CLASSIFICAÇÃO DE MODOS DE TRANSPORTE UTILIZANDO PADRÕES ORDINAIS COM INFORMAÇÃO DE AMPLITUDE

ISADORA CARDOSO PEREIRA DA SILVA

CLASSIFICAÇÃO DE MODOS DE TRANSPORTE UTILIZANDO PADRÕES ORDINAIS COM INFORMAÇÃO DE AMPLITUDE

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Heitor Soares Ramos Filho
Coorientador: Antonio Alfredo Ferreira Loureiro

Belo Horizonte

Janeiro de 2020

ISADORA CARDOSO PEREIRA DA SILVA

TRANSPORTATION MODE CLASSIFICATION THROUGH ORDINAL PATTERNS WITH AMPLITUDE INFORMATION

Dissertation 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 Master in Computer Science.

Advisor: Heitor Soares Ramos Filho
Co-Advisor: Antonio Alfredo Ferreira Loureiro

Belo Horizonte
January 2020

Ficha catalográfica elaborada pela bibliotecária Belkiz Inez Rezende Costa CRB 6ª Região nº 1510

Silva, Isadora Cardoso Pereira da.

S587t Transportation mode classification through Ordinal Patterns with amplitude information / Isadora Cardoso Pereira da Silva. – Belo Horizonte, 2020.
xxiv, 77 f.: il.; 29cm.

Dissertação (mestrado) – Universidade Federal de Minas Gerais – Departamento de Ciência da Computação.

Orientador: Heitor Soares Ramos Filho.
Coorientador: Antonio Alfredo Ferreira Loureiro.

1. Computação – Teses. 2. Mobilidade urbana – Teses. 3. Classificação de modos de transporte – Teses. 4. Análise de séries temporais – Teses. I. Orientador. II. Coorientador. III. Título.

CDU 519.6*65(043)

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

FOLHA DE APROVAÇÃO

Transportation Mode Classification through Ordinal Patterns with Amplitude Information

ISADORA CARDOSO PEREIRA DA SILVA

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

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

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

PROF. ALEJANDRO CÉSAR FRERY ORGAMBIDE
Instituto de Computação - UFAL

PROF. JEFERSSON ALEX DOS SANTOS
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 13 de Janeiro de 2020.

Acknowledgments

First of all, I would like to thank God for blessing me with the gift of life, today and every day. His grace in my life are countless.

I would like to thank my family, especially my parents, Roseneide and Enaldo, and my sisters, Isabella and Christiane, for all the efforts and dedication to give me the opportunity of studying; and the encouragement and love throughout this journey as well. For the same reason I am also grateful for my second family, Sarah, my girlfriend, and my friends Eddie, Kerol, Jazz, Aylla, Celsi, and many many others. You guys are the best people to have around.

I would like to express my deep gratitude to my advisors, Antonio Loureiro and Heitor Ramos. Besides the extreme competence as professors, they are especial to me for different reasons: Loureiro, an incredible smart person, with his experience in research and life, helped me to become a better researcher, and, also, a better person. And Heitor, that has patiently monitored and directed my work since undergrad, allowing the development of my skills (in research and life). His experience in research, and mainly his friendship, helped me to become a researcher with abilities to navigate in the landscape of science in a manner that I love and appreciate. Thank you so much.

My sincere thanks to my professors from LaCCAN – UFAL, Alla, Eliana, André Lages, and Alejandro, for showing me how to be the best scientist and person as possible. Also, I would like to thanks DCC – UFMG (an amazing place with excellent professors, staff, and students), for providing academic structures that enabled the development of this work. Also, I would like to acknowledge the financial support from the National Council for Scientific and Technological Development (CNPq).

"Miracles are not meant to be wished for; they are meant to be created by our own power."

(Norihiro Yagi, *Claymore*)

Resumo

A infraestrutura das cidades está passando por um estresse significativo, visto que a demanda pelos recursos básicos (como transporte, educação, saúde, etc.) está superando o fornecimento. Isso se dá devido ao crescimento desordenado, causado pela migração e aumento da população. Dessa forma, as comunidades científicas e industriais estão cada vez mais interessadas na elaboração de tecnologias baseadas na mobilidade humana que possam proporcionar um desenvolvimento mais sustentável e que sejam capazes de reduzir diversos problemas de locomoção, como congestionamentos, afim de aumentar a qualidade de vida dos cidadãos. Um passo crucial para atingir esses objetivos é a caracterização dos modos de transportes utilizados. É necessário desenvolver tecnologias que possam extrair esses dados sem a ativa participação do usuário, evitando-se dados incompletos e imprecisos.

Nesse contexto, o objetivo dessa dissertação é o desenvolvimento de um framework que, a partir de dados de localização do usuário, possa identificar os modos de transportes utilizados. Esse framework possui quatro etapas: (i) segmentação, (ii) extração de atributos, (iii) transformação de dados e (iv) classificação. Maior atenção é dada à terceira etapa, onde propõe-se uma transformação de dados baseada na distribuição de probabilidade dos Padrões Ordinais (PO), capaz de extrair a informação de amplitude presente nos dados – chamada de Padrões Ordinais com Informação de Amplitude (POIA). Em nossos experimentos, realizados em dados reais, mostra-se que POIA apresenta resultados de classificação superiores em relação a PO, um ganho de cerca de 10% de acurácia, indicando que POIA é uma técnica com potencial para a identificação de modos de transporte.

Palavras-chave: Classificação de Modos de Transporte, Classificação de Séries Temporais, Padrões Ordinais.

Abstract

The infrastructure of cities is experiencing significant stress, since the demand for basic resources (such as transport, education, healthcare, etc) is outstripping supply. This is happening due to the disorderly growth caused by migration and increasing of the world's population. Therefore, the scientific and industrial communities are investing in solutions based on human mobility that can provide a more sustainable development and reduce several commuting problems, such as traffic jam, in order to improve the life quality of humans. A critical step to achieve such goals is to characterize the transportation mode used. It is paramount the development of technologies that extract this kind of information, without the active participation of users on the act, hence avoiding inaccurate and incomplete data.

In this context, this dissertation aim to develop a framework that, from user location data, can identify the transportation modes used. This framework contains four step: (i) segmentation; (ii) feature extraction; (iii) data transformation; and (iv) classification. More attention is given to the third step, where we propose a data transformation based on Ordinal Patterns (OP) probability distribution, capable of extracting the amplitude information presented in data – called Ordinal Pattern with Amplitude Information (OPAI). In our experiments, performed in real data, we show that OPAI presents superior classification results compared to OP transformation, a gain of about 10% of accuracy, indicating that OPAI is a technique with potential for the identification of transportation mode.

Keywords: Transportation Mode Classification, Time Series Classification, Ordinal Patterns.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

The current chapter introduces this dissertation work, which is entitled "Transportation Mode Classification through Ordinal Patterns with Amplitude Information". We discuss the initial considerations and motivation of this work in Section 1.1. After, in Section 1.2, we show the goals we intend to achieve. Section 1.3 expounds the contributions made. Finally, Section 1.4 describes the organization which this work follows.

1.1 Motivation

In our world, most of the data generated has a temporal component, whether it is a natural process (e.g. weather, sound waves, and planet movements) or human-made (e.g. robotics and citizen mobility). Also, in human growth, time is always present when our brain is learning tasks like language, vision, and motion (Längkvist et al., 2014). To acquire a solid knowledge and understanding about the dynamics of life, we can examine phenomena through time, forming a discrete collection of observations, called time series.

In this context, trajectory data have an unique value. Since 2018, we have 55% of the world's population residing in urban areas, and it is estimated to reach 68% by 2050. It is caused by the migration from rural to urban areas and the increasing of the world's population. Such growth, generally in a disorderly way, is putting significant stress on city infrastructure, since the demand for basic services (i.e., transportation, education, healthcare, and safety) is outstripping the supply (United Nations and Social Affairs, 2018). Therefore, in order to solve the challenge of a sustainable development, which must meet the needs of their increasing population, cities are investing in solutions based on the study of human mobility. They want to understand and characterize how

humans commute, where and why they go, to develop essential technologies capable of reducing traffic jam, travel time, and also more sustainable transport solutions, which can improve the life quality of humans. A critical step to achieve such goals is to characterize the transportation mode used.

In the past, the data used in transportation mode characterization were obtained by surveys answered by volunteers, which often resulted in underreporting of short trips and in inaccurate and incomplete data (Biljecki et al., 2013). Recent advancements in positioning technologies, such as Global Positioning System (GPS) and ubiquitous sensors, have made possible to acquire location data in an inexpensive and straightforward way. For instance, we can see a remarkable growth of mobile phones users (expected to pass the five billion mark this year [1]), which have several sensors (including GPS) capable of obtaining information about user's location. Unfortunately, transportation mode information still relies on users manually labeling their data, which can lead to errors similar to those found by surveys. It is of paramount importance the development of technologies that can extract this kind of information, without the active participation of users on the act.

Nevertheless, time series data has been the subject of study for decades (Längkvist et al., 2014), attracting increasing attention from scientific and industrial communities. Mining time series data, however, faces several complexities, being considered by Yang and Wu (2006) one of the most challenging problems in data mining research due to its unique properties, such as heterogeneity and presence of noise. Additionally, we must face high dimensionality in time series data, a well-known challenge nowadays, in the big data era, where we have an unprecedented amount of data, being generated in high speed, for all application domains. Nonetheless, the most striking property of time series data is its spatio-temporal dependence, i.e., there is a relationship of data elements. In other words, a change in data points order could change their meaning. It opposes the common assumption made by many algorithms, such as Naïve Bayes, of independent and identically distributed observations, leading standard classification methods to perform poorly in time series (Bagnall et al., 2017).

1.2 Objectives

The main objective of this dissertation is to answer the following question:

> **Research problem:** Is it possible to extract transportation mode information from user's location?

[1] https://www.statista.com/statistics/274774/forecast-of-mobile-phone-users-worldwide/

To that end, a fundamental step is to understand the properties of transportation modes, i.e., how transportation modes can be characterized based on their uniqueness. We aim to analyze it derived from GPS data. After that, our goal is to model and extract the transportation mode knowledge from GPS data using Ordinal Patterns distribution. Thus, we tackle the main objectives of this dissertation answering three different questions:

- **What are the properties of transportation modes?** Although transportation mode classification is a studied field, its characterization is an important step to urban planning and understanding of humans mobility, for instance. With that, our goal is to investigate the properties of transportation modes, based on features extracted from GPS data, in order to understand its challenges and usefulness;

- **How can we extract transportation mode information from GPS data?** Our goal here is to use the features we extract from the analysis process to the design of a framework to distinguish between the transportation modes that can be used by a citizen to commute. First, we want a model that enables the knowledge extraction from trajectory data. This model must try to mitigate the issues of time series data and, hence, improve classification. Based on this model, the aim is to propose a methodology that, efficiently and effectively, mines transportation mode information from trajectory data.

- **Is Ordinal Patterns distribution capable of providing good characterization of transportation mode?** In the design of our framework, our hypothesis is that transforming trajectory data to Ordinal Patterns distribution, based on the characteristics of this method (such as computationally inexpensive transformation), can highlight the underlying features presented in the original data, positively contributing to the transportation mode classification field.

1.3 Contributions

Our main contributions can be summarized in:

- **Ordinal Patterns with Amplitude Information (OPAI).** We proposed a novel Ordinal Patterns (OP) representation with amplitude information, providing better classification results.

- **Characterization and analysis of transportation mode information based on OPAI.** We characterized and analyzed the properties of transportation modes, using the OPAI technique. With this, we developed a deeper understanding about the properties of transportation mode.
- **Definition, modeling, and application of Transportation Mode Classification Framework.** We proposed a novel transportation mode classification framework, using OPAI method and features extracted from GPS data.

1.4 Outline

The remainder of this dissertation is organized as follows. Chapter 2 provides an introductory overview about time series and transportation mode classification. Chapter 3 introduces and defines fundamental concepts about time series, trajectory data, and classification, used throughout this work. Chapter 4 discusses the OP distribution and our contribution to this transformation, OPAI. Chapter 5 presents the framework proposed to transportation mode classification, comparing our results to OP distribution and showing that our proposal overcome it in classification results. Chapter 6 summarizes the contributions of this dissertation and, moreover, we present some research directions for future works.

Chapter 2

Related Work

This work presents the design process of a transportation mode classifier from trajectory data. As this data is a type of time series data, our framework can be understood as a time series classifier. In this chapter, we present Time Series and Transportation Mode Classification, in Section 2.1 and Section 2.2, respectively. Lastly, Section 2.3 presents our final remarks, finalizing the chapter.

2.1 Time Series Classification

Among the possibilities in time series data mining, Time Series Classification (TSC) is an extensively studied problem – researchers have proposed hundreds of methods to solve this task (Bagnall et al., 2017). This section is divided in four subsections. First, in Section 2.1.1, we discuss the works that use traditional Machine Learning approaches to contribute in this field; Section 2.1.2 presents the Deep Learning approaches, Information Theory approaches are shown in Section 2.1.3, Section 2.1.4 reviews the Complex Network approaches, and, finally, in Section 2.1.5 we present a discussion about the works presented.

2.1.1 Traditional Machine Learning Approaches

The characterization and classification of time series is the subject of study of several areas and, as such, is widely explored. There are several contributions in the field of Machine Learning (ML). However, uniformly contradicting results in empirical evaluations demonstrate the need of comprehensive comparison and reproducible research. Knowing this, Wang et al. (2010) compared the effectiveness of 8 different representation methods, as well as 9 similarity measures and their variants, in 38 real-world data

sets from diverse application domains. They used a simple k-Nearest Neighbor (k-NN) (with k = 1) classifier combined with the evaluated methods. Their results showed that it is more difficult to achieve good classification results using representation methods as data dimensionality grows. Also, the evaluated similarity measures presented similar accuracy results.

Bagnall et al. (2017) contributed to extending the reproducible studies in time series classification evaluating the accuracy of 18+ time series classifiers. They separated such classifiers into different five groups, namely, shapelets, interval-based, dictionary-based, whole series and combinations. These categories are based on the type of discriminatory features the technique is attempting to find. They showed that the best results belong to the ensemble classifier Collective of Transformation Ensembles (COTE) (Bagnall et al., 2015). COTE combines 35 classifiers of the 18 evaluated algorithms (the change of parameters in the algorithms leads to different classifiers). All algorithms were evaluated in the UCR Time Series Archive (Chen et al., 2015), a repository composed of 85 univariate real-world data sets of many time series types, such as motion, simulated, ECG, and device. They concluded that COTE is on average 8% more accurate than others. However, COTE is hugely computationally intensive. Even more, COTE has an imbalance amount of methods from each domain, so, in case of similar training accuracy, the collective will be biased.

Lines et al. (2018) improved the COTE proposing a new hierarchical structure with probabilistic voting. The authors called such approach of Hierarchical Vote Collective of Transformation-based Ensembles (HIVE-COTE). HIVE-COTE overcomes the potential design bias by modularizing the elements of each group of classifiers. It allows only a single probabilistic prediction from each algorithm group (whole series; interval; shapelet; dictionary; and spectral). They compared the top-ranked algorithms in the study mentioned above with HIVE-COTE and two Convolutional Neural Networks (CNN) approaches. COTE presented a better result than the evaluated CNNs, but HIVE-COTE surpassed all of them. Still, HIVE-COTE presents the same complexity of COTE, being computationally intensive as well. For instance, training HIVE-COTE on a data set with only 1,500 time series can require 8 days of CPU time (Shifaz et al., 2019).

Shifaz et al. (2019) proposed Time Series Combination of Heterogeneous and Integrated Embeddings Forest (TS-CHIEF). It uses Proximity Forests (Lucas et al., 2019) to separate the time series into an ensemble of classification trees. They choose splitting functions based on similarity measures, dictionary representations and interval-based transformations, inspired by previous works, such as Lines et al. (2018). However, they do not use shapelets transformation because of their high computational com-

plexity. This approach achieves a better average rank than HIVE-COTE, but it does not present a significantly better improvement in accuracy. However, it is a scalable solution, and the authors claim that it is up to 46000 faster than HIVE-COTE in some cases.

These ensemble approaches are the state-of-the-art in ML field for TSC. Their results are based on different data representation, such as dictionary-based. It is a very common approach, based on the hypothesis that patterns that differentiate groups of time series occurs repeatedly. Therefore, dictionary-based approaches forms a frequency count of repeated patterns. This transformation is useful to reduce dimensionality, being a computationally inexpensive transformation. Bag of SFA Symbols (BOSS), proposed by Schäfer (2015), is used in HIVE-COTE and TS-CHIEF as their dictionary-based transformation, due to its high accuracy in the study executed by Bagnall et al. (2017). It surpasses other methods of this kind of transformation, such as BoP (Lin et al., 2012) and SAXVSM (Senin and Malinchik, 2013). The latter, however, presented results that are not significantly worse than BOSS. More works are being developed using this type of transformation, such as MiSTiCI (Raza and Kramer, 2019). Although it achieves results that are statistically not discernible from BOSS, MiSTiCI is a faster approach.

2.1.2 Deep Learning Approaches

Opposing the "shallow" algorithms described before, Fawaz et al. (2019a) investigated nine Deep Learning (DL) approaches in time series classification, which were evaluated in the same 85 data sets of Bagnall et al. (2017). ResNet (He et al., 2016), although not tailored to this task, won on 50 problems out of 85, beating some approaches built especially to deal with time series. They compared this deep architecture with the best classifiers in Bagnall et al. (2017), including COTE. Although COTE presented the best accuracy, the authors argue that ResNet has some advantages: in addition to training time, COTE needs a linear scan to perform classification, whereas ResNet classifies instantly; also, ResNet's hyperparameters does not need to be tuned for each data set but rather the same architecture was used for the whole benchmark. They suggested that further investigation of these hyperparameters should improve DL accuracy.

Still in DL methods, Fawaz et al. (2019b) proposed an ensemble of the current state-of-the-art DL models for time series classification, called Neural Network Ensemble (NNE). Their model is composed of 60 different DL networks: six different architectures, with ten different initial weight values for each. NNE reaches state-of-the-art results, having no significant difference from HIVE-COTE classification. These

results were concluded from the same database used in the previous mentioned works.

2.1.3 Information Theory Approaches

Techniques derived from Information Theory have also been successful in the characterization of time series. Such methods can distinguish time series using model-free techniques that also are computationally inexpensive and have low dimensionality. Bandt and Pompe (2002), in their seminal paper, proposed a time series transformation, named Ordinal Pattern (OP), which can be calculated for any type of time series, without model assumptions. They also proposed a complexity measure called Permutation Entropy (PE), that is extracted after the OP transformation. It was shown that PE is an appropriate complexity measure for chaotic time series, in particular in the presence of dynamical and observational noise.

Furthermore, several methods are derived from OP transformation, such as Entropy-Complexity Plane, proposed by Rosso et al. (2007). It is defined as the two-dimensional diagram obtained by plotting permutation statistical complexity (vertical axis) versus the PE (horizontal axis) for a given system. They showed that this method is particularly efficient at distinguishing between the deterministic chaotic and stochastic nature of time series, since the quantifiers have distinctive behaviors for different types of dynamics.

Many works use these aforementioned approaches in several domains, showing its importance in time series characterization. For instance, Aquino et al. (2015) characterized the behavior of vehicles through their velocities; Aquino et al. (2017) characterized the behavior of electric loads; Ribeiro et al. (2017) characterized the behavior of the crude oil price; and Zanin et al. (2012) reviewed the application of PE in biomedical and econophysics domains.

However, the original definition of PE, presented in Bandt and Pompe (2002), has the drawback of not capturing data amplitude. Many works proposed an improvement to accommodate this information, as we can see in Fadlallah et al. (2013) and Azami and Escudero (2016).

2.1.4 Complex Network Approaches

Another research direction that has also been successful in the characterization of time series is based on the transformation of the time series into graphs. Gao et al. (2017) reviewed several approaches and its applications in real-world data analysis, such as medical time series. Some examples are the visibility graph, by Lacasa et al. (2008),

and the horizontal visibility graph, by Luque et al. (2009). Using these strategies it is possible to construct networks that inherit the characteristics of the original time series – for instance, periodic series are transformed into regular graphs, and random series are transformed into random graphs. However, as each time series sample is transformed into a vertex of the graph, there is an impact on the scalability of these techniques, making them unfeasible for very long time series.

Recently, methods that combine more than one approach are emerging. Small (2013) proposed to obtain graphs from permutations of possible patterns in OP – the Ordinal Network. Such network represents the relation between consecutive patterns. It inherits some properties from OP transformation, such as simplicity, speed, robustness, and scalability, preserving the order in which the patterns occur. However, this network also inherit the OP drawbacks, such as the inability to detect data amplitude. Also, as with the OP transformation, this method does not apply to multivariate time series. Generalizations to multidimensional data were proposed by many works, such as in Zhang et al. (2017) and Guo et al. (2018).

2.1.5 Discussion

Time series is an ubiquitous data type, hence, of great importance to industry and research communities. In fact, as said before, Bagnall et al. (2017) relates that there are hundreds of methods to time series classification, in several computing fields, as we could see through this section.

ML algorithms have been the most used approach to solve this task and, recently, there was a growth of DL algorithms focused on this same goal. Furthermore, most of the works in this areas can be directly compared, since they use the same benchmark database to evaluate their effectiveness. Although these approaches are the state-of-the-art in TSC, ML methods rely deeply in time series transformations, such as dictionary-based. Based on it, a way to leverage this field is developing solutions using data transformation.

Moreover, Esling and Agon (2012) discuss that the No-Free-Lunch (NFL) theorem prevents the existence of the best algorithm, i.e., an algorithm that presents the best results in all domains, despite the effort to find it. According to such theorem, the good performance of an algorithm in any domain is "paid" by a poorer performance on another domain. However, Giraud-Carrier and Provost (2005) argue that the NFL theorem is of little relevance to research in ML, specifically in meta-learning. This statement is supported by the works presented in Section 2.1.1, in which ensemble classifiers are the state-of-the-art. A drawback of this kind of solution is the high

computational cost, since it has to use many transformations to visit each possible domain without prior knowledge about the data. As presented in Section 2.1.2, DL methods, with their high-level representation, overcome this issue.

Section 2.1.3 and Section 2.1.4 show that Information Theory (IT) and Complex Networks (CN) approaches are capable of characterizing time series in several domains. However, these fields are not as studied as ML and DL to time series classification. Even more, although possessing promising results, we cannot compare the results of IT and CN to ML and DL approaches with such easiness since they are used in different data. It is important to provide studies that deeper examine these areas and their power in time series classification.

2.2 Transportation Mode Classification

Transportation mode information, that is, the information about the different ways of transportation made by humans, can be acquired from several sources, not restricted to just one. For instance, Ermes et al. (2008) and Parkka et al. (2006) use GPS and wearable sensors data (e.g. body temperature and heart rate) to detect activities such as walking, rowing, and cycling. This kind of approach, however, obliges the user to carry several sensors in order to have their transportation mode information detected. Moreover, some works show that it is possible to combine sensor data with external information. As examples, we can cite the work of Shah et al. (2014), in which the authors classify different forms of motorized transport such as car, bus, and subway, using GPS data, transit route information published by transit agencies, and motion detection collected from phone's accelerometer. The disadvantage of this kind of approach is the need of models capable of collecting the external information every other time, since city information can change over time.

Accordingly, many works are focused on classifying using information from just one source. For instance, Huang et al. (2019) presented a systematic review about transportation mode detection using mobile phone data, in which they discussed the data, preprocessing steps, and transport mode identified in more than 20 works.

Therefore, in this dissertation, we concentrate our efforts in classification of GPS data. There are several works that use only GPS data as their source, as presented by Yang et al. (2018). We will objectively evaluate the works that can be compared to ours, i.e., those that use the same dataset, described in Section 3.3.3. Moreover, differently from the aforementioned work, that describes the transportation mode classification based on their sources (single or mixed sources), we will review based on the

computational approaches.

In Table 2.1, we summarize the related work presented in the field of Transportation Mode Classification using GPS data. The works are divided by the set of transportation mode used. To know which features are used in these works, please look for the study in this section. We present only the techniques that achieved the best result. Additionally, between parenthesis in the accuracy column, we see the relative accuracy (the gain, with ↑, or loss, with ↓) compared to the first work that used the same set of transportation mode. Some works use an unique set, hence, they cannot be compared to others. Moreover, the works marked with * employ a different data subset, making them unable to compare with the others. Also, works marked with † also cannot be adequately compared to others since they use a noise removal approach based on ground truth information, infringing the prediction procedure. Such technique may improve accuracy unrealistically as well.

The remainder of section is divided as follows. In Section 2.2.1, we discuss the traditional Machine Learning approaches. Section 2.2.2 details the Deep Learning approaches. Information Theory approaches are presented in Section 2.2.3. Lastly, Section 2.2.4 discusses the works presented.

2.2.1 Traditional Machine Learning Approaches

Zheng et al. (2008b) proposed a framework for transportation mode classification composed of four sequential steps: segmentation, feature extraction, inference and post-processing. First, they segmented the trajectories by identifying its change point, i.e., the point in which the users change their transportation mode. After that, they extracted, from each segment, the following features: length, mean velocity, expectation of velocity, variance of velocity, top three velocities, and top three accelerations. To classify the segments, the authors used four different classification algorithms, namely, Decision Tree (DT), Support Vector Machine (SVM), Bayesian Net (BN) and Conditional Random Field (CRF). Sequentially, instead of choosing the maximum probability as final result in the inference process, they consider the conditional probability between different transportation modes to re-calculate the probability of each segment. Therefore, they achieved an overall accuracy of 74.3% using DT, when classifying walking, bus, bike, and driving (car and taxi) as the possible transportation modes.

Extending their previous work, Zheng et al. (2008a) changed some steps in their proposed framework. The segmentation step still relies on change point detection. However, for feature extraction, the authors extracted, besides the aforementioned features, three new ones: heading change rate (HCR), stop rate (SR), and velocity

change rate (VCR). HCR stands for the change of direction of the transportation mode used – for instance, intuitively, the change of direction is more accentuate while walking or cycling than driving or taking a bus, since the latter transportation modes are constrained by a road. For SR and VCR, they noted that some transportation modes are likely to stop and diminish their velocity than others, for instance, while walking, an user stops or do it slowly more than while taking a bus or driving. The inference step only applies DT method to classify the transportation modes. And, finally, in the post-processing step, their algorithm takes the preliminary inference result and a spatial knowledge (a graph of change points constructed from the change point information extracted from the segments) to enhance the classification results. For the same transportation modes of the previous work, they achieved 76.2% of overall accuracy.

Xiao et al. (2017) proposed a method that includes data preprocessing, feature extraction, model classification and model evaluation. In data preprocessing, they removed duplicate entries in GPS data and outlier trajectories based on "common sense" (e.g., average speed of walking exceeding 10 m/s or biking exceeding 25 m/s). For feature extraction, they extracted two groups, called global and local features. The former refers to descriptive statistics for the entire trajectory, which makes trajectories more comparable, and the latter, extracted by profile decomposition, reveals more detail in movement behavior. The authors first extracted speed, acceleration, turn angle (the direction of the two consecutive points) and sinuosity (its winding path divided by the distance). With these information, for global features, they acquired, from each segmentation, the mean, standard deviation, mode, top three value, minimum three value, value range (the maximum value minus the minimum value), percentile, interquartile range, skewness, kurtosis, coefficient of variation, autocorrelation coefficient, trajectory length, HCR, VCR, and SR. For local features, a series of features were extracted from the profile decomposition algorithm. This included the mean and standard statistics of segment length per decomposition class and per parameter, the count of changes of decomposition classes, and the proportion per decomposition class account for the total number of points. Therefore, 111 features were used in this work. For the model classification, they used and compared the performances of traditional ML methods (k-NN, SVM, and DT) against tree-based ensemble models, namely Random Forest (RF), Gradient Boosting Decision Tree (GBDT), and Extreme Gradient Boosting (XGBoost). For walk, bus and taxi, bike, car, subway, and train, their best performance were achieved by XGBoost, with 90.77% of overall accuracy. With this, they concluded that tree-based ensemble models performed better than the traditional methods.

In Etemad et al. (2018), the authors proposed a framework of five steps: data segmentation, point features generation, feature extraction, noise removal, and normalization. The point features used in step two are distance, speed, acceleration, jerk, and bearing. Additionally, they calculated two new features, named bearing rate and the rate of bearing rate, which are important to quantify the change of the bearing in the movement of the transportation mode. After that, they extracted global and local features of each trajectory point. For global features, they used minimum, maximum, mean, median, and standard deviation values; and, for local features, they obtained five different percentiles (10, 25, 50, 75, and 90) of every point feature. In total, they computed 70 trajectory features (10 statistical measures including five global and five local features calculated for 7 point features) for each transportation mode example. To remove the noise, they used a method called median filter, that creates a mask based on the average speed of a trajectory, removing the abnormal values. After that, they normalized the features, using the Min-Max normalization. They classified different transportation modes set, with five classifiers, namely DT, RF, Neural Network (NN), Naïve Bayes (NB), Quadratic Discriminant Analysis (QDA). In all the cases, RF presented the best result. The classified sets are: (i) bike, car, walk, and bus (where they obtained 96.45% of overall accuracy); (ii) walk, bike, bus, driving (car and taxi), and train (93.55%); (iii) walk, bus and taxi, bike, car, subway, train (93.19%); (iv) walk, car, taxi, bike, subway, bus, train (90.20%); and (v) walk, bus, bike, driving (car and taxi) (93.61%). Therefore, they concluded that noise removal in transportation mode classification is an important step, improving the performance of the classifiers.

2.2.2 Deep Learning Approaches

To avoid the necessity of hand-crafted features, many researchers applied DL methods, that are capable of extracting many levels of representation without human interference. Dabiri and Heaslip (2018) explored the use of CNN architectures to predict transportation mode categorized in walk, bike, bus, driving (car and taxi), and train (in this only case, it involves subway as well). The authors comprised the kinematic features speed, acceleration, jerk, and bearing rate as matrices, to create a standard arrangement for the CNN scheme, which is, generally, an image. This data were manually preprocessed to remove abnormal values. They combined seven CNN models in an ensemble configuration, achieving 84.8% of overall accuracy. The authors highlight that removing anomalies, designing an efficient input layer with the appropriate motion characteristics, augmenting training data, tuning hyperparameters, and employing the bagging concept are the key factors to achieve such high accuracy.

Jiang et al. (2017) investigated the classification of four transportation modes (i.e., bike, car, walk, and bus) using Recurrent Neural Network (RNN). Instead of learning the RNN parameters directly in the features extracted from data, they map such features into a discretized space, based on the hypothesis that better features are learned in a smaller shared space (an approach similar to kernel trick (Hofmann, 2006), but without explicit designing of kernel functions). The features extracted were the point-based speed, average speed per segment, and standard deviation of speed per segment. Moreover, they used Hampel filter to remove ouliers from the features. They achieved an overall accuracy of 97.9%. Also, they applied the same technique to classify seven transportation modes, namely train, car, bus, subway, airplane, and bike, achieving 97.3% of accuracy. However, these results are applied in a subset, making it impossible to compare with the other works.

Endo et al. (2016) proposed a method to automatically extract features from raw trajectory data using Deep Neural Network (DNN). Specifically, the trajectory is represented as a 2D image data structure (called trajectory image) and higher-level features are extracted using fully-connected DNN with Stacked Denoising Autoencoder (SDA). They compared the proposed method with the hand-craft features extracted in Zheng et al. (2008a) and Zheng et al. (2008b) to classify walking, bus, car, bike, taxi, subway, and train. The authors concluded that their method present better accuracy results than such hand-craft features, but the combination of all the features obtained the best accuracy value, of 67.90%.

2.2.3 Information Theory Approaches

Information Theory (IT) methods are applied in Transportation Mode Classification as well. Zhang et al. (2015) extracted PE as a feature, along with average speed, speed variance, HCR, SR, and VCR to characterize walking, bike, bus, driving, train, and airplane. They classified using Extreme Learning Machine (ELM) for Feedforward Neural Network with single hidden layer and obtained an accuracy of 83.79%. However, similar to Jiang et al. (2017), they use a data subset, making it unfeasible to compare this work with others.

In Cardoso et al. (2019), we proposed a method to classify walk, bike, bus, and driving using IT features extracted from distance, latitude, and longitude. These features are PE, Statistical Complexity, and self-transition probability, proposed by (Borges et al., 2019). We classified using k-NN (k = 2), SVM (with linear and radial kernels), and DT. With SVM using radial kernel, the best result was obtained, 73.54% of overall accuracy.

2.2.4 Discussion

Transportation Mode Classification (TMC) using only GPS data as its source is a well studied field, as we could see in this Section. However, differently from Time Series Classification, this field lacks of an influential benchmark that can make it easier to compare the works. Knowing this, we discussed only studies made in the same dataset in order to contrast them.

An important aspect about the studies above is the presence of feature engineering. It can help the learning algorithm, but it requires domain expertise to develop the features. Some works used hand crafted features, such as Etemad et al. (2018); Xiao et al. (2017); Zheng et al. (2008a); while others used representation learning methods, as in Dabiri and Heaslip (2018) – Endo et al. (2016) used the two approaches. The main difference about both approaches is that hand crafted features generate interpretative models, since we known the features; representation learning, however, may not be understood by humans, due their high-level features.

Another essential point in these works is the data mining. Specifically, it involves segmentation and data cleaning. In the former, most works used the transportation mode information to segment data, which relies on ground truth information, being an acceptable step in literature when focusing on classification, due the need to create the observations dataset to perform such task. Zheng et al. (2008b), for instance, used walking information to slice the trajectories. In all the works reviewed here, the number of segments extracted from trajectory data are different. It is not possible to say which one is the correct total of segments. In other words, we cannot affirm if one segment is indeed about one trip or not, since the dataset is manually labelled, which may lead to errors such as underreporting. Though, since the dataset contains the ground truth information (which transportation mode a point belongs), at least it is guaranteed that the segments contains information about only one transportation mode.

Diverse strategies are used in data cleaning to remove the noise presented in trajectory data. Zheng et al. (2008a,b); Endo et al. (2016); Zhang et al. (2015); Cardoso et al. (2019), however, did not mention anything about this step. In Xiao et al. (2017); Dabiri and Heaslip (2018), this removing is based on "common sense", as said in Xiao et al. (2017). It means that they use the information about the transportation mode to classify which is an abnormal value – i.e., speed of walking exceeding 10 m/s. This approach goes against the inferring procedure, since we have to know a priori the transport information in the test data. Additionally, Etemad (2018) showed that this kind of noise removal can improve accuracy unrealistically – in their experiments,

the mean accuracy increased from 88.5% to 91.8% using this kind of method. Hence, we can assume that works that use this approach cannot be adequately compared to others. Other noise removal methods, such as filters, are also used, as shown in Jiang et al. (2017), and some works such as Dabiri and Heaslip (2018) used both approaches.

Furthermore, the evaluation methods, mostly accuracy of models, are heterogeneous as well. They include random cross validation, cross validation over users and no cross validation (simply a division of training and test set). The random cross validation is the method most used in ML evaluation. The over users variation, used by Endo et al. (2016), separated into training segments of 80% of users and test segments of 20% of users. They argue that the learned features are dependent on users due their habits or environments, so the random cross validation might provide optimistic results.

Also in the evaluation stage, some works, such as Zhang et al. (2015) and Jiang et al. (2017), provide their results in a data subset: they randomly choose 30 and 23 of 73 users, respectively, to apply their methods. It causes a "cherry picking" effect, preventing such studies to be adequately compared to others as well.

Finally, as seen in Table 2.1, even using the same dataset, the literature does not classify an unique set of transportation mode. For instance, the set used by Zheng et al. (2008b) is applied in three more works, whereas Zhang et al. (2017) diverge by classifying a different set, used only by themselves. It is just one more issue that prevents the comparison of the works in this field. Therefore, we can conclude that it is imperative to develop solutions that can unify the methodology of TMC to provide more comparable works, which can help the progress of such area.

2.3 Final Remarks

This chapter presented a summary of publications related to time series and transportation mode classification.

Although this work also presents a framework to transportation mode classification, similar to others cited in this chapter, this work differs from them by proposing a novel OP transformation, that considers the amplitude information. Moreover, this is a general approach that can be used to classify time series data from several domains, as will be demonstrated in this work.

Table 2.1: Summary of past research in Transportation Mode Classification using GPS data

Mode	Work	Method	Features	Accuracy (%)
walk, bus, bike, driving (car and taxi)	Zheng et al. (2008b)	DT	10	74.30
	Zheng et al. (2008a)	DT	13	76.20 (1.90 ↑)
	Etemad et al. (2018)	RF	70	93.61 (19.31 ↑)
	Cardoso et al. (2019)	SVM	4	73.54 (-0.76 ↓)
walk, bus, car, bike, taxi, subway, train	Endo et al. (2016)	DNN	23 + DNN	67.90
	Etemad et al. (2018)	RF	70	90.20 (22.3 ↑)
walk, bike, bus, train and subway, driving	Dabiri and Heaslip (2018)	ensemble of CNN	4	84.80 †
	Etemad et al. (2018)	RF	70	93.55 (8.75 ↑)
walk, bus and taxi, bike, car, subway, train	Xiao et al. (2017)	XGBoost	111	90.77 †
	Etemad et al. (2018)	RF	70	93.19 (2.42 ↑)
bike, car, walk, bus	Jiang et al. (2017)	RNN	3	97.90 *
	Etemad et al. (2018)	RF	70	96.45 (-1.45 ↓)
walk, bike, bus, driving, train, airplane	Zhang et al. (2015)	ELM	6	83.79 *
train, car, bus, subway, airplane, bike	Jiang et al. (2017)	RNN	3	97.30 *

Chapter 3

Preliminaries, Fundamentals, and Definitions

This chapter presents some preliminary concepts and definitions that are used throughout this dissertation. It is organized as follows. Section 3.1 discusses time series data. Section 3.2 explains time series data mining. Section 3.3 describes trajectory data. And we conclude this chapter in Section 3.4.

3.1 Time series

As technology increases, we can see the growing power of storages and processors. With this, it is possible to collect and keep data of real-world events for a long time. Hence, time series, being one of the most ubiquitous data type, have provided the opportunity of understanding several phenomena in the last decades (Aghabozorgi et al., 2015).

In this section, we will depict some definitions regarding time series that are fundamental to this dissertation. It is organized as follows. Section 3.1.1 defines time series.

3.1.1 Definition

The concepts of time series, as defined below, is inspired by Bagnall et al. (2017) and Esling and Agon (2012).

Definition. An *univariate time series* $\mathbf{X}(t) = \{x_1, x_2, \ldots, x_n\}$ is a sequence of n data points, record over time, where each element x_i represents a point in time, $i \in \mathbb{N}$.

Figure 3.1: The process of Knowledge Discovery in Databases (KDD)

Definition. A *multivariate time series* $\vec{\mathbf{X}} = \{X_1, X_2, \ldots, X_n\}$ is a sequence of n univariate time series, where each element X_i represents an univariate time series, $i \in \mathbb{N}$.

Throughout this work, "time series" refers to the univariate time series. When referring to multivariate time series, it will be fully indicated.

3.2 Time Series Data Mining

Data Mining is part of Knowledge Discovery in Databases (KDD), which is the process of converting raw data into practical information, as shown in Figure 3.1. It can be defined as follows.

Definition. *Data Mining* is the process of automatically discovering useful information in data. This discovering is made by applying techniques and extracting novel and useful patterns that might otherwise remain unknown.

The input data may have a variety of formats – in this work, we use time series data; more specifically, trajectory data.

The purpose of *preprocessing* step is to transform raw data into more appropriate format for subsequent analysis. It encompasses several methods, such as feature selection, data transformation, noise removal, dimensionality reduction, normalization, and many others. In this Section we will detail the methods used in this work, namely segmentation (Section 3.2.1) and data transformation (Section 3.2.2). This step is the most time-consuming in the overall KDD process.

Postprocessing consists of interpretation and evaluation of the obtained results. Generally, this step is made by a specialist. It includes, but not limited to, model evaluation, visualization, and pattern interpretation (Tan et al., 2005). Section 3.2.4 defines the evaluation methods used in this work.

Finally, performing the aforementioned step, information is acquired. In this work, we obtain knowledge about transportation mode classification, i.e., in which transportation mode the citizen is moving, based on their trajectory data.

Moreover, in Section 3.2.3 we will describe classification, a Data Mining method.

3.2.1 Segmentation

We define segmentation as follows.

Definition. *Segmentation* is the process of dividing a time series into segments, i.e., subsets of data, in order to reveal the underlying properties of its source.

This definition is based on the work of Lovrić et al. (2014).

Segmentation can be understood as a partition of a set, hence, time series can be redefined as follows.

Definition. A *time series* $\mathbf{X}(t)$ of size n is a set of non-empty segments S of $\mathbf{X}(t)$ such that every element x in $\mathbf{X}(t)$ is in exactly one of these segments.

In other words, a time series is a set of segments in which the union of the segments is equal to the time series itself and the intersection of two different segments is empty. We can define segment as follows.

Definition. A *segment* $\mathbf{S}(t) = \{s_1, s_2, \ldots, s_k\}$ is a consecutive sequence of points extracted from a time series $\mathbf{X}(t)$ with size n, where $k \leq n$.

Segmentation can assist the pattern discovery, and consequently, time series classification, by allowing the division of time series into groups. It helps the assimilation of the unique properties presented by different groups. In this dissertation, we segment the raw trajectory data into segments containing the transportation modes information.

3.2.2 Data transformation

Time series are usually high dimensional data. Therefore, work directly with such data, in their raw version, can be computationally expensive (Esling and Agon, 2012). In order to avoid this burden, we can apply data transformation.

Definition. *Data transformation*, or data representation, consists of applying a transformation directly to the time series into the same time domain, such as summarizing original data points into more comprehensible format, or changing the data from the time domain to another domain, e.g., frequency, shapelets, symbol-based.

The concept of data transformation presented above is inspired by Wilson (2017) and Bagnall et al. (2017).

The main motivation of data transformation is to emphasize the essential characteristics of the data in a concise way (Esling and Agon, 2012). According to Aghabozorgi et al. (2015), an appropriate data representation method can be the key component that affects the efficiency and accuracy of the solution. Hence, a proper transformation should not only reduce the dimensionality and remove noise, but it must preserve the critical local and global features of the original data as well (Wilson, 2017). Also, such transformation should be robust to data problems, such as missing data, outliers, and irregular time spacing, for instance.

As interest in mining of massive sized data continues to rise, since we are living in the "Big Data" era, data representation methods by transforming numeric time series into a finite number of discrete variables or symbols, has become more popular. Symbolic transformation meets the needs of data reduction, enabling an efficient computation and usage of memory space for data storage (Wilson, 2017).

Therefore, data transformation is an essential step in time series data mining and, hence, it is applied in this dissertation. We transform trajectory data into Ordinal Patterns distribution, a symbolic transformation, described in Chapter 4.

3.2.3 Classification

To achieve the goals intended in Data Mining, we can use Machine Learning (ML) techniques, defined as follows.

Definition. *Machine Learning* is an Artificial Intelligence field that studies algorithms capable of learning how to accomplish activities in general, making the machine able to behave in a non-programmed way.

This learning is made through of data observations, explicit instructions, or experiences. In general, the learning can be divided in supervised and unsupervised, being classification part of the supervised learning (James et al., 2013).

Definition. In the *Supervised learning*, the goal is to find a model that relates observations with predefined groups (known a priori), in order to characterize the current observations and predict the class of future observations.

In this context, we can define classification as

Definition. A *classification* algorithm is a model or a function M that predicts the class label $\hat{y}$ for a given input observation x, that is, $\hat{y} = M(x)$, where $\hat{y} \in \{c_1, \ldots, c_k\}$, with k labels, and each c_i is a class label.

To build such model, it is required a set of time series data with their actual class labels, which is called training set. In this set, the model will learn the relationship between data and groups. After learning M, it is possible to automatically predict the class of any new time series (Zaki et al., 2014). Usually, to evaluate the predictive power of the model M, we predict a set of time series data and compared it to their actual labels; this dataset is called test set and it must be data that are not present in the training set.

Most classic ML algorithms, including these developed for classification, do not work well on time series data due to their unique structure – often, time series have very high dimensionality, high feature correlation, and large amount of noise. Therefore, classification algorithms avoid operating on the original data; instead, they consider some higher-level representation or abstraction of time series (Ratanamahatana et al., 2010). In this dissertation, we use a data transformation called Ordinal Patterns before applying the classification algorithm.

A trend to improve performance in classification tasks are the use of Ensemble models, which can be defined as follows.

Definition. *Ensemble* models are the combination of different classifiers together. Such combination associates its classification decision, exploring their individual advantages, in an attempt to improve the classification results.

The use of Ensemble models can be seen, for instance, in several winner methods in classification competitions, such as the Netflix Prize[1] (Koren, 2009; Töscher et al., 2009; Piotte and Chabbert, 2009) and Kaggle[2]. In addition, Bagnall et al. (2015); Lines et al. (2018); Shifaz et al. (2019) and many others solutions show the power of such combination in time series classification, with Dabiri and Heaslip (2018) showing its importance in transportation mode classification.

The hypothesis behind the application of Ensemble models is that different algorithms are capable of capturing different patterns – even the best algorithm can fail to detect some classes, while others algorithms may detect such patterns correctly, although presenting an inferior overall performance. Hence, the prediction power of distinct classifiers may be complementary, leading to better performance compared to using a single classifier.

We compare two Ensemble classifiers – Gradient Boosting Decision Tree (GBDT) and Random Forest (RF). They were chosen based on the success of tree-based models

[1] `https://www.netflixprize.com/`
[2] `https://www.kaggle.com/`

over traditional methods to classify transportation mode, as shown by Xiao et al. (2017).

RF and GBDT classifiers combine Decision Tree (DT) predictors to produce their model. Hence, to understand them, it is important to learn about DT first.

DT is one of the simplest classification algorithms in data mining. It is a recursive, partition-based tree model that predicts the class for each data observation. To this, a DT uses an axis-parallel hyperplane to split the data space into two resulting regions and, consequently, the data are also divided into two parts. Recursively, each of these regions is split via axis-parallel hyperplanes until the points within a partition are relatively pure in terms of their class labels, i.e., most of the points belong to the same class. The resulting hierarchy of split decisions is the DT model, with the leaf nodes labeled with the majority class among points in those regions. A new data example is classified by recursively evaluating in which region it belongs until we reach a leaf node, so the class of this new observation is predicted as the class of the leaf (Zaki et al., 2014).

Figure 3.2 shows the tree representation of a DT model and its correspondent space split in axis-parallel hyperplanes. This model aims at classifying between different transportation modes: rail-based, walking, bike, road-based modes. We can see that first the model uses the speed information to split between "fast" (road-based and rail-based) and "slow" (bike and walking). After, the "fast" speed region is split into road-based and rail-based modes based on the distance they travel per points, if it is less than 30 meters, it is a road-based transportation mode, otherwise, it is a rail-based mode. Simultaneously, the "slow" speed region is divided into bike, if they travel more than 2 meters per points, and walking, if not. As it is a toy example, each leaf node is pure, i.e., contains only true positive examples. In real life, it rarely occurs, since real data are more complex.

To calculate the purity of a node, many measures can be used, such as entropy and classification error. In this work, we use the Gini Impurity (GI). This metric measures how often a randomly chosen element would be incorrectly identified if it was randomly included in a class, according to the distribution of classes in the partitions. In other words, GI measures how good a split is, based on how mixed (or separated)

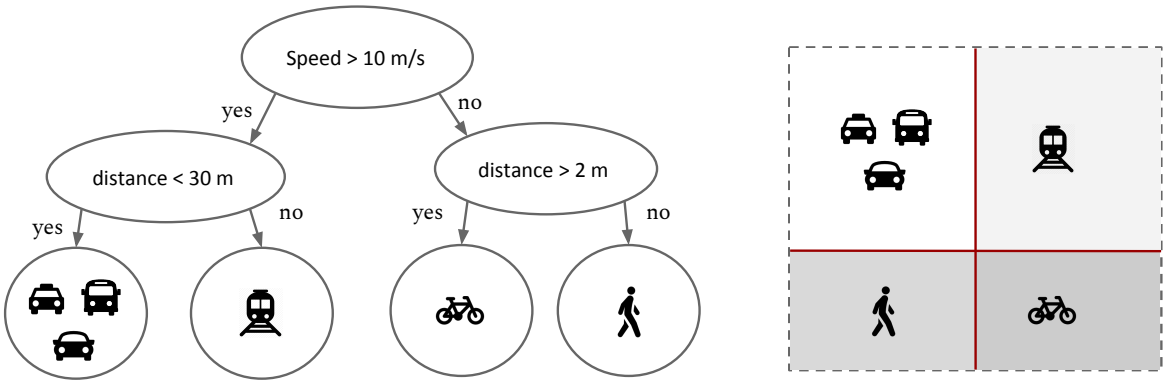


Figure 3.2: A tree representation of a decision tree model and its correspondent space split in axis-parallel hyperplanes

the classes are in the partitions. It is defined as follows.

$$\begin{aligned} GI &= \sum_{i=1}^{k} p(c_i) \sum_{j \neq i} p(c_j) = \sum_{i=1}^{k} p(c_i) \sum_{i=1}^{k} (1 - p(c_i)) = \sum_{i=1}^{k} p(c_i)((1 - p(c_i)) \\ &= \sum_{i=1}^{k} (p(c_i) - p(c_i)^2) = \sum_{i=1}^{k} p(c_i) - \sum_{i=1}^{k} p(c_i)^2 = 1 - \sum_{i=1}^{k} p(c_i)^2. \end{aligned} \tag{3.1}$$

where $p(c_i)$ is the probability of an observation with class i being chosen, $\sum_{j \neq i} p(c_j) = 1 - p(c_i)$ is the probability of a mistake in categorizing such observation, and k the number of classes.

If the partition is pure, the probability of the majority class is 1 and the probability of all other classes is 0, and thus, GI is 0. On the other hand, if each class is equally represented, the GI is equal to $(k-1)/k$. With this, we can see that higher values of Gini Index indicate more disorder and, consequently, lower values indicate more order in terms of the class labels (Zaki et al., 2014).

The RF is a kind of ensemble method known as Bootstrap Aggregating (Bagging). It creates a combination of weak classifiers (decision trees in this case), that are trained in various random sub-samples of the original dataset. These sub-samples are always of the same size of the original dataset, thus, since replacement is allowed, an observation can be in several sub-samples or even in the same sub-sample oftentimes. In this situation, every sample has an equal probability of being selected by a weak classifier, making the bagging model less susceptible to overfitting when applied to noisy data. Moreover, since DT are sensitive to data (i.e., if training data changes, the result can be quite different), the bagging scheme in RF model can help to reduce the variance of DT algorithm, which may improve the predictive results. The predicted

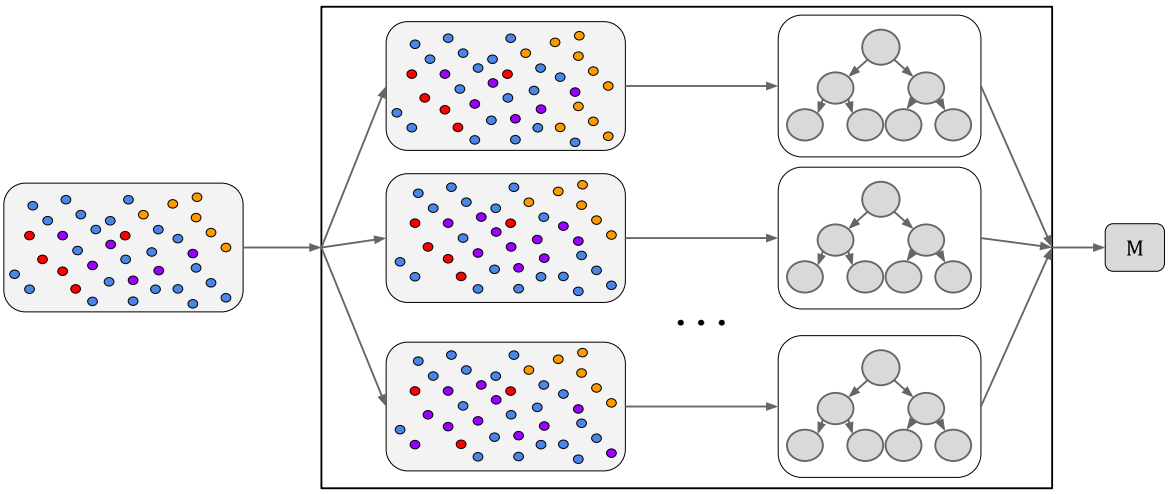


Figure 3.3: A Random Forest classifier

class to each data point is generally made by majority of votes (Tan et al., 2005). Another important characteristic of RF is that the generalization error depends on the strength and correlation of the individual trees. It means that the bagging scheme works better if the predictions are uncorrelated or weakly correlated. Nevertheless, the generalization error of RF converges to a limit as the number of trees in the forest becomes large (Breiman, 2001).

Figure 3.3 shows a RF model. From the original data, there are random sampling with replacement. These sub-samples train a DT model, that give their own prediction to each point in the dataset, and the RF model is responsible for joining the predictions to create its model. We note that the training in each DT model occurs in parallel, where one tree does not depend on any other.

GBDT is similar to RF in the sense of creating a combination of decision trees. However, unlike RF, where the training occurs in parallel fashion, GBDT trains its models sequentially, adding one by one. This ensemble scheme is known as Boosting. In Boosting, the algorithm adaptively changes the distribution of training examples so that the weak classifiers will focus on examples that are hard to classify. Hence, a tree is fit on the residual of the whole ensemble so far. In other words, GBDT uses trees that learn from the mistakes of the previous predictor. Differently from the traditional Boosting techniques, where the change in distribution is made by assigning weights to each training example (Tan et al., 2005), GBDT performs the same task by using gradients in the loss function, as shown by Friedman (2001). Hence, GBDT tries to reduce bias, whereas RF tries to reduce the variance (Xiao et al., 2017). A disadvantage presented by this model is that, if data is noisy, it is more likely to overfitting.

In Figure 3.4 we see a GBDT model. The first tree is trained in the original data.

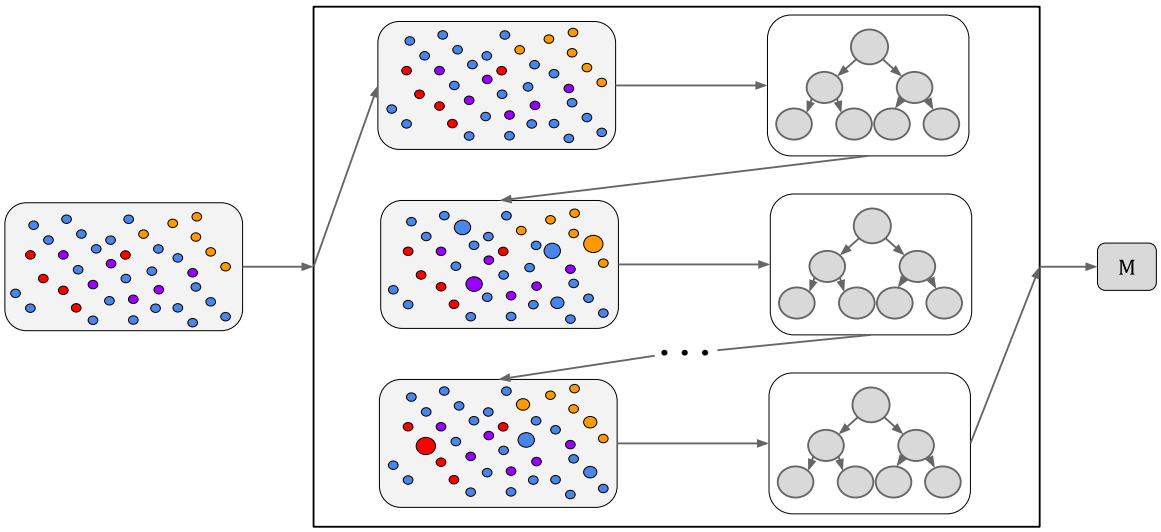


Figure 3.4: A Gradient Boosting Decision Tree classifier

The result of this model generates a modified dataset, in which some examples are bigger than others – they represent the examples that are not correctly learned by the first model, so their importance in classification is bigger than others. After, another tree is trained in the modified dataset. These steps, change importance of each point and train another model, is made throughout the whole ensemble. Hence, a GBDT model is fitted on the residual of all the trees presented.

3.2.4 Model Evaluation

The simplest way to evaluate the performance of a model of supervised learning is with a confusion matrix (also called error matrix), which exposes the predictions made versus the labels previously known. The structure of a confusion matrix is a squared matrix of the size of the number of classes, i.e., $L \times L$, where L is the number of labels.

Although the confusion matrix contains enough information to determine how well classification algorithms behave, may be more useful to reduce this information to a simple measure when assessing the predictive power of several algorithms. In this work, we will use the two most common measures of model evaluation used in transportation mode classification, and in data mining in general, defined as follows.

- **Accuracy (acc).** It measures how well a classifier correctly identifies or excludes a condition. In other words, this measure estimates how close the prediction of a classifier is of the actual labels. Acc is defined as

$$\text{acc} = \frac{\text{correct predictions}}{\text{total of observations}} = \frac{TP + TN}{P + N} = \frac{TP + TN}{TP + TN + FN + FP}, \tag{3.2}$$

where TP (true positive) represents the cases in which the predicted class of an observation is indeed its class; FP (false positive) refers to the cases where the predicted classes of a observation is mistaken, the model predicts that it belongs to a class but it does not (also known as type I error); in the TN (true negative) cases, the model predicts that a observation does not belong to a class and, in fact, it does not belong; and, finally, FN (false negative) describes the cases in which the model predicts that an observation does not belong to a class, but it actually belongs (also called type II error) (Tan et al., 2005).

- **F1-measure.** It is the weighted harmonic mean between precision (pre) and sensitivity (sen), defined as

$$\text{F1} = 2 \times \frac{\text{pre} \times \text{sen}}{\text{pre} + \text{sen}} = 2 \times \frac{TP}{TP + FP + FN}, \tag{3.3}$$

where precision represents the random variation of a model (so, the smaller, the better), as

$$\text{pre} = \frac{\sum_{i=1}^{L} \frac{TP_i}{TP_i + FP_i}}{L}, \tag{3.4}$$

and sensitivity explains how effectively a classifier identifies the positive prediction, as

$$\text{sen} = \frac{\sum_{i=1}^{L} \frac{TP_i}{TP_i + FN_i}}{L}. \tag{3.5}$$

In multi-class problems, such as the identification of many transportation modes, the evaluation measures can be calculated separately for each class, as in an one-versus-all scheme – this is called micro-averaged measures. Also, such measures can be computed as a unweighted average mean of all classes, called macro-averaged measures. While macro-averaging treats all classes equally, micro-averaging favors the larger. To choose which one to use depends on the goal (Sokolova and Lapalme, 2009). In this work, it is more suitable to consider macro-averaging measures, since our dataset is imbalanced (as we can see in Section 3.3.3).

Usually, to evaluate the performance of a classifier, the input dataset is randomly split into training and test set. The training set is used to learn the model M and the test set to evaluate the measures. However, in this type of evaluation, the results may be biased. For instance, the test set may have only easy or hard observations, leading to good or bad performance. To overcome such problem, reducing the variability and enhancing the confidence about the classification performance, the pre-defined partitioning is replaced by cross-validation.

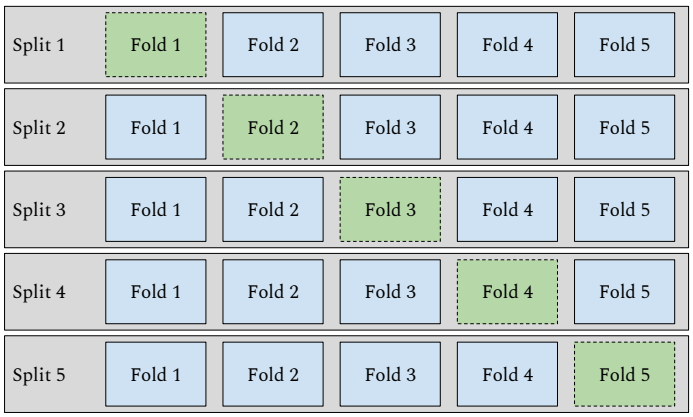


Figure 3.5: Illustration of cross-validation evaluation, with $K = 5$

In cross-validation, the dataset is divided into K equal-sized parts, called folds, namely $\{D_1, \ldots, D_K\}$. Each fold D_i is, in turn, treated as the test set, with the remaining folds $\bigcup_{j \neq i} D_j$ being used to train the model. After training the model, its total performance is the average performance in each test fold (Zaki et al., 2014). Figure 3.5 shows an illustration of this method. The green folds with dashed line are the test set in each split and the blue folds with full line are the training set. In the literature it is common to see K as 5 or 10. We used $K = 10$ in this work.

3.3 Trajectory Data

The development of information and communication technologies has made easier to acquire location data. Nowadays we are surrounded by a myriad of devices that are equipped with several different sensors, including those responsible for collecting location, such as Global Position Systems (GPS) and Radio Frequency Identification (RFID). Even sensors designed for different activities can be used to acquire location through Internet, as example, Wi-Fi. In this scenario, everywhere we go, we are surrounded by devices capable of locating not only humans, but also vehicles, animals, and other entities. Hence, with more location acquisition technology, we can obtain more data containing time-varying geographic information – the trajectory data.

This section is dedicated to discuss definitions about trajectory data that are important to this work. It is organized as follows. Section 3.3.1 defines trajectory data. Section 3.3.2 explains about data transformation. Section 3.3.3 examines the dataset used in this dissertation.

3.3.1 Definition

This definition below is inspired by Zheng (2015).

Definition. The mobility of an entity in a geographical space is recorded as a spatial *trajectory* $T = \{p_1, p_2, \ldots, p_n\}$, a trace of chronological points of size n, where each point p_i consists of a geospatial coordinate set and a timestamp, such as $p_i = (x_i, y_i, t_i)$, $i \in \mathbb{N}$.

Since x and y are varying together in time, we can consider a trajectory as a multivariate time series $T = \{X, Y\}$, where $X(t) = \{x_1, x_2, \ldots, x_n\}$ and $Y(t) = \{y_1, y_2, \ldots, y_n\}$.

3.3.2 Data Transformation

Trajectory data refers to our position on the Earth and is usually composed of latitude, longitude, and altitude. As described in Section 3.2.2, we can transform such information to emphasize the characteristics of the entity responsible for the trajectory; this transformation can be understood as the extraction of entity features, hence, this step is also called Feature Extraction. As an example, intuitively, humans can distinguish between a car and a bike by their speeds, a feature acquired from latitude and longitude. Knowing this, we transform latitude and longitude to five motion features, namely speed, acceleration, distance, bearing, and jerk, which will be described in this section.

- **Distance.** With this feature, we calculate the geographical distance between two succeeding GPS points. We use the geodesic distance between two trajectory points, which can be seen as a generalization of a straight distance to a curved surface. In other words, the geodesic distance is the shortest path between two points on the Earth, using the model of an ellipsoid of revolution. There are several ellipsoid models, being World Geodetic System (WGS) 84 the standard in cartography and satellite navigation, including GPS. Hence, we use it in this dissertation, as explained in Karney (2013). Figure 3.6 shows an example of geodesic distance, in solid blue line, and euclidean distance, in dashed red line, between two points.

- **Speed.** This feature calculates how fast an entity moves to one point to another. It can defined as in Equation 3.6.

$$S_{p_1} = \frac{dist(p_1, p_2)}{\Delta t}, \tag{3.6}$$

where *dist* is the distance between points p_1 and p_2, as described above; and $\Delta t = t_2 - t_1$, which is the difference time between the two points (p_1 with time t_1 and p_2 with time t_2). The SI units for its magnitude are meters per second ($\mathrm{m\,s^{-1}}$), so we calculate using these metrics.

- **Acceleration.** This feature determines the rate of changing of speed of an entity regarding to time. It is defined as in Equation 3.7.

$$A_{p_1} = \frac{S_{p_2} - S_{p_1}}{\Delta t}. \tag{3.7}$$

The SI units for its magnitude are $\mathrm{m\,s^{-2}}$.

- **Jerk.** It calculates the rate of changing of acceleration of an entity over time. The Equation 3.8 defines it.

$$J_{p_1} = \frac{A_{p_2} - A_{p_1}}{\Delta t}. \tag{3.8}$$

The SI units for its magnitude are $\mathrm{m\,s^{-3}}$.

- **Bearing.** This feature represents the direction of one point relative to another point. It is calculated in a clockwise direction, starting from the north line (Hills, 2000). Figure 3.7 (inspired by Hills (2000)) shows two examples of how to calculate bearing. In the first subfigure, the bearing of point p_1 from o is 65°. Although the point p_2 is 60° from North, as bearing is calculated in clockwise direction, its bearing from o is 300°. The second subfigure shows the relationship between two points for bearing calculation. The bearing of point p_1 from p_2 is 65°, whereas the bearing of point p_2 from p_1 is 245°. Equation 3.9 defines how to calculate bearing.

$$B_{p_1} = \arctan \frac{\sin(long_2 - long_1)\cos(lat_2)}{\cos(lat_1)\sin(lat_2) - \sin(lat_2)\cos(lat_2)\cos(long_2 - long_1)}, \tag{3.9}$$

where lat_1 and lat_2 refer to the latitude values of point 1 and 2, respectively; analogously, $long_1$ and $long_2$ refer to the longitude values of point 1 and 2, respectively. The bearing here is calculated in degrees, as it is its SI units.

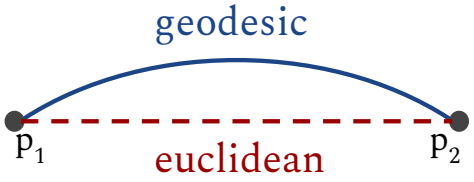


Figure 3.6: Geodesic (solid blue line) and euclidean distance (dashed red line)

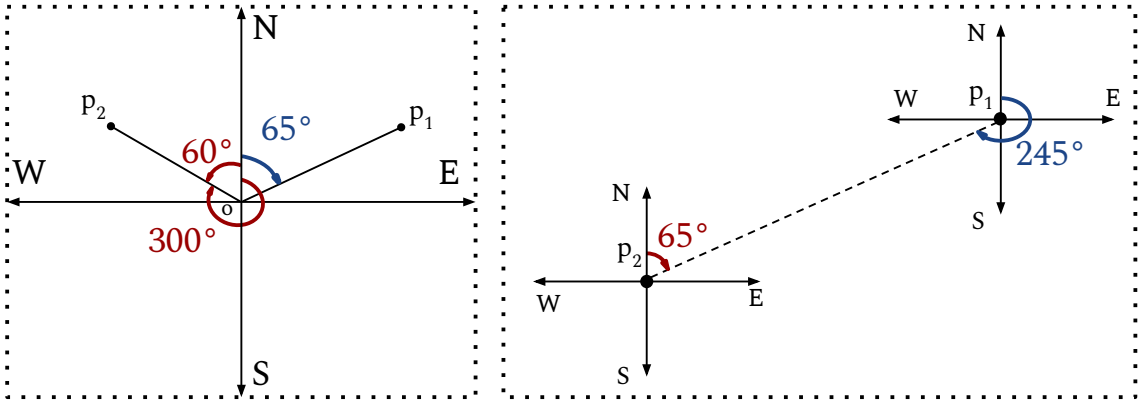


Figure 3.7: Bearing calculation examples

3.3.3 Dataset

In this work, we use the GeoLife[3] data, collected by Zheng (2015). This dataset presents GPS trajectories of 182 users over five years (from April 2007 to August 2012), containing latitude, longitude, and altitude information. Moreover, this dataset comprises 17,621 trajectories, with a total distance of 1,292,951 kilometers and a total duration of 50,176 hours. These trajectories are recorded by different sources (GPS loggers and GPS phones), having a variety of samples rates – more than 90% of them contains a dense representation, i.e., every 1 to 5 seconds or every 5 to 10 meters per point.

Among these users, 73 have the transportation mode information provided, such as driving, taking a bus, riding a bike, and walking. Table 3.1 lists the total distance and duration of the transportation modes. Other includes railway and boat, for instance.

3.4 Final Remarks

This section presented some concepts that are important to the development of this dissertation. The definitions given here comprised time series, data mining, and trajectory

[3] https://www.microsoft.com/en-us/download/details.aspx?id=52367

Table 3.1: Total distance and duration of transportation modes in GeoLife dataset

Transportation mode	**Distance (km)**	**Duration (hour)**
walk	10,123	5,460
bike	6,495	2,410
bus	20,281	1,507
car & taxi	32,866	2,384
airplane	24,789	40
other	9,493	404
total	14,304	12,953

fields. In time series, we defined univariate and multivariate data.

The data mining section described the steps used in this work, such as segmentation, classification, and model evaluation. More specifically, the classification part explained the Ensemble models, Random Forest and Gradient Boosting Decision Tree, that are used to classify transportation mode here.

After, we defined trajectory data and explained the features extracted in data transformation, such as distance, speed, acceleration, jerk, and bearing. Finally, we described the dataset used in this work.

Chapter 4

Ordinal Patterns Transformation

The unique properties presented by time series data, such as high dimensionality, presence of noise, and spatio-temporal dependence, impose many challenges to mining such data. In this context, a data transformation can mitigate these problems and improve classification.

In this chapter, we discuss the Ordinal Pattern (OP) transformation in Section 4.1. In Section 4.2, we propose a novel OP transformation that incorporates the amplitude information – which is not presented in the traditional OP method. We conclude this chapter in Section 4.3, presenting our final remarks.

4.1 Ordinal Patterns

Ordinal Patterns (OP) is a method of transforming time series data that does not require any model assumption about the time series. Therefore, it can be applied to any time series. This approach is based on the sequence that naturally arises from the time series, comparing the values that are in the same neighborhood and replacing them with a sequence of symbols (Bandt and Pompe, 2002).

Consider a time series $\mathbf{X}(t) = \{x_1, x_2, \ldots, x_n\}$ of size n, an embedding dimension $D \in \mathbb{N}$, and an embedding delay $\tau \in \mathbb{N}$ as well. We generate a sliding window $w_t \subseteq x$ in each time instant $t = \{1, \ldots, n - (D-1)\tau\}$, such as

$$w_t = \{x_t, x_{t+\tau}, \ldots, x_{t+(D-2)\tau}, x_{t+(D-1)\tau}\}, \tag{4.1}$$

i.e., each element within the sliding window is obtained from the time series in the time $t, \ldots, t + (D-1)\tau$. This corresponds to a time series sample at evenly spaced intervals.

The ordinal relation for each instant t consists of the permutation $\pi = \{r_0, r_1, \ldots, r_{D-1}\}$ of $\{0, 1, \ldots, D-1\}$, so that,

$$x_{t-r_{D-1}} \leq x_{t-r_{D-2}} \leq \cdots \leq x_{t-r_1} \leq x_{t-r_0}. \tag{4.2}$$

In other words, π represents the permutation of elements in the sliding window w_t, in ascending order. In order to obtain unique results, we define that, if a time series has elements such that $x_{t-r_i} = x_{t-r_{i-1}}$, we consider that $r_i < r_{i-1}$. Hence, the time series is converted to a set of ordinal patterns

$$\Pi_{OP} = \{\pi_1, \pi_2 \ldots, \pi_m\}, \tag{4.3}$$

where $m = n - (D-1)\tau$ and each π_m represents a permutation of the possible permutation set of $D!$ (Aquino et al., 2017).

The choice of D depends on the time series size and must satisfy the condition $n \gg D!$; the higher D is, the greater the time series length is necessary to have reliably extracted data (Rosso et al., 2007). Staniek and Lehnertz (2007) provide more details on this subject. For practical purposes, Bandt and Pompe (2002) recommend values such that $3 \leq D \leq 7$, which are adopted in this work.

4.1.1 Ordinal Patterns Probability Distribution

Once the ordinal patterns are constructed from the time series, we can transform it to another representation, the probability distribution. For all $D!$ possible permutation π of D, the relative frequency can be computed by the times a certain sequence appeared in the time series, divided by the number of total sequences. Thus, we obtain the histogram of the probability distribution $P \equiv \{p(\pi)\}$, which is defined by:

$$p(\pi) = \frac{\mid s_\pi \mid}{n - (D-1)\tau}, \tag{4.4}$$

where $\mid s_\pi \mid \in \{0, \ldots, m\}$ is the number of pattern observed of type π.

Figure 4.1 illustrates the process described above of extracting OP probability distribution from time series. **(i)** first, we have the original time series; **(ii)** we calculate sliding windows with D and τ values. We can see in the highlighted sliding window how D and τ behave: simplistically, D is relative to how many data points we consider to calculate the ordinal pattern, when $D = 3$, we use three points, $D = 4$, we use four points, and so on; τ refers to the spacing between two consecutive points in the sliding

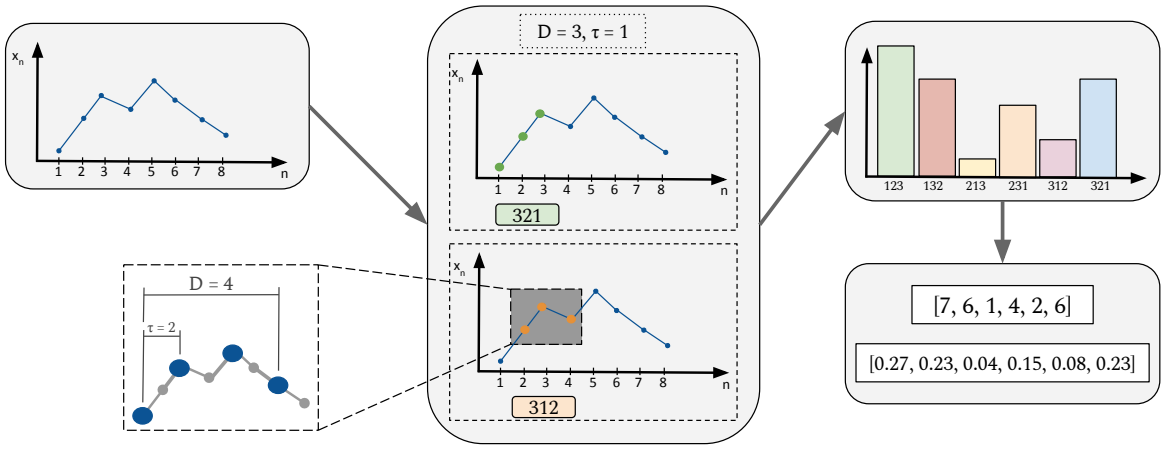


Figure 4.1: The process of extracting Ordinal Patterns from time series

window, when $\tau = 1$, we use immediate neighboring points, if $\tau = 2$, we use points that are two points apart, and so forth. **(iii)** After computing the ordinal patterns to all the time series, we extract the histogram of relative frequency. **(iv)** We can use the probability distribution or the frequency distribution of this histogram, as we can see in the last step of the process.

The OP probability distribution, and the OP transformation itself, contains several advantages, such as:

- **Simplicity and fast calculation.** The transformation of the time series into the set of ordinal patterns depends on the length n and the embedding dimension D. The time complexity to perform such transformation is bounded by $O(nD \log D)$, assuming that the permutations are obtained by sorting the data points of each sliding window by a common sort algorithm, such as merge sort (of complexity $O(D \log D)$ in the worst case). Using worse sort algorithms, such as selection sort, of complexity $O(D^2)$ in the worst case, the time complexity increases to $O(nD^2)$. However, as D is recommended to be at most 7, the sorting will take no more than 7 elements. To that end, the complexity of this approach is broadly reliant on the size n of the time series (Borges et al., 2019).

- **Scalability.** As said above, this is a simple and fast transformation, with a linear computational cost. It allows its application in huge time series. Besides that, the histogram of probability distribution is independent of the size of the time series – it always needs a space of $D!$.

- **Robustness.** This transformation is robust to the presence of observational and dynamic noise and it is also invariant to non-linear monotonic transforma-

tions (Aquino et al., 2017; Rosso et al., 2007).

Lastly, lets see an example about the OP transformation to better understanding. Suppose we have $X(t) = \{1, 10, 3, 5, 8, 1, 2, 3\}$. For $D = 3$ and $\tau = 1$, our sliding windows are

$$w = \{\{1, 10, 3\}, \{10, 3, 5\}, \{3, 5, 8\}, \{5, 8, 1\}, \{8, 1, 2\}, \{1, 2, 3\}\},$$

which give the ordinal patterns $\Pi_{OP} = \{312, 132, 321, 213, 132, 123\}$. The histogram is $\{1, 2, 1, 0, 1, 1\}$ and the probability distribution is approximately $p(\pi) = \{0.17, 0.33, 0.17, 0, 0.17, 0.17\}$.

4.2 Ordinal Patterns with Amplitude Information

Although presenting many advantages, as aforementioned presented, OP has some drawbacks, such as of not considering amplitude information. Ignoring the amplitude may lead to lower classification performance, since many time series can have important information in this feature. Knowing this, studies have tried to overcome it. For instance, Fadlallah et al. (2013) proposed a modification to calculate the Permutation Entropy (PE) (Bandt and Pompe, 2002), called Weighted-permutation entropy (WPE). They weighted the relative frequency of each extracted ordinal pattern with the variance of the corresponding part of the time series. However, if all the values in time series are identical, the variance forces WPE to zero.

Azami and Escudero (2016) proposed a modification to PE as well, named Amplitude-Aware Permutation Entropy (AAPE). They added a relative normalized value to the corresponding ordinal patterns probability distribution. This approach modifies directly the PE calculation, needing the extraction of this feature.

Sun et al. (2014) proposed the use of a pair of symbol: the first is the OP transformation. The second is obtained by splitting the range of the time series into equal regions. The amplitude level of the ordinal pattern is then described by the index of the bin within such pattern falls. In other words, they extract the bin value with the largest amplitude. However, they do not consider the case in which the ordinal patterns are not all in the same bin. For instance, in the case of outliers, they are not clear about which amplitude level is considered.

The proposal we introduce in this dissertation does not need the calculation of PE, using directly the histogram of probability distribution, but adding an amplitude

information. We call this approach of Ordinal Pattern with Amplitude Information (OPAI).

Similarly to Sun et al. (2014), we divide the range of the time series into equal q regions, of height h, transforming the continuous time series to discrete values. We calculate h as

$$h = \frac{\max(X) - \min(X)}{q}. \tag{4.5}$$

That is, we split the space between the maximum and minimum value found in the time series into q bins of the discrete space Q, which is composed of equal intervals with height h. We then map each time series value to its correspondent interval, i. e., $x_i \to Q$. We define Q as

$$Q = \{[\min(X), \min(X) + h], \ldots, (\min(X) + (q-1)h, \max(X)]\}, \tag{4.6}$$

where we can see that $\max(X) = \min(X) + qh$.

Furthermore, we note that if $q = 1$, OPAI is equal to OP transformation, since the only existing interval would contain any value between the maximum and the minimum value of the time series, which is the time series itself.

To calculate the amplitude information presented in each sliding window w_t (defined in Equation 4.1), we do the correspondence between the values in w_t to the discretized space Q, as

$$a_t = \max(Q_{w_t}) - \min(Q_{w_t}). \tag{4.7}$$

In other words, our amplitude information a is calculated by subtracting the maximum and the minimum value in the sliding window when transformed to the discretized space. It is different from h, that considers the values in all time series, while a_t uses the values in the sliding window of time t.

Hence, the time series is converted to a set of ordinal pattern containing the amplitude information, $\Pi_{OPAI} = \{(\pi_q, a_1), \ldots, (\pi_m, a_m)\}$, where $m = n - (D-1)\tau$ and each (π_m, a_m) represents the combination of the possible permutation set of $D!$ and the amplitude information.

The histogram of the probability distribution is calculated as in Equation 4.4, but, as stated above, the patterns observed in this approach is the cartesian product between the ordinal patterns π and the amplitude information a. Hence, the time complexity of histogram calculation is higher, $O(qD!)$.

In fact, properly speaking, we do not extract all the amplitude information of

each ordinal pattern, but its amplitude variation. It is possible to extract the complete amplitude information, but keeping it is computationally expensive, since the histogram is the product between amplitude information and the ordinal pattern, it would be $q^D D!$. For instance, if $D = 5$ and $q = 8$, the histogram would have more than 10 billions bins.

Due to this, our hypothesis is that the amplitude variation contained in each sliding window is enough to capture the amplitude information of the time series. This proposal inherits all the advantages presented by OP transformation, described in Section 4.1.

Figure 4.2 shows the process of extracting the OPAI transformation from time series. The process is very similar to OP transformation. Excluding the second step, where we calculate the ordinal patterns with the D and τ values, we also calculate the amplitude value q. Furthermore, the resulting histogram will be larger.

We will use the same example of the section before to illustrate the concepts presented here for clarification. Let the time series $X(t) = \{1, 10, 3, 5, 8, 1, 2, 3\}$. For $D = 3$, $\tau = 1$, and $q = 4$, we have $h = (10 - 1)/4 = 2.25$. It gives the space $Q = \{[1.0, 3.25], (3.25, 5.5], (5.5, 7.75], (7.75, 10.0]\}$, which will call from 0 to 3, respectively. Our sliding windows are

$$w_t = \{\{1, 10, 3\}, \{10, 3, 5\}, \{3, 5, 8\}, \{5, 8, 1\}, \{8, 1, 2\}, \{1, 2, 3\}\},$$

that give the ordinal patterns $\Pi_{OP} = \{312, 132, 321, 213, 132, 123\}$. To calculate the amplitude information, we subtract the maximum and minimum value of each sliding window. For instance, for $a_1 = \max(Q_{w_1}) - \min(Q_{w_1}) = 3 - 0 = 3$. So we have

$$a = \{3, 3, 3, 3, 3, 0\}.$$

Finally, we obtain the OPAI transformation patterns $\Pi_{OPAI} = \{(312, 3), (132, 3), (321, 3), (213, 3), (132, 3), (123, 0)\}$.

4.3 Final Remarks

This chapter explained the Ordinal Patterns transformation. This data representation is very useful in different time series domains, as shown in Section 2.1.3. However, it has the disadvantage of not storing amplitude information. Some approaches in literature tried to overcome it, but they require the PE extraction or considerate just the highest amplitude point.

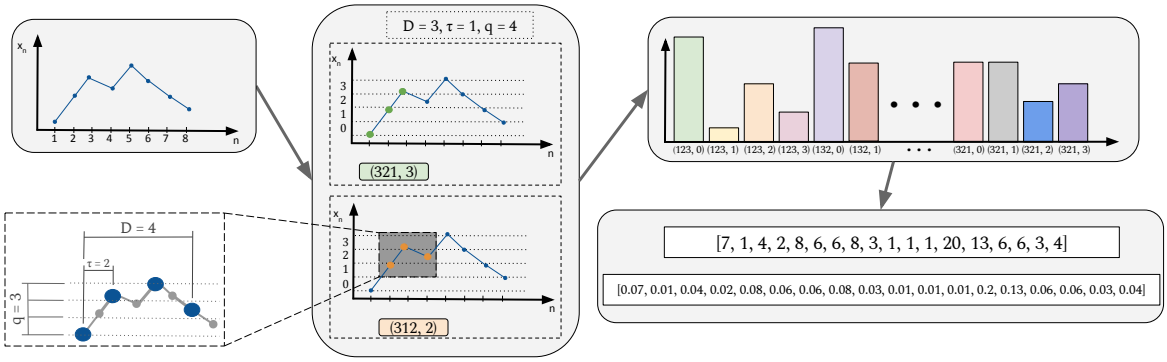


Figure 4.2: The process of extracting Ordinal Patterns with Amplitude Information from time series

In this dissertation, we propose a modification to the OP transformation in order to keep the amplitude information. Indeed, we only extract the amplitude variation, since to keep all the amplitude information is computationally expensive. We hypothesize that it is enough to capture the amplitude information, enhancing time series classification. Although our proposal has a slight higher computational cost to calculate the histogram of probability distribution, it inherits some characteristics of OP transformation, such as simplitcity, fast calculation, scalabitility, and robustness.

Chapter 5

Transportation Mode Classification

The urban areas need to develop sustainable solutions due to the high demand for their services, which is surpassing the supply. Such solutions can, for instance, reduce traffic jam and travel time, leading to an improvement in life quality. Hence, identify how citizens commute, that is, identify which transportation modes are used in a city is an important step for this task. This identification should rely on technologies that do not depend on the active participation of the user.

In other words, we need a framework capable of automatically extracting transportation mode information from user's location. This is our main objective, as described in Section 1.2. To achieve such goal, we unravel it in three specific objectives, that will be answer in this chapter. Section 5.1 investigates the properties of transportation modes, based on features extracted from GPS data; Section 5.2 proposes a methodology to mine transportation mode information from trajectory data; and Section 5.3 explores the characterization of transportation modes based on Ordinal Patterns with Amplitude Information (OPAI). Finally, Section 5.4 concludes this chapter with our final remarks.

5.1 Properties of Transportation Modes

This section is dedicated to answer our first specific objective:

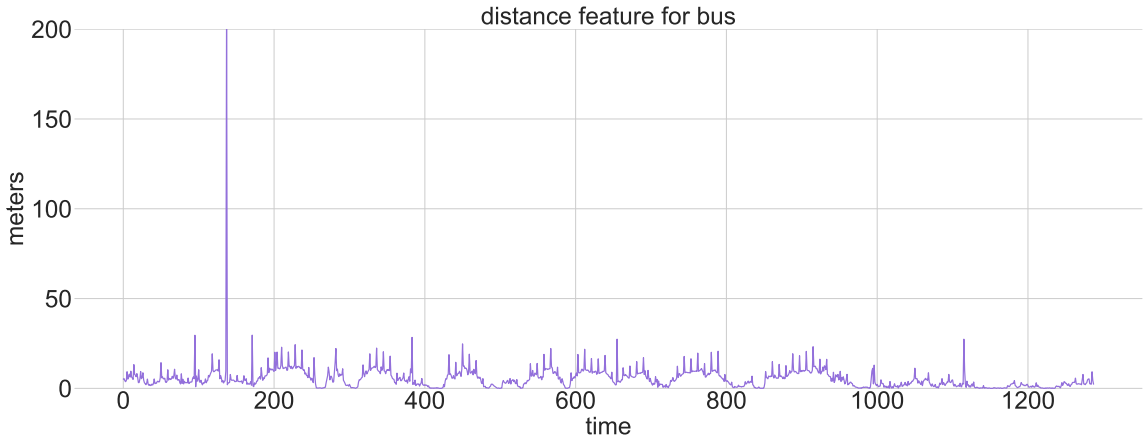
What are the properties of transportation modes?

This characterization is an important step in order to understand the peculiarities of each transportation mode and, hence, being able to distinguish between them. In fact, if we can distinguish the transportation mode used, we can develop solutions to such identification, leading us to the fully achievement of our main goal.

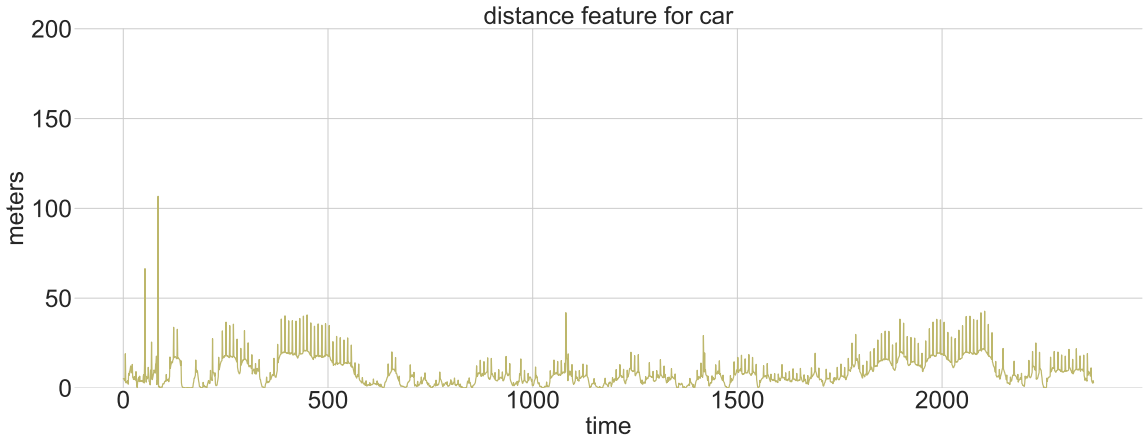
The first step to answer this question is to preprocess the trajectory data, transforming them to more comprehensible and concise features; data transformation is better discussed in Section 3.3.2. We transform the trajectories to five motion features, namely jerk, acceleration, speed, bearing, and distance. In Figure 5.1, we see examples of time series for the distance feature made by different transportation modes, namely, bus (the purple time series in Figure 5.1a), car (the golden in Figure 5.1b) and walking (Figure 5.1c, the green time series). The y axis is limited from 0 to 200 meters in all figures. We see that car and bus have more similarity in their features, with spikes along its entire length. Intuitively, car and bus, by being road-based transportation modes, can cover similar distance between points. Whereas the distance feature for walking is softer, without many spikes and the distance covered is smaller, with at most 10 meters. Therefore, although walking stands out from bus and car when described by the distance feature, car and bus are more difficult to discriminate. Our hypothesis is that combining the features with OPAI transformation, our proposal, we can distinguish between transportation modes, allowing good identification and, consequently, good classification results. In other words, we can say that such features can contrast some transportation modes and this difference can be more evident when using OPAI rather than OP transformation. To support our hypothesis, we analyze some examples of features for different transportation modes.

In Figure 5.2 we have the OP transformation (when $q = 1$, OPAI is the OP transformation), with $D = 3$ and $\tau = 1$, of distance values for bus (the purple bars in Figure 5.2a), car (Figure 5.2b, the gold bars), and walk (green bars in Figure 5.2c). The y axis is limited from 0 to 0.3 and represents the probability distribution from the π patterns. The π patterns are sorted, i.e., for $D = 3$, they are $\{123, 132, 213, 231, 312, 321\}$; they are hidden in the figures for easy reading. The OP transformation for bus and car have similar distribution, in which 123 and 321 have higher values than the other patterns, and these patterns also have similar values among them. Using OP, we know the trends in the trajectory, but we do not know the variation between them. For instance, for $D = 3$ and $\tau = 1$, $X_1 = \{0, 2, 3, 30\}$ and $X_2 = \{0, 5, 15, 30\}$ result in the same transformation $\Pi_{OP} = \{321, 321\}$. Hence, the distance between each point is not captured. We can interpret it as following: for OP, there is no difference if a car travels 2 meters and after 1 meter or 1 meters and after 27 meters, the pattern will be the same.

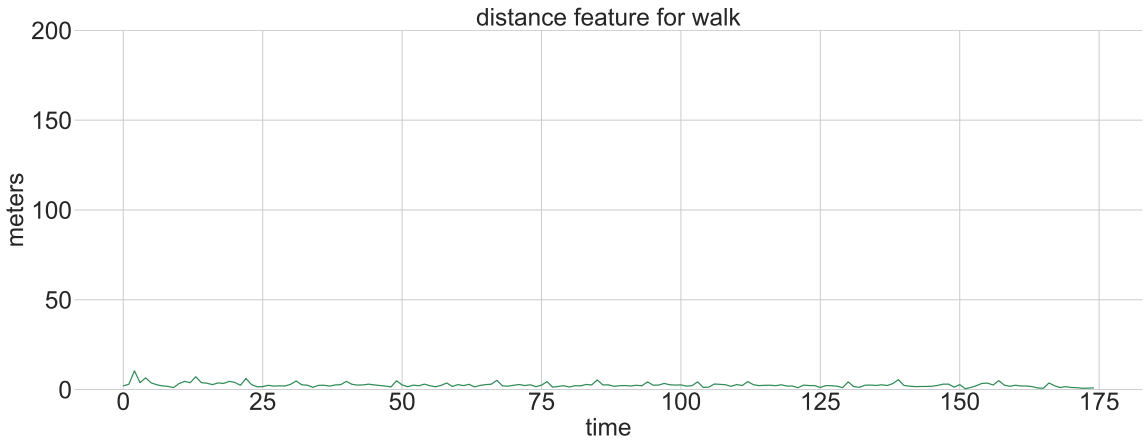
Figure 5.3 shows the OPAI transformation, with $D = 3$, $\tau = 1$, and $q = 16$ for the same trajectory represented in Figure 5.2. The colors are the same of the aforementioned figure. The distribution for OPAI transformation are more distinguished between car and bus than using OP. Although, as before, the patterns 123 and 321



(a) Time series for distance feature made by a bus

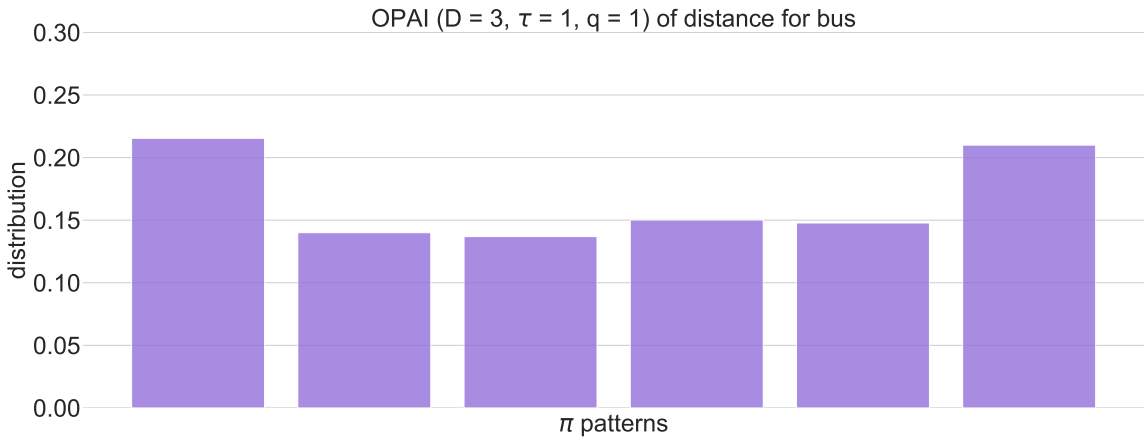


(b) Time series for distance feature made by a car

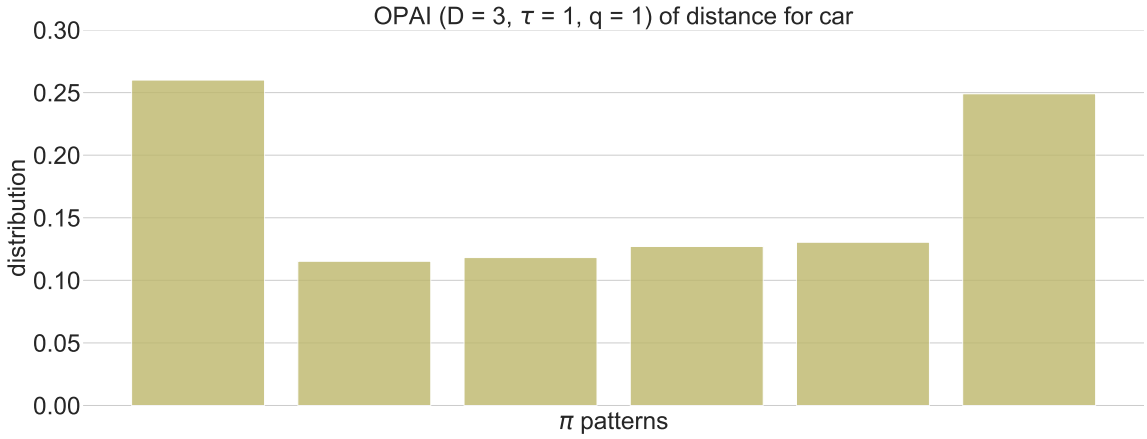


(c) Time series for distance feature made by foot (walking)

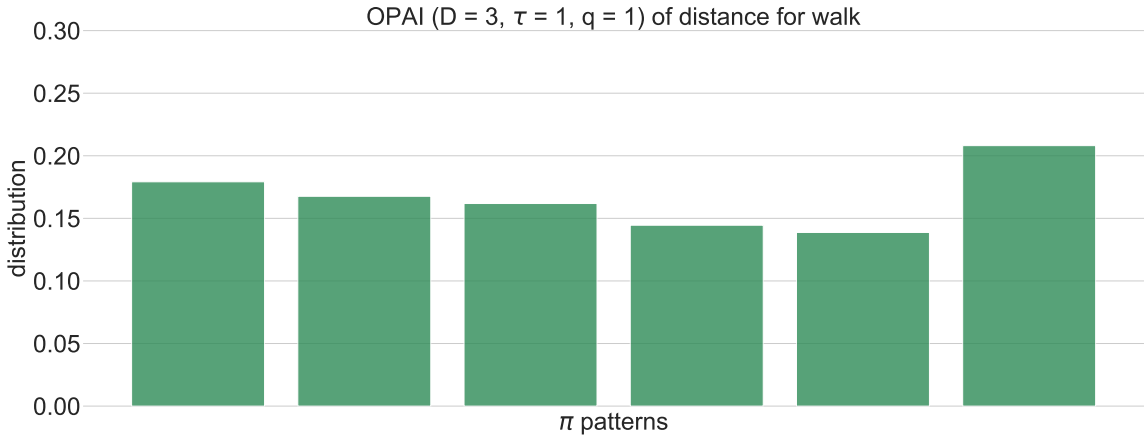
Figure 5.1: Example of time series for distance feature made by different transportation modes: bus, car, and walking, respectively.



(a) OP transformation $(D = 3, \tau = 1, q = 1)$ of distance for bus



(b) OP transformation $(D = 3, \tau = 1, q = 1)$ of distance for car



(c) OP transformation $(D = 3, \tau = 1, q = 1)$ of distance for walking

Figure 5.2: Example of OP transformation (with $D = 3$, $\tau = 1$, and $q = 1$) of distance for bus, car, and walk, respectively

have higher values than others, there is more than one representation for each pattern. In bus, we have more patterns with amplitude variation 1 and 2; whereas in car, amplitude variation of 2 and 3 appear as well. That is, in this trajectory, though most of the trends are up or down, there is variation in amplitude between the points. With OPAI transformation, it is easier to see the contrast between the road-based modes (car and bus) and walking, in which there is not a pattern with higher values than 0.01; in fact, most of the data is below 0.05. Using the same example as before to represent how OPAI can capture variations that OP cannot, for $D = 3$, $\tau = 1$, and $q = 16$, $X_1 = \{0, 2, 3, 30\}$ (in the discrete space $Q_1 = \{0, 1, 1, 15\}$) and $X_2 = \{0, 5, 15, 30\}$ ($Q_2 = \{0, 2, 7, 15\}$) result in the transformations $\Pi_{OPAI} = \{(321, 1), (321, 14)\}$ and $\Pi_{OPAI} = \{(321, 7), (321, 13)\}$, respectively.

5.2 Extraction of Transportation Mode Information from GPS Data

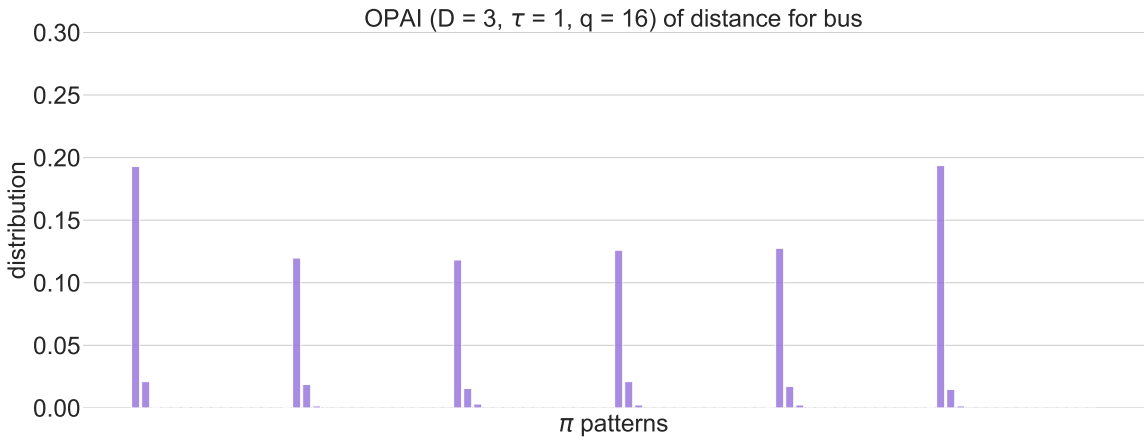
In this section, we want to answer our second specific objective:

How can we extract transportation mode information from GPS data?

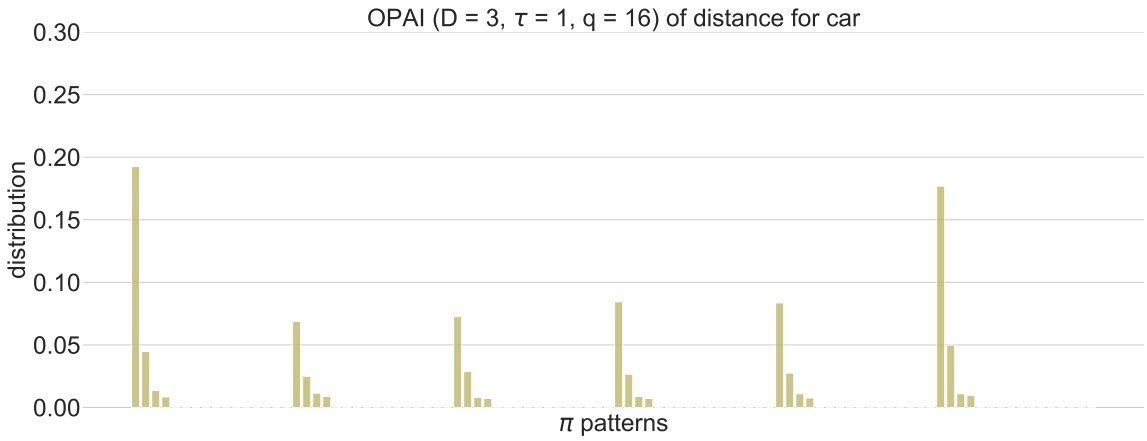
To answer this question and achieve our goal, we developed a framework composed of four steps, called Segmentation, Feature Extraction, Data Transformation, and Classification, as shown in Figure 5.4. In the first step, our framework receives a raw trajectory data (latitude, longitude, and altitude information) and segment them. From each segment, the framework extracts features related to transportation mode properties. After, these features are transformed and this transformation feds the classification algorithm. With these steps, which will be explained throughout this section, our framework is able to classify the transportation modes used. Moreover, we present a complexity analysis of our framework as well.

Segmentation

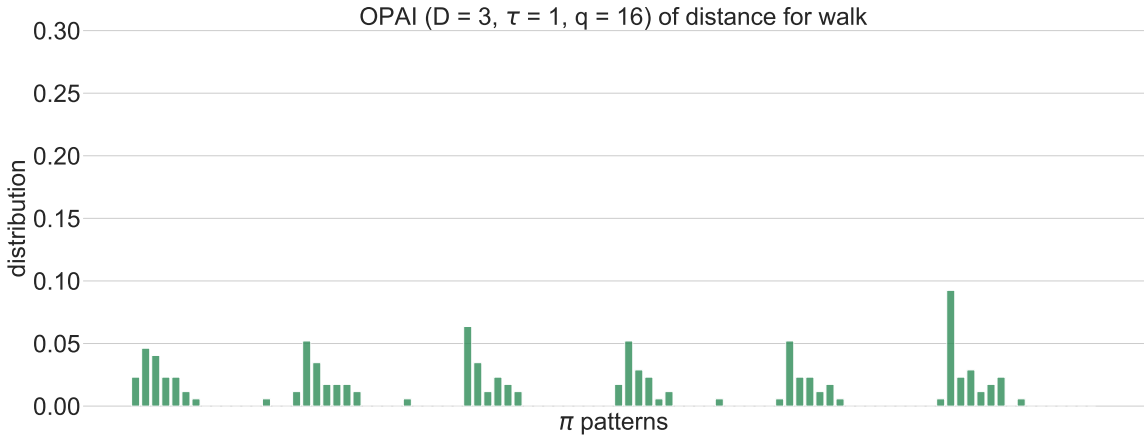
As said in Section 3.3.3, we use the portion of GeoLife dataset which contains the transportation mode information, that includes 73 users. We segment the raw trajectory provided by separating it by user, day, and transportation mode. This type of segmentation supposes that the transportation mode used in each raw trajectory is known. It is a strong assumption, however, since we are focusing in classification, we use such segmentation technique. Several works in literature do the same first step, such as Etemad et al. (2018) and Dabiri and Heaslip (2018), to cite a few.



(a) OPAI transformation ($D = 3, \tau = 1, q = 16$) of distance for bus



(b) OPAI transformation ($D = 3, \tau = 1, q = 16$) of distance for car



(c) OPAI transformation ($D = 3, \tau = 1, q = 16$) of distance for walking

Figure 5.3: Example of OPAI transformation (with $D = 3$, $\tau = 1$, and $q = 16$) of distance for bus, car, and walk, respectively

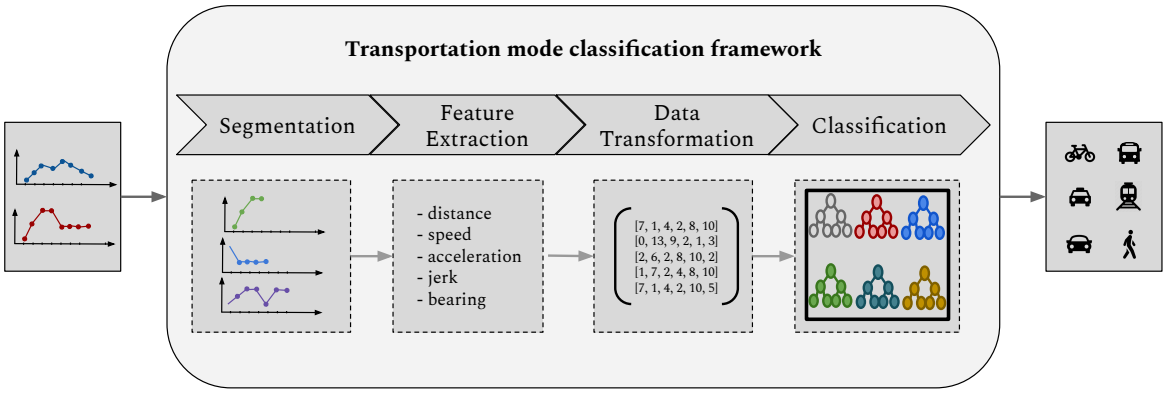


Figure 5.4: Transportation Mode Classification Framework

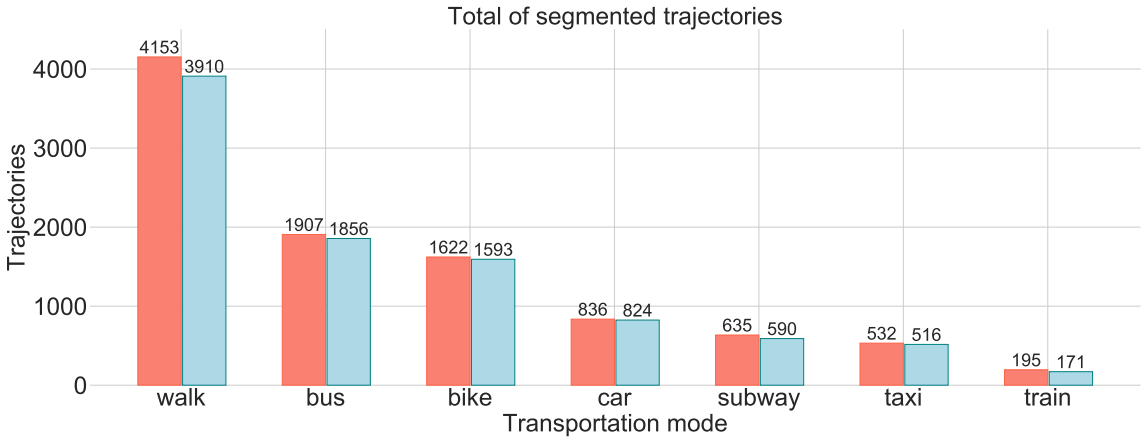


Figure 5.5: Total of segmented trajectories

Furthermore, we discard trajectory samples with less than 10 points, since small samples may input noise into the model due to a low quality data.

Figure 5.5 shows the segments obtained by splitting the raw trajectories (the red bars) and after discarding the short ones (the blue bars). Before discarding the short trajectories, in total, we have 9880 trajectories. The data are unbalanced, with most of it being walking trajectories (about 42%), followed by bus (19%), bike (18%), car (8%), subway (6%), taxi (5%), and train (2%) trajectories. After discarding, we have 9460 trajectories, with the following proportion: about 41% for walking trajectories, 20% for bus, 17% for bike, 9% for car, 6% for subway, 5% for taxi, and 2% for train.

Feature Extraction

This step comprehends the transformation of segmented trajectories to motion features. We call it feature extraction to differentiate from trajectory transformation into OPAI

representation.

Trajectory data is usually provided in latitude, longitude, and altitude. These features, however, are difficult to interpret. Hence, their transformation in another feature improve the transportation mode identification. We transform the trajectories (latitude and longitude information) into five motion features, namely geodesic distance, speed, acceleration, jerk, and bearing. Such features are handcrafted and require a previous knowledge about the problem domain, hence, this is the most time-consuming step.

Data Transformation

This step is responsible for transform the motion features to OPAI transformation. In other words, each motion feature of the trajectories is converted into its correspondent OPAI representation. Hence, for each trajectory, we have five representations, one for each feature.

This step requires the adjustment of the parameters: embedded dimension D, embedded delay τ, and amplitude regions q.

Classification

This step receives the motion features and classifies them using a tree-based Ensemble method.

To evaluate our classification, we use a cross-fold validation, with $K = 10$. Since our data are imbalanced, we stratify each class in order to force each fold to have the same class distribution, i.e., we preserve the percentage of sample for each class in each fold. It ensures that neither of the classes are over-represented, leading to increased unrealistically results.

The input of the classifier algorithm is a matrix of the motion features in OPAI representation, where each feature is a row of the matrix. Joining these features make the algorithm receive different information from the transportation modes motion that a trajectory is related to. With this, the classifier can learn more, hence, generalizing more such learning and improving its results.

Algorithm Complexity Analysis

We have the start and end point of each transportation mode used by each user separated from the raw trajectory data. Thus, the Segmentation step finds the trajectory related to the transportation mode label and split it. The cost for this operation is

$O(\mid \ell \mid rn)$, where $\mid \ell \mid$ is the size of transportation mode label data, r represents how many raw trajectories a user has, and n the size of user's trajectory.

The cost to extract each motion feature is $O(n)$. Since each feature is extracted separately, the cost for the Feature Extraction step is $O(n)$.

Algorithm 1 shows the pseudo-code to the OPAI transformation. In line 3 we calculate the q regions of height h and define the discrete space Q. It must find the maximum and minimum of the time series, calculate the Q regions and, after that, locate each trajectory value into Q – the cost for such operation is $O(n)$. The lines from 4 to 10 go through all the trajectories, doing the following operations: in line 5, we select the indices of points that will be transformed to OPAI – that is, we select every D points, spaced by τ value. This costs $O(1)$. Line 6 indeed obtains such points values from the trajectory data, based on the *indices* values, with the cost $O(1)$ as well. The ARGSORT function returns the indices that would sort an array, hence, in line 7 we obtain the ordinal pattern π. It uses a simple algorithm to sort, such as Merge sort, having a cost of $O(D \log D)$. In line 8, we identify the points (the same transformed to OP) in the Q space, again with cost $O(1)$. Finally, in line 9, we get the amplitude variation of the pattern, with cost $O(1)$. The costs for get the maximum and minimum value in the amplitude information is $O(D)$ for each operation. Hence, the complexity cost for the OPAI transformation is $O(nD \log D)$.

Algorithm 1 Ordinal Pattern with Amplitude Information Transformation

```
1: procedure OPAI_TRANSFORMATION(trajectory, D, τ, q)
2:     symbols = LIST()
3:     Q = CUT(trajectory, q)
4:     for p in RANGE(1, SIZE(trajectory) − (D − 1)τ) do
5:         indices = RANGE(p, p + (D − 1)τ, τ)
6:         values = LOC(indices, trajectory)
7:         π = ARGSORT(values)
8:         amplitude_information = LOC(indices, Q)
9:         a = MAX(amplitude_information) − MIN(amplitude_information)
10:        symbols.APPEND((π, a))
11:    end for
12: return symbols
13: end procedure
```

To obtain the histograms, we first obtain the list of patterns, that costs $O(qD!)$. After, we count each pattern obtained in the trajectory. It costs $O(n)$. Then, the cost to this step is $O(qD!)$. Therefore, the Data Transformation step, that includes the OPAI transformation and histogram construction, costs $O((qD!) + (nD^2)) = O(qD!)$.

The complexity of the Classification step is related to the classification algorithm used. For instance, a Random Forest classifier has cost $O(N^2 \times f \times t)$, where N is the total of trajectories, f is the total of features, and t the total of trees.

5.3 Characterization of Transportation Mode using Ordinal Patterns with Amplitude Information

In this section, we investigate the following question:

> *Is Ordinal Patterns distribution capable of providing good characterization of transportation mode?*

We believe that the Ordinal Patterns (OP) distribution, based on its characteristics (discussed in Chapter 4), is a transformation capable of highlighting the underlying patterns presented in trajectory data, leading to good classification results. The absence of amplitude information may impact negatively in such classification results, however. Knowing this, we believe that adding this information will enhance the transportation modes identification, hence, increasing the classification. Thereupon, in this work, we proposed a modification to OP that considers the amplitude information, called Ordinal Patterns with Amplitude Information (OPAI). The focus of this section is to analyze the behavior of our proposal, OPAI, when classifying transportation modes, and compare it to OP transformation. This comparison takes place through investigating how the parameters influence the classification values. We start with the q parameter in Section 5.3.1, followed by the parameters of the classifiers in Section 5.3.2, and D and τ parameters in Sections 5.3.3 and 5.3.4, respectively. After fine tuning the parameters, Section 5.3.5 is dedicated to analyze the confusion matrices of the classification that uses best parameters, in order to better understand it. Finally, in Section 5.3.6, we compare our proposal to the works presented in literature, previously discussed in Chapter 2.

All the implementation procedure in this dissertation was made in Python (version 3.7.3), using the Anaconda distribution (version 4.7.11)[1]. Specifically, we used the libraries geopy[2], to calculate the geodesic distance; xgboost[3] to perform the GBDT classifier; and scikit-learn[4] to the others classification steps (e.g., Random Forest classifier, average voting Ensemble scheme, K-fold).

[1] `https://www.anaconda.com`
[2] `https://geopy.readthedocs.io/en/stable/`
[3] `https://xgboost.readthedocs.io/en/latest/`
[4] `https://scikit-learn.org/stable/`

The machine in which they were executed had the following configuration: Ubuntu 18.04.3 OS, 20 × Intel(R) Core(TM) i9-9900X CPU @ 3.50GHz, and 125 GB RAM.

We performed the experiments in five sets of different transportation modes. These sets were evaluated in previous works as well, described in Section 2.2, helping the comparison of our results. The sets are:

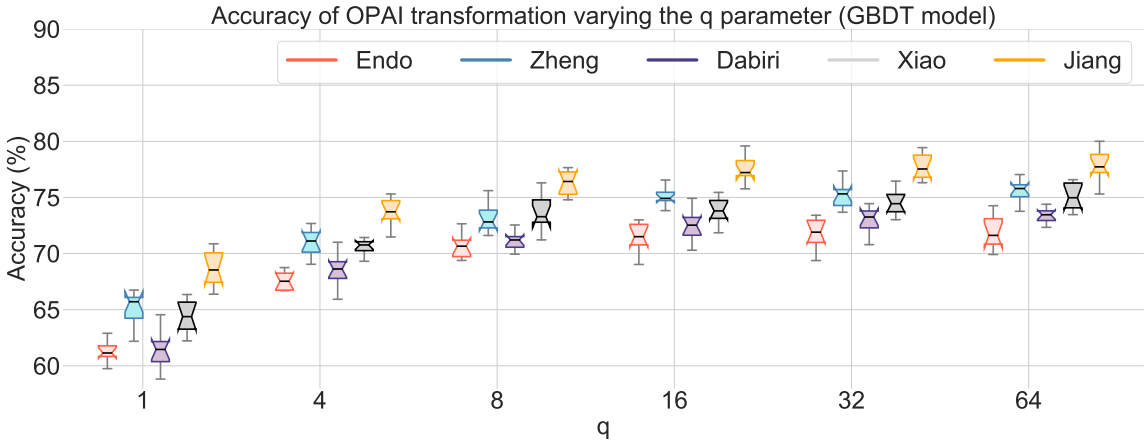
- {bus, car, subway, taxi, train, walk, bike} (used first by Endo et al. (2016));
- {walking, bike, bus, driving (car and taxi)} (Zheng et al. (2008a));
- {walk, bike, bus, driving, subway and train} (Dabiri and Heaslip (2018));
- {bike, car, walk, bus} (Jiang et al. (2017)); and
- {walk, bus, car, bike, taxi, train, subway} (Xiao et al. (2017)).

To facilitate the addressing about the sets, we refer to them in this work by the first author's name that used such partitions originally.

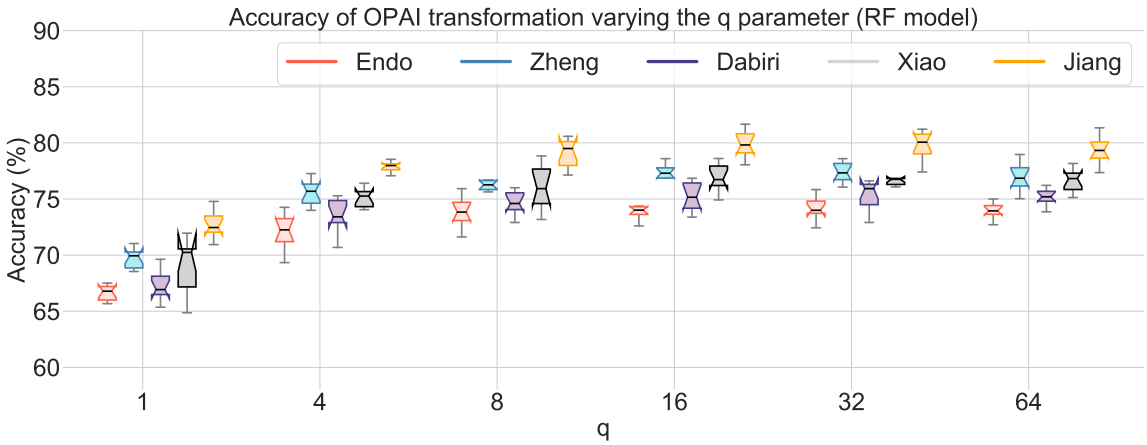
5.3.1 Impact of q in OPAI transformation

First, we analyze the impact of q in OPAI transformation. This impact is measured by accuracy results, since our focus in this work is transportation mode classification. The q parameter, exclusive to OPAI, represents the space binning, i.e., in how many q regions the space of the continuous time series is divided, in order to generate a discrete space. It is important to capture the amplitude variation – the variation in q determines the granularity of the discrete space. It means that bigger values give a finer granularity, whereas smaller values provide a coarse granularity. In other words, bigger values creates more regions with smaller heights, hence, the transformation from time series to discrete space will be more detailed. However, not necessarily bigger values deliver best accuracy results, since the discrete space with more regions may be less robust to noise.

We vary this parameter as $q = \{1, 4, 8, 16, 32, 64\}$ and fix the others, which are the OPAI parameters $D = 3$, $\tau = 1$, and 50 trees to build the model classifiers: Random Forest (RF) and Gradient Boosting Decision Tree (GBDT) is used to investigate the performance in this experiment. They are both tree-based ensemble methods, but they are build in different ways, thus, it is expected different classification values. Nevertheless, have in mind that when $q = 1$, OPAI is the same as OP transformation.



(a) Accuracy results using GBDT classifier



(b) Accuracy results using RF classifier

Figure 5.6: Accuracy results for variation in q parameter to OPAI transformation using different classifiers

Figure 5.6 shows the values of accuracy obtained by changing q to RF and GBDT models, in Figure 5.6b and 5.6a, respectively. We have five groups, where red boxes represent the Endo et al. (2016) set, the blue boxes display the Zheng et al. (2008a) set, purple boxes are the Dabiri and Heaslip (2018) set, the grey boxes portray the Xiao et al. (2017) set, and the yellow boxes express the Jiang et al. (2017) set. The y axis is zoomed from 58% to 90% to help showing the difference between the results.

Figure 5.6a shows the accuracy results when using the GBDT classifier. We see that the changing in the q parameter has influence in the performance. For instance, between $q = 1$ and $q = 4$, there is an average increase of about 5% in accuracy of all sets. This difference grows to about 10%, in average, between $q = 1$ and $q = 64$. However, there is little gain in accuracy values when q is from 16 to 64, with about

1% of difference in average. Hence, $q = 16$ is the best choice to this parameter in this scenario, since it will have a smaller histogram to store, with $qD! = 16 \cdot 3! = 96$ bins, against 192 and 384 when $q = 32$ and $q = 64$, respectively, making the classification faster.

Figure 5.6b shows the accuracy results of the RF model. Similar to the aforementioned case, with the GBDT classifier, when the q value is increased, the accuracy results improve. Between $q = 1$ and $q = 4$, for example, there are an average difference of about 5% in all sets. This discrepancy is greater when $q = 1$ and $q = 64$, for example, with about 8% in all sets, in average. Likewise, $q = 16$ is a better choice to this parameter with this classifier, in order to provide a smaller histogram without an intense decrease in accuracy.

The difference in accuracy results between OP (when $q = 1$) and OPAI transformation is easily seen in this experiment, with OPAI values surpassing OP in both cases. Moreover, between both classifiers, using the same q value, the results varies up to 5%, as we can see when $q = 1$. As the q value grows, the discrepancy in accuracy results decreases; for instance, when $q = 64$, the difference is of about 2%. Furthermore, in general, the RF model presents better accuracy results than GBDT model, although the latter has a smaller variance.

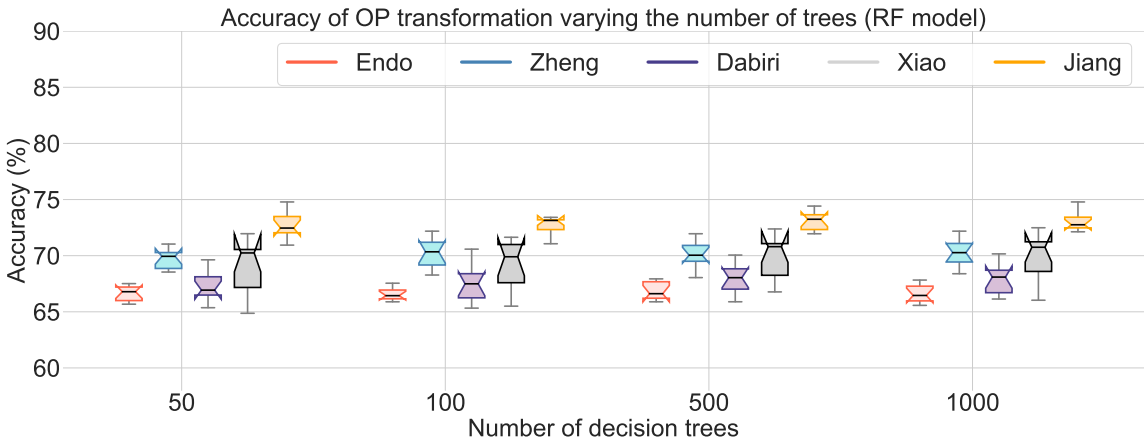
5.3.2 Impact of the parameters of the classifiers

In the next experiment, we compare the influence of the parameters of the classifiers. To this, although each classifier has several parameters, we vary only the number of decision trees presented in each model, since it is a common parameter. We fix the other parameters to $D = 3$, $\tau = 1$, and $q = 16$ and vary the number of trees as $trees = \{50, 100, 500, 1000\}$. We classify OP transformation as well, to compare it to our proposal.

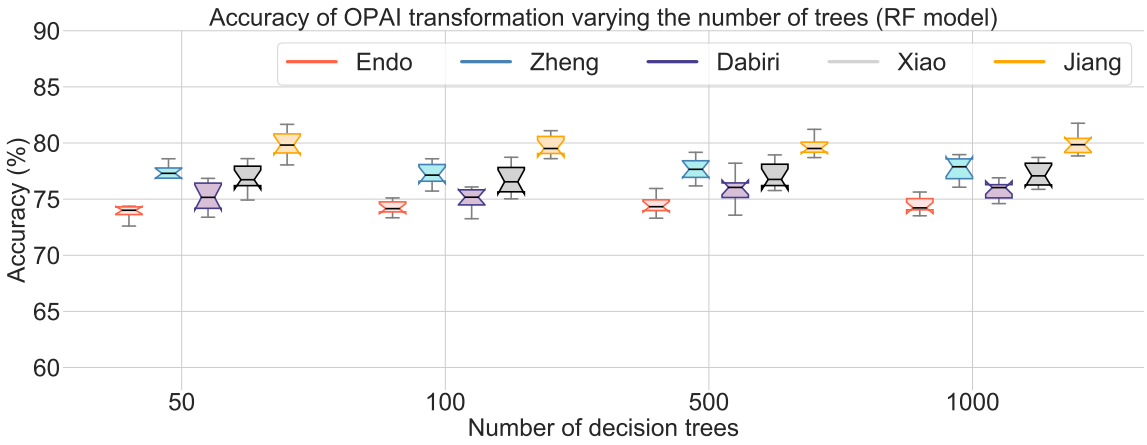
Figure 5.7 shows the accuracy results when varying the number of decision trees to GBDT and RF classifiers in OP and OPAI transformation. The scheme color is equal to the previous experiment.

In Figures 5.7a and 5.7b we see OP and OPAI transformation, respectively, classified with RF classifier. In OP transformation, the best number of trees is 500, however, little variance is presented in all the parameter value, being virtually the same accuracy. This steadiness also occurs in OPAI transformation. Comparing both transformation, we see that OPAI presents better accuracy results independent on the number of trees, with an improvement of about 8% in all sets, in average.

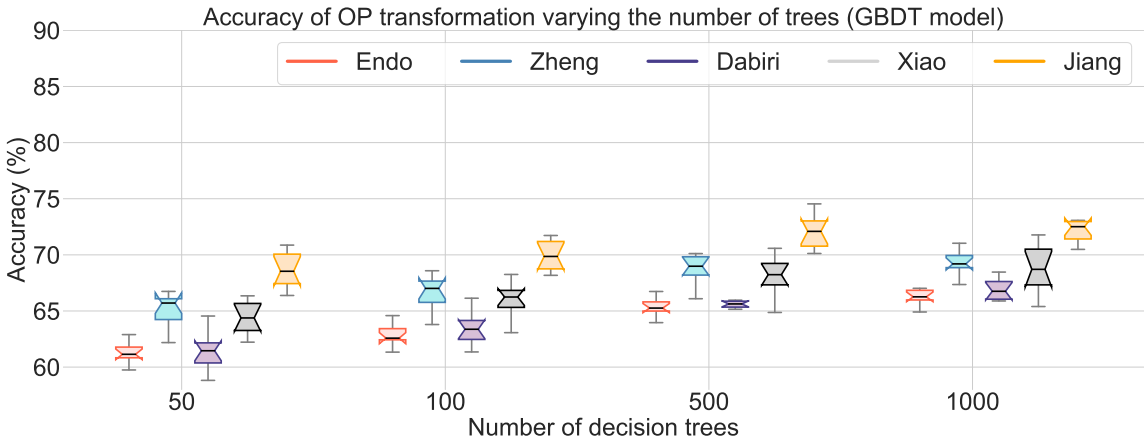
Figures 5.7c and 5.7d show OP and OPAI transformation classified with GBDT



(a) Accuracy results using RF classifier and OP transformation

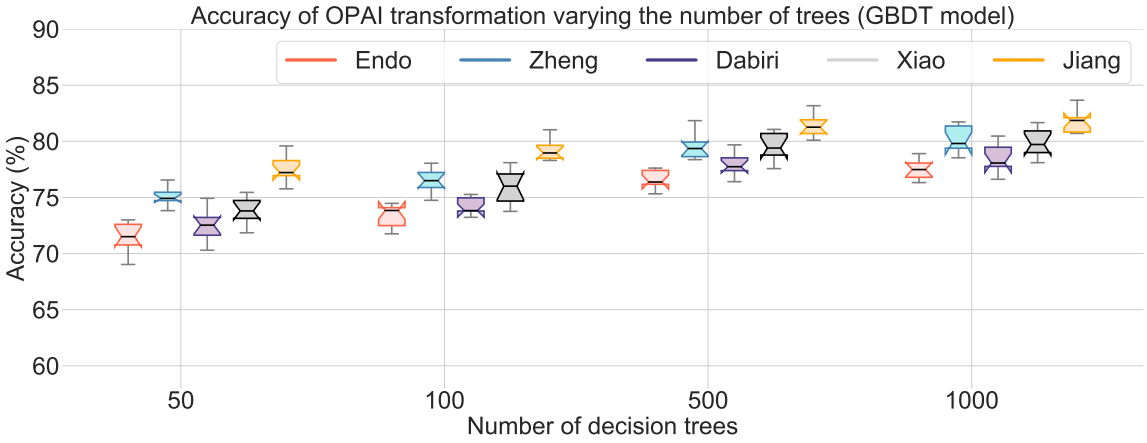


(b) Accuracy results using RF classifier and OPAI transformation



(c) Accuracy results using GBDT classifier and OP transformation

Figure 5.7: Accuracy results for variation in tree parameters to different classifiers using OP and OPAI transformation



(d) Accuracy results using GBDT classifier and OPAI transformation

Figure 5.7: Accuracy results for variation in tree parameters to different classifiers using OP and OPAI transformation

classifier, respectively. Differently from the RF classifier, as the number of trees grows, GBDT model provides better accuracy results to both transformation – OP increases about 5% in average in all sets and OPAI about 6% in average. Comparing the two transformations, OPAI transformation provides an increase of about 10% in accuracy, in average. Hence, independently of the classifiers, we can see that OPAI gives better accuracy results than OP.

Additionally, we can say that, when using OP transformation, RF is a better choice to the context of this dissertation. It presents an average gain of about 5% in accuracy. The RF classifier presents the advantage of being robust to the number of trees – the changing in accuracy results is minimal. When using the OPAI transformation, however, the GBDT model provides better results, with a gain of about 2%, in average, over the RF model.

5.3.3 Impact of D

In this section we evaluate the impact of D in the OPAI and OP transformation. Table 5.1 shows how many bins the histogram has when varying D. When $q = 1$, we have the OP transformation, and, although several values are possible to q in OPAI transformation, we evaluate only to $q = 16$, since this value were used in the previous experiment, where were demonstrated that it provided good accuracy results. The number of bins is calculated as $qD!$. Even though the acceptable range, determined by Bandt and Pompe (2002), is small, from 3 to 7, a factorial growth can easily turn the problem intractable. The table shows such growth; for $D = 7$ in OP transformation and

Table 5.1: Histogram size to OP and OPAI transformation

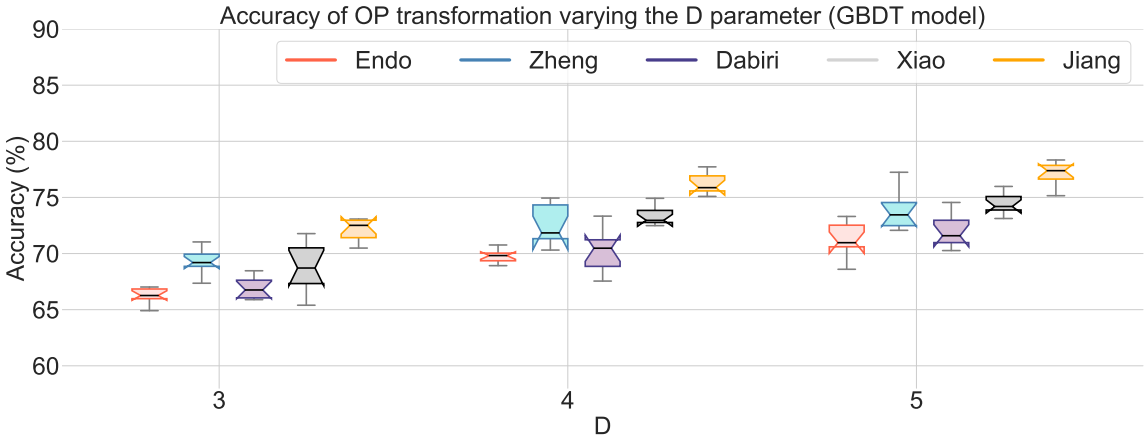
q	D	bins
1	3	6
1	4	24
1	5	120
1	6	720
1	7	5040
16	3	96
16	4	384
16	5	1920
16	6	11520
16	7	80640

from $D = 5$ in OPAI transformation, we have more than 1000 bins in each histogram, with $D = 6$ and $D = 7$ in OPAI having more than 10000. This can be a problem not only because of the storage, but capturing this amount of patterns in a time series can lead to overfitting: too many details that do not generalize well, damaging the classifier performance. Moreover, this big amount of bins will make the classifier very slow. Knowing this, we evaluate the accuracy results for $D = \{3, 4, 5\}$. We fix $\tau = 1$ and classify with GBDT using 1000 trees.

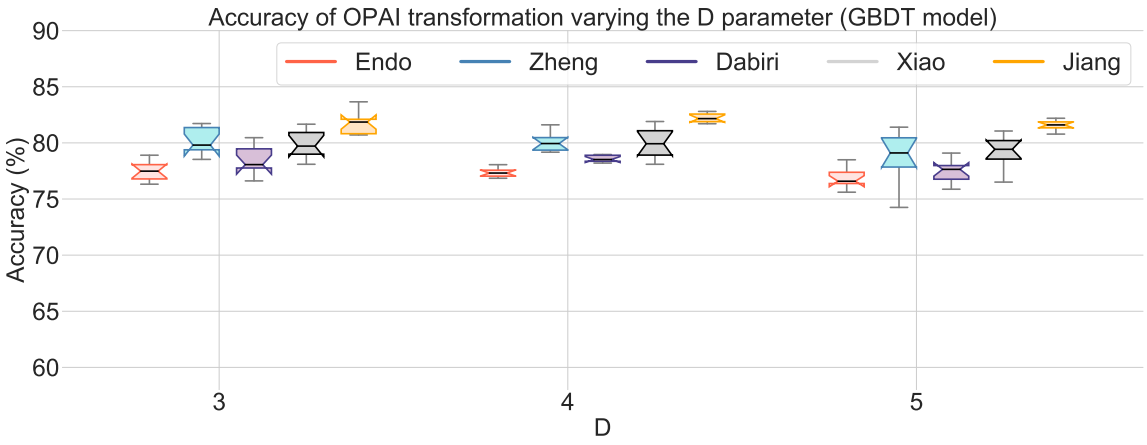
Figure 5.8 shows how the variation in D value affects the accuracy results for OP and OPAI classification. Comparing each D between both transformation, we see that the difference in accuracy results decreases as D increases: when $D = 3$, there is about 10% of difference in average in all sets; it reduces to about 7% when $D = 4$ and to about 5% when $D = 5$. In all cases, OPAI transformation gives better accuracy results than OP transformation.

The OP transformation, in Figure 5.8a, presents an increasing in accuracy when varying D; between $D = 3$ and $D = 4$, there is an improvement of about 3% (in average), and between $D = 4$ and $D = 5$, the difference is of about 2% in average. Hence, the best D value to this transformation is $D = 5$.

Figure 5.8b displays the accuracy results for OPAI transformation in variation of D. The parameter values $D = 3$ and $D = 4$ present similar results, whereas in $D = 5$ is possible to note a slight decrease in accuracy results. Thus, it is better to adopt $D = 3$ as the best choice to this transformation, since it produces a smaller histogram, leading to a faster classification.



(a) Accuracy results for D values in OP transformation



(b) Accuracy results for D values in OPAI transformation

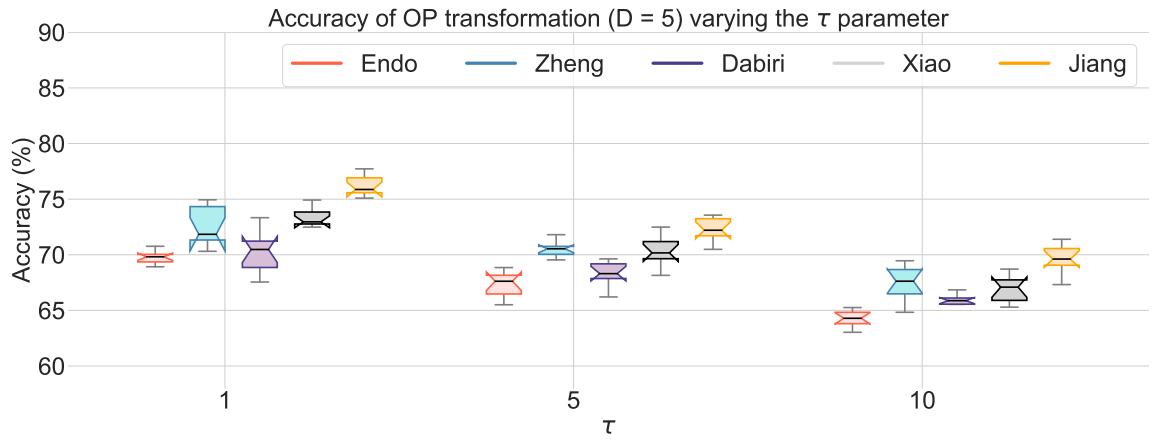
Figure 5.8: Accuracy results for variation in D parameter to OP and OPAI transformation

5.3.4 Impact of τ

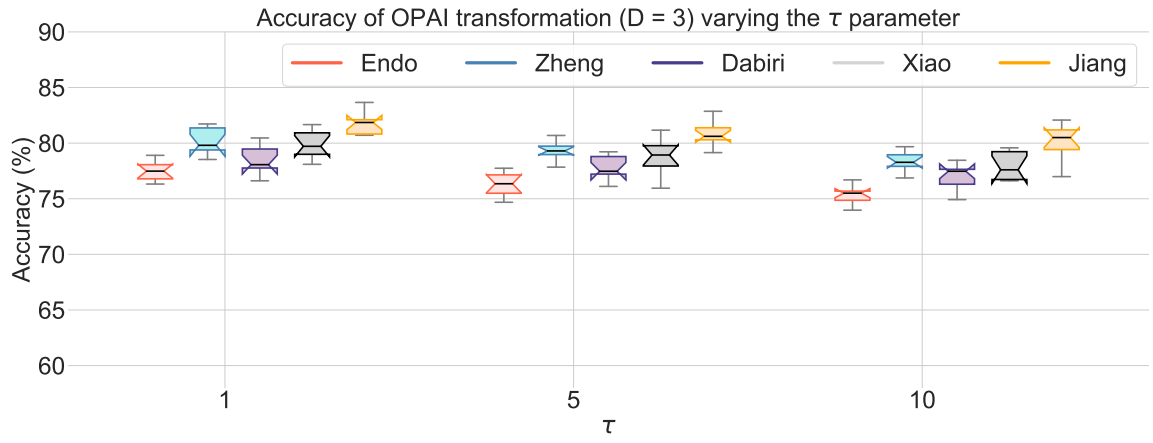
In the experiment described in this section, we evaluate the impact of τ in transportation mode classification using OP and OPAI transformation. The τ defines the delay between the data points used to the transformation. The maximum τ value depends on the time series size n and the dimension D, being limited by:

$$\tau < \frac{n}{D-1} \tag{5.1}$$

The greater the τ value, the greater must be the time series samples. For example, for $D = 3$ and $\tau = 1$, the time series has to be, at least, greater than 2 points; for



(a) Accuracy results for τ values in OP transformation



(b) Accuracy results for τ values in OPAI transformation

Figure 5.9: Accuracy results for variation in τ parameter to OP and OPAI transformation

$D = 3$ and $\tau = 2$, $n > 4$; for $D = 3$ and $\tau = 3$, $n > 6$; $D = 3$ and $\tau = 5$, $n > 10$; $D = 3$ and $\tau = 10$, $n > 20$; $D = 3$ and $\tau = 15$, $n > 30$, and so on. Moreover, the D value also influences in the size needed to the time series. As examples, for $D = 4$ and $\tau = 5$, the time series must be greater than 15; for $D = 5$ and $\tau = 5$, $n > 20$; $D = 7$ and $\tau = 5$, $n > 30$, and so forth. In other words, as τ and D increase, the longer the trajectories must be. With this, smaller trajectories, and, consequently, data, are discarded.

We analyze the accuracy results for $\tau = \{1, 5, 10\}$. We fix $D = 3$ and $q = 16$ to OPAI transformation and $D = 5$ to OP, which were the best parameters in the previous experiments. Both use the GBDT classifier with 1000 trees.

Figure 5.9 shows the accuracy results when varying τ for OP and OPAI transformation. In Figure 5.9a we see the OP transformation. As τ grows, the accuracy

Table 5.2: Accuracy and F1 results to OP ($D = 5, \tau = 1$) and OPAI ($D = 3, \tau = 1, q = 16$) transformation

	OP		**OPAI**	
	Acc (%)	**F1-macro** (%)	**Acc** (%)	**F1-macro** (%)
Endo	71.3435 (±1.35)	61.9762 (±2.31)	77.5051 (±0.90)	71.2442 (±1.70)
Zheng	73.8701 (±1.75)	70.0206 (±1.35)	80.2047 (±1.09)	77.1446 (±1.14)
Dabiri	71.9446 (±1.62)	67.6068 (±1.76)	78.5203 (±1.16)	75.7842 (±1.70)
Xiao	74.4087 (±1.40)	68.3089 (±2.74)	79.8834 (±1.12)	76.0084 (±2.63)
Jiang	76.9880 (±1.33)	73.1574 (±1.19)	81.7426 (±0.93)	78.9165 (±1.22)

results decrease of about 3% in average.

For OPAI transformation, in Figure 5.9b, also there is a drop in accuracy results, however, it is of about 1% in this case.

Comparing both transformation, we see that the difference between OP and OPAI increases from about 6% (when $\tau = 1$), to 9% when $\tau = 5$, and expands to about 11% in average, when $\tau = 10$.

Hence, we can presume that OPAI, besides given better accuracy results, is also less affected by τ variation.

Moreover, to both transformation, $\tau = 1$ is a better choice to the context of this dissertation.

5.3.5 Confusion Matrices

In this section we look more closely to the best results of OP and OPAI transformation presented in the previous experiments. Table 5.2 exhibits the metrics accuracy and F1-macro score to OP and OPAI transformation, along with their standard deviation. In all sets, the accuracy result is greater when using OPAI transformation. Specifically, in Endo set, the improvement is of 6.16%; in Zheng set, 6.33%; in Dabiri set, 6.58%; in Xiao set, 5.47%; and in Jiang set, 4.75%. Note that, for OP transformation, the best result is when $D = 5$. It means that, in this classification task, there is a greater need to fine tuning to achieve its higher accuracy result, with the difference between each D value, for instance, being of about 2% (from $D = 3$ to $D = 5$, thus, there is a loss of about 4% in accuracy). OPAI transformation, however, achieves its highest accuracy value with $D = 3$ and it is comparatively steady among its D values. The disadvantage of OPAI in contrast to OP is its speed (i.e., OPAI is slower than OP) and need for more storage space.

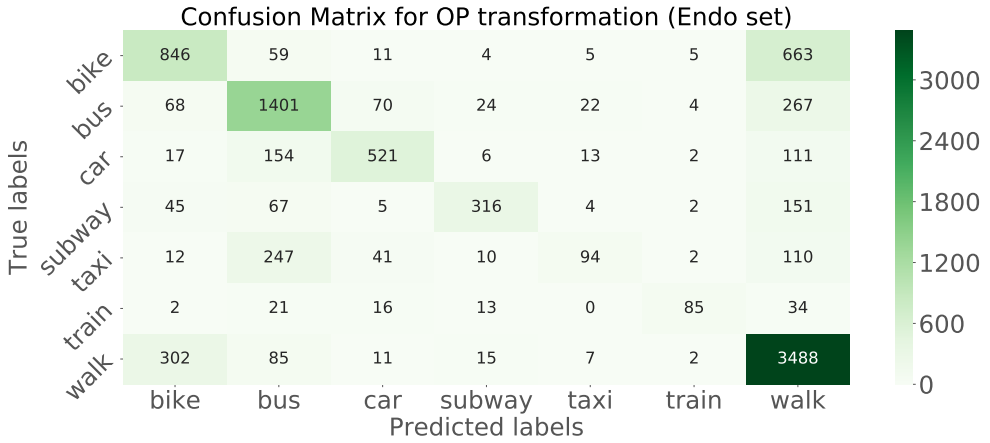
In Figure 5.10 we see the confusion matrices to each set using OP transformation.

Walk is a very misclassified transportation mode, in all sets, especially with bike. It makes sense since they, intuitively, have similar features, such as speed. Another confusion occurs between car and bus, transportation modes with alike features as well, such as distance. In Endo set (Figure 5.10a), we see that taxi is heavily misclassified as bus. The confusion matrix to Xiao set (Figure 5.10c), as well as Endo set (Figure 5.10a, show that this same situation appears between car and bus. Consequently, driving, that includes car and taxi, is also misplaced as bus, as we can see in Dabiri (Figure 5.10d) and Zheng (Figure 5.10b) set. It is a plausible condition, since all of these transportation modes are road-based, and may have similar features values, e.g., bearing. Moreover, in Endo set, taxi is also misclassified as walk. There is not an easy explanation to this fact. One can suppose that an extraordinary fact were occurring that forced the taxi to have similar features to walk or vice-versa. In fact, it would be necessary to analyze the events that were influencing the mobility captured by the GPS data to understand this situation.

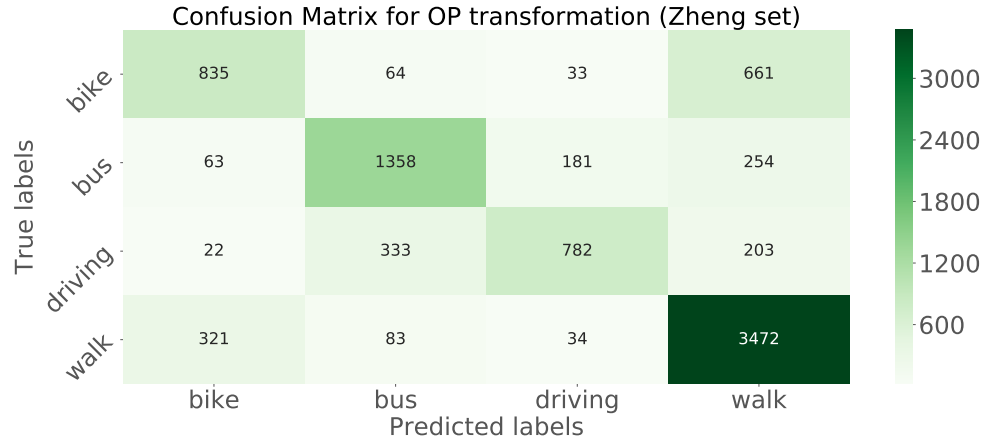
Figure 5.11 shows the confusion matrices to OPAI transformation. It makes the same mistakes as OP, such as taxi and car with bus and everything with walk. However, it is better to correctly classify the data examples, hence, increasing the true positives and decreasing the false positives. It leads to better accuracy and F1 results, as we can see in the previous experiments, presented from Section 5.3.1 to Section 5.3.4. For instance, in Endo set (Figure 5.11a), the taxi almost doubles its true positives, going from 94 to 170. However, OP is slight better to classify bus in all cases, although OPAI surpass it in the others transportation modes.

5.3.6 Comparison to previous works

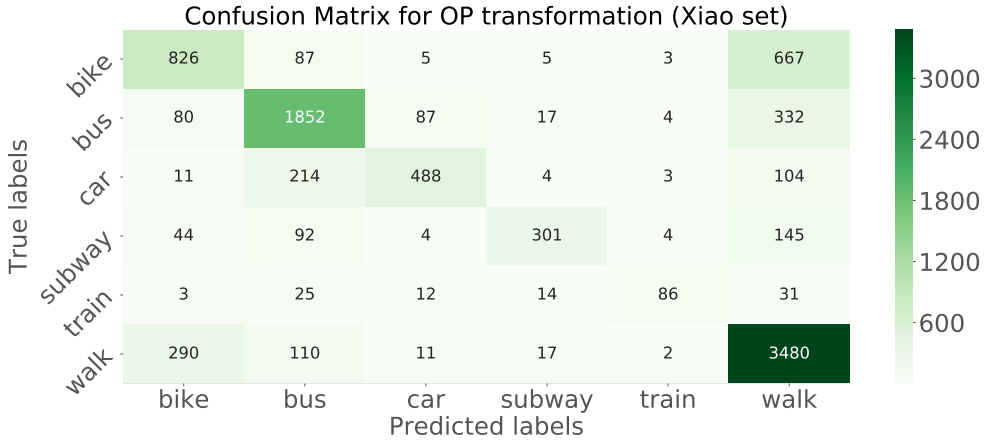
Finally, in our last evaluation, we analyze our proposal, OPAI transformation, and OP as well, compared to previous works presented in literature, which were reviewed in Chapter 2. Table 5.3 presents the accuracy results to our proposal, OP, and several other works found in literature. Only accuracy is exhibited, since it is the only metric common to all works. They are separated by the set used, hence, each work can be compared with others that use the same transportation modes to classify. In Endo and Zheng set, OPAI present the second best accuracy. Although there is a difference of about 13% in accuracy between our work and Etemad et al. (2018), it is important to have in mind that the latter uses 70 features to achieve such results, whereas OPAI uses only 5 features. We suppose that more features will enhance our results, rivaling it to the state-of-the-art. This assumption will be investigated in future. However, we still consider our results relevant in both cases, since it uses less feature and beats Deep



(a) Endo set using OP transformation

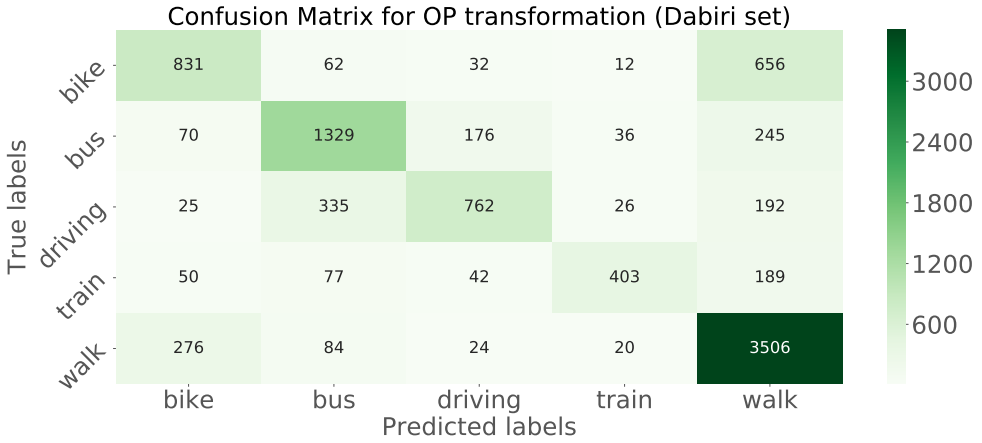


(b) Zheng set using OP transformation

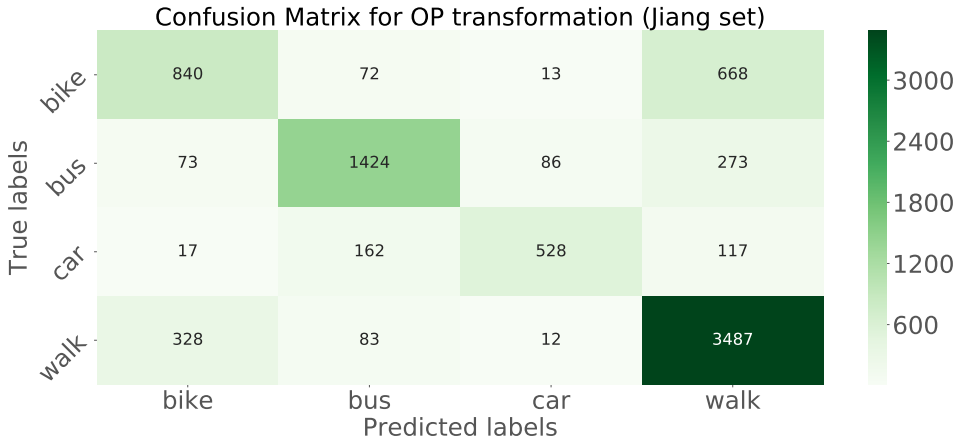


(c) Xiao set using OP transformation

Figure 5.10: Confusion Matrix using OP $(D = 5, \tau = 1)$ transformation



(d) Dabiri set using OP transformation

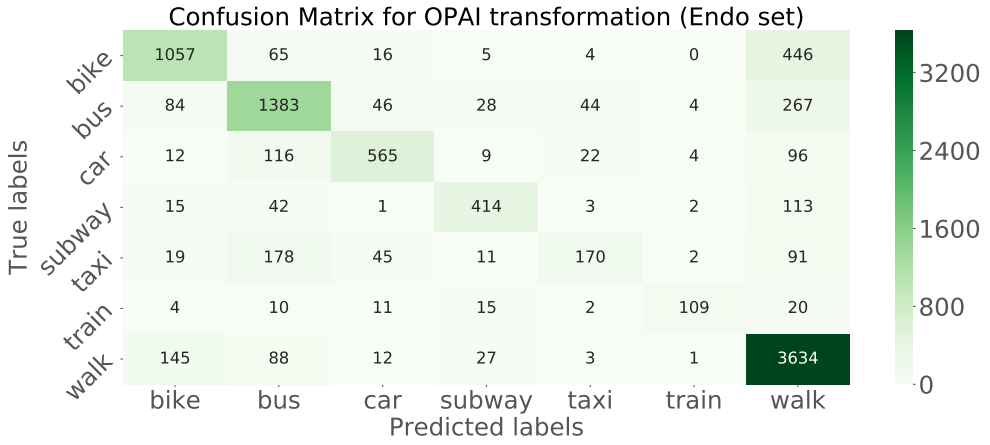


(e) Jiang set using OP transformation

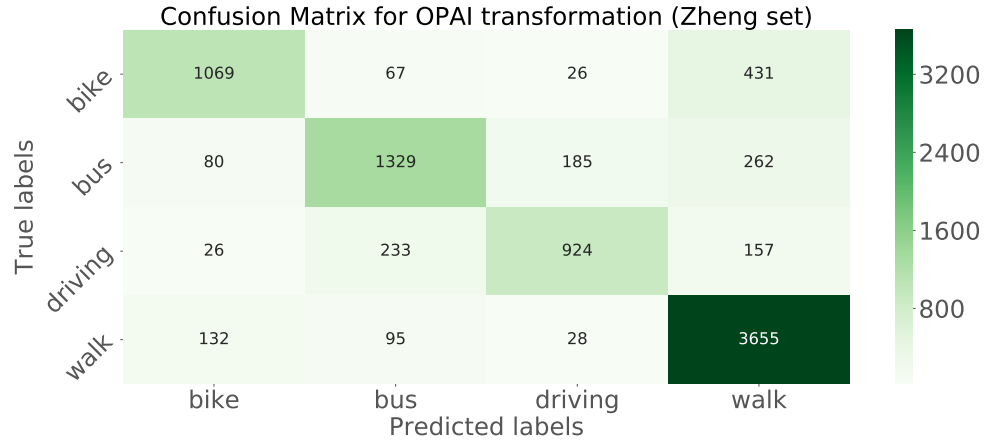
Figure 5.10: Confusion Matrix using OP ($D = 5, \tau = 1$) transformation

Learning approaches used by Endo et al. (2016), other machine learning approaches, such as Zheng et al. (2008a) and Zheng et al. (2008b), that uses more features than our proposal (10 and 13, respectively), and other our previous work (Cardoso et al., 2019), that uses features extracted from Ordinal Networks. It is important to note that Endo et al. (2016) divides their data using a different kind of cross-validation, which may influence their results.

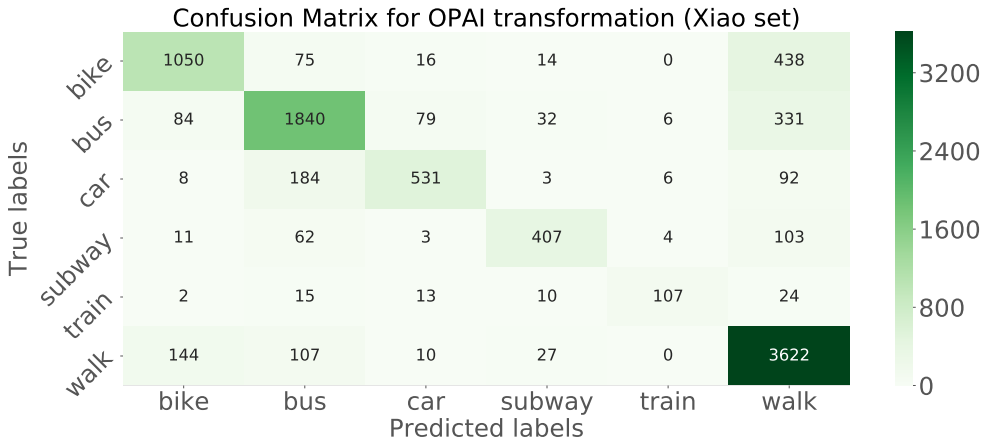
The works of Dabiri and Heaslip (2018) and Xiao et al. (2017) remove outliers based on ground truth information. As shown by Etemad (2018), this kind of noise removal can improve accuracy unrealistically. Moreover, this technique relies on already knowing the transportation mode information of test data, which is an erroneous assumption. Hence, in fact, such works cannot be adequately compared to others, they may have inflated results. Knowing this, we can see that OPAI still has accuracy



(a) Endo set using OPAI transformation

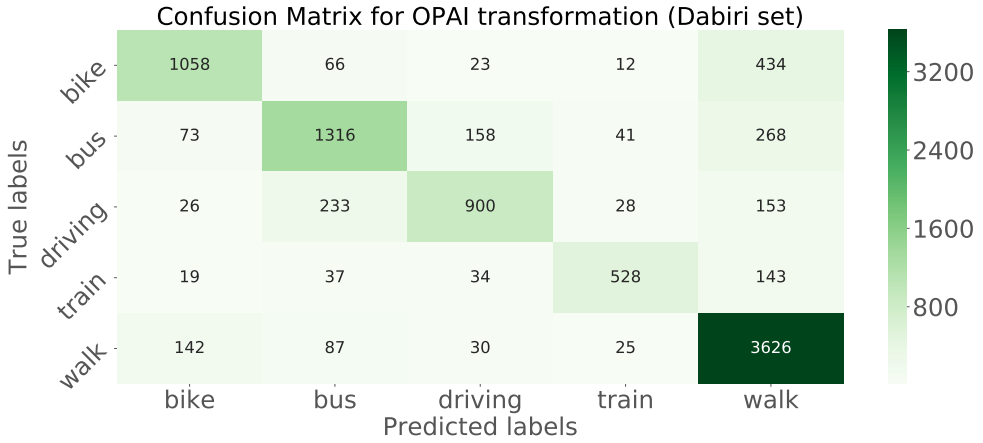


(b) Zheng set using OPAI transformation

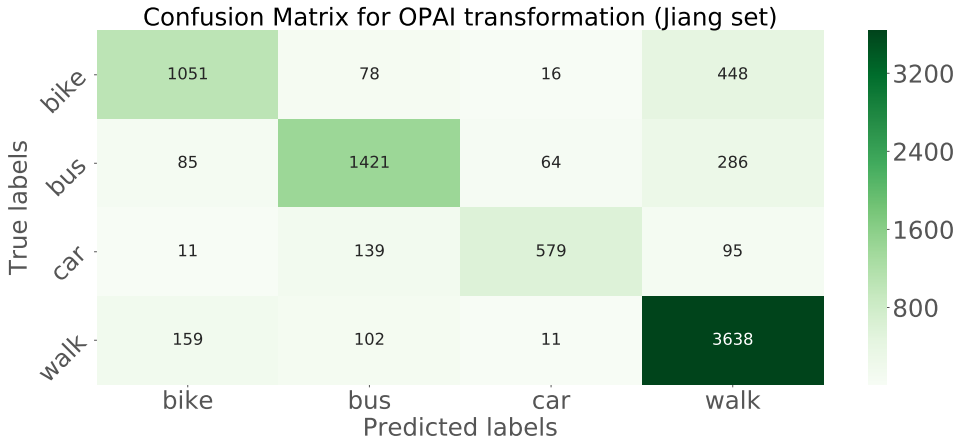


(c) Xiao set using OPAI transformation

Figure 5.11: Confusion Matrix using OPAI ($D = 3, \tau = 1, q = 16$) transformation



(d) Dabiri set using OPAI transformation



(e) Jiang set using OPAI transformation

Figure 5.11: Confusion Matrix using OPAI ($D = 3, \tau = 1, q = 16$) transformation

near Dabiri and Heaslip (2018), which use a bagging of Convolutional Neural Network (CNN) – their CNN models alone varies their accuracy from 69.2% to 79.8%. Therefore, we can see that OP and OPAI transformation can have similar results to CNN models. Furthermore, Xiao et al. (2017) use 110 features to obtain their results, whereas we use only 5. As aforementioned, we will investigate whether more features can improve our results.

Lastly, Jiang et al. (2017) use a subset to evaluate their work. It brings distrust to their result, making impossible to compare it to others.

Another interesting point about this result is that most of the studies presented in literature, such as Etemad et al. (2018); Endo et al. (2016); Xiao et al. (2017); Dabiri and Heaslip (2018); Jiang et al. (2017), perform their classification in the continuous space; differently of OPAI, as well as OP, which work in a smaller space, since they

Table 5.3: Comparing OP and OPAI transformation with other works

Set	Work	Acc (%)
Endo	Endo et al. (2016)	67.90
	Etemad et al. (2018)	90.20
	OP	71.34
	OPAI	77.51
Zheng	Zheng et al. (2008a)	74.30
	Zheng et al. (2008b)	76.20
	Etemad et al. (2018)	93.61
	Cardoso et al. (2019)	73.54
	OP	73.87
	OPAI	80.20
Dabiri	Dabiri and Heaslip (2018)	84.80
	Etemad et al. (2018)	93.55
	OP	71.94
	OPAI	78.52
Xiao	Xiao et al. (2017)	90.77
	Etemad et al. (2018)	93.19
	OP	74.41
	OPAI	79.88
Jiang	Jiang et al. (2017)	97.90
	Etemad et al. (2018)	96.45
	OP	76.99
	OPAI	81.74

transform the data to a discrete space. Hence, the amount of information available is not the same – the methods in the continuous space have more information at their disposal. Even so, our proposal can compete with them in classification results, overcoming some cases. The advantage of transform data to a smaller space is, as discussed in Section 3.2.2, the use of less space to storage and the gain of speed in classification.

5.4 Final Remarks

This chapter presented the Transportation Mode Classification proposed by this work. We evaluate how our transformation technique, OPAI, behaved when varying their and the classifier parameters. This analysis were made in comparison to OP. Our results showed that OPAI is more robust than OP to the variation of its parameters (D, τ,

and q) and achieves a better classification results in transportation mode classification, up to 10%, in average, than OP. In addition, we compared OPAI to previous literature works. Although it does not achieves the best result, OPAI presents competitive results using a much smaller set of features.

Chapter 6

Conclusion and Future Directions

In this work, we proposed a framework to Transportation Mode Classification (TMC). It is an important field of investigation, since it can help the cities to develop sustainable solutions that improves the life quality of citizens by reducing mobility issues, such as traffic jam and travel time. Therefore, it is a well-studied problem, with several solutions in many fields, such as Machine Learning, Deep Learning, Information Theory, and Complex Network.

The framework is composed of five steps: segmentation, feature extraction, data transformation, and classification. Since our focus in this dissertation is classification, segmentation and feature extraction were little explored. In segmentation, we split data based on day, time, and transportation mode used. To the feature extraction step, we extracted five motion features, namely geodesic distance, speed, acceleration, jerk, and bearing. In future works we plan to investigate deeper these two steps, by proposing segmentation methods that do not relies on transportation mode information and extracting more features. With that, we believe that our framework will be more accurate and reliable.

The data transformation step is the most important step in this work. We developed a modification to Ordinal Pattern (OP) that considers the amplitude information presented in time series data to transportation mode classification. We call it Ordinal Pattern with Amplitude Information (OPAI). Through experiments in real-world data, we demonstrated that OPAI presents better accuracy results than OP to transportation mode classification. Moreover, OPAI is more robust than OP to change in parameter values. A disadvantage of this method is that, since more patterns are extracted from time series, there is need for more storage and it is slight slower than OP – however, it still is a computationally inexpensive method, as OP, and also inherent others advantagens of OP, such as simplicity, scalability, and robustness to noise.

Although having various contributions in time series characterization, the Information Theory field lacks of contributions in Time Series Classification (TSC), including transportation mode classification. We expect that our work contribute to the development of Information Theory in TSC.

In classification, we evaluated the use of two techniques, namely Random Forest (RF) and Gradient Boosting Decision Trees (GBDT). Although both uses Decision Trees as their base classifier, they build their model in different ways – RF disposes its trees in a parallel fashion, whereas GBDT is sequential. GBDT was better to identify transportation modes, with better accuracy results.

Furthermore, there is not standardization of how to develop a investigation in TMC, making it hard to compare the works. For instance, some works use only a subset of data and others uses different kind of evaluation that may influence how the results are understood. As future work, we intend to develop a methodology that can really contrast the methods already made, similar to Bagnall et al. (2017), in order to understand the contribution of them and helps the development of the TMC.

Finally, the contributions made by this work, although being evaluated in TMC, may be taken to a wider context, that is, the time series classification field. We intend to investigate this hypothesis in future works.

Nevertheless, as future works, we also want to evaluate our contribution compared to other methods that adds amplitude information to OP distribution, in order to evaluate how different techniques with the same goal behave.

Previous to this work, in Cardoso et al. (2019), we presented some preliminary results of this dissertation, about the use of Ordinal Pattern distribution and Ordinal Networks to transportation mode classification.

Bibliography

Aghabozorgi, S., Shirkhorshidi, A. S., and Wah, T. Y. (2015). Time-series clustering-a decade review. *Information Systems*, 53:16–38.

Aquino, A., Cavalcante, T., Almeida, E., Frery, A., and Rosso, O. (2015). Characterization of vehicle behavior with information theory. *The European Physical Journal B*, 88(10):257.

Aquino, A., Ramos, H., Frery, A., Viana, L., Cavalcante, T., and Rosso, O. (2017). Characterization of electric load with information theory quantifiers. *Physica A: Statistical Mechanics and its Applications*, 465:277–284.

Azami, H. and Escudero, J. (2016). Amplitude-aware permutation entropy: Illustration in spike detection and signal segmentation. *Computer methods and programs in biomedicine*, 128:40–51.

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*, 88:174102.

Biljecki, F., Ledoux, H., and Van Oosterom, P. (2013). Transportation mode-based segmentation and classification of movement trajectories. *International Journal of Geographical Information Science*, 27(2):385–407.

Borges, J. B., Ramos, H. S., Mini, R. A., Rosso, O. A., Frery, A. C., and Loureiro, A. A. (2019). Learning and distinguishing time series dynamics via ordinal patterns transition graphs. *Applied Mathematics and Computation*, 362:124554.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Cardoso, I., Barros, P., Borges, J., Loureiro, A. A. F., and Ramos, H. (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*, pages 622–635, Porto Alegre, RS, Brasil. SBC.

Chen, Y., Keogh, E., Hu, B., Begum, N., Bagnall, A., Mueen, A., and Batista, G. (2015). The ucr time series classification archive. `www.cs.ucr.edu/~eamonn/time_series_data/`.

Dabiri, S. and Heaslip, K. (2018). Inferring transportation modes from GPS trajectories using a convolutional neural network. *Transportation research part C: emerging technologies*, 86:360–371.

Endo, Y., Toda, H., Nishida, K., and Kawanobe, A. (2016). Deep feature extraction from trajectories for transportation mode estimation. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 54–66. Springer.

Ermes, M., Pärkkä, J., Mäntyjärvi, J., and Korhonen, I. (2008). Detection of daily activities and sports with wearable sensors in controlled and uncontrolled conditions. *IEEE transactions on information technology in biomedicine*, 12(1):20–26.

Esling, P. and Agon, C. (2012). Time-series data mining. *ACM Computing Surveys*, 45(1):12.

Etemad, M. (2018). Transportation modes classification using feature engineering. *arXiv preprint arXiv:1807.10876*.

Etemad, M., Soares Júnior, A., and Matwin, S. (2018). Predicting transportation modes of GPS trajectories using feature engineering and noise removal. In Bagheri, E. and Cheung, J. C., editors, *Advances in Artificial Intelligence*, pages 259–264, Cham. Springer International Publishing.

Fadlallah, B., Chen, B., Keil, A., and Príncipe, J. (2013). Weighted-permutation entropy: A complexity measure for time series incorporating amplitude information. *Physical Review E*, 87(2):022911.

Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., and Muller, P.-A. (2019a). Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*.

Fawaz, H. I., Forestier, G., Weber, J., Idoumghar, L., and Muller, P. (2019b). Deep neural network ensembles for time series classification. *arXiv preprint arXiv:1903.06602*.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232.

Gao, Z.-K., Small, M., and Kurths, J. (2017). Complex network analysis of time series. *EPL (Europhysics Letters)*, 116(5):50001.

Giraud-Carrier, C. and Provost, F. (2005). Toward a justification of meta-learning: Is the no free lunch theorem a show-stopper. In *Proceedings of the ICML-2005 Workshop on Meta-learning*, pages 12–19.

Guo, H., Zhang, J.-Y., Zou, Y., and Guan, S.-G. (2018). Cross and joint ordinal partition transition networks for multivariate time series analysis. *Frontiers of Physics*, 13(5):130508.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hills, G. S. (2000). Year 10 interactive maths - second edition. `https://www.mathsteacher.com.au/year10/ch15_trigonometry/11_directions/23dir.htm`. Accessed on 2019-10-21.

Hofmann, M. (2006). Support vector machines-kernels and the kernel trick. *Notes*, 26.

Huang, H., Cheng, Y., and Weibel, R. (2019). Transport mode detection based on mobile phone network data: A systematic review. *Transportation Research Part C: Emerging Technologies*.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An introduction to statistical learning*, volume 112. Springer.

Jiang, X., de Souza, E. N., Pesaranghader, A., Hu, B., Silver, D. L., and Matwin, S. (2017). Trajectorynet: An embedded GPS trajectory representation for point-based classification using recurrent neural networks. In *Proceedings of the 27th Annual International Conference on Computer Science and Software Engineering*, pages 192–200. IBM Corp.

Karney, C. F. F. (2013). Algorithms for geodesics. *Journal of Geodesy*, 87(1):43–55.

Koren, Y. (2009). The Bellkor solution to the Netflix Grand Prize. *Netflix prize documentation*, 81(2009):1–10.

Lacasa, L., Luque, B., Ballesteros, F., Luque, J., and Nuño, J. C. (2008). From time series to complex networks: The visibility graph. *Proceedings of the National Academy of Sciences*, 105(13):4972–4975.

Längkvist, M., Karlsson, L., and Loutfi, A. (2014). A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11–24.

Lin, J., Khade, R., and Li, Y. (2012). Rotation-invariant similarity in time series using bag-of-patterns representation. *Journal of Intelligent Information Systems*, 39(2):287–315.

Lines, J., Taylor, S., and Bagnall, A. (2018). Time series classification with HIVE-COTE: The hierarchical vote collective of transformation-based ensembles. *ACM Transactions on Knowledge Discovery from Data*, 12(5):52:1–52:35.

Lovrić, M., Milanović, M., and Stamenković, M. (2014). Algoritmic methods for segmentation of time series: An overview. *Journal of Contemporary Economic and Business Issues*, 1(1):31–53.

Lucas, B., Shifaz, A., Pelletier, C., O'Neill, L., Zaidi, N., Goethals, B., Petitjean, F., and Webb, G. I. (2019). Proximity forest: an effective and scalable distance-based classifier for time series. *Data Mining and Knowledge Discovery*, 33(3):607–635.

Luque, B., Lacasa, L., Ballesteros, F., and Luque, J. (2009). Horizontal visibility graphs: Exact results for random time series. *Physical Review E*, 80:046103.

Parkka, J., Ermes, M., Korpipaa, P., Mantyjarvi, J., Peltola, J., and Korhonen, I. (2006). Activity classification using realistic data from wearable sensors. *IEEE Transactions on information technology in biomedicine*, 10(1):119–128.

Piotte, M. and Chabbert, M. (2009). The pragmatic theory solution to the Netflix Grand Prize. *Netflix prize documentation.*

Ratanamahatana, C. A., Lin, J., Gunopulos, D., Keogh, E., Vlachos, M., and Das, G. (2010). *Mining Time Series Data*, pages 1049–1077. Springer US, Boston, MA.

Raza, A. and Kramer, S. (2019). Accelerating pattern-based time series classification: a linear time and space string mining approach. *Knowledge and Information Systems*, pages 1–29.

Ribeiro, H. V., Jauregui, M., Zunino, L., and Lenzi, E. K. (2017). Characterizing time series via complexity-entropy curves. *Physical Review E*, 95:062106.

Rosso, O. A., Larrondo, H. A., Martin, M. T., Plastino, A., and Fuentes, M. A. (2007). Distinguishing noise from chaos. *Physical Review Letters*, 99:154102.

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.

Senin, P. and Malinchik, S. (2013). SAX-VSM: Interpretable time series classification using sax and vector space model. In *2013 IEEE 13th international conference on data mining*, pages 1175–1180. IEEE.

Shah, R. C., Wan, C.-y., Lu, H., and Nachman, L. (2014). Classifying the mode of transportation on mobile phones using GIS information. In *Proceedings of the 2014 ACM international joint conference on pervasive and ubiquitous computing*, pages 225–229. ACM.

Shifaz, A., Pelletier, C., Petitjean, F., and Webb, G. I. (2019). TS-CHIEF: A scalable and accurate forest algorithm for time series classification. *arXiv preprint arXiv:1906.10329*.

Small, M. (2013). Complex networks from time series: Capturing dynamics. In *2013 IEEE International Symposium on Circuits and Systems*, pages 2509–2512. IEEE.

Sokolova, M. and Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information processing & management*, 45(4):427–437.

Staniek, M. and Lehnertz, K. (2007). Parameter selection for permutation entropy measurements. *International Journal of Bifurcation and Chaos*, 17(10):3729–3733.

Sun, X., Small, M., Zhao, Y., and Xue, X. (2014). Characterizing system dynamics with a weighted and directed network constructed from time series data. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 24(2):024402.

Tan, P.-N., Steinbach, M., and Kumar, V. (2005). *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA. ISBN 0321321367.

Töscher, A., Jahrer, M., and Bell, R. M. (2009). The bigchaos solution to the Netflix Grand Prize. *Netflix prize documentation*, pages 1–52.

United Nations, D. o. E. and Social Affairs, P. D. (2018). World urbanization prospects: The 2018 revision (key facts). Technical report.

Wang, X., Ding, H., Trajcevski, G., Scheuermann, P., and Keogh, E. J. (2010). Experimental comparison of representation methods and distance measures for time series data. *CoRR*, abs/1012.2789.

Wilson, S. J. (2017). Data representation for time series data mining: time domain approaches. *Wiley Interdisciplinary Reviews: Computational Statistics*, 9(1):e1392.

Xiao, Z., Wang, Y., Fu, K., and Wu, F. (2017). Identifying different transportation modes from trajectory data using tree-based ensemble classifiers. *ISPRS International Journal of Geo-Information*, 6(2):57.

Yang, Q. and Wu, X. (2006). 10 challenging problems in data mining research. *International Journal of Information Technology & Decision Making*, 5(04):597–604.

Yang, X., Stewart, K., Tang, L., Xie, Z., and Li, Q. (2018). A review of GPS trajectories classification based on transportation mode. *Sensors*, 18(11):3741.

Zaki, M. J., Meira Jr, W., and Meira, W. (2014). *Data mining and analysis: fundamental concepts and algorithms*. Cambridge University Press.

Zanin, M., Zunino, L., Rosso, O. A., and Papo, D. (2012). Permutation entropy and its main biomedical and econophysics applications: a review. *Entropy*, 14(8):1553–1577.

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, L., Liu, L., Bao, S., Qiang, M., and Zou, X. (2015). Transportation mode detection based on permutation entropy and extreme learning machine. *Mathematical Problems in Engineering*, 2015.

Zheng, Y. (2015). Trajectory data mining: an overview. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(3):29.

Zheng, Y., Li, Q., Chen, Y., Xie, X., and Ma, W.-Y. (2008a). Understanding mobility based on GPS data. In *Proceedings of the 10th international conference on Ubiquitous computing*, pages 312–321. ACM.

Zheng, Y., Liu, L., Wang, L., and Xie, X. (2008b). Learning transportation mode from raw GPS data for geographic applications on the web. In *Proceedings of the 17th international conference on World Wide Web*, pages 247–256. ACM.