.3.2 for details about the blue and the red lines defining upper and lower bounds to support the selection of embedding parameters.

Complementary, Figure 9.6(b) shows the predicted (blue-solid) *vs* the expected (black-dashed) series for a *single* observation forecasting under 250 time steps. The image shows the forecasting using the last k-folded network. As it can be seen, the experiment suggests the network was capable to reveal the dynamics of the dataset.

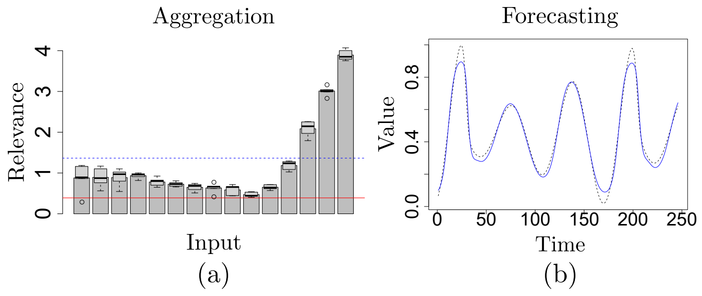


Figure 9.6: Results for the Rössler system. **(a)** Relevance of input neurons. **(b)** Comparison of the 250 forecasted (solid-blue) and expected (dashed-red) observations.

Moreover, despite the forecasted series recovered trends and periodicities, it is worth to mention that our model fails to predict further observations following the butterfly effect (Brock et al., 1992), *i.e.*, when a predicted observation is fed in a recurrent fashion to the dataset to be used as a new query. Such behavior is expected as our neural-network architecture was built to predict a single observation in the future. Figure 9.7 shows the result, where the black-dashed line is the original series and the red line illustrates our forecasting. In that case, we recommend to use our method to estimate the embedding parameters, reconstruct the phase space and apply a different regression model to recursively predict the series.

For instance, the blue line in Figure 9.7 shows the recursive forecasting using the Distance-Weighted Nearest Neighbors (DWNN) (Equation 5.10) applied over the embedding $(m = 3, \tau = 6)$. As it can be seen, results are better within the prediction horizon, *i.e.*, the initial set of observations that can be recursively predicted under some confidence level (Sano and Sawada, 1985; Alligood et al., 1996).

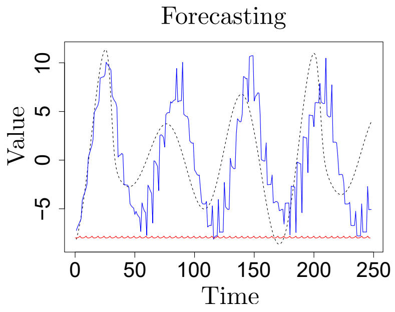


Figure 9.7: Recurrent forecasting of the Rössler system. The dashed-black line indicates the original time series. The solid red line represents the forecasting of our model, where as the solid blue line shows the forecasting results provided by DWNN under a phase space reconstructed using our estimation.

9.4.5 *Sunspot And Normal Distribution: Analyzing Real-World And Noisy Data*

In our last experiment, we evaluated the effectiveness of our method on the Sunspot series (Andrews and Herzberg, 1985), a dataset formed with real-world observations, having a fragment of it illustrated in Figure 9.8(a). In this situation, nothing is known about the series generating rule $R(\cdot)$ and no ground truth is available for assessing the quality of the estimated embedding parameters. In those scenarios, one can rely on the visual analysis and properties of both time series and embeddings (only seeing the first two or three dimensions of it) in attempt to validate the parameter estimation by their similarities to other well-known datasets.

Using our network approach on Sunspots with an initial search space $(m = 4, \tau = 3)$, we found the embedding parameters $(m = 2, \tau = 1)$, as illustrated in Figure 9.8. This estimation is also reinforced by the fact that the Sunspot dataset contains sinusoidal characteristics, as it was already discussed in Section 3.2.5.

As a consequence of analyzing real-world datasets, we also consider a pure-randomly generated time series following a Normal

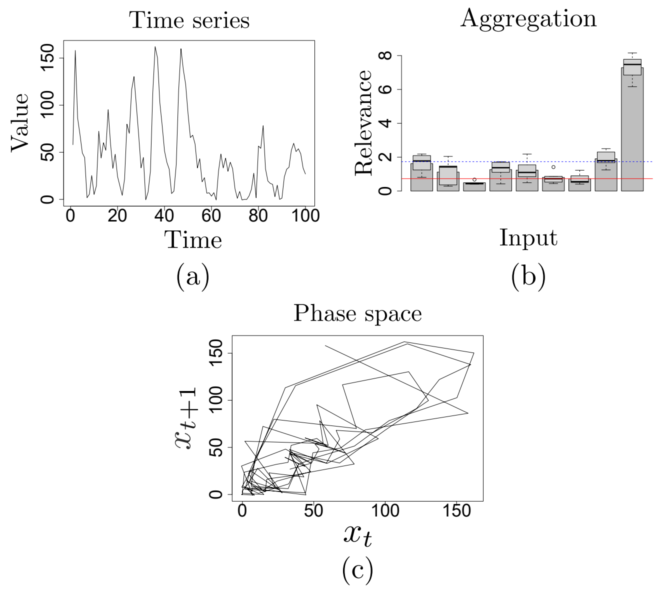


Figure 9.8: Relevance of dimensions for the Sunspot dataset.

distribution $\mathcal{N}(\mu = 0, \sigma^2 = 1)$, where μ and σ correspond to the mean and the standard deviation, respectively. Here, we estimated embedding parameters using an initial search space defined as $(m_{\max} = 5, \tau_{\max} = 5)$. Figure 9.9 shows the results. As one may notice, there is no trivial way to select a subset with the most relevant dimensions (which would provide m), nor a manner to point out a minimum below ϵ_{min} (which would give us τ). Moreover, the variance of relevances is very large for most dimensions, which goes in accordance with Chaos Theory (Alligood et al., 1996). In those circumstances, it is expected that the attractor of stochastic series is fully spread all over the embedding space in some hyperspherical organization (de Mello and Moacir, 2018), such that m is always equal to the maximum embedded dimension.

As a last experiment, we test the robustness of our method after adding Normal-based noise of $\mathcal{N}(0, \{0.2^2, 2^2, 4^2\})$ to the Lorenz system (similar results were obtained for other datasets), using a network with MEB $(m_{\max} = 4, \tau_{\max} = 5)$. Figure 9.10 illustrates the results. For relatively low amount of noise $(\sigma = 0.2)$, our method is still capable of recovering the phase-space dynamics, finding $(m = 3, \tau = 10)$ as embedding parameters, as shown in Figure 9.10(a). As the signal-to-noise ratio decreases, *i.e.*, as the amount of noise increases, the estimation got twisted (as expected), leading to a estimation of $(m = 2, \tau = 3)$ and $(m = 2, \tau = 5)$ for $\sigma = 2$ and $\sigma = 4$, respectively. It is worth to mention, however,

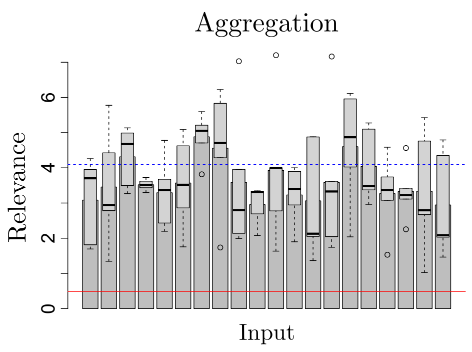


Figure 9.9: Relevance of dimensions for data produced using the Normal distribution $\mathcal{N}(0, 1^2)$. There is no evident manner to select the embedding parameters (m, τ) in this specific scenario.

that this problem leads to different inconsistencies when compared to variations on the search space (Section 9.4.3). There, even when too much dimensions were involved in the training, the variance on the box plots remained low for most of the dimensions. Here, the opposite scenario is observed: box plots show great variations in their quantiles even for few dimensions. Therefore, this experiment also shows that box plots are not just useful to show if the network has converged to solution, but also to qualitatively measure the amount of randomness in the time series. In such context, estimations from Figure 9.10(b) and Figure 9.10(c) are not trustworthy due to high variances over dimension relevances. Moreover, in those cases, its better to first filter the dataset to later proceed with further analysis.

9.5 FINAL CONSIDERATIONS

Several statistical approaches from the literature support time-series analyses, especially in terms of forecasting (Box and Jenkins, 2015). However, these cannot deal with complex and chaotic data. Dynamical Systems tackle such a problem by reconstructing time series into phase spaces, unveiling the relationships among observations, consequently leading to more consistent models. Methods have been proposed for the reconstruction of phase spaces by estimating the embedding parameters m and τ, following Takens' embedding theorem (Takens, 1981). As a main drawback, those methods rely on predefined measurements to compare different phase spaces and estimate the most adequate after analyzing a set of possibilities.

As an alternative, we proposed in this chapter the usage of an artificial neural network with a forgetting mechanism to implicitly

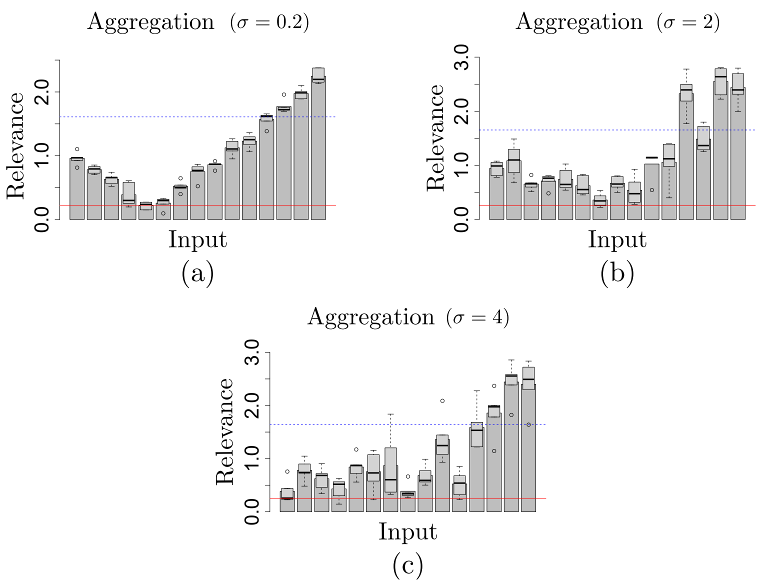


Figure 9.10: From **(a)** to **(c)**, our model estimated $(m = 3, \tau = 10)$, $(m = 2, \tau = 3)$ and, $(m = 3, \tau = 5)$ after adding $\mathcal{N}(0, \{0.2, 2, 4\})$ to the Lorenz system.

learn the embedding parameters while mapping input examples to their expected outputs. Despite similarities that our approach share with the method of (Manabe and Chakraborty, 2007), our method is simpler in the sense that it does not require hidden unit clarification, selective learning, or pruning heuristics during training. The single parameter our approach requires is the maximum embedding bound (MEB), which is used to define the length of the input layer. Moreover, we rely on a different normalization of initial weights, as well as a different criterion to define relevant dimensions, thus positively impacting the estimations of m and τ. We have performed experiments to assess the sensitivity of our approach to different random initializations and search space settings. As made evident throughout the experiments, our method achieved robust and consistent results for several datasets and MEB values.

In conclusion, we claim to have **positively answered research question RQ5: neural networks can be used to estimate the embedding pair.**

Several possible improvements to our work exist, as follows. First, one can attempt to tackle the butterfly effect (Brock et al., 1992) by proposing a network to output recursive forecasting in a more robust way. Secondly, as usual in deep learning, more data (available by considering more dynamical systems for which ground-truth embedding information is available from domain experts) can be used for training, thereby likely leading to networks that generalize better and across a larger palette of time-dependent phenomena.

10 CONCLUSION

We now summarize the main findings on the research work described in this thesis, as follows.

Firstly, we have explored the relationship between entropy and the independence of phase states (Chapter 5). For this, we relied on the Statistical Learning Theory (SLT) framework (Section 5.2) to show some phase spaces (embedding with different parameters m and τ) led to better learning models after applying the same regression function. Further, as SLT requires the input space to be formed by independent-and-identically-distributed (i.i.d.) data, we empirically show that phase states have satisfied such an assumption after testifying generalization. For this study, it was assumed data to come from a controlled environment. Furthermore, we studied the correlation of entropy with different types of embeddings, both numerically (Chapter 5) and visually (Chapter 8). Although we found that optimal embeddings do have a low entropy, the relation was not one-to-one. For example, we found that very different embeddings from the known optimal one can also have low entropy levels. This raises the question of what is, actually, a good definition of an optimal embedding. Intuitively, one would say that deterministic systems generally have well-designed structures in the phase space, as one state maps to a single other in the future. Conversely, stochastic processes tend to have phase states spread all over the phase space, such that such patterns are rarer to happen. However, as outlined above, we could not find a one-to-one correspondence between the distance from an optimal embedding and the entropy level. This means, also, that using entropy as a criterion to compute the parameters of an optimal embedding is a very difficult, if even possible, task.

As part of our studies concerning chaos theory and dynamical systems, we wanted to prove the effectiveness of phase space models. We have performed experimental analysis in the context of time series semi-supervised classification, where we have compared time series and phase space methods using self-learning methods (Chapter 6). As shown there, our method based on state spaces yields more accurate results than state-of-the-art methods. As such, this strengthens our belief that modeling dynamical systems via their (optimal or near-optimal) embeddings in phase space is a useful proposition.

We next revisited the context of the SLT, aiming to come up with a set of criteria and methodology for qualitatively assessing how well learning is ensured by algorithms that attempt to detect concept drift in data streams (Chapter 7). We adapted the SLT to account for this more challenging type of data (as compared to time series drawn from a single phenomenon) and next evaluated several state-of-the-art concept-drift-detection algorithms. Saliently, our evaluation showed that no algorithm (from the studied ones) fully complies with our criteria, thus can ensure learning. This result points out some limitations of existing concept drift detection methods (which should be overcome by future research), although it also provides a new theoretical framework to evaluate all such algorithms in a principled way.

On a more practical side, we examined the challenge of getting a visual insight into large high-dimensional datasets such as created by our time series. We proposed RadViz++, a visual exploration tool that overcomes several limitations of existing data visualization tools based on the radial metaphor (Chapter 8). We validated this tool on several real-world high-dimensional datasets. Next, we tried to visually explore (in another proposal) the set of phase spaces generated by various embeddings. Our exploration revealed several interesting (though, complex to interpret) patterns, and correlated positively with the difficulty of fully characterizing optimal embeddings by low entropy levels.

Finally, we turned back to the problem of estimating optimal embeddings, and used artificial neural networks to address this task (Chapter 9). The proposed approach is simple to implement, fast to execute, has only a few parameters, and yields values for the optimal embedding parameters m and τ which are very close to ground-truth values stated by the literature. Given these results, and the simplicity of our approach, we argue that exploring more refined deep-learning architectures is a promising way forward phase-space reconstruction.

Several future work is possible based on our results. From a theoretical viewpoint, it is interesting (and valuable) to further explore the relationship between entropy and optimal embeddings. By defining what an "optimal" embedding is (which may be a problem-specific question to answer), we believe that stronger correlations with entropy can be found. This aspect can benefit from refining and extending our phase-space visualizations to *e.g.*, show not only how much two phase spaces are different, but *what* precisely makes them different. Along the same lines, methods to visually summarize large and complex trajectories in phase spaces are of interest. Also, ways to visualize trajectories in phase spaces having more than three dimensions, *e.g.*, by the suitable use of dimensionality reduction, are an interesting extension to consider. At the practical endpoint, using different models fed by state-space features could improve the classification and prediction of time series, better than with the current methods employed in this thesis.

BIBLIOGRAPHY

H. D. I. Abarbanel, R. Brown, J. J. Sidorowich, and L. S. Tsimring. The analysis of observed chaotic data in physical systems. *Reviews of Modern Physics*, 65:1331–1392, 1993.

R. Agarwal. *Dynamical Systems and Applications*, volume 4 of *World Scientific series in applicable analysis.* World Scientific, Singapore, Malaysia, 1995.

T. Al-Khateeb, M. M. Masud, L. Khan, C. Aggarwal, J. Han, and B. Thuraisingham. Stream classification with recurring and novel class detection using class-based ensemble. In *Proceedings of the 12th International Conference on Data Mining*, pages 31–40, Brussels, Belgium, 2012. IEEE Press.

A. M. Albano, L. Smilowitz, P. E. Rapp, G. C. de Guzman, and T. R. Bashore. *Dimension calculations in a minimal embedding space: Low-dimensional attractors for human electroencephalograms.* Springer, Berlin-Heidelberg. Germany, 1987.

A. M. Albano, J. Muench, C. Schwartz, A. I. Mees, and P. E. Rapp. Singular-value decomposition and the Grassberger-Procaccia algorithm. *Physical Review A*, 38:3017–3026, 1988.

A. Albano, A. Passamante, and M. E. Farrell. Using higher-order correlations to define an embedding window. *Physica D: Nonlinear Phenomena*, 54(1):85–97, 1991.

K. T. Alligood, T. Sauer, and J. A. Yorke. *Chaos : an introduction to dynamical systems.* Textbooks in mathematical sciences. Springer, New York, United States, 1996.

D. F. Andrews and A. M. Herzberg. *Data: a collection of problems from many fields for the student and research worker.* Springer-Verlag, New York, NY, 1985.

B. R. Andrievskii and A. L. Fradkov. Control of Chaos: Methods and Applications. I. Method. *Automation and Remote Control*, 64(5):673–713, 2003.

A. M. Angel, G. J. Bartolo, and M. Ernestina. Predicting recurring concepts on data-streams by means of a meta-model and a fuzzy similarity function. *Expert Systems with Applications*, 46:87–105, 2016.

M. Ankerst, S. Berchtold, and D. Keim. Similarity clustering of dimensions for an enhanced visualization of multidimensional data. In *Proceedings of Information Visualization*, pages 52–60, Los Alamitos, United States, 1998. IEEE Press.

F. J. Anscombe. Graphs in Statistical Analysis. *The American Statistician*, 27(1):17–21, 1973.

F. D. N. Antonio. tseriesChaos: Analysis of nonlinear time series, 2013. URL CRAN.R-project.org/package=tseriesChaos. Last access on 2019-11-05.

M. Baena-García, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavaldá, and R. Morales-Bueno. Early drift detection method. pages 77–86, 2006.

S. Baluja. Probabilistic Modeling for Face Orientation Discrimination: Learning from Labeled and Unlabeled Data. In *Proceedings of the Conference on Advances in Neural Information Processing Systems*, pages 854–860, Denver, United States, 1999. MIT Press.

D. Bates and D. Watts. *Nonlinear regression analysis and its applications*. Wiley, New York, United States, 1988.

J. Benesty, J. Chen, Y. Huang, and I. Cohen. *Pearson Correlation Coefficient*. Springer, Berlin-Heidelberg, Germany, 2009.

K. P. Bennett and A. Demiriz. Semi-supervised support vector machines. In *Advances in Neural Information Processing Systems*, pages 368–374, Cambridge, United States, 1998. MIT Press.

J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems 24*, pages 2546–2554. Curran Associates, Granada, Spain, 2011.

D. J. Berndt and J. Clifford. Using Dynamic Time Warping to Find Patterns in Time Series. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, pages 359–370, Seattle, United States, 1994. AAAI Press.

E. Bertini, L. Dell'Aquila, and G. Santucci. SpringView: Cooperation of radviz and parallel coordinates for view optimization and clutter reduction. In *Proceedings of the Coordinated and Multiple Views in Exploratory Visualization*, Washington, United States, 2005. IEEE Press.

S. Bhardwaj, S. Srivastava, S. Vaishnavi, and J. R. P. Gupta. Chaotic time series prediction using combination of Hidden

Markov Model and Neural Nets. In *International Conference on Computer Information Systems and Industrial Management Applications*, pages 585–589, Krackow, Poland, 2010. IEEE Press.

A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavaldà. New Ensemble Methods for Evolving Data Streams. In *Proceedings of the 15th International Conference on Knowledge Discovery and Data Mining*, pages 139–148, Paris, France, 2009. ACM Press.

A. Bifet, G. Holmes, and B. Pfahringer. Leveraging Bagging for Evolving Data Streams. In *Machine Learning and Knowledge Discovery in Databases, European Conference*, pages 135–150, Berlin-Heidelberg, Germany, 2010. Springer.

C. M. Bishop. *Pattern recognition and machine learning.* Springer-Verlag, New York, United States, 2006.

A. Björck. *Numerical methods for least squares problems.* Society for Industrial and Applied Mathematics, Philadelphia, United States, 1996.

A. Blum and S. Chawla. Learning from Labeled and Unlabeled Data Using Graph Mincuts. In *Proceedings of the 11th International Conference on Machine Learning*, pages 19–26, San Francisco, United States, 2001. Morgan Kaufmann Publishers.

A. Blum and T. Mitchell. Combining Labeled and Unlabeled Data with Co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory*, pages 92–100, New York, United States, 1998. ACM Press.

S. Boccaletti and J. Bragard. *Handbook of Chaos Control.* Wiley-VCH Verlag GmbH & Co. KGaA, Weinheim, Germany, 2008.

B. Bollobás, A. M. Frieze, and T. I. Fenner. An Algorithm for Finding Hamilton Paths and Cycles in Random Graphs. *Combinatorica*, 7(4):327–341, 1987.

M. Bostock, V. Ogievetsky, and J. Heer. D3: Data-Driven Documents. *Transactions on Visualization & Computer Graphics*, 2011.

G. E. P. Box and G. M. Jenkins. *Time Series Analysis: Forecasting and Control.* Wiley, Hoboken, United States, 5rd edition, 2015.

S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, Cambridge, United Kingdom, 2004.

R. Bracewell. *The fourier transform and its applications.* McGraw-Hill Kogakusha, Boston, United States, 2nd edition, 1978.

J. L. Breedon and N. H. Packard. Nonlinear Analysis of Data Sampled Nonuniformly in Time. *Physics D*, 58(1-4):273–283, 1992.

W. Brock, D. Hsieh, and B. LeBaron. *Nonlinear dynamics, chaos, and instability.* MIT Press, Cambridge, United States, 1992.

B. Broeksema, T. Baudel, and A. Telea. Decision Exploration Lab: A Visual Analytics Solution for Decision Management. *Computer Graphics Forum*, 32(8):158–169, 2013.

D. S. Broomhead and G. P. King. Extracting Qualitative Dynamics from Experimental Data. *Physics D*, 20(2-3):217–236, 1986.

J. Brosz, M. A. Nacenta, R. Pusch, S. Carpendale, and C. Hurter. Transmogrification: casual manipulation of visualizations. In *Proceedings of the Symposium on User Interface Software and Technology*, pages 97–106, St. Andrews, Scotland, 2013. ACM Press.

M. D. Buhmann and M. D. Buhmann. *Radial Basis Functions.* Cambridge University Press, Cambridge, United States, 2003.

J. C. Butcher. A History of Runge-Kutta Methods. *Applied Numerical Mathematics*, 20(3):247–260, 1996.

T. Buzug and G. Pfister. Optimal delay time and embedding dimension for delay-time coordinates by analysis of the global static and local dynamical behavior of strange attractors. *Physical Review A*, 45:7073–7084, 1992a.

T. Buzug and G. Pfister. Comparison of algorithms calculating optimal embedding parameters for delay time coordinates. *Physica D: Nonlinear Phenomena*, 58(1):127–137, 1992b.

W. D. Cai, Y. Q. Qin, and B. R. Yang. Determination of phase-space reconstruction parameters of chaotic time series. *Kybernetika*, 44(4):557–570, 2008.

C. Canuto and A. Tabacco. *Taylor expansions and applications.* Springer, Milan, Italy, 2008.

L. Cao, A. Mees, and K. Judd. Dynamics from multivariate time series. *Physica D: Nonlinear Phenomena*, 121(1):75–88, 1998.

G. Carlsson and F. Mémoli. Classifying clustering schemes. *Foundations of Computational Mathematics*, 13(2):221–252, 2013.

K. Chakraborty, K. Mehrotra, C. K. Mohan, and S. Ranka. Forecasting the behavior of multivariate time series using neural networks. *Neural Networks*, 5(6):961–970, 1992.

D. Chang, K. V. Nesbitt, and K. Wilkins. The Gestalt Principles of Similarity and Proximity Apply to Both the Haptic and Visual Grouping of Elements. In *Proceedings of the 8th Australasian Conference on User Interface-Volume 64*, Ballarat, Australia, 2007. Australian Computer Society.

O. Chapelle and A. Zien. Semi-Supervised Classification by Low Density Separation. 2005.

O. Chapelle, B. Schölkopf, and A. Zien. *Semi-Supervised Learning*. The MIT Press, Cambridge, United States, 2010.

L. Chen. *Curse of Dimensionality*. Springer, New York, United States, 2009.

Y. Chen, B. Hu, E. J. Keogh, and G. Batista. DTW-D: Time Series Semi-supervised Learning from a Single Example. In *Proceedings of the 19th International Conference on Knowledge Discovery and Data Mining*, pages 383–391, Chicago, United States, 2013. ACM Press.

Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista. The UCR Time Series Classification Archive, 2015. URL cs.ucr.edu/~eamonn/time_series_data/. Last access on 2019-11-05.

Z. Y. Chen, B. H. Yao, and M. N. Guo. Algorithm Robustness Analysis for the Choice of Optimal Time Delay of Phase Space Reconstruction Based on Singular Entropy Method. *MATEC Web of Conferences*, 63, 2016.

S. Cheng, W. Xu, and K. Mueller. RadViz Deluxe: An Attribute-Aware Display for Multivariate Data. *Processes*, 5(4):75, 2017.

S. P. Clark. Estimating the fractal dimension of chaotic time series. *Lincoln Laboratory Journal*, 3(1), 1990.

D. Coimbra, R. Martins, T. Neves, A. Telea, and F. Paulovich. Explaining three-dimensional dimensionality reduction plots. *Information Visualization*, 15(2):154–172, 2016.

F. G. da Costa, R. A. Rios, and R. F. de Mello. Using dynamical systems tools to detect concept drift in data streams. *Expert Systems with Applications*, 60:39–50, 2016.

F. G. da Costa, F. S. L. G. Duarte, R. M. M. Vallim, and R. F. de Mello. Multidimensional surrogate stability to detect data stream concept drift. *Expert Systems with Applications*, 87:15–29, 2017.

A. Daneshpazhouh and A. Sami. Entropy-based outlier detection using semi-supervised approach with few positive examples. *Pattern Recognition Letters*, 49:77–84, 2014. ISSN 0167-8655.

W. H. Delashmit and M. T. Manry. Recent Developments in Multilayer Perceptron Neural Networks. 2005.

L. Di Caro, V. Frias-Martinez, and E. Frias-Martinez. Analyzing the Role of Dimension Arrangement for Data Visualization in Radviz. In *Proceedings of the Pacific Asia Knowledge Discovery and Data Mining*, pages 125–132, Hyderabad, India, 2010. Springer-Velag.

F. Diacu. Poincaré and the three-body problem. *Historia Mathematica*, 26(2):175–178, 1999.

M. Ding, C. Grebogi, E. Ott, T. Sauer, and J. A. Yorke. Estimating correlation dimension from a chaotic time series: when does plateau onset occur? *Physica D: Nonlinear Phenomena*, 69(3): 404–424, 1993.

D. Dua and E. Karra Taniskidou. UCI Machine Learning Repository, 2019. URL archive.ics.uci.edu/ml. Last access on 2019-11-05.

J. C. Dunn. Well-Separated Clusters and Optimal Fuzzy Partitions. *Journal of Cybernetics*, 4(1):95–104, 1974.

A. Einstein. *Investigations on the Theory of the Brownian Movement*. Dover Books on Physics Series. Dover Publications, New York, United States, 1956.

M. Ester, H. Kriegel, J. S, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 226–231, Portland, United States, 1996. AAAI Press.

E. R. Faria, J. Gama, and A. C. P. L. F. Carvalho. Novelty detection algorithm for data streams multi-class problems. In *Proceedings of the 28th Annual Symposium on Applied Computing*, pages 795–800, Coimbra, Portugal, 2013. ACM Press.

J. D. Farmer and J. J. Sidorowich. Predicting chaotic time series. *Physical Review Letters*, 59:845–848, 1987.

M. Farzad, H. Tahersima, and H. Khaloozadeh. Predicting the Mackey Glass Chaotic Time Series Using Genetic Algorithm. In *2006 SICE-ICASE International Joint Conference*, pages 5460–5463, Busan, South Korea, 2006. IEEE Press.

P. R. A. Firmino, P. S. de Mattos Neto, and T. A. Ferreira. Correcting and combining time series forecasters. *Neural Networks*, 50: 1–11, 2014.

F. Forstnerič. Embeddings, Immersions and Submersions. In *Stein Manifolds and Holomorphic Mappings*, volume 56 of *Ergebnisse der Mathematik und ihrer Grenzgebiete. 3. Folge / A Series of Modern Surveys in Mathematics*, pages 333–400. Springer, Berlin-Heidelberg, Germany, 2011.

A. M. Fraser and H. L. Swinney. Independent coordinates for strange attractors from mutual information. *Physical Review A*, 33(2):1134–1140, 1986.

J. Gama, P. Medas, G. Castillo, and P. Rodrigues. Learning with Drift Detection. In *Advances in Artificial Intelligence-17th Brazilian Symposium on Artificial Intelligence*, pages 286–295, Berlin-Heidelberg, Germany, 2004a. Springer.

J. Gama, P. Medas, and R. Rocha. Forest Trees for On-line Data. In *Proceedings of the Symposium on Applied Computing*, pages 632–636, Nicosia, Cyprus, 2004b. ACM Press.

J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *Computing Surveys*, 46(4), 2014.

C. A. Garcia and G. Sawitzki. nonlinearTseries: Nonlinear Time Series Analysis, 2015. URL `CRAN.R-project.org/package=nonlinearTseries`. Last access on 2019-11-05.

T. Gautama, D. P. Mandic, and M. M. V. Hulle. A differential entropy based method for determining the optimal embedding parameters of a signal. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 29–32, Hong Kong, China, 2003. IEEE Press.

S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992.

M. Ghomi and R. E. Greene. Relative isometric embeddings of Riemannian manifolds. *Transactions of the American Mathematics Society*, 363(1):63–73, 2011.

J. C. Gower and D. J. Hand. *Biplots*. CRC Press, Boca Raton, United States, 1995.

J. C. Gower, S. Lubbe, and N. Roux. *Understanding biplots*. Wiley, Chichester, United States, 2011.

P. B. Graben. Estimating and improving the signal-to-noise ratio of time series by symbolic dynamics. *Physical Review E-Statistical, Nonlinear, Biological, and Soft Matter Physics*, 64, 2001.

P. Grassberger and I. Procaccia. Measuring the strangeness of strange attractors. *Physica D*, 9:189–208, 1983.

M. Greenacre. *Biplots in practice.* Fundación BBVA, Bilbao, Spain, 2010.

N. Grira, M. Crucianu, and N. Boujemaa. Unsupervised and Semi-supervised Clustering: a Brief Survey. In *A Review of Machine Learning Techniques for Processing Multimedia Content*, Santa Clara, United States, 2004. Springer-Verlag.

D. Hammer, A. Romashchenko, A. Shen, and N. Vereshchagin. Inequalities for Shannon Entropy and Kolmogorov Complexity. *Journal of Computer and System Sciences*, 60(2):442–464, 2000.

L. Han, F. Escolano, E. R. Hancock, and R. C. Wilson. Graph characterizations from von Neumann entropy. *Pattern Recognition Letters*, 33(15):1958–1967, 2012.

M. Han and X. Wang. Robust neural predictor for noisy chaotic time series prediction. *The 2013 International Joint Conference on Neural Networks*, pages 1–5, 2013.

S. Haykin. *Neural Networks and Learning Machines*, volume 10 of *Neural networks and learning machines.* Prentice Hall, New York, United States, 2009.

R. Hegger, H. Kantz, and T. Schreiber. Practical implementation of nonlinear time series methods: The TISEAN package. *Chaos*, 9, 1999.

D. L. Hitzl. Numerical determination of the capture/escape boundary for the Hénon attractor . *Physica D: Nonlinear Phenomena*, 2(2):370–378, 1981.

V. Hodge and J. Austin. A Survey of Outlier Detection Methodologies. *Artificial Intelligence Review*, 22(2):85–126, 2004.

P. Hoffman, G. Grinstein, K. Marx, I. Grosse, and E. Stanley. DNA visual and analytic data mining. In *Proceedings of Visualization*, pages 437–441, Los Alamitos, United States, 1997. IEEE Press.

D. Holten. Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data. *Transactions on Visualization and Computer Graphics*, 12(5):741–748, 2006.

H. Hoogendorp, O. Ersoy, D. Reniers, and A. Telea. Extraction and Visualization of Call Dependencies for Large C/C++ Code Bases: A Comparative Study. pages 121–130, 2009.

G. Hulten, L. Spencer, and P. Domingos. Mining Time-changing Data Streams. In *Proceedings of the 7th International Conference on Knowledge Discovery and Data Mining*, pages 97–106, San Francisco, United States, 2001. ACM Press.

C. Hurter. *Image-Based Visualization: Interactive Multidimensional Data Exploration.* Morgan-Claypool Publishers, San Rafael, United States, 2015.

C. Hurter, A. Telea, and O. Ersoy. MoleView: An Attribute and Structure-Based Semantic Lens for Large Element-Based Plots. *Transactions on Visualization and Computer Graphics*, 17(12): 2600–2609, 2011.

C. Hurter, S. Carpendale, and A. Telea. Color Tunneling: Interactive Exploration and Selection in Volumetric Datasets. In *Proceedings of the Pacific Visualization Symposium*, Yokohama, Japan, 2014. IEEE Press.

K. Ikeda. Multiple-valued stationary state and its instability of the transmitted light by a ring cavity system. *Optics Communications*, pages 257–261, 1979.

N. Indurkhya and F. J. Damerau. *Handbook of Natural Language Processing.* Chapman & Hall/CRC, Boca Raton, United States, 2nd edition, 2010.

A. Inselberg. *Parallel Coordinates: Visual Multidimensional Geometry and Its Applications.* Springer-Verlag, Dordrecht, The Netherlands, 2009.

R. P. Ishii, R. A. Rios, and R. F. Mello. Classification of Time Series Generation Processes Using Experimental Tools: A Survey and Proposal of an Automatic and Systematic Approach. *International Journal of Computer Science Engineering and Technology*, 6(4):217–237, 2011.

J. H. M. Janssens, I. Flesch, and E. O. Postma. Outlier Detection with One-Class Classifiers from ML and KDD. In *Proceedings of the International Conference on Machine Learning and Applications*, pages 147–153, Miami Beach, United States, 2009. Computer Society.

J. Jedrzejowicz and P. Jedrzejowicz. Distance-Based Ensemble Online Classifier with Kernel Clustering. In *Proceedings of the 7th*

KES International Conference on Intelligent Decision Technologies, pages 279–289, Cham, Switzerland, 2015. Springer International Publishing.

P. Joia, D. Coimbra, J. A. Cuminato, F. V. Paulovich, and L. G. Nonato. Local Affine Multidimensional Projection. *Transactions on Visualization & Computer Graphics*, 17(12):2563–2571, 2011.

E. Kandogan. Star Coordinates: A Multi-dimensional Visualization Technique with Uniform Treatment of Dimensions. pages 9–12, 2000.

H. Kantz and T. Schreiber. *Nonlinear Time Series Analysis.* Cambridge nonlinear science series. Cambridge University Press, Cambridge, United Kingdom, 2004.

H. Kantz. A robust method to estimate the maximal Lyapunov exponent of a time series. *Physics Letters A*, 185(1):77–87, 1994.

J. L. Kaplan and J. A. Yorke. *Chaotic behavior of multidimensional difference equations.* Springer, Berlin-Heidelberg, Germany, 1979.

D. S. Karunasinghe and S. Y. Liong. Chaotic time series prediction with a global model: Artificial neural network. *Journal of Hydrology*, 323(1-4):92–105, 2006.

G. Kember and A. Fowler. A correlation function for choosing time delays in phase portrait reconstructions. *Physics Letters A*, 179 (2):72–80, 1993.

M. B. Kennel, R. Brown, and H. D. I. Abarbanel. Determining embedding dimension for phase-space reconstruction using a geometrical construction. *Physical Review A*, 45:3403–3411, 1992.

H. Kim, R. Eykholt, and J. Salas. Nonlinear dynamics, delay times, and embedding windows. *Physica D: Nonlinear Phenomena*, 127 (1-2):48–60, 1999.

G. P. King, R. Jones, and D. Broomhead. Phase portraits from a time series: A singular system approach. *Nuclear Physics B-Proceedings Supplements*, 2:379–390, 1987.

R. Klinkenberg and T. Joachims. Detecting Concept Drift with Support Vector Machines. In *Proc. Seventeenth International Conference on Machine Learning*, pages 487–494, San Francisco, United States, 2000. Morgan Kaufmann Publishers.

H. J. Korsch, H. J. Jodl, and T. Hartmann. *The Duffing Oscillator.* Springer, Berlin-Heidelberg, Germany, 2008.

B. Krawczyk and M. Woźniak. One-class classifiers with incremental learning and forgetting for data streams with concept drift. *Soft Computing*, 19(12):3387–3400, 2015.

J. Kruiger, A. Hassoumi, H. J. Schulz, A. C. Telea, and C. Hurter. Multidimensional Data Exploration by Explicitly Controlled Animation. *Informatics*, 4(3), 2017.

J. B. Kruskal and J. M. Landwehr. Icicle Plots: Better Displays for Hierarchical Clustering. *The American Statistician*, 37(2): 162–168, 1983.

D. Landau and K. Binder. *A guide to monte carlo simulations in statistical physics*. Cambridge University Press, Cambridge New York, 2005.

R. S. Laramee, H. Hauser, H. Doleisch, B. Vrolijk, F. H. Post, and D. Weiskopf. The State of the Art in Flow Visualization: Dense and Texture-Based Techniques. *Computer Graphics Forum*, 23 (2):203–221, 2004.

S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, New York, 2006.

J. M. Lee. *Introduction to smooth manifolds*. Graduate texts in mathematics. Springer, New York/Berlin-Heidelberg, United States/Germany, 2003.

M. Li and Z. H. Zhou. *SETRED: Self-training with Editing*. Springer, Berlin-Heidelberg, Germany, 2005.

W. Liebert and H. Schuster. Proper choice of the time delay for the analysis of chaotic time series. *Physics Letters A*, 142(2): 107–111, 1989.

W. Liebert, K. Pawelzik, and H. G. Schuster. Optimal Embeddings of Chaotic Attractors from Topological Considerations. *EPL (Europhysics Letters)*, 14(6), 1991.

H. R. Loo and M. N. Marsono. Online Data Stream Classification with Incremental Semi-supervised Learning. In *Proceedings of the 2nd Conference on Data Sciences*, pages 132–133, New York, United States, 2015. ACM Press.

U. V. Luxburg and B. Schölkopf. Statistical learning theory: models, concepts, and results. In *Inductive Logic*, volume 10 of *Handbook of the History of Logic*, pages 651–706, Amsterdam, The Netherlands, 2011. Elsevier.

H. G. Ma and C. z. Han. Selection of Embedding Dimension and Delay Time in Phase Space Reconstruction. *Frontiers of Electrical and Electronic Engineering in China*, 1(1):111–114, 2006.

L. van der Maaten and G. Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

Y. Manabe and B. Chakraborty. A novel approach for estimation of optimal embedding parameters of nonlinear time series by structural learning of neural network. *Neurocomputing*, 70(7-9): 1360–1371, 2007.

B. B. Mandelbrot. *The Fractal Geometry of Nature*. W. H. Freeman and Company, San Francisco, United States, 1977.

J. M. Martinerie, A. M. Albano, A. I. Mees, and P. E. Rapp. Mutual information, strange attractors, and the optimal estimation of dimension. *Physical Review A*, 45:7058–7064, 1992.

R. M. Martins, D. B. Coimbra, R. Minghim, and A. Telea. Visual analysis of dimensionality reduction quality for parameterized projections. *Computers & Graphics*, 41:26–42, 2014.

K. Marussy and K. Buza. *SUCCESS: A New Approach for Semi-supervised Classification of Time-Series*. Springer, Berlin-Heidelberg, Germany, 2013.

N. Marwan and C. L. Webber. *Mathematical and Computational Foundations of Recurrence Quantifications*. Springer International Publishing, Cham, Switzerland, 2015.

N. Marwan, M. C. Romano, M. Thiel, and J. Kurths. Recurrence plots for the analysis of complex systems. *Physics Reports*, 438 (5-6):237–329, 2007.

J. McClave. *Statistics*. Pearson, Prentice Hall, Upper Saddle River, United States, 2006.

R. McGill, J. W. Tukey, and W. A. Larsen. Variations of Box Plots. *The American Statistician*, 32(1):12–16, 1978.

T. McLoughlin, R. S. Laramee, R. Peikert, F. H. Post, and M. Chen. Over Two Decades of Integration-Based, Geometric Flow Visualization. *Computer Graphics Forum*, 29(6):1807–1829, 2010.

R. F. de Mello and A. P. Moacir. *A Practical Approach on the Statistical Learning Theory*, volume 1. Springer, New York, United States, 2018.

R. F. de Mello and L. T. Yang. Prediction of dynamical, nonlinear, and unstable process behavior. *The Journal of Supercomputing*, 49(1):22–41, 2009.

R. F. de Mello, Y. Vaz, C. H. Grossi, and A. Bifet. On learning guarantees to unsupervised concept drift detection on data streams. *Expert Systems with Applications*, 117:90–102, 2019.

R. F. de Mello. Improving the performance and accuracy of time series modeling based on autonomic computing systems. *Journal of Ambient Intelligence and Humanized Computing*, 2(1):11–33, 2011.

D. Mena-Torres and J. S. Aguilar-Ruiz. A Similarity-based Approach for Data Stream Classification. *Expert Systems with Applications*, 41(9):4224–4234, 2014.

B. Mendelson. *Introduction to Topology*. Dover Books on Mathematics. Dover Publications, New York, United States, 1975.

Q. Meng and Y. Peng. A new local linear prediction model for chaotic time series. *Physics Letters A*, 370(5-6):465–470, 2007.

M. A. Metzger. Applications of Nonlinear Dynamical Systems Theory in Developmental Psychology: Motor and Cognitive Development. *Nonlinear Dynamics, Psychology, and Life Sciences*, 1 (1):55–68, 1997.

S. Meyn and R. L. Tweedie. *Markov Chains and Stochastic Stability*. Cambridge University Press, Cambridge, United States, 2nd edition, 2009.

D. M. Mount. ANN: Approximate Nearest Neighbors, 2010. URL `cs.umd.edu/~mount/ANN/`. Last access on 2019-11-05.

A. Mucherino, P. J. Papajorgji, and P. M. Pardalos. *k-Nearest Neighbor Classification*. Springer, New York, United States, 2009.

M. Müller. *Dynamic Time Warping*. Springer, Berlin-Heidelberg, Germany, 2007.

T. M. Munzner. *Visualization Analysis and Design*. CRC Press/-Taylor & Francis Group, Boca Raton, United States, 2014.

S. Muthukrishnan. *Data streams : algorithms and applications*. Now Publishers, Boston, United States, 2005.

C. Myers, A. Singer, F. Shin, and E. Church. Modeling chaotic systems with hidden Markov models. In *International Conference on Acoustics, Speech, and Signal Processing*, volume 4, pages 565–568, San Francisco, United States, 1992. IEEE Press.

M. E. J. Newman. The structure and function of complex networks. *Society for Industrial and Applied Mathematics Review*, 45:167–256, 2003.

M. N. Nguyen, X. L. Li, and S. K. Ng. Positive Unlabeled Learning for Time Series Classification. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence-Volume Two*, pages 1421–1426, Barcelona, Spain, 2011. AAAI Press.

K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell. Text Classification from Labeled and Unlabeled Documents Using EM. *Machine Learning*, 39:103–134, 2000.

L. Nonato and M. Aupetit. Multidimensional Projection for Visual Analytics: Linking Techniques with Distortions, Tasks, and Layout Enrichment. *Transactions on Visualization and Computer Graphics*, 2018.

L. Nováková and O. Štěpánková. RadViz and Identification of Clusters in Multidimensional Data. In *Proceedings of Information Visualisation*, pages 104–109, Washington, United States, 2009. IEEE Press.

L. Nováková and O. Štěpánková. Visualization of trends using RadViz. *Journal of Intelligent Information Systems*, 37(3), 2011.

J. Ono, F. Sikansi, D. C. Correa, F. V. Paulovich, A. Paiva, and L. G. Nonato. Concentric RadViz: Visual Exploration of Multitask Classification. In *Proceedings of the Conference on Graphics, Patterns and Images*, pages 165–172, Salvador, Brazil, 2015. IEEE Press.

M. Otani and A. J. Jones. Guiding Chaotic Orbits. Technical report, Imperial College of Science Technology and Medicine, London, United Kingdom, 1997.

E. Ott. *Chaos in Dynamical Systems.* Cambridge University Press, Cambridge, United Kingdom/United States, 2002.

N. Packard, J. Crutchfield, J. Farmer, and R. Shaw. Geometry from a Time Series. *Physical Review Letters*, 45(9):712–716, 1980.

E. S. Page. Continuous Inspection Schemes. *Biometrika*, 41:100–115, 1954.

L. d. C. Pagliosa and R. F. de Mello. Applying a kernel function on time-dependent data to provide supervised-learning guarantees. *Expert Systems with Applications*, 71:216–229, 2017.

L. d. C. Pagliosa and R. F. de Mello. Semi-supervised time series classification on positive and unlabeled problems using cross-recurrence quantification analysis. *Pattern Recognition*, 80:53–63, 2018.

L. d. C. Pagliosa and A. C. Telea. RadViz++: Improvements on Radial-Based Visualizations. *Informatics*, 6(2), 2019.

L. d. C. Pagliosa, P. Pagliosa, and L. Nonato. Understanding Attribute Variability in Multidimensional Projections. In *Technical Papers of the 29th Conference on Graphics, Patterns and Images*, pages 297–304, São Paulo, Brazil, 2016. IEEE Press.

L. A. Pereira and R. da Silva Torres. Semi-supervised transfer subspace for domain adaptation. *Pattern Recognition*, 75:235–249, 2018.

F. H. Post, B. Vrolijk, H. Hauser, R. S. Laramee, and H. Doleisch. The State of the Art in Flow Visualisation: Feature Extraction and Tracking. *Computer Graphics Forum*, 22(4):775–792, 2003.

O. Postolache, J. M. Dias Pereira, and P. Girão. Application of RBF Neural Network in ADC Resolution Enhancement. 4:89–92, 1999.

R Development Core Team. R: A Language and Environment for Statistical Computing, 2008. URL `R-project.org`. Last access on 2019-11-05.

S. Raschka. Naive bayes and text classification I-introduction and theory. *Computing Research Repository*, 2014.

C. E. Rasmussen and Z. Ghahramani. Occam's razor. In *Advances in Neural Information Processing Systems 13*, pages 294–300, Cambridge, United States, 2001. MIT Press.

C. A. Ratanamahatana and E. Keogh. Everything you know about Dynamic Time Warping is Wrong. 2004.

C. A. Ratanamahatana and D. Wanichsan. *Stopping Criterion Selection for Efficient Semi-supervised Time Series Classification.* Springer, Berlin-Heidelberg, Germany, 2008.

P. E. Rauber, S. G. Fadel, A. X. Falcão, and A. C. Telea. Visualizing the Hidden Activity of Artificial Neural Networks. *Transactions on Visualization and Computer Graphics*, 23(1):101–110, 2017.

B. Ravindra and P. Hagedorn. Invariants of chaotic attractor in a nonlinearly damped system. *Journal of applied mechanics*, 65, 1998.

D. Reniers, L. Voinea, O. Ersoy, and A. Telea. The Solid* Toolset for Software Visual Analytics of Program Structure and Metrics Comprehension: From Research Prototype to Product. *Science of Computer Programming*, 79:224–240, 2014.

R. A. Rios. *Improving time series modeling by decomposing and analysing stochastic and deterministic influences.* PhD thesis, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2013.

R. A. Rios and R. F. de Mello. Improving time series modeling by decomposing and analyzing stochastic and deterministic influences. *Signal Processing*, 93(11):3001–3013, 2013.

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

J. Rissanen. Modeling by Shortest Data Description. *Automatica*, 14(5):465–471, 1978.

E. S. Ristad, P. N. Yianilos, and S. Member. Learning string edit distance. *Transactions on Pattern Analysis and Machine Intelligence*, 20:522–532, 1998.

A. Robledo and L. Moyano. Dynamical properties of superstable attractors in the logistic map. 2007.

L. Rokach and O. Maimon. *Clustering Methods.* Springer, Boston, United States, 2005.

M. T. Rosenstein, J. J. Collins, and C. J. D. Luca. A practical method for calculating largest Lyapunov exponents from small data sets. *Physica D: Nonlinear Phenomena*, 65(1):117–134, 1993.

M. T. Rosenstein, J. J. Collins, and C. J. D. Luca. Reconstruction expansion as a geometry-based framework for choosing proper delay times. *Physica D: Nonlinear Phenomena*, 73(1):82–98, 1994.

O. Rössler. An equation for continuous chaos. *Physics Letters A*, 57(5):397–398, 1976. ISSN 0375-9601.

R. Y. Rubinstein and D. P. Kroese. *Simulation and the Monte Carlo Method (Wiley Series in Probability and Statistics).* Wiley, Newark, United States, 2 edition, 2007.

M. Rubio-Sánchez, A. Sanchez, and D. J. Lehmann. Adaptable Radial Axes Plots for Improved Multivariate Data Visualization. *Computer Graphics Forum*, 36(3):389–399, 2017.

M. Rubio-Sanchez, L. Raya, F. Díaz, and A. Sanchez. A comparative study between RadViz and Star Coordinates. *Transactions on Visualization and Computer Graphics*, 22(1):619–628, 2015.

M. Sano and Y. Sawada. Measurement of the Lyapunov Spectrum from a Chaotic Time Series. *Physical Review Letters*, 55:1082–1085, 1985.

W. Schefler. *Statistics: concepts and applications.* Benjamin-Cummings Publishing Company, Menlo Park, United States, 1988.

T. Schreiber and A. Schmitz. Improved Surrogate Data for Nonlinearity Tests. *Physical Review Letters*, 77:635–638, 1996.

J. Serrà, X. Serra, and R. G. Andrzejak. Cross recurrence quantification for cover song identification. *New Journal of Physics*, 11, 2009.

J. Sharko, G. Grinstein, and K. A. Marx. Vectorized Radviz and Its Application to Multiple Cluster Datasets. *Transactions on Visualization and Computer Graphics*, 14(6):1444–1427, 2008.

R. Sheikhpour, M. A. Sarram, S. Gharaghani, and M. A. Z. Chahooki. A Survey on semi-supervised feature selection methods. *Pattern Recognition*, 64:141–158, 2017.

R. R. O. d. Silva, P.E. Rauber, R. M. Martins, R. Minghim, and A. Telea. Attribute-based Visual Explanation of Multidimensional Projections. In *Proceedings of the International EuroVis Workshop on Visual Analytics*, Cagliari, Italy, 2015. Eurographics.

M. Sperber. Quadtree and Octree. In *Encyclopedia of GIS*, pages 1695–1700. Springer International Publishing, Cham, Switzerland, 2017.

J. Stark. Delay Embeddings for Forced Systems. I. Deterministic Forcing. *Journal of Nonlinear Science*, 9(3):255–332, 1999.

A. Stefánsson, N. Končar, and A. J. Jones. A note on the Gamma test. *Neural Computing & Applications*, 5(3):131–133, 1997.

F. Takens. Detecting strange attractors in turbulence. In *Dynamical systems and turbulence*, pages 366–381. Springer, Berlin-Heidelberg, Germany, 1981.

A. Tamma and B. Lachman Khubchandani. Accurate determination of time delay and embedding dimension for state space reconstruction from a scalar time series. *ArXiv e-prints*, 2016.

S. C. Tan, K.M. Ting, and T.F. Liu. Fast anomaly detection for streaming data. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 1511–1516, Barcelona, Spain, 2011. AAAI Press.

E. Tejada, R. Minghim, and L. G. Nonato. On Improved Projection Techniques to Support Visual Exploration of Multidimensional Data Sets. *Information Visualization*, 2(4):218–231, 2003.

A. Telea. *Data Visualization: Principles and Practice.* Taylor & Francis, Boca Raton, United States, 2nd edition, 2014.

J. Theiler. Efficient algorithm for estimating the correlation dimension from a set of discrete points. *Physical Review A*, 36: 4456–4462, 1987.

J. Theiler. Estimating fractal dimension. *Journal of the Optical Society of America*, 7(6):1055–1073, 1990.

J. Theiler, S. Eubank, A. Longtin, B. Galdrikian, and J. D. Farmer. Testing for nonlinearity in time series: the method of surrogate data. *Physica D: Nonlinear Phenomena*, 58(1):77–94, 1992.

R. Tibshirani. Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1996.

L. Tu. *An Introduction to Manifolds.* Universitext. Springer New York, New York, United States, 2010.

W. Tucker. The Lorenz attractor exists. *Comptes Rendus de l'Académie des Sciences-Series I-Mathematics*, 328(12):1197–1202, 1999.

A. Tung, X. Xu, and B. C. Ooi. CURLER: Finding and visualizing nonlinear correlation clusters. In *Proceedings of the International Conference on Management of Data*, pages 233–245, Baltimore, United States, 2005. ACM Press.

R. M. M. Vallim and R. F. De Mello. Proposal of a New Stability Concept to Detect Changes in Unsupervised Data Streams. *Expert Systems with Applications*, 41(16):7350–7360, 2014.

T. Van Leeuwen and C. Jewitt. *The Handbook of Visual Analysis.* SAGE Publications, London/Thousand Oaks, United Kingdom/United States, 2000.

V. N. Vapnik. *Statistical Learning Theory.* Wiley, New York, United States, 1 edition, 1998.

A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang. Phoneme Recognition Using Time-delay Neural Networks. In *Readings in Speech Recognition*, pages 393–404. Morgan Kaufmann Publishers, San Francisco, United States, 1990.

L. Wang, J. Zhang, and H. Li. An Improved Genetic Algorithm for TSP. In *International Conference on Machine Learning and Cybernetics*, volume 2, pages 925–928, Hong Kong, China, 2007. IEEE Press.

S. Wang, L. L. Minku, D. Ghezzi, D. Caltabiano, P. Tino, and X. Yao. Concept drift detection for online class imbalance learning. In *The 2013 International Joint Conference on Neural Networks*, pages 1–10, Dallas, United States, 2013. IEEE Press.

S. Wang, J. Lu, X. Gu, H. Du, and J. Yang. Semi-supervised linear discriminant analysis for dimension reduction and classification. *Pattern Recognition*, 57:179–189, 2016.

M. Ward, G. Grinstein, and D. Keim. *Interactive Data Visualization: Foundations, Techniques, and Applications.* A. K. Peters, Natick, United States, 2010.

M. Wattenberg, F. Viégas, and I. Johnson. How to Use t-SNE Effectively. *Distill*, 2016.

L. Wei and E. Keogh. Semi-supervised Time Series Classification. In *Proceedings of the 12th International Conference on Knowledge Discovery and Data Mining*, pages 748–753, New York, United States, 2006. ACM Press.

H. Whitney. Differentiable manifolds. *Annals of Mathematics. Second Series*, 37(3):645–680, 1936.

A. Wong, L. Wu, P. B. Gibbons, and C. Faloutsos. Fast estimation of fractal dimension and correlation integral on stream data. *Information Processing Letters*, 93(2):91–97, 2005.

P. C. Wong and R. D. Bergeron. 30 Years of Multidimensional Multivariate Visualization. In *Scientific Visualization, Overviews, Methodologies, and Techniques*, pages 3–33, Washington, United States, 1997. IEEE Press.

H. Wu and S. Prasad. Semi-supervised dimensionality reduction of hyperspectral imagery using pseudo-labels. *Pattern Recognition*, 74:212–224, 2018.

H. L. Yap and C. Rozell. Stable Takens' Embeddings for Linear Dynamical Systems. *Transactions on Signal Processing*, 59(10): 4781–4794, 2011.

H. L. Yap, A. Eftekhari, M. B. Wakin, and C. J. Rozell. A first analysis of the stability of Takens' embedding. In *Global Conference on Signal and Information Processing*, pages 404–408, Atlanta, United States, 2014. IEEE Press.

G. G. Zanabria, L. G. Nonato, and E. Gomez-Nieto. iStar (i*): An interactive star coordinates approach for high-dimensional data exploratio. *Computers & Graphics*, 60:107–118, 2016.

M. D. Zeiler and R. Fergus. Visualizing and Understanding Convolutional Networks. In *Computer Vision-European Conference on Computer Vision*, pages 818–833, Cham, Switzerland, 2014. Springer International Publishing.

S. Zhong. Semi-Supervised Sequence Classification with HMMs. In *The Florida AI Research Society Conference*, pages 568–574, Plato Alto, United States, 2004. AAAI Press.

F. Zhou, W. Huang, J. Li, Y. Huang, Y. Shi, and Y. Zhao. Extending Dimensions in Radviz based on mean shift. In *Proceedings of Pacific Visualization Symposium*, pages 111–115, Hangzhou, China, 2015. IEEE Press.

X. Zhu, A. B. Goldberg, R. Brachman, and T. Dietterich. *Introduction to Semi-Supervised Learning*. Morgan and Claypool Publishers, San Rafael, United States, 2009.

ACKNOWLEDGMENTS

I would like to thank my supervisors for the guidance, patience, and teachings – either in academic matters as in life. To FAPESP (São Paulo Research Foundation, processes 2015/22406-4 and 2018/10652-9) for the financial support. To my friends and colleagues in Brazil and in The Netherlands for the good times together. I dedicate this manuscript to my family.

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

http://code.google.com/p/classicthesis/

Final Version as of January 15, 2020 (`classicthesis`).