UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas
Programa de Pós-Graduação em Ciência da Computação

João Batista Borges Neto

THE DYNAMICS OF INTERNET OF THINGS

Belo Horizonte
2021

João Batista Borges Neto

A DINÂMICA DA INTERNET DAS COISAS

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Minas Gerais, como requisito parcial à obtenção do título de Doutor em Ciência da Computação.

Orientador: Antonio Alfredo Ferreira Loureiro

Coorientador: Heitor Soares Ramos Filho

Belo Horizonte

2021

João Batista Borges Neto

THE DYNAMICS OF INTERNET OF THINGS

Thesis presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Doctor in Computer Science.

Advisor: Antonio Alfredo Ferreira Loureiro

Co-Advisor: Heitor Soares Ramos Filho

Belo Horizonte
2021

Borges Neto, João Batista.

B732d The dynamics of internet of things [manuscrito] / João Batista Borges Neto – 2021.
228 f. il.

Orientador: Antonio Alfredo Ferreira Loureiro.
Coorientador: Heitor Soares Ramos Filho.
Tese (doutorado) - Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação.
Referências: f.208-225.

1. Computação – Teses. 2. Internet das coisas – Teses. 3. Sistemas colaborativos – Teses. 4. Análise de séries temporais – Teses. I. Loureiro, Antonio Alfredo Ferreira. II. Ramos Filho, Heitor Soares II. Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Ciência da Computação. III. Título.

CDU 519.6*75 (043)

Ficha catalográfica elaborada pela bibliotecária Belkiz Inez Rezende Costa
CRB 6/1510 – Insituto de Ciências Exatas da UFMG

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

The Dynamics of Internet of Things

JOÃO BATISTA BORGES NETO

Tese defendida e aprovada pela banca examinadora constituída pelos Senhores:

[illegible]

PROF. ANTONIO ALFREDO FERREIRA LOUREIRO - Orientador
Departamento de Ciência da Computação - UFMG

[illegible]

PROF. HEITOR SOARES RAMOS FILHO - Coorientador
Departamento de Ciência da Computação - UFMG

[illegible]

PROF. ALEJANDRO CÉSAR FRERY ORGAMBIDE
School of Mathematics and Statistics - VUW

[illegible]

PROF. OSVALDO ANIBAL ROSSO
Instituto de Física - UFAL

[illegible]

PROFA. THAIS VASCONCELOS BATISTA
Departamento de Informática e Matemática Aplicada - UFRN

[illegible]

PROF. PEDRO OLMO STANCIOLI VAZ DE MELO
Departamento de Ciência da Computação - UFMG

[illegible]

PROF. MARIO SÉRGIO FERREIRA ALVIM JÚNIOR
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 25 de Novembro de 2021.

To my wife Patrícia and our son João Carlos.

Acknowledgments

The completion of this work would not have been possible without the help of many people along the way, which I am deeply grateful.

I especially thank my wife Patrícia and our son João Carlos, for being my source of motivation and inspiration, helping me to renew my strength to overcome the various challenges along this path. To my wife, for her dedication and support, for always being by my side, and for helping me build a beautiful family with love and care. To my son who, even without knowing it, helped me a lot to keep my feet on the ground and always seek to be a better person, to be a source of inspiration for him in the pursuit of his own goals. I deeply thank them for helping me to face the adversities more lightly.

I also thank all my family, without exception, especially to my mother Eliza, my brother Max, my grandmother Salomé (*in memoriam*), my grandfather João Borges (*in memoriam*), who have always been by my side, helping me to always believe in my goals. I would also like to give a special thanks to Carlos, Guacira, Dona Joana, Daniela, Carolina, Pablo, Aline, Anny, Isabela e Arthur, for all the prayers, positive thoughts, and words of encouragement that helped me to keep going.

I would like to thank my advisor, Prof. Antonio Loureiro, for believing in this project and for providing the opportunities for its realization. I also thank him for all his teachings and guidance, which since the beginning of my academic studies, even before I met him, had guided me through his works. For all the patience and motivation, always believing in me and always available to talk and help on both academic and professional matters, I will always be grateful.

I would also like to thank my co-advisor, Prof. Heitor Ramos, for all his help, which was essential to determine the path in the development of this work. To all professors and collaborators, especially to Aline Viana, Alejandro Frery, Osvaldo Rosso, Raquel Mini, and Renato Assunção, I deeply thank you for all the teachings and for their valuable contributions to overcome all the challenges over the years. I also thank professors Mário Alvim, Pedro Olmo, and Thaís Batista, for their essential contributions and corrections in order to obtain the final version of this work.

To my friends and colleagues, Prof. João Medeiros and Prof. Luiz Paulo Barbosa, a special thanks for all the support and for all the enlightening conversations and discussions, helping me in the moments of doubts and reflections. I would also like to thank all the friends of the DCT/UFRN, in especial to professors Adrianne Paula, Almir Miranda, Deilson Tavares (*in memoriam*), Fabrício Vale, Flavius Gorgônio, Márcio Barboza, Gilson Gomes, Humberto Rabelo, Karliane Vale, Taciano Silva, and to staff members Kattarine Lucena and Anderson Freire, for all their support and for allowing me to be focused on the development of this work. I also thank my friends, professors Alexandre Queiroga and Walber Silva, for the all discussions and conversations, which were very helpful and enlightening moments. I am also very grateful to all members of CERES and of the whole UFRN, for all the support, which allowed me to develop my research and to conclude my professional and academic formation.

To all professors and staff members of DCC/UFMG, I would like to thank everyone for all the support and conduct, which makes the PPGCC a special place for the development of high level research in our country. In particular, I would especially like to thank Sônia Borges, for all her immeasurable support, helping me with everything I needed, from the first to the last day, literally. I would like to thank all friends from Wisemap/UFMG who helped me during this work. Especially to Ana Paula, Bruno Rodrigues, Bruno Santos, Clayson, Evandro, Felipe, Guilherme, Ilo, Ivan, Kássio, Leonardo Cotta, Lucas, Manassés, Paulo, Thiago, and Toninho, for their help and for providing always enjoyable and encouraging moments.

"Time moves in one direction, memory another."
(William Gibson)

Resumo

Este trabalho investiga o comportamento dinâmico dos dados de sensores na Internet das Coisas (IoT, do inglês Internet of Things). Devido ao crescente número de iniciativas na IoT, com seu impressionante número de dispositivos coletando um grande volume de dados de fenômenos do mundo real, há uma iminente necessidade de soluções adequadas aos seus desafios. Uma parte importante da atual IoT é a Internet das Coisas Colaborativa (CoIoT, do inglês Collaborative IoT), que é composta, principalmente, por componentes baratos e mantidos por usuários comuns, afetando os dados gerados. Assim, soluções para a IoT devem considerar o aprimoramento da segurança de seus dispositivos, bem como a qualidade e confiabilidade dos seus dados, mas sendo a eficiência e robusto aos desafios deste novo cenário.

Um tópico que vem sendo usado com sucesso para compreender mais profundamente fenômenos do mundo real é o estudo da dinâmica, que visa entender como sistemas evoluem com o tempo. Uma importante ferramenta com sólidos resultados na análise da dinâmica de séries temporais é a transformação de padrões ordinais. Contudo, embora a dinâmica tenha o potencial de servir de base para novos domínios de representação para a análise de dados na IoT, há questões em suas transformações que devem ser tratadas para sua aplicação adequada.

Este trabalho tem como objetivos avançar o estado da arte na análise da dinâmica de séries temporais, em sua adequação para os desafios da IoT, e propor soluções baseadas em comportamentos dinâmicos para o uso mais confiável dos dados da IoT. Para avançar na aplicabilidade das transformações de padrões ordinais para cenários desafiadores, como é o caso da IoT, são propostas estratégias em duas principais direções. Uma primeira estratégia tem como objetivo prover a mínima dependência na seleção de parâmetros na transformação, considerando o comportamento multiescala de uma nova métrica proposta, a probabilidade de auto transições, que se mostraram úteis na distinção de dinâmicas de séries temporais. A segunda estratégia consiste em um índice de separabilidade de classes, que é um valioso método para estimar os parâmetros mais adequados para as transformações de padrões ordinais, no contexto da classificação de séries temporais. Em respeito à aplicação da análise da dinâmica de séries temporais para os cenários de IoT, primeiramente são dados esclarecimentos quanto ao contexto da CoIoT. Nós provemos um melhor entendimento sobre as principais características e propriedades dos dados gerados por seus sensores e seus principais problemas. Em seguida, são propostas estratégias para a classificação de dados de fenômenos físicos coletados pelos sensores da CoIoT e um método para incrementar a segurança dos dispositivos da IoT contra ataques de botnet, ambos considerando seus comportamentos dinâmicos. As estratégias propostas foram comparadas com trabalhos relacionados e os resultados demonstraram seus potenciais no avanço da aplicabilidade das transformações de padrões ordinais para os cenários da IoT. Nós mostramos que a construção desta nova representação auxilia na escalabilidade, evitando comparações com uma grande quantidade de dados, sendo robusta para os problemas dos dados da CoIoT. Assim, por meio dessas abordagens, é possível desenvolver soluções para a IoT que podem se beneficiar dos aspectos únicos de sistemas dinâmicos.

Palavras-chave: Internet das Coisas, Sensoreamento Colaborativo, Dinâmica de Séries Temporais, Transformações de Padrões Ordinais..

Abstract

This thesis investigates the dynamical behavior of time series data from Internet of Things (IoT) sensors. Because of the growing number of IoT initiatives, with its impressive number of devices collecting a large amount of data from real-world phenomena, there is an imminent need for solutions that are adequate to their issues. For instance, an important part of the current IoT is the Collaborative Internet of Things (CoIoT), which is mainly composed by cheap components and managed by common users, affecting the generated data. Thus, solutions for IoT must consider improving the security of those devices, as well as the quality and reliability their data, but being efficient and robust to the issues from this novel scenario.

A subject that has been successfully used for a deeper comprehension of several real-world phenomena is the study of dynamics, which aims to understand systems that evolve in time. An important tool with solid results concerning the analysis of time series dynamics is the ordinal patterns transformation. However, while the dynamics has the potential to be the basis for novel representation domains to the analysis of IoT data, there are issues on their transformations that must be handled for their proper applicability.

This work aims to advance the state-of-the-art in the analysis of time series dynamics, to be adequate for the IoT issues, and to propose solutions based on dynamical behavior for a more reliable use of data from IoT. In order to advance the applicability of ordinal patterns transformations for challenging scenarios, such as IoT, we propose strategies in two main directions. A first strategy is aimed to provide minimum dependency on the selection of parameters by the transformation, by considering the multiscale behavior of a novel proposed metric, the probability of self-transitions, which are shown to be useful for the distinction of time series dynamics. The second strategy consists of a class separability index, which is a valuable method to estimate the most adequate parameters for the ordinal patterns transformations, in the context of time series classification problems. With respect to the application of the analysis of time series dynamics to IoT scenarios, we first give an enlightenment on the CoIoT.

We provide a better understanding on the main characteristics and properties of data that are being generated by their sensors and its inherent problems. Then, we provide strategies for the classification of physical phenomena data collected by CoIoT sensors and a method to increase the security of IoT devices against botnet attacks, both considering their dynamical behavior. The proposed strategies were compared to related work and the results show their potentials on advancing the applicability of ordinal patterns transformations for the IoT scenarios. We show that the construction of this novel representation helps in the scalability, avoiding comparisons with a large number of data, and being robust to the problems of CoIoT data. Thus, by following these approaches, it is possible to develop solutions for IoT scenarios that can benefit from the unique aspects of dynamical systems.

Keywords: Internet of Things, Collaborative Sensing, Time Series Dynamics, Ordinal Patterns Transformations.

List of Figures

List of Tables

List of Algorithms

List of Theorems and Definitions

Contents

Chapter 1

Introduction

The study of dynamics aims to understand systems that evolve in time [Strogatz, 2018]. It is an important aspect to be concerned when dealing with time ordered data [Borges et al., 2019a]. For systems based on some learning strategy, to understand its dynamics is an essential step to a proper understanding of the data they are based on. When analyzing a system's dynamics, the main interest is to understand its behavior when changing from a previous state to the next one [Birkhoff, 1927]. For instance, with dynamics it is possible to discover if the system's behavior is deterministic, periodic, chaotic, stochastic, or even a combination of them [Rosso et al., 2007a].

The subject of dynamics has been used as a valuable tool for a deeper comprehension of several real-world phenomena. From the motion of planets or the swinging of a pendulum, to biological or environmental conditions, these are just a few examples of its applicability. For instance, one of the first studies from Lorenz, in 1963, which led to the development of key ideas of chaotic motion, was performed with the intuition of better understanding the behavior of the weather [Strogatz, 2018]. Thus, because of the inherent nature of their devices, which are built to interact with real-world phenomena, the Internet of Things (IoT) [Atzori et al., 2010; Miorandi et al., 2012; Gubbi et al., 2013] has a perfect application scenario that can benefit from the studies of dynamics. However, despite its notorious applicability, few studies are devoted to the analysis of dynamics in IoT [Borges et al., 2019b].

Indeed, any solution intended to be applied to IoT scenarios must deal with a hard challenge, which is the need to handle the impressive number of devices and generated data. For instance, to search for a specific sensor from this large number of available ones is a great challenge, requiring scalable and efficient solutions [Perera et al., 2014b; Stankovic, 2014]. Consequently, even for the best case scenario, simple algorithms should consider processing a lot of instances as the rule, not the exception. In addition to that, the high heterogeneity of devices, technologies, services, and environments is another challenge present in the current IoT scenarios [Gubbi et al., 2013; Stankovic, 2014].

An important part of the current IoT is the Collaborative Internet of Things (CoIoT) [Borges Neto et al., 2015]. CoIoT is mainly composed by Do-It-Yourself (DIY) initiatives, where common users deploy their own devices and collaboratively decide to make their data open and publicly available. However, due to their cheap components and the lack of precautions, or even knowledge, by their users, the quality of data produced is a concerning issue [Austen, 2015; Gao et al., 2015; Chen et al., 2016]. Furthermore, data and resources from CoIoT are usually not described or, when they are, they have a very poor description. Even if the descriptions are given, there is no guarantee that they are correct since there is no validation. This makes the search for some keyword or expression prone to errors [Borges Neto et al., 2015]. In this context, a reasonable direction to develop solutions able to overcome the challenging aspects of IoT is given by its dynamics.

1.1 Objectives

The main objectives of this thesis are to advance the state-of-the-art in the analysis of time series dynamics and propose solutions for more reliable use of data from CoIoT. More specifically, we aim to develop solutions based on the dynamics of time series data that could overcome the challenges present in CoIoT scenarios. Thus, we expect that this can lead to improvements in the reliability of the CoIoT-based systems.

Due to the characteristics of CoIoT scenarios, their generated data are subject to some challenging issues that directly affect their quality. For instance, they generate a large amount of time-ordered data that present issues such as missing samples, uneven time spaces between consecutive samples, and with unreliable precision. Thus, since the dynamics is an intrinsic aspect of the phenomena, it can be used to understand the behavior behind the data generated by those sensors. This may lead to a unified representation to model data collected from heterogeneous sensors at different scales and sampling rates.

The construction of this novel representation also helps in the scalability problem of CoIoT by avoiding comparisons with a large number of data. Instead, a small set of features can represent distinct aspects of the dynamics of a given phenomenon. Thus, by following this approach, developing CoIoT solutions can benefit from the unique characteristics of dynamical systems. Particularly, solutions based on learning processes, such as machine learning techniques of characterization or classification, can create models from the dynamical behavior of CoIoT sensors, being able to make better distinctions and predictions among them.

1.2 Contributions

The main contributions of this thesis are presented in the following.

1. **Advancements on the analysis of time series dynamics.**

   The study of dynamics has been widely applied for distinguishing different behaviors of time series' underlying phenomena [Rosso et al., 2007a]. Important assets with solid results concerning the distinction of time series dynamics are the ordinal patterns transformations and their derivations. In this context, the construction of the Causality Complexity-Entropy Plane (CCEP) [Martin et al., 2003, 2006; Rosso et al., 2007a], the probability distribution, and the transition graph are representations derived from the transformed set of ordinal patterns.

   However, to achieve significant results with these transformations, it is necessary to satisfy some requirements, such as defining the proper parameters and the large enough length of time series. These requirements may limit their applicability in specific challenging situations, such as those with short time series and where the parameters recommendations are not adequate.

   This contribution consists of the proposal of strategies to advance the applicability of ordinal patterns transformations for these challenging situations [Borges et al., 2019a], [Borges et al., Submitted 2021]. We proposed a novel metric computed from the ordinal patterns transition graphs, the probability of self-transitions. We show the feasibility of the applicability of this metric, advancing the hard task of distinguishing time series dynamics, even for scenarios with short time series, which is an impacting factor in this context. Furthermore, we also proposed a class separability index, which is a valuable method to estimate the most adequate parameters for the ordinal patterns transformations, which is intended to increase the applicability of the time series dynamics to a broader range of scenarios, such as of the time series classification problems.

2. **An enlightenment on the Collaborative Internet of Things.**

   The CoIoT initiatives consist of an essential data source for the Internet of Things. However, the literature neglected their proper characterization and clarification, which means that solutions based on their data were not adequately designed to handle their specificities correctly. This contribution aims to better understand the data generated by CoIoT sensors and its inherent problems. For this, we advance this subject by providing a definition and description for the CoIoT environments, which is helpful to highlight its main characteristics and properties.

With this contribution, we also provide results and analyses that help to improve the knowledge on the impact of such issues on the quality of sensed data [Borges Neto et al., 2015]. Thus, we provide a strategy for searching for a sensor in CoIoT, based on the query by content model, according to a given reference time series. By doing so, we highlight the need for strategies that are robust enough to mitigate the impact of the challenges on further development of solutions aiming to extracting knowledge from CoIoT sensors.

3. **Knowledge discovery on the Collaborative Internet of Things.**

   Solutions intended to discover some knowledge from CoIoT time series must consider the availability of a large amount of data and their inherent challenging aspects. This contribution consists of proposing strategies for knowledge discovery in the context of CoIoT, considering the characteristics and challenges present on their data. The main goal of this contribution is to advance the application of knowledge discovery strategies for the challenging scenarios of CoIoT [Borges et al., 2019b], [Borges et al., 2022].

   With this contribution, we advance the state-of-the-art in discovering knowledge from CoIoT data, especially regarding the time series classification problems. Our proposals are based on the dynamical behavior of the CoIoT time series, which are captured by the ordinal patterns transformations, Thus, according to the dynamical behavior of the CoIoT time series, we proposed a strategy for the classification of physical phenomena data collected by CoIoT sensors. By considering the dynamics from the time series underlying phenomena to perform its distinction, we advance the classification of time series from CoIoT sensors by considering their expected behavior, instead of the instances of its data points.

   Finally, another proposed solution related to this contribution is aimed to detect anomalies in CoIoT devices' behavior for the early detection of malicious attacks on them. By also considering the ordinal patterns transformations, we advance the state-of-the-art in detecting IoT botnet attacks by considering features obtained from the multiscale ordinal patterns transformations, which are extremely relevant for their distinction.

1.3 Thesis outline

The rest of this thesis is organized as follows. Chapter 2 presents an overview of Collaborative Internet of Things (CoIoT). This chapter provides our concept and assumptions on CoIoT, describing its potential and comparing it with its counterpart technologies. In addition, this chapter also discusses its main characteristics and challenges, which directly impact the usage and reliability of their data.

Chapter 3 provides an overview of the general concepts of time series analysis. This chapter presents the main concepts and definitions of time series and their dynamics. A brief analysis of some of the main time series transformations is also presented, focusing on the ordinal patterns transformations used to capture the dynamics of time series.

Chapter 4 presents a strategy for sensor searching within CoIoT scenarios. After characterizing the main properties of the sensed data in CoIoT we present our proposed strategy to search for a time series within the CoIoT scenario, based on the concept of query by content. This strategy, which explores collaborative filtering techniques, is aimed to deal with CoIoT data uncertainties and challenges.

Chapter 5 presents a method for the general problem of learning and distinguishing different time series dynamics. After discussing the main limitations and challenging aspects of current strategies, a novel metric is proposed: the probability of self-transitions extracted from ordinal patterns transition graphs. Based on this metric, in this chapter, we demonstrate that this strategy is suitable for scenarios with short time series and does not depend on selecting parameters.

Chapter 6 presents a strategy based on the analysis of time series dynamics to precisely identify time series data collected by CoIoT sensors. In this chapter, we first present a data characterization of the physical phenomena in CoIoT and propose a classification strategy to identify an appropriate CoIoT sensor correctly.

Chapter 7 presents the Time Series Classification via Class Separability (TSCLAS) strategy for the CoIoT data, which is based on the class separability analysis of the time series temporal dynamics. This strategy is based on two steps: the ordinal patterns transformations to capture the dynamics of time series, and the maximization of the proposed class separability measure to estimate the best parameters for such transformations. With the proposed TSCLAS strategy, we can classify time series from CoIoT based on their dynamics.

Chapter 8 presents a strategy for detecting botnets in IoT by the anomalies in their dynamical behavior. The strategy presented in this chapter is based on features obtained from the multiscale ordinal patterns transformations. The major contribution of this strategy is to propose an efficient solution to this critical challenge faced by the cyber-security community in recent years.

Finally, Chapter 9 presents our final remarks, summarizing the main contribution of this thesis, and provides future research directions.

Chapter 2

Collaborative Internet of Things

The Collaborative Internet of Things (CoIoT) can be understood as a special case of the IoT. However, although inheriting several characteristics from IoT, in this chapter, we show that there are specific aspects that can distinguish them from IoT. Section 2.1 presents some aspects of the nature of CoIoT sensors, and the factors that motivates their creation. Our concepts and definitions for CoIoT are presented in Section 2.2, by considering its main characteristics and particularities. A comparison between CoIoT and its counterpart, the Industrial Internet of Things (IIoT), is presented in Section 2.3. It is also discussed, in Section 2.4, the openness aspect of CoIoT and its potential for novel applications. Then, it is also highlighted the challenges and consequences when dealing with CoIoT sensors, in Section 2.5, which adds a number of particularities which directly impact on their usage and reliability. In Section 2.6 our final remarks are presented.

2.1 Introduction

A fundamental aspect of IoT is its ability of interacting with the environment. This makes it an important asset towards the convergence between physical and informational worlds, one of the pillars for the envisioned future from ubiquitous computing [Weiser, 1991]. In practice, IoT sensors have been the "eyes" and "ears" for the forthcoming systems that need to interact with some natural phenomena. From home automation [Coronado and Iglesias, 2016] and agricultural applications [Wu et al., 2014] to large smart cities [Zanella et al., 2014] and global weather forecasting [Greengard, 2014], the range of phenomena being monitored by IoT is impressive, and is still growing. The CoIoT has been one of the main enablers for the creation and deployment of IoT sensors, and a source of all the data for the current IoT.

The growing number of CoIoT initiatives is mainly driven by recent efforts to democratize the monitoring of environmental conditions, also known as citizen science [Austen, 2015]. These efforts combine the reducing cost of sensors with the increasing accessibility to the knowledge for better using this sort of data. These factors are enabling common users to actively contribute to the current IoT and have the potential to be a powerful asset for monitoring a diversity of phenomena, in a larger scale than industrial or governmental initiatives.

However, due to their cheap components and the lack of precautions, or even knowledge, regarding the right placement and manipulation of the sensors, they are more prone to errors, imprecisions and inaccuracies in their readings than more reliable (e.g., industrial [Chen et al., 2016]) sensors. The quality of data generated by CoIoT sensors has been the focus of scientific studies to testify its accuracy [Gao et al., 2015]. But its use is still a concern by researchers, scientists, and specialists. Most of them are still cautious regarding their reliability and the quality of data they produce, and decided not to use it until their quality improves [Austen, 2015].

In addition to that, because of the small size of sensors, most of the available data in current IoT is available outside the devices, in platforms designed to store and persist their data for later processing [Rawassizadeh and Kotz, 2017]. This separation between devices and data is good for technical reasons (e.g., availability, centrality), but it lacks in providing contextual information regarding the origin of the data. In those platforms, which is the case of ThingSpeak[1], the data and resources usually are not described or, when they are, have a very poor description. Even if the descriptions were given, since there is no validation, there is no guarantee that they are correct. Generally, they are described by a simple set of tags and textual information, freely assigned by their users, which makes the search for some keyword or expression prone to errors and misunderstandings [Borges Neto et al., 2015].

Table 2.1 summarizes the lack of descriptions for the collaborative sensors available in the ThingSpeak platform, collected at three distinct periods: October 31, 2015, and July 28, 2016, and April 14, 2017. As noted, the number of sensors increased approximately 90 % at each period, in relation to its predecessor. It improved from 4,064 to 7,427 available sensors in the first period, and to 13,013 in the second period. Some sensors are capable of monitoring more than one phenomenon, resulting in a different number of available time series. In turn, each time series represents a single monitored phenomenon. For this case, in 2015 we had a number of 15,170 time series ($\approx$ 3.73 per sensor), in 2016 a number of 23,660 time series ($\approx$ 3.18 per sensor), and

[1]ThingSpeak open data platform for the Internet of Things - `http://thingspeak.com`

in 2017 a number of 51,185 time series (≈ 3.93 per sensor). The analyzed descriptions was the *textual description*, the *geographical location* (coordinates) in which the sensor are placed, and the *tags*, which are keywords that express some semantic information regarding the sensor resources, the type of data, and monitoring phenomena.

Table 2.1. Summary of problems in the descriptions of CoIoT sensors in the ThingSpeak platform, for the periods of October 31, 2015, and July 28, 2016, and April 14, 2017.

Issue	**2015**	**2016**	**2017**
Number of sensors	4,064	7,427	13,013
Time series per sensor[2] (t/s)	15,170 (3.73 t/s)	28,966 (3.90 t/s)	51,185 (3.93 t/s)
No textual description	1,655 (40.72 %)	3,681 (49.56 %)	6,632 (50.96 %)
No location	3,208 (78.94 %)	6,046 (81.41 %)	10,856 (83.42 %)
No tags	3,233 (79.55 %)	6,179 (83.20 %)	11,045 (84.88 %)
No description and tags	1,564 (38.48 %)	3,499 (47.11 %)	6,318 (48.55 %)

By this analysis, it is clear that the number of available sensors has increased throughout the years, but also the problems. This reinforces the need for a better comprehension of CoIoT and their characteristics, which starts by its proper definition.

2.2 Concepts and assumptions

In order to better develop a proper definition for the CoIoT, we must first discuss its generalization, the Internet of Things. The IoT term has been widely used in recent years, sometimes as a buzzword by marketing industry and sometimes as an umbrella for various correlated topics [Atzori et al., 2010]. A comprehensive tracking for most of the IoT definitions, available from both technical and academic literature, has been made by the Postscapes team[3]. Figure 2.1 illustrates a cloud of tags representation for the most frequently used words in 65 definitions tracked by Postscapes. The top 20 more relevant words, starting by the most common word, are: internet, things, objects, networks, information, physical, world, communication, services, devices, protocols, virtual, connected, data, embedded, interact, people, smart, sensors, based.

These more relevant words reinforce the statement from Atzori et al. [2010] that the concept of IoT can be viewed from different perspectives: (i) Internet-oriented, (ii) Things-oriented, and (iii) Semantic-oriented.

Both Internet-oriented and Things-oriented visions brings to the scene its enabling technologies [Al-Fuqaha et al., 2015]. The first one follows a more technical sense, such

[2]Each sensor may generate one or more time series. t/s: time series per sensor.

[3]Postscapes' IoT Overview Handbook: `https://www.postscapes.com/iot/#definition`

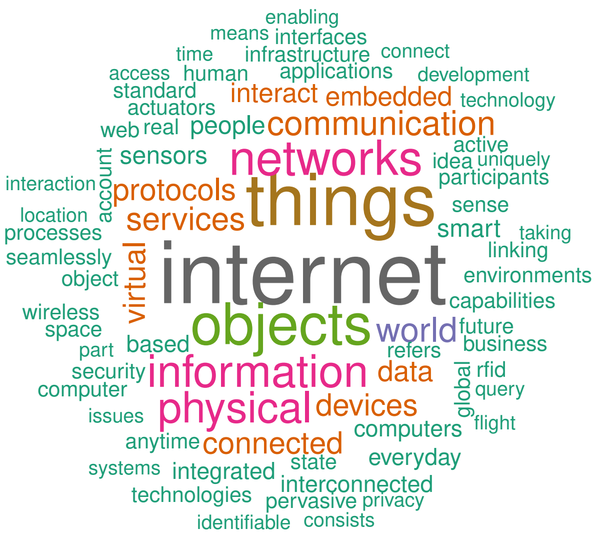


Figure 2.1. Representation as cloud of tags of the most frequently used words from the 65 definitions tracked by the Postscapes team.

as the sum of the technologies and protocols to create the infrastructure to interconnect the many available devices [Vasseur and Dunkels, 2010]. In this view, the Internet protocols plays a central role as main enablers for the communication capabilities to the objects, allowing their communication with all Internet compatible elements.

From simple Radio-Frequency IDentification (RFID) components to more complex technologies, the Things-oriented vision refers to the technologies for the creation of the so-called smart objects [Loureiro et al., 2012]. Originated from the concepts of wireless sensor networks [Akyildiz et al., 2002; Nakamura et al., 2007], these are devices with sensing and actuating capabilities, that are embedded in the physical world to perceive and control its state.

The Semantic-oriented vision is directly related to the data generated by the IoT sensors and the potential knowledge that systems could extract from them. This vision is closer to the envisioned ubiquitous computing [Weiser, 1991], where the interaction between systems and people in the computing aided environments will open many opportunities for new applications and services, making easier their everyday activities [Gubbi et al., 2013].

In this thesis it is assumed an approach closer to the Semantic-oriented vision of IoT. By focusing on the reliability aspects of the services provided within the IoT environments, we are automatically guided to follow a more data-driven approach, where all the services must rely on. Thus, the concept for IoT assumed in this thesis is presented in Definition 2.2.1. This concept is aligned with the most relevant terms presented in Figure 2.1, and is a direct adaptation from the works of Abu-Elkheir et al. [2013] and Qin et al. [2016].

Definition 2.2.1 (Internet of Things)**.** The Internet of Things is the network of computing enabled objects, or things, that allows the interaction between informational and physical worlds, by exchanging data via sensing services and performing control actions through actuating services, following the Internet protocols.

Once the concept for IoT is established, it is easier to define CoIoT. The collaborative aspect of IoT comes from the possibility of interaction between the involved parts aiming to provide enhanced services in IoT environments [Behmann and Wu, 2015]. From a data-driven perspective, the collaborative data gathering will lead to a significant increase in the amount of available data [Borges Neto et al., 2015], and the integration between them enables the creation of services that are more adequate to handle complex requirements [Chen et al., 2016] and more robust to deal with challenges in data [Borges Neto et al., 2015; Belkacem et al., 2017; Montori et al., 2018a]. With the collaborative initiatives it will not have isolated IoT solutions, instead, it is possible to create services by the composition of several sources of data, even without owning dedicated sensors [Montori et al., 2018a].

In practice, CoIoT initiatives follow a near crowdsensing configuration [Borges Neto et al., 2015; Montori et al., 2018a]. The main difference between the collaborative IoT and crowdsensing initiatives (e.g., participatory sensing) is the active role users play in the last case [Burke et al., 2006; Silva et al., 2014]. For instance, while users in participatory sensing actively participate in the process of sensing by contributing with their personal experiences, in CoIoT, users are responsible to configure, deploy and maintain their owned sensors, and the sensing process is performed by the devices [Borges Neto et al., 2015].

In CoIoT, users collaboratively choose to make their data open and publicly available through Internet. These initiatives are mainly composed by Do-It-Yourself (DIY) sensors, which may vary from simple sensors collecting weather conditions from a residence (e.g., temperature, humidity) to more complex environmental monitoring stations (e.g., air pollution, radiation levels). But, since CoIoT devices may be constructed from cheap components, they are more prone to errors of precision and accuracy in their readings than another professional ones. Also, due to its collaborative characteristics, users provide no guarantees regarding the responsiveness nor availability of their sensors. Thus, in this thesis, the assumed concept for CoIoT is presented in Definition 2.2.2.

Definition 2.2.2 (Collaborative Internet of Things)**.** The Collaborative Internet of Things is a specialized case of the Internet of Things, composed by those computing enabled objects in which their sensing and actuating services are collaboratively defined as open access and freely available.

Although simple, this definition for CoIoT is wide enough to represent the type of sensors we are dealing within the collaborative space of IoT. It covers both sensing and actuating services, but the vast majority are the sensing ones, since it is easier to make their data open than enabling the control of some environment available. Also, given the intention from sensor owners to collaborate, we assume the CoIoT consists of services that are open and freely available, discarding commercial initiatives. In fact, this question of openness and commercial initiatives deserves further discussion, which is given in next sessions.

2.3 Collaborative IoT vs. Industrial IoT

Around the world, IoT initiatives are appearing every day, by both industry and community. These two sides of IoT comprises two main competing aspects of users requirements. One is the innovation and profit of companies that want exclusive products and advantages to their competitors, provided by industry. The other is the effort for a common good, where citizens, for example, are the main interested parties in the process, that want to benefit from the whole IoT community potential.

Following the predictions of billions of dollars that this market will pay off in next years, industries have been investing a considerable amount of resources to be part of IoT. Big companies, such as Amazon, AT&T, IBM, Intel, Microsoft, Samsung, among others, have their own IoT divisions. Solutions have been made for several IoT layers, from the development of sophisticated miniaturized devices until IoT cloud platforms, specifically designed to store and process the data originated from IoT devices.

This side of IoT that is often called Industrial Internet of Things (IIoT) [Perry, 2016; Chen et al., 2016], or even commercial IoT, is in charge to develop reliable and secure IoT solutions. IIoT is strongly based on the quality of the data collected and, also, on adding value to their consumers products and services.

For instance, industrial IoT solutions are ranging from transportation systems for monitoring and tracking products throughout the world, improving the logistic and management of the whole transportation process, until energy consumption monitoring systems, enabling the control of the overall power demands of a building and reducing the power consumption through conservation and energy alternative generation methods. For these cases, the use of IoT products and solutions is governed by a set of service level agreements between the customers and companies, in order to the main-

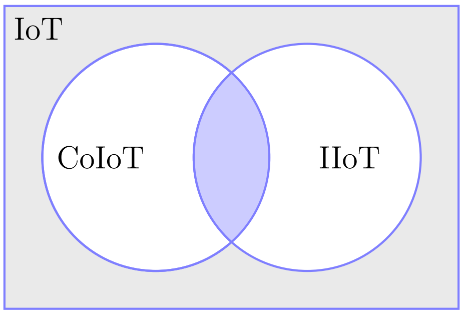


Figure 2.2. Representation as Venn diagram of the Internet of Things space, and their constituents subspaces of CoIoT and IIoT. The highlighted intersection between CoIoT and IIoT consists of the part of IoT sensors that are maintained by public organizations, research institutes, universities, laboratories, among others.

tenance of the quality of the services. Both the data generated by the IoT devices and the decisions taken based on them are also under this agreement. In this case, customers' data do not need to be public, and they have the necessary guarantees that both quality and privacy will be concerned by their IoT services.

In contrast to the industrial IoT, in the current IoT scenario, this level of agreements and quality is not a reality for all available sensors and data [Borges Neto et al., 2015]. However, there is an intersection between CoIoT and IIoT worlds that provides an interesting scenario for exploration, mainly by researchers and innovators. Figure 2.2 illustrates the subspace of CoIoT and IIoT scenarios, as parts of the IoT space. The highlighted intersection between these two subspaces consists of the IoT sensors that are maintained by public organizations, research institutes, universities, laboratories, among others. By having these specialized support behind, these organizations are able to both have some level of guarantees on the quality of their data, while making them freely available for anyone who are interested. The grey area in the figure, outside CoIoT and IIoT, can be understood as the unknown IoT devices spectrum, which are not visible nor accessible, i.e., their data are not publicly available.

As described by Borges Neto et al. [2015]; Borges et al. [2019b] and Montori et al. [2018b], IoT data is very challenging to handle. Besides the large amount and heterogeneity of sensors, issues such as missing readings, different rates, among others, make IoT a very unreliable scenario. Thus, in order to properly evaluate any proposed solution for IoT, a reasonable strategy for researchers and innovators is to choose data from the sensors within this intersection. For instance, an example of these reliable public data come from automated airport weather sensors, the Automated Surface Observing Systems (ASOS). These are reliable sensors that generate observations every minute, or every hour, according to the airport, and are used to support weather forecast activities and aviation operations[4]. Their reliability comes from constant monitoring of data quality, 24 hours per day, with maintenance as soon as a problem is detected.

From the whole IoT spectrum, this is the kind of sensors where their maintainers can provide more reliable data, mainly with respect to their precision, probability of correctness, and trustworthiness [Borges Neto et al., 2015; Buchholz et al., 2003]. Besides that, they are still able to make their data freely accessible. This reinforces the importance of the openness aspect of CoIoT, which is discussed in next section.

2.4 The openness of Collaborative IoT

To understand this growth of collaborative initiatives in IoT, we have to look back a few years, and take into account, as precedent, the expansion of the Internet in its early days. Motivated by the fundamental spirit of freedom and collaboration, the Internet grew as an open environment full of knowledge and opportunities [Cerf, 2012; Cerf and Quaynor, 2014]. Based on the free access to the information and technologies that comprise it, the users of the early Internet could easily become developers by simply studying the technologies behind it. Also, due to the simplicity of its development process, many users was allowed to create several new kinds of applications, from discussion boards to commercial systems based on new businesses models.

The case of the collaborative IoT has the potential to be similar to what occurred to the Internet. The constant reducing costs of embedded computing devices and the increasing accessibility and simplicity to develop for them are enabling common users, with just a few backgrounds in computing, to build their own sensors. Moreover, the increasing number of cloud and IoT platforms dedicated to store, process, and distribute these sensors' data, is enabling the creation of environments full of information with the potential to foster an unprecedented kind of new applications and services.

However, as well as what happens to the Internet, when it became more financially viable, the interest of private companies to commercially explore this new market increased. These private initiatives are essential to the rapid development of the IoT scenario, pushing it towards the standardization of protocols, platforms and tools. For instance, the lack of standards in the IoT world is one of the biggest challenges faced by the developers. But, on the other hand, it is also bringing limitations and closures to IoT. This is creating a scenario where, instead of having a single global IoT, we would have independent private IoT environments, where each user is able to only access their own devices and data, in silos [Ahmed et al., 2017].

[4]Automated Surface Observing Systems from the U.S. National Weather Service: `https://www.weather.gov/asos/asostech`.

Although some applications and systems need this kind of private IoT (e.g., where information confidentiality is a requirement), most users need to benefit from the collective IoT intelligence, and this can be achieved by the collaborative IoT efforts. If we lose the openness of data and, thus, the system's integration desired for a proper ubiquitous scenario, these limitations may affect the ubiquity level of applications. Since the more integration of systems the more data will be available, then, more context will be available to the smart applications. Thus, the openness of data is essential to enable the whole potential for creating the most powerful services of IoT.

Taking as example the smart cities scenarios, since cities are made by people, the idea of open data is much more meaningful. Most of their services have the citizens as directly affected by its benefits, whether these services are provided by the government or public companies. Indeed, some City Halls had also emerged with projects to increase their administration transparency by making their data publicly available[5]. But also, whether motivated by the spirit of collaboration and the sympathy with the open data initiatives[6] or simply by the benefits the users can have by making their data open, the collaborative IoT can be responsible for a valuable amount of data, that could be essential to the development of many of the envisioned smart services.

With respect to the open data from CoIoT, most of them are due to common users, which are responsible for the deployment of their own sensors and the delivery of the collected data. By opting for open distribution, they make their data publicly available through the Internet in a collaborative way. However, to make it useful for any application it is necessary to consider some specific characteristics within this scenario. In next section we discuss some challenges related to the CoIoT sensors and their main characteristics, following a data-centric viewpoint.

2.5 Challenges for the Collaborative IoT

As a special case of IoT, the CoIoT inherits all the challenging characteristics present in those scenarios. Furthermore, in addition to them, there are specific characteristics of CoIoT that require special attention. Some of them are related to the collaborative aspect from the sensors and others to their intended use. In the following, we present the main issues and, consequently, challenges related to the CoIoT sensors.

[5]Open data from Rio de Janeiro's City Hall: `http://data.rio`

[6]The Open Data Institute: `http://theodi.org`

2.5.1 Inherited issues from IoT

As discussed by Borgia [2014] and Al-Fuqaha et al. [2015], since IoT is the result of many technologies, it is expected that the challenges from each of them are also inherited. Thus, CoIoT also inherits the challenges from ioT. However, since there are several works that survey the IoT challenges in details, let us briefly discuss some of the most impactful ones that must be addressed for any IoT-like solution.

The high heterogeneity of devices, technologies, services, and environments, is an aspect that must also be considered for any solution based on IoT devices [Gubbi et al., 2013; Stankovic, 2014]. In the same way, the need for handling the impressive number of devices and data generated brings new problems of scalability and big data analytics to the scene [Stankovic, 2014]. Aspects related to the connectivity of these devices, such as addressing, routing, mobility, and management, are also important to be handled by a robust IoT application [Gubbi et al., 2013].

Another important characteristic of IoT is related to the possible limitation of hardware resources from their devices. Although not considered in Definition 2.2.1 of IoT, a common aspect for IoT sensors is the fact they constitute embedded systems composed of resource-constrained devices. These limitations may affect the computation, memory, communication, storage capabilities of these devices, which are, generally, powered by batteries [Guinard et al., 2010; Zeng et al., 2011]. The reason for not considering these restrictions in our definition is because this is not the case for all IoT sensors. For instance, due to the constant evolution of embedded technologies, which are enabling the creation of more powerful devices for IoT, restrictions are less important for these systems. However, because of its impact, this is an issue that must still be of concern for IoT based solutions. Indeed, IoT devices may not support complex algorithms and structures because of such limitations [Al-Garadi et al., 2020].

A critical requirement necessary to the success of this whole IoT world is the security [Miorandi et al., 2012]. Questions related to the authentication and authorization of devices, the integrity and protection of data, and privacy of users and environments, are examples of such concerns [Sicari et al., 2014]. Furthermore, because of their computational restrictions and misconfigurations, the IoT devices are an easy target for several types of attacks, which make their security an urgent concern [Bertino and Islam, 2017; Kolias et al., 2017; Blaise et al., 2020]. For instance, a recurrent attack involving IoT devices in recent years, which serves as basis for other attacks, is their infection by botnets [Bertino and Islam, 2017]. Once a device is compromised, it can be used for large scale orchestrated attacks, such as spam delivery and Distributed Denial-of-Service (DDoS) attacks [Bertino and Islam, 2017; Kolias et al., 2017]. This is a concerning aspect for the security of IoT, because an attack from the large number of devices in IoT is very hard to be handled by any system.

2.5.2 Absence of proper descriptions

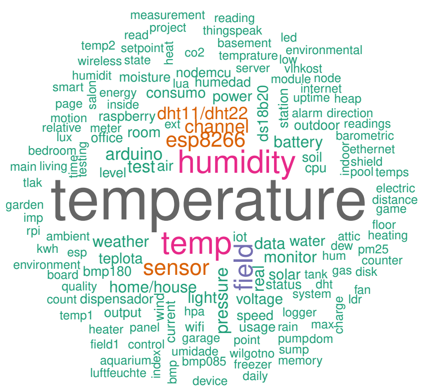
A fundamental task when looking for a specific sensor in the CoIoT environments is the search for sensors based on their descriptions and select a desired one from the resulting list [Perera et al., 2014b; Ahmed et al., 2016]. Some IoT platforms allow the searching for a particular sensor based on its meta-data informed by their owners (e.g., tags, location, description). As examples, we have the OpenSensors[7], ThingSpeak, and Xively[8] platforms. However, since owners by themselves are responsible for the definition of these meta-data descriptions, generally as freely text-based, they may not correctly correspond to the sensor services or even be empty.

To better illustrate this problem, in Table 2.1 we have summarized the lack of descriptions of collaborative sensors available in the ThingSpeak platform, collected at three distinct periods from 2015 to 2017. For this case, and following our assumptions for CoIoT in Section 2.2, we considered those sensors which were set as public by their owners. The number of collaborative sensors increased more than 3 times in this period. This demonstrates a clear increase in the popularity of CoIoT, which reinforces the expected tendency.

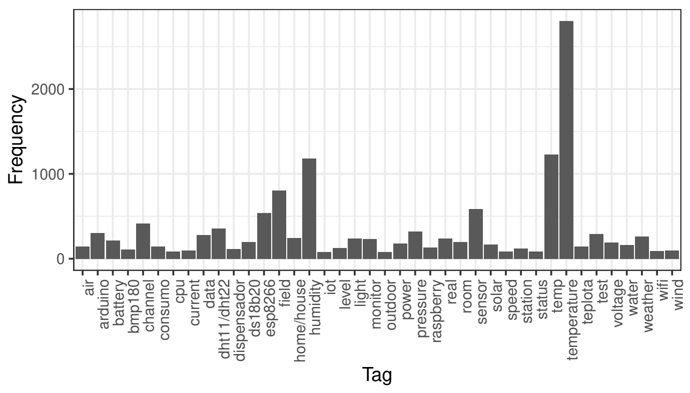
The analyzed descriptions was the *textual description*, the *geographical location* (coordinates) in which the sensor is placed, and the *tags*, which are keywords that give some meaning regarding the sensor resources, the type of data, and monitoring phenomena. The problem of lack of descriptions about sensors resources, that is growing with the increasing number of available sensors, is a challenging aspect impacting their useful utilization. As a consequence, when searching for sensors based on a set of parameters, some of them will not be reached, remaining occult to be used.

Another more subtle problem is that even if the descriptions were given, since there is no validation for them, there is no guarantee that they are correct and representative. For instance, when searching for a keyword like *temperature* it is possible to find sensors monitoring the environment temperature, or from the interior of an apartment, the temperature from the soil in agricultural applications, or even the temperature of a computer processor. To infer which of these sensors is the correct to us is a problem.

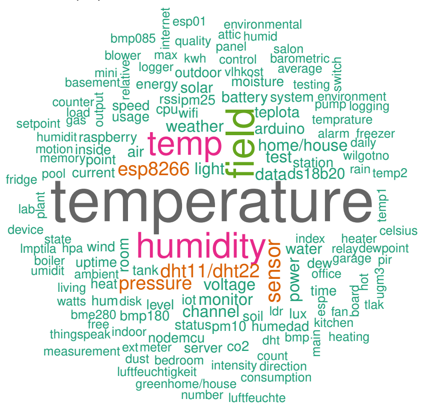
[7]OpenSensors IoT open data community - `http://opensensors.io`
[8]Xively IoT platform for connected devices - `http://xively.com`



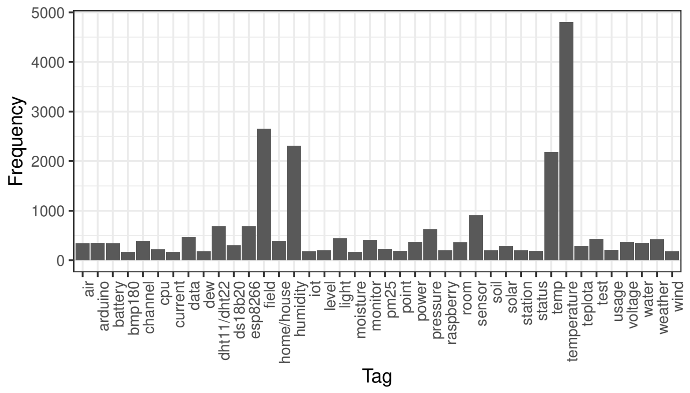
(a) Cloud of tags for 2015



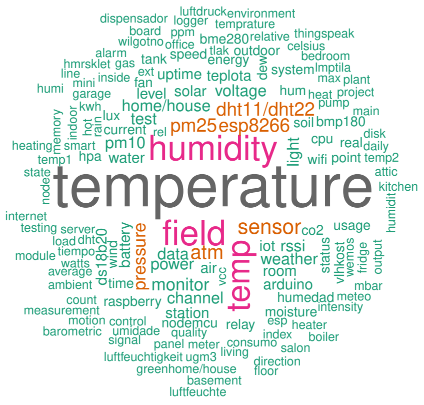
(b) Histogram for 2015



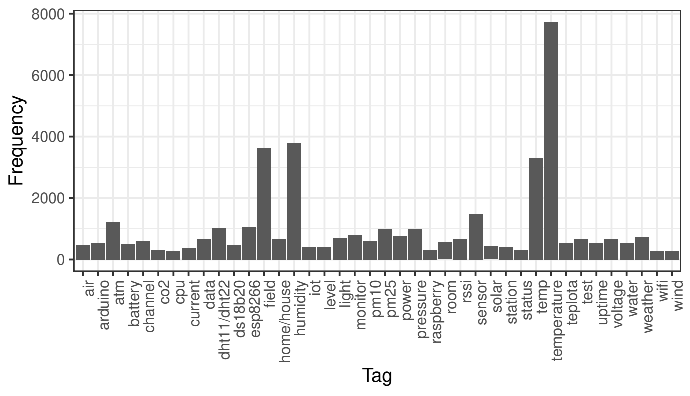
(c) Cloud of tags for 2016



(d) Histogram for 2016



(e) Cloud of tags for 2017



(f) Histogram for 2017

Figure 2.3. Cloud of tags and histograms of the most frequent words used by owners to describe their CoIoT sensors, their resources and operations. These data was obtained from descriptions of CoIoT sensors available in the ThingSpeak platform, collected at periods of October 2015, July 2016, and April 2017.

Figure 2.3 illustrates some words that owners often used to describe their devices, based on the sensors that had some description. It can be noticed that the majority of descriptions are related to the environmental phenomena, as well as sensors technical descriptions and properties. In fact, these descriptions are really poor, if compared to more complex ones such as semantic and ontology based [Barnaghi et al., 2013; Perera et al., 2014b; Qin et al., 2015]. For instance, the top-5 most frequent words for the 3 years are: temperature, humidity, temp, field, and sensor. These last 3 are generic descriptors, which add no valuable information to the monitored phenomena. This leads to the fact that, even when the descriptions of CoIoT sensors were provided, it may not be representative enough to correctly describe them. Thus, searching for a specific sensor in the CoIoT scenario by its descriptions may not retrieve the most appropriate results.

2.5.3 Availability of CoIoT sensors

Since there are no guarantees regarding any kind of quality criteria for the CoIoT sensors, an important aspect that arises when dealing with them is their availability. Those sensors may suddenly stop their activity, whether by sensor failures or being turned off by their owners, or even alternate between periods of inactivity, which is a reality in this collaborative scenario.

For this reason, the presence of gaps in the time series from the CoIoT sensors can not be treated as an exception, as occurs to many data based systems. There are a considerable number of studies proposing methods to deal with missing data, such as interpolation, imputation, and other strategies to the completion of the fault samples [Schafer and Graham, 2002]. However, given the large amount of data in CoIoT, the insertion of more data to complete gaps may not be the most efficient nor effective solution. For instance, the more data we have to deal the more impact in scalability problems we will have, increasing the complexity of problems in both time and space. Furthermore, it is harder to perform data imputation as the gaps increase, reducing the effectiveness of the data completion.

Another issue, also regarding the missing data, arises when it is necessary to perform some pairwise processing with the time series. If a time series presents gaps for a given time interval, there will not be a correspondent for that data in another time series to compare. Also, a factor impacting this pairwise time series processing is the differences between their lengths, due to different update intervals. We can easily find sensors updating data hourly, and others at seconds. Another cause for this problem is related to the communication of these cheap sensors, now via Internet, and the network problems may also impact on the differences in the moments of data arriving at the IoT platforms.

As a consequence of the problem of unevenly spaced time series, that is not always covered by most of the time series processing strategies and algorithms, is the trade-off between discarding data points or creating new ones. If the data points with no correspondent samples in a same time interval is discarded, it may cause a loss of some important information from the time series, that could be essential to their correct analysis. Or, if it is decided to complete these missing points, they may lead to inefficient computation, as previously discussed.

2.5.4 Reliability of CoIoT sensed data

Although these collaborative initiatives are real and tend to grow, scientists and specialists are still cautious regarding the reliability of these sensors and the quality of data they produce [Austen, 2015]. In fact, every sensing of a real world phenomenon is subject to some imprecision, since we only get an abstraction of it, but the industrial and scientific communities are based on high precision sensors which aims to reduce these imperfections to the possible minimum. However, due to the high cost of these industrial sensors, the CoIoT initiatives are mostly based on the use of cheap components and alternative ways to measure a given phenomenon, generally adding noise to the readings. In most of the CoIoT sensors, we are dealing with virtual sensors [Loureiro et al., 2012], which are sensors able to measure magnitudes that are very correlated to the desired one and can be used to infer the values of the desired phenomena.

For instance, the BAM-1020 Beta Attenuation Monitor[9] is a professional sensor certified by the U.S. Environmental Protection Agency to account for particle matters (PM) in the air, based on the principle of beta ray attenuation, which produces already known reliable data. On the other hand, the DustDuino[10] is a DIY initiative to build a low-cost air quality monitoring sensor to account for the same PM metric, but based on the more simple principle of light scattering when particles obstruct an infrared LED signal. Although the quality of data generated by this method was the focus of scientific studies to testify its accuracy [Gao et al., 2015], some researchers still decided not to use it until their quality could be improved [Austen, 2015].

Several examples of DIY sensors are easy to find, such as homemade anemometers

[9]BAM-1020 Continuous Particulate Monitor: `http://www.metone.com/docs/bam1020_datasheet.pdf`

[10]DustDuino particle concentrations monitor: `https://publiclab.org/wiki/dustduino`

to measure wind speed[11] and solar radiation level meters[12]. However, these less reliable sensor components may present more problems, such as noise and outliers, than the industrial ones. In addition, problems may also be caused by the wrong operations from their owners, such as misplacement of the sensors in a proper location, to reduce the impact of external interferences. Thus, regarding the reliability of the CoIoT sensors, it is necessary to assume that errors and uncertainties are common aspects affecting their correct operation and measurements. Furthermore, solutions toward minimizing these problems in the data must be considered before properly using them.

2.6 Conclusions

This chapter presented our definition for the Collaborative Internet of Things, discussing its characteristics and what makes it a particular case of the IoT. In the same way, a discussion about the similarities and differences between CoIoT and IIoT is presented. It is also given an insight on the potential of using the intersection among these two scenarios for precise studies and researches on CoIoT. A discussion about the need for an open IoT, as its potential of applications, is also presented. Finally, the challenges that must be considered when dealing with CoIoT-based solutions is presented, from the ones inherited from IoT to the specificities of this new collaborative scenario.

[11]Example of a DIY anemometer: `https://thingspeak.com/channels/10021`
[12]Example of a DIY solar radiation monitor: `https://thingspeak.com/channels/9892`

Chapter 3

Time series analysis: General concepts and transformations

In this chapter, we delve into the fundamental aspects of time series and the dynamics of their underlying phenomena. After a brief motivation, in Section 3.1, a conceptualization of time series analysis and their dynamics is presented in Section 3.2. Concepts and definitions related to time series from CoIoT are presented in Section 3.3. A brief analysis on some of the main time series transformations is given in Section 3.4, illustrating which aspect of the data can be highlighted by each of them. Thus, we detail, in Section 3.5, the ordinal patterns transformation, which is our chosen transformation for capturing the dynamics of time series. In Section 3.6, we discuss the interpretation of the ordinal patterns transformation, that will serve as the basis for further solutions to the identified problems in CoIoT. Our final remarks are presented in Section 3.7.

3.1 Introduction

The range of phenomena being monitored by CoIoT is impressive, resulting in a large amount of highly heterogeneous data. Among this diversity, a common shared aspect is the notion of time related to each of their data samples. Thus, a reasonable way of dealing with the CoIoT data is by modeling them as time series. Time series is one of the most ubiquitous modeling technique in data analysis. Many real world problems, from different domains, have solutions based on the analysis of time series. Astronomy, biology, climate, environment, finances, and medicine, although very distinct and with different objectives, are a few examples of areas that benefit from the analysis of time series [Aghabozorgi et al., 2015].

With time series representation, it is possible to model the evolution of a phenomenon in time. The way the data are constructed by events ordered in time, enables strategies for discovering important time related information, such as predictions of future trends and historical analysis of data behavior. Another important approach when dealing with time series consists of strategies aiming to discover knowledge from the data. This is related to the application of data mining strategies for time series, such as indexing, clustering, classification, anomaly detection, among others [Keogh and Kasetty, 2003; Lin et al., 2002, 2003; Bettaiah and Ranganath, 2014].

However, most of the solutions in literature for time series data mining make assumptions regarding time series regularities, that are not valid for real world scenarios [Hu et al., 2013]. As described in Section 2.5, there are some challenges one need to consider when dealing with the real CoIoT data, such as the massive number of sensors, poor descriptions, differences in magnitude and resolutions, problems in data as noise, imprecision, and missing data points. Given this non-exhaustive list of problems, it is clear that solutions to handle these data should consider a method that are precise in representing their characteristics, but also robust to their issues [Borges et al., 2019b].

In this thesis, our strategy for dealing with the issues present in CoIoT data is based on the transformation of the time series onto another domain distinct of time. This transformation is a step prior to any data mining technique analysis, so it can mitigate the impact of those challenges and reduce their impact on the discovered knowledge. Our strategies are based on the ordinal patterns transformation, a cornerstone contribution from Bandt and Pompe [2002]. This transformation, which is detailed in Section 3.5, consists of constructing a set of symbolic ordinal patterns from time series data, which can be used for distinguishing between different dynamics [Rosso et al., 2007a].

By following this strategy, we are able to learn the intrinsic dynamics of the time series underlying phenomena, and are not dependent on a reference sensor, unlike the previous solution for searching a time series in CoIoT [Borges Neto et al., 2015]. Before presenting more details about the ordinal patterns transformations, let us first conceptualize time series and their dynamics. After that, we are also able to define the time series from CoIoT and the representations used in this thesis.

3.2 Time series analysis

The notation of time series for representing time ordered data is a consistent format that arises on several knowledge domains, belonging to a select group of always relevant research topics. Throughout decades, an expressive number of studies involving time series data mining were proposed, such as characterization, clustering, classification, among others [Keogh and Kasetty, 2003; Lin et al., 2002, 2003; Bettaiah and Ranganath, 2014]. However, it is still possible to eventually find a novel viewpoint, or that was not properly covered yet, with the potential to become a feasible approach for the time series analysis community. In this thesis, we evaluate the feasibility of temporal dynamics for discovering knowledge when analyzing time series data.

3.2.1 Time series concepts

Before presenting more details on time series dynamics, let us establish the concepts of time series used throughout this work. We follow the general concept of time series expressed in Definition 3.2.1, which is based on the notation of Bagnall et al. [2017].

Definition 3.2.1 (Time series)**.** A time series $\mathbf{x} = (x_1, x_2, \ldots, x_m)$ is a sequence of m data points, with $m = |\mathbf{x}|$, indexed by time in increasingly order. Each element x_i represents the i-th data point, where $i \in \mathbb{N}$ maps this point to a given time stamp.

Eventually, a collection of time series is aggregated as a dataset for further processing and analysis, which is formalized by Definition 3.2.2.

Definition 3.2.2 (Time series dataset)**.** A time series dataset $\mathbf{D} = \{\mathbf{x}_i\}_{i=1}^{n}$ is a collection of n time series, where each i-th time series having length m.

For some learning strategies, the time series from a given dataset must have a label, used from training and testing phases. Definition 3.2.3 presents the concept for labeled time series datasets. More details on learning strategies is given in Section 3.2.3.

Definition 3.2.3 (Labeled time series dataset)**.** A labeled time series dataset $\mathbf{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ is a collection of n pairs of time series and labels, where each i-th time series having length m and a specific label $y_i \in \mathcal{Y}$, from a given set of c classes $\mathcal{Y} = \{y_1, y_2, \ldots, y_c\}$.

With these definitions, we are able to follow our discussion on time series dynamics and their importance for time series analysis.

3.2.2 Time series dynamics

The study of dynamics consists of the understanding of systems that evolve in time [Strogatz, 2018]. Accordingly to Birkhoff [1927], these are systems which have the property that, for a given set of n real variables $x_1, x_2, \ldots, x_n$, its state at a specific time t can be completely expressed by those variables. There are two main types of dynamical systems: differential equations, which are used to describe their evolution in continuous time, and iterated maps (also known as difference equations), which describe systems evolving in discrete time [Strogatz, 2018].

With respect to the analysis of time series dynamics, they are governed by discrete time evolutions, so we are interested in the study of iterated maps as the type of dynamical systems. Consequently, when analyzing these time series dynamics, we are interested in understanding its behavior when changing from a previous state to the next one [Birkhoff, 1927]. By understanding this behavior, it is possible to extract information about the dynamics of the time series underlying process, and use this information for distinguishing their dynamics, such as deterministic, stochastic, and chaotic behaviors [Rosso et al., 2007a; Zunino et al., 2012; Ravetti et al., 2014].

Although very useful, the distinction of time series by their dynamics is not an easy task. For instance, an intriguing aspect of time series arising from chaotic systems is that they share several properties with stochastic processes, making their distinction a hard task [Rosso et al., 2015, 2013; Ribeiro et al., 2017; Rosso et al., 2007a; Zunino et al., 2012; Ravetti et al., 2014; Kulp and Zunino, 2014; Ye et al., 2017].

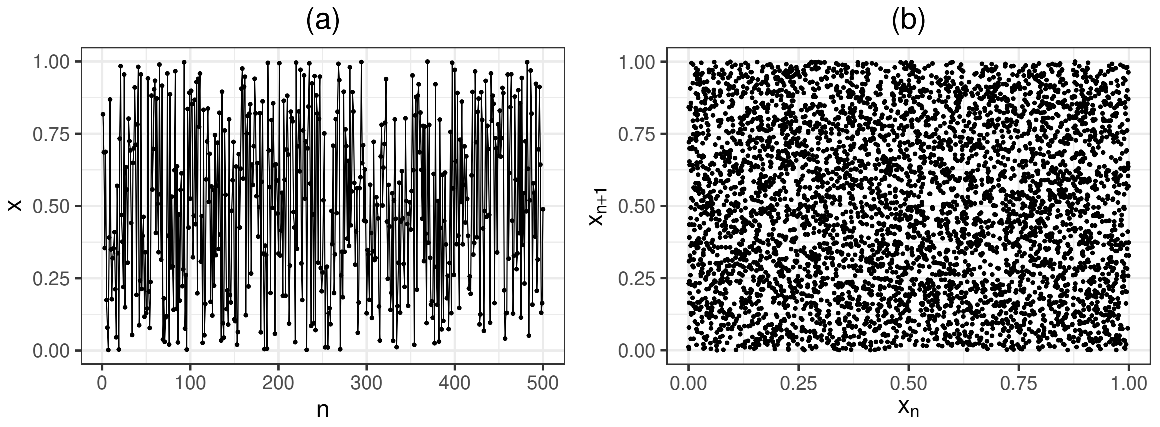


Figure 3.1. Illustration of (a) the first 500 data points of a time series generated by a random uniform distribution $\mathcal{U}(0, 1)$ on the interval $[0, 1]$ and (b) a scatter plot of the whole time series of length $m = 5000$, consisting of the relation between their consecutive points (x_n, x_{n+1}).

Figures 3.1a and 3.2a illustrate the first 500 data points of time series generated by a random uniform distribution $\mathcal{U}(0, 1)$ on the interval $[0, 1]$ and by a logistic chaotic map, with $\rho = 4$ and $x_0 = 0.1$, respectively. Although these time series look similar, which may be confused as the same random process, their underlying construction processes are very distinct. The random time series is constructed completely at random

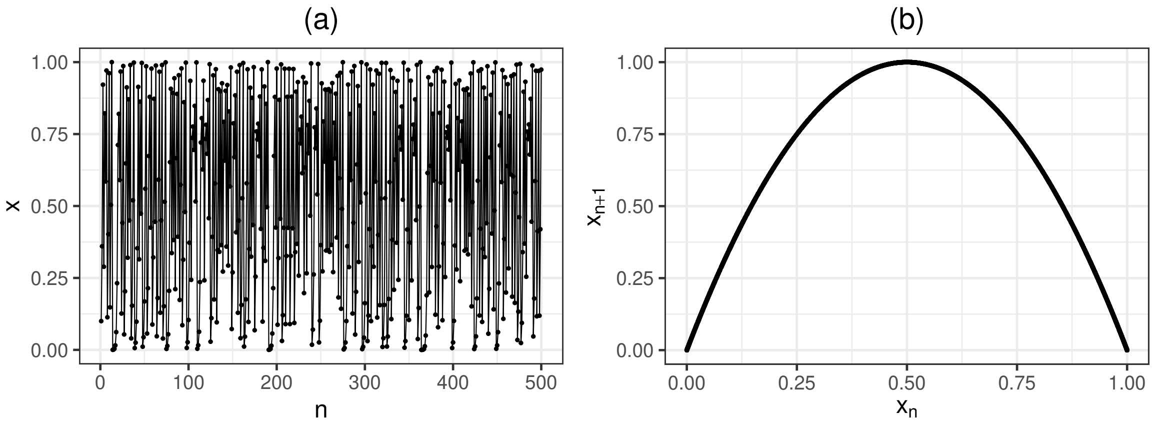


Figure 3.2. Illustration of (a) the first 500 data points of a time series generated by a logistic chaotic map, with $\rho = 4$ and $x_0 = 0.1$, and (b) a scatter plot of the whole time series of length $m = 5000$, consisting of the relation between their consecutive points (x_n, x_{n+1}).

following a uniform distribution, each data point being randomly chosen. For instance, the next point x_{n+1} does not have any relation with the current data point x_n, which can be observed by the scatter plot of Figure 3.1b. On the contrary, the time series from the logistic chaotic map is constructed by following a function $F : x_n \to x_{n+1}$, which is given by

$$x_{n+1} = \rho x_n (1 - x_n), \tag{3.1}$$

with $0 \leq x_n \leq 1$ and $0 \leq \rho \leq 4$ [May, 1976; Ravetti et al., 2014]. Thus, each point is not chosen at random, instead, there is a relation between the next x_{n+1} with the current x_n. Figure 3.2b illustrate this dependence between consecutive points by showing a scatter plot for each (x_n, x_{n+1}).

A chaotic system is an interesting type of dynamical system because it has a deterministic formula that rules their evolution but, for specific conditions, it presents a seemingly erratic, aperiodic behavior, and can be very sensitive to small variations on their initial values [Strogatz, 2001; Toker et al., 2020]. As described by Strogatz [2001], a system is considered chaotic if two nearby states moves apart from each other exponentially fast. Furthermore, a chaotic system is also bounded by some limits and their values at different states never go to infinity [Toker et al., 2020]. For instance, the chaotic behavior of the logistic map occurs when the increase rate is $\rho = 4$, and their values at different states are bounded by the interval $[0, 1]$.

To better discuss the potential of using this study of dynamics to the analysis of time series, let us first describe some concepts and classical knowledge discovery strategies, which are presented in next section.

3.2.3 Knowledge discovery in time series

Most of the recent solutions on data analysis already consider the availability of a large amount of data, which is exactly the case for IoT. A trend in this analysis, which aims to discover knowledge from these data, consists of applying data mining to time series. From the literature in time series data mining [Keogh and Kasetty, 2003; Lin et al., 2002, 2003; Bettaiah and Ranganath, 2014], in the following we define the main strategies to adapt data mining concepts for time series.

Indexing (query by content). A conventional strategy to find an element from a given database consists of searching for it based on an information that could describe it (e.g., tags, descriptions) [Perera et al., 2014b]. For the data mining domain, given a time series $\mathbf{x}$ and a dataset $\mathbf{D}$, the search is made by looking for the most similar time series $\mathbf{z} \in \mathbf{D}$ based on a similarity/dissimilarity measure $d(\mathbf{x}, \mathbf{z})$. In this case, the searching information is the content from the time series itself.

Clustering. Instead of having a time series as starting point to make comparisons, as the indexing case, for clustering strategies it is assumed no previous information regarding time series. In this case, the objective of clustering is to find groups of similar time series from a given dataset $\mathbf{D}$, based on a similarity/dissimilarity measure $d(\mathbf{x}, \mathbf{z})$, where $\mathbf{x}, \mathbf{z} \in \mathbf{D}$. The intention is to discover such groups, clusters, in an unsupervised learning fashion, increasing the similarities between time series inside the same cluster, while increasing the dissimilarities among different clusters.

Classification. Differently from clustering, classification is a supervised learning process that assumes a training step, where a dataset $\mathbf{D} = (\mathbf{x}_i, y_i)_{i=1}^{n}$ with labeled time series is used to first learn their classes, where $y_i \in \mathcal{Y}$ is the set of known classes. Then, given an unlabeled time series $\mathbf{z}$, the intention is to assign it to one of the previously learned classes y_i.

Segmentation (summarization). Given a large time series $\mathbf{x}$, with $m = |\mathbf{x}|$ being its length, the intention for the segmentation is to construct an approximation $\bar{\mathbf{x}}$ for it that summarizes its mains characteristics. The intention is that the new length $k = |\bar{\mathbf{x}}|$ be much smaller than the original, $k \ll m$, such that the benefits of using $\bar{\mathbf{x}}$ exceed the losses from the approximation.

Anomaly detection. For a given, or learned, model of the "normal" behavior from a time series $\mathbf{x}$, the intention here is to find the sections of $\mathbf{x}$ that significantly diverge from that normal behavior. An anomaly, thus, can be defined as an unexpected behavior from a subsequence within a given time series [Chandola et al., 2009]. Another approach for anomaly detection consists of comparing the behavior of a different time series $\mathbf{z}$ with the learned one. Thus, it is inferred if this behavior is similar to the previous learned, i.e., regular, or if it is different, indicating an anomaly [García-Teodoro et al., 2009].

Motif detection. While anomalies are used to identify unexpected behavior, a motif is defined as a pattern that is recurrent within a time series [Lin et al., 2002; Yankov et al., 2007]. For a time series $\mathbf{x}$, the aim here is to find a subsequence $\mathbf{s}_{i,j}$ within $\mathbf{x}$, with $i, j \in \{1, \dots, m\}$, $i < j$, and $m = |\mathbf{x}|$, that repeats along $\mathbf{x}$. The repetitions are computed based on a similarity/dissimilarity measure $d(\mathbf{s}_{i,j}, \mathbf{s}_{k,l})$ for two different subsequences, either by looking for exactly or approximate matching.

3.3 The CoIoT time series

As described in Section 2.5, the CoIoT scenarios have several challenges that requires special attention. These challenges are directed responsible for adding issues on their generated time series, such as unevenly spaced intervals and missing data. To define this challenging type of time series, in this section, we revisit the time series concepts presented in Section 3.2.1 by considering these novel issues in those definitions.

3.3.1 Modeling time intervals

To discuss how unevenly time spaces impacts the analysis of CoIoT time series, let us extend the Definition 3.2.2, which describes an ideal time series dataset to consider different time intervals between consecutive data points. But, first, we have to briefly discuss the time index from our previous time series notation.

Thus, for each index i presented in Definition 3.2.1 of time series notation, let us assume a function $T \colon \mathbb{N} \to \mathbb{R}$, which maps each i to the sequential notion of time $t \in \mathbb{R}$. For some well-defined and controlled scenarios, the time interval $\delta \in \mathbb{R}$ between subsequent data points x_i and x_{i+1} is constant, such that $\forall i \in \mathbb{N} \colon \delta = T(i+1) - T(i)$, and $T(i) = (i-1)\delta + 1$. Thus, a representation for the time series, with respect to its time intervals, can be expressed by $\mathbf{x} = < x_1, x_{\delta+1}, x_{2\delta+1}, \dots, x_{(m-1)\delta+1} >$.

A collection of n time series with equal time intervals can be arranged in a dataset

$$\mathbf{D}_1 = \mathbf{D} = \{\mathbf{x}_i\}_{i=1}^{n}. \tag{3.2}$$

In this case, the length $|\mathbf{x}_i|$ is the same for each ith time series, and the overall size of the dataset is, thus, $n \times m$. When different time series may have different time intervals, a dataset within this case can be modeled as a collection of tuples, such that,

$$\mathbf{D}_2 = \{(\mathbf{x}_i, \delta_i)\}_{i=1}^{n}. \tag{3.3}$$

In this case, each ith time series $\mathbf{x}_i$ contains its own time intervals δ_i, and different time series may have different lengths, according to its time intervals.

Thus, if $\mathbf{x}_i = (x_1^i, x_2^i, \ldots, x_m^i)$ and $\mathbf{x}_j = (x_1^j, x_2^j, \ldots, x_k^j)$, with $m \neq k$, the size of $\mathbf{D}_2$ can be obtained by computing $\sum_{i=1}^{n} |\mathbf{x}_i|$.

There are cases where different time intervals can occur within the same time series, configuring a case of unevenly spaced sampling points [Möller-Levet et al., 2003]. For instance, a given time series with length n, in this case, can be modeled by considering a specific time interval for each data point. Thus, let $\boldsymbol{\delta} = \{0, \delta_2, \ldots, \delta_m\}$ be the vector of individual time intervals, a time series indexed by its time intervals is given by $\mathbf{x} = < x_1, x_{1+\delta_2}, x_{1+\delta_2+\delta_3}, \ldots, x_{1+\sum_{i=2}^{m}\delta_i} >$. A dataset composed by n time series with unevenly spaced sampling points can be expressed as

$$\mathbf{D}_3 = \{(\mathbf{x}_i, \boldsymbol{\delta}_i)\}_{i=1}^{n}. \tag{3.4}$$

Another modeling strategy for this last case is to assume that the unevenly spaced intervals is defined by a random process. Thus, each δ_i is a random variable following a probability distribution p_i. A dataset representing a collection of such time series can be expressed as

$$\mathbf{D}_4 = \{(\mathbf{x}_i, \Delta_i)\}_{i=1}^{n}, \tag{3.5}$$

where each $\Delta_i \sim p_i$ is defined over the set of real valued time intervals. The number of data points per time series is also a random variable, so it is also for the whole dataset. For a long enough time series, the average δ_μ and other statistics can be estimated from the time series.

These time series models are useful to understand what kind of problems are present in the data. For instance, while the first model is sufficient for most of the regular problems in time series classification [Bagnall et al., 2017], this last model is useful for describing sensors where data is gathered according to a random event.

3.3.2 Modeling data availability

Another important aspect to consider when modeling time series is related to the mode its data are available. According to this, a data analysis can be classified as offline or online [Gama et al., 2014]. In traditional data analysis, the data are first collected and datasets are processed offline, being available from the beginning. For instance, this is the case for most of the classification strategies in current literature [Bagnall et al., 2017]. On the other hand, there are cases where data are available as streams, where data points continuously arrive each at a time. For these cases, time series are dynamically constructed, and data analysis should be performed online, processed as new data points are received [Gama, 2012].

To represent this transient behavior, let $\mathbf{x} = (\ldots, x_1, \ldots, x_{t-1}, x_t, \ldots, x_m, \ldots)$ be a time series constructed from a data stream, in which each data point is received at a time. To make a parallel with the static time series representation, it can be mapped by the observed interval from this stream, comprising the points from x_1 to x_m. A simple dataset representation for a collection of streaming time series can be defined in terms of this observed interval, being similar to the previously presented ones.

In addition to the fact that the whole time series is not available to be processed at the beginning of the analysis, another important aspect is that the underlying process generating the data streams can change over time [Kifer et al., 2004; Boracchi and Roveri, 2014]. Let us assume that the data points in the interval $\mathbf{x}_{[1,t-1]} = (x_1, \ldots, x_{t-1})$ are generated following a probability distribution p_1, and data points in the interval $\mathbf{x}_{[t,n]} = (x_t, \ldots, x_n)$ follows a different distribution p_2. Thus, since $p_1 \neq p_2$, the moment t, when the distribution changes, is named the change point [Kifer et al., 2004].

To detect this change point is the aim of many strategies in data stream processing. However, since it is not guaranteed to exactly find t, the intention is to detect the change by a significant difference between probability distributions p_1 and p_2 [Kifer et al., 2004]. The better strategy should minimize the number of occurrences of false-positives (FP), where changes were erroneously detected, the number of false-negatives (FN), where real changes were not detected, and the detection-delay (DD), comprising the sampling interval between the real change point t and its actual detection t^*, given by $t^* - t$ [Boracchi and Roveri, 2014].

3.3.3 Defining the CoIoT time series

In Chapter 2, the Collaborative Internet of Things was defined, and its main characteristics were presented. From the challenges faced by any solution applied to CoIoT, it is possible to highlight those which directly impact on its data generating process. An aspect that always deserves mention is the large amount of data, generated by a vast number of heterogeneous sensors. However, one of the main challenges when dealing with this massive scaling of CoIoT [Stankovic, 2014] is that their sensors are producing streams of data at high-speed [Gama, 2012].

The data generating time interval ranges from a few seconds to minutes, or even hours, between each data point. Data analysis can be performed both online, by processing the most recent data points, or offline, by considering the history of time series. For both cases, the streaming data can be mapped into a static representation by cutting the time series by an observation interval. The historical representation of the time series is, generally, stored in large big data platforms (.e.g, ThingSpeak) and is available for use after querying by a particular keyword(s) that matches their descriptions. Although this does not directly affect data, it may impact on the correct sensor search procedures, since descriptions are rare and poorly assigned [Borges Neto et al., 2015].

So far, given an ideal scenario of CoIoT, their time series can be modeled by a dataset similar to $\mathbf{D}_2$, presented in Equation 3.3, where each time series has its own fixed time interval δ. However, although these characteristics are challenging enough to any proposed strategy, such as the scaling factor and the differences in lengths between time series, in practice, there are more issues that must be concerned.

The time series from CoIoT sensors may have missing data points, variations in their time intervals, and more imprecise data, with noise and outliers being a real issue. This is mainly due to their cheap components, that can easily fail or generate uncertain data. Furthermore, the data transferred over the best-effort protocols of Internet, and the lack of commitment to data quality by sensors' owners are also concerning aspects. Thus, a model that best fits the CoIoT scenarios is by assuming a collection of time series with random time interval, as described by $\mathbf{D}_4$, in Equation 3.5.

Figure 3.3 gives an example of a temperature sensor from an Arduino-based weather station[1] located in Massachussets, USA, and available in the ThingSpeak platform. This time series data was collected from the period between 1st January 2015 to 28th July 2016 and illustrates these mentioned problems. In Figure 3.3a the time series for the temperature values (°F) is presented, and in Figure 3.3b the time intervals (s) between each consecutive data point is shown.

A data missing in the series is detected by an increasing in the time intervals. In this case, if a single data point is missed, the time interval doubles, and so on. The normal behavior for this sensor is to collect data at intervals of 60 seconds. Around the

[1]Example of an Arduino-based weather station: `https://thingspeak.com/channels/12397`

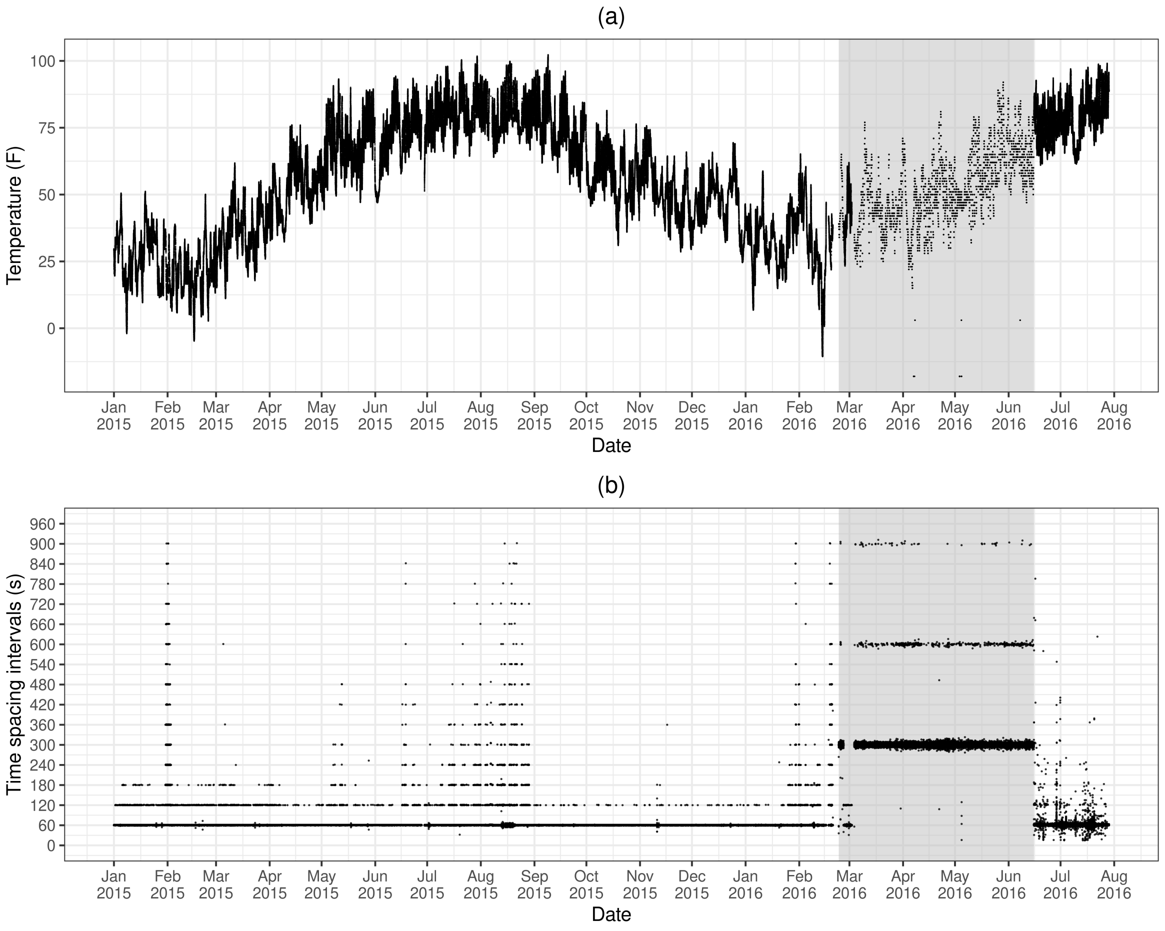


Figure 3.3. Example of (a) time series of a temperature sensor from an Arduino-based weather station, and (b) the time intervals between consecutive data points.

beginning of February 2015 and between mid-June 2015 to August 2015, the number of missing points considerably increased, reaching 16 consecutive data points. The exact reason behind this behavior is unknown. Depending on the data missing rates, it is more difficult to assure the time interval is constant, and some cases are just errors from the missing points, or if it is variable.

This behavior was dominant throughout the period of 2015, as shown in Figure 3.3b. However, after 3 days of inactivity, in 23rd February 2016, the sensor starts collecting data at intervals of 300 seconds, and remains at this rate until 15th June 2016. This period is illustrated as a dashed gray area in the figures. This changing behavior may affect several data analysis, since data points were more sparse, as seen in Figure 3.3a. Another important changing at this point is the increasing uncertainty in time intervals. Even after returning to previous behavior, these values were more random, fluctuating at 60 seconds, but with an increasing error range. These factors reinforce the definition of CoIoT as a challenging type of time series, with streaming data generation and random time intervals.

3.4 Time series representations

Time series notation is a powerful representation that can be used for extracting knowledge of data in time domain. However, some problems require different representations in order to extract additional information about different behaviors [Bagnall et al., 2015, 2017]. A reasonable way to improve the analysis of time series consists of applying a transformation to the data, resulting in a different representation domain. This strategy has the intention to highlight the more distinguishable aspects, which are not readily available from the raw data [Bagnall et al., 2012; Wang et al., 2013]. For some successful approaches, the use of a transformation on the time series data is an essential step prior the data analysis [Dempster et al., 2020].

Some time series transformations are intended to reveal novel aspects of data, but some of them are used to mitigate some problems, such as summarizing original data points into a more comprehensible format [Wilson, 2017]. Thus, there are transformations performed on the same time domain, i.e. the transformed data continues indexed by time, and there are transformations that change the data from time domain into a different domain (e.g., frequency, power spectrum, wavelets, auto correlations, shapelets [Bagnall et al., 2012, 2015, 2017; Lines et al., 2012; Wang et al., 2013]). In the following, some time series representations suitable for CoIoT scenarios are presented.

3.4.1 Time domain representations

The first aspect a proper representation must deliver is to minimize the problems from CoIoT time series. Once the type of time series from CoIoT sensors is defined, see Section 3.3.3, it is possible to discuss a time series representation that best fits its characteristics. From a time series knowledge discovery perspective, as described in Section 3.2.3, and given the challenges from our time series, such as the heterogeneity and diversity of time series lengths and characteristics, it is clear that solutions based on the raw data is impracticable. For instance, taking the clustering applications as example, the computation of some distance measures (e.g., Euclidean, Fréchet, Dynamic Time Warping (DTW) [Montero and Vilar, 2014]) from the raw data points of a pair of time series is not suitable, since the same length for time series can not be assumed.

Furthermore, even when the time series have equal length, they could be long enough so the computation for direct pairwise comparison could be unfeasible. So, a direct solution to this case can be an adjustment of the time series by only considering the data points that occur at similar timestamp on both time series. A similar timestamp is used because it is hard to assure the data is being collected exactly at the same time, so a time interval should be considered. Problems with this approach are twofold: (i) to define what is the ideal time interval to assume that two data points belong to a similar moment in time, and (ii) the data points from a time series without its correspondent in another will be discarded. This last issue is important since information is being lost, which is essential for the proper knowledge discovery.

A time series representation that is being largely used in literature with considerable success is the approach proposed by Lin et al. [2003], named the Symbolic Aggregate approXimation (SAX) algorithm. SAX transforms the time series into a symbolic representation, with dimensionality reduced and which symbols from a common alphabet a. The steps to the SAX transformation are

1. Normalize each time series to have a zero mean and unitary standard deviation, to compare them with the same offsets and amplitudes. For a given time series $\mathbf{x} = (x_1, \ldots, x_m)$, where $\bar{\mathbf{x}}$ and $s_{\mathbf{x}} \neq 0$ are, respectively its mean and standard deviation, this normalization is given by

$$x_i = \frac{x_i - \bar{\mathbf{x}}}{s_{\mathbf{x}}}. \tag{3.6}$$

2. Reduce the dimensionality of time series, so they have the same number of samples w, with $w \ll m$. This is made with their Piecewise Aggregate Approximation (PAA) algorithm to discretize the time series by performing a transformation $f : \mathbb{R}^m \to \mathbb{R}^w$. Let us partition the time series in w pieces so $\mathbf{x} = (c_1, \ldots, c_w)$, with data points divided into each partition c_i. The PAA discretization creates a new time series $C = (\bar{c_1}, \ldots, \bar{c_w})$, where each $\bar{c_i}$ corresponds to the mean of data points within the partition c_i. This is obtained by

$$\bar{c_i} = \frac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} c_j. \tag{3.7}$$

3. Symbolize the PAA discretization of the time series into a new representation of sequential values (the SAX symbols). Each $\bar{c_i}$ is replaced by a symbol from the alphabet $a = \{0, \ldots, \alpha\}$, by splitting the range of time series values in equiprobable regions, considering the data points, after normalization, spreads as a Normal distribution $\mathcal{N}(0, 1)$. This is achieved by computing breakpoints $B = \{\beta_1, \ldots, \beta_{\alpha-1}\}$ between these regions, where the breakpoint divides the Normal distribution in

α regions, in which each of them have the same probability $1/\alpha$ (the same area under the distribution curve).

After the transformation to the SAX symbolic representation, in an ideal scenario, the time series will have the same length w, composed by the same magnitude of values, defined by α, being ready to pairwise comparisons. However, in the CoIoT case, it is possible that time series have gaps, depending on different periods of data collection or even on fails in sensors, and this will impact on the SAX transformation.

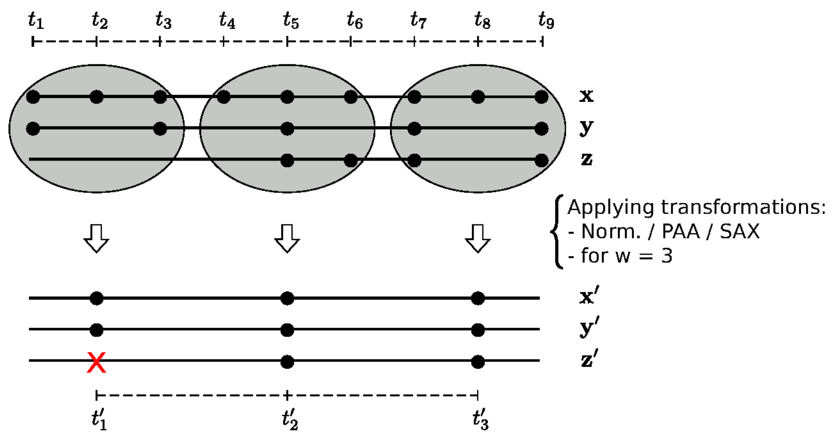

t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9
x
y
z
Applying transformations:
- Norm. / PAA / SAX
- for w = 3
x'
y'
z'
t'_1 t'_2 t'_3

Figure 3.4. Graphical illustration of time series transformation with the SAX symbolic representation.

To illustrate this case, in Figure 3.4 the time series $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ are transformed following the SAX symbolic representation, with $w = 3$, creating the new time series $\mathbf{x}'$, $\mathbf{y}'$, and $\mathbf{z}'$, respectively. Let us assume that the time spacing intervals for these time series are, respectively, $\delta_x = 1$, $\delta_y = 2$, and $\delta_z = 1$. Both $\mathbf{x}$ and $\mathbf{y}$ perfectly generated their data samples at their time intervals, but $\mathbf{z}$ only has data from time t_5, and a data miss occurred at t_8. For the two time series, the SAX transformation perfectly reduced their length to 3 data points, represented as new time stamps t'_1, t'_2, and t'_3. The data missing in $\mathbf{z}$ at t_8 was not a problem, so the mean considered t_7 and t_9 to obtain it. However, since the data points t_1, t_2, and t_3 are missing, the new value t'_1 can not be computed, yielding a missing data point even after transformation.

While other time domain representations exist [Wilson, 2017], this example reinforces that transformations under the same time domain may still have some of the same problems, although minimized, as the original raw data.

3.4.2 Features domain representations

In opposite to the strategies based on the time domain of time series, an important step towards a better representation for the CoIoT time series is their transformation into a set of features [Wang et al., 2006; Fulcher et al., 2013; Fulcher and Jones, 2014]. A feature can be any measure computed (extracted) from the time series that summarizes a specific aspect of them. A common approach is to extract more than a single feature for each time series, so different aspects could be represented. For a given time series $\mathbf{x}$ with length m, by extracting k features from it, one is performing a transformation $f : \mathbb{R}^m \to \mathbb{R}^k$, with, usually, $k \ll m$.

In fact, the first advantage of strategies based on the extraction of features from time series is the dimensionality reduction and equalization of different lengths for the time series, which seems to be very appropriate to our problem. However, two main aspects must be considered before features are extracted, so their representativeness could be valid to the problem domain: (i) the decision of which features to extract, and (ii) from which representation they will be extracted. The chosen features must be related to the problem domain, and this decision should assure that the main characteristics of the time series are being captured.

In order to express the main aspects from time series of different natures, Wang et al. [2006] proposed to use a set of statistical features, namely: trend, seasonality, periodicity, serial correlation, skewness, kurtosis, non-linearity, self-similarity, and chaos. From these features, seven were extracted from the raw time series and six from a transformed version of it, by adjusting the de-trending and de-seasonalizing the time series, giving a total of 13 features. Fulcher and Jones [2014] proposed the extraction of near 9,000 features, where some of them are different due to parameters variations. These features cover a wide range of time-series characteristics, such as basic statistics, linear correlations, stationarity, information theoretic measures, among others.

Although very interesting approaches, for both cases, the best results were only achieved after applying a feature selection strategy. Both cases used a greedy forward search algorithm that incrementally adds one feature at a time. From a set of k features, each feature is verified, one-by-one, by a quality measure, and the best feature is selected. The previous selected feature is, then, verified in conjunction with the other $k - 1$ remaining features, selecting the one that best performed together with the first. This process is repeated until all features were selected, or by a termination criterion, e.g., there is no more improvement in the quality measure used. While this is feasible for well-behaved and controlled scenarios, it may not be the best strategy to be applied for the CoIoT case, given the magnitude of its numbers, as previously discussed.

On the other hand, by extracting the features directly from the raw time series representation, some issues may invalidate their representativeness. For instance, one must consider the case where some features may not be computed simply because of problems found in the data. Otherwise, some features may have their computed values affected by problems in data. It is important that the chosen methods be robust to data problems, such as data missing, outliers, irregular time spacing, etc. Furthermore, their computation from the raw data may lead to a problem in which the features became dependent on the current instance of the time series, instead of a representation of the characteristic behind it. This may invalidate the set of features if their values changes as a function of time, and it may require another computation of features, that could be computationally expensive [Wang et al., 2006].

For the CoIoT applications, it is reasonable to consider the features that could extract the behavior of the phenomena in which the sensor is monitoring, and not being dependent on the current sample. Thus, an appropriate strategy should be to extract the features after performing a time series transformation into a different domain. Such domains are able to highlight different aspects that can be used as the basis for the extraction of features, better representing the time series intrinsic characteristics [Wang et al., 2006; Baydogan et al., 2013].

In the following, we focus our attention on two time series transformations which gives significant results when applied to real world time series data: the transformation from time series into a graph representation and a set of ordinal patterns.

3.4.3 Graph domain representations

Another direction that demonstrated to be successful in capturing essential characteristics of time series is based on their transformation into network representations. This transformation enables the time series analysis from a different view point, now it can be made through graph and complex networks theories [Zhang and Small, 2006; Zhang et al., 2008; Gao and Jin, 2012; Tang et al., 2013; Gao et al., 2016; Lacasa et al., 2008; Luque et al., 2009; Ravetti et al., 2014; Gonçalves et al., 2016].

A successful approach in representing time series main characteristics through networks is the Visibility Graph (VG), which is constructed by looking to the visibility of their elements [Lacasa et al., 2008; Luque et al., 2009; Ravetti et al., 2014; Gonçalves et al., 2016]. Let $G_V = (V, E)$ be the visibility graph constructed from a time series $\mathbf{x} = (x_1, \ldots, x_m)$. Each data point x_i from the time series is represented by a vertex $v_i \in V$, such that $V = (v_1, \ldots, v_m)$ in the visibility graph. Two vertices are connected if they satisfy the visibility criterion. There are two approaches considering this criterion definition, the visibility graph [Lacasa et al., 2008] and the Horizontal Visibility Graph (HVG) [Luque et al., 2009].

For the visibility graph, there is an edge (v_a, v_b) between vertices v_a, and v_b if it is possible to trace a line between their respective data points in the time series without intersecting intermediate data points [Lacasa et al., 2008]. For instance, the visibility criterion is true for two data points x_a and x_b if, for all points x_c between them,

$$x_c < x_a + (x_b - x_a)\frac{c-a}{b-a} \qquad \forall c \text{ such that } a < c < b. \tag{3.8}$$

More strictly, for the horizontal visibility graph [Luque et al., 2009] approach, the visibility criterion is true if it is possible to trace a horizontal line between two data points x_a and x_b without intercepting any intermediate data points x_c. In this case,

$$x_a, x_b > x_c \qquad \forall c \text{ such that } a < c < b. \tag{3.9}$$

These visibility approaches enable the construction of networks that inherit several characteristics of the underlying time series. For instance, periodic time series are converted into regular graphs and random series into random graphs [Lacasa et al., 2008; Luque et al., 2009]. However, given the one-to-one correspondence between data samples and network vertices, visibility graphs often grow with the length m of time series, which is not scalable when considering the CoIoT scenarios.

As pointed out by Ravetti et al. [2014], one may fail in correctly distinguishing the characteristics of time series by analyzing only simple graph parameters, so relatively sophisticated graph measures must be employed. While this reinforces the fact that graph representations can be an interesting transformation to capture essential time series aspects, it also emphasizes the need for a more efficient graph representation.

3.5 Ordinal patterns representations

In this thesis, we propose a deeper exploration on the use of time series dynamical behavior as a feasible domain for the IoT data. A representation of time series that is able to capture this behavior is based on the ordinal patterns transformation. This transformation has been extensively discussed in recent studies, where it was shown to be well suited for real-world data [Rosso et al., 2013; Aquino et al., 2015; Rosso et al., 2016; Aquino et al., 2017; Ribeiro et al., 2017]. Furthermore, it was proposed with a main focus in the simplicity and fast calculation, and is also robust to observational and dynamical noises. These are all important properties for our application scenario, which will be exploited in our proposal.

In this section we introduce this transformation, which represents the time series data as a set of Ordinal Patterns (OP), and their further analysis, showing how to discriminate between different behaviors for the time series dynamics. All codes of transformations presented here are available at a public repository[2].

3.5.1 Ordinal patterns transformation

Bandt and Pompe [2002] proposed a methodology for the transformation of time series that has been a cornerstone contribution in the study of time series dynamics [Aquino et al., 2017; Zunino et al., 2012; Rosso et al., 2013; Aquino et al., 2015; Rosso et al., 2016; Aquino et al., 2017; Ribeiro et al., 2017]. This transformation consists of creating a set of symbolic patterns, according to the ordinal relation between successive data points from the time series. It is based on two parameters, the embedding dimension D, which is related to the length of the patterns, and the embedding delay τ, which defines the time scale interval between consecutive data points considered for D.

Formally, for a given time series $\mathbf{x} = (x_1, \dots, x_n)$ of length n, an embedding dimension $D \in \mathbb{N}$, and an embedding delay $\tau \in \mathbb{N}$, the method consists of generating sliding windows $\mathbf{w}_t \subseteq \mathbf{x}$ of length D each, for each time instant t, such that the elements within each sliding window can be separated by intervals of size τ, corresponding to a sample of the time series by regular spaced intervals [Zunino et al., 2012]. Thus, the sliding window for each instant $t = 1, \dots, n-(D-1)\tau$ is defined as

$$\mathbf{w}_t = (x_t, x_{t+\tau}, \dots, x_{t+(D-2)\tau}, x_{t+(D-1)\tau}). \tag{3.10}$$

The ordinal relation [Rosso et al., 2007a] for each sliding window consists of the necessary permutation in the elements of $\mathbf{w}_t$, so they are sorted in ascending order with respect to their values. Thus, for each window $\mathbf{w}_t$, at a given instant t, the ordinal pattern consists of the permutation $\pi_t = (r_1, r_2, \dots, r_D)$ of $(1, 2, \dots, D)$ such that

$$x_{t+r_1-1} \leq x_{t+r_2-1} \leq \cdots \leq x_{t+r_{D-1}-1} \leq x_{t+r_D-1}. \tag{3.11}$$

The method originally proposed by Bandt and Pompe does not consider an embedding delay, so it is equivalent to the case where $\tau = 1$, and the sliding windows were sampled by consecutive data samples. Consequently, for $\tau > 1$ the sliding windows consists of τ-spaced data points.

Figure 3.5 illustrates the process of constructing the ordinal patterns through the ordinal patterns transformation. For the presented time series, with $D = 3$ and $\tau = 1$, the sliding windows of length 3 are sequentially obtained and the ordinal relation for each of them are computed. For instance, the first sliding window obtained at $t = 1$

[2]Implementations of the Bandt and Pompe's ordinal patterns transformations: `https://github.com/labepi/bandt_pompe`.

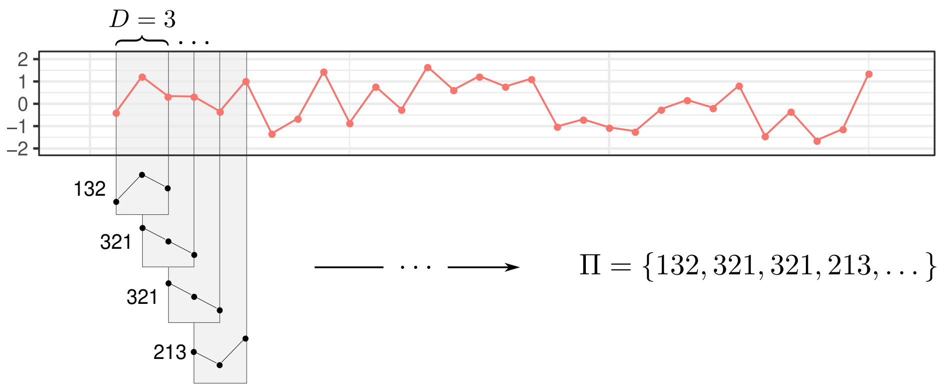


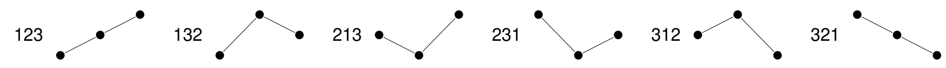
Figure 3.5. Illustration of the ordinal patterns transformation process, considering an embedding dimension $D = 3$ and an embedding delay $\tau = 1$.

is given by $\mathbf{w}_1 = (w_{1,1}, w_{1,2}, w_{1,3}) = (-0.37694, 1.22490, 0.34387)$, and the necessary permutation to order it is to switch the second with third elements, keeping the first data point as it is. This is equivalent to the new ordered $\mathbf{w}'_t = (w_{1,1}, w_{1,3}, w_{1,2})$. So, this corresponds to the 132 pattern, as presented in the figure.

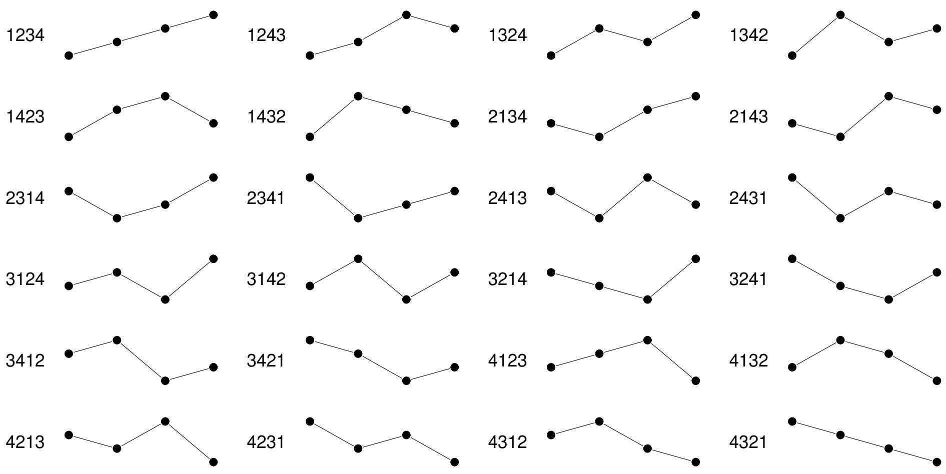
For the second and third sliding windows we have $\mathbf{w}_2 = (1.22490,\ 0.34387,\ 0.32845)$ and $\mathbf{w}_3 = (0.34387, 0.32845, -0.33761)$, respectively. Since both of them are in reverse order, they need the same permutation to be ordered, which corresponds to the 321 patterns. It is worth noting the transformation does not consider variances in the amplitudes of data points, the pattern is computed only with respect to the ordinal relation between them, thus the ordinal name. Even if the values of two sliding windows are way distinct in magnitudes, if they need the same permutation to be ordered, they will be represented as the same ordinal pattern.

After the transformation, the time series is converted onto a sequence of ordinal patterns $\Pi = \{\pi_i : i = 1, \dots, m\}$, where $m = n - (D - 1)\tau$ and each π_i represents a permutation from the set of $D!$ possible permutations. The choice of D depends on the length n of the time series, and the condition $n \gg D!$ must be satisfied in order to obtain reliable statistics. For practical purposes D is in the interval between 3 and 7 [Bandt and Pompe, 2002; Zunino et al., 2012]. In Figure 3.6 we present all the $D!$ possible permutation patterns for (a) $D = 3$ and (b) $D = 4$, following the graphical notation stated by Parlitz et al. [2012].

Algorithm 1 illustrates the steps necessary to perform the transformation of a given time series $\mathbf{x}$ of size n to its ordinal patterns Π. The time complexity of this algorithm is $O(nD^2)$, assuming that the permutation is obtained by sorting each sliding window by the function `Order()` in Line 4, which can be a simple sorting algorithm such as the selection sort. However, given the practical recommendation of D being a small value ($D \in \{3, \dots, 7\}$), the sorting algorithm will take at most 7 elements, and



(a)



(b)

Figure 3.6. Illustration of all $D!$ possible permutation patterns for (a) $D = 3$ and (b) $D = 4$, following the graphical notation from Parlitz et al. [2012]. Each pattern represents a size D subsequence of data points for a time series, in which different levels for the points represent different values from the time series.

the complexity of this strategy depends largely on the size n of the time series, which is $O(n)$ in practice.

```
ALGORITHM 1: OrdinalPatterns
Input: The time series $\mathbf{x}$ of size $n$, the embedding dimension $D$, and the embedding
       delay $\tau$.
Output: The sequence of ordinal patterns $\Pi$.
// The sequence of ordinal patterns
1 $\Pi \leftarrow \{\pi_i : i = 1, \dots, n - (D-1)\tau\}$ ;
2 for $t \leftarrow 1$ to $n - (D-1)\tau$ do
      // Get the sliding window w at $t$
3     $\mathbf{w}_t \leftarrow x_t, x_{t+\tau}, \dots, x_{t+(D-2)\tau}, x_{t+(D-1)\tau}$ ;
      // The current permutation pattern index
4     $\pi_i \leftarrow$ Order($\mathbf{w}_t$) ;
5 return $\Pi$;
```

Once the ordinal patterns are constructed from the time series, the next step is to analyze them in order to capture the dynamics of their generating time series. Two reasonable methods to do this is by taking into account the frequency of patterns and the sequence of their occurrences. For the first method, the analysis is performed through the ordinal patterns probability distribution p_π, and for the second case, we map the transitions between adjacent patterns to a directed weighted graph, the ordinal patterns transition graph G_π. Both transformations are derived from the first ordinal patterns transformation, obtained from the resulting set of ordinal patterns Π. Thus, features can be extracted from p_π and G_π to expand the knowledge regarding the set of patterns, and, consequently, their underlying time series dynamics.

3.5.2 Ordinal patterns probability distribution

Given the set Π of ordinal patterns, obtained from the transformation of a given time series, the Ordinal Patterns Probability Distribution (OPPD), denoted by p_π, consists of assigning a probability distribution to the permutations identified in the time series. Thus, for each possible permutation $\pi_t \in \Pi$, with $t \in \{1, \ldots, D!\}$, let $|s_{\pi_t}| \in \{0, \ldots, m\}$ be the number of observed patterns of type π_t, then $p_\pi = \{p(\pi_t) : \forall t \in 1, \ldots, D!\}$ is defined as

$$p(\pi_t) = \frac{|s_{\pi_t}|}{n - (D-1)\tau}, \tag{3.12}$$

which satisfies the conditions $p(\pi_t) \geq 0$ and $\sum_{\pi_t} p(\pi_t) = 1$.

Figure 3.7 gives examples of ordinal patterns probability distributions constructed from different time series. Figures 3.7a–3.7d illustrate random time series with different correlation levels between their points. Those time series were synthetically generated according to their power spectra f^{-k} [Ravetti et al., 2014], where (a) is a white noise ($k = 0$), (b) pink noise ($k = 1$), (c) red/brown noise ($k = 2$), and (d) a black noise ($k = 3$). Their respective ordinal patterns probability distributions where constructed with $D = 3$ and $\tau = 1$, and are presented in Figures 3.7e–3.7h. For this example, the frequency of patterns 123 and 321 are higher as the correlation levels increases in the series, indicating the more regular behavior [Rosso et al., 2007a]. A suitable strategy to account for different time series dynamics is possible by computing information theory quantifiers from p_π, as presented in next sections.

Algorithm 2 illustrates the steps to compute the ordinal patterns probability distribution p_π from a given time series $\mathbf{x}$ of length n. Since the number of ordinal patterns from the time series obtained after the ordinal patterns transformation is $|\Pi| = n - (D-1)\tau$, and $n \gg D!$, this algorithm's time complexity is bounded by $O(n)$.

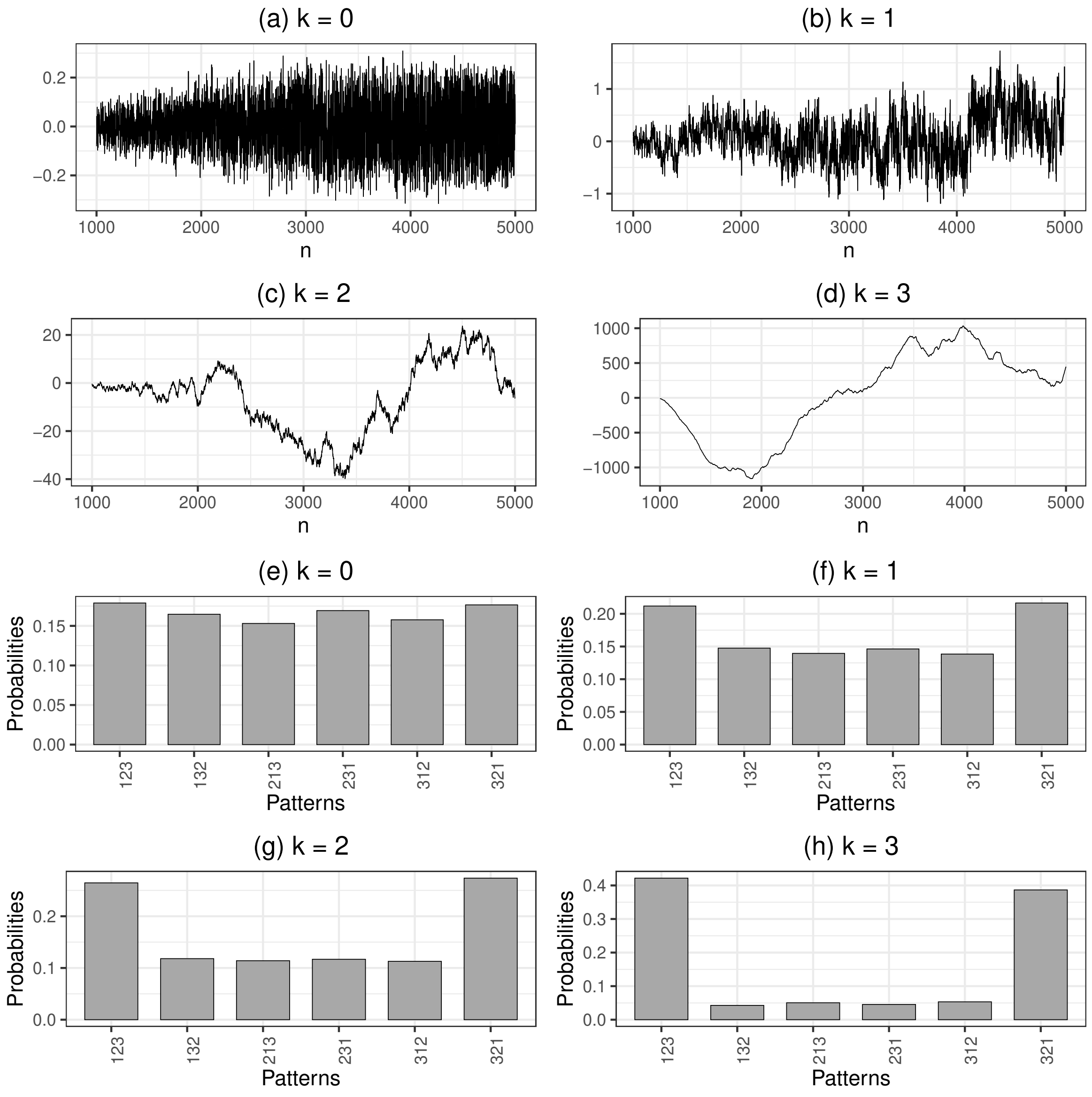


Figure 3.7. Illustration of ordinal patterns probability distributions constructed from random time series with different correlation levels. Where (a) is a white noise ($k = 0$), (b) pink noise ($k = 1$), (c) red/brown noise ($k = 2$), and (d) a black noise ($k = 3$). Their corespondent ordinal patterns probability distributions are presented in (e), (f), (g), and (h), respectively. Each time series was generated with 5,000 samples in length, and the ordinal patterns transformations were computed with $D = 3$ and $\tau = 1$.

3.5.3 Ordinal patterns transition graphs

Recent approaches for time series representations lie at the crossing between ordinal patterns and graph representations from time series. They are based on the transformation of time series into networks according to their ordinal patterns [Sorrentino et al., 2015; McCullough et al., 2015; Kulp et al., 2016; Zhang et al., 2017]. These

ALGORITHM 2: OrdinalPatternsPD

Input: The sequence of ordinal patterns Π, and the embedding dimension D.
Output: The ordinal patterns probability distribution p_π.

```
   // The distribution of permutations
1  $p_\pi \leftarrow \{p_{\pi_i} = 0 : i = 1, \dots, D!\}$ ;
2  for $i \leftarrow 1$ to $|\Pi|$ do
       // Counting the pattern frequency
3      $p_{\pi_i} \leftarrow p_{\pi_i} + 1/|\Pi|$ ;
4  return $p_\pi$;
```

approaches consider each $D!$ possible ordinal pattern as a vertex in the graph.

Given the sequence of ordinal patterns Π, the Ordinal Patterns Transition Graph (OPTG), denoted by G_π, represents the relations between consecutive ordinal patterns, and is defined as a directed weighted graph $G_\pi = (V, E)$ with $V = \{\pi_i : i = 1, \dots, D!\}$, where each vertex corresponds to one of the $D!$ possible permutations for an embedding dimension D, and a set of directed weighted edges $E = \{(\pi_i, \pi_j) : \pi_i, \pi_j \in V\}$.

A directed edge connects two ordinal patterns in the graph if they appear sequentially in the time series, representing a transition between them, thus the name "ordinal patterns transition graph". There is an edge $(\pi_i, \pi_j) \in E$, between the vertices π_i and π_j, if there is a pair of ordinal patterns $\Pi_t = \pi_i$ and $\Pi_{t+1} = \pi_j$, $1 \le t \le n-(D-1)\tau-1$. The weight function $w : E \to \mathbb{R}$ of an edge represents the probability of a specific transition in Π. Thus, the weight of a given edge (π_i, π_j) is given by

$$w(\pi_i, \pi_j) = \frac{|\Pi_{\pi_i,\pi_j}|}{m-1}, \tag{3.13}$$

where $|\Pi_{\pi_i,\pi_j}| \in \{0, \dots, n-(D-1)\tau-1\}$ is the number of transitions between permutations π_i and π_j, with $\sum_{\pi_i,\pi_j \in V} w(\pi_i, \pi_j) = 1$. Once this graph representation is constructed, some studies use unweighted edges [McCullough et al., 2015; Kulp et al., 2016] to represent only the existence of such transitions. However, in this work, we follow the approach to consider the relative frequency of transitions as the weights of edges [Sorrentino et al., 2015; Zhang et al., 2017].

Figure 3.8 illustrates the ordinal patterns transition graphs constructed from the same random time series of Figure 3.7. Each graph is constructed with $D = 3$, which corresponds to the graphs vertices, and $\tau = 1$. The weights in transitions between vertices corresponds to the relative frequencies, computed by accounting consecutive ordinal patterns.

Algorithm 3 presents the steps to compute the ordinal patterns transition graph G_π from the set of ordinal patterns Π obtained from a given time series $\mathbf{x}$ of length n.

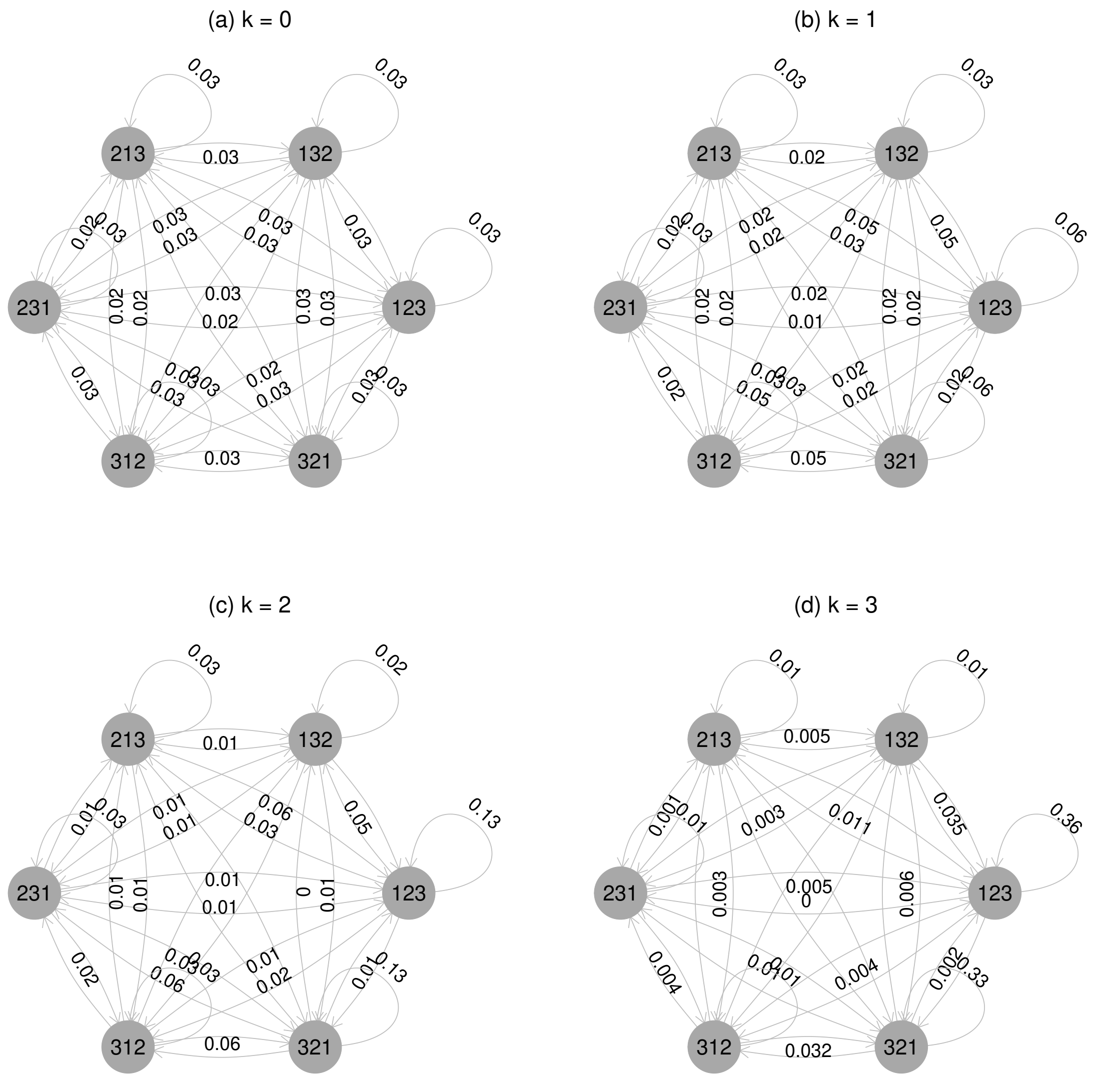


Figure 3.8. Illustration of ordinal patterns transition graphs constructed from the same random time series with different correlation levels presented in Figure 3.7, with $D = 3$ and $\tau = 1$. Where (a) is a white noise ($k = 0$), (b) pink noise ($k = 1$), (c) red/brown noise ($k = 2$), and (d) a black noise ($k = 3$).

The graph G_π is represented here as an adjacency matrix of order $D! \times D!$. The same analysis of time complexity made for the Algorithm 2 can be performed for this case. Thus, this algorithm's time complexity is bounded by $O(n)$, since it directly depends on the number of ordinal patterns $|\Pi| = n - (D-1)\tau$, and $n \gg D!$.

A deeper exploration on the properties of ordinal patterns transition graphs and their feasibility for the characterization of distinct time series dynamics is presented in Section 5.3. With respect to the analysis of both ordinal patterns distributions and

ALGORITHM 3: OrdinalPatternsTG

Input: The sequence of ordinal patterns Π, and the embedding dimension D.
Output: The ordinal patterns transition graph G_π.

```
// The adjacency matrix representation
1 $G_\pi \leftarrow \{g_{\pi_i,\pi_j} = 0 : i = 1, \dots, D! \text{ and } j = 1, \dots, D!\}$ ;
2 for $i \leftarrow 2$ to $|\Pi|$ do
      // Counting the transition patterns frequencies
3     $g_{\pi_{i-1},\pi_i} \leftarrow g_{\pi_{i-1},\pi_i} + 1/(|\Pi| - 1)$ ;
4 return $G_\pi$
```

transition graphs, in the following it is presented an overview on the interpretation of these resulting transformations and a set of features extracted from both of them.

3.5.4 Numerical analysis of the transformations

Given the previously discussed ordinal patterns transformations, a reasonable strategy to capture the characteristics from distinct dynamics consists of extracting relevant features from those transformations. For the present work we consider using metrics extracted from both the ordinal patterns probability distribution p_π and the ordinal patterns transition graph G_π.

3.5.4.1 Features from ordinal patterns distribution

From the ordinal patterns probability distribution p_π, we compute the following features

1. the normalized permutation entropy ($H_S[p_\pi]$),
2. the statistical complexity ($C_{JS}[p_\pi]$), and
3. the Fisher information measure ($F[p_\pi]$).

These features are information theory quantifiers that have already been used as significant measures for the proper characterization of the underlying time series dynamical behavior [Rosso et al., 2007a; Zhang et al., 2008; Kulp and Smith, 2011; Tang et al., 2013; Aquino et al., 2015; Gonçalves et al., 2016; Aquino et al., 2017; Ribeiro et al., 2017].

Permutation entropy Following the initial purpose of Bandt and Pompe [2002], the information quantifiers are calculated from the distribution p_π for all $D!$ permutations π of order D.

Since p_π is defined on the permutations of neighboring values, the authors proposed the *permutation entropy*, based on the classical entropy of Shannon, which is defined as

$$H[p_\pi] = -\sum_{t=1}^{D!} p(\pi_t) \log p(\pi_t), \tag{3.14}$$

where $0 \leq H[p_\pi] \leq \log D!$. The permutation entropy, equivalently to the Shannon entropy, is a measure of uncertainty associated to the process described by p_π [Aquino et al., 2017]. Lower values of $H[p_\pi]$ represent an increasing or decreasing sequence of values in the permutation distribution, indicating that the original time series is deterministic. On the other side, high values of $H[p_\pi]$ indicate a completely random system [Bandt and Pompe, 2002].

Normalized permutation entropy The maximum value for $H[p_\pi]$ occurs when all $D!$ possible permutations have the same probability of occurring, which is the case for the uniform distribution p_u of permutations. Thus, $H_{\max} = H[p_u] = \log D!$, where $p_u = \{1/D!, \ldots, 1/D!\}$. [Zunino et al., 2012].

Rosso et al. [2007a] defined the normalized Shannon entropy, from the permutation entropy case, as

$$H_S[p_\pi] = \frac{H[p_\pi]}{H_{\max}}, \tag{3.15}$$

where $0 \leq H_S[p_\pi] \leq 1$.

Statistical complexity Another statistical measure that can be computed from the ordinal patterns probability distribution p_π is the statistical complexity. While the entropy gives the notion of the uncertainty of a system, spaning the extremes of perfect predictable to completely randomness, the statistical complexity is used to represent the uncertainty between those extremes. Defined by Lamberti et al. [2004], this measure presents a different perspective regarding the knowledge of some underlying process, capturing the relationship between the dynamical components of a system, such as determinism and randomness, giving the idea of the disequilibrium between them.

Based on the Jensen-Shannon divergence JS between the associated probability distribution p_π and the uniform distribution p_u, i.e., the trivial case for the minimum knowledge from the process, the statistical complexity is given by

$$C_{JS}[p_\pi] = Q_{JS}[p_\pi, p_u] H_S[p_\pi], \tag{3.16}$$

where $p_\pi = \{p(\pi)\}$ is the ordinal patterns probability distribution, p_u is the uniform distribution over $\{1, 2, \ldots, D!\}$, and $H_S[p_\pi]$ is the normalized Shannon entropy, as previously described.

The disequilibrium $Q_{JS}[p_\pi, p_u]$ is given by

$$\begin{aligned} Q_{JS}[p_\pi, p_u] &= Q_0 JS[p_\pi, p_u] \qquad (3.17) \\ &= Q_0 \left\{ S\left[\frac{p_\pi + p_u}{2}\right] - \frac{S[p_\pi] + S[p_u]}{2} \right\}, \qquad (3.18) \end{aligned}$$

where S is the Shannon entropy measure. Q_0 is a normalization constant, given by

$$Q_0 = -2\left\{ \left(\frac{D! + 1}{D!}\right) \ln(D! + 1) - 2\ln(2D!) + \ln(D!) \right\}^{-1}, \qquad (3.19)$$

which is equal to the inverse of the maximum value of $JS[p_\pi, p_u]$, so $0 \leq Q_{JS} \leq 1$ [Aquino et al., 2017; Rosso et al., 2007b].

Fisher information Following the characterization of distinct dynamics from the ordinal patterns probability distribution p_π, the Fisher information is a measure able to capture the concentration of a given distribution. Contrary to the Shannon entropy, that gives a notion of the uncertainty of a system by measuring the global spreading of the distributions, the Fisher information is considered to have a locality property because it considers the differences among consecutive probabilities within the distribution [Sanchez-Moreno et al., 2009; Rosso et al., 2015].

The Fisher information for the discrete case is given by

$$F[p_\pi] = F_0 \sum_{t=1}^{D!-1} \left(\sqrt{p_{t+1}} - \sqrt{p_t}\right)^2, \qquad (3.20)$$

where F_0 is a normalization constant defined as

$$F_0 = \begin{cases} 1 & \text{if } p_{i^*} = 1 \text{ for } i^* = 1 \text{ or } i^* = N \text{ and } p_i = 0, \forall i \neq i^*, \\ 1/2 & \text{otherwise.} \end{cases} \qquad (3.21)$$

By measuring these local changes, the Fisher information quantifier increases as the density is more concentrated [Sanchez-Moreno et al., 2009]. For instance, for a distribution with density concentrated at a single value, $F[p_\pi] \approx 1$, while $H[p_\pi] \approx 0$.

3.5.4.2 Features from ordinal patterns transition graph

From the ordinal patterns transition graph G_π we extracted the following set of features

1. number of edges (N_E),
2. normalized Shannon entropy of edges weights distribution ($H_S[E_w]$),
3. statistical complexity of edges weights distribution ($C_{JS}[E_w]$), and
4. Fisher information of edges weights distribution ($F[E_w]$).

E_w is the distribution of edges weights from the ordinal patterns transition graph.

Once we have the graph representation of the transition between patterns, there are several possible features to be computed. For instance, Ravetti et al. [2014] construct their metrics from the nodes degree distribution, but this requires a large number of vertices to be relevant, which is not our intention, since this leads to the need for a large D, increasing the computational time for execution. On the other hand, following the studies of Zhang et al. [2017], Borges et al. [2019a], and Olivares et al. [2020], we focus on the features that represent the transitions themselves, which are related to the graph edges.

Number of edges Since the edges of G_π represent the occurrence of transitions between consecutive patterns, the number of edges is an important indicator of the time-series dynamics. For instance, as the randomness of a process increases, it also increases the chances for all possible transitions in the graph to occur. The number of edges is computed by

$$N_E = |E[G_\pi]|. \tag{3.22}$$

Information theory quantifiers from edges weights distribution Similarly to the features computed from p_π, we calculate the same information theory quantifiers from edges weights of G_π. These are the normalized Shannon entropy of edges weights ($H_S[E_w]$), the statistical complexity of edges weights ($C_{JS}[E_w]$), and the Fisher information of edges weights ($F[E_w]$).

In general, the feasibility of using those features for the characterization of time series dynamics, such as distinguishing noisy, chaotic and deterministic behaviors, has already been proven by literature [Rosso et al., 2007a, 2013; Aquino et al., 2015; Gonçalves et al., 2016; Rosso et al., 2016; Aquino et al., 2017; Ribeiro et al., 2017]. However, to assure their reliability, the length n of the time series must be long enough so the sampling in the $D!$ space of patterns is representative, i.e., $n \gg D!$. Among other problems, this avoids missing patterns, when the time series is not long enough so all the possible patterns can be observed [Rosso et al., 2012]. A typical value used in literature is $D = 6$, requiring the length of time series must be large, usually $n \geq 10^5$.

Thus, in this work it is given an important attention to the features extracted from the ordinal pattern transition graph, which can achieve significant results for small values of D and, consequently, a smaller length of time series is required.

In Section 5.5 we present a novel feature, named probability of self-transitions (p_{st}), extracted from the ordinal patterns transition graph, which consists of a valuable information for distinguishing different time series dynamics. However, before that, in the following sections we present methods for the interpretation of the ordinal patterns transformations and their features. In addition, we discuss some challenges and limitations with these strategies.

3.6 Interpretation of Bandt-Pompe's method

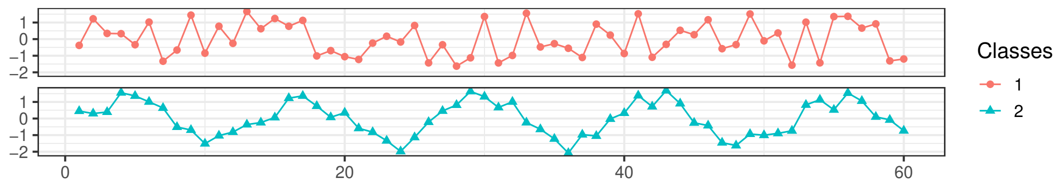
In the following sections, we present a general discussion on the interpretation of the ordinal patterns transformations and an analysis of distinguishing time series dynamics with the causality complexity-entropy plane.

3.6.1 Interpretation of the resulting transformations

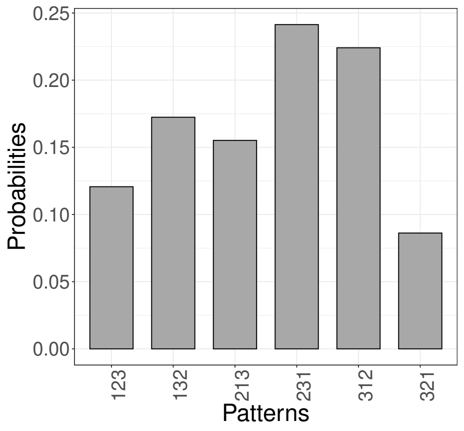
After the aforementioned ordinal patterns transformations, it is possible to identify different aspects on the resulting representations, which are related to the dynamics of their underlying time series. Figure 3.9 illustrates these transformations with examples of time series from the well-known Synthetic Control[3] dataset [Alcock et al., 1999]. This dataset is composed by 600 synthetic generated time series, divided in six different classes, each of them following a distinct control chart patterns: normal, cyclic, increasing trend, decreasing trend, upward shift, and downward shift [Alcock et al., 1999]. Each time series has a length of 60 data points, and all values normalized between $[-2, 2]$. Figure 3.9a illustrates two time series following the normal dynamic (class 1), which is constructed by random generated numbers, and the cyclic dynamic (class 2), which are random numbers plus a sinusoidal signal.

Although these time series were both constructed from the same random dynamics, the time series of class 2 represents a distinct dynamics by adding a deterministic sinusoidal component, composing a new mixture dynamics. Figures 3.9b-e illustrate the ordinal patterns probability distributions and transition graphs from these two time series, constructed with an embedding dimension $D = 3$ and an embedding delay

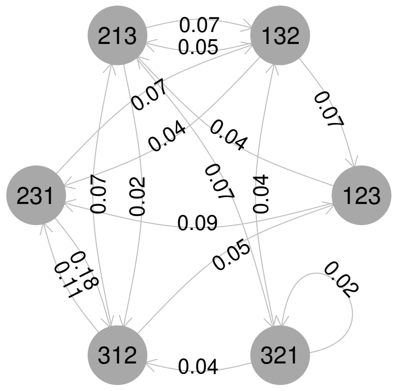
[3]The Synthetic Control dataset is part of the UCR time series classification archive [Chen et al., 2015; Dau et al., 2018], it is available at `http://timeseriesclassification.com/description.php?Dataset=SyntheticControl`.



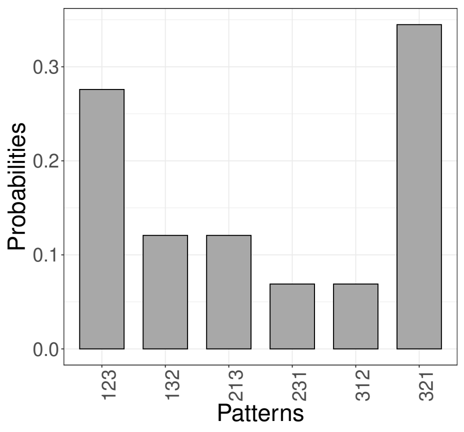
(a)



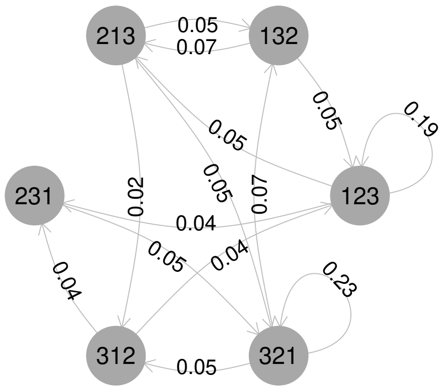
(b)



(c)



(d)



(e)

Figure 3.9. Illustration of (a) two time series samples from the (1) normal class and (2) cyclic class of the Synthetic Control dataset [Alcock et al., 1999], and their ordinal patterns transformations: the ordinal patterns (b) probability distribution and (c) transition graph of the class 1, and (d) probability distribution and (e) transition graph of the class 2. Both transformations were constructed with embedding dimension $D = 3$ and embedding delay $\tau = 1$.

$\tau = 1$. For the normal time series, by following a pure random behavior, we can observe a more uniform distribution among the $D!$ patterns, which can be seen in Figure 3.9b, than the distribution of patterns from the cyclic time series, in Figure 3.9d. It is important to remember the condition $n \gg D!$, which establishes that the time series length n must be long enough, so all patterns have the opportunity to occur, and the behavior be better identified [Rosso et al., 2012]. However, even for $n = 60$ in this dataset, we can see the most deterministic behavior from the cyclic time series results in a more concentration of ascending and descending patterns, 123 and 321, respectively.

With respect to the ordinal patterns transition graphs for the normal and cyclic classes, Figures 3.9c and 3.9e, respectively, the same behavior can be observed in the edges weights of the resulting graphs. While for the normal random class the edges weights are uniformly distributed among the transitions between vertices, for the cyclic class the transitions are more concentrated between the most frequent patterns. Thus, the weights of the loop edges between 123 and 321 patterns are higher than for the other transitions. Another distinction between the two graphs are the number of transitions between patterns. Given the concentration of ascending and descending transitions, the rest of transitions are less frequent and, consequently, the transitions among them.

3.6.2 Causality Complexity-Entropy Plane

Both normalized Shannon entropy and statistical complexity measures are quantifiers from information theory that are suitable to extract knowledge regarding the dynamic behavior of a given process. With the normalized Shannon entropy it is possible to measure the amount of uncertainty one may have from the process, ranging from the certain prediction of the possible values to the maximum uncertainty (uniform distribution) [Lamberti et al., 2004]. The statistical complexity measure permits to reveal some details of the dynamics of the process, discerning among different degrees of periodicity and chaos [Rosso et al., 2007a].

By the application of those measures to evaluate the associate ordinal pattern probability distribution from a given time series it is possible to extract characteristics from their dynamics and distinguish between noisy time series and those with some sort of dynamics. Rosso et al. [2007a] proposed the utilization of these measures from the ordinal patterns distribution (p_π) to distinguish among different behaviors of time series. For this, they proposed the Causality Complexity-Entropy Plane (CCEP), that is a 2-dimensional metric space built by the statistical complexity $C_{JS}[p_\pi]$ as the y-axis and the normalized Shannon entropy $H_S[p_\pi]$ as the x-axis.

An important characteristic of the CCEP is the localization of regions in the plane where reside time series with specific dynamical behaviors. For instance, as pointed out by the authors, the method permits distinction between deterministic, stochastic, and chaotic time series, as well as the placement for different noisy time series. The authors also showed that, for different values of H_S and order D, the statistical complexity measure ranges between a minimum and maximum values (cf. Rosso et al. [2007a]).

More details and examples of the CCEP for different time series dynamics is presented in Section 5.2.2.

3.7 Conclusions

This chapter presented introductory concepts for time series, discussing some models for better describing the challenges presented in the data of CoIoT scenarios. We also presented some classical time series representations, which are obtained by the transformation of the time ordered data onto a novel domain. By following a discussion of the best time series representation for CoIoT, the chosen ordinal pattern transformations were presented. We showed how this novel representation can be used for capturing time series dynamics and suggested a set of features that can be used for its numerical analysis. Finally, some issues and limitations of the ordinal patterns representations are presented, leading to our contributions towards their mitigation.

Chapter 4

Sensing in the Collaborative Internet of Things

The CoIoT is helping with a considerable increasing in the amount of IoT data, but also bringing several new challenges. An important challenge is to search for and select a correct and reliable sensor when there is no such description or when it is imprecise. In this chapter, we present a strategy for dealing with this issue of searching for a specific sensor in the IoT. In Section 4.1 we present a contextualization that motivates our work. In Section 4.2, we briefly discuss the main efforts in sensor search and selection strategies. In Section 4.3, we investigate the properties of sensed data in the collaborative IoT, and in Section 4.4 we present our proposed strategies to search for a time series within the CoIoT scenario, based on the query by content concept, which deals with its data uncertainties and challenges. Section 4.5 presents our experiments with real sensed data, in order to evaluate the feasibility and limits of our approach. Finally, in Section 4.6, we present our conclusions and future research directions.

4.1 Introduction

Internet of Things (IoT) is much about exchanging and analyzing a large amount of unstructured and heterogeneous data, collected from different sources [Miorandi et al., 2012; Gubbi et al., 2013]. Unstructured and heterogeneous sensed data have different characteristics and properties, which can be used to classify the collected data into two broad groups: data that have a reliable service to represent its “reference data”, and data that do not have. Examples of the former group are temperature, pressure, humidity, wind speed and UV radiation that have their corresponding “reference” values available at different Websites. Examples of the latter group are fine particulate matter, sound pressure level and concentrations of both organic compounds and metals in sediment and tissue that are not readily available. Notice that once IoT devices become more and more common, variables present in the latter group will start appearing in the former one. In this chapter, we only consider variables belonging to the first group that have reference values. Actually, the group with reference values represent variables that typically the sensor technology is widely available due to its advances/cost or importance for the society/particular users.

Regardless of the group, IoT devices are entities acting as providers and/or consumers of data related to the physical world, with different computing capabilities and resource limitations, ranging from small sensors to more powerful and complex systems. Notice that IoT encompasses different technologies such as wireless sensor networks, RFID tags and all sort of devices with embedded computing capability. However, our focus is on the data obtained from those entities, rather than on the physical systems aspects such as how the data was collected, processed and transmitted. From now on, we opted to adopt the term "sensor" for those devices/things, as it can also be found in the well-established literature of wireless sensor networks [Nakamura et al., 2007].

To deal properly with such massive data, a key feature is to have IoT services to support automated reasoning, which heavily depends on a standardized format and model for the collected data, *i.e.*, a semantic description of its content and metadata [Miorandi et al., 2012]. However, as presented in Section 2.2, the Collaborative Internet of Things (CoIoT) is a special case of IoT where most of their sensors will be deployed by different owners, generally common users, and for different purposes. In this scenario, the collaborative aspect comes from the fact that users make their data freely available by setting them as public. As a consequence, data privacy is not a concern, which might not be a problem for the kind of data at hand, whereas for other applications, security issues related to the IoT domain are typically a serious issue [Sicari et al., 2015].

Thus, while the CoIoT is helping to increase considerably the amount of IoT data, it also brings several new challenges, being one of them the data quality. For the important challenge of searching and selecting a correct and reliable sensor in IoT, the main research questions that arises are:

1. How reliable are the sensed data available in IoT and which aspects impact their accuracy?

2. Is it possible to combine sensed data from different sensors in order to improve the data quality?

In this chapter, we characterize the properties of real sensed data in the IoT, potentially contributed by different sources, including sensors from general users, *i.e.*, the CoIoT. We show that, in order to safely use data available in the IoT, it is necessary a filtering process to increase their reliability. In this direction, we propose a new simple and powerful approach that helps to select reliable sensors, exploring collaborative filtering [Adomavicius and Tuzhilin, 2005]. We tested our method for different types of sensed data and the results show that the proposed method is very effective to select reliable sensors, regardless if they are part of a controlled set of sensors or not.

4.2 Related work

There are at least two aspects to consider when we study sensing in the IoT environments. The first one is a proper sensor search and selection strategy to ensure the desired sensors are found. The second one deals with the quality of these selected sensors, since the behavior of a system depends completely on the observations it infers about the environment, which is based on the sensed data.

4.2.1 Solutions for Sensor Search and Selection

The first aspect that may affect the quality of the sensing is how to correctly match the expected data to the actually collected one. The description of the services plays a key role to this, since the richer the service description is, the better its matching will be [Rong and Liu, 2010] and, thus, it allows us to better interpret and understand the service itself [Perera et al., 2014a]. To address this problem, the solutions should begin by a proper description of the service functionalities [Miorandi et al., 2012].

As established by Mian et al. [2009], service description is an abstraction of a service's functionalities and characteristics. A simple and unambiguous method for describing services is to use the Universal Description, Discovery and Integration (UUID) [Leach et al., 2005]. However, even this method, that results in a practical unique service identifier, has no relation to the service itself, *i.e.*, it cannot capture the service details in order to help users to choose the service according to its specificities, nor the users will be able to compare two different services.

Zhang et al. [2010] point out that a proper service description is not enough to suggest the best service to the users' needs. Instead, to handle the characteristics of IoT services, it is indispensable that the consumer be able to perceive the current state of the service it will require. In other words, the service consumer should follow the principles of context awareness [Dey et al., 1999] and be able to intelligently identify and autonomously adapt to service changes. Thus, in addition of being aware of the state of its users, an IoT-based system should also be able to take into account the dynamics of the IoT services they are based on.

To address this question, Perera et al. [2014b] propose an ontology-based context framework to allow a comprehensive sensor search and selection functionality that best suits the user requirements. Once the system knows the user requirements and priorities, it will be able to search for the appropriated sensors, rank them according to the described preferences and select the number of sensors the user defines.

On the other hand, the service discovery, which fully depends on the service description, still needs a special attention. Ververidis and Polyzos [2008] define service discovery as a process that enables network entities to advertise their services, query about services provided by other entities, select the most appropriate matched services and invoke them. The goal of that work is to facilitate the use of the IoT platform by automating the service operation and making a seamless integration between the physical and the informational worlds [Wei and Jin, 2012].

There are several proposals to deal with the service discovery challenge in the Internet domain. For the Web service discovery, the UDDI also provides a directory service where entities can register and search for services [Clement et al., 2004]. The WS-Discovery solution [Modi and Kemp, 2009] deals with the Web service discovery tasks. Other examples of service discovery are the Apache River [Apache River Community, 2018], the advancement of the Sun's Jini, Universal Plug and Play (UPnP) protocol [UPnP Forum, 2011], Service Location Protocol (SLP) [Guttman et al., 1999] and the Bluetooth Service Discovery Protocol [Bluetooth SIG, 2010]. However, in the IoT scenario, there are more challenges to be solved than simply applying those strategies, as discussed above.

Wei and Jin [2012] also state that context-aware service discovery approaches should be used to provide, besides autonomous adaptation, the ability to deal with the uncertainties and temporal contexts of IoT. Rambold et al. [2009] highlight that an autonomic service discovery approach should consider the possibility of monitoring a service and, in case it is not available (because of a service failure, for instance) or its description is outdated, to discover a service replacement. Rong and Liu [2010] present a survey and classification of the main discovery approaches for context-aware Web services.

4.2.2 Solutions for the Quality of Sensing in IoT

Besides the previously mentioned efforts about strategies for searching and selecting one or more sensors, considering specific contextual information from them, the current state of the art of a collaborative IoT environment is still far from providing complete information available. For instance, the most common pieces of information available from a sensor platform and middleware for an IoT application are the *sensor location*, *sensor type*, *keywords* and the data they generate. Some examples of these platforms are the Linked Sensor Middleware (LSM) [Phuoc et al., 2011], the Global Sensor Networks (GSN) [Aberer et al., 2007], the Microsoft SensorMap [Nath et al., 2007], the Thingspeak platform, among others.

To properly search for and select a desired sensor, the designer of an IoT system must consider those pieces of information and the collected data itself, including its quality. This means that, once a sensed data is given, it will be necessary to analyze its behavior and detect if this is the desired data, *i.e.*, if this data is similar to the expected data generated by a correct sensor (reference value). This can be useful for both searching for a desired sensor and determining which sensor to select when a set of sensors is available.

In general, the sensed data will be used in context-aware systems [Dey et al., 1999], and, thus, the quality of the sensed data can be compared to the Quality of Context (QoC) concept. Buchholz et al. [2003] define QoC as any information that describes the quality of information used as context information. They also establish the differences between QoC and Quality of Service (QoS), which refers to any information that describes how well a service performs, and Quality of Device (QoD), which means any information about the devices' technical properties and capabilities. According to those definitions, they propose the following QoC parameters to measure the quality of a context[Buchholz et al., 2003]: *precision*, *probability of correctness*, *trustworthiness*, *resolution* and *up-to-dateness*.

The *precision* of the sensed data is related to how the sensed data reflects the current reality of the physical phenomenon. *Probability of correctness* denotes the probability that a given data is correct. *Trustworthiness* is similar to the probability of correctness, but it is used to rate the quality of the entity which generated the data. *Resolution* denotes the information granularity, and *up-to-dateness* describes the age of the data.

Li et al. [2012] extend the QoC definition for pervasive environments. They investigate the challenges in providing data quality in these environments and propose three metrics to quantitatively observe the quality of these data and their data sources:

currency, *availability* and *validity*. *Currency* is related to the up-to-dateness metric above, but it represents the temporal utility of the data, from the moment it is created until is useless. The *availability* is more concerned about the capability of an entity to generate data. Finally, the *validity* is defined by a set of rules that can be used to validate the sensed data according to the previous knowledge about the expected properties of the data.

Despite the fact that all these metrics asses data quality, the current collaborative IoT environments suffer from a lack of information about their services, which can impact the correct analysis of their data and sources. In general, there is neither a precise description about the data a sensor generates nor the functionalities they provide, nor a metadata about the state of these sensors. Furthermore, we can not expect to have a different scenario in the short term since this would probably call for a standardization effort in that case. Thus, it is necessary to investigate this current scenario and analyze possible solutions to improve the reliability of IoT services to be deployed.

In that direction, Sicari et al. [2014] propose an architecture to ensure both the security and the quality of IoT data. The main aspect in that work related to our discussion is that the collected data is first processed and analyzed to extract relevant information about it (quantitatively) and from its sources. The resulting information becomes its metadata.

In our work, we consider a previous step before employing the sensed data, since in the collaborative IoT we can not rely on the information obtained from sensors and their services. More specifically, we study the worst case scenario, where no information about the data exists at all, and discuss a method to select the appropriate data to be used. We perform our analysis and discuss our assumptions based on properties and behavior we observe from the real data itself.

4.3 Characterization of the CoIoT data

The study of the properties of the sensed data in CoIoT may be very useful to define strategies to deal with sensor data uncertainties. To investigate that, we collected and analyzed public data from real sensors deployed in a city-wide area. In this section we study the properties of that sensed data and present two algorithms to improve their reliability, establishing a proper sensor selection and refinement of the sensed data.

In the following analysis, we consider two datasets, both collected from a region around London, UK, extracted from a collaborative IoT. The first dataset includes only temperature sensors, *i.e.*, sensors in which the temperature tag was included in their descriptions. Note that this procedure may not consider some temperature sensors whose description does not include this tag. The second dataset includes all collected sensors, without any sort of filtering. We were able to collect data for temperature, humidity, pressure, wind speed, and more specific sensors, such as binary sensors that indicate the turn on/off of lights and watts consumption.

Those datasets will allow us to assess the behavior of the sensed data in a controlled environment and, then, relax this criterion to include different types of data. The parameters of these collections are detailed in Table 4.1. We also collected reference values from conventional weather forecast services. This reference was used to select appropriate temperature sensors (Dataset 1). For this particular case, we were interested in outdoor temperature data. To select those sensors, we compared data generated by sensors and data generated by the weather forecast service (reference service) computing the correlation coefficient among them.

The selection of a reference service concerns the reliability we should have on the data generated by that particular source. Obviously, to determine how trustworthy a service is, it is necessary a previous knowledge about both the sensing phenomenon itself and the corresponding reference values. For our study, it is easy to find a reliable reference service for our data type. However, for other data types, we may use the reputation of the related service to assess its reliability. In this sense, some criteria already discussed here can be used in this assessment, such as the precision of data generated, probability of correctness, trustworthiness and availability [Dey et al., 1999].

Table 4.1. Parameters for gathering data. Samples were collected from the region near London, UK, at different moments. The common parameters describe the intervals used for the data analysis.

Common Parameters	
Location	**London, UK**
Coordinates	lat. = 51.53, lon. = -0.10
Collection interval (min)	5
Range intervals (distance from coordinates in km)	10–1000
Reference weather service	Forecast.io
Dataset 1	
From	08 May 2014 14:05
To	12 May 2014 08:35
Total of samples	$\approx$ 1070
Dataset 2	
From	07 October 2014 18:55
To	13 October 2014 12:40
Total of samples	$\approx$ 1500

We considered the following five options of weather forecast services as our reference: Forecast.io[1], Open Weather[2], Weather Underground[3], World Weather Online[4], Yahoo Weather[5]. As shown in Figure 4.1, all weather forecast services were similar in the period of the analysis. We have chosen the Forecast.io service as our reference data, since it is close to the average merged sample, with a Mean Squared Error (MSE) of 0.0685161 for Dataset 1, and 0.1571421 for Dataset 2.

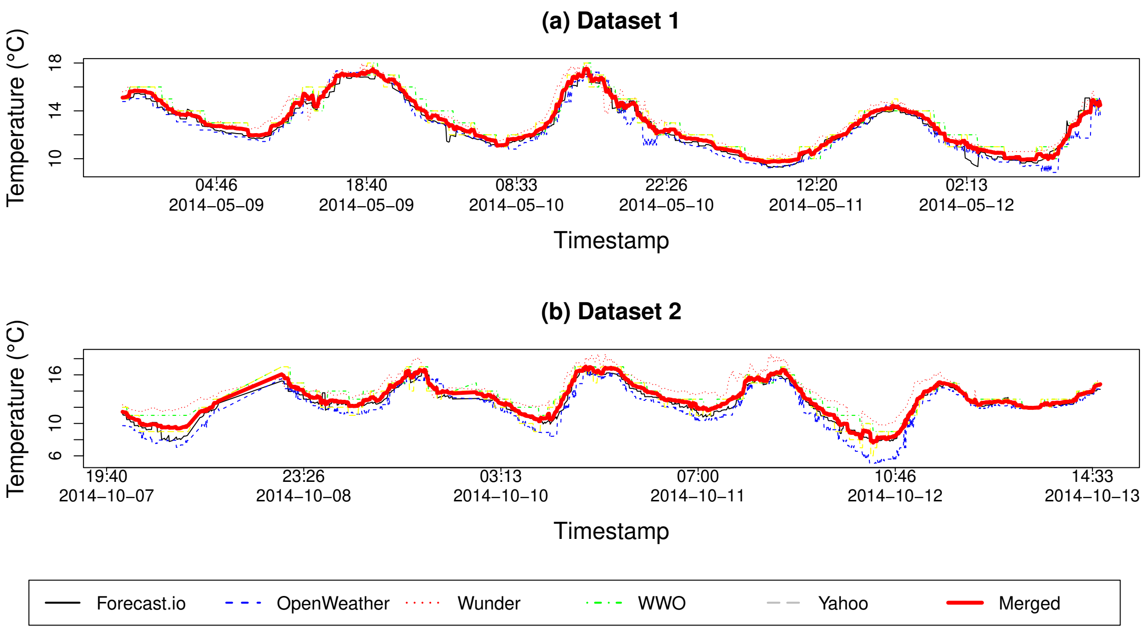


Figure 4.1. Comparison between five possible reference services and the average merged sample, for the (a) Dataset 1 and (b) Datasets 2.

Notice that IoT scenarios will comprise a massive amount of unstructured and heterogeneous data, leading to different data classes, which have their own characteristics, boundaries, trends and patterns. As already discussed, in this work, we only consider variables belonging to the group that has reference values. The process of suggesting a reference value for variables that do not have a reference data service promptly available is still an open issue and will be addressed in the future work.

In this work, we analyze the data class with a reference service by taking as our case study a single and controlled variable: weather data. This data class has been studied at a great extent in the literature of sensor networks. Furthermore, it exemplifies the sort of sensing data we can find several reliable weather forecast services

[1]Forecast.io weather service is now Dark Sky, it's available at `https://darksky.net/forecast`.
[2]OpenWeather service is available at `https://openweathermap.org`.
[3]Weather Undergound is available at `https://www.wunderground.com`.
[4]World Weather Online is available at `https://www.worldweatheronline.com`.
[5]Yahoo Weather is available at `https://www.yahoo.com/news/weather`.

to represent our reference data. For different and more specific data classes, it is not possible yet to obtain a reference as simple as this. For those cases, we need a better understanding about the corresponding classes and their features.

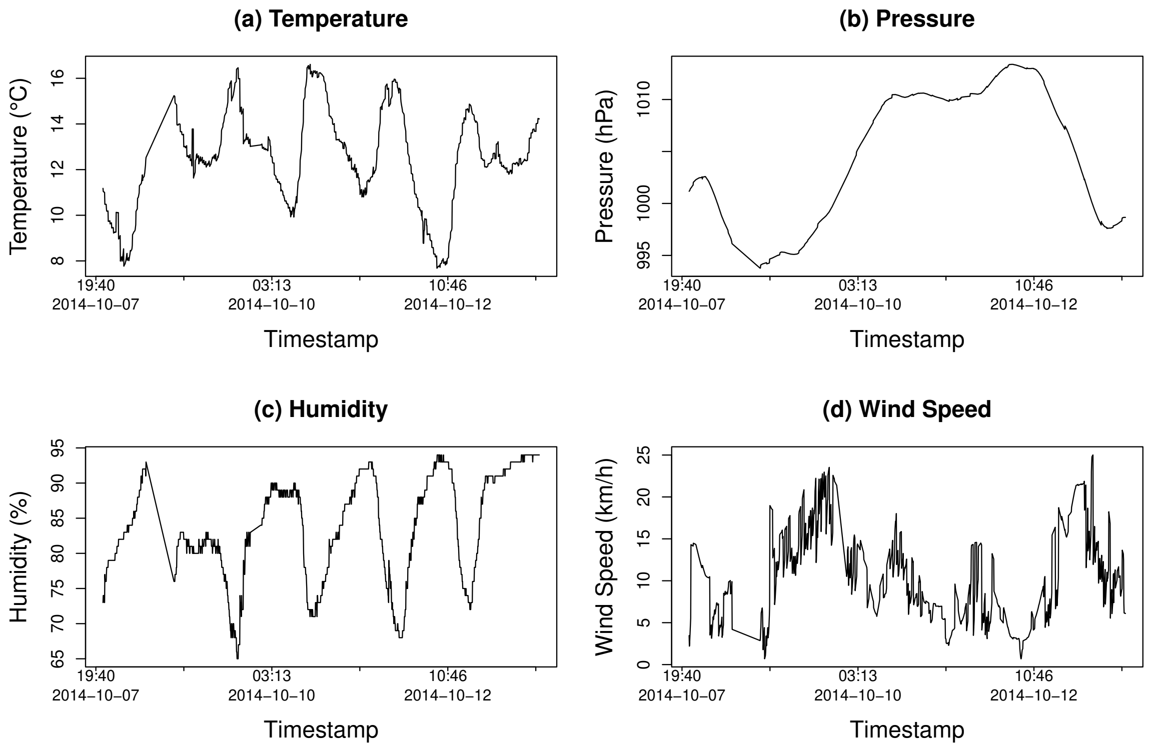


Figure 4.2. Reference data samples for different types of data (a) temperature, (b) pressure, (c) humidity and (d) wind speed (Dataset 2).

We also collected four different types of data from Forecast.io weather forecast service: temperature, pressure, humidity and wind speed. Figure 4.2 illustrates the time series behavior of the obtained samples.

4.4 Sensor search and selection in IoT

Based on this characterization of CoIoT data, we designed algorithms for creating sensing services in the current collaborative IoT environments, establishing two directions:

A) Sensor search and selection, and

B) Sensor refinement.

In the following sections we give more details on these strategies.

4.4.1 Sensor Search and Selection

To search for a real sensor data, we rely on the public available data from Xively[6], a popular collaborative IoT platform. Through that Web service, users can make available their sensor readings to the cloud, directly from the embedded devices, collect data using a common API, and read and manage data from a remote application or a Web browser. In the Xively platform, the sensor data can be grouped into blocks called feeds, each one having the following attributes: feed ID, tags, status, description, location (e.g., name of city or coordinates) and data streams. A data stream is the sensed data obtained from a sensor and is described by its data stream ID, tags, current value, min and max values and the unit of the data generated. In some cases, not all of those pieces of information are filled by the sensor owners or, when they are provided, they are not precise.

Searching for the desired sensors using the Xively REST API is simple and can be accomplished through a Web browser, such as the following example:

```
http://api.xively.com/v2/feeds.json?
    key=XIVELY_API_KEY&lat=51.53&lon=-0.10&
    q=temperature&distance=50&status=live
```

In this example, we are searching for sensors located in a range of 50 km from the coordinates of London, UK, that have in their description the tag “temperature” and in which the status is live, meaning the sensor is actually generating data. As we can see, the Xively query above is very simple and only contains a tag and coordinates. This means that the sensed data can be different from what we are looking for. Besides, the result does not include context information about the data themselves, which can lead to a wrong sensor selection, impacting the quality of the sensed data.

For example, Figure 4.3 shows different sensor data collected for temperature. While the blue dashed line (Sensor 1) means a sensor reading for an outdoor temperature, the red dotted line (Sensor 2) corresponds to a sensor reading for the temperature of a water tank. For this specific case, the data owner filled the description for the temperature data, but this is not the general case. The black solid line represents our reference data. This exemplifies the need for a specific sensor selection strategy for this kind of sensed data.

In our case, we want to select sensors measuring temperature, located at a given range from a given coordinate, but that their temperature corresponds to similar conditions of the outdoor environment. Since there is no way to specify this requirement in

[6]The Xively IoT platform is now owned by Google, and its services is subject to new conditions. The new service is available at `https://cloud.google.com/solutions/iot`.

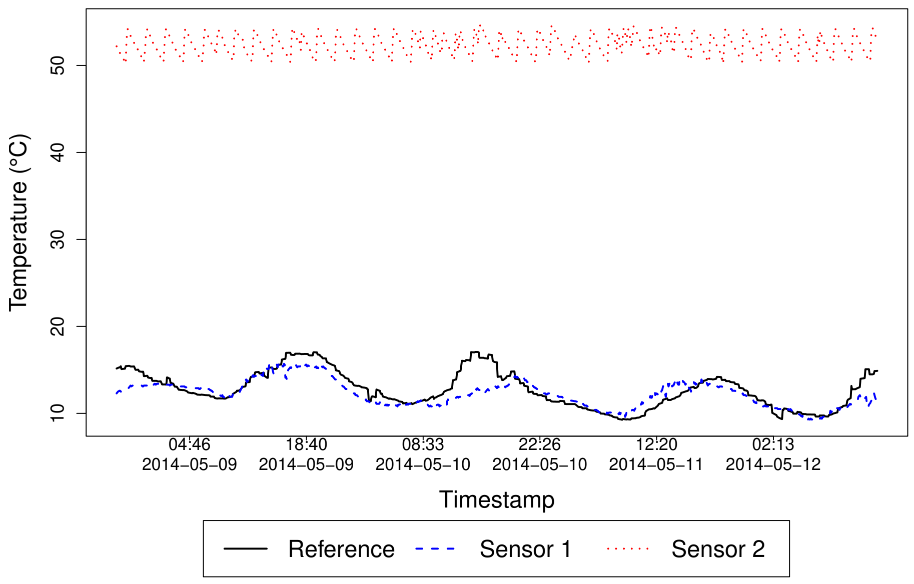


Figure 4.3. Different sensor readings for the "temperature" tag, but with different meanings (Dataset 1).

the search parameters, the application must infer, according to the data value, whether it could be generated by a reliable sensor or not.

It is clear the difference between reliable and unreliable sensor data. A reliable sensor generates data more correlated to the reference data, and can be a trusted source of the temperature, in our example. Thus, if the reference corresponds to the temperature of the outdoor environment, a trusted sensor data will be the one that is somehow influenced by the environment temperature variation. This influence is measured by behavior similarities among the two sensed data.

For our first analysis, we consider the statistic measure of the Pearson's sample correlation coefficient (r) between the reference data and the collected data from the sensors. The formula for r is described in Equation 4.1,

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}}, \tag{4.1}$$

where X_i is the ith value of the reference data, Y_i is the ith value of the sensor data to compare, and $\bar{X}$ and $\bar{Y}$ are their sample mean, respectively. The value of r can vary between $[-1, 1]$, and the more correlated two samples are, the closer to one is $|r|$.

For the example of Figure 4.3, the correlation coefficient between the outdoor Sensor 1 and the reference data is $r = 0.8$, and for the particular water tank temperature Sensor 2, $r = -0.04$, which can indicate that this is a good metric to filter trusted sensors in a given set. Figure 4.4 depicts a scatter plot that summarizes the correlations between the reference data (x-axis) and the cited sensors.

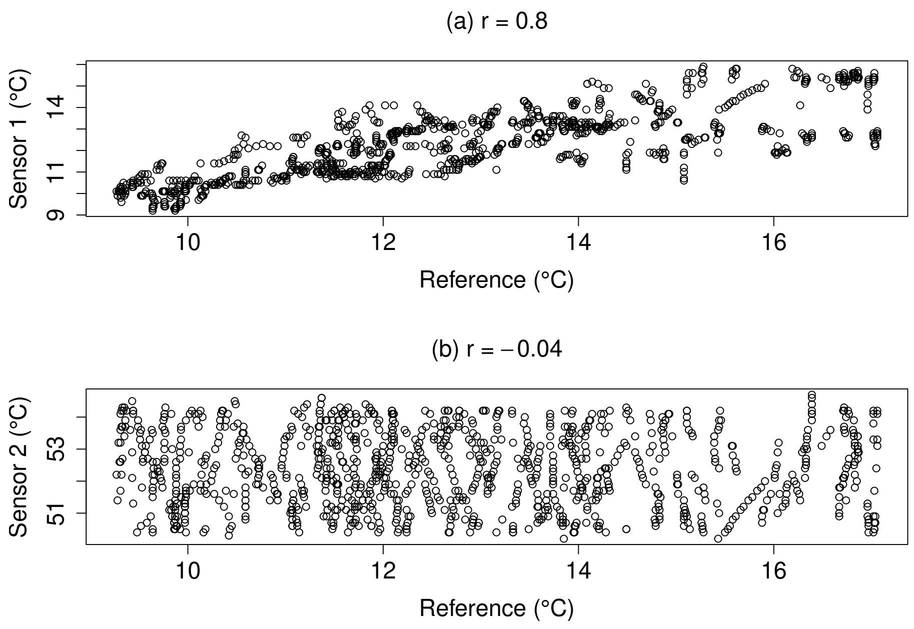


Figure 4.4. Scatter plot between the reference data (x-axis) and (**a**) the correlated Sensor 1; and (**b**) the uncorrelated Sensor 2 (Dataset 1).

However, the correlation coefficient is not the only parameter to indicate whether a sensor is reliable or not. For example, consider Figure 4.5, which represents the analysis of the correlation coefficient between two naturally uncorrelated data samples, our temperature reference data and a carbon monoxide (CO) sensor, from Dataset 2. Since the correlation coefficient r does not consider the scale of the compared data (see Equation 4.1), even if the magnitude of the samples are very apart, the correct values of r will depend on the number of considered samples. For the case of Figure 4.5b, we can observe the increasing of r up to 0.9255456, when we have only 200 samples. After that point, the value of r tends to decrease, and with 892 samples we have $r = 0.3539141$. Thus, if we consider a sensor to be reliable, in our case we would need a correlation value $r = 0.8$. To assume that the sensor is unreliable, we would need at least 224 samples.

Notice that we can not depend on the amount of samples we have, as stated before in Section 4.2.2, and, thus, we need to include in the sensor selection strategy more knowledge about the data we are dealing with. As proposed by Li et al. [2012], we also consider a set of validity rules $\mathcal{V} = \{v_i, \ldots, v_l\}$ sensors must satisfy to be considered reliable. The data validity is a metric based on the estimation that the observation, according to the specificities of each problem, does not deviate from an acceptable range, in comparison to a known and expected behavior for that data. For example, in London, UK, the average maximum temperature between June and September is around 23 °C whereas the minimum is 11 °C.

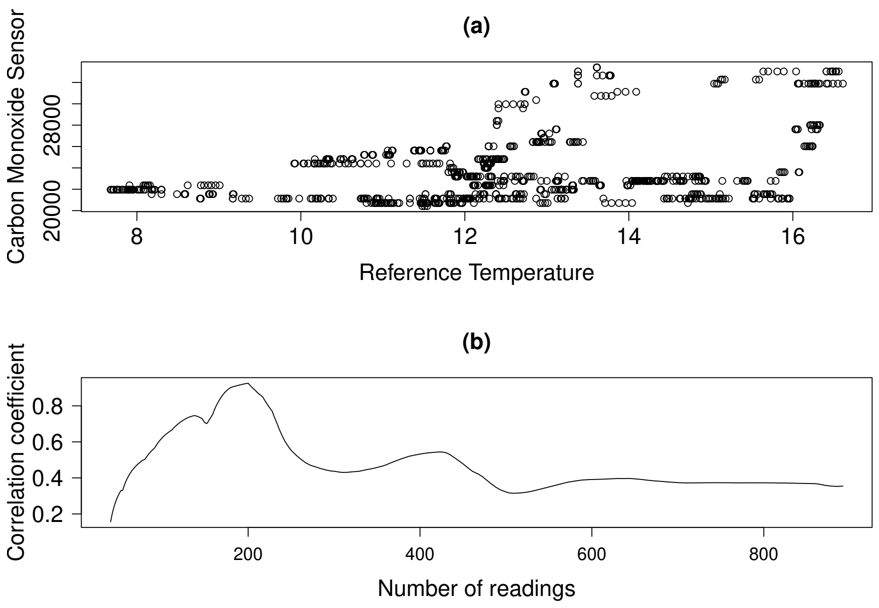


Figure 4.5. (**a**) Scatter plot between a temperature and a carbon monoxide sensor; and (**b**) the analysis of its correlation coefficient when the number of samples increases (Dataset 2).

Algorithm 4 summarizes this sensor selection strategy considering the aforementioned issues. It is important to emphasize that the readings must be adjusted, since, in a given time, a sensor might fail sending its readings, and this gap can be a problem in the computation. In the algorithm, we adjust this by computing new vectors $\mathbf{s}^*$ and $\mathbf{r}^*$ in which their values correspond to the entries that occur in both sensor and reference readings, respectively, *i.e.*, there are no null values (λ) for both vectors.

4.4.2 Sensor Refinement

The correlation coefficient and the validity rules seem to be reasonable metrics to be used as a first strategy for sensor selection, to filter the sensors that best fit our purposes. However, in order to safely use all data from these sensors, we still need to consider some issues. For instance, consider the example of Figure 4.6, from Dataset 1, which illustrates the time series and scatter plots between two sensed data and a reference data. Even with a relatively high correlation between the sensors and the reference, $r = 0.777395$ for Sensor 1 and $r = 0.864393$ for Sensor 2, the reading values given by the sensors are systematically higher than that provided by the reference. For the case of Sensor 1, apparently the temperature values follow the increase of the

ALGORITHM 4: SensorSelection

```
Input: $\mathbf{S}_{n\times m}$ matrix of time series for $n$ sensors with $m$ samples each, a vector
       $\mathbf{r} = \{r_1, \dots, r_m\}$ of a reference time series, a correlation coefficient limit
       $\rho \in [-1, 1]$, and a set of validity rules $\mathcal{V} = \{\upsilon_1, \dots, \upsilon_l\}$.
Output: A set $\mathcal{R}$ of reliable sensors.
$n \leftarrow$ rows($\mathbf{S}$);
$m \leftarrow$ cols($\mathbf{S}$);
$\mathcal{R} \leftarrow \emptyset$;
// For each sensor entry in the time series matrix $\mathbf{S}$.
for $i \leftarrow 1$ to $n$ do
    // Adjust the valid readings among the reference and the sensor.
    $\mathbf{s}^* \leftarrow \{\ s_{i,j} \mid (s_{i,j} \neq \lambda)$ and $(r_j \neq \lambda), 1 \leq j \leq m\ \}$;
    $\mathbf{r}^* \leftarrow \{\ r_j \mid (s_{i,j} \neq \lambda)$ and $(r_j \neq \lambda), 1 \leq j \leq m\ \}$;
    // Compute the correlation coefficient between $\mathbf{s}^*$ and $\mathbf{r}^*$.
    $\rho^* \leftarrow \mathrm{cor}(\mathbf{s}^*, \mathbf{r}^*)$;
    // The correlation satisfies the given limit?
    if $\rho^* \geq \rho$ ;
     then
        // The sensor entry satisfies all validity rules?
        if $\bigwedge_{k=1}^{l} \upsilon_k(\mathbf{s}^*)$ ;
         then
            $\mathcal{R} \leftarrow \mathcal{R} \cup i$;
        end
    end
end
return $\mathcal{R}$
```

environment temperature, but present a slower response to its decrease. However, for Sensor 2 readings, we can see that, while the measured temperature values are related to the reference, *i.e.*, there is an influence of the environment temperature in this sensor, all its values are an order of magnitude higher than the reference values.

In Figure 4.6, when we analyze the temperature measured by Sensor 2, we can argue that the given metrics can not ensure the reliability of this sensor data. Note that the correlation coefficient (≈ 0.8 for this case) can represent an influence of the environment temperature on the sensor, and can be considered acceptable for the validity rules at the period of the reading. However, it gives no more information about the correctness of the data, and how reliable the sensor is. We can not make sure whether the temperature is really higher at that location or the data is coming from an indoor sensor, which would explain that value.

On the other hand, when we look at the bigger picture, we can observe some interesting patterns that can be used to handle this uncertainty. Figure 4.7a shows the sensors and their data streams found in Dataset 1. As we increase the range (distance

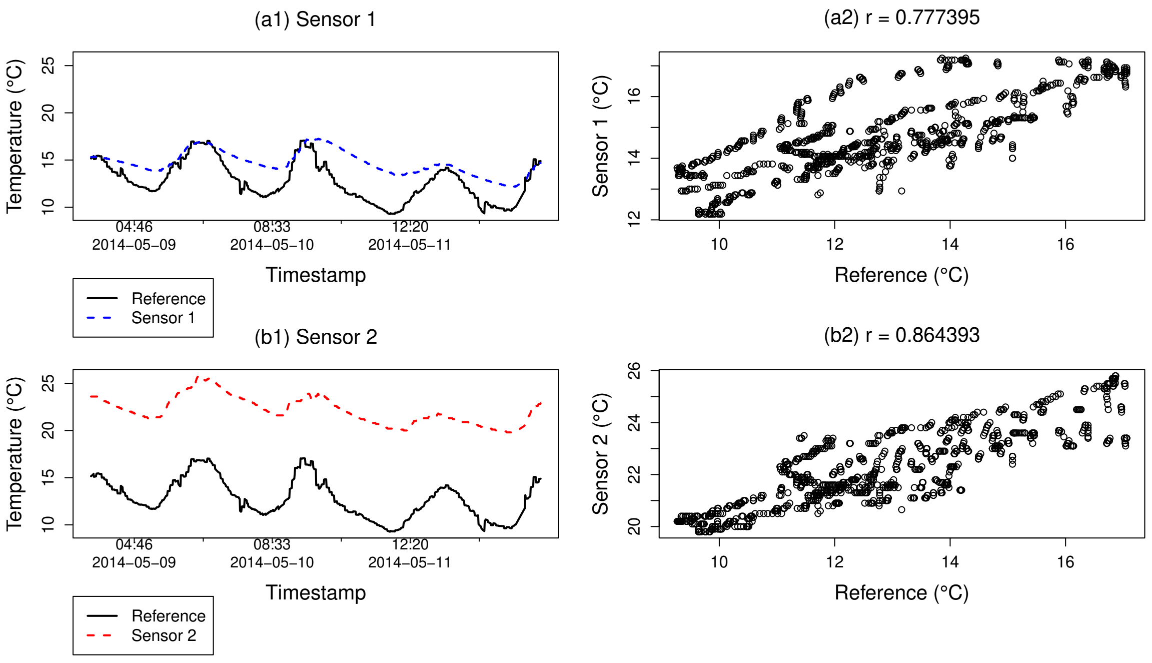
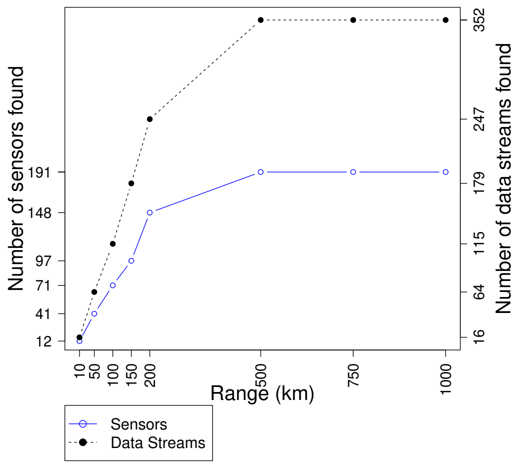


Figure 4.6. Time series (**a1**) and scatter plot (**a2**) analysis for a $r = 0.777395$ related sensor, and time series (**b1**) and scatter plot (**b2**) analysis for a $r = 0.864393$ sensor (Dataset 1).

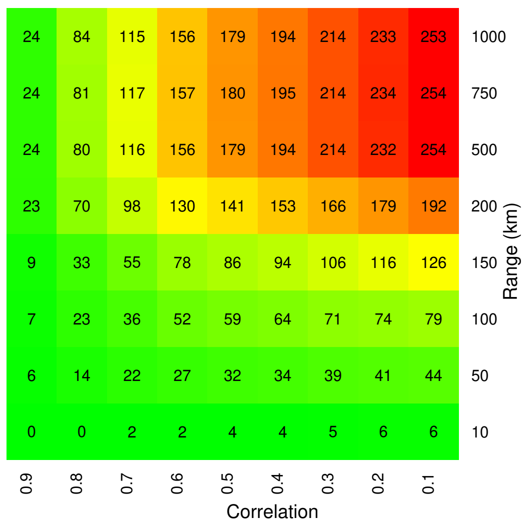
in kilometers from the central coordinate point), the number of available sensors (feeds) and their data streams also increase. On average, a sensor provides approximately two data streams (1.84). Applying the sample correlation coefficient r as a metric to filter the considered trusted data, we have the results showed in the heat map of Figure 4.7b, which represents the selected data streams when varying the range and the correlation coefficient limit l. From the set of all data streams in a given range, we selected those which give $r \geq l$ in comparison with the reference sensor data.

If we take those sensors considered reliable within a range of 50 km and the correlation coefficient $r \geq 0.8$, we have 14 sensors to handle. Figure 4.8a illustrates a time series analysis of those sensors. Given the 14 reliable sensors, which we assumed to be influenced by the environment temperature, ten of them are close to the reference data, and four are dislocated by a given value Δ_i, for each sensor i, considering the value of the reference sensor.

Given a set $\mathcal{R} = \{r_1, r_2, \ldots, r_n\}$ of reliable sensors, let $\mathcal{D} \subseteq \mathcal{R}$ be the subset of sensors that are dislocated from the reference values, which we assume to be the correct temperature value for that region at a given time t. Let us assume to be constant the amount Δ_i a sensor r_i is dislocated from the reference sensor throughout time t. Thus,



(a)



(b)

Figure 4.7. Analysis of the number of sensors found in Dataset 1. (**a**) Number of sensors and their data streams found for different sensor search ranges (Dataset 1); (**b**) Heat map for the number of data streams found when varying the range of the desired location area and the correlation coefficient limit (Dataset 1).

the time series $y_{i,t}$ for each sensor r_i can be defined as

$$y_{i,t} = \theta_t + \Delta_i I_{[r_i \in \mathcal{D}]} + \varepsilon_t, \tag{4.2}$$

where θ_t is the reference sensor value at time t, $\varepsilon_t \sim N(0, \sigma^2)$ is a Gaussian random variable that represents an i.i.d. (independent and identically distributed) error with expected value 0 and variance σ^2 at t, and $I_{[r_i \in \mathcal{D}]}$ is a binary random variable, where $I_{[r_i \in \mathcal{D}]} = 1$ if the sensor r_i belongs to the set $\mathcal{D}$, or $I_{[r_i \in \mathcal{D}]} = 0$, otherwise.

Based on that, we can define an estimation $\hat{\theta}_t$ of the value of the reference θ_t, using Equation 4.2, as

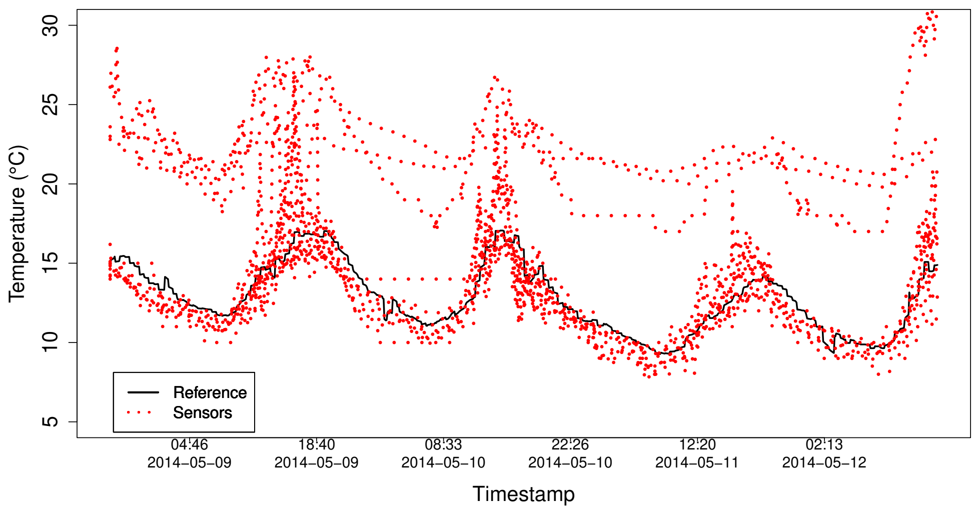
$$\hat{\theta}_t = \frac{1}{n} \sum_{k=1}^{n} \left(y_{i,t} - \hat{\Delta}_i \hat{I}_{[r_i \in \mathcal{D}]} \right). \tag{4.3}$$

To estimate $\hat{I}_{[r_i \in \mathcal{D}]}$, we compute a pairwise-distance matrix $\mathbf{P}_{n \times n}$ for the time series of the correlated sensors, where each element $p_{i,j} 1 \leq i, j \leq n, i \neq j$, corresponds to the average squared distance between the two time series of sensors r_i and r_j, and is denoted by

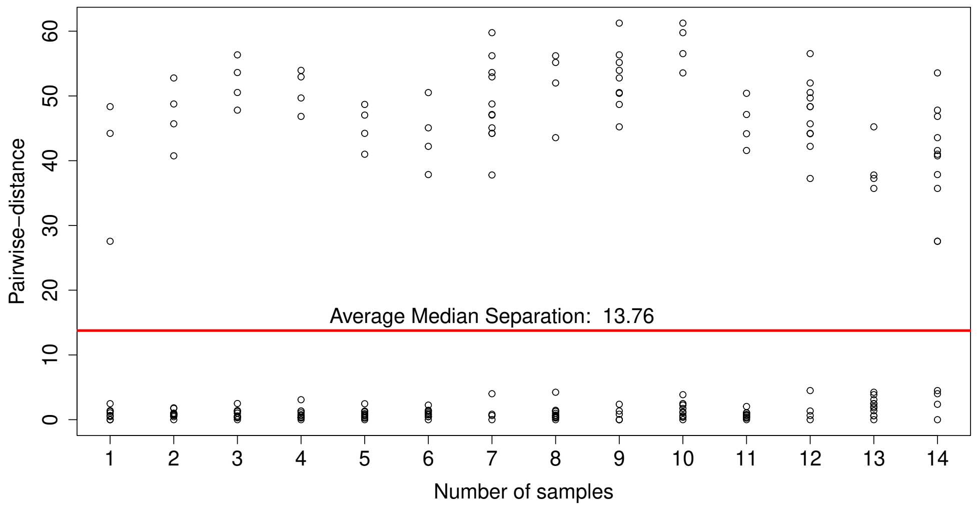
$$p_{i,j} = p_{j,i} = \frac{1}{T} \sum_{t=1}^{T} \left(y_{i,t} - y_{j,t} \right)^2, \tag{4.4}$$

where T is the final time t in the time series.

Figure 4.8b illustrates the pairwise-distance between those 14 sensors, according



(a)



(b)

Figure 4.8. Analysis of the time-series and pairwise-distances between 14 reliable sensors with $r \geq 0.8$ to the reference temperature data (Dataset 1). (**a**) Time-series analysis of those 14 reliable sensors; (**b**) Pairwise-distance between the time series of the reliable sensors.

to the computed values of $\mathbf{P}$. We can see that most of the correlated samples have a small distance among themselves, and a higher distance for the dislocated sensors. Thus, sensors that have the median pairwise-distance above the average median of them, the red straight line in Figure 4.8b, can be considered to belonging to $\mathcal{D}$. In this case, $\mathcal{D} = \{r_7, r_9, r_{12}, r_{14}\}$.

The estimated value $\hat{\Delta}_i$ for each sensor r_i dislocated from the non-dislocated sensors can be denoted as the median of deviations from all the pairwise-distances between this sensor and the others, *i.e.*, for each row i of $\mathbf{P}$, we can compute $\hat{\Delta}_i$ as follows

$$\hat{\Delta}_i = \text{median}(p_{i,j} - \text{median}(p_{i,j})), 1 \leq j \leq n. \quad (4.5)$$

This equation is a slight variation of the median absolute deviation, without the absolute value computation, to adjust the correct value, positive or negative. For our example, $\hat{\Delta}_7 = 11.7309425$, $\hat{\Delta}_9 = 9.2068333$, $\hat{\Delta}_{12} = 9.8445421$, and $\hat{\Delta}_{14} = 11.8916635$.

Algorithm 5 describes the necessary steps to compute the set $\mathcal{D}$ of dislocated sensors and their dislocation values $\Delta_i, \forall i \in \mathcal{D}$.

ALGORITHM 5: SensorRefinement

```
Input: $\mathbf{S}_{n\times m}$ matrix of time series for $n$ sensors with $m$ samples each, and a set
       $\mathcal{R} = \{r_1, r_2, \ldots, r_n\}$ of reliable sensors.
Output: Set $\mathcal{D}$ of dislocated sensors and vector $\Delta$ of their dislocations from the
        majority.
1  $\mathcal{D} \leftarrow \emptyset$
2  $\Delta \leftarrow \{\Delta_i = 0 \mid 1 \leq i \leq n\}$
3  $\mathbf{m} \leftarrow \{m_i = 0 \mid 1 \leq i \leq n\}$
4  $\mathbf{P}_{n\times n} \leftarrow \{p_{i,j} = 0 \mid 1 \leq i, j \leq n\}$
   // Compute the average squared pairwise-distance between each two sensors.
5  for $i \leftarrow 1$ to $n-1$ do
6      for $j \leftarrow i+1$ to $n$ do
           // Adjust the valid readings among the reliable sensors.
7          $\mathbf{u} \leftarrow \{\ s_{r_i,k} \mid (s_{r_i,k} \neq \lambda)$ and $(s_{r_j,k} \neq \lambda), 1 \leq k \leq m\ \}$;
8          $\mathbf{v} \leftarrow \{\ s_{r_j,k} \mid (s_{r_i,k} \neq \lambda)$ and $(s_{r_j,k} \neq \lambda), 1 \leq k \leq m\ \}$;
9          $p_{i,j} \leftarrow p_{j,i} \leftarrow \text{mean}((\mathbf{u} - \mathbf{v})^2)$
10     end
11 end
   // Compute the median pairwise-distance for each sensor.
12 for $i \leftarrow 1$ to $n$ do
13     $m_i \leftarrow \text{median}(\{p_{i,j} \mid 1 \leq j \leq n\})$
14 end
   // The limit to consider a sensor dislocated.
15 $l \leftarrow \text{mean}(\mathbf{m})$
   // The dislocated sensors.
16 $\mathcal{D} \leftarrow \{i \mid m_i > l\}$
   // Compute the dislocation $\Delta_i$ for each reliable sensor.
17 for $i \leftarrow 1$ to $n$ do
18     $\Delta_i \leftarrow \text{median}(p_{i,j} - m_i)$
19 end
20 return $(\mathcal{D}, \Delta)$
```

Considering the points mentioned above, we can argue that, for a sufficient large number of sensors, if we observe a pattern between the correlated sensors, it is possible to estimate the correct signal of the temperature at a given region. This is accomplished by a collaborative filtering strategy [Adomavicius and Tuzhilin, 2005], assuming that most of the sensors in such a region have a higher tendency to be close to the correct temperature value. Also, based on this assumption, we can estimate the deviation Δ_i of sensors that are a magnitude apart from the majority, to improve the quality of the sensed data.

An important point to discuss regarding the benefits of the proposed strategy is about the security concerns of the obtained data quality in the Collaborative IoT. Coen-Porisini and Sicari [2012] present a detailed study of the problem of a malicious sensor generating erroneous data, and, here, we emphasize two malicious behaviors: (i) data is modified by a constant factor; and (ii) data is modified by the addition of a random error, a noise.

In the first behavior, the sensor readings are apart from the correct data values by a constant. In this case, the data signal still holds a reasonable correlation with the reference signal, being, thus, selected as reliable. After the refinement step, their values will be corrected according to the majority of sensors, by the collaborative filtering strategy. For the second case, when the data is modified by a random noise, it is presumable that, according to the magnitude of the noise increments, this signal will be considered unreliable and invalidated, and, thus, it will not be used. This occurs because it will not be correlated enough with the reference data, according to the correlation limit established.

4.5 Evaluation of Our Approaches

In the first experiment we use Dataset 1, described in Section 4.3, which contains only temperature data shared in the popular Xively collaborative IoT platform. We apply Algorithm 4 to search and select the most reliable sensors in this dataset.

The parameters passed to Algorithm 4 are the matrix of all sensors found in a range of 100 km from the coordinates of London, UK (see Table 4.1 for details), the Forecast.io weather service as the temperature reference data, and the correlation limit of $r = 0.8$. The validity rules for the class of temperature data were arbitrarily defined, based on an empirical knowledge of the data behavior, according to the following restrictions:

- The average temperature of the sampled data must be bounded by a factor of 2 from the average temperature of the reference data, *i.e.*, $\frac{1}{2}m_{ref} \leq m_{smp} \leq 2m_{ref}$, where m_{ref} and m_{smp} are the average temperature values for the reference data and the sampled data, respectively;

- The maximum temperature value of the sample $\max_{smp}$ can not be above the bound of 4 times the average reference temperature, $\max_s mp \leq 4m_{ref}$; and
- The minimum temperature value of the sample $\min_{smp}$ is lower bounded by a factor of 4 from m_{ref}, $\min_{smp} \geq \frac{1}{4}m_{ref}$.

Figure 4.9 illustrates the pairwise-distance of the 22 selected sensors for this computation. After selecting those sensors, which we assume to be reliable, we compute the dislocation values Δ_i for each sensor i applying Algorithm 5.

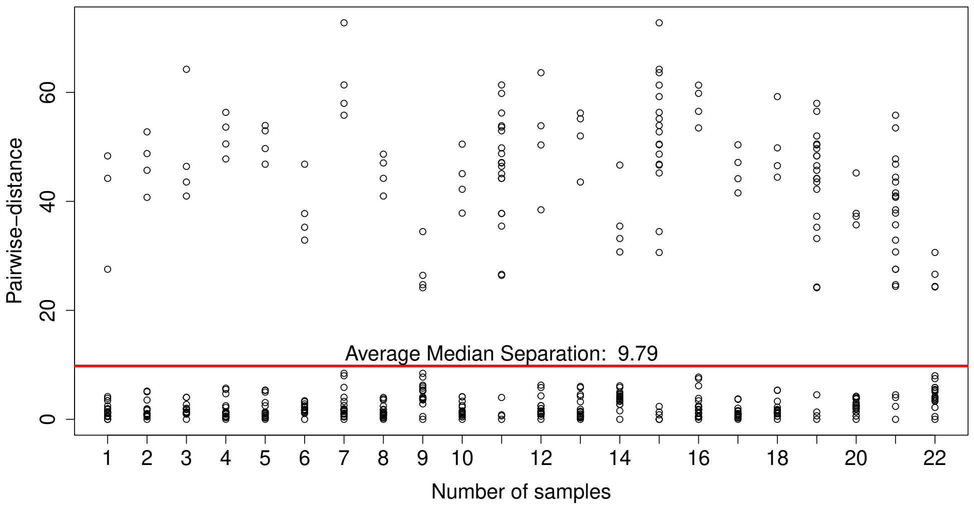


Figure 4.9. Pairwise-distances for temperature sensors in a range of 100 km from the coordinates of London, UK, in Dataset 1, considering only sensors with correlation $r \geq 0.8$.

To validate our assumption that the combination of the readings for most of the reliable sensors converge to a correct value of the measured data, *i.e.*, the reference temperature data, we estimate the temperature $\hat{\theta}$.

Figure 4.10 shows the estimation of the reference data, based on Equation 4.3. Figure 4.10a shows the sensors that are 0.8 positive correlated with this reference, resulting in an MSE = 2.244199. Figure 4.10b shows the same estimation process, but relaxing the correlation criterion, considering all sensors with correlation coefficient $r \geq 0.2$ to the reference sensor, resulting in an MSE = 16.85198. As we can see, the more correlated the sensors are with the reference, the better the capacity of estimating $\hat{\theta}$ is. Figure 4.10 also shows that the estimation error increases as we consider more unreliable sensors.

We did the same analysis for Dataset 2, but considering a broader range of data classes. In this case, the goal is to select a desired group of sensors from a set of

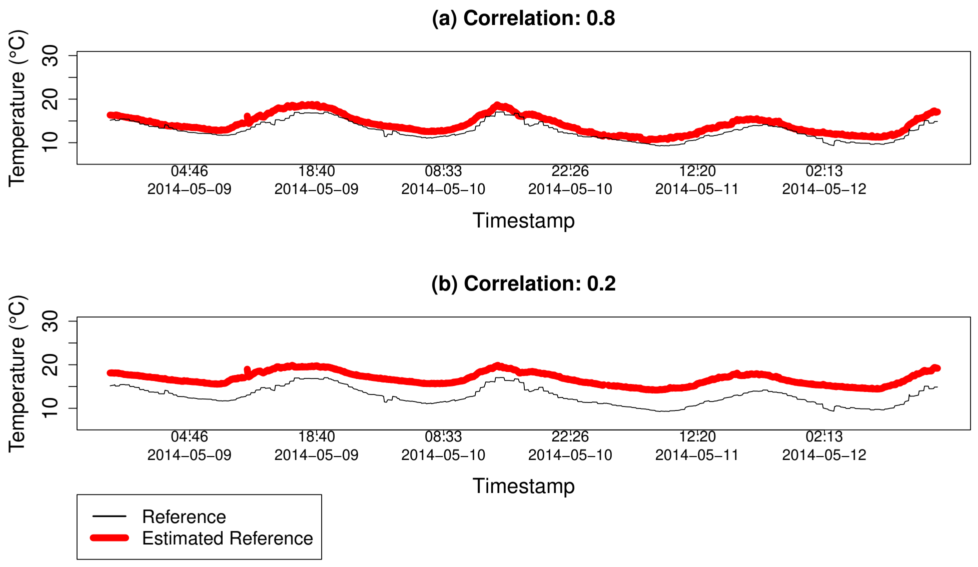


Figure 4.10. Estimation of the temperature $\hat{\theta}$ for two different correlation limits in a range of 100 km from the coordinates of London, UK, in Dataset 1: (**a**) for sensors with correlation $r \geq 0.8$ with MSE = 2.244199; and (**b**) for sensors with correlation $r \geq 0.2$ with MSE = 16.85198.

heterogeneous classes of sensors. In this work, we consider selecting reliable sensors for temperature, pressure, humidity, and wind speed.

For the temperature sensor selection, we considered the same steps and parameters of the previously mentioned experiment for Dataset 1. Figure 4.11 presents the resulting pairwise-distances for the 18 sensors found in Dataset 2. It is important to consider the time interval between collections, almost five months, to explain the difference in the number of detected sensors. In Figure 4.12, we have the estimation analysis for these reliable sensors. For different correlation limits, $r \geq 0.8$ and $r \geq 0.2$, we have the mean squared errors of 2.045454 and 15.63293, respectively, which are very close to the results in the previous experiment.

These results show that the proposed method is a reasonable approach to select reliable sensors, regardless of having a controlled set of sensors or not. Even when mixing the available sensors with different data classes, the selection of the reliable sensors gives a good approximation of the assumed correct temperature values.

For the next data classes, we considered the following parameters for Algorithm 4:

- For the pressure data, the correlation limit is defined as $r \geq 0.7$, and the validity rule ensures that the average pressure of the sample data is upper and lower bounded by a factor of 4 from the average of the reference data.

- For the humidity data, we consider the correlation limit of $r \geq 0.65$ and the validity rules consider a valid data to be a maximum humidity value of up to

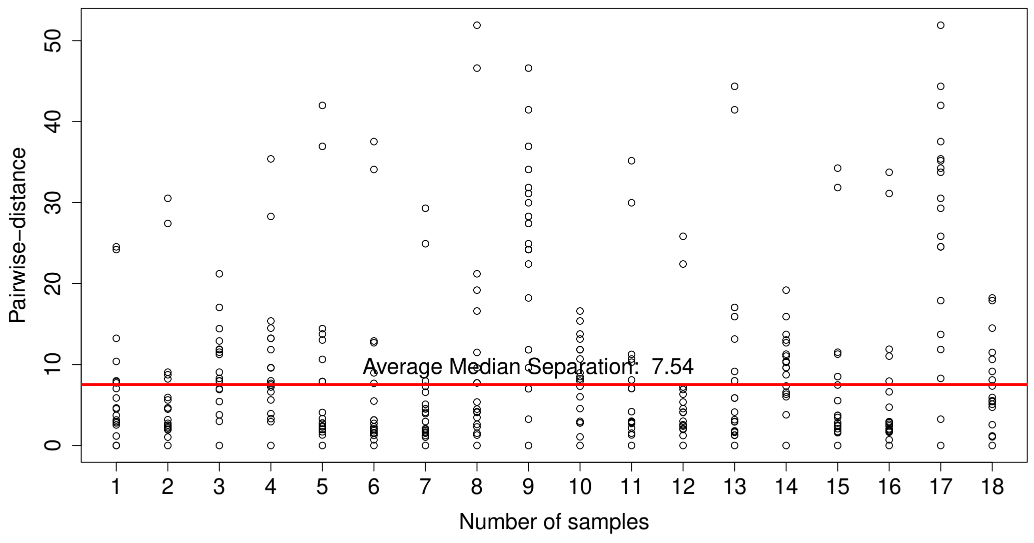


Figure 4.11. Pairwise-distances for temperature sensors in a range of 100 km from the coordinates of London, UK, in Dataset 2, considering only sensors with correlation $r \geq 0.8$.

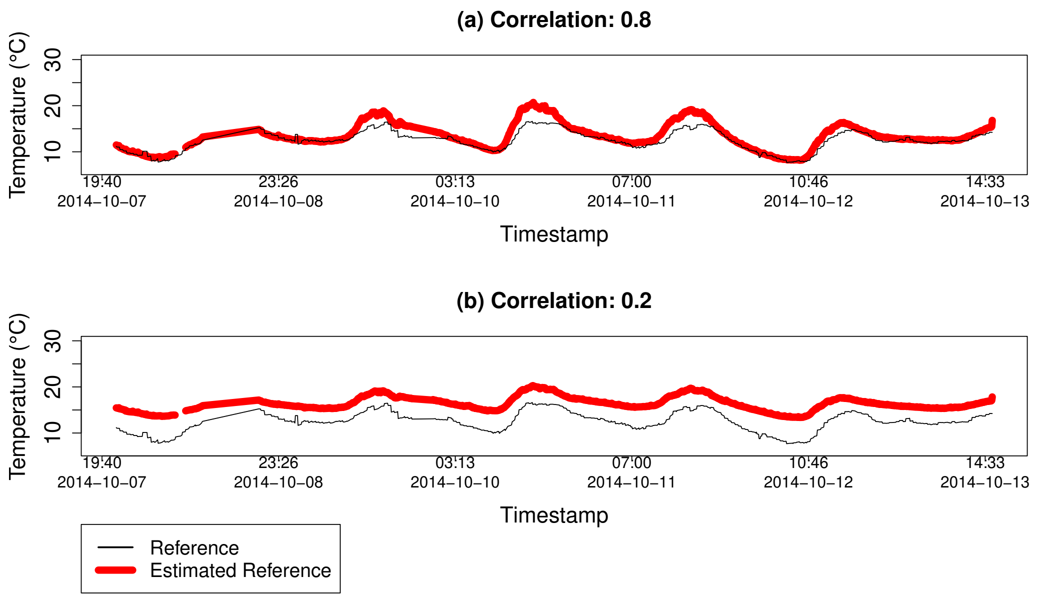
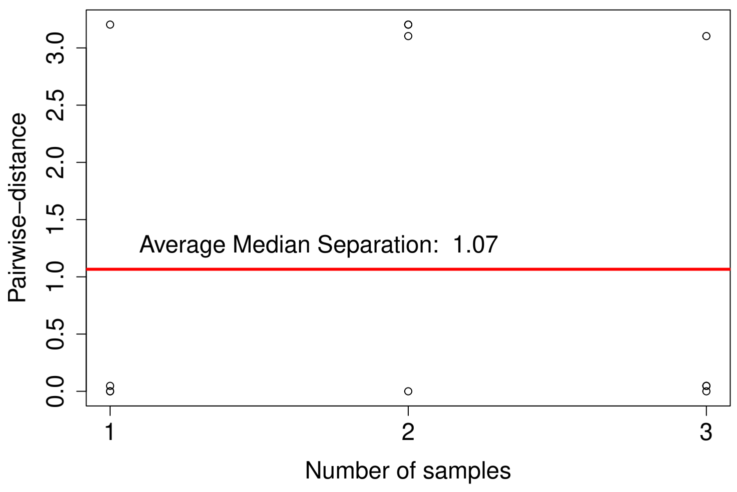


Figure 4.12. Estimation of the temperature $\hat{\theta}$ for two different correlation limits in a range of 100 km from the coordinates of London, UK, in Dataset 2: (**a**) for sensors with correlation $r \geq 0.8$ with MSE = 2.045454; and (**b**) for sensors with correlation $r \geq 0.2$ with MSE = 15.63293.

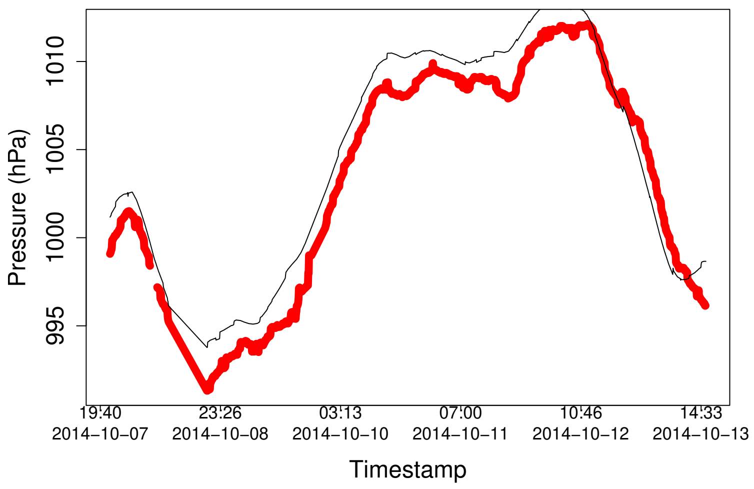
100 % and the minimum not less than 0 %.

- For wind speed, even with a correlation limit of $r \geq 0.5$, only one sensor was selected.

For the pressure data (Figure 4.13), we selected only three reliable sensors follow-



(a)

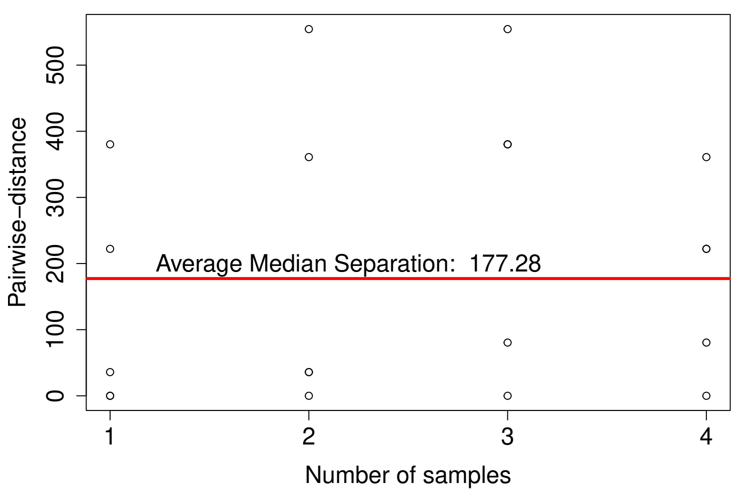


(b)

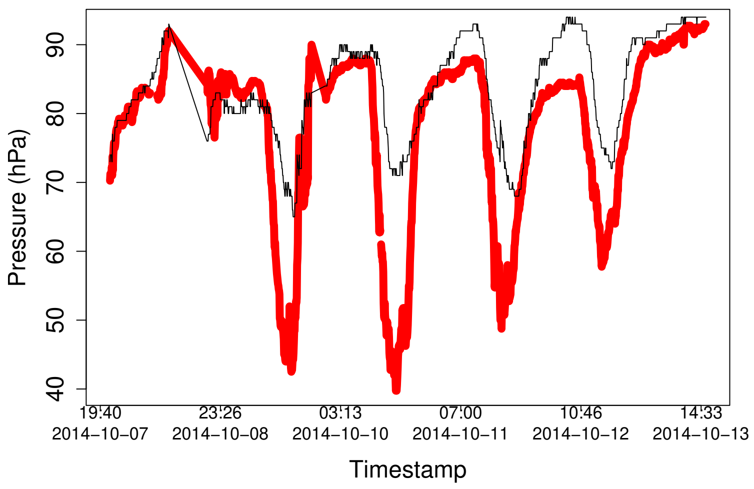
Figure 4.13. Analysis of pressure data from sensors in a range of 100 km from the coordinates of London, UK, in Dataset 2, considering a limit of $r \geq 0.7$. (**a**) Pairwise-distances for pressure sensors; (**b**) Estimation of the pressure, resulting in an MSE = 2.66978.

ing our approach with $r \geq 0.7$, but the refinement was able to combine those sensors and obtain an MSE of 2.66978. However, to the case of the humidity data (Figure 4.14), with a limit $r \geq 0.65$, our approach was able to select only four reliable sensors, and the MSE was in the order of 104.213.

For the wind speed data, Figure 4.15 shows the only sensor selected with a low correlation coefficient $r = 0.5206303$ between it and the reference data. For this case, there is no estimation of the correct value, and we must consider the readings of that



(a)



(b)

Figure 4.14. Analysis of humidity data from sensors in a range of 100 km from the coordinates of London, UK, in Dataset 2, considering a limit of $r \geq 0.65$. (**a**) Pairwise-distances for humidity sensors; (**b**) Estimation of the humidity, resulting in an MSE = 104.213.

sensor as our wind speed value.

In the experiments we conducted above for the second dataset, the number of selected sensors was significantly lower than the number of temperature sensors (Dataset 1). Thus, when the number of selected sensors, which provide the more reliable value, decreases, it is possible that there is no majority of sensors to converge to the expected output of the given environment data. Thereby, our approach depends on the number of sampled data, which we can not guarantee to occur. However, as

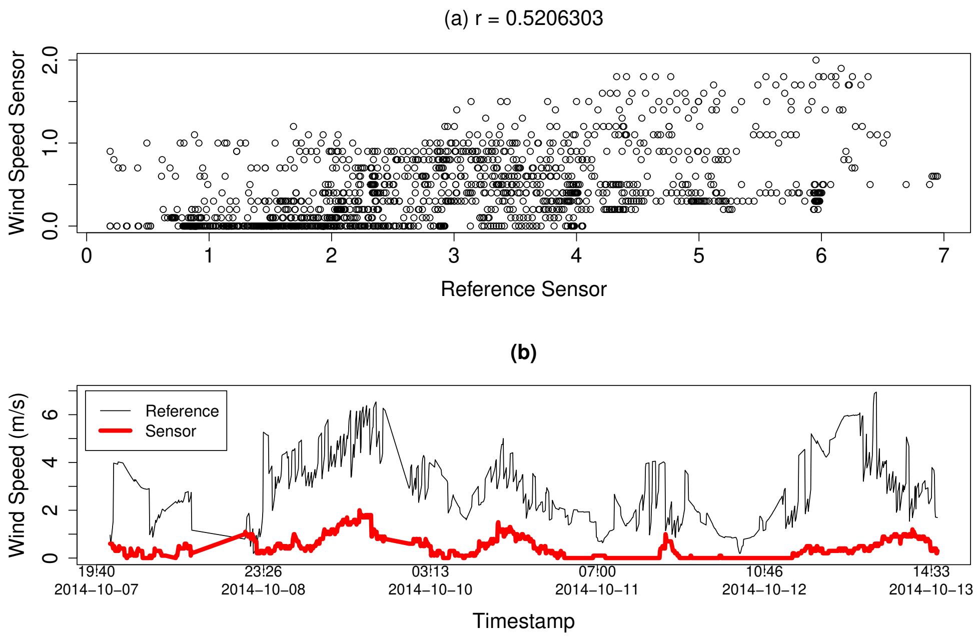


Figure 4.15. Analysis of the single wind speed sensor, with $r = 0.5206303$, in a range of 100 km from the coordinates of London, UK, in Dataset 2: (**a**) scatter plot between the reference and this sensor; and (**b**) the time series analysis of its readings.

stated earlier, in this era of IoT and big data, we can expect to have more and more data available in the future, and, if it is the case, makes our approach feasible.

In addition, on the majority of the selected sensors, the definition of the correlation limit value is crucial to the proper execution of our strategy. It is clear that the chosen correlation limit r directly impacts the number of selected sensors. As r approaches 1, the expected number of sensors that generate a data signal highly correlated to the reference value decreases and, conversely, this number is expected to increase when r approaches 0. However, it is also important to note that, even if the number of selected sensors increases, due to the choice of $r \approx 0$, the quality of the sensor refinement algorithm will decrease, since it is expected that the mean squared error between the fused signals and the reference signal increases.

The trade-off between increasing the number of selected sensors and decreasing the MSE of their aggregation calls for a solution to determine the most appropriate correlation limit r. The optimum value of the correlation limits, for a given data class to be monitored, maximizes the number of selected sensors using the sensor selection strategy (Algorithm 4), and, once aggregated using the sensor refinement adjustments (Algorithm 5), results in the minimum mean squared error between them and the reference signal for this data class. In the case of our experiments, the choice of r was empirically determined by analyzing the selected sensors, and testing r in the reasonable range of 0.5 to 0.9.

4.6 Conclusions

Dealing with the uncertainties of the sensed data in a collaborative IoT scenario is a new and important problem that must be considered before the development of systems that can benefit from this data. The core of ubiquitous computing systems is based on the knowledge they infer about the state of the physical environment and, thus, the reliability of the sensed data will directly impact the decisions and context-awareness of those systems.

In this chapter, we characterized the properties of the available sensed data from real deployed sensors in the current collaborative IoT, for which we have a reference value. We concluded that to use data available in the IoT safely, we need to perform a filtering process followed by a refinement of the selected data to increase its reliability. Thus, we proposed a simple and effective approach to select and refine the data from reliable sensors by a collaborative filtering technique.

Our assumption is that the readings of most of the reliable sensors converge to the correct value of the measured data. This hypothesis was validated through an experiment using data collected from real deployed sensors. The results show that the proposed method is a promising approach to select reliable sensors, regardless of having a controlled set of sensors or not. Even when mixing the available sensors with different data classes, the selection of the reliable data provides a good approximation of the assumed correct values based on a reference sample.

When the number of selected sensors decreases, we may not have enough sensors. The collaborative filtering technique can not be expected to converge to the correct value of the given variable (in our example, environmental data). However, we can expect to have more sensors available in the future (possibly in the order of hundreds of millions) since we are entering a new era of computing technology with a powerful presence of the Internet of Things, and, thus, this issue will probably be minimized for different classes of sensor data.

As future work, we plan to extend our analysis of the properties of the sensed data in the IoT and its different classes to better understand how the number of sensors impacts the collaborative filtering strategy, and infer about its precision limits. This analysis can help us determine the minimum subset of available sensors in the collaborative IoT needed to assure the data quality.

In addition, an interesting point we plan to investigate is related to the heterogeneity and limitations of IoT devices. Since IoT comprises different devices with different capabilities, the data accuracy provided by such elements can be compromised due to their limitations. Probably, the more limited the device is, the lower the data accuracy will be. Thus, we intend to study how these limitations can be used as another characteristic in the sensor selection strategy.

It is also very important to investigate how the lack of a reference sensor influences the design of a solution based only on the analysis of the data itself. In that direction, we would like to study some possible strategies and the corresponding costs to model this as a mathematical/computational problem, such as an optimization problem, or even as a problem of statistical inference.

Finally, another direction is to explore the potential of ubiquitous computing by combining different data sources (e.g., humidity, rain precipitation, and even social networks) to improve the data reliability and accuracy.

Chapter 5

Learning time series dynamics via ordinal patterns transition graphs

Strategies based on the extraction of measures from ordinal patterns transformation, such as probability distributions and transition graphs, have reached relevant advancements in distinguishing different time series dynamics. However, the reliability of such measures depends on the appropriate selection of parameters and the need for large time series. In this chapter, we present a method for the characterization of time series, based on the probability of self-transitions, that is suitable for scenarios with short time series, and that does not depend on the selection of parameters.

In Section 5.1, we introduce the problem and motivate our claim on the use of the ordinal patterns transition graph for distinguishing time series dynamics. A discussion of limitations and challenging aspects of the ordinal patterns transformations is presented in Section 5.2. More details and a discussion on the properties of those transition graphs are given in Section 5.3. Section 5.4 presents an analysis on the forbidden and missing transition patterns, a special case of this type of graphs, and Section 5.5 presents our proposed probability of self-transition measure. In Section 5.6, we present our results for learning and distinguishing different time series dynamics. Finally, Section 5.7 discuss our final remarks of this chapter.

5.1 Introduction

An essential initial step for any learning strategy is to properly understand the data. When dealing with time ordered data, as presented in Chapter 3, an important aspect to be concerned is their dynamics. With the knowledge of how a system evolves over time, it is possible the development of better solutions, by understanding the underlying data generating process [Rosso et al., 2015; Gama et al., 2014]. However, there are few studies of using dynamical aspects of time series for knowledge discovery, which is the basis for several data mining strategies (e.g., classification, anomaly detection).

By following the cornerstone contributions of Bandt and Pompe [2002], the ordinal patterns transformations, detailed in Section 3.5, are a powerful tool for the characterization of time series dynamics [Rosso et al., 2013; Aquino et al., 2015; Rosso et al., 2016; Aquino et al., 2017; Ribeiro et al., 2017]. However, the feasibility of such characterization depends on the proper choice of parameters and has some limitations impacting on the clear separability between distinct time series dynamics.

Recent approaches to obtain knowledge about the time series dynamics consider the creation of graphs based on the observed ordinal patterns of a given time series [McCullough et al., 2015; Sorrentino et al., 2015; Kulp et al., 2016; Zhang et al., 2017]. These ordinal patterns transition graphs (G_π) are constructed after the transformation of a time series onto the set of ordinal patterns, taking into account the transitions between consecutive patterns. As described in Section 3.5.3, each $D!$ possible pattern is a vertex in the graph, and a directed edge connects two vertices in the graph if they appear sequentially in the time series.

The analysis of these graphs is often performed on their structure, by accounting both graph measures and information quantifiers, which may require creating graphs with a large enough number of vertices. For the case of G_π, the number of vertices depends on D. Thus, the reliability of such metrics may suffer from the same inherited problems of the underlying ordinal patterns transformations. For instance, the range $6 \leq D \leq 10$ may invalidate the strategy for small time series [McCullough et al., 2015]. Another challenge when dealing with strategies based on ordinal patterns transformations is that there is still no clear definition regarding the choice of the embedding delay τ. This is also an aspect to consider when constructing the transition graphs, since inappropriate values may obscure important characteristics of the phenomenon under analysis [Zunino et al., 2012].

In this study, we present a novel method for the characterization of time series dynamics based on the probability of self-transitions (p_{st}), a measure extracted from the graph G_π. The main contributions of this work are the following:

1. we show that the p_{st} measure is directly related to the temporal correlation of time series, which is a valuable indicative of their underlying dynamics and adequate to their proper characterization;

2. we advance the state of the art by providing a precise distinction among different time series dynamics within those challenging regions in the plane, as discussed

above, even for short time series. In our experiments, we use time series with length $n = 5 \times 10^3$, being necessary small values of D, in contrast to previous research results that considered lengths from $2^{15} (\approx 3.2 \times 10^4)$ to 10^7, i.e., one to four orders of magnitude of difference; and

3. we reduce the dependence of the ordinal patterns transformation on the selection of the parameters. We assume small values of D and, in order to be independent on the choice of τ, we propose evaluating the evolution of their behavior for different time scales.

Furthermore, this strategy also aims to contribute in motivating the community towards a broader applicability of the ordinal patterns transformation, being possible to be used for several different domains, such as Medical Sciences, Engineering, Mathematics, Computing and Physics, where time series representing the dynamics of a system must be characterized and better understood.

5.2 Limitations of Bandt-Pompe's method

The effectiveness of using ordinal patterns for the characterization of time series has already been largely discussed by many recent studies [Rosso et al., 2013, 2016; Ribeiro et al., 2017; Rosso et al., 2007a]. These are strong results that have driven extensive discussions in the literature. However, most of them are confined to theoretic problems in areas such as physics and statistics. When trying to apply those strategies to a broader domain, there are some issues that still hinder the achievement of similar success. In the following, we conjecture that specific aspects of Bandt-Pompe's proposal and the current methods used to capture time series behaviors are responsible for adding limitations to a wide applicability.

5.2.1 Strong dependence on parameters definition and time series length

Strategies based on ordinal patterns transformations depend on the proper choice of parameters D and τ. For instance, to assure the reliability of measures, the length n of the time series must be long enough so the sampling in the $D!$ space of patterns is representative, i.e., $n \gg D!$ [Rosso et al., 2013; Zunino et al., 2012]. For practical purposes, D is recommended to be within the interval between 3 and 7 [Bandt and Pompe, 2002]. If this condition is not satisfied, it may lead to analysis of statistics that are not consistent to the real time series behavior and the occurrence of missing patterns [Rosso et al., 2012]. Recall that the number of possible patterns grows with $D!$. Thus, if the time series is not long enough, the absence of some patterns may occur, but just due to its limited length n, not because these patterns do not occur within the time series behavior. This effect also occurs on the derived transformations, such as the probability distribution and transition graph transformations.

For instance, for large values of D, the constructed graph may have a large number of vertices, up to $D!$. Thus, if the time series length is also not long enough, some of these vertices may be wrongly absent. However, for the current strategies based on ordinal patterns transformations, the best results are achieved when large values of D are used. For instance, the range $6 \leq D \leq 10$ has been suggested [McCullough et al., 2015] for the extraction of representative behaviors from the constructed graphs. This fact practically invalidates the application of those strategies for small time series, which represent a severe limitation for many real-world scenarios.

Another challenge when dealing with ordinal patterns is the choice of the embedding delay τ. The most suitable τ is domain specific, and inappropriate values may obscure important characteristics of the phenomenon under analysis [Zunino et al., 2012]. A common choice found in many studies is $\tau = 1$. Although some of them follow different strategies, such as using the first zero of the autocorrelation of the time series [McCullough et al., 2015], there is still no closed strategy for choosing its value.

One of our contributions consists of a strategy to advance the applicability of Bandt-Pompe's method, being possible to be used even for short time series, in contrast to previous strategies. Moreover, the proposed method also presents minimum dependency on the selection of parameters. We must consider small values of D and be independent of the choice of τ. We accomplish this last requirement by comparing the behavior of time series when evaluated as a function of the embedding delay τ, which is better described later on.

5.2.2 Clear separability of time series dynamics within CCEP

The distinction of time series dynamics is not an easy task. For instance, because they share several properties, the distinction between time series from chaotic and stochastic processes is a hard task [Rosso et al., 2015, 2013; Ribeiro et al., 2017; Rosso et al., 2007a; Zunino et al., 2012; Ravetti et al., 2014; Kulp and Zunino, 2014; Ye et al., 2017]. The CCEP, presented in Section 3.6.2, has been successfully applied [Rosso et al., 2013, 2007a] to effectively distinguish noise from chaos, by placing each type of series at different locations in the plane. Figures 5.1a and 5.1b show the CCEP for $D = 3$ and $D = 6$, respectively, in which we reproduce the scenario for the distinction of 27 chaotic maps and 15 colored random noises.

Table 5.1. Descriptions of the 27 chaotic maps used in the characterization [Rosso et al., 2013]. They are arranged into three different groups, and each chaotic map may have one or more dimensions. Each time series arising for these chaotic maps dimensions are defined in text by the syntax "n-d", where $n = \{1, \ldots, 27\}$ represents the number of the chaotic map and $d = \{x, y, z\}$ its respective dimension. More details regarding the generation of these maps and their initial conditions can be found in Ravetti et al. [2014], including the supplementary materials.

#	**Description**	**Group**	**Dimensions**
1	Logistic map		{x}
2	Sine map		{x}
3	Tent map		{x}
4	Linear congruential generator		{x}
5	Cubic map		{x}
6	Ricker's population model	Noninvertible maps	{x}
7	Gauss map		{x}
8	Cusp map		{x}
9	Pinchers map		{x}
10	Spence map		{x}
11	Sine-circle map		{x}
12	Hénon map		{x,y}
13	Lozi map		{x,y}
14	Delayed logistic map		{x,y}
15	Tinkerbell map		{x,y}
16	Burgers' map	Dissipative maps	{x,y}
17	Holmes cubic map		{x,y}
18	Dissipative standard map		{x,y}
19	Ikeda map		{x,y}
20	Sinai map		{x,y}
21	Discrete predator-prey map		{x,y}
22	Chirikov standard map		{x,y}
23	Hénon area-preserving quadratic map		{x,y}
24	Arnold's cat map	Conservative maps	{x,y}
25	Gingerbreadman map		{x,y}
26	Chaotic web map		{x,y}
27	Lorenz three-dimensional chaotic map		{x,y,z}
	Total of time series		**44**

Altogether, we analyzed 59 time series: 15 random noises and 44 chaotic maps, considering their dimensions. The chaotic maps are arranged into three different groups (noninvertible, dissipative and conservative maps), and each chaotic map may have one or more dimensions. Table 5.1 presents the list of maps, with dimensions in curly

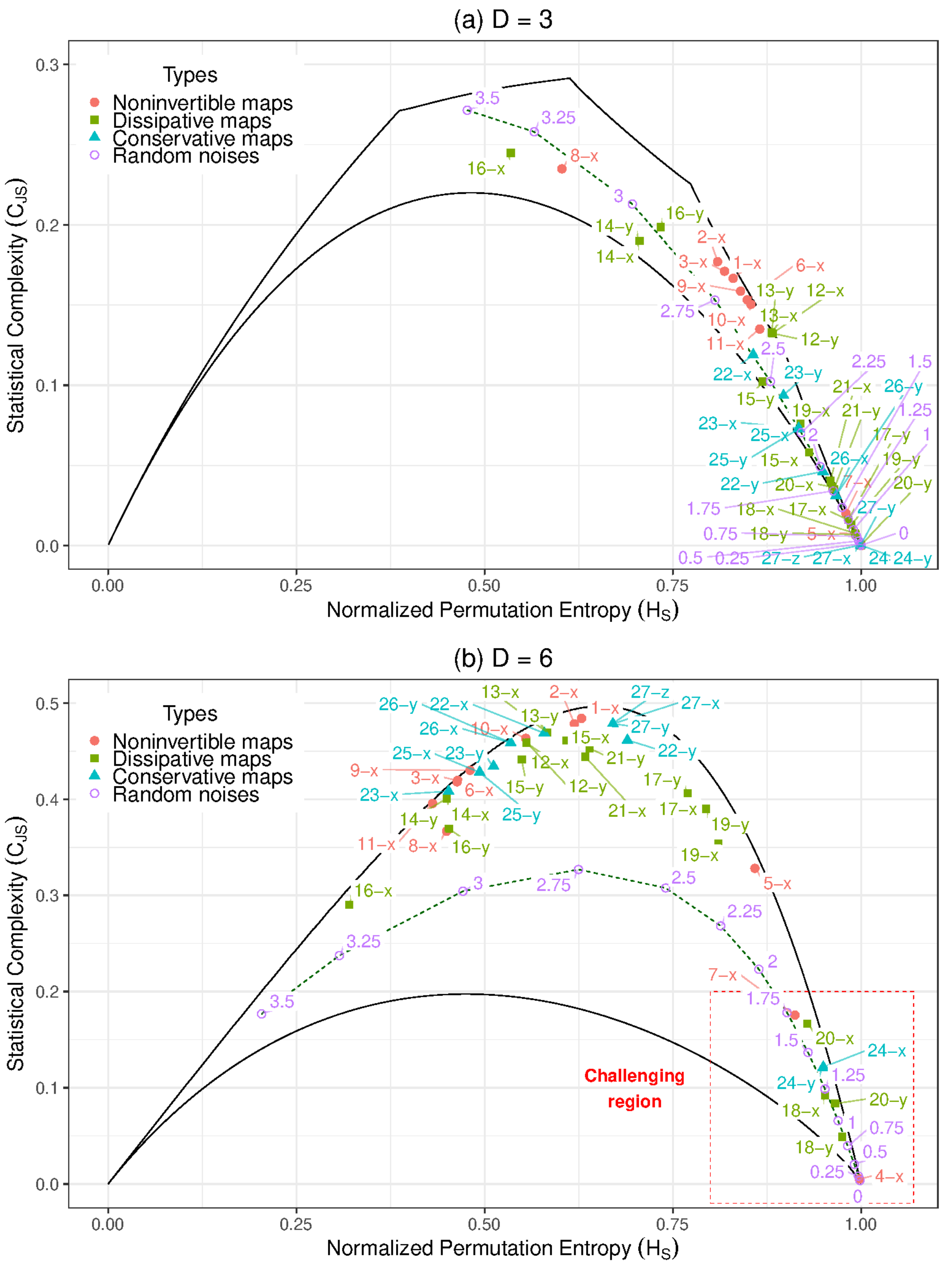


Figure 5.1. Reproduction of the CCEP [Rosso et al., 2013] of 27 chaotic maps and 15 random noises. Chaotic maps are illustrated by their notation "n-d", where n is the map number and d its dimension. Random noises are illustrated as sequences of open circles with dashed lines, with respective values k, defined by their power spectra f^{-k}. The measures were computed from series with length 10^5 each, after discarding the first 5,000 samples, with $\tau = 1$, and (a) $D = 3$ and (b) $D = 6$. For random noises, each measure was obtained as the average of 10 runs with different seeds. Continuous lines represent the minimum and maximum limits for statistical complexities [Rosso et al., 2007b].

brackets. The random noises were created to have different levels of correlation between points, defined by their power spectra f^{-k}. Each noisy time series was generated with $n = 5{,}000$ samples and the ordinal patterns transformation was computed as the average of 10 runs with different seeds.

Figure 5.1a shows a version of the CCEP for $D = 3$, which is not sufficient for distinguishing the time series by positions in the plane. Instead, larger values of D (such as $D = 6$ in Figure 5.1b) are able to contrast different placements for them. However, as the method requires larger values of D, it also requires longer time series. The time series length to compute this plane is $n = 10^5$.

Considering the case for $D = 6$, the colored noises have a well-defined placement in the plane, with intermediate values of C_{JS} and, for the entropy, there is an inverse relation between H_S and the correlation degree k. The random time series, with lower k, have larger entropy, while the more correlated noises, with higher k, present lower H_S. The general placement for the chaotic maps is close to the upper limits of C_{JS}, being straightforward to be distinguished from noises. However, there are a few chaotic maps whose placement lies at the bottom right of the plane, with large H_S and low C_{JS}, thus being very similar to random noises with small correlation degree k, hindering their distinction.

This challenging region is highlighted by a dashed-line rectangle in the bottom-right of Figure 5.1b. It comprises the colored random noises with $k = \{0, \ldots, 1.75\}$, and chaotic maps: linear congruential generator (4-x), Gauss map (7-x), dissipative standard map (18-x, 18-y), sinai map (20-x, 20-y), and Arnold's cat map (24-x, 24-y). Although the CCEP method is able to mostly distinguish noisy from chaotic time series, these chaotic maps could be easily confused as noises. There is no clear separability between those time series within the challenging region, so a distance-based algorithm could not be applied to correctly make the distinction among them. Thus, another contribution aimed with this work is to present a method for distinguishing different time series dynamics, even within this region.

5.3 Properties of ordinal patterns transition graphs

The ordinal patterns representation was proposed focusing on its simplicity and fast calculation. It was intended to enable the extraction of information measures from time series that could have similar results of other well-known methods, such as the Lyapunov exponent for chaotic systems [Wolf et al., 1985], but with a lower computational cost. Thus, once transition graphs are constructed from the ordinal patterns, some properties are directly inherited from that transformation. However, there are specific properties that could be noted when comparing it with other classical graph representations for time series.

5.3.1 Simplicity and fast calculation

The process of constructing transition graphs is very simple and fast. It only depends on the number $m = n - (D - 1)\tau$ of ordinal patterns, where n is the time series length, accounting the number of transitions in $m - 1$ steps. The transformation from a time series into the set of ordinal patterns, in turn, depends on the length n of the time series and the embedding dimension D. The time complexity to perform this transformation is bounded by $O(nD^2)$, assuming the permutations are obtained by sorting each sliding window by a simple sorting algorithm, such as selection sort, in $O(D^2)$ and the embedding delay $\tau = 1$ is the worst case. However, for practical purposes, D is recommended to be small [Bandt and Pompe, 2002], $3 \leq D \leq 7$, so the sorting will take at most 7 elements, and the time complexity largely depends on n.

5.3.2 Robustness

Since transition graphs are created from ordinal patterns, the robustness in creating this set is also inherited. This transformation is robust to the presence of observational and dynamic noise, and also invariant with respect to nonlinear monotonous transformations [Aquino et al., 2017; Rosso et al., 2007a]. Although losing the amplitude of the original time series, it is still suitable for the analysis of experimental data, avoiding amplitude threshold dependencies that affect other methods based on range partitions [Zunino et al., 2012].

5.3.3 Scalability

With respect to the scalability of the transition graphs, we can make a comparison with the visibility graphs approach. The main difference between G_π and VG approaches [Lacasa et al., 2008; Luque et al., 2009] is the scalability of graphs. In VG or HVG, each time series sample is mapped onto a vertex in the graph. Although this transformation can be satisfactorily performed in terms of time complexity, it requires a large space, which can be critical for long time series. Instead, the number of vertices for a transition graph is given by the embedding dimension D, independently of the time series length, and it is limited by $D!$, where $3 \leq D \leq 7$ [Bandt and Pompe, 2002].

Even for $D = 7$, it is easy to find time series in practice which will lead to a VG with a larger number of vertices.

5.4 Forbidden and Missing Transition Patterns

Deterministic time series have an important characteristic: some patterns may be forbidden and will not occur in the set of ordinal patterns Π, after data transformation, no matter the time series length [Amigó et al., 2006]. Instead of forbidden patterns, time series with stochastic components, if long enough, will exhibit all possible patterns. Unobserved patterns in random time series are not forbidden, but missing, as the result of the finite sample size [Rosso et al., 2012].

The same phenomenon can also be observed with respect to transition graphs. Similarly, there are forbidden and missing transition patterns, where some transitions between consecutive patterns do not occur. Unobserved patterns, whichever their nature, will not appear as a vertex in the constructed graph. Also, if a transition between two patterns does not occur in te set Π, this edge will not be present in the transition graph.

A special case of forbidden transition patterns that always occurs when $\tau = 1$, no matter the time series dynamics and value of D, is presented in Theorem 5.4.1.

Theorem 5.4.1 (Forbidden transitions)**.** Given a time series x with length n, if the embedding delay $\tau = 1$ is used for computing ordinal patterns, its constructed ordinal patterns transition graph G_π will always present forbidden transitions, independently of n, of the chosen embedding dimension D, and of the dynamics of its underlying phenomena.

Proof. Let $\mathbf{x} = (x_1, \ldots, x_n)$ be a time series that will be transformed into a set Π with $D = 3$ and $\tau = 1$. For a given time instant t, the sliding window $\mathbf{w}_t = (x_t, x_{t+1}, x_{t+2})$ is the subset of $\mathbf{x}$ that will be used for computing the pattern at t.

Let us assume, without loss of generality, that the pattern at t is $\pi_t = 123$, representing an ascending sequence of time series points, i.e., $x_t \leq x_{t+1} \leq x_{t+2}$. The next sliding window $\mathbf{w}_{t+1}$ will be composed of the sample points $(x_{t+1}, x_{t+2}, x_{t+3})$, as the last two sample points were already fixed.

Since we already know that $x_{t+1} \leq x_{t+2}$, then, the possible values for x_{t+3} are $x_{t+1} \leq x_{t+2} \leq x_{t+3}$, $x_{t+1} \leq x_{t+3} \leq x_{t+2}$, and $x_{t+3} \leq x_{t+1} \leq x_{t+2}$.

Thus, the possible ordinal patterns for these options at $t + 1$ are $\pi_{t+1} = 123$, $\pi_{t+1} = 132$, and $\pi_{t+1} = 312$, respectively.

By generalization, for each of the $D!$ patterns, the next possible transition patterns are limited to only D other patterns, since $D! - D$ patterns are forbidden due to the fixed previous points. $\square$

Corollary 5.4.2 (Maximum number of edges)**.** As a consequence of Theorem 5.4.1, the maximum number of edges, including loops, for a transition graph when $\tau = 1$ is bounded by

$$|E(G_\pi)|_{\tau=1} \leq D! \cdot D, \tag{5.1}$$

where $E(G_\pi)$ is the set of edges in G_π.

This limitation is given only for the case of $\tau = 1$, but it occurs for any value of D. For $\tau > 1$, the number of transitions depends on the time series behavior, and the maximum number of edges is given by a complete graph with loops, i.e.,

$$|E(G_\pi)|_{\tau>1} \leq D!^2. \tag{5.2}$$

If a transition graph has fewer edges, we may consider a scenario with more forbidden transitions or with missing transition patterns, but this last case occurs when the time series is not long enough so all the possible patterns can be observed.

5.5 Probabilty of self-trantisions

The transition graphs are important assets on the analysis of time series dynamics. Zhang et al. [2017] proposed to use transition graphs from ordinal patterns for the identification of joint dynamical changes in multivariate time series by using $D = 2$ and $\tau = 1$. Although $D = 2$ is very small in the univariate case, they extend the number of patterns by the combination of patterns for each dimension of the time series. For instance, in an m-dimensional time series, the number of patterns considered increases from $D!$ to $D!^m$. Their main contribution relies on the analysis of the entropy computed over the weights of edges in the graph (transitions between patterns), after the removal of the self-transitions (loops) in the graph.

However, since the self-transitions is directly related to the temporal correlation of time series, it is a valuable indicative for their underlying dynamics, and should not be discarded. The way edges are set is an important aspect to the further graph analysis. For Sorrentino et al. [2015] and McCullough et al. [2015], the weights for the edges are normalized such that all transitions from a given vertex sum up to 1, creating a Markov chain representation for the series. On the other hand, a normalization used by Zhang et al. [2017] considers the case where the sum of all weights is 1. The main difference for this last notation is that the presence of self-transitions (loops) in the graph represents the proportions of the occurrence of consecutive patterns.

These transitions are the basis for the computation of our probability of self-transitions (p_{st}) [Borges et al., 2019a], an important characterization criterion for different aspects of time series dynamics, which is presented in Definition 5.5.1.

Definition 5.5.1 (Probability of self-transitions)**.** The probability of self-transitions is defined as the probability of the occurrence of a sequence of equal patterns within the set of ordinal patterns. Assuming the normalization of edges weights, such that the sum of all weights is 1.

Formaly, p_{st} can be expressed as

$$p_{\mathrm{st}} = p(\pi_i, \pi_i) = \sum_{i \in \{1, \ldots, D!\}} w(v_{\pi_i}, v_{\pi_i}). \tag{5.3}$$

When considering the adjacency matrix $A_\pi = \{a_{\pi_i,\pi_j} : \pi_i, \pi_j \in \Pi\}$ of G_π, we can check that p_{st} is the trace of A_π. Thus, it can also be denoted as $p_{\mathrm{st}} = \sum_{\pi_i \in \Pi} a_{\pi_i,\pi_i}$, representing the sum of its diagonal.

In the following sections it is presented a method for the characterization of time series, by evaluating the probability of self-transitions from transition graph representations of their ordinal patterns. In fact, we show that the self-transitions, when evaluated as a function of the embedding delay τ, represent a valuable indicative of the main characteristics of the time series, even for small values of D, and independently on a single value of τ.

5.6 Characterization of time series dynamics

A key aspect when dealing with graph representations of time series is to discover which characteristics from the series are inherited by the graph, and how they can be extracted [Lacasa et al., 2008]. In this section, we apply learning algorithms for the characterization and distinction of time series, based on the measures extracted from the ordinal patterns transition graphs. We show how different dynamics can be expressed by analyzing inherited characteristics from periodic, random, and chaotic synthetically generated time series.

5.6.1 Periodic time series

An important issue in a graph constructed from periodic time series is how to identify their period within the graph properties. For some graph representations, which is the case for the VG and HVG transformations, periodic time series are mapped onto regular graphs and the degree distribution of the vertices has a peak suggesting the period [Lacasa et al., 2008]. For the ordinal patterns transformation, the number of vertices is limited to $D!$, avoiding a precise analysis of the degree distribution. Thus, an indicative of the time series period can be given from the transition graph when considering the p_{st} as a function of the embedding delay τ.

As a first analysis, let us consider the example of a periodic time series used by Lacasa et al. [2008] to illustrate their visibility graph approach. Its first 20 samples, from a total length $n = 1{,}000$, are illustrated in Figure 5.2a. It is a series of period $T = 4$, composed of n/T repetitions of the same pattern $(0.87, 0.49, 0.36, 0.83)$.

After the ordinal patterns transformation with $D = 3$ and $\tau = 1$, due to its determinism, we have forbidden patterns that will not be observed, no matter the length of the time series [Rosso et al., 2012; Amigó et al., 2006]. The G_π for $D = 3$ and $\tau = 1$ is illustrated in Figure 5.2c. The transitions between the four observed patterns have the same probability of occurrence, and there are no self-transitions, which give a $p_{\text{st}} = 0$.

However, different values of τ lead to different transition graphs. For instance, in Figure 5.2d, for $\tau = 2$, we have only two patterns but four possible transitions. In this case, $p_{\text{st}} = 0.5$, considering the sum of loops. For $\tau = 3$, in Figure 5.2e the graph is similar to the case $\tau = 1$, but patterns 213 and 312 were replaced by 132 and 231, respectively. Figure 5.2f shows the case for $\tau = 4$, when the embedding delay is equal to the period of the series. Here we have a particular behavior: only pattern 123 is observed and, thus, all possible transitions are loops and $p_{\text{st}} = 1$. This occurs because when $\tau = T$ all sliding windows extract equal values from the time series, and the sequence is considered already ordered. Repeated behaviors can be observed in Figures 5.2c–5.2f for different values of τ.

Since forbidden patterns do not occur in all periodic time series, it is not expected that only one pattern will be observed like this example. Thus, we have to generalize this observation to be able to suggest the period of a given periodic time series based on the p_{st}. Thus, we conjecture that p_{st} assumes maximal values when τ is equal to the period T or multiples of it, i.e., $\tau = nT, n = \{1, 2, 3, \dots\}$. This maximal values can be observed as peaks in the values of p_{st}, as shown in Figure 5.2b, which occurs regardless of D.

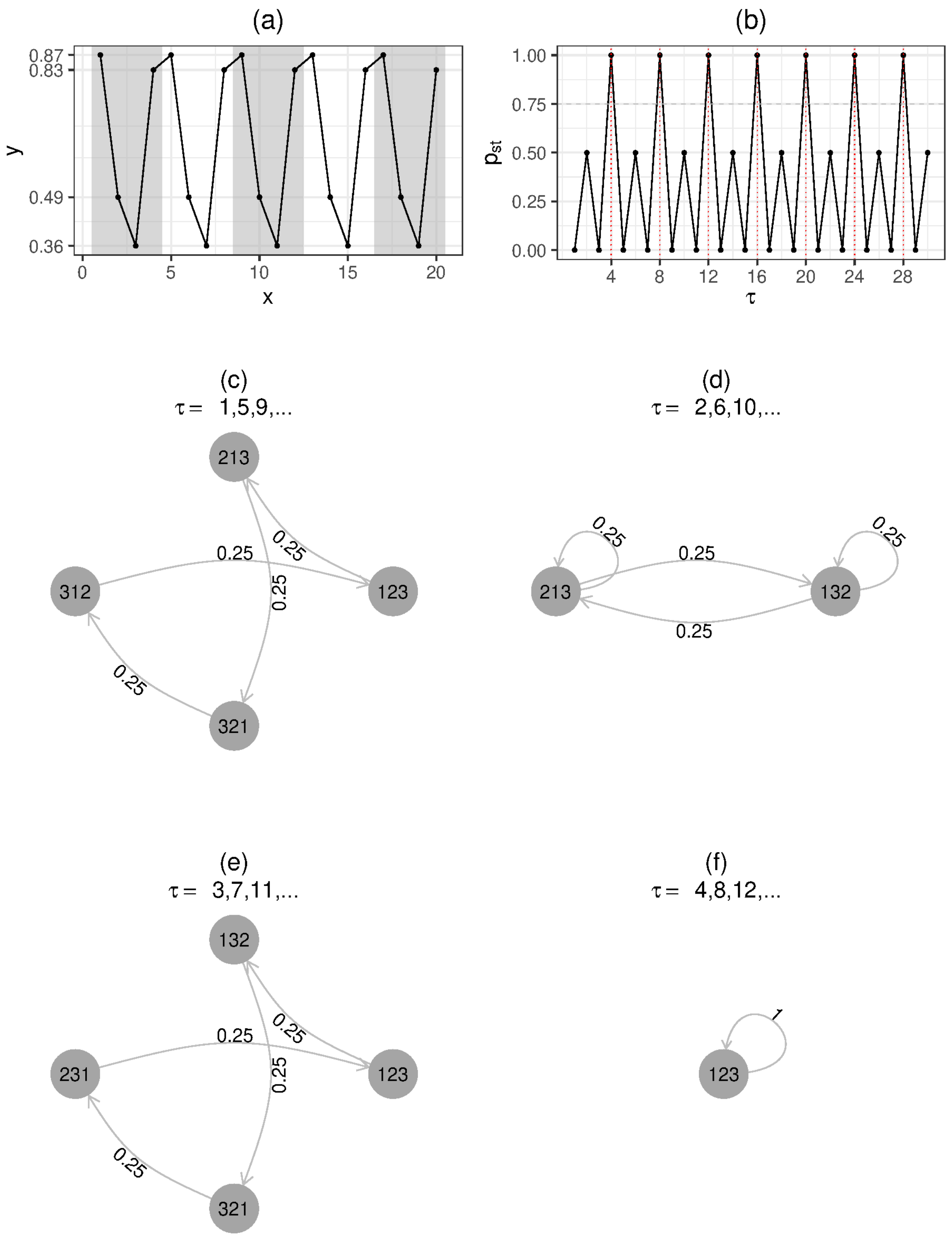


Figure 5.2. Example of (a) the first 20 samples of the periodic time series from VG [Lacasa et al., 2008] and (b) its p_{st} with $D = 3$ as function of $\tau \in \{1, \ldots, 30\}$. It is also shown the G_π for the time series, with $D = 3$, and (d) $\tau = \{1, 5, 9, \ldots\}$, (e) $\tau = \{2, 6, 10, \ldots\}$, (f) $\tau = \{3, 7, 11, \ldots\}$, (g) $\tau = \{4, 8, 12, \ldots\}$.

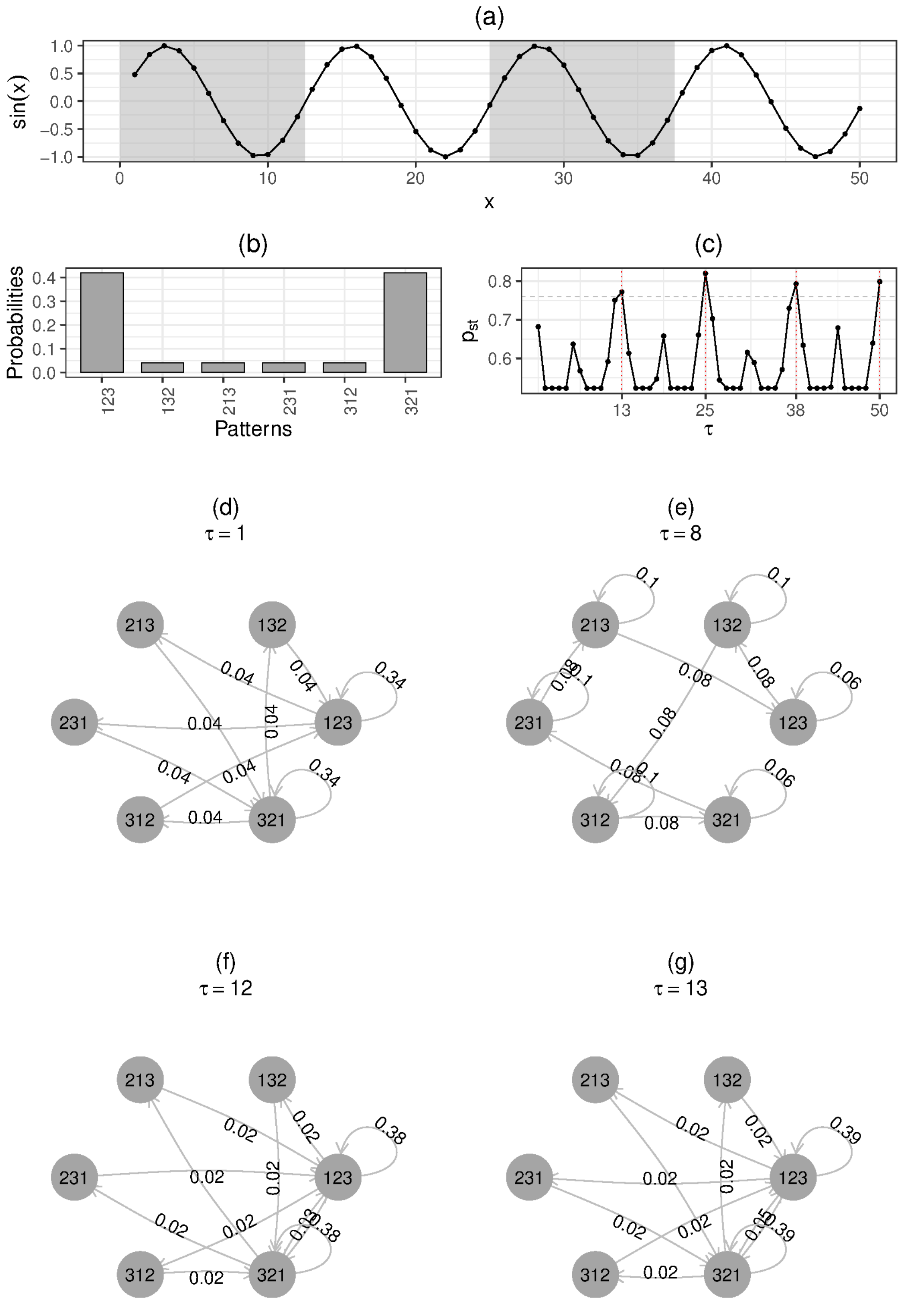


Figure 5.3. Example of (a) the first 50 samples of a sinusoidal time series with period $T = 12.5$, (b) its ordinal patterns probability distribution after the ordinal patterns transformation with $D = 3$ and $\tau = 1$, and (c) its p_{st} as a function of $\tau \in \{1, 2, \ldots, 100\}$. Some G_π for this time series are illustrated, for $D = 3$ and (d) $\tau = 1$, (e) $\tau = 8$, (f) $\tau = 12$, and (g) $\tau = 13$.

To examine this statement, let us consider the synthetic periodic time series, constructed from a sinusoidal function with period $T = 12.5$, in which its first 50 samples are shown in Figure 5.3a. Without loss of generality, let us consider as example the cases for an embedding dimension $D = 3$, which enables a more clear visualization and explanation given the reduced number of $D!$ patterns. The ordinal patterns distribution of this time series is illustrated in Figure 5.3b for $D = 3$ and $\tau = 1$, which reveals nothing regarding the time series period but shows that all possible patterns are observed. In fact, for any value of τ, all patterns will be observed for a large enough time series, and the main differences that may be observed are the number of edges and the probabilities of the transitions, as illustrated in Figures 5.3d–5.3g for different values of τ. According to these figures, we can see a repetition in the behavior of the graphs for different values of τ.

For the case of the sinusoidal series, a simple strategy to obtain the period consists in averaging the intervals between the values of τ where the peaks are higher than a particular threshold. As shown in Figure 5.3c, we have $T = (12 + 13 + 12 + 13 + 12 + 13)/6 = 12.5$ as peaks for the p_{st}, when considering a peak the values higher than a threshold of 0.76. This result is in agreement with [Zunino et al., 2012], where the normalized Shannon entropy and statistical complexity measures are evaluated as functions of τ for periodic time series. For those series, values are close to zero when τ matches the period or multiples of it.

5.6.2 Random time series

Studies in literature have investigated methods for identifying and characterizing randomness in time series [Rosso et al., 2015, 2007a; Ravetti et al., 2014; Rosso et al., 2012; Ye et al., 2017]. Following the visibility graph approaches, random time series are mapped onto random graphs, with an exponential degree distribution [Lacasa et al., 2008; Luque et al., 2009]. While this can be used for a randomness test, in our case, besides this randomness identification, we also want to express the randomness level present at each time series.

In our strategy using the transition graph G_π, we assume a long enough random time series, so missing patterns are not expected to occur [Amigó et al., 2006], and also no missing transition patterns when $\tau > 1$, as described in Theorem 5.4.1. Also, graphs will have a fixed number of $D!$ vertices, and the number of edges will reach their bound $D!^2$, defined in Equation 5.2. In this case, each vertex will have the same degree, so the degree distribution does not reveal anything regarding the time series structure. Thus, we propose using the p_{st} as an indicative to the presence of some randomness, and to suggest at which levels the noise affects it.

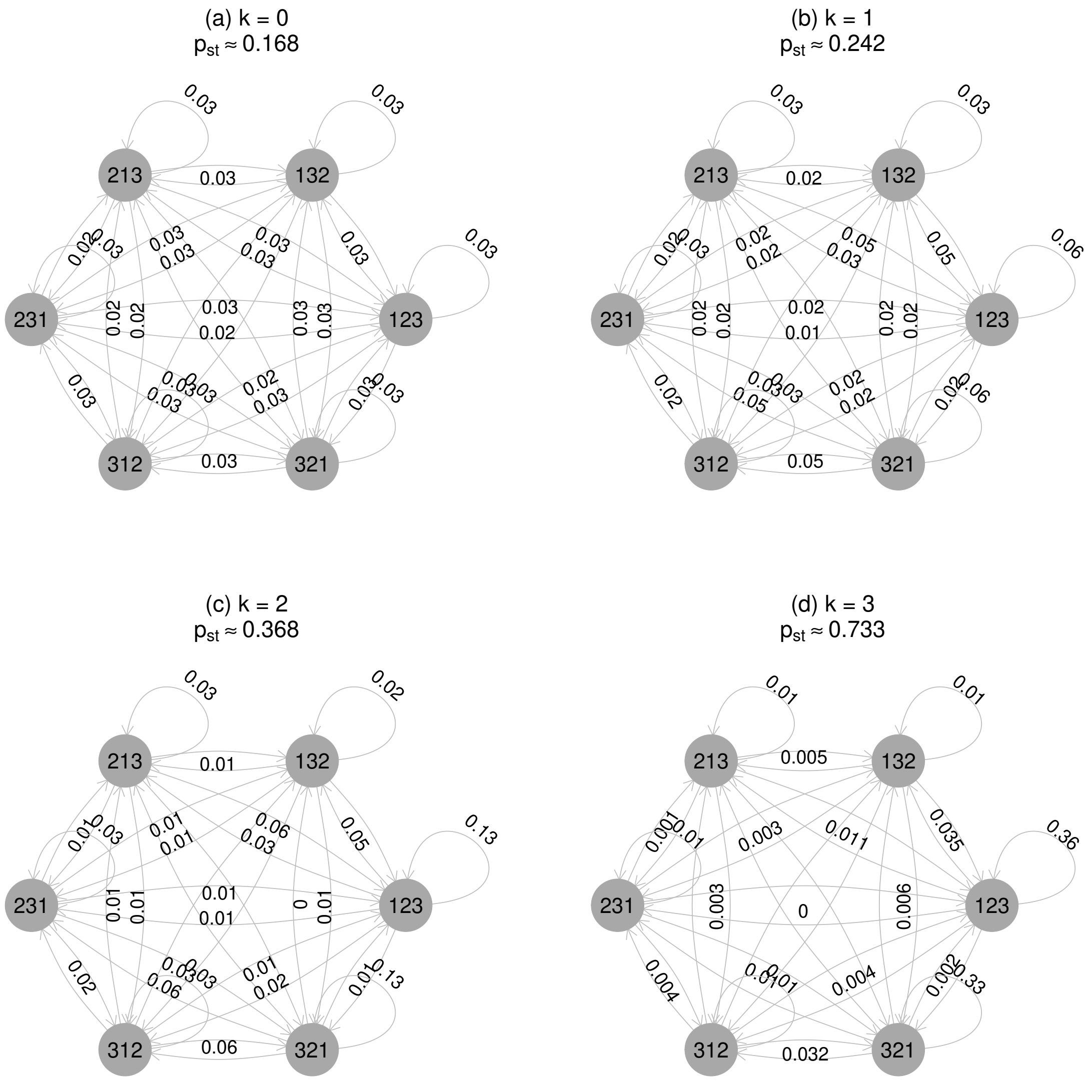


Figure 5.4. Analysis of G_π from the random time series with different correlation levels presented in Figure 3.7. Where (a) is a white noise ($k = 0$), (b) pink noise ($k = 1$), (c) red/brown noise ($k = 2$), and (d) a black noise ($k = 3$). Their correspondent ordinal patterns transition graphs are presented (a) for the white noise ($k = 0$), (b) for the pink noise ($k = 1$), (c) for red/brown noise ($k = 2$), and (d) for the black noise ($k = 3$). Each G_π was constructed from the ordinal patterns transformations computed as the average of 10 runs with different seeds for $D = 3$ and $\tau = 2$. The first 1,000 points were omitted for the time series for better illustrate their randomness after the generation, and some minor transition probabilities (edges weights) were removed from the graphs to clear their visualization.

To evaluate different randomness levels, we used the synthetically generated time series from Figure 3.7, where Figures 3.7a–3.7d illustrate, respectively, a time series with a white ($k = 0$), pink ($k = 1$), red/brown ($k = 2$), and black ($k = 3$) noises.

The transition graphs presented in Figures 5.4a–5.4d and were generated with from the ordinal patterns transformations computed as the average of 10 runs with different seeds for $D = 3$ and $\tau = 2$.

Figure 5.4a shows the ordinal patterns transition graph created from the white noise time series. When $D = 3$ and $\tau = 2$, all the 36 possible transitions occurred, as expected by Equation 5.2, with the same probability $p_{\mathrm{st}} \approx 1/36 \approx 0.03, \forall i, j \in \{1, \ldots, D!\}$. This reinforces the notion that for a completely random time series (and it occurs independently of its marginal distribution) all the patterns [Rosso et al., 2012] and all the transitions between consecutive patterns are expected to occur with the same probability, for long enough time series. Also, as the correlation between observations increases, i.e., $k > 0$, the randomness decreases and is expected that this should be observed in the graphs. In fact, when evaluated with examples of pink, red, and black noises, illustrated in Figures 5.4b–5.4d, p_{st} increases with the noise correlation. The values assumed for these noises are, approximately, 0.168, 0.242, 0.368, and 0.733, respectively.

Following our strategy to better understand these randomness levels, in Figure 5.5 we present an analysis of the behavior of p_{st} for random time series when evaluated as functions of τ. We considered colored noises ranging from $k = 0$ to $k = 3.5$, by steps of $\Delta_k = 0.25$, illustrated as different data points for each k. We constructed the ordinal patterns transformations for $D = \{3, 4, 5, 6\}$ and evaluated them as functions of $\tau = \{2, \ldots, 50\}$. In our analysis, we discarded the case for $\tau = 1$ to avoid the influence of forbidden transition patterns, as previously described in Theorem 5.4.1.

For the particular case of white noise ($k = 0$), there is a special situation where increasing τ does not affect p_{st}, giving approximately $1/D!$ for all combinations of D and τ. However, for $k > 0$ there is a growing tendency in p_{st} as τ increases. It is possible to see a direct relation between p_{st} and its determinism. The more deterministic the series, by the increase of k, the higher the p_{st}. This increasing determinism as a function of τ was also pointed out by Zunino et al. [2012], and we could also verify its influence here. Another point related to this growing tendency of p_{st} is its behavior between different values of D. As Figure 5.5 shows, there is a direct relation between D and an easy distinction of k. For instance, for $D = 3$, we can clearly see a better distinction of different correlation levels for lower values of k. On the contrary, for $D = 6$, lower values of k became hard to distinguish and higher values of k became clearer.

Following our strategy of distinguishing the randomness levels in time series, based on curves of Figure 5.5, one may consider its characterization by analyzing the growing behavior of p_{st}. For this analysis, we quantify the correlation degree of random time series by fitting a regression model for each of these noise curves. For

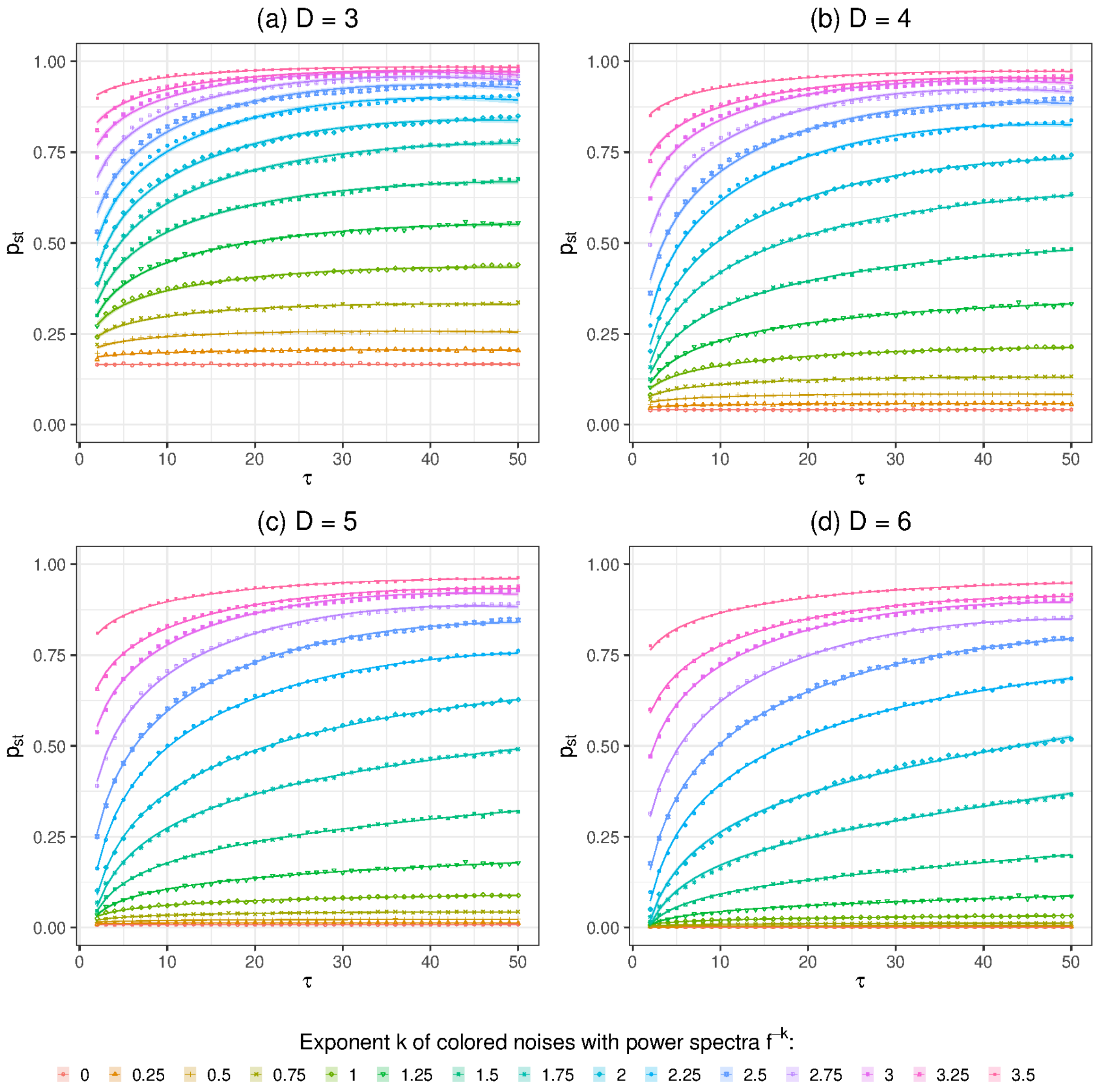


Figure 5.5. Analysis of p_{st} from G_π of different colored noises with power spectra f^{-k}, ranging from $k = 0$ to $k = 3.5$, by steps of $\Delta_k = 0.25$, as a function of $\tau = \{2, \dots, 50\}$, for $D = \{3, 4, 5, 6\}$. Fitted values for regression model are illustrated for each curve as solid lines, with upper and lower limits of 0.95 confidence intervals as shaded areas.

each $D = \{3, 4, 5, 6\}$, we considered the following linear model

$$p_{\mathrm{st}} = \beta_0 + \beta_1 \log \tau + \beta_2 \tau^2 + \varepsilon, \tag{5.4}$$

where $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ is a random error following a normal distribution with zero mean and unknown variance σ^2. This model assumes the presence of an intercept, a logarithmic growing for the curves as τ increases and a quadratic component to better adjust the model. Other models may be used, but this one is satisfactory for our purpose.

The fitted curves for each noise are plotted as solid lines in Figure 5.5, with shaded 95 % confidence regions. The unbiased Mean Squared Error (MSE) of residuals in the regression model fitted for all random noises ranges from 3.58×10^{-8} to 1.7×10^{-4}

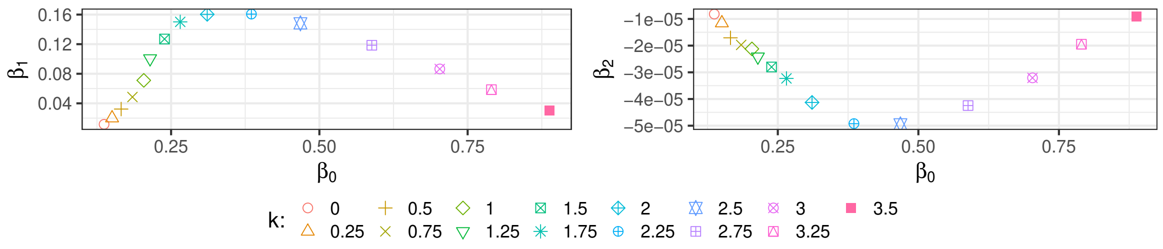


Figure 5.6. Analysis of the parameters β_0, β_1, and β_2 from the regression model presented in Equation 5.4, fitted for the colored random noises of Figure 5.5a, considering the case for $D = 3$.

Figure 5.6 shows the estimated parameters of the model applied to the p_{st} curves of Figure 5.5a, with $D = 3$. Parameters β_1 and β_2 are presented as functions of the intercept β_0. As β_0 increases, β_1 and β_2 present similar behaviors, but in opposite directions. For both cases, each noise level k is placed at different regions, indicating they can be used for distinction.

Thus, to evaluate the model, we performed the classification of time series with different randomness degrees, with the intention to effectively distinguishing among them. The dataset used for this task is composed of 150 time series, with 10 samples for each of the 15 different colored noises with power spectra f^{-k}, ranging from $k = 0$ to $k = 3.5$, by steps of $\Delta_k = 0.25$. The dataset was split in the proportion of 70 % and 30 % of the time series for the training and test sets, respectively, by keeping the same number of time series for each k. For each time series, we computed their p_{st} values for each $D = \{3, 4, 5, 6\}$, as functions of $\tau = \{2, \dots, 50\}$. After fitting the regression model presented in Equation 5.4, we only extract the parameters β_0 and β_1 for the fitted model. The reason for this choice is simply by the similar behavior between β_1 and β_2, as shown in Figure 5.6.

For the classification, we used the Support Vector Machine (SVM) [Cortes and Vapnik, 1995], evaluated with radial (SVMR), linear (SVML), polynomial (SVMP), and sigmoid (SVMS) kernels. Table 5.2 presents the classification accuracy results achieved with this strategy. We can see its success in correctly distinguishing these different random levels. For SVM with both radial and linear kernels, we achieved 100 % accuracy when $D = 3$ and $D = 4$. This highlights the effectiveness and efficiency of the method, since the lower the values of D, the lower the resource consumption.

Table 5.2. Classification accuracy (%) results achieved with our strategy for distinguishing time series with different randomness levels. We present the results of different classification algorithms for $D = \{3, 4, 5, 6\}$.

D	SVMR	SVML	SVMP	SVMS
3	**100**	**100**	93.33	51.11
4	**100**	**100**	97.78	48.89
5	97.78	97.78	95.56	57.78
6	95.56	95.56	88.89	68.89

5.6.3 Chaotic time series

As discussed before, a hard task when dealing with time series from both stochastic processes and chaotic systems is to distinguish them. This is an intriguing aspect since these systems share several properties, even being completely different in concept [Rosso et al., 2007a; Zunino et al., 2012; Ye et al., 2017]. Many studies were proposed to tackle this issue, such as the CCEP method [Rosso et al., 2013, 2007a], which is able to mostly distinguish noisy from chaotic time series, but some chaotic maps and random noises could still be misclassified. As presented in Section 5.2.2, there is no clear separation between those time series within the challenging region in Figure 5.1, so a distance-based algorithm could not be applied to correctly make the distinction among them.

Before presenting the proposed method for distinguishing time series, let us first revisit the previously discussed p_{st} as functions of τ and understand its behavior when considering the chaotic maps. Figure 5.7 analyzes the challenging chaotic maps (composed of 8 chaotic time series) and the Cusp map. In that figure we can also see the fitted line and 0.95 confidence interval region for the regression model presented in Equation 5.4. We see that p_{st} for some of these time series, highlighted in Figure 5.7a, does not necessarily follow a stable behavior as compared to the fitted results for the random time series in Figure 5.5. For instance, the linear congruential generator (4-x), which presents the more challenging placement in CCEP, at the far bottom right of the plane, does not grow with τ.

This leads to a poor fit of the regression model, which can be used as a considerable difference in behavior from the random series. On the other hand, there are chaotic maps where the regression model fits quite well, such as the Holmes cubic map (17-x, 17-y) and the Cusp map (8-x), presented in Figures 5.7b and 5.7c, respectively. If compared to Figure 5.5, we note that both curves and fitted regression model are similar, making their distinction more difficult.

Thus, in order to distinguish chaotic maps from random time series, based on their behavior as a function of τ, we have to consider both the parameters from the regression

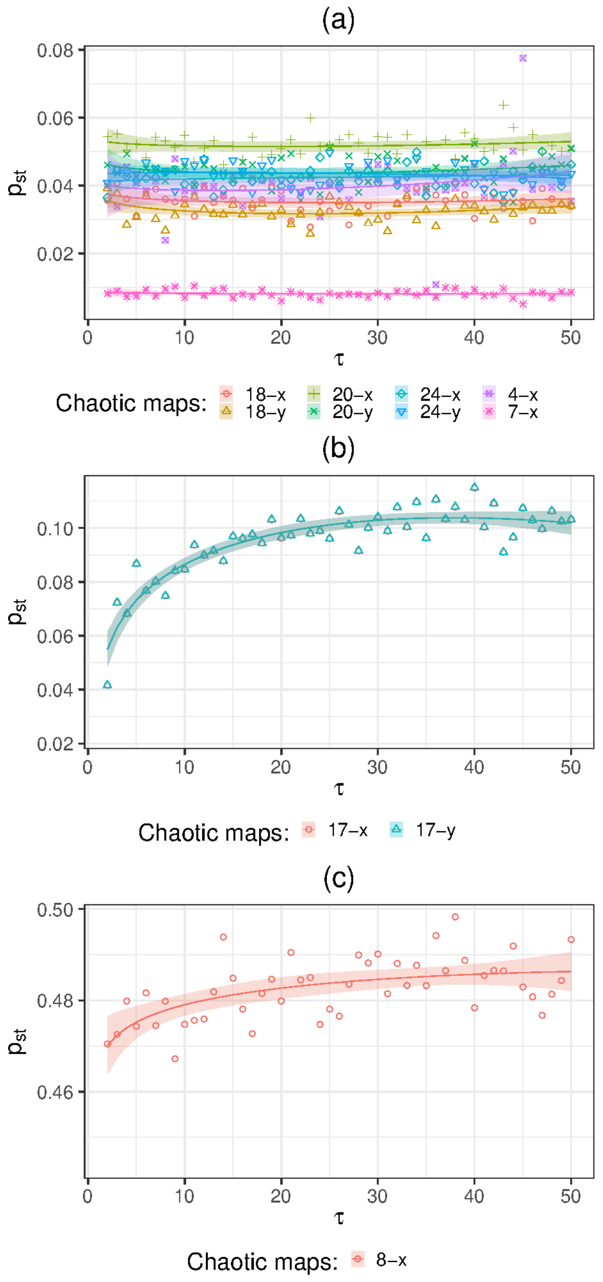


Figure 5.7. Analysis of (a) challenging chaotic maps from Figure 5.1b. These are the linear congruential generator (4-x), Gauss map (7-x), dissipative standard map (18-x, 18-y), sinai map (20-x, 20-y), and Arnold's cat map(24-x, 24-y). The (b) Holmes cubic map (17-x, 17-y) and (c) Cusp map (8-x) are also presented. Curves are computed for $\tau = \{2, \ldots, 50\}$ and $D = 4$. The fitted values for the regression model (Equation 5.4) are shown as solid lines, with upper and lower limits of 0.95 confidence intervals as shaded areas.

model of Equation 5.4 and an analysis of the fitted model. The set of used features comprises the parameters β_0 and β_1, as used for the random time series case, together with the error for the fitted model represented by its coefficient of determination R^2. Thus, we performed an experiment that aims to evaluate our strategy in distinguishing each of the 59 time series presented in the analysis of Figure 5.1 as "Random noise" or "Chaotic map". For each ith time series, we transformed it onto a feature vector $f_i = \{\beta_0^i, \beta_1^i, R^{2,i}\}$, and used the k-means [Macqueen, 1967] clustering algorithm for partitioning the time series into these two clusters. We compute these features for $D = \{3, 4, 5, 6\}$ and $\tau = \{2, \ldots, 50\}$ over time series with 5,000 observations in length, which is sufficient for these values of D.

Table 5.3. Clustering results for distinguishing 59 time series (random process and chaotic systems). The accuracy values and errors for this distinction are also presented.

D	**Accuracy (%)**	**Errors**
3	93.22	0, 8-x, 17-x, 17-y
4	94.91	0, 17-x, 17-y
5	**96.61**	0, 0.25
6	94.91	0, 0.25, 8-x

Table 5.3 presents the clustering results, with the accuracy and errors for this distinction. The best result is achieved for $D = 5$, with just 2 errors out of 59 cases, which gives an accuracy result of 96.61 %. The errors in this case are, exactly, the random noises with correlation degree $k = 0$ (white noise) and $k = 0.25$. In fact, as mentioned in Section 5.6.2, these lower levels of k are the ones with practically constant growing tendency as τ increases, mainly for higher values of D, which give poor fitting for the regression model, thus, being identified as a chaotic map. However, for this case, all the remaining time series were correctly identified. For other values of D, we achieved lower accuracies, 93.22 % for $D = 3$, and 94.91 % for $D = 4$ and $D = 5$, where some chaotic maps were misidentified as random noises. These are exactly the cases illustrated in Figures 5.7b and 5.7c, the Holmes cubic maps (17-x, 17-y) and Cusp map (8-x), respectively, whose behaviors are similar to noises.

These results show the efficacy of the proposed method for distinguishing different time series dynamics, which is a hard task with several proposals found in literature. Also, they highlight the simplicity of the method, which could be easily employed in conjunction with simple learning algorithms, giving reasonable results with low computational complexity.

5.7 Conclusions

The approach based on the p_{st} from G_π, presented in this chapter, provides a valuable tool for characterizing time series. We show that p_{st}, when evaluated as a function of the embedding delay τ, can be successfully applied as an indication of different time series dynamics. For periodic time series, when τ matches the period (or multiples) of the time series, the measure reaches maximal levels, suggesting a strategy to discover it. This confirms the results from previous studies about the observed behavior of information theory quantifiers when also evaluated as a function of τ.

We show, for random time series with different degrees of correlation and colored noises, that p_{st} is directly related to their correlation. It follows a monotonically increasing function of τ for all colored noises except for the white noise ($k = 0$). For this particular case, the measure remains constant for all values of τ, implying a higher randomness level. However, we could precisely distinguish those different randomness levels using a supervised learning (classification) algorithm, reaching 100 % accuracy.

To distinguish random noises from chaotic maps, we show that the behavior of p_{st}, when evaluated as a function of τ, is different for these types of time series. Thus, we map this behavior by fitting a regression model for each curve and show that, while the proposed model almost perfectly fits the random series, it does not properly represent chaotic maps. In fact, by assessing the model parameters and residual errors when fitting, the difference in their behaviors were used to supply an unsupervised learning (clustering) algorithm that correctly identified 96.61 % of the underlying dynamics.

In general, our method is a simpler alternative for the characterization of time series, even short ones (we used here 5,000 data points). However, there are still a few time series in which their dynamics are misclassified. For those cases, an alternative solution for future studies might be the integration between our proposed method with already established techniques, such as other more complex learning algorithms. When applicable, one strategy could solve the drawbacks of the other, yielding a more robust and general solution. Furthermore, the method can also be evaluated when applied to real time series to verify the results when facing new challenges present in real-world scenarios.

Chapter 6

Physical phenomena classification in the Internet of Things

By understanding the dynamics behind a given time series, it is possible to perform its distinction based on their expected behavior, not being subject to the instances of its data points. In this chapter, we present a strategy based on the analysis of time series dynamics for the precise identification of physical phenomena data collected by IoT sensors. In Section 6.1, we introduce the problem and give a contextualization for the proposed work. Section 6.2 discusses the related work and motivates our main problem. Section 6.3 presents a data characterization of the physical phenomena used in this work, and Section 6.4 shows our classification strategy to correctly identify an appropriate sensor monitoring a given physical phenomenon. Section 6.5 concludes the work and presents some open research issues.

6.1 Introduction

When dealing with the massive scale of data from IoT [Stankovic, 2014], a challenging question that arises is *"how to precisely characterize and classify physical phenomena in data collected by a large population of sensors?"* In this question, there are two issues to examine. First, a poor description of both data and resources of most deployed sensors in current IoT solutions, with their sensors deployed all over the world. As presented in Chapter 2, they are described by a simple set of tags and textual information, freely assigned by their owners. This makes the analysis and, consequently, the understanding of the related physical phenomena prone to errors and misunderstandings. Second, the high heterogeneity of sensors brings differences in the resolution and magnitude of their collected data. This may affect distance-based algorithms when making comparisons between data collected by pairs of sensors.

Given such issues, we reinforce the claim that solutions to handle data collected from IoT platforms should avoid the use and the processing of raw data. Instead, it is recommended (i) the extraction of features able to capture data attributes of the physical phenomena, and (ii) the analysis of such features using machine learning techniques (e.g., classification and clustering) [Aghabozorgi et al., 2015]. This has the benefit of minimizing or circumventing the effects of inadequate descriptions and high data variability, as mentioned above.

In order to perform the feature extraction and feature analysis, we will model the data collected from sensors as time series. However, since we are interested in physical phenomena, it is important to properly identify which features to extract. This may lead to a problem in which features can depend on a given instance of the data and not represent the phenomena behind it. For instance, a feature vector of different sensors, even for the same phenomenon, but created at different moments and locations, may lead to completely different values, which may invalidate any further analysis [Wang et al., 2006].

To tackle all these challenges related to the precise identification of physical phenomena in the IoT context, we make the following contributions:

1. We use information theory quantifiers as features to precisely characterize physical phenomena. In this work, we will consider temperature, relative humidity, atmospheric pressure and wind speed, which are physical phenomena typically monitored in IoT solutions. These quantifiers allow us to know the behavior behind a given physical phenomenon;

2. We model data collected from heterogeneous sensors, at different scales and sampling rates, as a unified representation by using the ordinal patterns transformations (Section 3.5). We advance the state-of-the-art of understanding and classifying IoT data by designing strategies based on the phenomena' expected behavior, rather than on the raw data of a sensor itself. This also helps in the scalability problem in IoT by avoiding comparisons with a large number of time series; and

3. We perform the classification of physical phenomena by analyzing their placement on the Causality Complexity-Entropy Plane (CCEP). We compare their placement with previously learned placement regions from known physical phenomena. We also employ a robust method to minimize the lack of a proper distance measure for this plane, helping further investigations and reducing the

gap between the theory from Bandt-Pompe transformations and their application to problems in real-world scenarios.

6.2 Related work and problem definition

Time series analysis is a research topic that gained a new breath in last years, mainly due to the increasing attention for big data, machine learning, and IoT initiatives. A considerable number of algorithms and strategies has been proposed [Aghabozorgi et al., 2015], in which classification and clustering are the most representative ones for the supervised and unsupervised cases, respectively.

These solutions are, basically, divided in those which use raw time series data and those in which some processing is applied before extracting information. The algorithms based on raw data generally compute some distance metric (e.g., Euclidean, Fréchet, Dynamic Time Warping (DTW) [Montero and Vilar, 2014]) between pairs of time series. Those with the lower distances are grouped together (clustering), or identified as similar (classification). However, as previously discussed in Section 3.3.3, for IoT scenarios, in which there is a vast heterogeneity of sensors, some assumptions can not be made. For instance, most of these solutions assume that time series have similar length, or they are sampled at the same constant rate. Furthermore, due to the high number of available sensors, it is not scalable to perform direct comparisons on raw data, making many strategies unsuitable for this scenario.

On the other hand, strategies based on extracting features from the time series seem to be more appropriate to the problem. These solutions are based on the analysis of a new vector of features extracted from the time series [Wang et al., 2006; Fulcher et al., 2013]. One of the first advantages of this strategy is the dimensionality reduction of the time series, which are expected to have a very large length for in IoT. This reduction and equalization of sizes was also the subject of many studies in the literature, via some smoothing technique or symbolic approximation [Lin et al., 2003].

Another point to be concerned in feature extraction is related to the decision of which features to compute. Since we are interested in physical phenomena, this may lead to features that can be dependent on the current instance of time series, not representing the phenomena behind it. For instance, features from different time series, even for a same given phenomenon, may lead to completely different values, which may invalidate any further analysis. For these cases, they may change as a function of time and space, requiring another extraction for new features, that could be computationally expensive [Wang et al., 2006].

Thus, it is reasonable to consider as features those metrics that could represent the behavior of a phenomenon measured by the sensor, independently of the current sample. In this direction, metrics originated from information theory seem to be appropriate to extract this knowledge from a phenomenon. The use of information theory quantifiers has been applied for characterizing time series in several studies. These metrics proved to be effective in distinguishing different dynamic behaviors of time series [Rosso et al., 2007a; Zunino et al., 2010; Rosso et al., 2007b], being useful in the characterization of real-world time series [Aquino et al., 2017; Gonçalves et al., 2016].

In this work, we use information theory quantifiers to characterize the behavior behind physical phenomena. Our aim is to better identify time series by just comparing their behaviors, not their data points. This allows avoiding the comparisons for identification, between a large number of time series, thus, solving the scalability problem in IoT. We also base the extraction of the quantifiers in a transformation from the time series, which is valid for the dimensionality reduction of the data as well as to increase their robustness to IoT problems, discussed in the following sections.

6.3 Characterization of physical phenomena

Focusing on the provision of a better understanding of real-world data, this section presents a characterization of the following physical phenomena: temperature, atmospheric pressure, humidity, and wind speed. As previously discussed in Section 3.5, the ordinal patterns transformation has some properties that make it well suited for its application to real data. Furthermore, using the normalized Shannon entropy (H_S) and the statistical complexity (C_{JS}) as information quantifiers along with the CCEP, allow us distinguishing the different time series behaviors, as presented in Section 3.6.2.

6.3.1 Dataset selection

When dealing with IoT sensors, there is some uncertainty and unreliability that must be considered before dealing with their data. For instance, for sensors available in IoT platforms, such as the case of ThingSpeak, the data and resources usually are not described or, when they do, have a very poor description. Even if the descriptions were given, since there is no validation, there is no guarantee that it is correct. Generally, they are described by a simple set of tags and textual information, freely assigned by their owners and/or users, which makes the search for a keyword or expression prone to errors and misunderstandings [Borges Neto et al., 2015]. In Table 2.1, we presented a summary of this lack of descriptions for the IoT sensors available in the ThingSpeak platform. It highlighted the increasing of these problems in the available sensors at this platform, from the period of 2015 to 2017.

For those sensors where some description was provided, Fig. 2.3 illustrated some of the most frequent words used to describe them. While it can be noticed that most descriptions are, indeed, related to environmental and physical phenomena, they are still really poor. The top-5 most frequent words for the three years are: *temperature*, *humidity*, *temp*, *field*, and *sensor*. These last three are generic descriptors, which add no valuable information to the monitored phenomena.

Thus, given this uncertainty and unreliability of sensors, in order to correctly evaluate our hypothesis, we decided to apply our proposed strategy in those sensors lying in the intersection between CoIoT and IIoT. As discussed in Section 2.3, these are IoT sensors that are maintained by organizations, which can guarantee the quality of data, but are freely available for anyone who are interested. Thus, to investigate the feasibility of using CCEP on the study of physical phenomena, we considered time series data measured by international airport stations from 60 different airports from all regions of the USA territory. Table 6.1 presents the list of these places. Data are measures from the historical weather conditions of temperature, relative humidity, atmospheric pressure, and wind speed, in the period from 2000 to 2015. All data were collected from the Weather Underground[1] platform.

6.3.2 Dataset characterization

The dataset was randomly split in half, where the time series from 30 places was used to the characterization step and the other 30 to validate the classification of the phenomena. To illustrate the behavior of the data we are dealing with, Figure 6.1 gives an example of time series for 2015, measured at the Logan International Airport, Boston, MA. For each phenomenon, it is also showed a Lowess smoothing of the data with smoother span $f = 0.1$.

We can see that the behavior of the data for each different phenomenon is quite different. From a more seasonal behavior for temperature to a more "random" behavior for the wind speed. Figure 6.2 shows the ordinal patterns probability distributions, presented in Section 3.5.2, for the dataset in the period between 2000 and 2015, considering an embedding dimension of $D = 4$. We can see that, for each phenomenon, the ordinal patterns have different probability distributions. This is the behavior that is captured by the aforementioned information quantifiers H_S and C_{JS}.

[1]Weather Underground – `http://wunderground.com`.

Table 6.1. List of places in the USA, ordered by state, used for the characterization and classification phases.

ID	Place	Type	ID	Place	Type
1	Anchorage	F	31	Detroit	C
2	Phoenix	C	32	Minneapolis	F
3	Los Angeles	C	33	Charlotte	C
4	Oakland	F	34	Raleigh	F
5	Ontario	F	35	Omaha	F
6	Sacramento	F	36	Newark	F
7	San Diego	F	37	Albuquerque	F
8	San Francisco	C	38	Las Vegas	C
9	San Jose	F	39	Buffalo	F
10	Santa Ana	F	40	New York (Central Park)	C
11	Denver	C	41	New York (JFK)	F
12	Hartford	F	42	New York (LaGuardia)	C
13	Fort Myers	C	43	Cleveland	C
14	Fort Lauderdale	C	44	Columbus	F
15	Jacksonville	F	45	Cincinnati	F
16	Miami	C	46	Portland	C
17	Orlando	F	47	Philadelphia	C
18	Tampa	C	48	Pittsburgh	F
19	West Palm Beach	F	49	Nashville	F
20	Atlanta	C	50	Austin	C
21	Honolulu	C	51	Dallas	C
22	Kahului	F	52	Dallas (Fort Worth)	C
23	Chicago (Midway)	C	53	Houston (Bush)	F
24	Chicago (O'Hare)	C	54	Houston (Hobby)	C
25	Indianapolis	F	55	San Antonio	C
26	New Orleans	F	56	Salt Lake City	F
27	Boston	C	57	Washington (Reagan)	C
28	Baltimore	C	58	Washington (Dulles)	F
29	Kansas City	F	59	Seattle	C
30	St. Louis	C	60	Milwaukee	F

Legend: C - Used for Characterization / F - Used for Classification

To better understand these behaviors, Figure 6.3 presents the CCEP for the whole time series, considering different embedding dimensions $D = \{4, \ldots, 7\}$. The rightmost region of CCEP (near $H_S = 1$ and $C_{JS} = 0$) represents a totally random behavior such as a white noise. On the other hand, the time series that present strong regularity and more correlation between neighboring values tend to lie in the leftmost part of CCEP (near $H_S = 0$ and $C_{JS} = 0$). The region of high C_{JS}, i.e., the upper-center region of the plane represents time series with chaotic behavior [Rosso et al., 2007a].

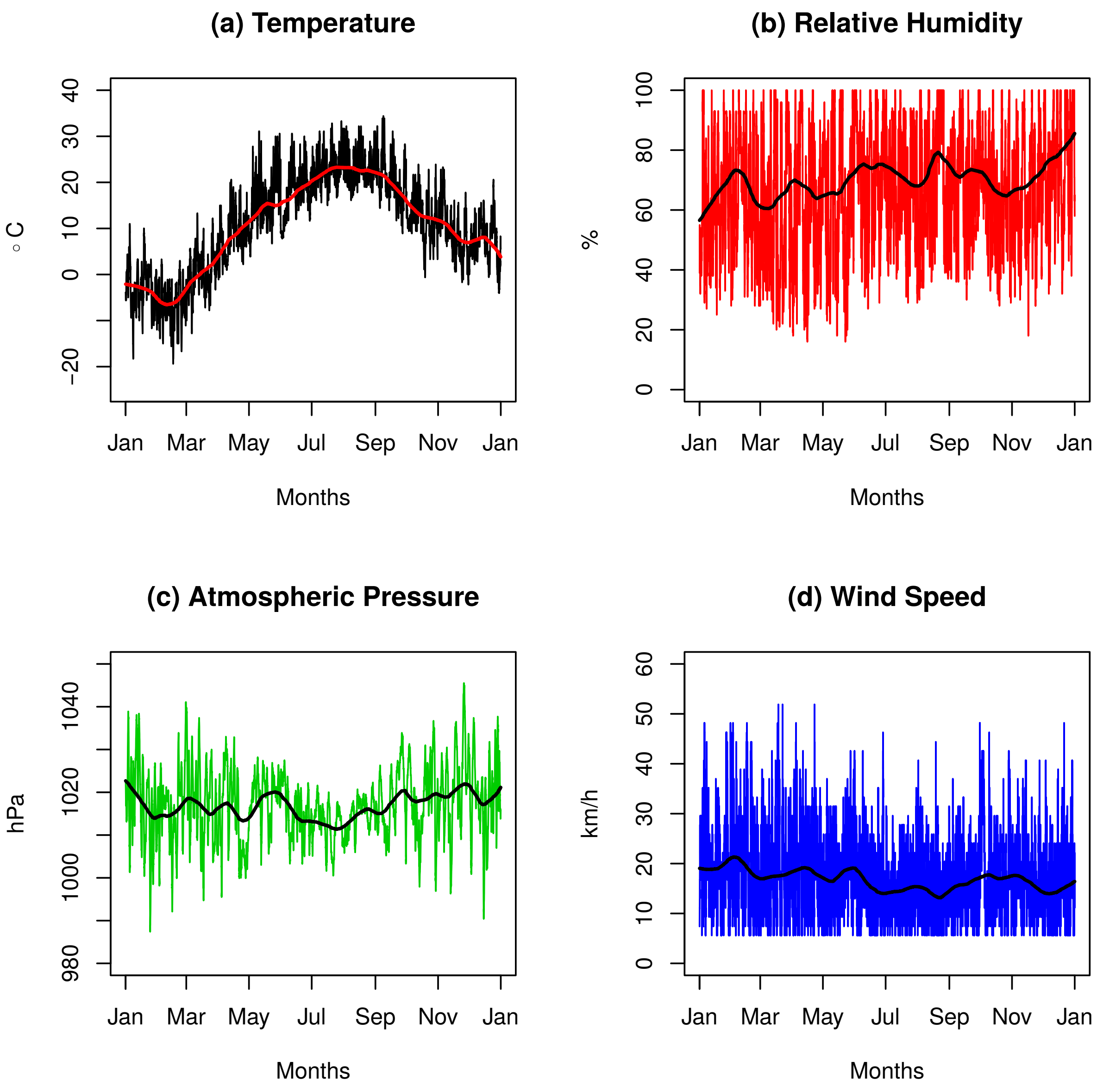


Figure 6.1. Historical data of 2015 from the Logan International Airport, Boston, MA. Phenomena (a) temperature, (b) relative humidity, (c) atmospheric pressure, and (d) wind speed, along with a Lowess smoothing ($f = 0.1$).

Figure 6.3 shows that, for all values of D, all phenomena lied in the region of medium-high entropy values ($0.6 < H_S < 1$), which are similar to the region that characterizes "colored" random noises, representing different correlation values in the time series structures [Rosso et al., 2007a]. Also, as D increases, the regularities of the phenomena are better captured by the information quantifiers. This can be expressed by the increasing of C_{JS}, the decreasing of H_S, and the more clear separation between the points for each phenomenon in the plane.

Following our analysis, in order to verify the feasibility of this method in characterizing the physical phenomena *per se*, Figure 6.4 presents the CCEP representation

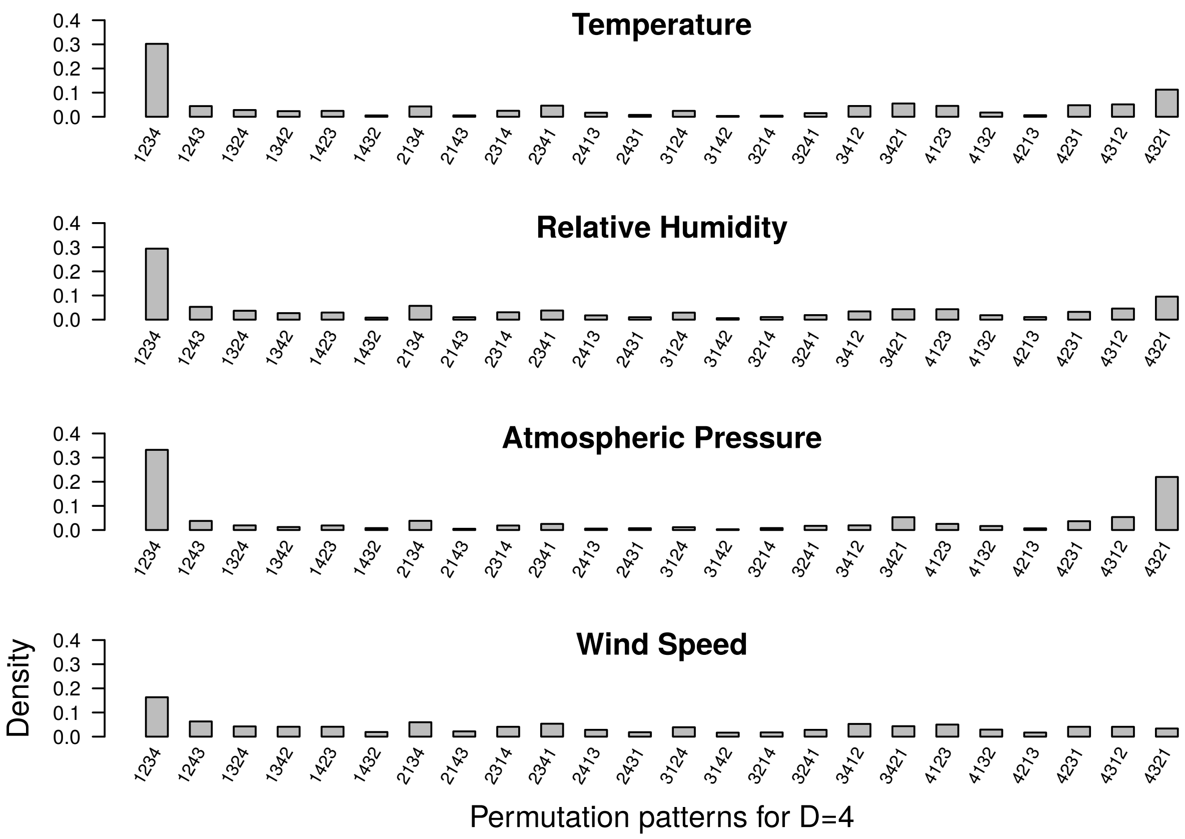


Figure 6.2. Ordinal patterns probability distributions of Boston's weather historical data in the period between years 2000 and 2015. All plots considered $D = 4$.

for all phenomena studied herein, from the 30 places used for the characterization phase, according to Table 6.1, with an embedding dimension $D = 6$. The reason for choosing $D = 6$ is due to a trade-off between a concise capture of the regularities of the phenomena and the small value of $D!$, which impacts on the required length of the time series.

Fig. 6.4 also shows that, with exception for the atmospheric pressure, all the phenomena lie at a specific region in the plane and has an "expected behavior", described by the shape in which the points are scattered. The time series for the atmospheric pressure present different behaviors in different places, and are more spread over the plane.

Fig. 6.5 illustrates the two most extreme time series, i.e., the ones which are farther from their cluster regions when considering temperature and atmospheric pressure. We can see that, while there is a small variation between the temperatures of Denver and Phoenix, for the atmospheric pressure, this difference is more apparent, and the time series of Denver seems to be more noisy than that for Honolulu. This difference on behavior results in different values of the information quantifiers and,

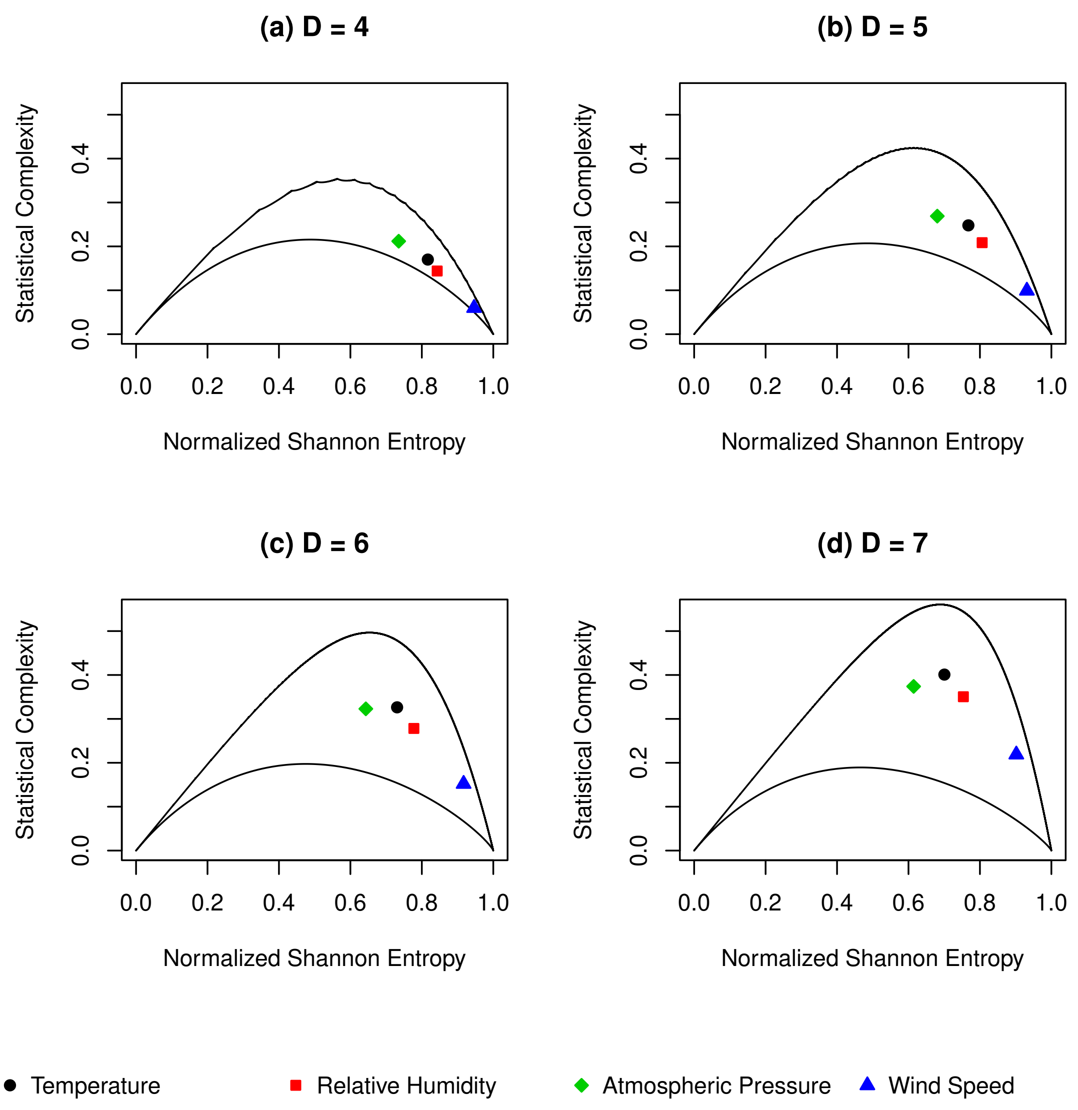


Figure 6.3. CCEP of Boston's weather historical data between years 2000 and 2015, for $D = \{4, \ldots, 7\}$.

thus, different placements in CCEP. This may lead to the need of a study about the geographical influence on these measures, which we will conduct as a future work.

6.4 Classification of Physical Phenomena

In this section, we present our strategy for the classification of physical phenomena from the IoT sensors. The first step is to learn the expected behaviors from different phenomena and, hence, identify their placement regions in the CCEP by clustering their placements. Thus, we will be able to verify if a given time series are similar to an

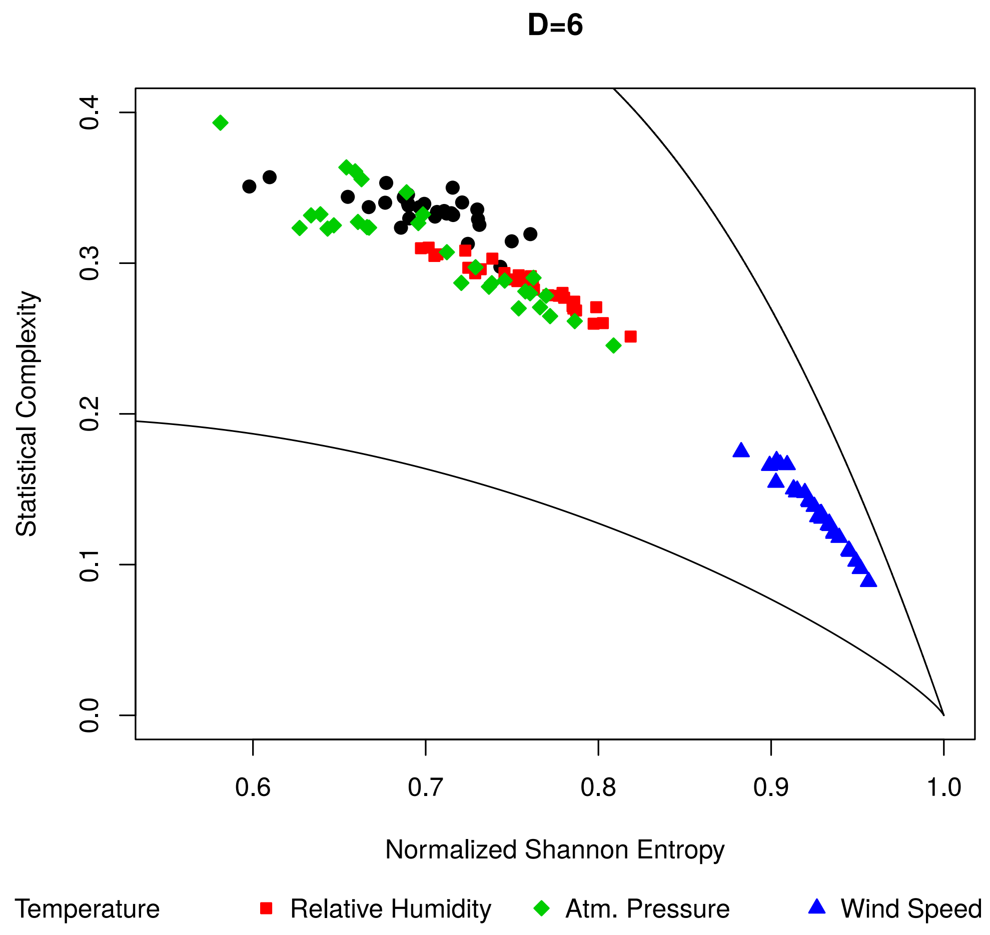


Figure 6.4. CCEP of historical weather data in the period between years 2000 and 2015 of all the places used for characterization, with $D = 6$.

already known phenomenon, by comparing the distance of its placement in the plane to the previous learned placements.

6.4.1 Identifying regions on CCEP

To define the expected placement region in the CCEP for a given phenomenon, we have to consider their concentration of points in the plane and estimate a centroid to represent it. Figure 6.6 presents a heat map, built with a Kernel Density Estimation (KDE) from the points showed in Figure 6.4, illustrating the concentration of points in CCEP for the phenomena under analysis. We can see that, for temperature, relative humidity and wind speed, there is a regularity in the concentration of points around a particular natural centroid. Furthermore, they form three different grouped clusters.

For the case of atmospheric pressure, the points are more spread and two centroids are highlighted, resembling a bimodal data. Issues related to mixture models to fit bimodal data will not be covered in the present work and will be the subject of a

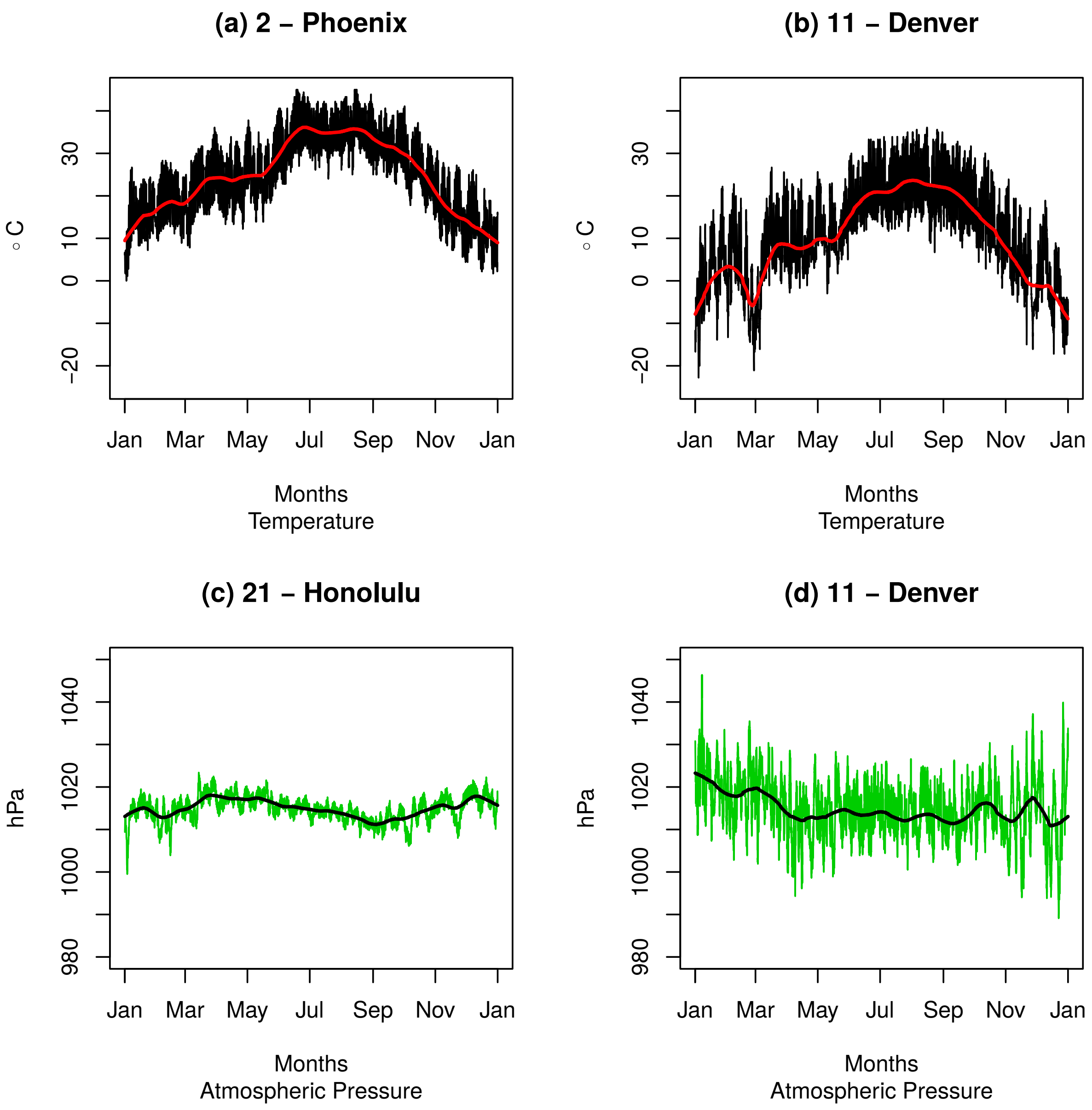


Figure 6.5. Examples of historical data for 2015 of the most divergent places in the CCEP. For the temperature of (a) Phoenix and (b) Denver, and the atmospheric pressure for (c) Honolulu and (d) Denver, with a Lowess smoothing ($f = 0.1$).

future study. For our current purposes, it is sufficient to identify the centroids in which the points are surrounding and accept the fact that the atmospheric pressure can be characterized by two different regions on the plane.

Another point we must also be concerned when discovering the regions in the plane is about the effects of noise and imprecision in the information quantifiers. Thus, before obtaining the centroids, we first apply the Skinny-dip clustering algorithm [Maurus and Plant, 2016] on the points for each phenomenon. Skinny-dip is a noise-robust clustering algorithm based on the Hartigan's dip test of modality and is able to rea-

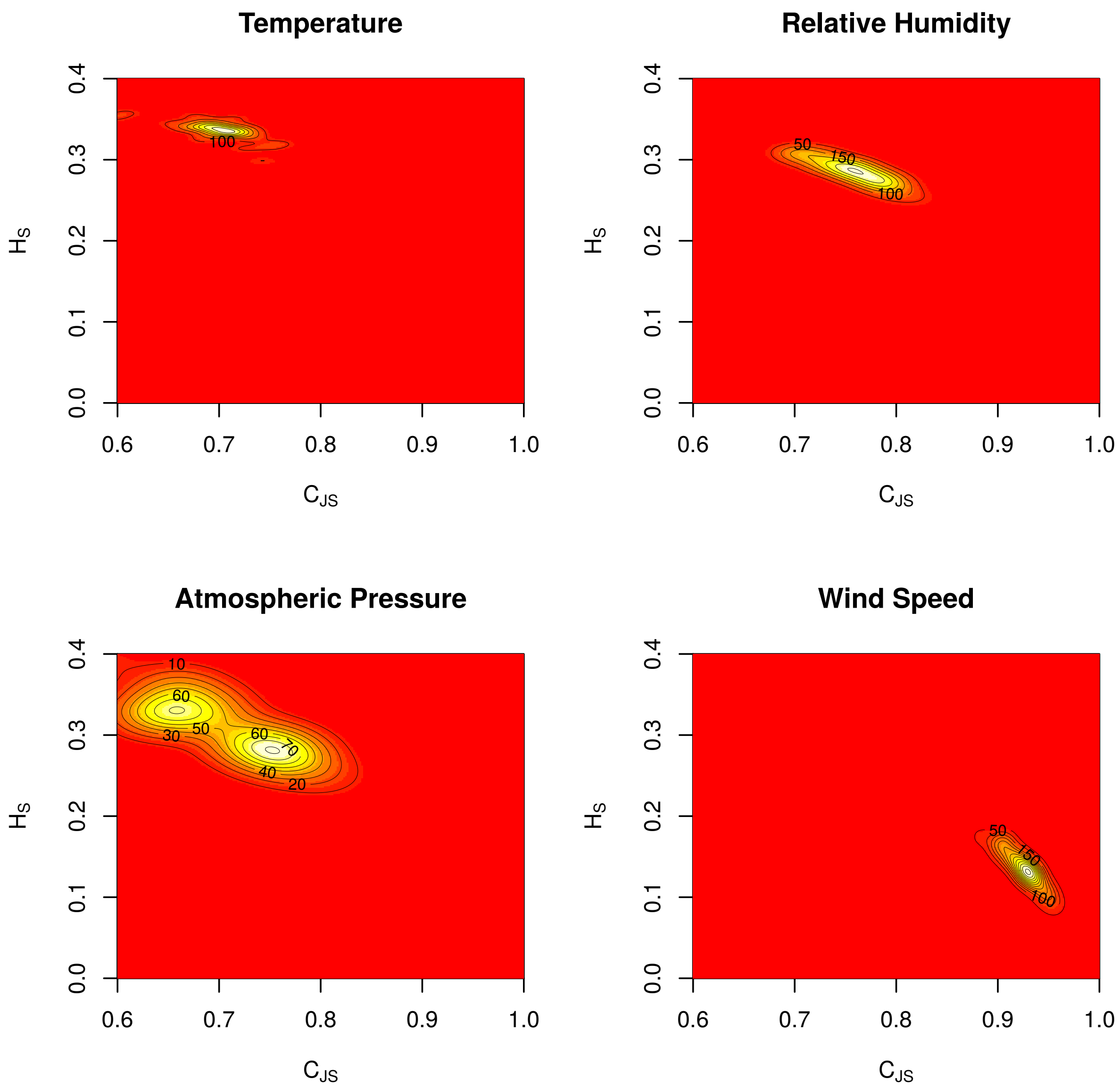
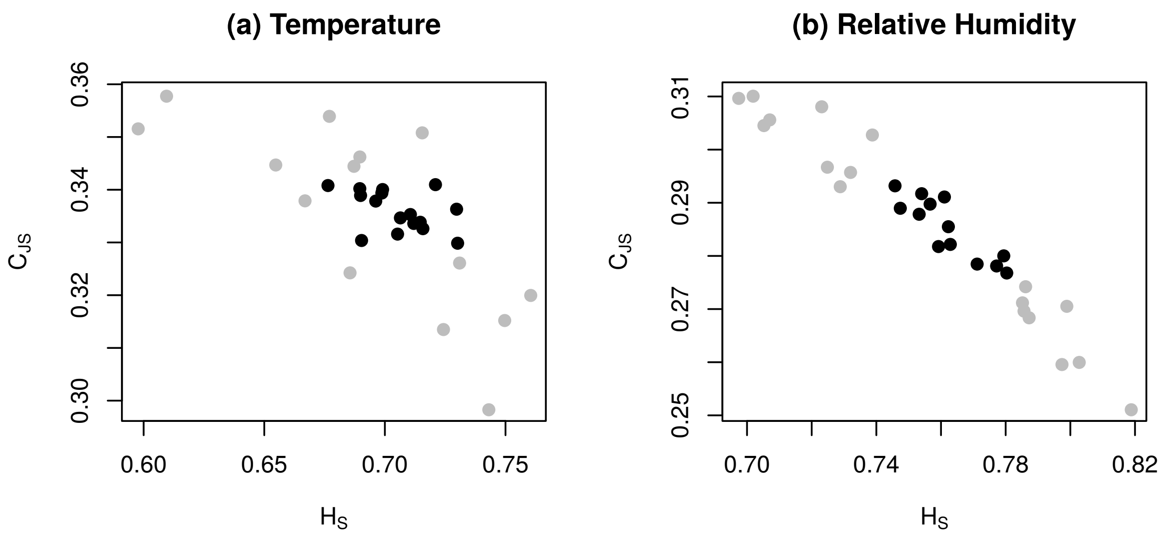


Figure 6.6. Heat map illustrating the concentration of points from information quantifiers related to the characterization of the temperature, relative humidity, atmospheric pressure, and wind speed phenomena, with $D = 6$.

sonably detect the more distinguishing concentration of points in a given region, even under a rate of 80 % of noise [Maurus and Plant, 2016].

Figure 6.7 depicts the resulting clusters after the Skinny-dip processing for the points of each phenomenon. Gray points are those considered noisier by the algorithm. We can see that, for the atmospheric pressure points, there are clearly two formed clusters. After discovering the most significant points, we performed a KDE to each cluster and interpolate between their points to compute the centroids. Table 6.2 illustrates the values of the centroids computed for each discovered cluster.



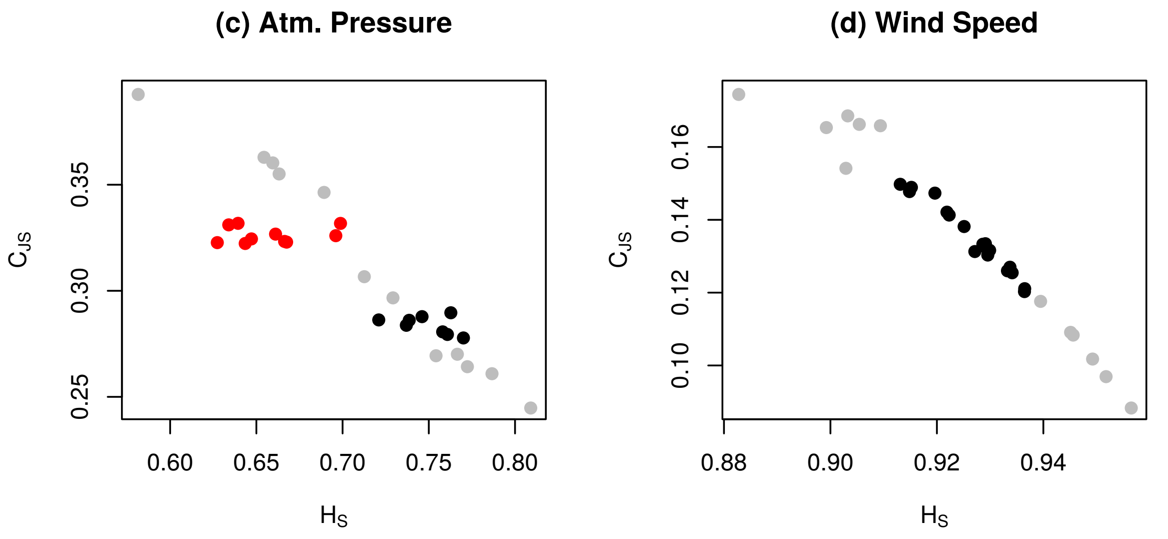


Figure 6.7. Most relevant points discovered by Skinny-dip clustering algorithm for the physical phenomena, with $D = 6$.

6.4.2 Classification of time series in the CCEP

The next step towards the classification of a given time series is to place it in the CCEP and verify if it lies close to some know phenomenon placement. A point to consider here is that the notion of distance in this plane is still an open research question. For instance, although the values for H_S ranges from 0 to 1, the values for C_{JS} are bounded by their limits, impacting on direct distance metrics. This occurs because C_{JS} behavior is governed by patterns in the probability distribution space that gives, for the same entropy, different levels of statistical complexity.

However, to illustrate the feasibility of the method for the classification of physical phenomena, we used the Euclidean distance, despite not being the most appropriate,

Table 6.2. Centroids of CCEP regions for each phenomenon, from a kernel density estimation on the most relevant points.

Phenomenon	H_S	C_{JS}
Temperature	0.711	0.334
Relative Humidity	0.755	0.290
Atmospheric Pressure 1	0.739	0.286
Atmospheric Pressure 2	0.658	0.324
Wind Speed	0.930	0.131

to compute the nearest centroid for the time series of the places present in our dataset. We show the classification results in terms of Accuracy (A), true positive rate per class (tp) and false positive rate per class (fp), where $A = \frac{tp+tn}{tp+tn+fp+fn}$, where tn is the true negative rate and fn is the false negative rate.

Table 6.3 summarizes the results for the classification process using the CCEP method with the Euclidean distance (CCEP$_E$) to discover the type of the time series. To perform this identification, we compute the Euclidean distance between the position of a given time series in the plane to the already known centroids, showed in Table 6.2. We use a simple classification technique that assigns a given time series to the same type of the closest centroid.

Table 6.3. Results of the number of true positive and false positive identifications for the physical phenomena time series with the CCEP method for the Euclidean distance (CCEP$_E$).

Metric	**Value**
Accuracy (A)	0.75 (90/120)
Accuracy (without Atm. Pressure)	0.93 (84/90)
Temperature (tp)	0.83 (25/30)
Humidity (tp)	0.67 (20/30)
Pressure (tp)	0.53 (16/30)
Wind Speed (tp)	0.97 (29/30)
Temperature (fp)	0.10 (9/90)
Humidity (fp)	0.10 (9/90)
Pressure (fp)	0.12 (11/90)
Wind Speed (fp)	0.01 (1/90)

For the two general configurations we consider the cases where the atmospheric pressure time series, as their centroids, were present in the experiment and not. We can see that, since the atmospheric pressure phenomenon was the most difficult to estimate its behavior, when it is included in the experiment the number of correct identifications is about 75 %, which means specifically a number of 90 out of 120 in total. The total of 120 series is from the 30 places used for classification, such that each place has four time series for their phenomena. Without the pressure being considered, this number increases to 93 %, 84 out of the 90 total time series were correctly classified.

For each individual phenomenon class, we can see a reasonable total of true positives, with the wind speed being the most correctly identified, with a total of 97 %. The atmospheric pressure has the lowest true positive rate, only 53 %. This result is indeed expected, since the behavior of the pressure phenomenon was the hardest to be characterized. On the other hand, as the most stable phenomena in the characterization, the wind speed was the one with the best results.

Another important aspect for this classification is regarding the false positive values. In fact, according to the application that will be using an IoT sensor, maybe worse than not finding a sensor to be used is to find a wrong sensor. For this sort of problems, the method also seems to be reasonable with the highest rate of wrong identification being 12 % for the atmospheric pressure phenomenon. Furthermore, as mentioned before, even with this promising results, the Euclidean distance is not the most appropriate for the current method. It is expected that, with a proper distance metric to this plane, these results will be improved.

6.5 Conclusions and future directions

In this chapter, we proposed the application of information theory quantifiers, namely the normalized Shannon entropy and statistical complexity, to extract knowledge regarding the expected behavior of physical phenomena in the context of IoT. We showed that the time series dynamics, obtained by the ordinal patterns transformation, and the analysis of CCEP plane provide a robust approach to face the challenges related to this particular scenario.

To perform the classification of the physical phenomena, we proposed a definition of regions within the plane. Those placement regions were defined by the application of a noise-robust strategy for finding their centroids, which resulted in significant quality for the process. All these contributions clearly advance the state of the art in the characterization and classification of physical phenomena in IoT.

As future work, we open several questions related to the advances in the definition of distance metrics in the CCEP space, the need for studying the geographical influence on this information measures, and novel approaches to minimize the effect of IoT related problems in the correct characterization and identification of physical phenomena.

Chapter 7

Time Series Classification via Class Separability Maximization

The classification of time series is a fundamental research topic that has gained a new breath in the last years, mainly due to the increasing needs from big data and Internet of Things (IoT) communities in better understanding the data they are dealing with [Tsai et al., 2014; Sezer et al., 2018; Mohammadi et al., 2018]. A considerable number of strategies was already proposed for the task of time series classification in general [Bagnall et al., 2017]. However, for the challenging scenario of IoT, the use of traditional methods is not always possible [Borges Neto et al., 2015].

In this chapter, we present the Time Series Classification via Class Separability (TSCLAS) strategy for the IoT data, which is based on the class separability analysis of the time series temporal dynamics. Section 7.1 introduces the problem and motivates our contribution. In Section 7.2 we discuss studies related to our proposal, focusing on class separability strategies and the use of the ordinal patterns transformations in other problem domains. Section 7.3 presents our proposed class separability measure for time series dynamics, and Section 7.4 introduces TSCLAS, our strategy for using these measures in the time series classification. Section 7.5 shows our experiments and results. Finally, Section 7.6 presents our final conclusions and future directions.

7.1 Introduction

To process IoT data from different sources, one must consider handling large time series data generated at different rates, of different types and magnitudes, possibly having issues concerning uncertainty, inconsistency, and incompleteness due to missing readings and sensor failures [Borges Neto et al., 2015; Qin et al., 2016; Karkouch et al., 2016; Sezer et al., 2018; Liu et al., 2020]. These problems directly affect data quality and, consequently, make it harder, or even impracticable, to apply traditional classification methods to them [Borges et al., 2019b]. For instance, missing readings create gaps within the time series, which may require a prior imputation step on the raw data [Wu, 2021]. However, it can be impracticable as the gap increases, affecting the performance and scalability of classification methods. Furthermore, the randomness of missing readings create time series with different lengths, while most classifiers assume time series of equal length.

In this work, we propose TSCLAS, a time series classification strategy for the IoT data, based on the class separability analysis of their temporal dynamics. Since TSCLAS must be suited for IoT scenarios, in addition to the requirements for the classification task itself, it is designed to be robust to the following challenges: (i) the large length and (ii) incompleteness of IoT data. For both challenges, we follow the claim that solutions to handle data from IoT sensors should avoid using raw data, considering their transformation to another representation domains, which are less sensitive to those problems [Borges et al., 2019b]. To consider the first challenge, we follow a fast and scalable method to transform the large time series in a small set of features, reducing the processing time of a further classification strategy. Consequently, for the second challenge, we show that, with the proper transformation on the raw data, we are able to mitigate the problems derived from the incompleteness of data, and propose a more robust time series classification approach.

Our proposal is based on the ordinal patterns transformation, detailed in Section 3.5, which is a representation domain for the characterization of data according to its dynamics, indicating how a system evolves over time [Rosso et al., 2015; Borges et al., 2019a]. By considering its dynamics, it is possible to evaluate its data generating process, giving us the ability to capture the inner characteristics of the underlying phenomena, which is a feasible strategy for dealing with the issues of IoT data. The ordinal patterns transformation, proposed by Bandt and Pompe [2002], is a powerful tool to evaluate time series dynamics, that has been extensively discussed and applied to real-world studies [Rosso et al., 2013; Aquino et al., 2015; Rosso et al., 2016; Aquino et al., 2017; Ribeiro et al., 2017]. Furthermore, this transformation focuses on the simplicity and the fast calculation, being invariant to differences in magnitudes, which are essential aspects of our scenario. However, despite the transformation's ability for the characterization and distinction of time series dynamics [Rosso et al., 2007a, 2013], when applied to real world challenging data, which is the case for IoT, some aspects must be considered in order to increase its classification potential. An essential aspect is, thus, the choice of transformation parameters (D,τ), which is an important step in the classification process.

The importance of choosing the best parameters for the ordinal patterns transformation is clear, but most studies in the literature use off-the-shelf values. For instance, the choice of the best embedding dimension D is still an open question, where the current solutions follow the recommendations from literature [Bandt and Pompe, 2002], by choosing $D \in \{3, \ldots, 7\}$. The main aspect to consider here is that the chosen D depends on the time series length n, satisfying the condition that $n \gg D!$. On the other hand, for the embedding delay, a common choice is to assume $\tau = 1$. However, although these values are enough for a controlled scenario, it may impact the classification results for the challenging IoT data. In this work, we investigate the selection of these parameters, discussing the main conditions for the embedding dimension D and providing a strategy for finding the most appropriate embedding delay τ.

Thus, for given a train/test dataset, TSCLAS performs a strategy for selecting the best parameters for the ordinal patterns transformation, based on the maximization of the class separability of the time series dynamics within the training dataset. It is important to mention that this is not a hyperparameter tuning approach, where the best parameters are defined by their classification results assessed on a different validation set. Instead, we only consider the training dataset and its classification potential given the separation of their classes for different parameters. Our class separability analysis is performed on the Causality Complexity-Entropy Plane (CCEP) [Rosso et al., 2007a, 2013], which is detailed in Section 3.6.2. We show that the best distinction of time series dynamics within the CCEP occurs whenever there is a clear separability between the time series classes, since the distribution of their points does not intersect with others. Thus, our strategy consists of choosing the best parameters that maximize the class separability of the distribution of points for the classes on the plane.

Thus, once the parameters are chosen, TSCLAS is applied for the classification of IoT data, based on a combination of features extracted from the ordinal patterns probability distribution and the ordinal patterns transition graph, both derived transformations from the set of computed ordinal patterns. As shown in the following, it is scalable to the length of time series and robust to missing data gaps, common issues from IoT context not properly covered by traditional classification strategies.

In summary, in this work, we present the following contributions:

1. a class separability measure for time series dynamics, according to their displacement on the CCEP, indicating its potential for correctly distinguishing the time series,

2. a method for choosing the best parameters for the ordinal patterns transforma-

tions according to the maximization of the separability measure of time series classes, and

3. a time series classification strategy for IoT data, which is scalable to time series length and robust to missing gaps, through features extracted from the ordinal patterns transformations.

7.2 Related Work

The top-rank state-of-art solutions for time series classification consist of general-purpose classifiers, independently of the type and application domain of the time series. One of the first proposals with significant results classification that remains relevant nowadays is the Time Series Forest (TSF) classifier [Deng et al., 2013], which is an ensemble algorithm that improves the well-established Random Forest (RANDF) method. The use of an ensemble of classifiers is a common basis for these strategies, which is the case of the Bag-of-SFA-Symbols (BOSS) [Schäfer, 2015], the Ensemble of Elastic distances (EE) [Lines and Bagnall, 2015], and Random Interval Spectral Forest (RISE) [Lines et al., 2018] classifiers. Following this approach, the solutions with the most significant results, but that also take the ensemble method to its extreme, is the Flat COllective of Transformation-based Ensembles (Flat-COTE) [Bagnall et al., 2015], with 35 classifiers in its ensemble, followed by its hierarchical version, the Hierarchical Vote COllective of Transformation-based Ensembles (HIVE-COTE) [Lines et al., 2016]. However, most of these strategies require a considerable processing time for training its classifiers, even for small datasets, which are not adequate for IoT scenarios. Consequently, a simpler classification algorithm, such as the K-Nearest Neighbors (KNN), is a feasible solution for scenarios with large time series. The interested reader might refer to the "bake of" work of Bagnall et al. [2017] for more details regarding these classifiers.

For the challenging IoT time series, unlike the general-purpose classifiers, the proposed solutions for classifying IoT data are very domain-specific, making them hard to reproduce in different contexts. For instance, Montori et al. [2018b] proposed a classification strategy considering a scenario where some textual information to describe the IoT data is provided. However, not all sensors have a description available, making this approach unfeasible. Another aspect usually considered when classifying IoT data, as surveyed by Wu [2021], consists of strategies that are robust to their issues, mainly the data incompleteness. In this direction, Postol et al. [2019] proposed a strategy for the classification of noisy and incomplete IoT data based on topological data analysis

(TDA). However, although their solution achieves good results, their proposal is well suited for time series that spans over months, with the best results with one month for training and eight months for testing. Although not directly related to the time series classification task, some strategies try to be robust to data imperfections in the realm of industrial IoT time series. Chen et al. [2019] proposed a strategy for predicting future values of time series even in the presence of missing data, based on a variational inference algorithm for a kernel dynamic Bayesian network (KDBN). In the same direction, Bekiroglu et al. [2020] proposed a strategy for predicting the output measure of a sensor by combining multiple inputs, which can handle the case when there are imperfections in these inputs.

The ordinal patterns transformation, which is detailed in Section 3.5, consists of transforming a time series into a set of ordinal patterns, according to two parameters: the embedding dimension D and the embedding delay τ. Roughly speaking, the parameter D is used for constructing sliding windows of size D, which will define the ordinal patterns, and the parameter τ corresponds to the time scale interval used to sample the consecutive points of those sliding windows.

This transformation was successfully applied by many works in literature, from a more theoretic perspective of discriminating time series dynamics, such as noise from chaos [Rosso et al., 2007a; Zunino et al., 2012; Rosso et al., 2013; Ravetti et al., 2014; Kulp and Zunino, 2014; Rosso et al., 2015; Borges et al., 2019a; Olivares et al., 2020] to a more applied approach for the characterization of diverse time series dynamics, such as vehicular behavior [Aquino et al., 2015], electric load [Aquino et al., 2017], analysis of ECG data [Kulp et al., 2016], analysis of physiological signals [Wang et al., 2016], and the classification of handwritten signatures [Rosso et al., 2016] and physical phenomena [Borges et al., 2019b].

The choice of parameters (D,τ) for the ordinal patterns transformation is an important part of such studies, but most of them use their off-the-shelf values. Following the recommendations from literature [Bandt and Pompe, 2002], the embedding dimension is chosen from $D \in \{3, \dots, 7\}$, which must consider the time series length n, such that $n \gg D$. For the embedding delay, it is common to just assume $\tau = 1$, and the sliding windows are sampled from consecutive points from the time series. In fact, choosing the best parameters in this context is not easy, since it may consider different approaches according to the task, such as finding periodicity in data and estimating the scale of delayed systems [Zunino et al., 2012]. However, as shown by Zunino et al. [2012] and Borges et al. [2019a], it is possible to discover important characteristics of the time series when evaluating the extracted metrics as a function of τ. Consequently, different values of (D,τ), instead of the off-the-shelf ones, may be used to achieve better results in time series classification, by revealing their similarities and distinctions.

Our proposal for selecting the best parameters for the ordinal patterns transformation is based on the class separability of time series dynamics. The separation of classes is a classical question that arises in many distinct areas. From molecular biology [Unger and Chor, 2010; Mao and Wenyin Tang, 2011] to astronomy [Zheng and Zhang, 2008; Makhija et al., 2019], whenever there is a graphical representation of points that belong to different classes, there is the need to understand to what extent we can separate a particular subset from the others [Cerdeira et al., 2012; Wang and Lei Wang, 2008]. The class separability is a measure used to identify the regions of different classes, which can be used to distinguish them [Thornton, 1998; Cerdeira et al., 2012]. It consists of geometric analysis of the space where points are distributed, which tries to find regions or boundaries that separate each class [Thornton, 1998]. This generalizes the linear separability concept. The Thornton's Geometric Separability Index (GSI) [Thornton, 1998] was one of the first defined separability measures. It is defined as the proportion of data points whose classification labels are the same as those of their nearest neighbors, accounting for the degree to which inputs associated with the same output tend to cluster together [Greene, 2001]. This method has shown that it is possible to empirically analyze the predictability at the core of the learning methods. This led to novel class separability measures that have been proposed for different problems [Zheng and Zhang, 2008; Wang and Lei Wang, 2008; Unger and Chor, 2010; Mao and Wenyin Tang, 2011; Cerdeira et al., 2012; Makhija et al., 2019].

In this work, we apply this concept of class separability analysis to the graphical presentation of time series dynamics from the causality complexity-entropy plane. The CCEP representation has been extensively discussed in recent studies used for the characterization of real-world data, such as electrical load, vehicular behavior, handwritten signatures, and IoT time series [Rosso et al., 2013; Aquino et al., 2015; Rosso et al., 2016; Borges et al., 2019b]. However, contrary to most class separability indices in the literature, which are based on some distance measure to identify different classes, distances in the CCEP are not easy to consider. For instance, given the minimum and maximum limits for statistical complexities in the plane, as illustrated in Section 7.3, a straight line between two points is not always possible, since it may not respect those limits. Thus, we took another path to propose the separability index of TSCLAS, which is based on the estimation of classes regions and the intersection among them.

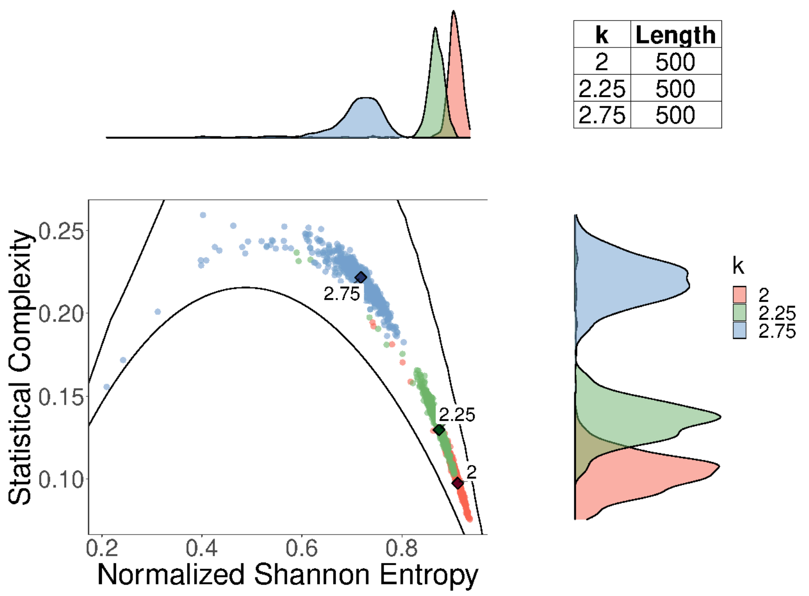
7.3 The class separability of time series dynamics

A powerful tool for the characterization of time series dynamics is the Causality Complexity-Entropy Plane (CCEP), proposed by Rosso et al. [2007a], and detailed in Section 3.6.2. It was successfully applied as a method to distinguish different time series dynamics, according to the placement of (H_S, C_{JS}) pairs. For instance, it is useful for the hard task of distinguishing noise from chaos [Rosso et al., 2007a; Zunino et al., 2012; Rosso et al., 2013; Kulp and Zunino, 2014; Ravetti et al., 2014; Rosso et al., 2015; Ribeiro et al., 2017; Ye et al., 2017]. However, although this method can distinguish most of these dynamics, some conditions must be satisfied by the time series data, so it can be placed at the expected position.

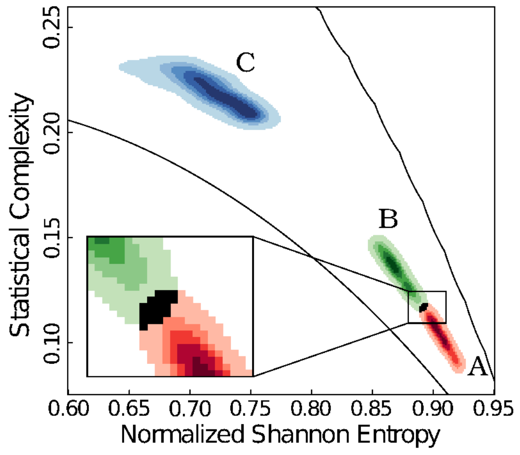
Firstly, to assure the reliability of the computed measures, the time series length n must be long enough so the sampling in the $D!$ space of patterns is representative, i.e., $n \gg D!$ [Rosso et al., 2013; Zunino et al., 2012]. If this condition is not satisfied, it may lead to a statistical analysis that is not consistent with the real-time series behavior and the occurrence of missing patterns [Rosso et al., 2012]. Furthermore, even if the length is satisfied, in real-world scenarios, data may not be pure concerning a single dynamic behavior. A dominant behavior may be corrupted by spurious noise or even by rounding or truncating values, which may affect the precision when measuring their dynamics. These issues add a certain degree of uncertainty to the measures, impacting their placement in the CCEP.

Figure 7.1a shows a CCEP, for $D = 4$ and $\tau = 1$, illustrating the placements of 1,500 colored random noise time series, each of them with $n = 1{,}000$ samples, and defined by their power spectra f^{-k} [Larrondo et al., 2006], representing three equal sized classes with $k \in \{2, 2.25, 2.75\}$. Classes are represented as red, green, and blue points, respectively, for each k (represented by the highlighted diamond points in the figure). Continuous lines represent the minimum and maximum limits for statistical complexities Their marginal density distributions also illustrates the spread of points along H_S and C_{JS} axes.

It can be noted that the potential for correctly distinguishing the time series dynamics with the CCEP is related to the placement distribution of points for each class in the plot. Assuming the distribution of points in Figure 7.1a, the points of the class $k = 2.75$ are more straightforward to distinguish than the others since the distribution of their points does not intersect with others. Otherwise, the intersection between classes for $k = 2$ and $k = 2.25$ makes their distinction harder since there is no clear separability between those time series classes, which, however, may lead to misclassifications. Thus, to measure this potential of classification with the CCEP, let



(a) CCEP with points placements according to its k and their marginal distributions along the H_S and C_{JS} axes.



(b) Transformed grid from the CCEP of Figure 7.1a, after filtering the most significant points and estimating the classes distributions with KDE ($n_{KDE} = 250$). Intersection area is highlighted.

Figure 7.1. Illustration of CCEP, for $D = 4$ and $\tau = 1$, of time series generated as colored random noises, defined by their power spectra f^{-k} [Larrondo et al., 2006], representing three equal sized classes with $k \in \{2, 2.25, 2.75\}$. Continuous lines represent the minimum and maximum limits for statistical complexities.

us define class separability as following.

Definition 7.3.1 (Class separability)**.** Let $\mathcal{D}$ be a given dataset composed of a number of time series, with each of them labeled from a set of classes $\mathcal{C} = \{c_1, \ldots, c_\ell\}$. The class separability for $\mathcal{D}$, projected over a CCEP, with pairs (H_S, C_{JS}) computed with D and τ, corresponds to the ratio between the intersected areas from the distribution of points and the non-intersected areas.

To analyze the class separability for this scenario, we have to define our version of a separability index S_I for the classes of a given dataset. Before describing S_I, let us establish some requirements:

1. It should be robust to outliers to reduce the effects of noise and imprecision of the measurements on the analysis.

2. It should be non-parametric, given the placement area of points in CCEP, it is not easy to fit a priori distribution for the data to calculate their intersection ranges.

3. It should be independent of the number of instances per class, and, thus, we must consider the area covered by the points in a broad sense, and not computing the index per point.

These requirements are essential due to the possible expensive computational cost that quickly arises when dealing with the magnitudes of IoT data. So, as a solution for requirement (1), we propose using a method for removing the outliers by computing the points by their concentration range. Any method could be applied here, but we follow the approach described in Borges et al. [2019b], which achieved good results by using the Skinny-dip [Maurus and Plant, 2016] clustering strategy on the points for each class. Skinny-dip is a noise-robust clustering algorithm based on the Hartigan's dip test of modality, which can reasonably detect the more distinguished concentration of points in a given region. In this work, we use a statistical significance level of $\alpha = 0.05\,\%$ for estimating the clusters.

For requirements (2) and (3), we propose estimating class distributions, after filtering the most significant points, by a Kernel Density Estimation (KDE). For our case, it is sufficient to apply a two-dimensional KDE given our bivariate case with the H_S and C_{JS} axes of CCEP. As a result, KDE will transform the CCEP onto an $n_{KDE} \times n_{KDE}$ square grid. A large n_{KDE} impacts on the precision of the estimated regions and has implications on the processing time. Figure 7.1b shows the combined version of the resulting grids for each class from the CCEP of Figure 7.1a, after applying the methods above with $n_{KDE} = 250$.

Let us model these grids as $n_{KDE} \times n_{KDE}$ matrices $\mathbf{A} = \{a_{i,j}\}$, $\mathbf{B} = \{b_{i,j}\}$, and $\mathbf{C} = \{c_{i,j}\}$, corresponding to the areas covered by the classes distributions formed by points from colored noises with $k = 2$, $k = 2.25$, and $k = 2.75$, respectively. Without loss of generality, let us consider the case of the class distribution from $k = 2$ modeled by $\mathbf{A}$. Each one of their elements is defined as $a_{i,j} = d_{i,j}$, with $i, j \in \{1, \ldots, n_{KDE}\}$, where $d_{i,j}$ represents the values according to the estimated density of points at that location from its underlying CCEP.

Thus, for each ij^{th} element, we want to measure the proportion of the density this element of $\mathbf{A}$ has concerning other distributions intersecting at these positions. Then, each $a_{i,j}$ element is normalized as

$$a'_{i,j} = \frac{a_{i,j}}{\sum_{c \in \mathcal{C}} c_{i,j}}, \tag{7.1}$$

where $c \in \mathcal{C}$ is used as a correspondence for all the classes of labels in the given dataset.

Consequently, the possible values of $a'_{i,j}$ are

$$a'_{i,j} = \begin{cases} 0 & \text{if } a_{i,j} = 0, \\ 1 & \text{if } c_{i,j} = 0, \forall c \in \mathcal{C} \text{ and } c \neq a, \\ (0,1) & \text{otherwise.} \end{cases} \tag{7.2}$$

Note that the proportion $0 < a'_{i,j} < 1$ only occurs in case of intersection at that point.

After the normalization of all values, having a normalized matrix $\mathbf{A}'$, we can compute the individual separability s_c for each class $c \in \mathcal{C}$. Let $A' = \{a'_{i,j} | a'_{i,j} > 0\}$ be the set of elements of $\mathbf{A}'$ with non-zero values. If $m = |A'|$, then, the individual separability for a given class A is given by

$$s_A = \frac{\sum_i \sum_j a'_{i,j}}{m}. \tag{7.3}$$

The following Lemmas are directly derived from the equations above.

Lemma 7.3.1 (Maximum individual separability)**.** For a given class distribution without intersection with any other class, its separability achieves the maximum value equals 1.

Proof. If a normalized grid $\mathbf{A}'$, transformed from a given class distribution has no intersections, then, $a'_{i,j} = 1, \forall i, j \in \{1, \ldots, n_{KDE}\}$. Thus, its individual separability is given by

$$s_A = \frac{\sum_i \sum_j 1}{m} = \frac{m}{m} = 1. \tag{7.4}$$

□

Lemma 7.3.2 (Minimum individual separability)**.** The minimum individual separability for a given class distribution occurs when all ℓ classes distributions are completely overlapped, and it is equal to $1/\ell$.

Proof. Let ℓ be the number of class distributions in a given dataset. If all classes are perfectly overlapped, then their estimated grids will be equally distributed over the CCEP. So, all elements of a given normalized grid $\mathbf{A}'$ have intersections with all other classes, with the same density values and, consequently, $a'_{i,j} = 1/\ell, \forall i, j \in \{1, \ldots, n_{KDE}\}$. Thus, its individual separability is given by

$$s_A = \frac{\sum_i \sum_j 1/\ell}{m} = \frac{m/\ell}{m} = 1/\ell. \tag{7.5}$$

□

Finally, given the individual separability for all classes in $\mathcal{C}$, the separability index S_I is given by

$$S_I = \frac{1}{\ell} \sum_{c \in \mathcal{C}} S_c. \tag{7.6}$$

The bounds for the class separability index S_I is presented by Theorem 7.3.3.

Theorem 7.3.3 (Bounds of separability index)**.** The separability index S_I for a given dataset $\mathcal{D}$, composed of a number of time series labeled from a set of classes $\mathcal{C} = \{c_1, \ldots, c_\ell\}$, with respect to its projected CCEP, is bounded by $1/\ell \leq S_I \leq 1$.

Proof. Following Lemmas 7.3.1 and 7.3.2, the maximum value of the separability index S_I of a given dataset occurs when all classes $c \in \mathcal{C}$ achieve their maximum individual separabilities s_c. Conversely, the minimum S_I occurs when s_c is minimum for all $c \in \mathcal{C}$. Thus, the maximum separability index $S_I^{\max}$ is given by

$$S_I^{\max} = \frac{1}{\ell} \sum_{c \in \mathcal{C}} S_c^{\max} = \frac{1}{\ell} \sum_{c \in \mathcal{C}} 1 = \frac{\ell}{\ell} = 1, \tag{7.7}$$

and the minimum separability index $S_I^{\min}$ is given by

$$S_I^{\min} = \frac{1}{\ell} \sum_{c \in \mathcal{C}} S_c^{\min} = \frac{1}{\ell} \sum_{c \in \mathcal{C}} 1/\ell = \frac{\ell/\ell}{\ell} = \frac{1}{\ell}. \tag{7.8}$$

□

Algorithm 6 illustrates and summarizes all steps for computing the class separability index of a given dataset. Besides all the steps above, in line 5, we have to perform an additional step to consider the CCEP limits. This step is necessary since the KDE method only estimates the density with a grid according to the parameters and is not aware of these limits. However, since they are strict for the CCEP, assuring all points are always within limits, we have to change all grid elements outside the CCEP limits to zero. Otherwise, it will impact the metrics computation. It is worth noting that the robust clustering Skinny-Dip presents a significance level and is used as a threshold for estimating the points within their clusters, and, thus, we set $\alpha = 0.05$.

The class separability index S_I depends on a given dataset and other instances may show different S_I values. This fact may cause differences, for instance, between the expected classification potential when analyzing a given training dataset, and the resulting accuracy for the testing dataset.

ALGORITHM 6: ClassSeparability

```
Input: Lists of computed features from the $m$ time series of a given dataset,
       $\mathbf{h} = \{h_1, \ldots, h_m\}$ and $\mathbf{s} = \{s_1, \ldots, s_m\}$, for the normalized Shannon entropy and
       statistical complexity, respectively. Their labels $\mathbf{y} = \{y_1, \ldots, y_m\}$, where
       $\forall i, y_i \in \mathcal{C} = \{c_1, \ldots, c_\ell\}$. And an embedding dimension $D$.
Output: The class separability index $S_I$ for the given parameters.
   // Estimating density grids for each class
1  for $k \leftarrow 1$ to $\ell$ do
       // Filtering features by class
2      $\mathbf{h} \leftarrow \{h_j : y_j = c_k\}$ ; $\mathbf{s} \leftarrow \{s_j : s_j = c_k\}$ ;
       // Filtering outliers features with skinny-dip
3      $\mathbf{h}, \mathbf{s} \leftarrow$ skinnyDipCluster ($\mathbf{h}$, $\mathbf{s}$, $\alpha \leftarrow 0.05$);
       // Computing the grid of class $k$ with KDE
4      $\mathbf{A}^k \leftarrow$ KDE ($\mathbf{h}$, $\mathbf{s}$, $n_{KDE} \leftarrow 250$);
       // Zeroing grid elements outside the CCEP limits
5      $\mathbf{A}^k \leftarrow$ limitCCEP ($\mathbf{A}^k$);
       // Converting densities to probability
6      $\mathbf{a}^k_{i,j} \leftarrow \mathbf{a}^k_{i,j} / \sum_{i=1}^{n_{KDE}} \sum_{j=1}^{n_{KDE}} \mathbf{a}^k_{i,j}$
   // Normalizing each element of the classes as [0,1]
7  for $i \leftarrow 1$ to $n_{KDE}$ do
8      for $j \leftarrow 1$ to $n_{KDE}$ do
9          for $k \leftarrow 1$ to $\ell$ do
10             $\mathbf{a}^k_{i,j} \leftarrow \mathbf{a}^k_{i,j} / \sum_{c \in \mathcal{C} c_{i,j}}$
   // Computing individual separabilities for each class
11 for $k \leftarrow 1$ to $\ell$ do
       // The number of elements in the $k$-th grid with non-zero values
12     $\mathbf{A}' = \{a_{i,j} | a_{i,j} > 0\}|$ ; $m \leftarrow |\mathbf{A}'|$ ;
       // Computing the individual separability for the class
13     $s_k \leftarrow \sum_i \sum_j a'_{i,j} / m$
   // Computing the class separability index
14 $S_I \leftarrow 1/\ell \sum_{c \in \mathcal{C}} s_c$
15 return $S_I$;
```

7.4 TSCLAS: time series classification via class separability

The class separability index S_I measures the potential for correctly distinguishing time series according to their dynamics, and, thus, it is reasonable to consider the maximization of S_I to achieve better classification results. For the domain of ordinal patterns transformations, both embedding dimension D and embedding delay τ have important roles in capturing different aspects of time series dynamics, which could indicate such maximization task [Rosso et al., 2007a; Larrondo et al., 2006; Rosso et al., 2013; Zunino et al., 2012].

Concerning the embedding dimension D, as emphasized in Section 7.3, there are recommendations for its proper definition, which is directly related to the time series length. Since $D!$ is the number of possible ordinal patterns (symbols) to consider, the length n of time series must be long enough so that the sampling in the $D!$ space of patterns is representative, i.e., $n \gg D!$ [Rosso et al., 2013]. On the other hand, there is no simple rule of thumb for defining the embedding delay τ. Although for most studies, it is sufficient to define $\tau = 1$, different values of τ are used for specific tasks, such as finding periodicity in data and estimating the scale of delayed systems [Zunino et al., 2012]. Furthermore, as shown by Zunino et al. [2012] and Borges et al. [2019a], it is possible to discover important characteristics of the time series when evaluating the extracted metrics as a function of τ.

Thus, in this work, we follow the approach of analyzing the behavior of the classes' metrics for different values of embedding delays τ, intending to find the most appropriate one for the maximization of S_I. This is formally defined by Problem 7.4.1.

Problem 7.4.1 (Maximization of class separability index S_I)**.** For a given dataset $\mathcal{D}$ and an embedding dimension D, find $\tau \in \mathcal{T}$ that maximizes the class separability index S_I. The search space $\mathcal{T} = \{1, \ldots, \tau_{\max}\}$ consists of the set of possible embedding delays, with respect to both D and the length n of time series in $\mathcal{D}$.

7.4.1 Maximization of S_I via multiscale approach

The maximization of class separability index via a multiscale approach consists of computing S_I for each pair (D, τ), with $\tau = \{1, \ldots, \tau_{\max}\}$. Different values for τ correspond to sampling data with different interval scales within the time series data points. In this case, we have a limit for the maximum possible value of $\tau_{\max}$, which depends on the length n of the time series and the chosen embedding dimension D, and is bounded by

$$\tau_{\max} < \frac{n}{D-1}. \tag{7.9}$$

This limit affects the number of observed ordinal patterns within a time series, so this bound ensures at least one observed pattern. For the experiments from this work, we have datasets with time series ranging from $n = 96$ to $n = 30{,}240$, representing data collected at 1 minute intervals for one day up to 3 weeks. If we consider $D = 6$, this range gives a $\tau_{\max} = 19$ for the former case and $\tau_{\max} = 6{,}048$ for the latter. However, such large values of τ are unfeasible in practice, since a higher τ will create sliding windows with very distant elements from the time series, losing any relation among them. For our analysis, we have decided to set $\tau_{\max} = \min(\tau_{\max}, 30)$, which we empirically found as a reasonable trade-off between capturing the temporal correlation dependence of the time series and taking into account the computational costs of the algorithm.

Algorithm 7 presents our strategy for finding the best τ^* that maximizes the separability index S_I, for a given dataset. The method consists of computing the S_I for each τ within the list of embedding delays $\mathcal{T}$, lines 2-11. For a given pair (D, τ), the normalized Shannon entropy and statistical complexity are computed from the ordinal patterns probability distributions of each time series (lines 5-8). With these features, we can create the CCEP and compute its class separability index (line 9). After these rounds, the maximum τ is chosen (lines 10 and 11).

7.4.2 Time series classification strategy

TSCLAS is based on a combination of features extracted from the ordinal patterns probability distribution (p_π) and the ordinal patterns transition graph (G_π), both obtained from the set of ordinal patterns (Π) from these time series. To compute those features, we consider the ordinal patterns transformations for a given embedding dimension D and, for the embedding delay, we chose the best τ^* according to the strategy presented in Algorithm 7, which is the one that maximizes the class separability index S_I for the present data.

The whole classification strategy is presented in Algorithm 8. It assumes as inputs a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$, which is composed by m time series $\mathbf{x}_i = \{x_1, \ldots, x_n\}$ of size n each, with each of them having its label in $\mathbf{y} = \{y_1, \ldots, y_m\}$. As parameters for the ordinal patterns transformations, it expects an embedding dimension $D \in \mathbb{N}$, and a list of embedding delays $\mathcal{T} = \{\tau_1, \ldots, \tau_\ell\}$ to find the best τ^*.

After randomly splitting the datasets into the training and testing subsets, $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$, respectively, in Line 1, the next step is to obtain the τ^* that maximizes the

ALGORITHM 7: FindTau

```
Input: A dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$, containing $m$ time series $\mathbf{x}_i = \{x_1, \ldots, x_n\}$ of size $n$
       each, with labels $\mathbf{y} = \{y_1, \ldots, y_m\}$. An embedding dimension $D$, and a list of
       embedding delays $\mathcal{T} = \{\tau_1, \ldots, \tau_\ell\}$.
Output: The $\tau^*$ that maximizes the separability index $S_I$ of $\mathcal{D}$.
1  $S_I \leftarrow 0$ ;
2  forall $\tau \in \mathcal{T}$ do
       // Lists for the features extracted from each of the $m$ time series of
          $\mathcal{D}$
3      $\mathbf{h} \leftarrow \{h_i = 0 : i = 1, \ldots, m\}$ ;
4      $\mathbf{s} \leftarrow \{c_i = 0 : i = 1, \ldots, m\}$ ;
5      for $i \leftarrow 1$ to $m$ do
           // Computing the ordinal patterns probability distributions for the
              pair $(D, \tau)$
6          $p_\pi \leftarrow$ OrdinalPatternsPD($\mathbf{x}_i, D, \tau$) ;
           // Features extracted: normalized Shannon entropy and statistical
              complexity
7          $h_i \leftarrow$ ShannonEntropy($p_\pi$) ;
8          $s_i \leftarrow$ StatisticalComplexity($p_\pi$) ;
       // Calculating the class separability index
9      $S_I' \leftarrow$ ClassSeparability($\mathbf{h}, \mathbf{s}, \mathbf{y}, D$) ;
10     if $S_I' > S_I$ then
11         $\tau^* \leftarrow \tau$
12 return $\tau^*$;
```

ALGORITHM 8: Classification

```
Input: A dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$, containing $m$ time series $\mathbf{x}_i = \{x_1, \ldots, x_n\}$ of size $n$
       each, with labels $\mathbf{y} = \{y_1, \ldots, y_m\}$. An embedding dimension $D$, and a list of
       embedding delays $\mathcal{T} = \{\tau_1, \ldots, \tau_\ell\}$.
Output: The predicted labels $\mathbf{y}_{\text{pred}}$ for the test dataset.
   // Dataset train/test split
1  $\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{test}} \leftarrow$ TrainTestSplit ($\mathcal{D}$, $train_{pct}$) ;
   // Finding the best tau for the train set
2  $\tau^* \leftarrow$ FindTau ($\mathcal{D}_{\text{train}}, D, \mathcal{T}$) ;
   // Computing features for the selected tau
3  $\mathcal{F}_{\text{train}} \leftarrow$ ExtractFeatures ($\mathcal{D}_{\text{train}}, D, \tau^*$) ;
4  $\mathcal{F}_{\text{test}} \leftarrow$ ExtractFeatures ($\mathcal{D}_{\text{test}}, D, \tau^*$) ;
   // Pre-procesing the features: center and scale
5  $\mathcal{F}_{\text{train}} \leftarrow$ ScaleData ($\mathcal{F}_{\text{train}}$) ;
6  $\mathcal{F}_{\text{test}} \leftarrow$ ScaleData ($\mathcal{F}_{\text{test}}$) ;
   // Training step
7  $model \leftarrow$ Training ($\mathcal{F}_{\text{train}}$)
   // Prediction step on test datset
8  $\mathbf{y}_{\text{pred}} \leftarrow$ Prediction ($model$, $\mathcal{F}_{\text{test}}$)
9  return $\mathbf{y}_{\text{pred}}$;
```

class separability for the training split, Line 2. It is worth mentioning that, following this method for selecting τ^*, we avoid the need to perform successive classifications on a validation set as in a conventional hyperparameter tuning approach. Instead, we only need to evaluate the maximization of the proposed separability index with the training set, which is more efficient and adequate for the IoT scenarios.

Thus, once we have D and τ^*, we can extract the features from both datasets (Lines 3 and 4), which are further scaled to have zero mean and unit variance (Lines 5 and 6). The computed features are those described in Section 3.5.4, with addition to the probability of self-transitions measure p_{st}, described in Section 5.5. The features are presented here as $\mathcal{F}_{\text{train}}$ and $\mathcal{F}_{\text{test}}$ from the training and testing subsets, respectively.

For the time series classification step, once we have the dataset properly converted, any classification algorithm may be used, according to the availability and necessity of the problem. For the chosen classifier, the training and prediction steps are presented in lines 7 and 8, respectively. However, as shown in Section 7.5, given the importance of considering the large numbers of IoT environments, we chose the random forest classifier [Breiman, 2001], so this step may correspond to both accuracy and processing time requirements. Section 7.5.1 presents more details regarding the chosen classification algorithm and its parameters.

7.5 Results

To evaluate TSCLAS, we performed experiments of time series classification on datasets collected from real-world sensors, and compared our results with known classification algorithms from the literature. Besides, we considered common data quality issues to evaluate the robustness of TSCLAS. In the following, we describe the materials and methods for our experiments, and the results achieved for the considered scenarios.

7.5.1 Materials and methods

As described by Borges Neto et al. [2015]; Borges et al. [2019b] and Montori et al. [2018b], IoT data is very challenging to handle. Besides the large amount and heterogeneity of sensors, issues such as missing readings, different rates, among others, make IoT a very unreliable scenario. Thus, in order to properly evaluate TSCLAS, we have to choose those sensors from the whole IoT spectrum that can provide more reliable data, mainly with respect to their precision, probability of correctness, and trustworthiness [Borges Neto et al., 2015; Buchholz et al., 2003]. We decided to perform our experiments using data collected from automated airport weather sensors, the

Automated Surface Observing Systems (ASOS). These are reliable sensors that lies in the intersection between CoIoT and IIoT, as presented in Section 2.3. These sensors generate observations every minute, or every hour, according to the airport, and are used to support weather forecast activities and aviation operations[1]. Their reliability comes from constant monitoring of data quality, 24 hours per day, with maintenance as soon as a problem is detected.

The typical format of these data is the METAR format[2], standardized by the International Civil Aviation Organization (ICAO), which follows a common syntax used for weather reporting around the world[3]. However, to collect our data, we use the weather data archive of the Iowa Environmental Mesonet (IEM), from the Iowa State University, which collects the ASOS data, stores and makes them available via an API and a Web interface[4]. In our experiments, we collect data with a 1-minute interval, available only for the ASOS from the United States, and with a 1-hour interval, available for ASOS from several countries in the world.

With these data, we can perform experiments with different setups by varying the data interval granularity, time span, and geographical locations. Thus, we evaluate TSCLAS on time series of (i) temperature, (ii) atmospheric pressure, (iii) wind speed, and (iv) wind direction, composing the four classes in our datasets, considering both 1-minute and 1-hour time intervals. For the 1-minute interval experiment, the time series data were collected from 803 airports in the U.S. with geographical locations illustrated in Figure 7.2a, and, for the 1-hour interval, the time series data were collected from 3742 airports around the world, as depicted in Figure 7.2b. The validity of our experiments is corroborated by the fact that the time series related to weather phenomena are the most common types of data in the current IoT [Borges Neto et al., 2015; Borges et al., 2019b; Montori et al., 2018b].

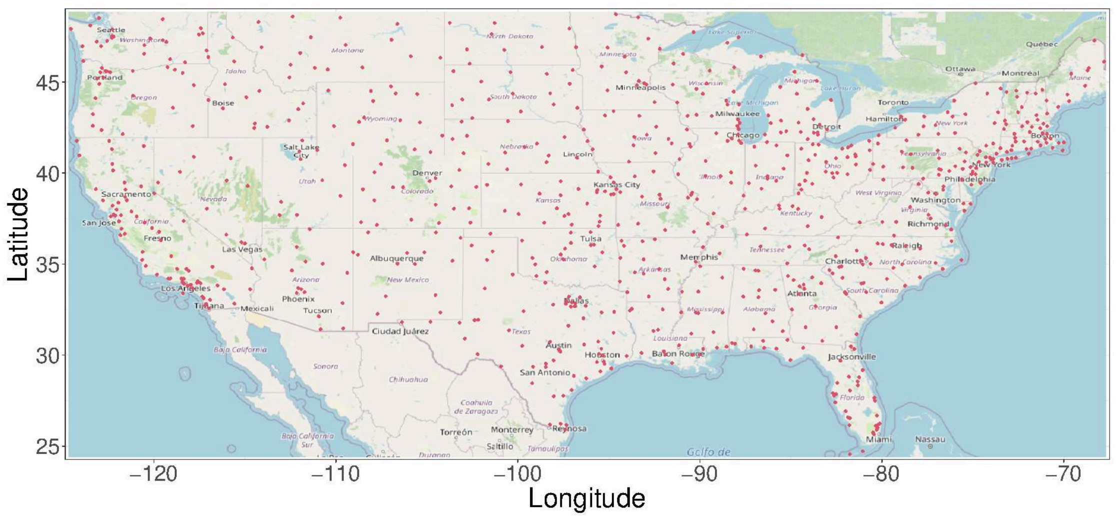
We may choose any classification algorithm to perform the time series classifications, as described in Algorithm 8. However, given the characteristics of IoT scenarios, we decided for the random forest classifier [Breiman, 2001]. Since we are dealing with large IoT datasets, this decision is motivated by the good reported accuracy and fast processing time of the random forest classifier, besides its robustness to outliers and noise. Furthermore, given the relevance of the computed features for each time series,

[1]Automated Surface Observing Systems from the U.S. National Weather Service: `https://www.weather.gov/asos/asostech`.

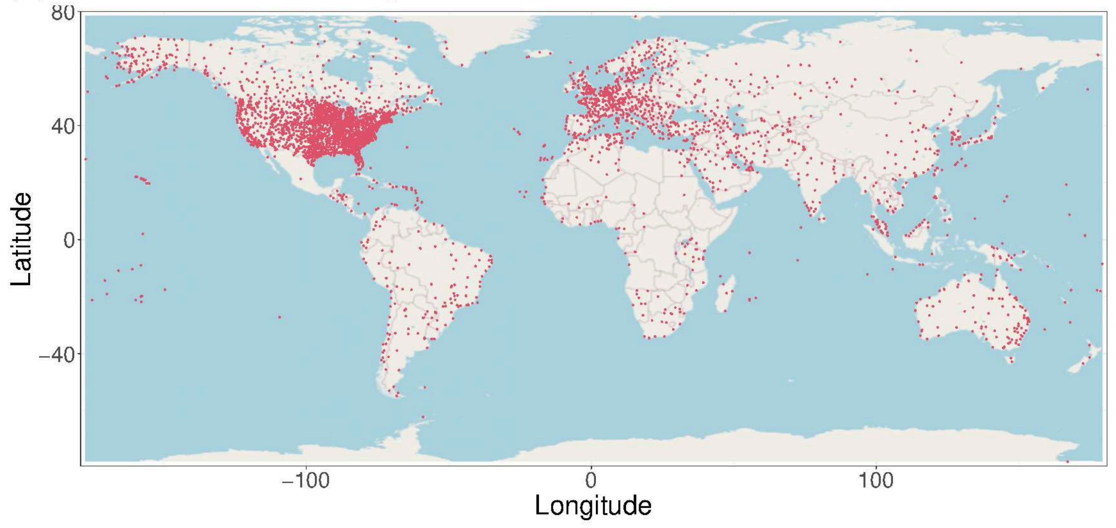
[2]METAR description at Wikipedia: `https://en.wikipedia.org/wiki/METAR`.

[3]Guide to decode the ASOS - METAR format: `https://www.weather.gov/media/wrh/mesowest/metar_decode_key.pdf`, and a more intuitive METAR decoder (in portuguese): `https://www.redemet.aer.mil.br/?i=facilidades&p=decodificacao-metar`.

[4]Iowa Environmental Mesonet ASOS-METAR Data Download: `https://mesonet.agron.iastate.edu/ASOS/`.



(a) ASOS locations of 803 airports in the United States used for the 1-minute dataset.



(b) ASOS locations of 3742 airports from around the world used for the 1-hour dataset.

Figure 7.2. Geographical locations of the airports where ASOS data were collected.

the importance of the random forest internal estimation variable is a proper solution for the classification. For the random forest classifier, two main parameters must be given: the number of trees in the forest, where we fixed as $ntree = 200$, and the number of variables randomly sampled as candidates at each split, also fixed as $mtry = 2$ in our experiments. The decision to fix these parameters and not perform any parameter optimizations is to conduct a fair comparison between all algorithms, independently of their programming languages and libraries specificities.

In order to evaluate our proposal with respect to the related work, we decided to compare our results with known classification algorithms from the literature. As described in Section 7.2, the strategies that are specifically designed for the classification of IoT data are very domain specific, making it difficult to adapt to other contexts, or the authors of those papers does not make their code available, making them difficult to be reproducible. Thus, TSCLAS is compared to four different classification algorithms from literature, which are well known and easy to reproduce. The Random Forest classifier (RANDF), to test the classification of the raw time series without our transformation step; the k-nearest neighbors (KNN), to compare with this well-known distance-based algorithm, mostly used as a ground truth for time series classification with $k = 1$, and the Dynamic Time Warping (DTW) as a distance method [Ratanamahatana and Keogh, 2005; Bagnall et al., 2017]; the Time Series Forest Classifier (TSF) [Deng et al., 2013], a composition algorithm that improves the random forest method; and the Random Interval Spectral Forest (RISE) [Lines et al., 2018], a more recent frequency-based algorithm that performs an ensemble of the trees with features extracted from the spectral domain of time series. For these tree-based algorithms, we also fix the number of trees as $n\_estimators = 200$ and the number of splits as $min\_samples\_split = 2$, to have a fair comparison between all methods and our strategy.

All experiments were performed in a computer with Intel Core i9-9900X CPU at 3.50 GHz $\times$ 20, 128 GB RAM, and running a Linux Ubuntu 18.04.4 LTS 64-bit. We implemented TSCLAS in R 3.4.4 [R Core Team, 2018], with some excerpts of code in C++. All pieces of code used in our experiments are available at a public repository[5]. The algorithms from the literature used to compare are implemented in Python 3.6.9, within the *sktime* library [Löning et al., 2019], a unified toolbox for machine learning with time series, except for the RANDF, which is implemented with the scikit-learn library [Pedregosa et al., 2011].

Table 7.1 presents the versions of software and libraries used in our experiments.

Another step necessary when comparing TSCLAS to the literature solutions is to transform the raw collected time series into a proper dataset to be read by the algorithms. Although it is not necessary to TSCLAS, for the other strategies we have to first adjust the time series to have equal length. For this, given the expected number of data points for a given time span and a fixed time interval between the points, we adjusted each observation from the collected data by its timestamp to its corresponding position in the adjusted time series. For instance, for making a dataset from the 1-minute raw time series data, considering a 1-day time span, with a 1-minute interval between observations, each time series should have 1,440 samples. Thus, by consid-

[5]Classification of IoT data with ordinal pattern transformations: `https://github.com/labepi/tsclas`. This code requires the codes for Bandt-Pompe ordinal pattern transformations, available at `https://github.com/labepi/bandt_pompe`.

Table 7.1. List of main software and libraries versions used in the experiments in this work.

Software	**Details**	**Version**
R	R software system.	3.4.4
caret	R package used for preprocessing data (centering and scaling) and for creating the dataset splits.	6.0.86
randomForest	R package used for the RANDF classifier.	4.6.14
diptest	Hartigans' Dip Test for Unimodality, used for the skinny-dip clustering.	0.75-7
MASS	Used for the kernel density estimation KDE.	7.3-49
Rcpp	Used for the codes excerpts in C++.	1.0.4.6
matrixcalc	Used for matrix operations when extracting features.	1.0-3
Python	Python software system.	3.6.9
scikit-learn	A machine learning library for the Python. Used for the RANDF classification algorithm and accessory functions.	0.23.1
pandas	Python tool used for data analysis and manipulation tool.	1.1.0
numpy	A library for the Python programming language used for mathematical operations.	1.19.1
sktime	A unified toolbox for machine learning with time series. Used for the time series classification algorithms of literature.	0.4.1

ering a starting date, which for our experiments was January 1, 2020, we place each observation at the correct position according to its timestamp. For timestamps without observations, we have a missing data sample. For evaluating TSCLAS in different setups, we create different datasets by varying these parameters, which impact the time series lengths for each case.

For the 1-hour raw time series, the time interval is fixed in 1 hour, but we may vary the time span. However, for these time series, before adjusting the timestamp, we have to synchronize the hourly observations from different time series according to their minute of data sampling. This is necessary because each ASOS sensor collects data at different minutes from the hour. For instance, some of them make their samples at 00:05 h, 01:05 h, 02:05 h, and so on, while others collect their samples at 00:42 h, 01:42 h, 02:42 h, and so on. Thus, we discover each minute of collection for the ASOS sensors by considering the minute where most of the data is present. After that, we consider this minute as the offset for this sensor and use it to center all data samples at minute 0 of each hour. Although this may potentially distort the moment of data collection by at most 59 minutes, it is necessary to create the dataset for the literature algorithms. It is important to mention that this issue does not impact the ordinal patterns construction of our strategy.

For both 1-minute and 1-hour datasets, we only consider adding to the dataset those time series with at least 80 % of valid data samples for the given time parameters. For those time series satisfying this condition, before classifying them using literature algorithms, we apply a linear interpolation for missing data imputation. We have tested different approaches for the data imputation, and our conclusion is that a simple linear approach provides the best trade-off between the resulting accuracy and processing time.

Table 7.2 presents the different configuration setups used in our experiments.

Table 7.2. Configurations for the 1-minute and 1-hour experiments and the lengths of time series for each time span and interval parameters.

1-minute experiments					**1-hour experiments**			
Time span	**Time series length per interval**				**Time span**	**Time series length**	**Time span**	**Time series length**
	1 min	**5 min**	**10 min**	**15 min**				
1-day	1,440	288	144	96	1-month	744	4-months	2,904
1-week	10,080	2,016	1,008	672	2-months	1,440	5-months	3,648
2-weeks	20,160	4,032	2,016	1,344	3-months	2,184	6-months	4,368

Finally, another parameter necessary for the classification step is the training percentage used for the dataset train/test split, which is defined as $train_{pct} = 0.8$ for all experiments involving dataset split, giving the 80/20 split as commonly used in the machine learning literature. We also assure that each one of the four classes present in our data has the same number of instances on both train and test splits, so the datasets are balanced with respect to the number of classes. Finally, we assure the precision of our results by presenting them as the average of 30 resamples of randomly selected train/test splits, with 95 % confidence interval of the error margin.

7.5.2 Results and discussion

In this section, we present our results on the classification of time series for the 1-minute and 1-hour ASOS time series. To evaluate TSCLAS and compare its behavior in contrast to the known solutions from literature, we perform three different experiments, as presented in the following.

7.5.2.1 Classification of 1-minute time series

In this experiment, we evaluate the capacity of TSCLAS to correctly classify the 1-minute ASOS time series under different conditions, varying time span of data collection, and time interval between consecutive observations. Figure 7.3a presents the classification accuracy of TSCLAS by considering four different embedding dimensions $D = \{3, 4, 5, 6\}$, and for the KNN, RANDF, TSF, and RISE, algorithms for the 1-day time span. Table 7.3 presents all mean accuracies for these regular time series, best results per configuration are in bold.

We can observe that, for the 1-minute time interval, TSCLAS achieves an accuracy around 0.89, behind TSF and RISE, which are in the order of 0.91, and ahead of KNN and RANDF. We can also see that our accuracies considerably decay as the time interval between samples increases for 5, 10 and 15 minutes, being more accentuated for higher values of D. While KNN and RANDF remain barely constant, similar behavior is also observed for both TSF and RISE, but more smoothly. This expected behavior was presented in Section 3.5.1, i.e., for a given time series with length n, the condition $n \gg D!$ must be satisfied to obtain reliable statistics with the ordinal patterns transformations [Zunino et al., 2012]. Since we only have 96 samples for 15-minute time intervals in a single day, c.f. Table 7.2, there is not enough data for the method, mainly when D is higher.

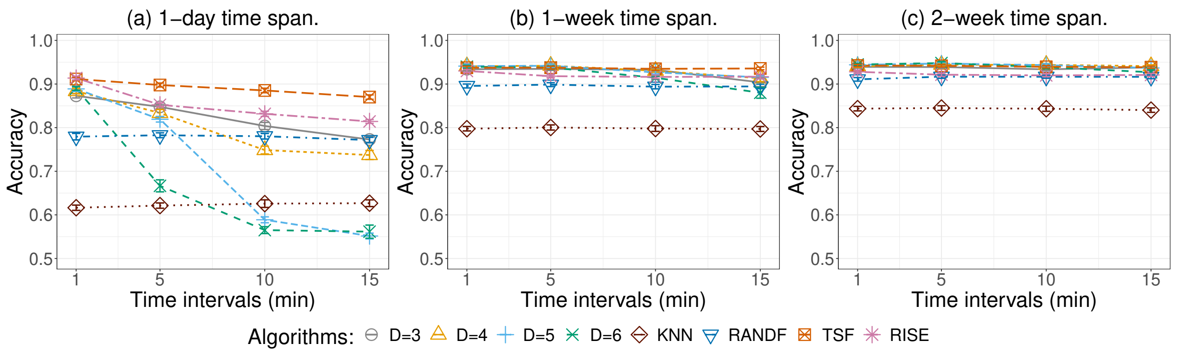


Figure 7.3. Accuracies of 1-minute ASOS time series for time spans of (a) 1-day, (b) 1-week, and (c) 2-week, starting in January 1, 2020, evaluated for time intervals of 1, 5, 10, and 15 minutes between consecutive observations.

On the other hand, as presented in Figure 7.3b, as the number of samples increases when considering a one week of data, the achieved accuracies also increase. For instance, for a 1-minute time interval, the time series length grows from 1,440 to 10,080 observations, and we have accuracies around 0.94 for all values of D, tying whith TSF. However, we can also observe the dependency of TSCLAS on the number of samples, as the time interval increases and, consequently, the n decreases up to 672 samples for 15 minutes, decreasing our achieved accuracies.

The length of the time series is an important aspect to consider, but it is not the only one impacting our classification results. For instance, the length of the time series for the 1-week experiment with a 15-minute interval of Figure 7.3b is shorter than the time series for the 1-day experiment with a 1-minute interval of Figure 7.3a. However, the accuracy of the former is higher than the latter, i.e., around 0.9 for all Ds, except for $D = 6$, which has 0.88, the same as before. This behavior occurs because the method's success depends on two aspects: number of observations, as discussed above,

Table 7.3. Classification accuracies for different configurations of the 1-minute experiment, for the regular time series and their versions with missing data gaps. Best results per configuration are highlighted in bold.

Time span	**Time interval**	**TSCLAS**				**KNN**	**RANDF**	**TSF**	**RISE**
		D=3	**D=4**	**D=5**	**D=6**				
			Regular time series						
1-day	1 min	0.87	0.88	0.89	0.89	0.62	0.78	**0.91**	**0.91**
	5 min	0.85	0.83	0.82	0.67	0.62	0.78	**0.90**	0.85
	10 min	0.80	0.75	0.59	0.57	0.63	0.78	**0.89**	0.83
	15 min	0.77	0.74	0.55	0.56	0.63	0.77	**0.87**	0.81
1-week	1 min	0.93	**0.94**	**0.94**	**0.94**	0.80	0.90	**0.94**	0.93
	5 min	0.93	**0.94**	**0.94**	**0.94**	0.80	0.90	**0.94**	0.92
	10 min	**0.93**	**0.93**	**0.93**	0.91	0.80	0.89	**0.93**	0.92
	15 min	0.90	0.91	0.92	0.88	0.80	0.89	**0.94**	0.92
2-week	1 min	**0.94**	**0.94**	**0.94**	**0.94**	0.84	0.91	**0.94**	0.93
	5 min	0.94	0.94	**0.95**	**0.95**	0.84	0.92	0.94	0.92
	10 min	0.93	**0.94**	**0.94**	**0.94**	0.84	0.92	**0.94**	0.92
	15 min	**0.94**	**0.94**	**0.94**	0.93	0.84	0.92	**0.94**	0.92
			10 % gap						
1-day	1 min	0.87	0.87	0.89	0.88	0.61	0.77	0.90	**0.91**
	5 min	0.83	0.83	0.80	0.71	0.62	0.77	**0.87**	0.84
	10 min	0.79	0.75	0.75	0.54	0.62	0.77	**0.86**	0.82
	15 min	0.75	0.72	0.70	0.52	0.62	0.77	**0.86**	0.80
1-week	1 min	0.93	**0.94**	**0.94**	**0.94**	0.79	0.89	0.93	0.93
	5 min	0.93	**0.94**	**0.94**	0.93	0.78	0.88	0.93	0.91
	10 min	0.92	**0.93**	0.92	0.91	0.78	0.88	**0.93**	0.91
	15 min	0.91	0.90	0.91	0.89	0.79	0.88	**0.92**	0.91
2-week	1 min	**0.94**	**0.94**	**0.94**	**0.94**	0.82	0.89	**0.94**	0.93
	5 min	**0.94**	**0.94**	**0.94**	**0.94**	0.83	0.90	0.93	0.92
	10 min	0.93	**0.94**	**0.94**	0.93	0.82	0.90	0.93	0.92
	15 min	**0.94**	**0.94**	0.93	0.93	0.82	0.90	0.93	0.92
			30 % gap						
1-day	1 min	0.84	0.86	0.85	0.85	0.58	0.72	0.88	**0.89**
	5 min	0.79	0.76	0.75	0.73	0.58	0.73	**0.84**	0.81
	10 min	0.74	0.69	0.70	0.66	0.60	0.73	**0.82**	0.79
	15 min	0.68	0.68	0.65	0.59	0.60	0.72	**0.81**	0.77
1-week	1 min	0.93	**0.94**	**0.94**	**0.94**	0.71	0.82	0.92	0.92
	5 min	0.92	**0.93**	**0.93**	**0.93**	0.72	0.83	0.91	0.89
	10 min	**0.92**	**0.92**	0.91	0.89	0.72	0.82	0.90	0.89
	15 min	0.90	**0.91**	0.88	0.87	0.73	0.82	0.89	0.88
2-week	1 min	0.93	**0.94**	**0.94**	**0.94**	0.76	0.85	0.92	0.92
	5 min	**0.94**	**0.94**	**0.94**	**0.94**	0.76	0.85	0.92	0.91
	10 min	**0.94**	**0.94**	**0.94**	0.93	0.76	0.85	0.91	0.90
	15 min	0.92	**0.93**	**0.93**	0.91	0.75	0.86	0.91	0.90
			50 % gap						
1-day	1 min	0.82	0.82	0.81	0.77	0.56	0.68	**0.85**	0.81
	5 min	0.74	0.72	0.70	0.67	0.55	0.68	**0.81**	0.73
	10 min	0.66	0.62	0.59	0.53	0.56	0.67	**0.79**	0.72
	15 min	0.60	0.58	0.51	0.51	0.57	0.67	**0.76**	0.69
1-week	1 min	0.92	**0.93**	**0.93**	**0.93**	0.70	0.79	0.91	0.87
	5 min	**0.92**	**0.92**	**0.92**	0.91	0.70	0.79	0.89	0.81
	10 min	**0.91**	0.90	0.89	0.87	0.70	0.79	0.88	0.82
	15 min	**0.89**	0.86	0.84	0.83	0.69	0.79	0.87	0.80
2-week	1 min	0.93	**0.94**	**0.94**	**0.94**	0.72	0.82	0.92	0.90
	5 min	0.93	**0.94**	**0.94**	0.93	0.72	0.82	0.91	0.88
	10 min	**0.93**	**0.93**	**0.93**	0.92	0.72	0.81	0.90	0.87
	15 min	**0.93**	0.92	0.91	0.90	0.72	0.82	0.90	0.87

and representativeness of the phenomena of the collected data.

For instance, the temperature values of METAR are reported as degree Celsius with two integer digits, without decimal places, which are rounded up. The same rounding is also applied to other data types. Thus, with an 1-minute interval, small temperature variations are lost, and the values remain constant as long as the temperature does not increase/decrease enough. These constant numbers are misinterpreted by the method as a deterministic behavior that does not exist. However, by increasing the time interval between data observations, the number of consecutive samples tends to reduce, and the phenomena' real conditions can be correctly represented by data. For the 1-minute interval samples, the increase in the number of observations masquerade some data representation issues, and we have the best accuracy results for that case. The same behavior is observed for a 2-week time span, presented in Figure 7.3c, but with better results for all time intervals, once the number of samples for 15 minutes is large enough. In general, TSCLAS ties with the best classifier, TSF, even beating it in the configuration of 2-week and 5 min interval.

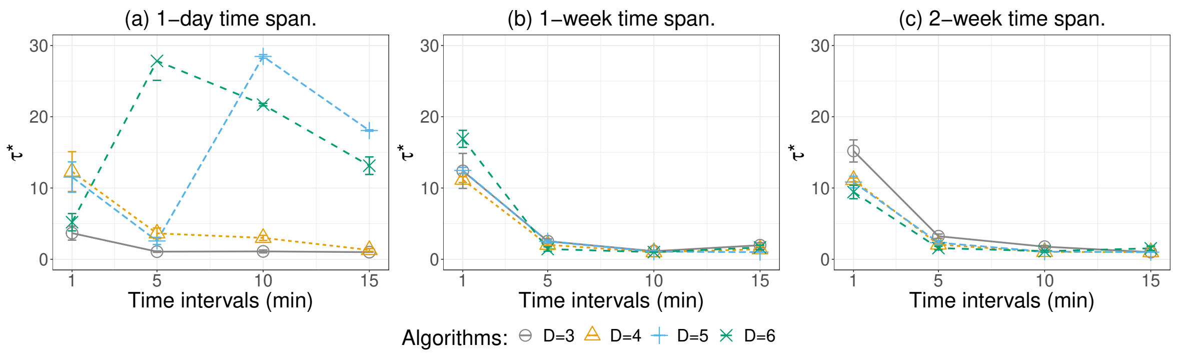


Figure 7.4. Selected values of τ^* during the maximization of class separability index for the configurations of 1-minute ASOS time series, for (a) 1-day, (b) 1-week, and (c) 2-week time spans.

The issue of the time series length versus the representativeness duality is also present in TSCLAS during the maximization of the class separability index. Figure 7.4 presents the average selected τ^* for 1-minute configurations. We can see that, for the 1-week and 2-week spans, in Figures 7.4b and 7.4c, respectively, as the time interval increases the selected τ tends to 1, since the spacing between observations already provides a significant data representation. On the other hand, for 1-minute intervals, we need large values of τ to minimize the impact of rounding.

For the 1-day case, presented in Figure 7.4a, since we have small time series, this behavior is observed only for $D = 3$ and $D = 4$. For $D = 5$ and $D = 6$, we have an inconsistent behavior of the selected τ's because the length n of time series does not satisfy the recommendation for $n \gg D!$, as previously discussed in Section 3.5.1 [Rosso et al., 2007a, 2013]. This reinforces the fact that larger values of D are not recommended for the configurations with short time series since the method cannot compute enough patterns to represent the phenomenon.

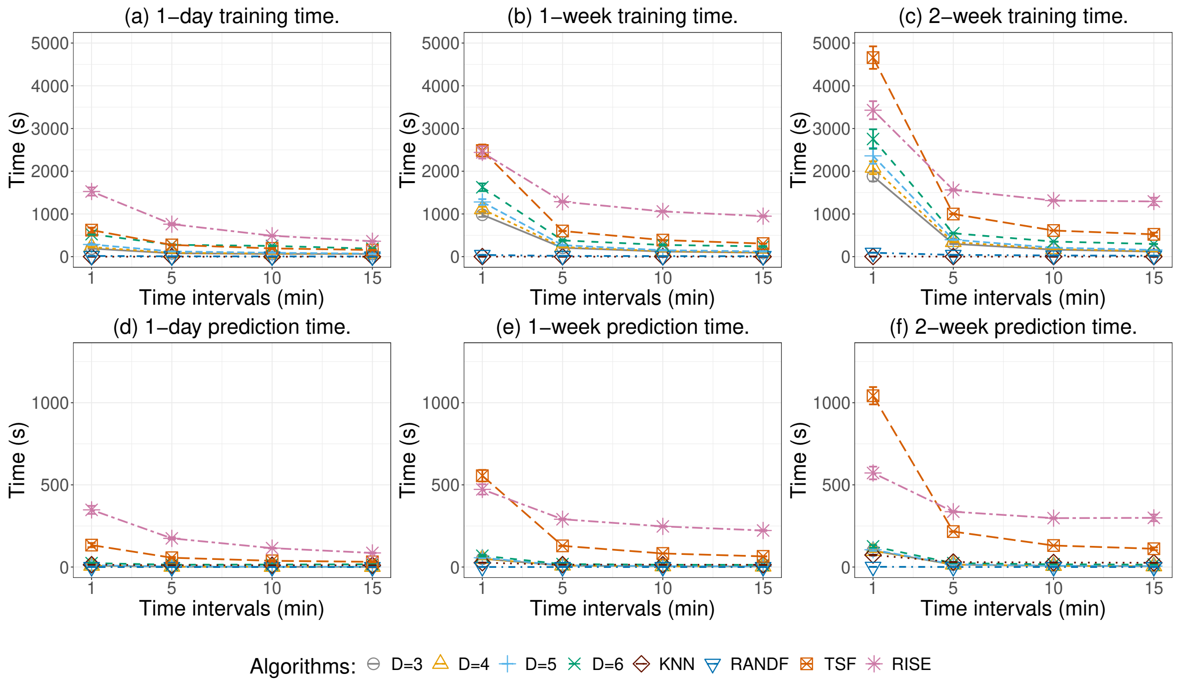


Figure 7.5. Average time spent for the classification of 1-minute ASOS time series, considering their training steps with (a) 1-day, (b) 1-week, and (c) 2-week times spans, and their prediction steps for (d) 1-day, (e) 1-week, and (f) 2-week time spans.

Concerning the time spent on the time series classification, Figure 7.5 presents the average times for the training and prediction steps for the 1-minute ASOS time series. Overall, the time spent for both training and testing increases with the data length. The algorithms with lower processing times are KNN and RANDF, followed by TSCLAS, and, finally, TSF and RISE. For TSCLAS, the time increases with the embedding dimension D, which can easily be observed in Figures 7.5b and 7.5c of the training steps, but it is also similar to the other configurations. TSCLAS requires more training time than KNN and RANDF, but presents better classification accuracies. Given the current availability of cloud and fog resources, the training step can be processed in those powerful machines, which might favor our strategy. Besides, the time spent in the prediction step is competitive to KNN and RANDF, as can be observed in Figures 7.5d-f.

In summary, when TSCLAS is applied to long enough time series, for configurations of 1-week and 2-weeks, it beats the accuracies of KNN and RANDF and is very close to TSF and RISE, even beating them for some configurations. Furthermore, even when not winning these last two, its processing times for both training and prediction are lower, making it a competitive strategy. On the contrary, for the 1-day configurations, TSCLAS is not the best solution since the conditions are not appropriate to achieve its full potential.

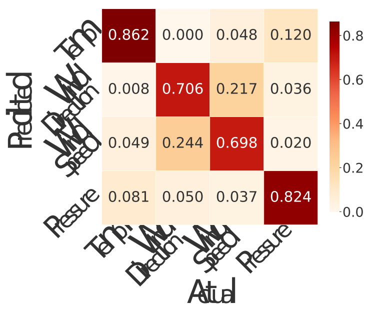
Another perspective for evaluating TSCLAS is presented in Figure 7.6, which illustrates our worst and best-case scenarios for each embedding dimension D. Our worst results occur when there are fewer observations per time series, thus, Figures 7.6a-d illustrate the confusion matrices for the experiments configured with 1-day time span and 15 minutes of time interval between observations, for $D = \{3, 4, 5, 6\}$, respectively. Each confusion matrix was obtained by accumulating the actual-predicted classes for all 30 resamples from each experiment configuration, which was then normalized between 0 and 1 and presented as heat maps. From those figures, we can observe the errors increasing as a function of D, but those errors occur in some particular ways. For instance, it is more likely to misclassify temperature with atmospheric pressure and vice-versa. The same occurs between the wind direction and wind speed time series, which are more misclassified. This behavior occurs due to the similarity between these phenomena' intrinsic dynamics, which are easier to confuse when there is not enough data.

Concerning the best classification results, Figures 7.6e-h, present the heat maps of the confusion matrices for the experiments with a 2-week time span and 1-minute time interval between observations, which are the configurations with more data for the considered embedding dimensions. We can note the reduction of classification errors for all values of D, but there is still a concentration of errors among the wind direction and wind speed time series. Since both time series represent distinct views of the same physical phenomenon (wind), it is harder to make a perfect distinction among them. This issue is inherited from the ordinal pattern transformations, which go deep into the phenomena' dynamic characteristics. At this point, the time series from temperature and atmospheric pressure are correctly classified 99 % of the time.

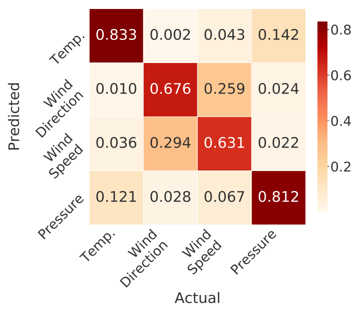
7.5.2.2 Classification of 1-minute time series with missing data

To evaluate the robustness of TSCLAS, we present the results for experiments when we have missing data in the time series. We simulated the case where sensors stopped generating data for a while, and then return their operation, creating continuous gaps in time series. These are concerns that easily occur in current IoT scenarios and that we must consider when classifying their data [Borges Neto et al., 2015; Borges et al.,

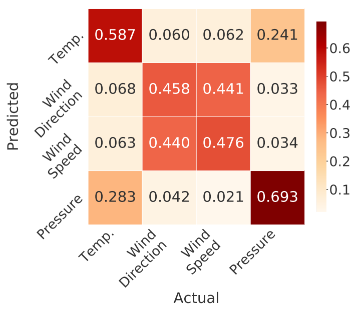
(a) Worst for $D = 3$.



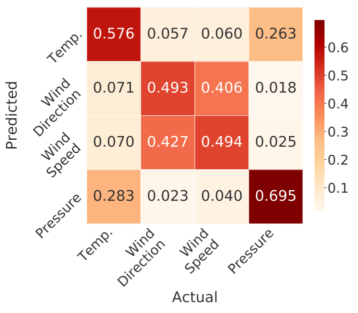
(b) Worst for $D = 4$.



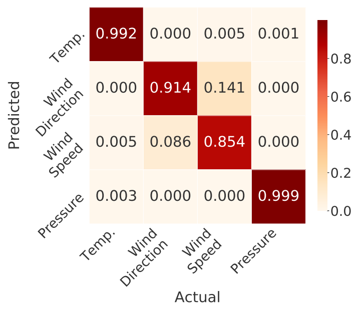
(c) Worst for $D = 5$.



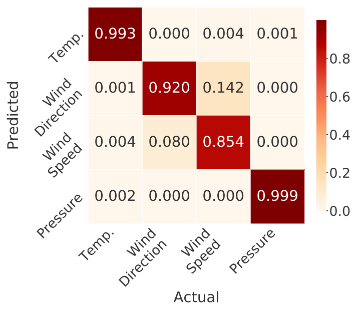
(d) Worst for $D = 6$.



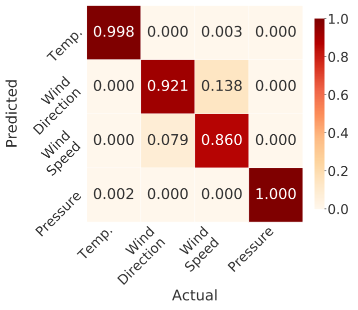
(e) Best for $D = 3$.



(f) Best for $D = 4$.



(g) Best for $D = 5$.



(h) Best for $D = 6$.

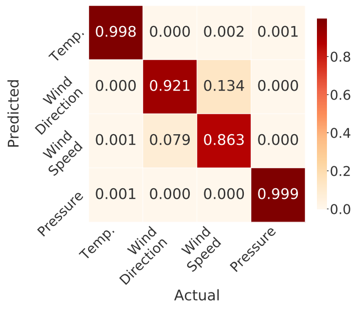


Figure 7.6. Heat maps of the confusion matrices for the worst and best results, with respect to the classification accuracies, for $D = \{3, 4, 5, 6\}$. Worst results occur for 1-day time span and 15 minutes of time interval between observations, and the best ones are achieved for 2-week time span with 1 minute of time interval.

2019b]. The problem with gaps is that (i) they reduce the number of observations, which may affect our method as previously mentioned, and (ii) it is harder to perform data imputation as the gaps increases, which may affect the performance of other methods.

To evaluate at what extent this is a problem for the compared strategies, we tried three different gap sizes, which are 10 %, 30 %, and 50 % of the original length of time series. For each time series, given a gap size, the moment in time the gap occurs is randomly selected within the time series. Starting from the point corresponding to this selected moment in time, all consecutive data points of the time series are removed until completing the gap length for each case. The achieved accuracies for the algorithms for the 1-minute datasets, considering different time spans, time intervals, and gap sizes. are illustrated in Figure 7.7 and its numerical results are presented in Table 7.3.

Figures 7.7a-c present the impacts of the increase in the gap size in the algorithms' accuracy for the 1-day time span. For TSCLAS, we can observe this impact by the steeper negative slope as the gap size increases. In contrast to Figure 7.3, which consists of the classification of the original data without gaps, by reducing the number of observations with gaps, the accuracy of TSCLAS is more damaged. This is due to the violation of the $n \gg D!$ requirement for the transformation, as explained in Sections 3.5.1 and 5.2.1. For instance, for a 1-day time span, with a 15-minute interval, and 50 % of gaps in data, we only have 48 observations to compute the ordinal patterns. The impact on other strategies was only noted for 30 % and 50 % of gaps. While TSF and RISE have a similar but smoother effect than ours, RANDF and KNN have their general accuracy lower but constant concerning the increase in the time intervals.

This same constant, but lower, behavior of RANDF and KNN also occurs for these algorithms on the 1-week and 2-week time spans, presented in Figures 7.7d-f and 7.7g-i, respectively. For other strategies, the increase in gaps still impacts their accuracy, but this is mitigated as the time span increases. However, as the number of observations per time series increases with the growth in time span, the impact of gaps is less perceived on our strategy than on the TSF and RISE. For instance, considering Figure 7.7i and Table 7.3, for 2-week time span with a 50 % gap size, the best accuracies for TSCLAS, for different configurations and values of D, range from 0.93 to 0.94, beating the best results of other compared classifiers. This behavior represents a minimal impact of gaps in TSCLAS, reinforcing our claim that, once we have the representativeness of the phenomena on the collected data, our performance is a matter of enough observations in the time series.

We evaluate the time spent by the methods to classify the time series with missing data. Without loss of generality, we take as an example the case for 50 % gaps for both training and prediction steps, presented in Figure 7.8. By comparing these times to the ones for the classification steps without gaps, presented in Figure 7.5, we can see similar behavior for all methods. However, while TSCLAS has a reduction in the time spent

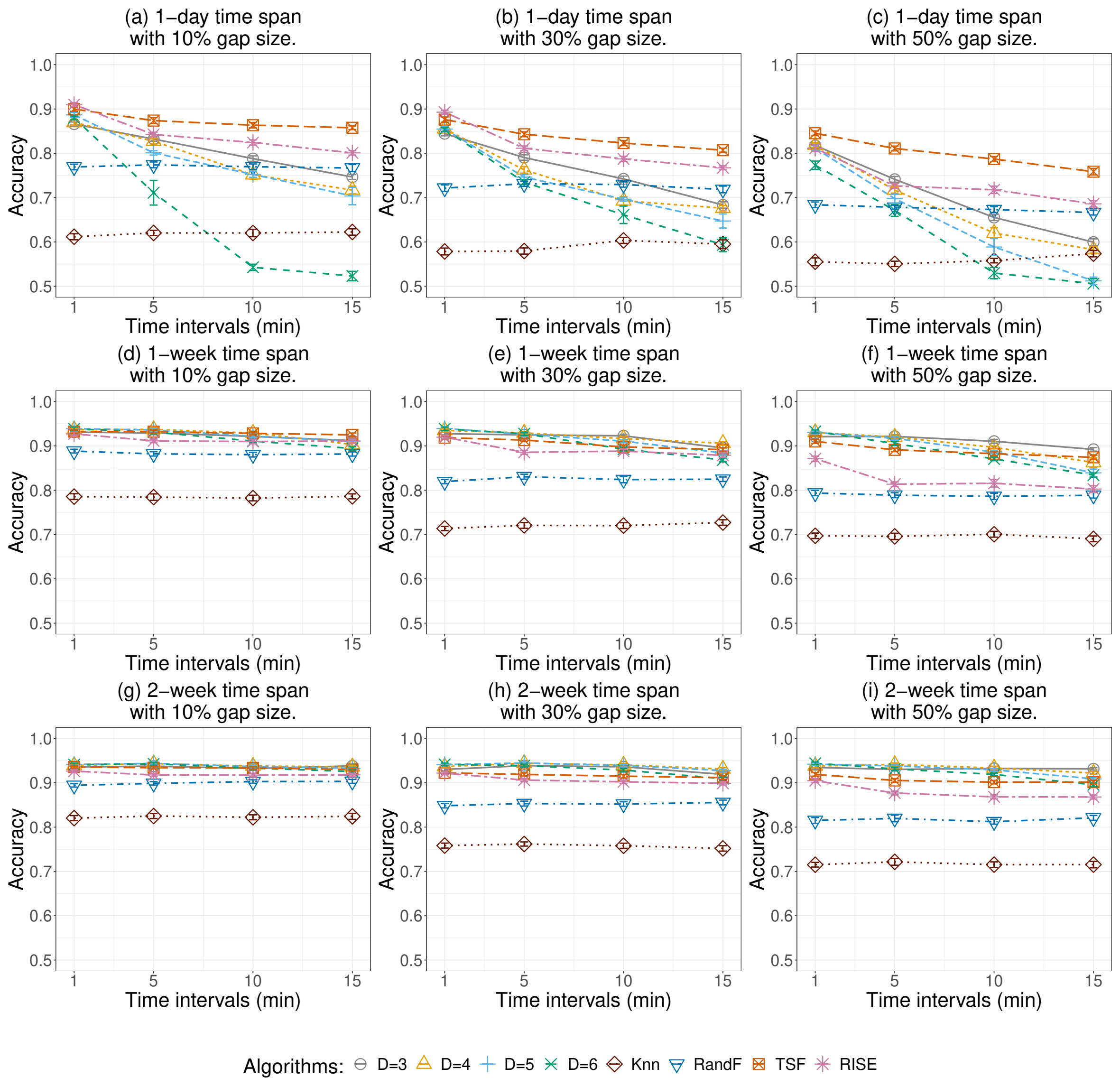


Figure 7.7. Classification accuracies of 1-minute ASOS time series with missing data, simulated as continuous gaps randomly positioned within the time series. Figures correspond to combinations 1-day, 1-week, and 2-week time spans, for gaps with 10 %, 30 %, and 50 % of the length of the time series, evaluated for different time intervals of 1, 5, 10, and 15 minutes between consecutive observations.

for both training and prediction steps in general, the other methods increased their spending times. This occurs due to the reduced time series length as the gap increases, which, for our method, also reduces the number of ordinal patterns to compute.

The methods that require some data imputation need more pre-processing time to adjust the data, increasing their total time. Take TSF, for instance, the most prominent case for a 2-week time span with 1-minute time interval. It increased its training and prediction times from approximately 4,700 and 1,040 seconds, to approximately 5,840

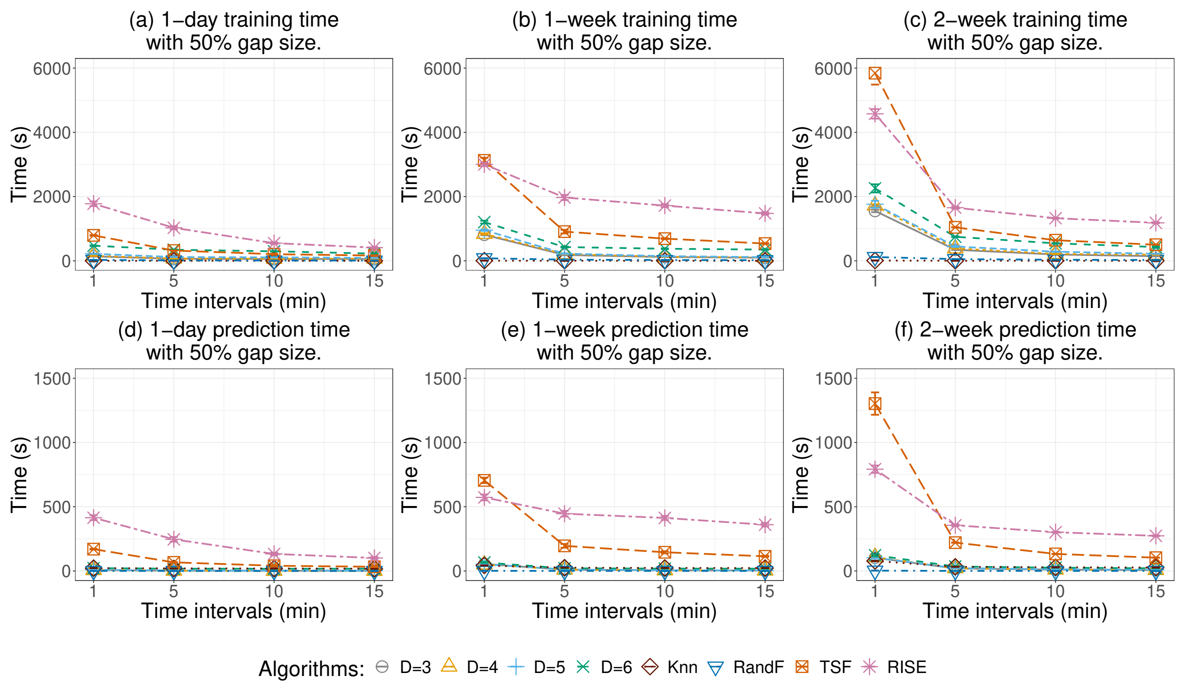


Figure 7.8. Average time spent for the classification of 1-minute ASOS time series with 50 % of time series samples as missing data. The missing data points were simulated as continuous gaps, randomly positioned within the time series. We consider evaluating the training step with (a) 1-day, (b) 1-week, and (c) 2-week times spans, and the prediction step for (d) 1-day, (e) 1-week, and (f) 2-week time spans.

and 1,300 seconds, respectively. Oppositely, our worst-case scenario, with $D = 6$, decreased its training and prediction times from approximately 2,750 and 125 seconds, to approximately 2,250 and 122 seconds, respectively. This indicates the valuable robustness of TSCLAS concerning the missing data problem in time series, a common issue present in real-world IoT data.

7.5.2.3 Classification of 1-hour time series

To evaluate TSCLAS in a broader sense, we perform the classification of the 1-hour ASOS time series. While the 1-minute ASOS correspond to time series from airports within the United States, the 1-hour ASOS consists of time series from around the world, as illustrated in Figure 7.2b. For this experiment, we consider the time series from all 3,742 airports as a single dataset, performing 30 classifications with randomized splits each, as described in Section 7.5.1. Figure 7.9a presents the classification accuracies achieved by the classifiers for different time spans, ranging from 1-month to 6-months.

We can see that all classifiers are impacted by the number of observations per time

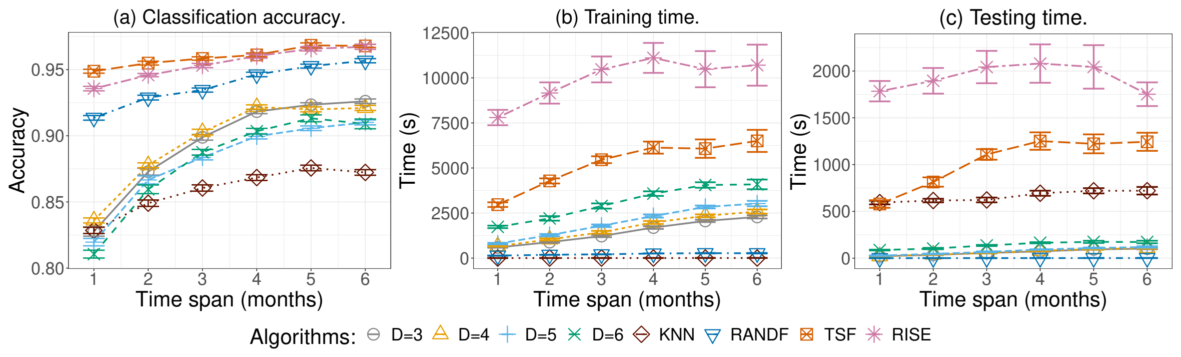


Figure 7.9. Results for the experiments on the 1-hour ASOS time series, consisting of (a) the classification accuracy, and times spent on (b) the training and (c) the prediction steps. All results were evaluated considering time spans of 1, 2, 3, 4, 5, and 6-months, with a starting date of January 1, 2020.

series, but TSCLAS is the most dependent on the time series length. As the length increases, our achieved accuracy also increases. It ranges from the worst accuracy level of 0.81 for $D = 6$ in the 1-month time span, with only 744 observations, to our highest accuracy of 0.93 for $D = 3$ in the 6-month time span, where there are 4,368 observations, as presented in Table 7.2. However, we can see by the tendency of the curves that we are bounded at this highest accuracy. This behavior occurs due to the low representativeness of the time series dynamics that we can capture from these phenomena by having a fixed 1-hour time interval. Unlike the 1-minute ASOS time series, where lower time intervals, up to 15 minutes, give reasonable representativeness for the phenomena dynamics, the 1-hour intervals are too much spaced. However, as can also be seen, this is not a problem for the others strategies in comparison with ours, where TSF reaches classification accuracies between 0.95 and 0.97, for the 1-month and 6-month configurations, respectively.

By looking at Figures 7.9b and 7.9c, which present the time for training and prediction, respectively, we can note that both TSF and RISE need more time to achieve their highest accuracies. On the other hand, KNN, which practically does not have operations to do in the training step, presents a considerable time for the prediction step. TSCLAS presents reasonable times for both steps, being below TSF and RISE in the training step and only losing for the RANDF algorithm in the prediction step. In fact, regarding the trade-off between classification times and achieved accuracy, for all algorithms, RANDF appears as the best choice for this 1-hour time interval scenario, with accuracies increasing from 0.91 to 0.95 and a short processing time at both steps.

In summary, we can conclude that, although TSCLAS is still competitive in this scenario, it is not the most favorable for it. On the contrary, time series with smaller granularities, such as the 1-minute scenarios, are more suitable. In fact, we can also note that, in the current IoT scenarios, it is more likely to have sensors collecting data at small intervals, even in seconds, than at 1-hour intervals. Thus, justifying the main objectives for TSCLAS to be a valuable strategy for the classification of IoT time series data according to their dynamical behavior.

7.6 Conclusion

In this chapter, we presented TSCLAS, a strategy for time series classification, using features extracted from ordinal patterns transformations, demonstrated to be a very suitable solution for classifying data generated by IoT sensors. TSCLAS is exceptionally adaptable to time series length and robust to missing data gaps. We conducted several experiments using real-world meteorological time series, leading to significant classification accuracies for both time series from 1-minute and 1-hour intervals between consecutive data points. The method achieved better results when the time series length are large enough and when the data points represent the measured phenomena well.

Besides that, TSCLAS presents lower computation times for training and testing, being competitive enough compared to the tested algorithms from the literature. By simulating a missing data problem as consecutive gaps in the time series, which is common and likely to occur in IoT scenarios, TSCLAS surpasses the algorithms from literature in our tests, even with gaps as long as 50 % of time series data points.

Furthermore, although further studies are needed to confirm it, we also argue that the proposed class separability analysis can be a valuable method for estimating the classification potential of ordinal patterns transformations. Preliminary data analysis for a given dataset can be helpful as a first step in knowing if its time series classes are distinguishable or not. Another future work includes the application of the proposed classification strategy to other datasets in other domains to assure its viability in different scenarios. Finally, we plan to evaluate the robustness of TSCLAS to other IoT common problems, such as precision, correction, and trustworthiness.

Chapter 8

IoT Botnet Detection based on Multiscale Time Series Dynamics

The detection of botnet attacks on the Internet of Things (IoT) is a critical challenge faced by the cyber-security community in recent years, given its high degree of severity. For instance, botnets are used to trigger Distributed Denial-of-Service (DDoS) attacks, which are further amplified by a large number of IoT devices. In this chapter we tackle this urgent task, which is motivated in Section 8.1. Section 8.2 discusses studies related to our proposed strategy, focusing on solutions for anomaly and botnet detections in IoT scenarios. Section 8.3 presents our proposed strategy for detecting botnets in IoT by the anomalies in their dynamical behavior, using the features obtained from the multiscale ordinal patterns transformations. Section 8.4 shows our experiments and results. Finally, Section 8.5 presents our conclusions and future directions.

8.1 Introduction

The growing number of Internet of Things (IoT) devices can create several new services and solutions to ease modern life [Atzori et al., 2010; Gubbi et al., 2013; Borges et al., 2019b]. Ranging from small sensors to powerful devices, the number of IoT-enabled devices is estimated to reach up to 25 billion in the next years [Schmeißer and Schiele, 2020]. However, because of their computational restrictions and misconfigurations, IoT devices are easy targets for attacks, making their security an urgent concern [Bertino and Islam, 2017; Kolias et al., 2017; Blaise et al., 2020].

A recurrent attack involving IoT devices in recent years, serving as the basis for other attacks, is their infection by bots. A bot is a malicious software that can be used for remote controlling these devices by an attacker, the botmaster. A network of devices compromised by bots is a botnet [Bertino and Islam, 2017]. The main security threat is that once a device is compromised, the power of an attacker goes from collecting data of its targets, such as password cracking and keylogging, to large-scale orchestrated attacks, such as spam delivery and Distributed Denial-of-Service (DDoS) attacks [Bertino and Islam, 2017; Kolias et al., 2017].

Furthermore, although an attack from a small IoT device might not be harmful, as pointed by aKolias et al. [2017], the number of IoT devices compensates for their lack of computational resources. For instance, between March and April 2019, a massive botnet attack that used more than 400,000 IoT devices worldwide was detected. This attack produced more than 292,000 requests per minute, aiming to affect the availability of a remote server[1]. This is an aspect that enhances the impact of DDoS attacks, making them hard to be handled by any powerful server.

Thus, following the threat model proposed by Abbas et al. [2020], and considering the severity and destructive potential of DDoS attacks once empowered by botnets of billions of compromised IoT devices [Al-Garadi et al., 2020], there is an urge for solutions to identify and mitigate their impact. However, since DDoS attacks are difficult to defend against, given the volatility and different attack patterns [Wang et al., 2018], the early detection of such digital threats is a reasonable solution. This is possible by evaluating anomalies in the typical behavior of IoT devices. Thus, given the expected behavior of IoT devices, the detection of any deviation can be signaled as an anomaly [García-Teodoro et al., 2009]. In fact, following the claim of Meidan et al. [2018], this approach is well suited for IoT since their devices are dedicated to a specific task, and sudden changes may indicate potential attacks of compromised devices [Bertino and Islam, 2017].

In this paper, our main objectives are to investigate the following research questions:

1. Do these behavioral changes in devices' operation affect their temporal dynamics?
2. In the affirmative case, can these dynamics be used to detect anomalous behavior, which represents an attack?

The temporal dynamics indicate how a system evolves, being an essential aspect for distinguishing data that changes its behavior as a function of time [Rosso et al., 2007a, 2015; Borges et al., 2019a]. We argue that this duality between IoT devices' typical and anomalous behavior is an aspect their temporal dynamics can capture. Thus, our

[1]Massive Botnet Attack Used More Than 400,000 IoT Devices. Accessed in April 04, 2021. `https://www.bankinfosecurity.com/massive-botnet-attack-used-more-than-400000-iot-devices-a-12841`.

proposed strategy for detecting botnet attacks in IoT consists of evaluating anomalies in the dynamical behavior of devices' network traffic during regular and attack periods, aiming to distinguish them.

For instance, once a device is infected by the Mirai malware [Kolias et al., 2017], one of the most prominent botnets for DDoS attacks, it automatically does some operational steps that change its network traffic. Some of them include scanning the network for new victims, sending reports and receiving commands from a Command and Control (C&C) server, and send attack traffic to a target server [Kolias et al., 2017; Meidan et al., 2018]. This change in the network traffic can be interpreted as an anomalous behavior and be used to detect a compromised device [Kolias et al., 2017].

To capture these dynamics, we use the ordinal patterns transformation, a cornerstone contribution from Bandt and Pompe [2002], described in Section 3.5. For this work, we add another intrinsic characteristic of dynamical systems: strong dependence on the scale used for sampling the signals [Zunino et al., 2012]. Thus, we consider evaluating the time series following a multiscale approach, i.e., using different time intervals to construct the ordinal patterns. With this approach, we can understand how different time series evolve as well as their temporal correlation dependence. In fact, the multiscale analysis of time series has been demonstrated to be useful information for the correct characterization of time series dynamics [Zunino et al., 2012; Borges et al., 2019a].

In summary, our main contributions to this work are:

1. We propose a strategy for detecting botnet attacks in IoT based on anomalies in the dynamics of network traffic captured by the ordinal patterns transformation.

2. We propose using a set of features extracted from the multiscale ordinal patterns transformations for representing the changes in dynamics of IoT devices, which are inputs for the anomaly detection algorithm.

3. We demonstrate that with the proper transformation, which is able to capture distinguishing aspects from the typical phenomena, it is possible to use simple methods for detecting anomalies in the behavior of devices.

With these contributions, we advance the state of the art of detecting IoT botnet attacks in a simpler and more efficient way than other strategies from literature. Following the strategy of Nomm and Bahsi [2018], we apply a shallow anomaly detection algorithm. Instead of using deep learning methods, we use the Isolation Forest, a model-free anomaly detection algorithm with linear time complexity and low memory requirements [Liu et al., 2008, 2012; Hariri et al., 2019]. In fact, from the experiments of Nomm and Bahsi [2018], the Isolation Forest was the algorithm with the best results.

8.2 Related Work

The network traffic data is an essential asset for monitoring and evaluating devices' behaviors in many domain areas. Strategies range from remote distinguishing operating systems [Medeiros et al., 2010] to the malfunction detection of devices by the unexpected behavior of their communications [Chandola et al., 2009; Boukerche et al., 2020]. Thus, information extracted from network traffic is of fundamental importance to the analysis of a device's behavior.

With respect to the security concerns of networked devices, unexpected behavior of a device may indicate it is hacked and performing undesired operations. Thus, an important strategy to discover security issues on devices is detecting anomalies in their network traffic. Chandola et al. [2009] presented a survey on anomaly detection techniques and applications. They noted that once a device is compromised by a worm, for example, the anomalous traffic is more frequent than the regular traffic. Likewise, García-Teodoro et al. [2009] discussed strategies for detecting abnormalities in network traffic and their important role for intrusion detection systems.

Many solutions were proposed to detect network traffic anomalies. [Agarwal and Mittal, 2012] proposal is based on the entropy of network measures. After extracting the network measures, they compute their normalized entropy which are further classified with Support Vector Machine (SVM) as normal or attack traffic. The work of Yu et al. [2013] consists of detecting flooding attacks by observing network measures of devices, which are collected by Simple Network Management Protocol (SNMP) requests. After collecting data, such as the number of received, sent, and error packets, they use a C4.5 decision tree to identify the attacks.

For the specific case of detecting botnet attacks, Wang et al. [2018] presented a study on the characterization and analysis of the behavior of 50,704 different Internet DDoS attacks observed during seven months. For our concerns, they reveal that understanding DDoS attack patterns is the key to defending against them. One of the most prominent patterns they identified is some periodicity of the inter-attack time interval for the DDoS attacks, which may be an interesting aspect for their detection.

Mirsky et al. [2018] proposed the Kitsune, a Network Intrusion Detection System (NIDS), which uses an ensemble of autoencoders to detect attacks on local networks. At the core of Kitsune is a feature extraction framework used for capturing measures from the network traffic, which serves as input for the autoencoders. This framework is used for extracting network measures that will serve as a basis for other related proposals, including this one.

Blaise et al. [2020] proposed a Collaborative Intrusion Detection System (CIDS) for detection of anomalies in network traffic by considering the aggregated traffic at target ports. They assume that, once a device is compromised, the sudden rise in traffic towards a port is the first steps of their operation. The authors claim that it is possible to early identify a botnet attack by focusing on port-based detection.

Given the increase of IoT botnets and their harmful impact on current network threats, many studies have investigated this problem focusing on anomaly detection. Meidan et al. [2018] proposed detecting IoT botnets using autoencoders. They follow the Kitsune method [Mirsky et al., 2018] for extracting network traffic measures, which is the input for their autoencoders. The measures were extracted from real IoT devices during normal (benign) and attack operations, which were infected by the Mirai and Bashlite botnets [Bertino and Islam, 2017; Kolias et al., 2017]. The resulting N-BaIoT dataset, detailed in Section 8.4.1, is publicly available and used for other studies.

Nomm and Bahsi [2018] proposed a strategy for detecting IoT botnets based on shallow methods, rather than using deep learning models as Meidan et al. [2018], which the authors affirm to be more appropriated for IoT devices. The authors propose a feature selection approach to consider only the best network traffic measures for further classification as benign and attack event. Alqahtani et al. [2020] also used the N-BaIoT dataset for their experiments to detect IoT botnet attacks. The authors propose a feature selection strategy for simplifying the number of network measures in the original N-BaIoT dataset. With the best features, they use eXtreme Gradient Boosting (XGBoost) as a classification method. Unlike the other proposals, which used only the benign traffic data for training their models, in this last one, the authors considered both benign and attack data for the model training step. Popoola et al. [2021] proposed a federated deep learning method for zero-day botnet attack detection in IoT edge devices. Their strategy uses a federated averaging strategy to aggregate all local models from edges, producing a global deep neural network model. The authors used a subset of five attacks from the N-BaIoT dataset.

To put our work in perspective, let us compare it with the previous related work. Our focus is to detect anomalies in the N-BaIoT network traffic dataset, collected by Meidan et al. [2018]. Thus, we can compare our results with their work and the work of Nomm and Bahsi [2018], Alqahtani et al. [2020], and Popoola et al. [2021] which also use the same dataset. Our first difference from them is in the number of network measures used to detect anomalies. These works use all the 115 measures from the

N-BaIoT or select the 3 to 10 best ones. Instead, we only consider one single network traffic measure, the number of packets sent from the device.

8.3 Detection of botnet attacks in IoT

This section presents our strategy of using the ordinal patterns transformations for detecting anomalies in IoT devices. Our proposed strategy consists of three main parts:

1. Construct time series from the number of packets sent by a suspicious device directly from its network traffic.
2. Transform time series, via the multiscale ordinal patterns transformation, onto a set of features that can capture the dynamical behavior of the devices' operation.
3. Classify the transformed time series to distinguish an anomalous from a regular devices' behavior.

In the following, we detail this proposed strategy.

8.3.1 The network traffic capture

Figure 8.1 illustrates the network model used to capture the traffic of packets from a given suspicious IoT device, following the Kitsune framework [Mirsky et al., 2018]. Let us suppose a scenario where all the traffic from a given device flows through a local router. Thus, an agent within this router will passively capture and parse the received packets to compute network traffic measures.

Our strategy only needs data from a single network traffic measure, the number of packets from a device, aggregated within a time window of 100 ms. However, since we evaluate the evolution of this measure over time, we need to collect consecutive samples of it to create a time series, which will be further used to detect the anomalies.

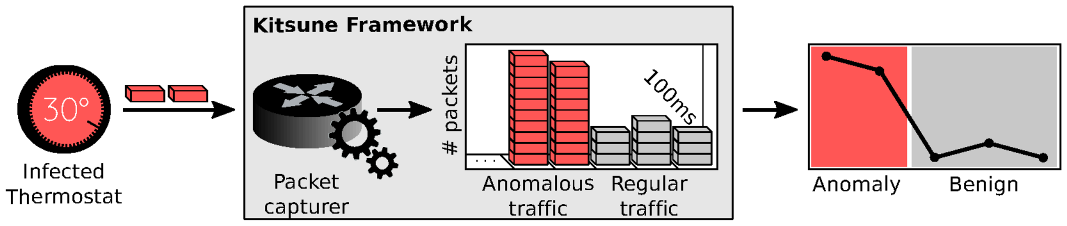


Figure 8.1. Network model to capture the IoT devices' traffic.

The next step consists of transforming the constructed time series by the multiscale ordinal patterns transformation, described in Section 8.3.2. After this transformation, we will have a set of relevant features to represent the characteristics from the distinct dynamics of the devices' operations. Then, these features are the input for the Isolation Forest anomaly detection algorithm [Liu et al., 2008], described in Section 8.3.3.

8.3.2 The multiscale ordinal patterns transformation

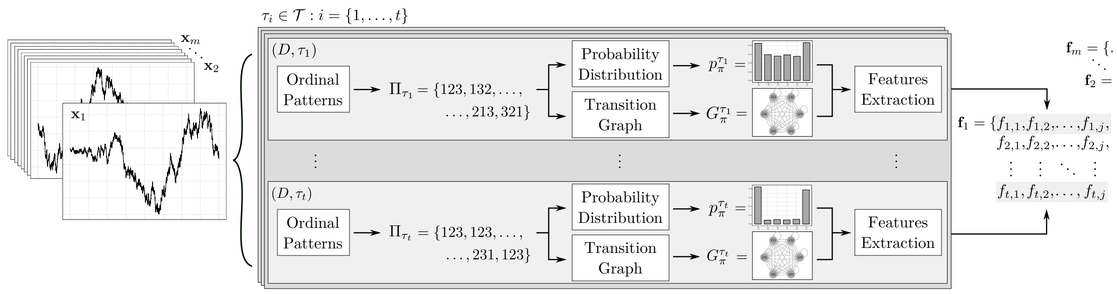

$\tau_i \in \mathcal{T} : i = \{1, \ldots, t\}$
$\mathbf{x}_m$
$\mathbf{x}_2$
$\mathbf{x}_1$
(D, τ_1)
Ordinal Patterns
$\Pi_{\tau_1} = \{123, 132, \ldots, \ldots, 213, 321\}$
Probability Distribution
Transition Graph
$p_\pi^{\tau_1} =$
$G_\pi^{\tau_1} =$
Features Extraction
(D, τ_t)
Ordinal Patterns
$\Pi_{\tau_t} = \{123, 123, \ldots, \ldots, 231, 123\}$
Probability Distribution
Transition Graph
$p_\pi^{\tau_t} =$
$G_\pi^{\tau_t} =$
Features Extraction
$\mathbf{f}_m = \{\ldots\}$
$\mathbf{f}_2 = \{\ldots\}$
$\mathbf{f}_1 = \{f_{1,1}, f_{1,2}, \ldots, f_{1,j}, f_{2,1}, f_{2,2}, \ldots, f_{2,j}, f_{t,1}, f_{t,2}, \ldots, f_{t,j}$

Figure 8.2. Process for the transformation of a time series $\mathbf{x}_i$ onto a new vector of features $\mathbf{f}_i$, based on the multiscale ordinal patterns transformations, for a given embedding dimension D and a list of embedding delays $\mathcal{T}$ of length t.

Figure 8.2 depicts the detailed process to perform the transformations necessary on the time series. For a given time series $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,m})$, of length m, the ordinal patterns transformation is applied as described in Section 3.5.1. The parameters are D and $\tau_i \in \mathcal{T}$, with $i = \{1, \ldots, t\}$, corresponding to a given embedding dimension and an embedding delay (time interval) from the list, respectively.

For each pair (D, τ_i), the result of this step is a set of ordinal patterns Π_{τ_i} representing the transformation of the time series with D and $\tau_i \in \mathcal{T}$ as parameters. Thus, given the list of considered embedding delays, each time series is transformed into t different sets of ordinal patterns.

Once a set of ordinal patterns Π_{τ_i} is obtained, it is transformed onto both an ordinal patterns probability distribution ($p_\pi^{\tau_i}$) and an ordinal patterns transition graph ($G_\pi^{\tau_i}$), as described in Sections 3.5.2 and 3.5.3, respectively. From both $p_\pi^{\tau_i}$ and $G_\pi^{\tau_i}$, a set of features is extracted, consisting of the new representation for the time series $\mathbf{x}_i$, which will be used to create the new features vector. Therefore, for each time series $\mathbf{x}_i$ and for each pair (D, τ_i), we have the set of features

$$\mathbf{f}_{\tau_i} = \{f_{\tau_i,1}, f_{\tau_i,2}, \ldots, f_{\tau_i,j}\}, \tag{8.1}$$

where j is the number of features extracted.

In this work, from a given time series $\mathbf{x}_i$, there is a total of $j = 8$ features extracted for each pair (D, τ_i). These are the features presented in Sections 3.5.4, plus the probability of self-transitions p_{st}, described in Section 5.5. Thus, the vector presented in Equation 8.1 can be instantiated to the following features,

$$\mathbf{f}_{\tau_i} \quad = \quad \{N_E^{\tau_i}, H_S[E_w]^{\tau_i}, C_{JS}[E_w]^{\tau_i}, F[E_w]^{\tau_i}, p_{st}^{\tau_i}, H_S[p_\pi]^{\tau_i}, C_{JS}[p_\pi]^{\tau_i}, F[p_\pi]^{\tau_i}\}. \tag{8.2}$$

The next step for transforming the time series corresponds to the composition of a vector $\mathbf{f}$ of all extracted features for each $\tau_i \in \mathcal{T}$, representing the final transformation. To illustrate this step, Equation 8.3 shows the set of features $\mathbf{f}$ constructed from the time series $\mathbf{x}$.

$$\mathbf{f} = \bigcup_{i=1}^{t} \mathbf{f}_{\tau_i} = \bigcup_{i=1}^{t} \{f_{\tau_i,1}, f_{\tau_i,2}, \ldots, f_{\tau_i,j}\} \tag{8.3}$$

After these steps, the new ordinal patterns domain for the raw time series corresponds to a new features vector of length tj. Algorithm 9 describes the transformation process of a given time series onto this novel features representation, following the multiscale ordinal patterns transformations.

ALGORITHM 9: MultiscaleTransformation

```
Input: A time series $\mathbf{x} = \{x_1, \ldots, x_m\}$ of length $m$, an embedding dimension $D$ and a
       list of embedding delays $\mathcal{T}$ of length $t$.
Output: The resulting $\mathbf{f} = \{f_1, \ldots, f_{tj}\}$ features after the multiscale transformations
        process.
// Transforming the time series for each $\tau$
1 foreach $\tau_i \in \mathcal{T}$ do
      // Main transformation
2     $\Pi_{\tau_i} \leftarrow$ ordinalPatterns($\mathbf{x}, D, \tau_i$);
      // Probability Distribution
3     $p_\pi^{\tau_i} \leftarrow$ ordinalPatternsPD($\Pi_{\tau_i}, D$);
      // Transition Graph
4     $G_\pi^{\tau_i} \leftarrow$ ordinalPatternsTG($\Pi_{\tau_i}, D$);
      // Extracting the features
5     $\mathbf{f}_{\tau_i} \leftarrow$ extractFeatures($p_\pi^{\tau_i}, G_\pi^{\tau_i}$);
// Joining features from all $\tau$'s as a single features vector
6 $\mathbf{f} = \{\mathbf{f}_{\tau_1}, \ldots, \mathbf{f}_{\tau_t}\}$;
7 return $\mathbf{f}$;
```

8.3.3 The anomaly detection strategy

For detecting anomalies in IoT devices, our method requires time series samples to learn its behavior. These time series are constructed by consecutive observations of a single network traffic measure at distinct moments. Thus, for a given network measure k, our proposal consists of training a model with the time series observations of k during the benign operation of the devices and, then, performing the anomaly detection for unseen observations.

8.3.3.1 The training phase

For training the model, let us assume the dataset of time series from a given network measure k, collected during its benign operation, is available. Let the matrix $\mathbf{X}^k_{n\times m} = [\mathbf{x}^k_1, \mathbf{x}^k_2, \ldots, \mathbf{x}^k_n]^\top$ be this dataset consisting of n distinct time series for a specific network measure k. For the sake of clarity, let us assume each time series $\mathbf{x}^k_i = x^k_{i,1}, \ldots, x^k_{i,m}$ consists of consecutive samples and has the same length m. However, it is important to highlight that the method does not require time series of equal lengths.

After performing the multiscale transformation for each time series, described in Section 8.3.2, the novel matrix of features vectors $\mathbf{F}^k_{n\times tj} \leftarrow [\mathbf{f}^k_1, \mathbf{f}^k_2, \ldots, \mathbf{f}^k_n]^\top$ is obtained. This matrix of features will be used for training a model with the Isolation Forest anomaly detection algorithm. The processes for training the model c^k, specific for the network traffic measure k, is presented in Algorithm 10.

```
ALGORITHM 10: ModelTraining
Input: The time series dataset $\mathbf{X}^k_{n\times m}$ for a specific network traffic measure, an
       embedding dimension $D$, and a list of embedding delays $\mathcal{T}$ of length $t$.
Output: The resulting classification model $c^k$ for the network traffic measure $k$.
// Transforming each time series from the dataset onto a set of features
1 foreach $\mathbf{x}^k \in \mathbf{X}^k$ do
      // Main transformation
2     $\mathbf{f}^k \leftarrow$ MultiscaleTransformation($\mathbf{x}^k, D, \mathcal{T}$);
// The matrix of transformed features
3 $\mathbf{F}^k_{n\times tj} \leftarrow [\mathbf{f}^k_1, \mathbf{f}^k_2, \ldots, \mathbf{f}^k_n]^\top$
// Fitting a model
4 $c^k \leftarrow$ IsolationForest($\mathbf{F}^k, n_{\text{trees}}$);
5 return $c^k$;
```

8.3.3.2 The anomaly detection phase

For the anomaly detection phase, a given time series of unseen observations is passed through the isolation forest model c^k, which results in an anomaly score s. According to the value of s, this time series will be concluded as a regular time series, i.e., similar to the time series presented to the model during its training or an anomalous time series.

Thus, let $\mathbf{y}^k = (y_1^k, \ldots, y_m^k)$ be a time series of m suspicious observations from the network traffic measure k. As in the case when training the model, the observations are consecutive samples collected from the same network measure k. After computing the features vector $\mathbf{f}^k$, with the multiscale ordinal patterns transformation, its anomaly score s is computed with the isolation forest model c^k.

The behavior of the given time series is predicted by simply evaluating the value of s according to a defined anomaly threshold e. As originally defined by Liu et al. [2008], significant anomaly scores potentially indicate an anomaly. Section 8.4.1 gives more details about the value for this threshold as a limit for our detection strategy. Algorithm 11 illustrates these steps for the anomaly detection phase.

ALGORITHM 11: AnomalyDetection

Input: A suspicious time series $\mathbf{y}^k$ for the network traffic measure k, the isolation forest model c^k, an anomaly score threshold e, an embedding dimension D, and a list of embedding delays $\mathcal{T}$ of length t.

Output: The detection result of the suspicious time series as *benign* or *attack* traffic.

```
   // Applying the multiscale transformation
1  $\mathbf{f}^k \leftarrow$ MultiscaleTransformation($\mathbf{y}_k, D, \mathcal{T}$);
   // Computing the anomaly score
2  $s \leftarrow$ predictInstance($c^k, \mathbf{f}^k$);
   // Evaluate the time series behavior
3  if $s < e$ then
4      return benign;
5  else
6      return attack;
```

8.4 Results and Discussion

This section presents our results and discussions about the botnet detection in IoT network traffic, using the identification of anomalies in the behavior of their temporal dynamics. To evaluate our approach, we compare our results with related work in the literature by performing experiments on the real-world N-BaIoT network traffic dataset [Meidan et al., 2018]. However, we first give more details on the chosen methods and decisions, such as the data preprocessing, software, and packages versions.

8.4.1 Materials and methods

The experiments presented in this work used the N-BaIoT public dataset[2]. This dataset, created by Meidan et al. [2018], consists of network traffic collected from nine IoT devices during both their regular operation (benign) and under botnet attacks (anomaly). The list of devices includes one thermostat, one baby monitor, one webcam, two doorbells, and four security cameras. The anomalous network traffic was generated by infecting each device with the Mirai and Bashlite botnet families [Bertino and Islam, 2017; Kolias et al., 2017].

The anomalous network traffic was generated by infecting each device with the Mirai and Bashlite botnet families [Bertino and Islam, 2017; Kolias et al., 2017]. These are two of the most prominent botnets that are still a risk for the current IoT. For instance, although the Mirai source code was released in late 2016 [Kolias et al., 2017], in 2021, we still find numerous Mirai-based botnets infecting and exploring vulnerabilities of IoT devices[3]. Furthermore, the operation of a device once infected by these botnets is similar to the original botnets they are derived from, such as contacting a C&C server, flooding the network, and scanning for other vulnerable devices. Thus, this makes the N-BaIoT dataset still relevant for evaluating anomalous behavior.

For each botnet, the authors captured the network traffic data under five different attacks. For the Mirai botnet, the attacks were:

1. ACK flooding;
2. SCANning the network for new victims;
3. SYN flooding;
4. UDP flooding; and
5. UDPPLAIN, which is another UDP flooding but with fewer options.

For the Bashlite botnet, the attacks were:

1. SCANning the network for new vulnerable devices;

[2]N-BaIoT dataset is available to download at the UCI Machine Learning Repository: `http://archive.ics.uci.edu/ml/datasets/detection_of_IoT_botnet_attacks_N_BaIoT`.

[3]Dark Mirai botnet targeting RCE on popular TP-Link router. Accessed in Dec. 14, 2021. `https://www.bleepingcomputer.com/news/security/dark-mirai-botnet-targeting-rce-on-popular-tp-link-router/`.

2. TCP flooding;
3. UDP flooding;
4. JUNK, which consists of sending spam data; and
5. COMBO of sending spam data and connecting to a specific IP address and port.

Thus, there are 11 distinct classes, ten classes of attacks, and one class of benign data, for each of the nine IoT devices.

Table 8.1 presents the dataset properties, illustrating the devices used for the experiments and the numbers of samples for each of the 11 classes. It can be observed that there is no traffic data from the Mirai botnet for two devices, only for the Bashlite. Another important aspect, as Alqahtani et al. [2020] pointed out, is the unbalanced length of classes, with the Mirai botnet dominating the number of samples and the benign data being the class with fewer samples.

To collect the network traffic measures for each of these samples, the authors followed the method proposed by Mirsky et al. [2018], the Kitsune network intrusion detection system. Thus, for each one of these 11 classes, for each IoT device, after capturing the raw traffic data from those devices, the authors aggregated the samples within five different time windows: 1 min, 10 s, 1.5 s, 500 ms, and 100 ms; which are coded as their decay factor λ as L0.01, L0.1, L1, L3, and L5, respectively. Furthermore, they computed 23 distinct network traffic measures for each of these time windows, consisting of statistics from the packet's sender and the traffic between the packet's sender and receiver [Mirsky et al., 2018]. Consequently, each sample of Table 8.1 consists of a snapshot with 115 distinct network traffics measures.

To handle the samples from the N-BaIoT dataset, the other related work, presented in Section 8.2, uses as input for their anomaly detection methods the samples snapshot directly, whether it be the total 115 measures or a subset of them. Instead, our method needs to construct a time series from consecutive samples of a single network measure to learn their temporal dynamics. Thus, for our experiments, we chose the `MI_dir_L5_weight` network traffic measure, as the dataset notation, to be our k measure, as described in Section 8.3.3. This measure consists of the number of packets (weight) originated from a single device, represented by a pair of source MAC and IP addresses (MI), aggregated by time windows of 100 ms (L5), constructed in an incremental approach [Mirsky et al., 2018]. The reason for choosing this measure follows the results of Alqahtani et al. [2020], which analyzed the measures' importances by their Fisher scores. However, differently from those authors, we chose the measure with the smallest time window, while they chose the one with the highest time window (L0.01).

Table 8.1. Description of the N-BaIoT dataset, illustrating the IoT devices used in the experiment and the number of samples for each benign and attack classes.

#	**Device**	**Benign**	**Mirai**		**Bashlite**	
1	Danmini Doorbell	49,548	ACK	102,195	COMBO	59,718
			SCAN	107,685	JUNK	29,068
			SYN	122,573	SCAN	29,849
			UDP	237,665	TCP	92,141
			UDPPLAIN	81,982	UDP	105,874
			TOTAL:	652,100	**TOTAL**:	316,650
2	Ecobee Thermostat	13,113	ACK	113,285	COMBO	53,012
			SCAN	43,192	JUNK	30,312
			SYN	116,807	SCAN	27,494
			UDP	151,481	TCP	95,021
			UDPPLAIN	87,368	UDP	104,791
			TOTAL:	512,133	**TOTAL**:	310,630
3	Ennio Doorbell	39,100	–	–	COMBO	53,014
			–	–	JUNK	29,797
			–	–	SCAN	28,120
			–	–	TCP	101,536
			–	–	UDP	103,933
			TOTAL:	0	**TOTAL**:	316,400
4	Philips B120N10 Baby Monitor	175,240	ACK	91,123	COMBO	58,152
			SCAN	103,621	JUNK	28,349
			SYN	118,128	SCAN	27,859
			UDP	217,034	TCP	92,581
			UDPPLAIN	80,808	UDP	105,782
			TOTAL:	610,714	**TOTAL**:	312,723
5	Provision PT 737E Security Camera	62,154	ACK	60,554	COMBO	61,380
			SCAN	96,781	JUNK	30,898
			SYN	65,746	SCAN	29,297
			UDP	156,248	TCP	104,510
			UDPPLAIN	56,681	UDP	104,011
			TOTAL:	436,010	**TOTAL**:	330,096
6	Provision PT 838 Security Camera	98,514	ACK	57,997	COMBO	57,530
			SCAN	97,096	JUNK	29,068
			SYN	61,851	SCAN	28,397
			UDP	158,608	TCP	89,387
			UDPPLAIN	53,785	UDP	104,658
			TOTAL:	429,337	**TOTAL**:	309,040
7	Samsung SNH 1011 N Webcam	52,150	–	–	COMBO	58,669
			–	–	JUNK	28,305
			–	–	SCAN	27,698
			–	–	TCP	97,783
			–	–	UDP	110,617
			TOTAL:	0	**TOTAL**:	323,072
8	SimpleHome XCS7 1002 WHT Security Camera	46,585	ACK	111,480	COMBO	54,283
			SCAN	45,930	JUNK	28,579
			SYN	125,715	SCAN	27,825
			UDP	151,879	TCP	88,816
			UDPPLAIN	78,244	UDP	103,720
			TOTAL:	513,248	**TOTAL**:	303,223
9	SimpleHome XCS7 1003 WHT Security Camera	19,528	ACK	107,187	COMBO	59,398
			SCAN	43,674	JUNK	27,413
			SYN	122,479	SCAN	28,572
			UDP	157,084	TCP	98,075
			UDPPLAIN	84,436	UDP	102,980
			TOTAL:	514,860	**TOTAL**:	316,438
	TOTAL	555,932	3,668,402		2,838,272	

Concerning the parameters chosen for the algorithms and experiments, we varied the length m of the time series constructed from the selected measure in our experiments to check its impact on the detection results. We also evaluated the values of the embedding dimension D, and the length t of the embedding delays list $\mathcal{T}$ by varying their values. The number of trees used in our experiments was $n_{\text{trees}} = 300$, which provided good results with reasonable execution time. The remaining Isolation Forest parameters was kept as default. For the anomaly score threshold e, which is used to conclude a benign or attack behavior, as described in Section 8.3.3, we chose the fixed value of $e = 0.59$. The original Isolation Forest work [Liu et al., 2008] partially supports the reason for this choice since it shows that potential anomalies can be identified when the anomaly score is $s \geq 0.6$. However, by empirical observations, we obtained the value of $e = 0.59$ was more appropriate, giving better detection results.

Our algorithms and experiments were implemented in the R statistical software suite [R Core Team, 2018], version 4.0.3, with some code excerpts in C++. For the Isolation Forest anomaly detection, we used the R package `isotree`[4] version 0.2.7. All codes used in our experiments are available at a public repository[5]. All experiments were performed on a computer with Intel Core i9-9900X CPU at 3.50 GHz x 20, 128 GB RAM, and running a Linux Ubuntu 18.04.4 LTS 64-bit.

8.4.2 Temporal dynamics of a Botnet attack

The success of the whole anomaly detection process depends on the right choice of the features and their ability to represent the different behaviors from the time series. When dealing with the ordinal patterns transformations, this involves the proper definition of the parameters (D, τ).

The embedding dimension D depends on the length n of the time series, since it will define the alphabet of size $D!$ to which the possible patterns will be mapped. As previously discussed in Section 3.5.1, the main recommendation is that the condition $n \gg D!$ must be satisfied to obtain reliable statistics [Rosso et al., 2007a; Zunino et al., 2012]. Given the rapid increase of the factorial function, for practical purposes, Bandt and Pompe [2002] recommended $D \in \{3, \dots, 7\}$.

For the embedding delay τ, although there is no strict recommendation, it is already known that τ is significantly related to some intrinsic time-space characteristics of the series, such as periodicity and time scales [Zunino et al., 2012]. While a common choice of several studies is $\tau = 1$, we show that different values of τ enable the features

[4]isotree: Isolation-Based Outlier Detection: `https://cran.r-project.org/web/packages/isotree/index.html`.

[5]The repository with the code for our IoT Botnet Detection is available at `https://github.com/labepi/anomaly_iot`. This code requires the Bandt-Pompe ordinal patterns transformations, available at `https://github.com/labepi/bandt_pompe`.

extracted from both p_π and G_π to result in a more separable space between the different classes.

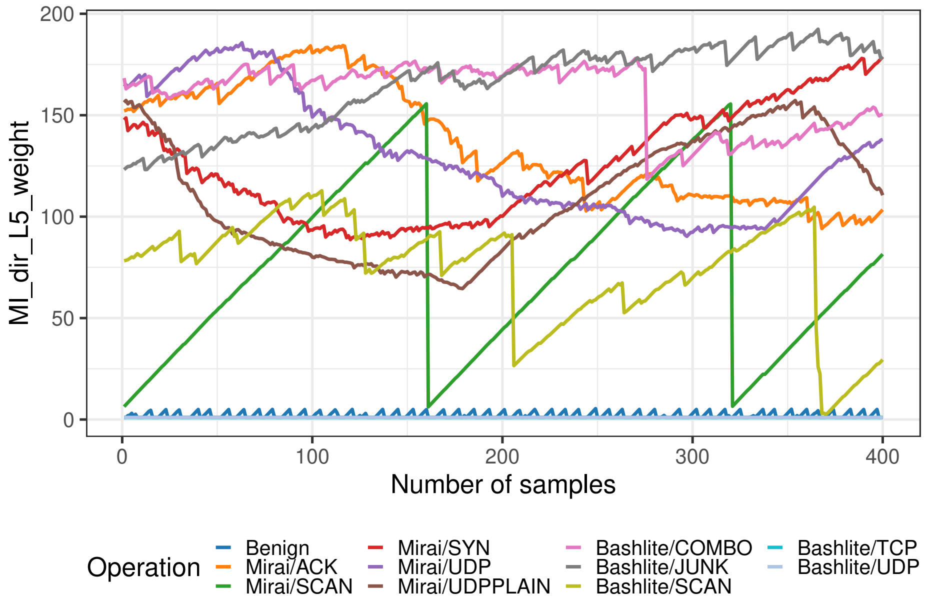


Figure 8.3. Illustration of the first 400 samples of the chosen MI_dir_L5_weight network measure for different operations of the Danmini Doorbell device of the N-BaIoT dataset.

Figure 8.3 illustrates the first 400 samples of the chosen `MI_dir_L5_weight` network measure for different operations of the Danmini Doorbell device from the N-BaIoT dataset. It can be noticed that different moments of the operation result in very distinct behaviors for this measure. For instance, the samples collected during the benign operation have a pattern similar to the Additive Increase, Multiplicative Decrease (AIMD) operation of TCP. Instead, other botnet operations, such as the ACK or SYN operations from Mirai, are more random, with no apparent pattern.

By considering the temporal dynamics of these behaviors, we can observe some patterns that can be used for distinguishing them. Figure 8.4 presents the Causality Complexity-Entropy Plane (CCEP), proposed by Rosso et al. [2007a] and described in Section 3.6.2, computed from the time series of Figure 8.3, with $D = 3$ and $\tau = 1$. The CCEP is a plane formed by the $H_S[p_\pi]$ and $C_{JS}[p_\pi]$ features, computed from the ordinal patterns probability distribution p_π, as the x-axis and the y-axis of the plane, respectively.

According to the $(H_S[p_\pi], C_{JS}[p_\pi])$ pair, different time series dynamics are expected to be placed at specific regions of the plane, bounded by minimum and maximum limits of the statistical complexity. In general, random dynamics are placed to the right of the plane, while the more deterministic behaviors are placed to the left [Rosso

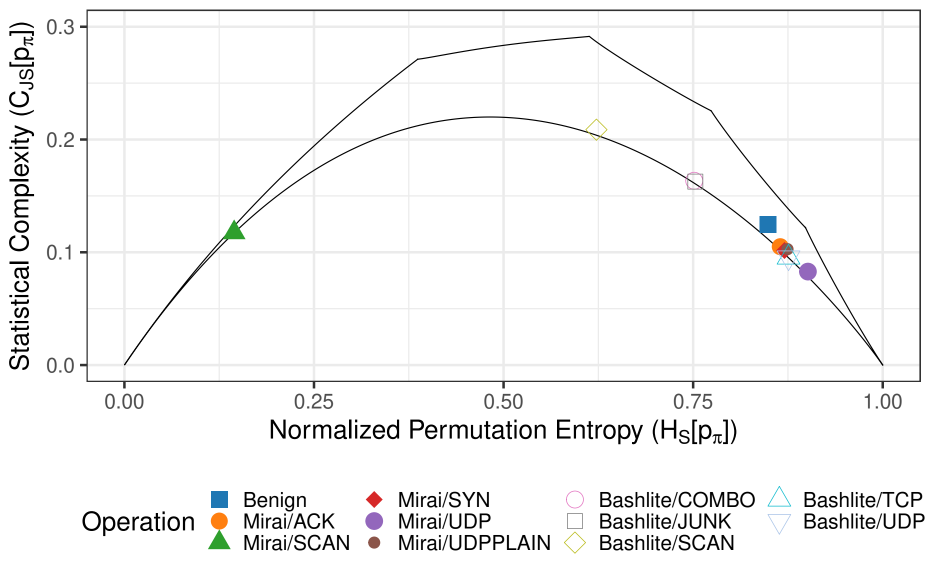


Figure 8.4. Illustration of the CCEP for the different operations of the MI_dir_L5_weight feature, presented in Figure 8.3, computed with $D = 3$ and $\tau = 1$.

et al., 2007a; Ravetti et al., 2014; Borges et al., 2019a]. For instance, the SCAN operation of Mirai botnet presents the highest deterministic dynamics, followed by the SCAN of the Bashlite botnet. While the COMBO and JUNK operations of Bashlite have a slight determinism, the other operations present a more random behavior.

This clear distinction of SCAN is an essential asset for botnet detection since it is one of their most common operations leading to applying the method for other botnets [Bertino and Islam, 2017; Kolias et al., 2017]. However, the distinction between regular and other attack operations is not clear in the CCEP. To address this, Figure 8.5 presents the feasibility of our multiscale approach for a better distinction of those different operations. It illustrates the behavior of the p_{st} feature, described in Section 5.5, evaluated as a function of the embedding delay $\tau \in \{1, \ldots, 10\}$, with fixed $D = 3$. Other features are not presented to save space. It can be observed that as τ increases, the separation between the device's regular and other attack operations is easier to distinguish.

8.4.3 IoT botnet detection results

This section presents our results for the detection of botnets in IoT, by applying this multiscale approach to the samples from the N-BaIoT dataset.

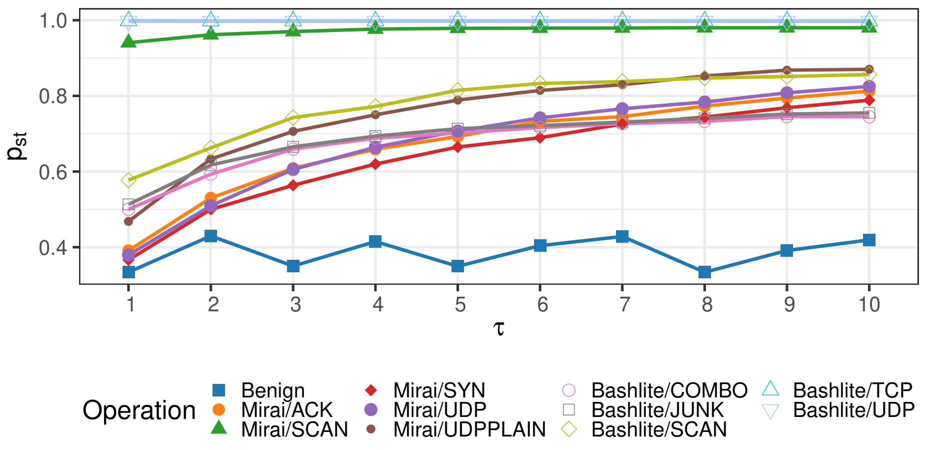


Figure 8.5. Illustration of the probability of self transition (p_{st}) from the MI_dir_L5_weight measure, during distinct operations of a device, presented in Figure 8.3, evaluated for $D = 3$ and $\tau \in \{1, \ldots, 10\}$.

8.4.3.1 One model for all devices

This first experiment consists of creating a single model for all dataset devices, which are used for detecting anomalies for each of them. It is based on a similar experiment made by Nomm and Bahsi [2018], used to evaluate the performance of one model created regardless of the device.

To create this model, we first have to construct time series from consecutive samples of the original dataset, as described in Section 8.4.1. Once the time series for both benign and attack data are constructed considering all devices, we used two-thirds of the benign time series for training phase. The remaining one-third of the benign time series plus all the attack time series are used for testing phase. This split is similar to the experiments in Meidan et al. [2018], except that, since we do not have an optimization parameter phase, all the two-thirds are used for training.

Since the time series length is an important aspect to consider by the ordinal patterns transformations, we consider evaluating the accuracy of our model for different time series lengths. Figure 8.6 presents the achieved accuracies for our method, combining different parameters of $D = \{3, 4\}$ and $t = \{5, 10\}$. This last parameter t is used for the maximum value of τ in the multiscale approach, i.e., $\tau_i \in \mathcal{T}$, with $i = \{1, \ldots, t\}$.

It can be noticed that the accuracies increase with the length m of time series presented to the method, with significant results occurring for $m \geq 500$. Also, the best results are achieved when $D = 3$ and $t = 10$, reaching the maximum value of 99.5 % for larger values of m. This result beats Nomm and Bahsi [2018], which achieved a maximum of 95.6 % accuracy in their similar experiment, and it is also better than

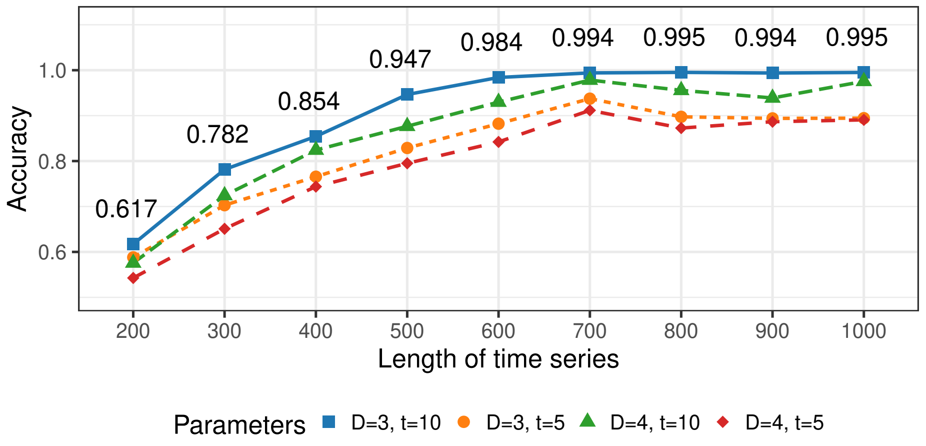


Figure 8.6. Accuracy results for the one model for all devices experiment, evaluated as a function of the time series length, combinations of D and t, and maximum τ.

Popoola et al. [2021], which achieved an average accuracy of 99 % with their federated learning strategy. It is also important to emphasize that reaching the best values with $D = 3$ is a fundamental aspect of the method, since larger values of D require more processing time. Table 8.2 summarizes the classification accuracies comparing our strategy, for different configurations of D and τ with time series length $n = 1000$, with the related work from literature.

Table 8.2. Classification accuracies for different configurations of D and t, with time series length $n = 1000$, comparing with the related work in the one model for all devices experiment. Best result is highlighted in bold.

Work		Accuracy (%)
Our strategy	$D = 3, t = 5$	89.4 %
	$\mathbf{D = 3, t = 10}$	**99.5** %
	$D = 4, t = 5$	89.0 %
	$D = 4, t = 10$	97.5 %
Nomm and Bahsi [2018]		95.6 %
Popoola et al. [2021]		99.0 %

For the detection performance, the method's sensitivity and specificity results for the configuration with the best accuracy ($D = 3$ and $t = 10$) are presented in Figure 8.7. The sensitivity, which measures the proportion of benign (positives) correctly identified, is practically constant for all lengths, with a slight decay for larger values. However, the essential aspect to assess in our case is the achieved specificity, which accounts the

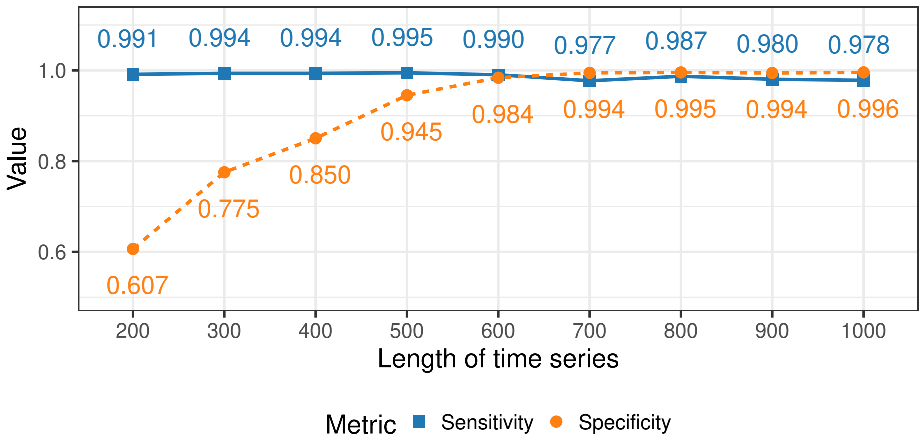
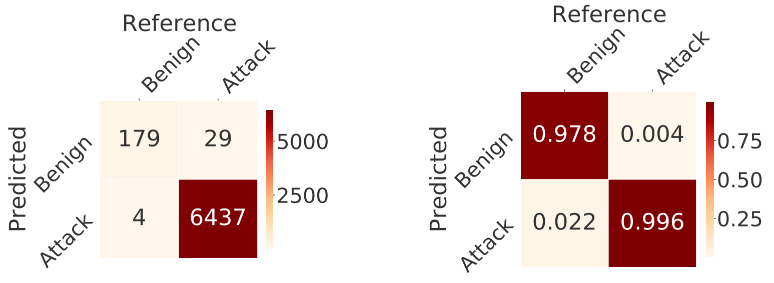


Figure 8.7. Sensitivity and specificity results for the one model for all devices experiment, considering the configuration with the best overall accuracy from Figure 8.6 ($D = 3$ and $t = 10$), evaluated as a function of the time series length.



(a) Raw values **(b)** Proportional values

Figure 8.8. Confusion matrix for the one model for all devices experiment, considering the configuration with the best specificity in Figure 8.7 ($D = 3$, $t = 10$, and $m = 1{,}000$).

proportion of attacks (negatives) correctly identified since it is most important for us to not miss a genuine attack. In this situation, the specificity increases with the length of time series, with significant results occurring for $m \geq 700$. Figure 8.8 illustrates the confusion matrix for the configuration with the best specificity result (99.6 %), occurring for $m = 1{,}000$.

8.4.3.2 One model per device

This section follows another common experiment setup from the literature, which consists of fitting a model for each device. This analysis assesses the ability of the method to individually detect the botnet attacks considering only the data that each device produced. Figure 8.9 presents the accuracy results for the one model per device experiment. The devices are identified by the same numbers as presented in Table 8.1. Each bar represents the average accuracy of all ten botnet attacks considered in the N-BaIoT dataset, previously described in Section 8.4.1, with a confidence interval at 95 % of confidence level.

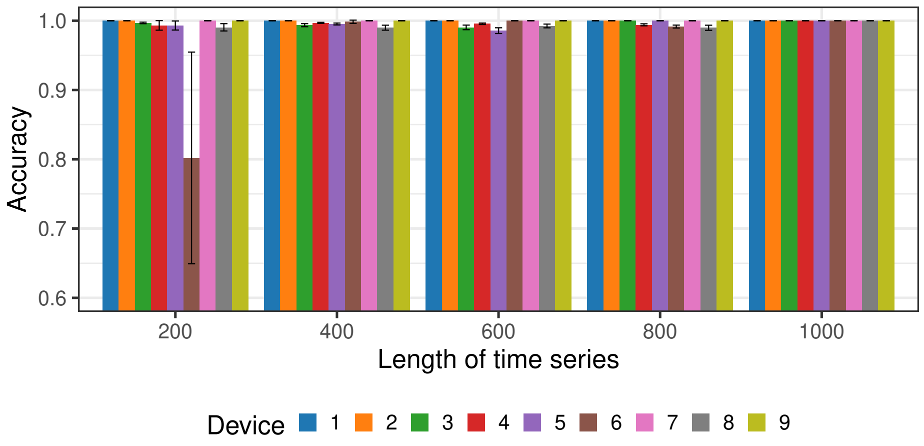


Figure 8.9. Accuracy results for the one model per device experiment, evaluated as a function of the time series length, considering $D = 3$ and $t = 10$. The identification of devices is the numbers presented in Table 8.1.

It can be observed that, in general, the models for specific devices present significant accuracy results. However, the length of the time series is also an important aspect of this experiment. When the length is $m \geq 400$, all devices reach high accuracies, ranging from a minimum of 98.5 % when $m = 600$ for the device (5) Provision PT 737E Security Camera, to a maximum of 100 % for all devices when $m = 1{,}000$.

The worst cases occur when $m = 200$, where few data hinder the models' ability to capture the temporal dynamics of attacks correctly. For instance, for device (6) Provision PT 838 Security Camera, when time series length is $m = 200$, the detection accuracy is 80.19 %, with confidence interval of ± 0.21. To assure this is far beyond other results, the second-worst detection accuracy for $m = 200$ occurs for device (8) SimpleHome XCS7 1002 WHT Security Camera, with an accuracy of 99.05 %, with confidence interval of ± 0.007.

However, when we take a closer look at this specific worst case, Figure 8.10 shows that the method can reach significant accuracies for some botnet attacks. This is the case for the ACK, SCAN, and SYN attacks of Mirai, and the TCP and UDP attacks of Bashlite, with more than 90 % of accuracy for all of them. To infer what is happening here, we should have access to more information from the dataset collection process from Meidan et al. [2018], but only the processed data is available. Given its discrepancy with other results, and since we are dealing with real-world data, it is

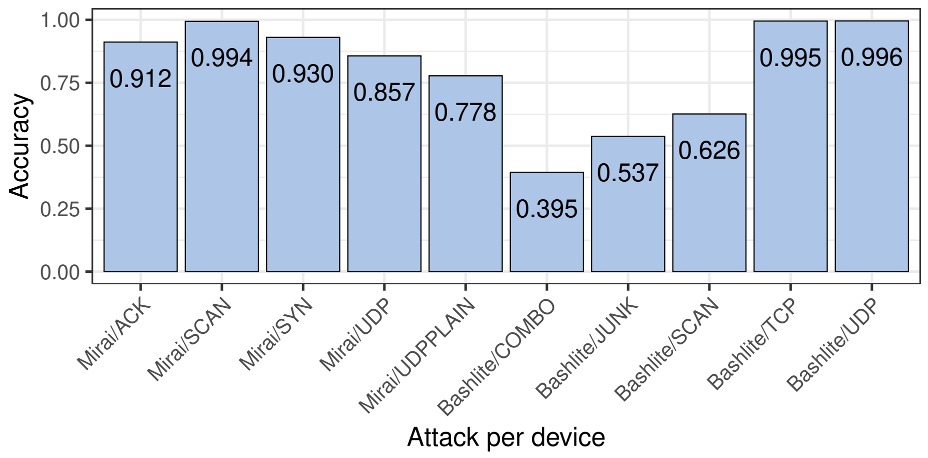


Figure 8.10. Accuracy results per single attack, for the one model per device experiment, considering the worst case of Figure 8.9, when device is the (6) Provision PT 838 Security Camera and time series length $m = 200$.

reasonable to suggest that some spurious data was added during the data collection of this specific device.

This is valid to consider once, as the number of samples increases, the method can correctly distinguish between attack and benign data for all devices. In fact, for the devices (1) Danmini Doorbell, (2) Ecobee Thermostat, (7) Samsung SNH 1011 N Webcam, and (9) SimpleHome XCS7 1003 WHT Security Camera, all detection accuracies are 100 %, independently of the time series length.

With respect to the sensitivity and specificity of the results, considering the case with time series length $m = 1{,}000$, since all devices correctly detected all of their attacks, both sensitivity and specificity are 100 %. These results surpass the related work for similar experiments with a model per device. For instance, Meidan et al. [2018] have a false positive rate of 0.007 (1−specificity), and Nomm and Bahsi [2018] have an average accuracy of 96.64 % for all devices. Alqahtani et al. [2020] also achieve 100 % accuracy when performing an experiment with a model per device; however, the authors used both benign and attack data during the training phase, differently from us. While this improves their method's ability to differentiate benign and attack events correctly, it may impact their ability to detect novel attacks not seen during the training phase.

Table 8.3 summarizes the classification results of our strategy, considering the configuration with $D = 3$ and $t = 10$ with the time series length $n = 1000$, in comparison with the related work in the one model per device experiment.

Table 8.3. Classification results for our strategy, with the configuration of $D = 3$ and $t = 10$, with time series length $n = 1000$, comparing with the related work in the one model per device experiment.

Work	Result	Observation
Our strategy	100% for all devices	When $m = 1000$
Meidan et al. [2018]	FPR 0.007	Using all 115 measures
Nomm and Bahsi [2018]	96.64% for all devices	Not the best
Alqahtani et al. [2020]	100% for all devices	Benign and attack for training

8.4.4 Detection time analysis

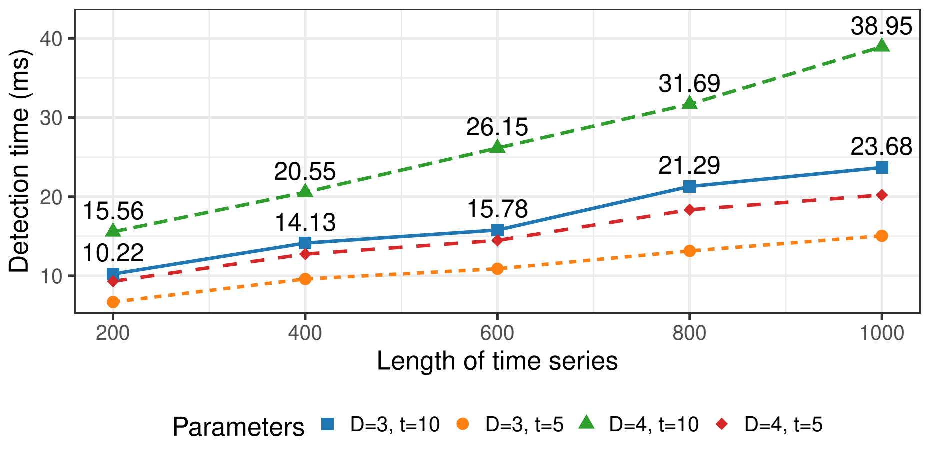


Figure 8.11. Average detection time for the one model for all devices experiment, evaluated as a function of the time series length, combinations of D and maximum τ.

To evaluate the time spent by our method to detect an attack, we analyzed the time spent for the anomaly detection phase by the one model for all devices experiment as a function of the time series length. This is the most time-consuming experiment as it handles more data. Figure 8.11 shows the average detection time over 10 realizations, for different combinations of D and maximum τ. The linear increase on m confirms the theoretical analysis of the computational cost in Section 3.5.1. In the worst case, our method needs under 40 ms to detect an anomaly. However, considering the case which resulted in better accuracies, when $D = 3$, the maximum $\tau = 10$, and the time series length is $m = 1{,}000$, the detection time is under 24 ms.

Another aspect to consider in this analysis is the time spent extracting the measure and constructing the time series. For instance, for $m = 1{,}000$ and for the chosen measure, which needs 100 ms windows to compute the number of packets from a given device, we need another 100 seconds to construct the time series before the transformation step. In fact, the related work does not consider the time spent to extract measures when evaluating their detection time. Thus, since all other methods require network measures with larger time windows, they all need at least 1 minute to compute their measures and then perform their detection.

8.5 Conclusion

In this chapter, we presented a solution for the detection of botnet attacks in IoT by identifying anomalies in the temporal dynamics of their devices. Their dynamics were obtained by features extracted after the multiscale ordinal patterns transformation, from time series constructed of the number of packets transmitted by devices. Using the features computed during the regular operation of the devices, we trained an Isolation Forest model to detect anomalies when presented to features computed from the data collected when the devices are under attack.

To evaluate our proposal, we conducted two main experiments using the N-BaIoT dataset: the first consists of creating a single model for all dataset devices, and the second consists of fitting a specific model per device. While the first experiment is used to evaluate the detection of anomalies regardless of the device, the second analyzes the method's ability to individually detect the botnet attacks considering only the data each device produced.

Our results show that the proposed method can reach significant detection accuracies for both experiments, overcoming other strategies from the literature. For the first experiment, the best results achieved maximum accuracy of 99.5 %, beating the related work, which achieved a maximum 95.6 % of accuracy in their similar experiment. For the second experiment, we reached the maximum accuracy of 100 %. Other related solutions also achieved this accuracy, but they used both regular and attack data to train the model, reducing their ability to detect novel attacks.

We obtained these results when the time series length was $m = 1{,}000$, indicating that its value is fundamental to the transformation process, i.e., this is an important aspect to consider in our method. This is not a drawback since both best results are obtained when $D = 3$, requiring less processing time. As future work, we plan to investigate alternatives to reduce this dependence on large time series. Furthermore, given the urgent need for security solutions in IoT, future solutions should consider transfer learning and online strategies to follow the dynamical evolution of the botnets better, which our proposal can facilitate being another advantage of our method. Finally, another future work includes the application of the proposed method to other datasets, considering other botnets, to assure its viability to different scenarios.

Chapter 9

Conclusion and Future Work

This chapter presents the dissertation final conclusion and discusses directions for future work. Section 9.1 presents our final remarks and a summary of the thesis contributions. In Section 9.2, we present potential future research directions following the contributions of this work. Finally, in Section 9.3, we list the publications produced during the development of this work that are related to the topics of this thesis.

9.1 Summary

In this thesis, we proposed advancements on the analysis of time series dynamics and their applicability for the Internet of Things scenarios. We first provided a deeper comprehension on the main characteristics of IoT data, specifically by analyzing its part that is publicly available, the Collaborative Internet of Things. With this understanding, we presented the main challenges when dealing with these kinds of data. We show that, besides the always mentioned IoT issues, there are novel aspects that must be considered. A main challenge, which is specific to CoIoT devices, is the absence of proper descriptions for the type of sensors and their collected data. This is caused due to the collaborative aspect of this environment, where common users are responsible for both deploying and maintaining their devices.

While this aspect directly affects on the quality of results when searching for sensors within CoIoT, it has also a significant impact on the availability and reliability of devices and their data. Thus, we conclude that to extract knowledge from these data, we must consider transforming the raw time series onto a novel representation domain that is robust to the aforementioned issues. To do so, we propose using the dynamics of the time series, which are obtained by the ordinal patterns transformation. We show that the dynamics is an appropriate representation domain for knowledge discovery applications within the IoT scenarios, such as classification and anomaly detection.

However, to improve the applicability of ordinal patterns transformations to the IoT scenarios, we proposed solutions that consider features extracted from both ordinal patterns distribution and ordinal patterns transition graph, which are transformations derived from the original ordinal patterns. For this, a novel metric was proposed, the probability of self-transitions, which is a metric extracted from the ordinal patterns transition graphs. We show that, when evaluated in a multiscale fashion, the behavior of this metric is a valuable tool for the characterization of time series dynamics. Furthermore, the proposed method presented significant results even when considering small time series, which is an important aspect for the scalability issues of IoT.

With respect to the applicability of time series dynamics for the IoT scenarios, we show that different types of real-world time series data, such as those from weather phenomena, presented different dynamics, which can be used for distinguishing them. To do so, we first provided an analysis of distinguishing these types of data based on their placement on the CCEP. We show that each type of data have their our region in the CCEP, and proposed a noise-robust strategy to find centroids from these regions, which resulted in significant improvement on the classification of these time series.

Following the analysis of the CCEP regions for different types of phenomena data, we proposed a class separability index to find the best parameters for the ordinal patterns transformations. Thus, we show that the best parameters was those which maximizes this index, resulting in better distinctions of time series phenomena. We also show that this method is an important tool for IoT scenarios, due to its ability in adapting to different time series length and robustness to missing data gaps. Also, the proposed class separability analysis can be a valuable method to estimate the classification potential of ordinal patterns transformations. This prior data analysis can be a first step in knowing if its time series classes are distinguishable or not.

Finally, we proposed a solution for the detection of botnet attacks in IoT, an urgent need for the always evolving security concerns within IoT environments. We do so by identifying anomalies in the temporal dynamics of features extracted after the multiscale ordinal patterns transformation, from time series constructed of the number of packets transmitted by devices. We show the potential of this method to learn the regular operation of IoT devices and to detect anomalies when they are under attack. With these results, we advance the state of the art of detecting IoT botnet attacks in a simpler and more efficient way.

9.2 Future research directions

Following the contributions of this thesis, in this section we present potential research directions for each presented strategy. We consider open questions that was not covered and need further investigation.

In the following, we present some possibilities for future work:

- The first research direction is related to the strategy for searching for a sensor in CoIoT, based on the query by content model. We plan to extend the analysis of the proposed collaborative filtering strategy, which may answer the question on the minimum subset of time series required to assure the reliability of the method. In the same direction, it is also important to investigate alternatives to application cases where there is not a reference sensor to be based on. Which leads to the analysis of potential solutions that could consider the combination of different data types to improve the reliability of the presented method. Finally, we also plan to investigate the impact of the limitations and particular aspects of CoIoT devices in the quality of the collected data.

- With respect to the advancements on the analysis of time series dynamics, we suggest further studies on the integration of the proposed method with other already established strategies. This is motivated due to the valuable aspects of the proposed feature, the probability of self-transition, on distinguishing time series dynamics with multiscale transformations. For instance, a promising direction could be their addition in general time series classification strategies. A possibility is to consider solutions based on ensemble of classifiers, where one strategy could solve the drawbacks of another, yielding a more robust and general solution.

- For the contributions related to the applicability of time series dynamics on the IoT scenarios, a first aspect to consider as future work is the study and definition of distance metrics in the CCEP space. Given the importance of this plane to our contributions, and considering its particular limits, there is the need for novel metrics to better calculate the placement of different time series dynamics. Another important aspect to consider, given the dynamical evolution of IoT and, consequently, their problems, is the proposition of ordinal patterns transformations as an online method.

These directions can be useful for the advancement of the proposed strategies on the IoT scenarios, but can also lead to their applicability to other domains. Thus, one can study the viability of the contributions of this thesis to different scenarios, which may result as novel approaches for knowledge discovery in time series, such as characterization, classification, and anomaly detection.

9.3 List of Publications

In the following, we list the publications related to this thesis that were produced as the results of this work.

- Borges Neto, J., Silva, T., Assunção, R., Mini, R., and Loureiro, A. (2015). Sensing in the Collaborative Internet of Things. *Sensors*, 15(3):6607--6632
- Borges, J. B., Ramos, H. S., Mini, R. A., Rosso, O. A., Frery, A. C., and Loureiro, A. A. (2019a). Learning and distinguishing time series dynamics via ordinal patterns transition graphs. *Applied Mathematics and Computation*, 362:124554
- Borges, J. B., Ramos, H. S., Mini, R. A. F., Viana, A. C., and Loureiro, A. A. F. (2019b). The Quest for Sense: Physical phenomena Classification in the Internet of things. In *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 701--708. IEEE
- Borges, J. B., Medeiros, J. P. S., Barbosa, L. P. A., Ramos, H. S., and Loureiro, A. A. F. (2022). IoT Botnet Detection based on Anomalies of Multiscale Time Series Dynamics. *Submitted to IEEE Transactions on Knowledge and Data Engineering*, accepted for publication
- Borges, J. B., Ramos, H. S., and Loureiro, A. A. F. (Submitted, 2021). A classification strategy for Internet of Things data based on the class separability analysis of time series dynamics. *Submitted to ACM Transactions on Internet of Things*, second round

The following publications are the results from the interaction and collaboration on the application of the analysis of data from sensors to other research projects.

- Silva, T. H., Vaz de Melo, P. O. S., Almeida, J. M., Borges, Neto, J. B., Tostes, A. I. J., Celes, C. S. F. S., Mota, V. F. S., Cunha, F. D., Ferreira, A. P. G., Machado, K. L. S., and Loureiro, A. A. F. (2015). Redes de Sensoriamento Participativo: Desafios e Oportunidades. In Martinello, M., Ribeiro, M. R. N., and Rocha, A. A. A., editors, *Minicursos / XXXIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 266--315. Sociedade Brasileira de Computação

- Silva, T., Celes, C., Borges Neto, J. B., Mota, V., da Cunha, F., Ferreira, A., Ribeiro, A., Vaz de Melo, P., Almeida, J., and Loureiro, A. (2016). Users in the urban sensing process. In Dobre, C. and Xhafa, F., editors, *Pervasive Computing: Next Generation Platforms for Intelligent Data Collection*, pages 45--95. Elsevier

- Santos, B. P., Silva, L. A. M., Celes, C. S. F. S., Borges Neto, J. B., Vieira, M. A. M., Vieira, L. F. M., Goussevskaia, O. N., and Loureiro, A. A. F. (2016). Internet das Coisas: da Teoria à Prática. In Gonçalves, F. A. S. L. C. L. F. and Freitas, P. G. A. E. S., editors, *Minicursos / XXXIV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, chapter 1, pages 1--50. Sociedade Brasileira de Computação, Salvador, Brasil

- Celes, C. S. F. S., Nunes, I. O., Borges Neto, J. B., Silva, F. A., Cotta, L., Melo, P. O. S. V., Ramos Filho, H. S., Andrade, R. M. C., and Loureiro, A. A. F. (2017). Big Data Analytics no Projeto de Redes Móveis: Modelos, Protocolos e Aplicações. In Fernandes, A. J. G. A., Cerqueira, E. C., Ramos, H. S., and de Lacerda, S. F., editors, *Minicursos / XXXV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, chapter 1, pages 1--58. Sociedade Brasileira de Computação

- Cardoso, I., Barros, P., Borges, J., Loureiro, A. A. F., and Ramos, H. S. (2019). Classificação de Séries Temporais Através de Grafos de Transição de Padrões Ordinais. In *Anais do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2019)*, pages 622--635. Sociedade Brasileira de Computação - SBC

Bibliography

Abbas, S. G., Zahid, S., Hussain, F., Shah, G. A., and Husnain, M. (2020). A Threat Modelling Approach to Analyze and Mitigate Botnet Attacks in Smart Home Use Case. In *2020 IEEE 14th International Conference on Big Data Science and Engineering (BigDataSE)*, pages 122--129. IEEE.

Aberer, K., Hauswirth, M., and Salehi, A. (2007). Infrastructure for Data Processing in Large-Scale Interconnected Sensor Networks. In *Mobile Data Management, 2007 International Conference on*, pages 198--205, Mannheim, Germany.

Abu-Elkheir, M., Hayajneh, M., and Ali, N. (2013). Data Management for the Internet of Things: Design Primitives and Solution. *Sensors*, 13(11):15582--15612.

Adomavicius, G. and Tuzhilin, A. (2005). Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734--749.

Agarwal, B. and Mittal, N. (2012). Hybrid Approach for Detection of Anomaly Network Traffic using Data Mining Techniques. *Procedia Technology*, 6:996--1003.

Aghabozorgi, S., Seyed Shirkhorshidi, A., and Ying Wah, T. (2015). Time-series clustering – A decade review. *Information Systems*, 53:16--38.

Ahmed, E., Yaqoob, I., Hashem, I. A. T., Khan, I., Ahmed, A. I. A., Imran, M., and Vasilakos, A. V. (2017). The role of big data analytics in Internet of Things. *Computer Networks*, 129:459--471.

Ahmed, M., Liu, L., Hardy, J., Yuan, B., and Antonopoulos, N. (2016). An efficient algorithm for partially matched services in internet of services. *Personal and Ubiquitous Computing*, 20(3):283--293.

Akyildiz, I., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002). Wireless sensor networks: a survey. *Computer Networks*, 38(4):393--422.

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., and Ayyash, M. (2015). Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys and Tutorials*, 17(4):2347--2376.

Al-Garadi, M. A., Mohamed, A., Al-Ali, A. K., Du, X., Ali, I., and Guizani, M. (2020). A Survey of Machine and Deep Learning Methods for Internet of Things (IoT) Security. *IEEE Communications Surveys & Tutorials*, 22(3):1646--1685.

Alcock, R. J. J., Manolopoulos, Y., Laboratory, D. E., Informatics, D. O., and Others (1999). Time-Series Similarity Queries Employing a Feature-Based Approach. *7th Hellenic conference on informatics, Ioannina, Greece*, pages 1--9.

Alqahtani, M., Mathkour, H., and Ben Ismail, M. M. (2020). IoT Botnet Attack Detection Based on Optimized Extreme Gradient Boosting and Feature Selection. *Sensors*, 20(21):6336.

Amigó, J. M., Kocarev, L., and Szczepanski, J. (2006). Order patterns and chaos. *Physics Letters A*, 355(1):27--31.

Apache River Community (2018). Apache River. Technical report, The Apache Software Foundation.

Aquino, A. L. L., Cavalcante, T. S. G., Almeida, E. S., Frery, A. C., and Rosso, O. A. (2015). Characterization of vehicle behavior with information theory. *The European Physical Journal B*, 88(10):257.

Aquino, A. L. L., Ramos, H. S., Frery, A. C., Viana, L. P., Cavalcante, T. S. G., and Rosso, O. A. (2017). Characterization of electric load with Information Theory quantifiers. *Physica A: Statistical Mechanics and its Applications*, 465:277--284.

Atzori, L., Iera, A., and Morabito, G. (2010). The Internet of Things: A survey. *Computer Networks*, 54(15):2787--2805.

Austen, K. (2015). Environmental science: Pollution patrol. *Nature*, 517(7533):136--138.

Bagnall, A., Davis, L., Hills, J., and Lines, J. (2012). Transformation Based Ensembles for Time Series Classification. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, pages 307--318, Philadelphia, PA. Society for Industrial and Applied Mathematics.

Bagnall, A., Lines, J., Bostrom, A., Large, J., and Keogh, E. (2017). The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 31(3):606--660.

Bagnall, A., Lines, J., Hills, J., and Bostrom, A. (2015). Time-Series Classification with COTE: The Collective of Transformation-Based Ensembles. *IEEE Transactions on Knowledge and Data Engineering*, 27(9):2522--2535.

Bandt, C. and Pompe, B. (2002). Permutation Entropy: A Natural Complexity Measure for Time Series. *Physical Review Letters*, 88(17):174102.

Barnaghi, P., Wang, W., Dong, L., and Wang, C. (2013). A Linked-Data Model for Semantic Sensor Streams. In *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, pages 468--475. IEEE.

Baydogan, M. G., Runger, G., and Tuv, E. (2013). A Bag-of-Features Framework to Classify Time Series. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2796--2802.

Behmann, F. and Wu, K. (2015). *Collaborative Internet of Things (C-IOT): for Future Smart Connected Life and Business*. John Wiley & Sons Ltd, Chichester, UK.

Bekiroglu, K., Srinivasan, S., Png, E., Su, R., and Lagoa, C. (2020). Recursive approximation of complex behaviours with IoT-data imperfections. *IEEE/CAA Journal of Automatica Sinica*, 7(3):656--667.

Belkacem, I., Nait-Bahloul, S., and Sauveron, D. (2017). Enhancing dependability through profiling in the collaborative internet of things. *Multimedia Tools and Applications*.

Bertino, E. and Islam, N. (2017). Botnets and Internet of Things Security. *Computer*, 50(2):76--79.

Bettaiah, V. and Ranganath, H. S. (2014). An analysis of time series representation methods. *Proceedings of the 2014 ACM Southeast Regional Conference on - ACM SE '14*, pages 1--6.

Birkhoff, G. (1927). *Dynamical Systems*, volume 9 of *Colloquium Publications*. American Mathematical Society, Providence, Rhode Island.

Blaise, A., Bouet, M., Conan, V., and Secci, S. (2020). Detection of zero-day attacks: An unsupervised port-based approach. *Computer Networks*, 180(April).

Bluetooth SIG (2010). Specification of the Bluetooth System Version 4.0. Specification, Bluetooth SIG.

Boracchi, G. and Roveri, M. (2014). Exploiting self-similarity for change detection. *Proceedings of the International Joint Conference on Neural Networks*, pages 3339--3346.

Borges, J. B., Medeiros, J. P. S., Barbosa, L. P. A., Ramos, H. S., and Loureiro, A. A. F. (2022). IoT Botnet Detection based on Anomalies of Multiscale Time Series Dynamics. *Submitted to IEEE Transactions on Knowledge and Data Engineering.*

Borges, J. B., Ramos, H. S., and Loureiro, A. A. F. (Submitted, 2021). A classification strategy for Internet of Things data based on the class separability analysis of time series dynamics. *Submitted to ACM Transactions on Internet of Things.*

Borges, J. B., Ramos, H. S., Mini, R. A., Rosso, O. A., Frery, A. C., and Loureiro, A. A. (2019a). Learning and distinguishing time series dynamics via ordinal patterns transition graphs. *Applied Mathematics and Computation*, 362:124554.

Borges, J. B., Ramos, H. S., Mini, R. A. F., Viana, A. C., and Loureiro, A. A. F. (2019b). The Quest for Sense: Physical phenomena Classification in the Internet of things. In *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 701--708. IEEE.

Borges Neto, J., Silva, T., Assunção, R., Mini, R., and Loureiro, A. (2015). Sensing in the Collaborative Internet of Things. *Sensors*, 15(3):6607--6632.

Borgia, E. (2014). The Internet of Things vision: Key features, applications and open issues. *Computer Communications*, 54:1--31.

Boukerche, A., Zheng, L., and Alfandi, O. (2020). Outlier Detection: Methods, Models, and Classification. *ACM Computing Surveys*, 53(3):1--37.

Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1):5--32.

Buchholz, T., Schiffers, M., Küpper, A., and Schiffers, M. (2003). Quality of Context: What It Is And Why We Need It. In *Proceedings of the workshop of the HP OpenView University Association*, pages 1--14, Geneve, Switzerland.

Burke, J. A., Estrin, D., Hansen, M., Parker, A., Ramanathan, N., Reddy, S., and Srivastava, M. B. (2006). Participatory sensing. In *First Workshop on World-Sensor-Web: Mobile Device Centric Sensory Networks and Applications*, pages 117--134, Boulder, USA.

Cardoso, I., Barros, P., Borges, J., Loureiro, A. A. F., and Ramos, H. S. (2019). Classificação de Séries Temporais Através de Grafos de Transição de Padrões Ordinais. In *Anais do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2019)*, pages 622--635. Sociedade Brasileira de Computação - SBC.

Celes, C. S. F. S., Nunes, I. O., Borges Neto, J. B., Silva, F. A., Cotta, L., Melo, P. O. S. V., Ramos Filho, H. S., Andrade, R. M. C., and Loureiro, A. A. F. (2017). Big Data Analytics no Projeto de Redes Móveis: Modelos, Protocolos e Aplicações. In Fernandes, A. J. G. A., Cerqueira, E. C., Ramos, H. S., and de Lacerda, S. F., editors, *Minicursos / XXXV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, chapter 1, pages 1--58. Sociedade Brasileira de Computação.

Cerdeira, J. O., Martins, M. J., and Silva, P. C. (2012). A Combinatorial Approach to Assess the Separability of Clusters. *Journal of Classification*, 29(1):7--22.

Cerf, V. G. (2012). It's the net, stupid. *IEEE Internet Computing*, 16(3):96.

Cerf, V. G. and Quaynor, N. (2014). The internet of everyone. *IEEE Internet Computing*, 18(3):96.

Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A Survey. *ACM Computing Surveys*, 41(3):1--58.

Chen, L., Wang, L., Han, Z., Zhao, J., and Wang, W. (2019). Variational inference based kernel dynamic bayesian networks for construction of prediction intervals for industrial time series with incomplete input. *IEEE/CAA Journal of Automatica Sinica*, 7(5):1--9.

Chen, Y., Keogh, E., Hu, B., Begum, N., Bagnall, A., Mueen, A., and Batista, G. (2015). The UCR Time Series Classification Archive.

Chen, Y., Lee, G., Shu, L., and Crespi, N. (2016). Industrial Internet of Things-Based Collaborative Sensing Intelligence: Framework and Research Challenges. *Sensors*, 16(2):215.

Clement, L., Hately, A., von Riegen, C., and Rogers, T. (2004). UDDI Spec Technical Committee Draft 3.0.2. Oasis committee draft, OASIS.

Coen-Porisini, A. and Sicari, S. (2012). Improving data quality using a cross layer protocol in wireless sensor networks. *Computer Networks*, 56(17):3655--3665.

Coronado, M. and Iglesias, C. A. (2016). Task automation services: Automation for the masses. *IEEE Internet Computing*, 20(1):52--58.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273--297.

Dau, H. A., Bagnall, A., Kamgar, K., Yeh, C.-c. M., and Zhu, Y. (2018). UCR time series archive 2018.

Dempster, A., Petitjean, F., and Webb, G. I. (2020). ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery*, 34(5):1454--1495.

Deng, H., Runger, G., Tuv, E., and Vladimir, M. (2013). A time series forest for classification and feature extraction. *Information Sciences*, 239:142--153.

Dey, A. K., Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., Steggles, P., and Abowd, G. D. (1999). Towards a Better Understanding of Context and Context-Awareness. In *Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, HUC '99, pages 304--307, London, UK, UK. Springer-Verlag.

Fulcher, B. D. and Jones, N. S. (2014). Highly Comparative Feature-Based Time-Series Classification. *IEEE Transactions on Knowledge and Data Engineering*, 26(12):3026--3037.

Fulcher, B. D., Little, M. a., and Jones, N. S. (2013). Highly comparative time-series analysis: the empirical structure of time series and their methods. *Journal of The Royal Society Interface*, 10(83):20130048.

Gama, J. (2012). A survey on learning from data streams: current and future trends. *Progress in Artificial Intelligence*, 1(1):45--55.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4):1--37.

Gao, M., Cao, J., and Seto, E. (2015). A distributed network of low-cost continuous reading sensors to measure spatiotemporal variations of PM2.5 in Xi'an, China. *Environmental Pollution*, 199:56--65.

Gao, Z.-K., Small, M., and Kurths, J. (2016). Complex network analysis of time series. *EPL (Europhysics Letters)*, 116(5):50001.

Gao, Z.-K. K. and Jin, N.-D. D. (2012). A directed weighted complex network for characterizing chaotic dynamics from time series. *Nonlinear Analysis: Real World Applications*, 13(2):947--952.

García-Teodoro, P., Díaz-Verdejo, J., Maciá-Fernández, G., and Vázquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers and Security*, 28(1-2):18--28.

Gonçalves, B. A., Carpi, L., Rosso, O. A., and Ravetti, M. G. (2016). Time series characterization via horizontal visibility graph and Information Theory. *Physica A: Statistical Mechanics and its Applications*, 464(September):93--102.

Greene, J. (2001). Feature subset selection using Thornton's separability index and its applicability to a number of sparse proximity-based classifiers. In *Proceedings of the Pattern Recognition Association of South Africa*.

Greengard, S. (2014). Weathering a New Era of Big Data. *Association for Computing Machinery. Communications of the ACM*, 57(9):12.

Gubbi, J., Buyya, R., Marusic, S., and Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645--1660.

Guinard, D., Trifa, V., Wilde, E., and Berkeley, U. C. (2010). A resource oriented architecture for the Web of Things. In *2010 Internet of Things (IOT)*, pages 1--8, Tokyo, Japan. IEEE.

Guttman, E., Perkins, C., Veizades, J., and Day, M. (1999). Service Location Protocol, Version 2. RFC 2608 (Proposed Standard).

Hariri, S., Carrasco Kind, M., and Brunner, R. J. (2019). Extended Isolation Forest. *IEEE Transactions on Knowledge and Data Engineering*, 33(4):1--1.

Hu, B., Chen, Y., and Keogh, E. (2013). Time Series Classification under More Realistic Assumptions. In *Proceedings of the 2013 SIAM International Conference on*

Data Mining, number 1, pages 578--586, Philadelphia, PA. Society for Industrial and Applied Mathematics.

Karkouch, A., Mousannif, H., Al Moatassime, H., and Noel, T. (2016). Data quality in internet of things: A state-of-the-art survey. *Journal of Network and Computer Applications*, 73:57--81.

Keogh, E. and Kasetty, S. (2003). On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration. *Data Mining and Knowledge Discovery*, 7(4):349--371.

Kifer, D., Ben-David, S., and Gehrke, J. (2004). Detecting Change in Data Streams. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, pages 180--191, Toronto, Canada. VLDB Endowment.

Kolias, C., Kambourakis, G., Stavrou, A., and Voas, J. (2017). DDoS in the IoT: Mirai and Other Botnets. *Computer*, 50(7):80--84.

Kulp, C. W., Chobot, J. M., Freitas, H. R., and Sprechini, G. D. (2016). Using ordinal partition transition networks to analyze ECG data. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 26(7):73114.

Kulp, C. W. and Smith, S. (2011). Characterization of noisy symbolic time series. *Physical Review E*, 83(2):26201.

Kulp, C. W. and Zunino, L. (2014). Discriminating chaotic and stochastic dynamics through the permutation spectrum test. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 24(3):33116.

Lacasa, L., Luque, B., Ballesteros, F., Luque, J., and Nuno, J. C. (2008). From time series to complex networks: The visibility graph. *Proceedings of the National Academy of Sciences*, 105(13):4972--4975.

Lamberti, P. W., Martin, M. T., Plastino, A., and Rosso, O. A. (2004). Intensive entropic non-triviality measure. *Physica A: Statistical Mechanics and its Applications*, 334(1-2):119--131.

Larrondo, H., Martín, M. T., González, C. M., Plastino, A., and Rosso, O. (2006). Random number generators and causality. *Physics Letters A*, 352(4-5):421--425.

Leach, P., Mealling, M., and Salz, R. (2005). A Universally Unique IDentifier (UUID) URN Namespace. Technical report 4122, Internet Engineering Task Force.

Li, F., Nastic, S., and Dustdar, S. (2012). Data Quality Observation in Pervasive Environments. In *2012 IEEE 15th International Conference on Computational Science and Engineering*, pages 602--609, Paphos, Cyprus. IEEE.

Lin, J., Keogh, E., Lonardi, S., and Chiu, B. (2003). A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery - DMKD '03*, page 2, New York, New York, USA. ACM Press.

Lin, J., Keogh, E., Lonardi, S., and Patel, P. (2002). Finding motifs in time series. *Proc. of the 2nd Workshop on Temporal Data Mining*, pages 53--68.

Lines, J. and Bagnall, A. (2015). Time series classification with ensembles of elastic distance measures. *Data Mining and Knowledge Discovery*, 29(3):565--592.

Lines, J., Davis, L. M., Hills, J., and Bagnall, A. (2012). A shapelet transform for time series classification. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '12*, page 289, New York, New York, USA. ACM Press.

Lines, J., Taylor, S., and Bagnall, A. (2018). Time Series Classification with HIVE-COTE. *ACM Transactions on Knowledge Discovery from Data*, 12(5):1--35.

Lines, J., Taylor, S., Bagnall, A., and Anglia, E. (2016). HIVE-COTE: The Hierarchical Vote Collective of Transformation-Based Ensembles for Time Series Classification. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, volume 12, pages 1041--1046. IEEE.

Liu, C., Nitschke, P., Williams, S. P., and Zowghi, D. (2020). Data quality and the Internet of Things. *Computing*, 102(2):573--599.

Liu, F. T., Ting, K. M., and Zhou, Z.-H. (2008). Isolation Forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413--422. IEEE.

Liu, F. T., Ting, K. M., and Zhou, Z.-H. (2012). Isolation-Based Anomaly Detection. *ACM Transactions on Knowledge Discovery from Data*, 6(1):1--39.

Löning, M., Bagnall, A., Ganesh, S., Kazakov, V., Lines, J., and Király, F. J. (2019). sktime: A Unified Interface for Machine Learning with Time Series. *Workshop on Systems for ML at NeurIPS 2019*, (NeurIPS).

Loureiro, A. A. F., a.F. Loureiro, A., and Loureiro, A. A. F. (2012). Sensing, tracking and contextualizing entities in ubiquitous computing. In *Proceedings of the 15th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems - MSWiM '12*, page 3, New York, New York, USA. ACM Press.

Luque, B., Lacasa, L., Ballesteros, F., and Luque, J. (2009). Horizontal visibility graphs: Exact results for random time series. *Physical Review E*, 80(4):46103.

Macqueen, J. (1967). Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1(233):281--297.

Makhija, S., Saha, S., Basak, S., and Das, M. (2019). Separating stars from quasars: Machine learning investigation using photometric data. *Astronomy and Computing*, 29:100313.

Mao, K. Z. and Wenyin Tang (2011). Recursive Mahalanobis Separability Measure for Gene Subset Selection. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(1):266--272.

Martin, M. T., Plastino, A., and Rosso, O. A. (2003). Statistical complexity and disequilibrium. *Physics Letters A*, 311(2-3):126--132.

Martin, M. T., Plastino, A., and Rosso, O. A. (2006). Generalized statistical complexity measures: Geometrical and analytical properties. *Physica A: Statistical Mechanics and its Applications*, 369(2):439--462.

Maurus, S. and Plant, C. (2016). Skinny-dip: Clustering in a Sea of Noise. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1055--1064, New York, NY, USA. ACM.

May, R. M. (1976). Simple mathematical models with very complicated dynamics. *Nature*, 261(5560):459--467.

McCullough, M., Small, M., Stemler, T., and Iu, H. H.-C. (2015). Time lagged ordinal partition networks for capturing dynamics of continuous dynamical systems. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 25(5):53101.

Medeiros, J. P. S., Brito, A. M., and Motta Pires, P. S. (2010). An Effective TCP/IP Fingerprinting Technique Based on Strange Attractors Classification. In Garcia-Alfaro, J., Navarro-Arribas, G., Cuppens-Boulahia, N., and Roudier, Y., editors, *Data Privacy Management and Autonomous Spontaneous Security (Lecture Notes in Computer Science)*, volume 5939 LNCS, pages 208--221. Springer Berlin Heidelberg.

Meidan, Y., Bohadana, M., Mathov, Y., Mirsky, Y., Shabtai, A., Breitenbacher, D., and Elovici, Y. (2018). N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders. *IEEE Pervasive Computing*, 17(3):12--22.

Mian, A. N., Baldoni, R., and Beraldi, R. (2009). A Survey of Service Discovery Protocols in Multihop Mobile Ad Hoc Networks. *IEEE Pervasive Computing*, 8(1):66--74.

Miorandi, D., Sicari, S., De Pellegrini, F., and Chlamtac, I. (2012). Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, 10(7):1497--1516.

Mirsky, Y., Doitshman, T., Elovici, Y., and Shabtai, A. (2018). Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection. In *Proceedings 2018 Network and Distributed System Security Symposium*, number May 2019, Reston, VA. Internet Society.

Modi, V. and Kemp, D. (2009). Web Services Dynamic Discovery (WS-Discovery) Version 1.1. Oasis standard 1 july 2009, OASIS.

Mohammadi, M., Al-Fuqaha, A., Sorour, S., and Guizani, M. (2018). Deep Learning for IoT Big Data and Streaming Analytics: A Survey. *IEEE Communications Surveys & Tutorials*, 20(4):2923--2960.

Möller-Levet, C. S., Klawonn, F., Cho, K.-h., and Wolkenhauer, O. (2003). Fuzzy Clustering of Short Time-Series and Unevenly Distributed Sampling Points. In R. Berthold, M., , Lenz, H.-J., , Bradley, E., , Kruse, R., , and Borgelt, C., editors, *Advances in Intelligent Data Analysis V*, pages 330--340. Springer Berlin Heidelberg, Berlin, Heidelberg.

Montero, P. and Vilar, J. A. (2014). TSclust : An R Package for Time Series Clustering. *Journal of Statistical Software*, 62(1):1--43.

Montori, F., Bedogni, L., and Bononi, L. (2018a). A Collaborative Internet of Things Architecture for Smart Cities and Environmental Monitoring. *IEEE Internet of Things Journal*, 5(2):592--605.

Montori, F., Liao, K., Jayaraman, P. P., Bononi, L., Sellis, T., and Georgakopoulos, D. (2018b). Classification and Annotation of Open Internet of Things Datastreams.

In Hacid, H., Cellary, W., Wang, H., Paik, H.-Y., and Zhou, R., editors, *Web Information Systems Engineering – WISE 2018 (Lecture Notes in Computer Science)*, volume 1, pages 209--224. Springer International Publishing.

Nakamura, E. F., Loureiro, A. A. F., and Frery, A. C. (2007). Information fusion for wireless sensor networks. *ACM Computing Surveys*, 39(3):9.

Nath, S., Liu, J., and Zhao, F. (2007). SensorMap for Wide-Area Sensor Webs. *Computer*, 40(7):90--93.

Nomm, S. and Bahsi, H. (2018). Unsupervised Anomaly Based Botnet Detection in IoT Networks. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1048--1053. IEEE.

Olivares, F., Zanin, M., Zunino, L., and Pérez, D. G. (2020). Contrasting chaotic with stochastic dynamics via ordinal transition networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 30(6):063101.

Parlitz, U., Berg, S., Luther, S., Schirdewan, A., Kurths, J., and Wessel, N. (2012). Classifying cardiac biosignals using ordinal pattern statistics and symbolic dynamics. *Computers in Biology and Medicine*, 42(3):319--327.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825--2830.

Perera, C., Zaslavsky, A., Christen, P., Georgakopoulos, D., Member, S., Zaslavsky, A., Christen, P., Georgakopoulos, D., Member, S., Zaslavsky, A., Christen, P., and Georgakopoulos, D. (2014a). Context Aware Computing for The Internet of Things: A Survey. *IEEE Communications Surveys & Tutorials*, 16(1):414--454.

Perera, C., Zaslavsky, A., Liu, C. H., Compton, M., Christen, P., and Georgakopoulos, D. (2014b). Sensor Search Techniques for Sensing as a Service Architecture for the Internet of Things. *IEEE Sensors Journal*, 14(2):406--420.

Perry, M. J. (2016). Evaluating and Choosing an IoT Platform of. Technical report, PTC.

Phuoc, D. L., Quoc, H. N. M., Parreira, J. X., Hauswirth, M., Le-Phuoc, D., and Quoc, H. N. M. (2011). The linked sensor middleware-connecting the real world and the semantic web. Technical report, Semantic Web Challenge 2011, Bonn, Germany.

Popoola, S. I., Ande, R., Adebisi, B., Gui, G., Hammoudeh, M., and Jogunola, O. (2021). Federated Deep Learning for Zero-Day Botnet Attack Detection in IoT Edge Devices. *IEEE Internet of Things Journal*, XX(X):1--1.

Postol, M., Diaz, C., Simon, R., and Wicke, D. (2019). Time-Series Data Analysis for Classification of Noisy and Incomplete Internet-of-Things Datasets. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 1543--1550. IEEE.

Qin, Y., Sheng, Q. Z., and Curry, E. (2015). Matching Over Linked Data Streams in the Internet of Things. *IEEE Internet Computing*, 19(3):21--27.

Qin, Y., Sheng, Q. Z., Falkner, N. J. G., Dustdar, S., Wang, H., and Vasilakos, A. V. (2016). When things matter: A survey on data-centric internet of things. *Journal of Network and Computer Applications*, 64:137--153.

R Core Team (2018). A Language and Environment for Statistical Computing.

Rambold, M., Kasinger, H., Lautenbacher, F., and Bauer, B. (2009). Towards Autonomic Service Discovery A Survey and Comparison. In *2009 IEEE International Conference on Services Computing*, number Section II, pages 192--201. IEEE.

Ratanamahatana, C. A. and Keogh, E. (2005). Three Myths about Dynamic Time Warping Data Mining. In *Proceedings of the 2005 SIAM International Conference on Data Mining*, pages 506--510, Philadelphia, PA. Society for Industrial and Applied Mathematics.

Ravetti, M. G., Carpi, L. C., Gonçalves, B. A., Frery, A. C., and Rosso, O. A. (2014). Distinguishing Noise from Chaos: Objective versus Subjective Criteria Using Horizontal Visibility Graph. *PLoS ONE*, 9(9):e108004.

Rawassizadeh, R. and Kotz, D. (2017). Datasets for Mobile, Wearable and IOT Research. *GetMobile: Mobile Computing and Communications*, 20(4):5--7.

Ribeiro, H. V., Jauregui, M., Zunino, L., and Lenzi, E. K. (2017). Characterizing time series via complexity-entropy curves. *Physical Review E*, 95(6):062106.

Rong, W. and Liu, K. (2010). A Survey of Context Aware Web Service Discovery: From User's Perspective. In *2010 Fifth IEEE International Symposium on Service Oriented System Engineering*, pages 15--22. Ieee.

Rosso, O. A., Carpi, L. C., Saco, P. M., Gómez Ravetti, M., Plastino, A., and Larrondo, H. A. (2012). Causality and the entropy–complexity plane: Robustness and missing ordinal patterns. *Physica A: Statistical Mechanics and its Applications*, 391(1-2):42--55.

Rosso, O. A., Larrondo, H. A., Martin, M. T., Plastino, A., and Fuentes, M. A. (2007a). Distinguishing Noise from Chaos. *Physical Review Letters*, 99(15):154102.

Rosso, O. A., Olivares, F., and Plastino, A. (2015). Noise versus chaos in a causal Fisher-Shannon plane. *Papers in Physics*, 7(November 2014):070006.

Rosso, O. A., Olivares, F., Zunino, L., De Micco, L., Aquino, A. L. L., Plastino, A., and Larrondo, H. A. (2013). Characterization of chaotic maps using the permutation Bandt-Pompe probability distribution. *The European Physical Journal B*, 86(4):116.

Rosso, O. A., Ospina, R., and Frery, A. C. (2016). Classification and Verification of Handwritten Signatures with Time Causal Information Theory Quantifiers. *PLOS ONE*, 11(12):e0166868.

Rosso, O. A., Zunino, L., Pérez, D. G., Figliola, A., Larrondo, H. A., Garavaglia, M., Martín, M. T., and Plastino, A. (2007b). Extracting features of Gaussian self-similar stochastic processes via the Bandt-Pompe approach. *Physical Review E*, 76(6):061114.

Sanchez-Moreno, P., Yanez, R. J., and Dehesa, J. (2009). Discrete Densities and Fisher Information. In *Difference Equations and Applications*, number January, pages 291--298.

Santos, B. P., Silva, L. A. M., Celes, C. S. F. S., Borges Neto, J. B., Vieira, M. A. M., Vieira, L. F. M., Goussevskaia, O. N., and Loureiro, A. A. F. (2016). Internet das Coisas: da Teoria à Prática. In Gonçalves, F. A. S. L. C. L. F. and Freitas, P. G. A. E. S., editors, *Minicursos / XXXIV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, chapter 1, pages 1--50. Sociedade Brasileira de Computação, Salvador, Brasil.

Schafer, J. L. and Graham, J. W. (2002). Missing data: Our view of the state of the art. *Psychological Methods*, 7(2):147--177.

Schäfer, P. (2015). The BOSS is concerned with time series classification in the presence of noise. *Data Mining and Knowledge Discovery*, 29(6):1505--1530.

Schmeißer, S. and Schiele, G. (2020). coSense: The Collaborative Sensing Middleware for the Internet-of-Things. *ACM/IMS Transactions on Data Science*, 1(4):1--21.

Sezer, O. B., Dogdu, E., and Ozbayoglu, A. M. (2018). Context-Aware Computing, Learning, and Big Data in Internet of Things: A Survey. *IEEE Internet of Things Journal*, 5(1):1--27.

Sicari, S., Cappiello, C., De Pellegrini, F., Miorandi, D., and Coen-Porisini, A. (2014). A security-and quality-aware system architecture for Internet of Things. *Information Systems Frontiers*, 18(4):665--677.

Sicari, S., Rizzardi, A., Grieco, L., and Coen-Porisini, A. (2015). Security, privacy and trust in Internet of Things: The road ahead. *Computer Networks*, 76(15):146--164.

Silva, T., Celes, C., Borges Neto, J. B., Mota, V., da Cunha, F., Ferreira, A., Ribeiro, A., Vaz de Melo, P., Almeida, J., and Loureiro, A. (2016). Users in the urban sensing process. In Dobre, C. and Xhafa, F., editors, *Pervasive Computing: Next Generation Platforms for Intelligent Data Collection*, pages 45--95. Elsevier.

Silva, T., S. Vaz De Melo, P., Almeida, J., and F. Loureiro, A. (2014). Large-scale study of city dynamics and urban social behavior using participatory sensing. *IEEE Wireless Communications*, 21(1):42--51.

Silva, T. H., Vaz de Melo, P. O. S., Almeida, J. M., Borges, Neto, J. B., Tostes, A. I. J., Celes, C. S. F. S., Mota, V. F. S., Cunha, F. D., Ferreira, A. P. G., Machado, K. L. S., and Loureiro, A. A. F. (2015). Redes de Sensoriamento Participativo: Desafios e Oportunidades. In Martinello, M., Ribeiro, M. R. N., and Rocha, A. A. A., editors, *Minicursos / XXXIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 266--315. Sociedade Brasileira de Computação.

Sorrentino, T., Quintero-Quiroz, C., Aragoneses, A., Torrent, M. C., and Masoller, C. (2015). Effects of periodic forcing on the temporally correlated spikes of a semiconductor laser with feedback. *Optics Express*, 23(5):5571.

Stankovic, J. A. (2014). Research Directions for the Internet of Things. *Internet of Things Journal, IEEE*, 1(1):3--9.

Strogatz, S. H. (2001). Exploring complex networks. *Nature*, 410(6825):268--276.

Strogatz, S. H. (2018). *Nonlinear Dynamics and Chaos*. CRC Press.

Tang, J., Wang, Y., and Liu, F. (2013). Characterizing traffic time series based on complex network theory. *Physica A: Statistical Mechanics and its Applications*, 392(18):4192--4201.

Thornton, C. (1998). Separability is a Learner's Best Friend. In Bullinaria, J. A., Glasspool, D. W., and Houghton, G., editors, *4th Neural Computation and Psychology Workshop, London, 9–11 April 1997*, pages 40--46, London. Springer.

Toker, D., Sommer, F. T., and D'Esposito, M. (2020). A simple method for detecting chaos in nature. *Communications Biology*, 3(1):11.

Tsai, C.-W., Lai, C.-F., Chiang, M.-C., and Yang, L. T. (2014). Data Mining for Internet of Things: A Survey. *IEEE Communications Surveys & Tutorials*, 16(1):77--97.

Unger, G. and Chor, B. (2010). Linear separability of gene expression data sets. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 7(2):375--381.

UPnP Forum (2011). Universal Plug and Play Device Architecture Reference Specification Version 1.1. Technical report, UPnP Forum.

Vasseur, J.-P. P. and Dunkels, A. (2010). *Interconnecting Smart Objects with IP*. Morgan Kaufmann Publishers Inc.

Ververidis, C. and Polyzos, G. (2008). Service discovery for mobile Ad Hoc networks: a survey of issues and techniques. *IEEE Communications Surveys & Tutorials*, 10(3):30--45.

Wang, A., Chang, W., Chen, S., and Mohaisen, A. (2018). Delving Into Internet DDoS Attacks by Botnets: Characterization and Analysis. *IEEE/ACM Transactions on Networking*, 26(6):2843--2855.

Wang, J., Shang, P., Shi, W., and Cui, X. (2016). Dissimilarity measure based on ordinal pattern for physiological signals. *Communications in Nonlinear Science and Numerical Simulation*, 37:115--124.

Wang, L. and Lei Wang (2008). Feature Selection with Kernel Class Separability. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(9):1534--1546.

Wang, X., Mueen, A., Ding, H., Trajcevski, G., Scheuermann, P., and Keogh, E. (2013). Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, 26(2):275--309.

Wang, X., Smith, K., and Hyndman, R. (2006). Characteristic-Based Clustering for Time Series Data. *Data Mining and Knowledge Discovery*, 13(3):335--364.

Wei, Q. and Jin, Z. (2012). Service discovery for internet of things: a context-awareness perspective. In *Proceedings of the Fourth Asia-Pacific Symposium on Internetware - Internetware '12*, pages 1--6, Qingdao, China. ACM Press.

Weiser, M. (1991). The computer for the 21st century. *Scientific American*, 265(3):94--104.

Wilson, S. J. (2017). Data representation for time series data mining: time domain approaches. *Wiley Interdisciplinary Reviews: Computational Statistics*, 9(1):1--6.

Wolf, A., Swift, J. B., Swinney, H. L., and Vastano, J. A. (1985). Determining Lyapunov exponents from a time series. *Physica D: Nonlinear Phenomena*, 16(3):285--317.

Wu, M., Wang, Y., and Liao, Z. (2014). A new clustering algorithm for sensor data streams in an agricultural IoT. *Proceedings - 2013 IEEE International Conference on High Performance Computing and Communications, HPCC 2013 and 2013 IEEE International Conference on Embedded and Ubiquitous Computing, EUC 2013*, pages 2373--2378.

Wu, Y. (2021). Robust Learning-Enabled Intelligence for the Internet of Things: A Survey From the Perspectives of Noisy Data and Adversarial Examples. *IEEE Internet of Things Journal*, 8(12):9568--9579.

Yankov, D., Keogh, E., Medina, J., Chiu, B., and Zordan, V. (2007). Detecting time series motifs under uniform scaling. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '07*, page 844, New York, New York, USA. ACM Press.

Ye, B., Chen, J., Ju, C., Li, H., and Wang, X. (2017). Distinguishing chaotic time series from noise: A random matrix approach. *Communications in Nonlinear Science and Numerical Simulation*, 44(September):284--291.

Yu, J., Kang, H., Park, D., Bang, H.-C., and Kang, D. W. (2013). An in-depth analysis on traffic flooding attacks detection and system using data mining techniques. *Journal of Systems Architecture*, 59(10):1005--1012.

Zanella, A., Bui, N., Castellani, A., Vangelista, L., and Zorzi, M. (2014). Internet of Things for Smart Cities. *IEEE Internet of Things Journal*, 1(1):22--32.

Zeng, D., Guo, S., and Cheng, Z. (2011). The Web of Things: A Survey (Invited Paper). *Journal of Communications*, 6(6):424--438.

Zhang, J. and Small, M. (2006). Complex Network from Pseudoperiodic Time Series: Topology versus Dynamics. *Physical Review Letters*, 96(23):238701.

Zhang, J., Sun, J., Luo, X., Zhang, K., Nakamura, T., and Small, M. (2008). Characterizing pseudoperiodic time series through the complex network approach. *Physica D: Nonlinear Phenomena*, 237(22):2856--2865.

Zhang, J., Zhou, J., Tang, M., Guo, H., Small, M., and Zou, Y. (2017). Constructing ordinal partition transition networks from multivariate time series. *Scientific Reports*, 7(1):7795.

Zhang, R., Zettsu, K., Kidawara, Y., and Kiyoki, Y. (2010). Exploiting Service Context for Web Service Search Engine. In Chen, L., Tang, C., Yang, J., and Gao, Y., editors, *Web-Age Information Management*, volume 6184 of *Lecture Notes in Computer Science*, pages 363--375. Springer Berlin Heidelberg.

Zheng, H. and Zhang, Y. (2008). Feature selection for high-dimensional data in astronomy. *Advances in Space Research*, 41(12):1960--1964.

Zunino, L., Soriano, M. C., Fischer, I., Rosso, O. A., and Mirasso, C. R. (2010). Permutation-information-theory approach to unveil delay dynamics from time-series analysis. *Physical Review E*, 82(4):46212.

Zunino, L., Soriano, M. C., and Rosso, O. A. (2012). Distinguishing chaotic and stochastic dynamics from time series by using a multiscale symbolic approach. *Physical Review E*, 86(4):46210.

Acronyms

AIMD	Additive Increase, Multiplicative Decrease
ASOS	Automated Surface Observing Systems
BOSS	Bag-of-SFA-Symbols
C&C	Command and Control
CCEP	Causality Complexity-Entropy Plane
CIDS	Collaborative Intrusion Detection System
CoIoT	Collaborative Internet of Things
DDoS	Distributed Denial-of-Service
DIY	Do-It-Yourself
DTW	Dynamic Time Warping
EE	Ensemble of Elastic distances
Flat-COTE	Flat COllective of Transformation-based Ensembles
GSI	Geometric Separability Index
GSN	Global Sensor Networks
HIVE-COTE	Hierarchical Vote COllective of Transformation-based Ensembles
HVG	Horizontal Visibility Graph

ICAO	International Civil Aviation Organization
IEM	Iowa Environmental Mesonet
IIoT	Industrial Internet of Things
IoT	Internet of Things
KDE	Kernel Density Estimation
KNN	K-Nearest Neighbors
LSM	Linked Sensor Middleware
MSE	Mean Squared Error
NIDS	Network Intrusion Detection System
OP	Ordinal Patterns
OPPD	Ordinal Patterns Probability Distribution
OPTG	Ordinal Patterns Transition Graph
PAA	Piecewise Aggregate Approximation
QoC	Quality of Context
QoD	Quality of Device
QoS	Quality of Service
RANDF	Random Forest
RFID	Radio-Frequency IDentification
RISE	Random Interval Spectral Forest
SAX	Symbolic Aggregate approXimation
SLP	Service Location Protocol
SNMP	Simple Network Management Protocol
SVM	Support Vector Machine

TSCLAS	Time Series Classification via Class Separability
TSF	Time Series Forest
UPnP	Universal Plug and Play
UUID	Universal Description, Discovery and Integration
VG	Visibility Graph
XGBoost	eXtreme Gradient Boosting