eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Konputazio Zientziak eta Adimen Artifizialaren Saila
Departamento de Ciencias de la Computación e Inteligencia Artificial

Contributions to time series data mining departing from the problem of road travel time modeling

by

Usue Mori Carrascal

Supervised by Alexander Mendiburu and Jose A. Lozano

Dissertation submitted to the Department of Computer Science and Artificial Intelligence of the University of the Basque Country (UPV/EHU) as partial fulfilment of the requirements for the PhD degree in Computer Science

Donostia - San Sebastián, October 2015

To my family.

Acknowledgments

The present dissertation would not have been possible without the guidance, support and advice of my advisors Jose A. Lozano and Alexander Mendiburu. Thank you for the great working atmosphere you have created, for encouraging and cheering me up in the difficult moments and, of course, for all the free coffees.

I must also mention all the members of the Intelligent Systems Group for all the help they have provided, coffee breaks, and discussions about research and other issues; it has all contributed to an easier and more agreeable journey to this point. Special thanks to Borja Calvo for the idea and design of the cover and for letting me borrow one of his photographs.

Next, acknowledgements to all the members of the LAMBDA group, lead by Prof. Zhou, from the University of Nanjing, and thanks to Prof. Dasgupta and all the people from the Computer Science and Engineering Department at the University of California, San Diego, for hosting me during my respective visits. I strongly believe that both stays have widened my knowledge and view of research in machine learning. Also, thanks to Prof. Keogh and his group from the University of California, Riverside for being so welcoming and for all the helpful comments and insights, which have helped me improve my work.

I am also grateful to the Computer Science and Artificial Intelligence Department and especially to Yosu Yurramendi for the support at the beginning of this process. Also thanks to the University of the Basque Country UPV/EHU itself for providing the funding that has made this dissertation possible, and to the Saiotek and IT-609-13 programs (Basque Government), TIN2013-41272P (Spanish Ministry of Science and Innovation) and the NICaiA Project PIRSES-GA-2009-247619 (European Commission) for partially supporting this dissertation.

I am also much obliged to Tecnalia Research & Innovation and The Iñaki Goenaga Foundation for funding the first period of the realization of this thesis dissertation. From Tecnalia I must mention the support received from Maite Álvarez, Ana Torre, Adrián Galdrán and Aitor Álvarez.

A big thank you to my friends, for bearing all my moaning, grumbling and whining during this period with good humor, and for making me laugh, disconnect and recharge batteries. Special thanks to Agus, who is always there, no matter what. *Mila esker kuadrila!*

Last but not least I would like to thank my family, who has been an inspiration and the main reason why I began and have finished this dissertation. Aita, *milesker umore eta lasaitasunagatik, momentu zailetan irribarrea ateratzen lagundu didatenak.* Ama *alabengatik hainbeste kezkatu eta lan egiteagatik, laguntza etengabeagatik, mila aholku erabilgarrigatik eta, finean, beti gugan hainbeste sinisteagatik.* Libe, *nik bukatu berri dudan bidai hau zu hastear zaude orain, munduko zorte onena opa dizut eta nahi duzun guztia lortzeko gai zarela ez ahaztu!.* Lertxun, *ia niri bezain beste kostatu zaizu tesi hau eta mila esker eguneroko animo eta goxotasunagatik, zuri esker ere bada hau.*

Contents

Reserve your right to think, for even to think wrongly is better than not to think at all.

Hypatia of Alexandria.

1

Introduction

An Advanced Traveler Information System (ATIS) collects, processes and presents road traffic information to the users, assisting them in their trips and supporting them in making the necessary *pre-trip* and *en-route* decisions. With this purpose in mind, traffic modeling becomes one of the most important tasks of ATIS, because it enables the description, simulation and forecasting of the traffic variables of interest. Among all the traffic variables that can be modeled (flow, road occupancy, speed, travel time, etc.), travel time acquires a special relevance in ATIS, because the concept can be easily understood by travelers. As such, travel time estimation and prediction are two of the most common traffic modeling problems ATIS has to deal with.

The importance of these two modeling problems has raised the interest of the research community in the past few years, and has thus resulted in a vast number of publications and model proposals. However, a detailed analysis and review of the state-of-the-art shows that not all the proposed models are adequate for all the study sites, traffic situations, available data, etc. In this context, the combination or fusion of models seems to be one of the most promising research lines, because it allows the use of specific and more suitable models for each case.

A specific type of combined or hybrid travel time models are those that initially pre-process the data using clustering algorithms, with the aim of identifying and separating the different traffic patterns that may be present. Then, a separate and suitable travel time model is built for each cluster, allowing the construction of more specific models for each case. Particularly, a special case of these combination models relies on the paradigm of time series clustering, in which each instance to be clustered is a whole time series, for example, the sequence of travel time measurements collected throughout a day.

Since the presence of diverse daily traffic patterns is evident in most cases, time series clustering seems to be a logical and promising approach. Nevertheless, most authors forget about the long-term patterns in the data, and somehow segment and pre-process the data to enable the application of con-

ventional clustering schemes. In this context, the benefits and disadvantages that arise when using time series clustering within travel time models have not been empirically studied in the literature, to the best of our knowledge. Moreover, in practice, building and applying these combination models based on time series clustering entails some difficulties.

To begin with, clustering a time series dataset requires making some non-trivial decisions, such as selecting a suitable distance measure. There are innumerable distance measures specifically designed for time series data, and it has been demonstrated previously in the literature that there is no unique distance measure that is suitable for all the databases. Indeed, it seems that the specific characteristics of each database have a strong impact on the performance of the different existing measures. In this context, an interesting question that will be addressed in this dissertation is whether, given a set of characteristics of the database, it is possible to automatically select a suitable distance measure(s) from a set of candidates.

Finally, we must not forget that the final objective of combined travel time models is usually to predict or estimate travel times. As such, recall that after applying the clustering process, a separate model is built for each cluster. If we focus on the task of prediction, the aim is to provide travel time forecasts for future trips by using these models. However, in order to do this, we must decide which model to use in each case by assigning the new incoming data sequence to a given cluster or traffic pattern. This must be done online, using only the data collected until the time the prediction is made. Of course, if the data available at the time of prediction is not informative enough to choose a given cluster, it might be risky to lean towards a model trained for a very specific traffic behavior, and it is probably better to apply a more general model.

In short, it is desirable to assign the incoming sequences to a given cluster or traffic pattern as soon as possible while maintaining some degree of accuracy in these assignments. This problem can be understood as a problem of early classification of time series, and in this dissertation we will provide a general method to solve this task, focusing on improving some of the flaws of the existing proposals.

Finally, returning to the problem of travel time modeling, the application of these two contributions (the automatic distance measure selector and the early classifier) to this specific problem is interesting because it allows the identification of specific features of this problem that may motivate modifications or adaptations on the proposed methods.

In summary, the following dissertation will depart from the problem of travel time modeling which will give way to two main contributions on time series mining: the selection of a suitable distance measure when clustering time series databases and the early classification of time series based on probabilistic models. Then, we will return to the problem of travel time modeling to analyze the performance of our proposals within this specific problem.

1.1 Outlook of the dissertation

This dissertation is divided into four parts. In Part I we introduce some basic concepts and definitions that will be used throughout the dissertation. This part is divided into two separate chapters; Chapter 2 focuses on providing a general introduction to the problems of travel time estimation and prediction, and Chapter 3 is devoted to the basics of time series mining and time series distance measures. Next, in Part II, we analyze the literature and provide a complete review and taxonomy of the state-of-the-art on travel time estimation and prediction, analyzing the virtues and flaws of the existing proposals. Departing from the conclusions obtained from this analysis of the state-of-the-art, in Part III we focus on combined travel time prediction models that incorporate time series clustering pre-processes. After demonstrating that time series clustering can be beneficial for travel time prediction in Chapter 5, we focus on the two main methodological contributions to time series data mining, which will be useful when building the aforementioned combined models. First, in Chapter 6, we provide a method to automaticaly select the most suitable distance measure(s) to cluster a time series database from a set of candidates. Then, in Chapter 7 we center our attention on solving the problem of early classification of series, which will be necessary to assign new time series to a given cluster. Once the two contributions on time series mining have been introduced, we return to the problem of travel time prediction and, in Chapter 8, we analyze the performance of these two proposals (automatic distance selection and early classification) within this particular problem. Finally, in Part IV, we present the main conclusions of this dissertation as well as some possible future research lines and the publications that have resulted from this work.

Part I

Preliminaries

2

Travel time estimation and prediction

In recent years, technological advances have enabled the collection and dissemination of real-time traffic information and this, combined with growing traffic volume and congestion, has triggered an increasing interest in traffic modeling [58]. These models and algorithms are the baseline for two types of systems: Advanced Traffic Management Systems (ATMS) and Advanced Traveller Information Systems (ATIS). ATMS are generally used by traffic engineers and administrators in order to enhance mobility and obtain a more efficient and safe traffic in road networks. On the contrary, ATIS are aimed at providing commuters with the necessary traffic information and tools to enable decision making [220].

While variables such as flow, occupancy and speed are very common and useful in ATMS, modeling travel time is more popular in ATIS, because it is a very intuitive concept and can be easily understood by travelers [217]. Travel time is defined as the total time for a vehicle to travel from one point to another over a specified route, taking into account the stops, queuing delay and intersection delay [255]. Given its utility, the modeling of this traffic variable is a recurrent research topic.

As commented in the introduction, travel time modeling is the practical application from which we depart to come up with all the contributions presented in this dissertation. As such, as a first step, it is necessary to introduce some basic aspects of travel time modeling, and also to properly define two of the most important travel time modeling problems on which we will focus in this dissertation: travel time estimation and travel time prediction.

2.1 Basic aspects of travel time modeling

Before focusing on the specific travel time modeling problems, it is crucial to have some knowledge on two basic aspects: the variety of **data sources** that can be used in the modeling process and the different types of **areas of the traffic network** that are usually studied in the literature.

2.1.1 Data sources

Traffic data sources can be classified in several ways but in this case, since the target of this review is travel time, the classification presented by Wu et al. [232] and Lim and Lee [125] has been chosen, which groups the sensors based on their ability to directly obtain travel time measurements. In this manner, traffic sensors are divided into point and interval detectors.

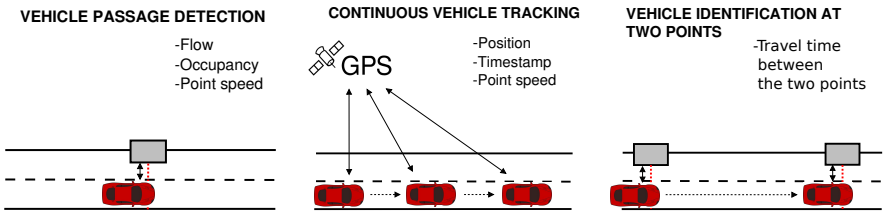
2.1.1.1 Point detectors

As can be seen in Figure 2.1a, this type of detector is set in fixed points of the road and captures traffic variables in these specific points.

The most conventional point detectors are the inductive loops that can be further categorized into single and double loop detectors [37, 220, 243]. On the one hand, **single loop detectors** consist of a single induction loop that generates a magnetic field and is able to detect the passing of large metallic objects, in this case vehicles. These detectors output variables such as flow (number of passing vehicles/hour) and occupancy (% of the time that the detector is occupied). On the other hand, **double loop detectors** consist of a pair of single loop detectors set very close to each other. This pair of sensors is capable of obtaining flow and occupancy but they can also collect speed and vehicle lengths, by using the travel time of the vehicles between the two sensors [114].

The speed captured by double loop detectors is called point speed and is generally only valid to describe the speed at the point where the sensor is situated [190], as can be seen in Figure 2.1a. In practice, it is very common that this velocity information is provided in an aggregated form, where the measurements of several vehicles are combined in different forms. The most simple and common aggregation method is denominated *time mean speed* and, in this case, the point speeds of vehicles that cross the detector in the same discrete time interval are averaged using the arithmetic mean [211]. In addition, some other aggregation methods ranging from the harmonic mean to more complex probabilistic models [190], aim to approximate the *space mean speed*, which will be properly defined in the following sections and is a speed measure which is more adequate for describing a whole road section [79]. However, as good as these approximations may be in some cases, it must not be forgotten that the data is captured in a single point and, therefore, the results are generally only reliable for that spot, especially in congested or varying state situations, where the vehicle speed does not remain constant [190].

Furthermore, these sensors provide accurate data, are not affected by external factors and are widely deployed in the roadways, but their installation and maintenance is expensive and complicated [7]. For this reason, in recent years, other solutions such as **video image detection methods** are gaining approval because of their low installation cost and high accuracy. These



(a) Point detectors. (b) Probe vehicles. (c) AVI detectors.

Fig. 2.1. Traffic data sources.

detectors use video cameras and image processing methods to obtain vehicle counts and speeds at specific points of the road. Their main drawback is that they are usually susceptible to external factors such as the weather, and that they need periodic maintenance [114].

2.1.1.2 Interval detectors

Interval detectors capture data that enables the direct calculation of travel time between two points, as opposed to point detectors that are only able to describe a single point of the road. This type of detector can be further divided into two main groups [232]: floating or probe vehicles and, automatic vehicle identification techniques (AVI) both of which are represented in Figures 2.1b and 2.1c, respectively.

As can be seen, **floating and probe vehicles** are a sample of vehicles which circulate in the traffic network and provide information about their trajectories. The main difference between them is that floating vehicles are specifically employed for data collection purposes, whereas probe vehicles are passive vehicles that travel in the road network for other reasons [211]. They are both generally equipped with cell-phones and/or Global Positioning Systems (GPS) and send location, direction and speed information every few seconds [114]. This assures an accurate representation of the trajectory of the vehicles, and travel times between two points can be derived easily and reliably [7]. For more information on different cellular positioning techniques and trials using probe vehicles, the interested reader can access [246].

In contrast, **AVI systems** can be of various types, from manual surveys [159] to automatic toll collection systems [60], vehicle mounted transponders of different types and roadside beacons [130], video cameras and license plate matching techniques [218] or the more recent bluetooth [11] and WIFI [1] based detection systems. These devices, detect and identify vehicles only at the beginning and at the end of the study segment and calculate the travel time directly from this data [130, 211].

Interval detectors guarantee high accuracy and quality description of the traffic situation [125]. However, there are some practical inconveniences when using them.

For a start, many interval detectors are not able to detect all the vehicles in the road and so, the representativeness of the detected vehicle sample is an issue that must be considered [130]. The sample size needed for accurate representation of traffic is generally quite large and difficult to obtain in real situations [29, 33, 99, 166, 186, 196, 212], especially in the case of probe vehicles [114].

Furthermore, some interval detectors such as GPS enabled probe vehicles provide irregularly spaced or intermittent data [242]. Modeling this type of data is more complex due to the uncertainty and lack of information associated to the irregularity of the sampling. Because of this, most travel time models are aimed at regularly spaced data.

Another issue associated with interval detectors is privacy [114, 211, 246]. Data collection using interval detectors frequently requires the individual vehicles to provide their identification, which might reveal sensitive and private information [90]. Although the methodologies to encrypt the data will not be studied within the scope of this dissertation, they should be considered when using interval detectors, because they ensure preservation of privacy, support the user's trust and enable larger and more representative vehicle samples.

Finally, the main reason why, in the literature, point detectors are more frequently used than interval detectors is that point detectors are already installed in many roads and highways, whereas intervals detectors are still not present in most of our road networks. In any case, in the past few years, the popularity of interval detectors has increased considerably with the popularization of smart phones and their presence in road networks is expected to grow substantially in the near future.

2.1.2 Study site

Each area of the traffic network has specific characteristics that determine the behavior of the traffic on it. Based on these features, we identify two main types of study areas: freeways or highways and urban or arterial roads.

Most travel time models in the literature are set in **freeway or highway segments** where traffic is generally uninterrupted. The main reason is that the acquisition of data and the construction of the model for these roads is simpler than for other road types [224].

On the contrary, research on **urban and arterial road segments** is not so common because traffic sensors are not always available in these sites. Moreover, traffic in urban sites is more complex and the modeling process varies because factors such as signal and intersection delays must be taken into account [239]. In these cases, some authors attempt to model travel time by modeling the cruising time and the delays separately and then summing both components [32, 89, 98, 127]. Furthermore, since the traffic is more variable

in urban contexts due to traffic lights, queues, bus stops, etc. more elaborate models must be sought that take this variability into consideration and are able to provide reliable and useful travel time values [253].

In any case, in both highway or urban environments, since travel time depends on the origin and destination, and given the huge number of possible combinations of origins and destinations in a road network, ATIS normally use methods that calculate travel time information at a link or section level [33]. A section can be defined as the distance between two intersections in an urban environment, the distance between two entry and exit ramps in a highway or, generally, the distance between two detectors. In this manner, most authors concentrate in separately modeling a limited number of links and then obtaining the travel time of longer trajectories by summing the travel times of the links or sections that constitute it.

2.2 Travel time estimation and prediction: definitions and differences

Although the concepts estimation and prediction are sometimes vague in the literature and even used equivalently by some authors, they are two different modeling problems with different objectives and characteristics.
As can be seen in Figure 2.2, **travel time estimation** consists in reconstructing travel times of trips completed in the past based on data collected during the trip [13]. Generally, the authors do not focus on individual travel times but aim to provide a travel time value that will give a general idea of the traffic situation in a certain road section and a certain time frame.

In this line, although providing an estimation of the travel time distribution is more informative and reliable, the most common objective in the literature is to provide some sort of mean travel time for a given road section R. The definitions of *true mean travel time* are various in the literature but, the most common can be theoretically reduced to the following framework [150]:

$$TT_R = \frac{L}{\bar{v}_{space}} \tag{2.1}$$

where L is the length of the study section and $\bar{v}_{space}$ is the *space mean speed*. Space mean speed has been defined in several ways in the literature, but two main groups of definitions can be distinguished [79]. The first group of authors define the space mean speed as the division between the total distance travelled by a set of vehicles inside a road section and their total time of travel. The second group defines space mean speed as the mean speed of all vehicles in a road section in a given instant of time. By applying this definition in consecutive instants of time an average speed value that is representative of the whole section can be obtained. In spite of the various definitions, all authors agree that in order to describe a whole road section and obtain travel

time values, space mean speed must be used. Furthermore, since L is generally well known, the modeling of space mean speed and travel time in this case are essentially equivalent and therefore, studies referring to both of them will be taken into account and treated equally in the rest of the dissertation.

Based on this framework and on the different definitions of space mean speed, we distinguish the two main theoretical interpretations of true mean travel time that coexist in the literature. On the one hand, some authors focus on obtaining the mean travel time of all the vehicles departing in route R in a given time interval t of length Δ. On the other hand, some others concentrate on calculating the mean travel time of all the vehicles that travel in section R during time interval t by also including the vehicles that were already inside the section when the time interval started and the ones that do not finish traversing the section. Depending on the data available and the interests at hand one definition or another are used by the researchers.

The main problem with these definitions is that in real world problems, when only data from traffic sensors is available, it is generally not possible to calculate them directly. As can seen in Figure 2.1a, point detectors are only able to capture data in specific locations of the study segment. Since the space mean speed and travel time are section wide variables, if traffic conditions

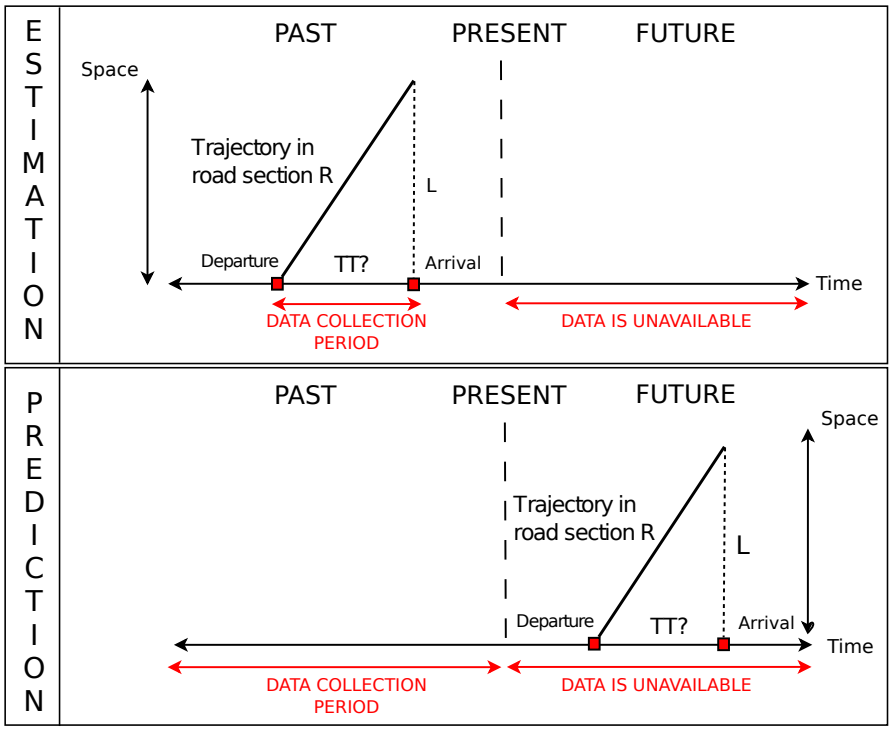


Fig. 2.2. Difference between travel time estimation and prediction

are not constant, the data from point detectors is not sufficient to calculate mean travel time easily and accurately. Among the interval detectors the main problem is that, in real situations, it is not always possible to track all the vehicles, or even a sufficient sample, which is mandatory to reliably calculate Equation 2.1. Another typical problem, that affects all types of detectors and must be dealt with in order to obtain reliable travel time values, is the presence of noisy, missing or bad quality data.

So, since the mean travel time can not generally be calculated empirically, approximation or estimation schemes will be necessary to obtain travel time values that will be used to extract statistics on the performance of new traffic measures [221], to update information of variable message signs and speed control systems [26] and as a baseline to calculate input data [215] and validation data [220] for more complex travel time prediction algorithms.

As opposed to estimation methods, the objective of the **travel time prediction** models is to forecast the travel time for a trajectory that will start in the moment the prediction is made (present) or in the future. For this purpose, traffic and contextual data available in the present together with data from the past will be used (See Figure 2.2).

Lately, the need for traffic predictions has become indispensable due to the increasing congestion in the road networks. In this line, Wu et al. [232] state that traffic prediction is beneficial for ATIS because it provides the necessary *pre-route* and *en-route* information to schedule and choose the most adequate routes in each situation. Moreover, in [227], it has been proven that the frequent access to diverse travel information has a notable impact on the citizens travel decisions, leading to a more efficient use of the traffic network.

As with estimation of travel time, most authors concentrate on the prediction of mean travel time as opposed to individual travel times. A few more recent cases provide confidence intervals [106, 120, 218] or probability distributions [65, 88, 93, 253] for the mean travel time but, in general, the authors focus on obtaining a single mean value. Taking this into account, a typical travel time predictive model can be formulated as:

$$TT_R(t+\Delta) = f_\Delta(D(t), D(t-1), ..., D(1)) \quad \Delta = 0, 1, 2, ... \tag{2.2}$$

where, t is the time interval when the prediction is made (present) and $TT_R(t+\Delta)$ is the travel time on link R of vehicles departing at time interval $t+\Delta$. $\{D(t), D(t-1), ..., D(1)\}$ is the set of data collected from the first time interval for which there is available data until the time interval the prediction is made (t). It must be noted that this framework includes real time information and historical data, which are both necessary to perform accurate predictions. f is, as in the previous case, the function that relates the explicative variables with the target variable.

Δ is called the *prediction horizon* and, in general, the authors have concentrated on short-term prediction, which considers the predictions up to one hour in the future. The cases where travel time is predicted for further than

one hour in the future are few in the literature [134, 187] because it is difficult for these type of predictions to be robust enough to be used in ATIS and mostly commercial web application have focused on this task. However, Simroth and Zähle [187] suggest that long term predictions are necessary to enable route selection in long trajectories or entire networks and, therefore, they should be further elaborated in the future. In relation to this, it is important to mention that the relevance of real time information degrades as the prediction horizon increases and the utility of historical data becomes more and more useful [184].

3

Time series data mining

In recent years, the increase in data collecting devices has generated a new type of database where each instance consists of an entire time series. The main characteristics of this type of data are its high dimensionality, its dynamism, its auto-correlation and its noisy nature [67], all of which complicate the study and pattern extraction to a large extent. In view of this, many researchers have focused on finding new methods of analysis and on adapting the existing data mining algorithms to obtain useful information from these databases. This has resulted in the creation of a separate area of research denominated time series data mining.

As stated in the introduction, in this dissertation we will work on two contributions to time series data mining departing from the problem of travel time modeling. But, for this, it is important to first introduce some essential aspects about time series data mining. As such, in this chapter we present some basic concepts and notations regarding time series data mining, and we introduce the problems of time series clustering and classification, which will appear throughout the dissertation, defining them in detail and providing some natural application scenarios.

Additionally, many time series data mining tasks require the selection of a measure that will quantify the similarity or dissimilarity between time series. Accordingly, in the past few years a vast number of distance measures specific for time series have been proposed by the research community. In this chapter, we introduce a set of popular time series similarity measures that will be used in the different contributions contained in this document.

3.1 Time series data mining

With the proliferation of data collecting devices and sensors, much of the data that is collected nowadays is of temporal nature. However, as shown in the following definitions, there are different types of temporal data:

Definition 1. *A time series is an ordered sequence of pairs (timestamp, value) of fixed length N:*

$$TS = \{(t_i, x_i),\ \ i = 1, ..., N\} \tag{3.1}$$

where we assume that the timestamps (t_i) take positive and ascending real values. The values of the time series (x_i) may be univariate or multivariate and can take real or discrete values.

In this dissertation we will mainly focus on univariate time series that take real values.

Definition 2. *A data stream is an ordered sequences of pairs (timestamp, value) of unknown and possibly infinite length:*

$$S = \{(t_i, x_i),\ \ i = 1, 2, 3, ...\} \tag{3.2}$$

where, once again, we assume that the timestamps (t_i) take positive and ascending real values.

Definition 3. *A database of time series is an unordered collection of time series.*

The time series contained in a time series database can all be of the same length, or can have different lengths.

As with other types of data, researchers have focused on trying to extract useful information from all these types of temporal data, and this has resulted in a specific area of research denominated time series mining. Although the most popular objective in time series data mining is forecasting future values of time series, other tasks such as time series representation, query by content, classification, clustering, segmentation, etc. have also become very popular in the past few years [64, 67]. In this dissertation, we will mainly focus on time series clustering and classification.

3.1.1 Time series classification

Time series classification (see Figure 3.1) is a supervised data mining task where, given a training set of time series $\mathbf{TS} = \{TS_1, TS_2, ..., TS_n\}$ and their respective class labels $\mathbf{C} = \{C_1, C_2, ..., C_n\}$, the objective is to build a classifier that is able to predict the class label of any new time series as accurately as possible [236, 237].

There are many application scenarios that naturally adapt to the supervised classification of time series. Some examples are using electrocardiography (ECG) data to predict if a patient has heart disease or not, or detecting money laundering by analyzing transaction sequences in a bank [236].

Based on the utility of the task, there have been many proposals in the literature to solve it, and common classifiers such as support vector machines,

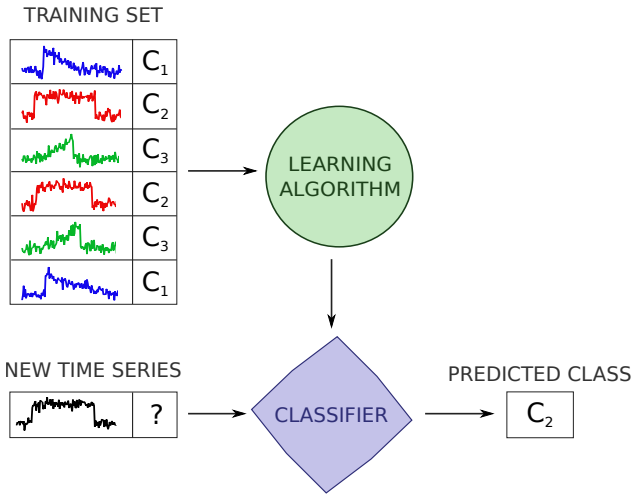


Fig. 3.1. Time series classification.

neural networks, Bayesian classifiers or decision trees have been applied [64]. However, the basic 1NN classifier combined with some specific time series distance measures, such as Dynamic Time Warping, which will be introduced in the following section, has proved to be particularly difficult to beat [228].

3.1.2 Time series clustering

Time series clustering (see Figure 3.2) is an unsupervised data mining task where, given a dataset of time series $\mathbf{TS} = \{TS_1, TS_2, ..., TS_n\}$, the objective is to find homogeneous and natural underlying groups in the dataset. The aim is thus to divide the dataset into a set of groups that maximize the within-group object similarity and minimize the between-group object similarity [64].

This time series data mining task is useful when no labeled data is available. As such, it has been previously used for market segmentation in energy

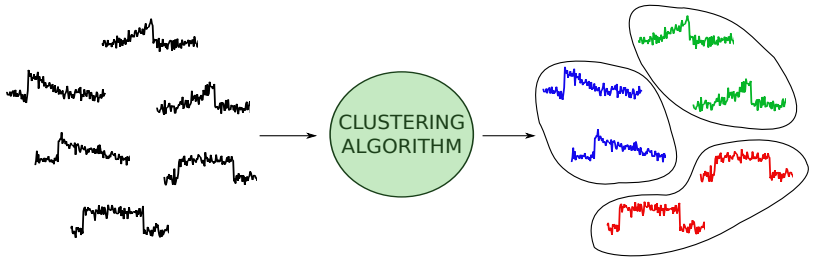


Fig. 3.2. Time series clustering.

consumption series, for grouping temperature series, to find patterns in gene representation databases, and in many other application scenarios [124].

As stated by Liao [124], there are three crucial aspects to be taken into account when clustering a time series database. First, time series clustering typically requires the definition of a distance measure that will quantify the similarity between the different time series. The selection of a suitable distance measure is a critical step, which will be studied more in depth in Chapter 6. Secondly, a clustering algorithm must be selected. In essence, any common clustering algorithm could be used for time series clustering, but the specific characteristics of the data should be taken into account when choosing one over another. Finally, an adequate evaluation measure will enable us to choose between different clustering solutions, and should be selected depending on the information and objective at hand [225].

3.2 Time series distance measures

As mentioned initially, many time series mining tasks, such as clustering and classification, typically require the definition of a distance measure. Based on the particular characteristics of temporal data, in the past few years, the scientific community has published a vast portfolio of specific distance measures to work with time series [228].

As can be seen in Figure 3.3, these distance measures can be divided into 5 distinct groups based on [64] and the vignette of the `TSclust` package of `R` [141]. *Shape-based distances* are based on comparing the raw values of the series, *edit-based distances* are adaptations of the edit distance to numerical series and *feature-based distances* are based on comparing certain features extracted from the series. Next, *structure-based distances* include (i) model-based approaches, where a model is fit to each series and then the comparison is made between models, and (ii) complexity-based approaches, where the similarity between two series is measured based on the quantity of shared information. Finally, *prediction-based distances* are based on the similarity of the forecasts obtained from different time series.

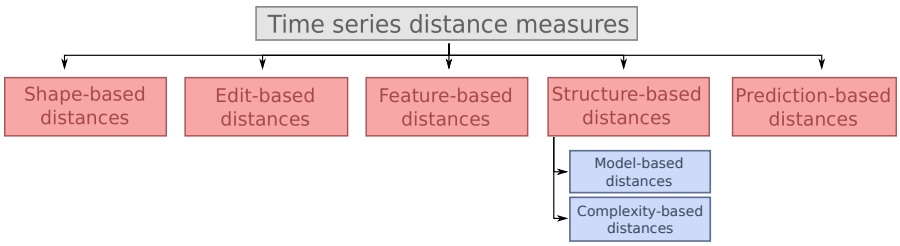


Fig. 3.3. Taxonomy of time series distance measures.

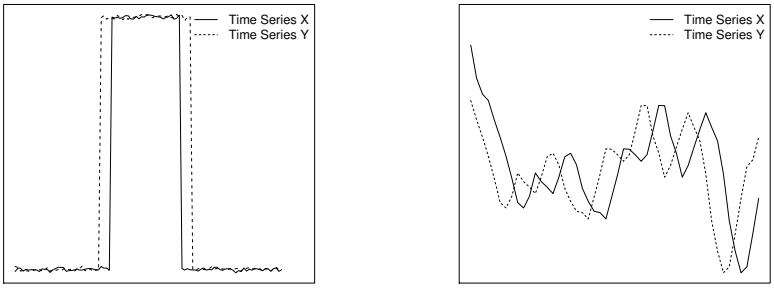
In this document we will explain five distance measures in more detail, because they will be used in the different contributions of the dissertation. These distance measures have been chosen due to their different characteristics and because they appear frequently in recent reviews. The interested reader can find additional information about other measures in [64, 124, 228]. Moreover, the implementation of over 20 distance measures is made available through the `TSdist` package [142], created as part of this thesis, and which is explained in more detail in Appendix B.

3.2.1 Euclidean distance

Euclidean distance (ED) is a shape based measure and one of the most common measures applied in data mining. Consequently, many researchers have directly extended its use to time series. As with other types of data, given two time series $X = \{x_0, x_1, ..., x_{N-1}\}$ and $Y = \{y_0, y_1, ..., y_{N-1}\}$, the ED between them is calculated as follows:

$$ED(X, Y) = \sqrt{\sum_{i=0}^{N-1} (x_i - y_i)^2} \tag{3.3}$$

Although ED has been widely used in many application fields, various researchers have pointed out that it is not always an adequate measure for time series [228]. First, it is only able to deal with series of equal length. Moreover, ED is highly susceptible to noise and outliers, which are common in temporal sequences. Finally, it is based on the comparison between points collected at the same time interval and, as shown in Figure 3.4, time series frequently suffer transformations in the time axis while still maintaining a similar shape.



(a) Locally warped time series. (b) Shifted time series.

Fig. 3.4. Temporal transformations in time series.

3.2.2 Dynamic Time Warping

In order to overcome the inconveniences of rigid distances such as ED, many other shape based similarity measures have been specifically designed for time series data. Among them the most popular is probably Dynamic Time Warping (DTW) [10]. This distance is able to deal with transformations such as local warping and shifting and, furthermore, it allows the comparison between series of different length.

As shown in Figure 3.5a, the objective of this distance is to find the optimal alignment between two series $X = \{x_0, x_1, ..., x_{N-1}\}$ and $Y = \{y_0, y_1, ..., y_{M-1}\}$, by searching for the minimal path in a distance matrix (D) that defines a mapping between them. Each entry of the matrix D is defined by the ED between a pair of points (x_i, y_j).

This optimization problem is subject to three restrictions [124]. The boundary condition forces the path to start in position $D(0, 0)$ and to end in $D(N-1, M-1)$. The continuity condition restricts the step size, forcing the path to continue through one of the adjacent cells. Finally, the monotonicity condition forbids the path to move backwards in the positions of the matrix. Based on this, the problem is reduced to solving the following recurrence:

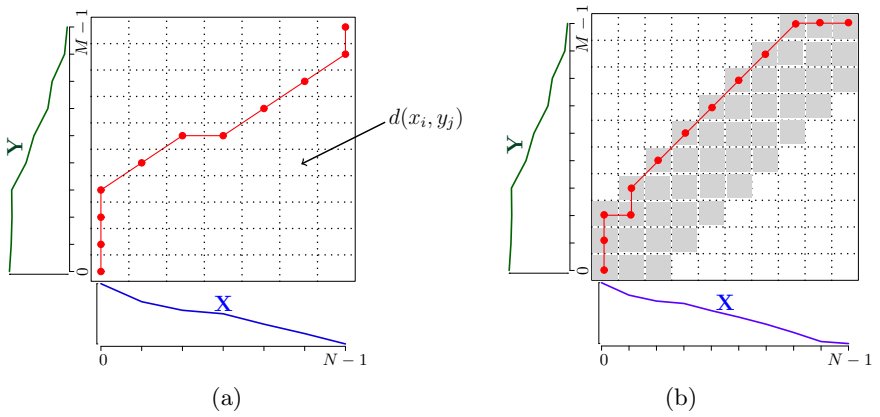


Fig. 3.5. Illustration of basic DTW calculation, and the restricted version with a Sakoe-Chiba windowing.

$$DTW(X,Y) = \begin{cases} 0 & \text{if } M-1 = N-1 = 0 \\ \inf & \begin{array}{c}\text{if } M-1=0 \\ \text{or} \\ N-1=0\end{array} \\ \begin{array}{l} d(x_0,y_0) + min\{DTW(Rest(X),Rest(Y)), \\ DTW(Rest(X),Y), DTW(X,Rest(Y))\} \end{array} & \text{otherwise} \end{cases} \tag{3.4}$$

where d is the ED and, being $X = \{x_0, x_1, ..., x_{N-1}\}$ and $Y = \{y_0, y_1, ..., y_{M-1}\}$, $Rest(X)$ and $Rest(Y)$ are defined as $\{x_1, ..., x_{N-1}\}$ and $\{y_1, ..., y_{M-1}\}$. This recurrence is typically solved by using dynamic programming.

Additionally, it must be noted that it is quite common to add an extra temporal constraint to DTW by limiting the number of vertical or horizontal steps that the path can take consecutively. As the simplest example, the classical Sakoe-Chiba band [183] places a symmetric band around the main diagonal and forces the path to stay inside this band (see Figure 3.5b). This adjustment avoids the matching of points that are very far from each other in time and, in addition, it reduces the computation cost [228].

3.2.3 Edit Distance for Real Sequences

Edit distance was initially presented to calculate the similarity between two sequences of strings and is based on the idea of counting the minimum number of edit operations (delete, insert and replace) that are necessary to transform one sequence into the other.

The problem of working with real numbers is that it is difficult to find exact matching points in two different sequences and, therefore, the edit distance is not directly applicable. Different adaptations have been proposed in the literature and the Edit Distance for Real Sequences is one of the most common.

By using the delete and insert operations, all these distances are able to work with series of different length.

Specifically in EDR, in order to adapt it to numerical values, the distance between the points in the time series is reduced to 0 or 1 [28]. If two points x_i and y_j are closer to each other in the absolute sense than a user specified ϵ, they will be considered equal. On the contrary, if they are farther apart, they will be considered distinct and the distance between them will be considered 1.

As an additional property, EDR permits gaps or unmatched regions in the database but it penalizes them with a value equal to their length. All this summarizes into the following recursion that is converted into an iteration by means of dynamic programming as in the previous case:

$$EDR(X,Y) = \begin{cases} N & \text{if } M-1=0 \\ M & \text{if } N-1=0 \\ min\{EDR(Rest(X),Rest(Y)) + d_{edr}(x_0,y_0), & \\ EDR(Rest(X),Y)+1, EDR(X,Rest(Y))+1\} & \text{, otherwise} \end{cases} \tag{3.5}$$

where d_{edr} represents the distance between two points in the series and takes a value of 0 or 1, as explained above.

Finally, as with DTW, a Sakoe-Chiba windowing may be added to the EDR distance in order to avoid excessive computational burden and pathological matching.

3.2.4 Fourier Coefficients based distance

As its name indicates, the similarity calculation in this case is based on comparing the Discrete Fourier Transform coefficients of the series. As such, this distance measure can be categorized among the feature based distance measures.

It is important to note that the Fourier coefficients are complex numbers that can be expressed as $X_f = a_f + b_f i$. In the case of real sequences such as time series, the Discrete Fourier Transform is symmetric and therefore it is sufficient to study the first $\frac{N}{2}+1$ coefficients. Furthermore, it is commonly considered that, for many time series, most of the information is kept in their first n Fourier Coefficients, where $n < \frac{N}{2}+1$ [3].

Based on all this information, the distance between two time series X and Y with Fourier Coefficients $\{(a_0,b_0),...,(a_{\frac{N}{2}},b_{\frac{N}{2}})\}$ and $\{(a'_0,b'_0),...,(a'_{\frac{N}{2}},b'_{\frac{N}{2}})\}$ is given by the ED between the first n coefficients:

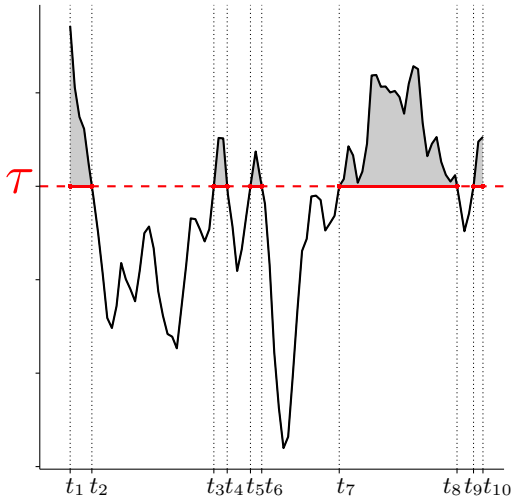
$$F(X,Y) = \sqrt{\sum_{i=0}^{n}((a_i-a'_i)^2+(b_i-b'_i)^2)} \tag{3.6}$$

3.2.5 TQuest distance

TQuest was presented by Aß falg et al. [5] and is classified as a feature based distance in [64]. In this manner, instead of comparing the raw values of the series, it studies the similarity of a set of features extracted from them.

As can be seen in Figure 3.6, the idea is to define the set of time intervals in a time series that fulfill the following conditions:

1. All the values that the time series takes during these time intervals must be strictly above a user specified threshold τ.
2. They are the largest possible intervals that satisfy the previous condition.



$$S(X,\tau) = \{(t_1,t_2),(t_3,t_4),(t_5,t_6),(t_7,t_8),(t_9,t_{10})\}$$

Fig. 3.6. Time series representation method used by the TQuest distance.

The distance between two time series X and Y that are represented by the interval sets $S(X,\tau)$ and $S(Y,\tau)$ is defined as follows:

$$\begin{aligned} TQuest(X,Y) = &\frac{1}{|S(X,\tau)|}\sum_{s\in S(X,\tau)}\min_{t\in S(Y,\tau)} d(s,s')+ \\ &\frac{1}{|S(Y,\tau)|}\sum_{s'\in S(Y,\tau)}\min_{s\in S(X,\tau)} d(s',s) \end{aligned} \tag{3.7}$$

where the distance between two intervals $s = (s_l, s_u)$ and $s' = (s'_l, s'_u)$ is calculated as:

$$d(s,s') = \sqrt{(s_l - s'_l)^2 + (s_u - s'_u)^2} \tag{3.8}$$

Part II

Contributions to travel time modeling: a taxonomy and analysis of the state-of-the-art

4

A review of travel time estimation and forecasting for Advanced Traveler Information Systems

As commented in Chapter 2, both the estimation and prediction of travel time are very useful for ATIS, and have been extensively studied in the literature. Furthermore, as stated on more than one occasion, these two modeling problems are the starting point of this dissertation. In this context, the aim of this chapter is to provide a comprehensive review on these two topics, building a complete background analysis of the available models and algorithms.

The remainder of this chapter is organized as follows. In Sections 4.1 and 4.2, extensive categorizations of the methodologies used in the literature for travel time estimation and prediction will be described, respectively. Once this has been introduced, the evaluation and validation of the resulting models is analyzed in Section 4.3. Finally, a discussion on the topic will be presented in Section 4.4 that will lead to a set of possible future research lines shown in Section 4.5.

4.1 Travel time estimation models

Recall that estimation algorithms calculate travel times of trajectories that have already ended, using data captured during the trip. In this context and, as explained in Section 2.2, estimation models strongly depend on the characteristics and modality of the available data. Since each type of traffic sensor provides different traffic information, it is convenient to classify the estimation methods depending on the source of data used in their construction.

4.1.1 Travel time estimation from point detectors

For many years, double or single inductive loops have been the most widely used traffic sensors [193] and therefore the vast majority of the travel time estimation algorithms from point detectors are based on them. However, most of the algorithms that will be presented next could be used equivalently with

point detectors other than inductive loops, because the type of data they require can also be collected from other types of point detectors [114].

4.1.1.1 Single loop detectors

As explained in Section 2.1.1, single loop detectors are only able to capture traffic flow and the occupancy of the detector. The few existing travel time estimation approaches using data from single loop detectors can be divided into three main classes: traffic theory based, data based and hybrid methodologies.

The first class of techniques apply relations between traffic variables, obtained from **traffic flow theory**, to extract travel time values from flow data [23, 24, 25, 129, 145, 150, 223, 251]. These estimation methods are essentially based on flow conservation and propagation principles [13, 24], but they use different approaches to traffic dynamics and diverse traffic theory identities.

On the contrary, in **data based methods**, equations from traffic theory are ignored and diverse statistical and machine learning methods are used to create new structures that relate flow, occupancy and travel time using the data as a baseline. Some of the most recurrent and successful data based models are the artificial neural networks. These models are inspired by the structure and functional aspects of the biological networks that neurons form in the brain and are able to construct complex non-linear relations between the input and the output variables. A wide variety of different neural structures have been proposed from fuzzy neural networks [153] to multilayer perceptron, radial basis and probabilistic networks [100]. Other data based models are polynomial regression models, which have been reviewed in [188], time series modeling techniques such as cross correlation functions [48] and a stochastic regression model that assumes that the travel times of vehicles arriving at a detector at a given time interval follow the same probability distribution [161].

As a special case, an **hybrid method** for travel time estimation is presented by Dailey [49] where traffic flow formulas are incorporated into a data based state space model in order to obtain spot speed measurements from occupancy and flow values. In a second step, the spot speeds are converted into travel time estimates by assuming a linear behavior of the speed between detectors and using a theoretical relation between speed and travel time.

It must be noted that, apart from Dailey [49], many other researchers have focused their attention on imitating double loop detectors by estimating point speed using data from single loop detectors. Unlike Dailey [49], most of these authors only focus on obtaining spot speed estimations and, since travel time is the topic of interest in this case, these studies will not be included in this review. However, they can be helpful if used in combination with other estimation methods and should not be dismissed.

A summary of all the methodologies presented in this section is presented in Table 4.1, including the advantages and disadvantages of each of the proposals.

Table 4.1. Travel Time Estimation Methods with Single Loop Detectors

Method Type	Advantages	Disadvantages	Specific Methods
Traffic Theory based methods	-Realistic theoretical relations are applied -Appropriate for capturing dynamic characteristics of traffic	-Expertise in traffic theory is necessary -All entry and exit ramps between detectors must be monitored	Flow conservation equations and traffic dynamics [23, 24, 25, 129, 145, 150, 223, 251]
Data based methods	-Underlying structures in the data can be found -No expertise in traffic theory is necessary	-A lot of data is needed and the quality of the data conditions the precision -The models are site specific	-Artificial Neural Networks [100, 153] -Polynomial Regression methods [188] -Cross Correlation functions [48] -Stochastic model based on equal probability distribution assumptions [161]
Hybrid methods	-The stochastic nature of traffic variables is included into the traffic flow identities.	-The estimation must be done in two steps: estimation of spot speeds and estimation of travel times.	-Extended Kalman filter based on traffic flow identities [49]

4.1.1.2 Double loop detectors

Double loop detectors are able to capture flow, occupancy and speed at the point where the detector is situated. Contrary to space mean speed, which is an area wide variable, the speed captured by double loop detectors is only valid for a specific point, and it is not reliable to assume that it can represent the whole study site [46, 190]. Because of this, very different approaches have been proposed to extend the data to the whole target site and provide accurate travel time estimations. A summary of all the methodologies for travel time estimation from double loop detector data is provided in Table 4.2.

The first and most common approaches are denominated **trajectory methods**. As explained in Section 2.1.2, when modeling travel time, the road is usually divided into smaller links and, normally, each link is defined as the road length between two detectors. The detector at the beginning of the link is called the upstream detector, while the one at the end of the link is the downstream detector. With this configuration, the main goal of trajectory methods is to estimate the travel time for each link by somehow extending the point speed collected by the loop detectors to the whole road section. Once this is done, the travel time for longer routes is obtained by summing the traversing times of the links that constitute the trajectory.

The most simple way to extend the point speed measurements to a whole link is by using piece-wise constant methods, where the mean speed captured in one of the sensors that delimits the link will directly represent the entire link [13, 192, 221]. Other approaches combine speeds from both upstream and downstream detectors or even use speeds from neighboring links [46, 185, 192, 200, 221]. All these interpolation methods are widely used and present similar good performances in free flow conditions [192]. However, they demand a dense spacing of the detectors, typically one detector every 500 meters [210], and generally, they do not provide good enough solutions in congested situations. Furthermore, missing and erroneous data are common when using loop detectors and data filling and cleansing methods should be considered in order to obtain higher accuracy values [185].

In order to better capture the traffic dynamics in varying or congested situations, some authors apply **traffic theory methods** [42, 104, 119, 215, 251] to estimate travel time from double loop detectors. These models are essentially based on kinematic wave theory and, reconstruct vehicle trajectories by studying the propagation of traffic and queues in different situations.

Apart from these studies, there has been some interest in trying to use double loop detectors as if they were interval detectors by using **vehicle re-identification**, which attempts to find a signature that uniquely identifies each vehicle at two consecutive detectors. As opposed to interval detectors such as AVI detectors, it is not so easy to uniquely identify vehicles using data from loop detectors. However, various techniques have been developed with this objective based on using vehicle lengths [41, 43], vehicle inductance values [2, 147, 198] or vehicle clusters, also called platoons [131].

Table 4.2. Travel Time Estimation Methods with Double Loop Detectors

Method Type	Advantages	Disadvantages	Specific Methods
Trajectory methods	-Simplicity -Very popular in practical applications	-Inaccurate in congested and varying flow conditions, especially when detector density is low -Heavily sensitive to errors and noise in data	Simple extensions of point speed to entire links [46, 192, 200, 221]
Vehicle Re-identification methods	Travel time is obtained directly from loop detectors, which are widely deployed in our networks	Individual vehicle information is necessary which is not always available	-Re-identification using length [41, 43] -Re-identification using vehicle inductance values [2, 147, 198] - Re-identification using volume and flow [131]
Traffic Theory based methods	-More complex and realistic relations are applied -Appropriate for capturing dynamic characteristics of traffic	-Expertise on traffic theory models is necessary -Individual speeds or headways might be necessary	Kinematic wave and traffic propagation formulas [42, 104, 119, 215]
Data based methods	No expertise in traffic theory is needed and good results are obtained	A lot of data is needed and the quality of the data conditions the precision	-Artificial Neural Networks [159] -Markov Chains [243] -Simple Bayesian Estimators [159]

This re-identification approach has some advantages compared to AVI interval detection [147]. First, the identification of the vehicles is anonymous and does not invade the privacy of the drivers. Second, there is no need for active population participation since all vehicles can be identified with the loop detectors and no special device has to be installed in the vehicles. Furthermore, some of these methods can also be applied with more basic detectors such as single loop detectors. Finally, these systems have more capacity to detect incidents, because not only travel time is observable but also spot speeds which give additional information. However, double loop detectors usually only provide aggregates of speed, flow and volume and since individual vehicle data may not be available, these contributions are not very popular for real world applications [193, 223].

Finally, less common approaches for travel time estimation from double loop detectors are **data based methods** based on statistical and machine learning models, such as simple Bayesian estimators, feed forward neural networks [159] and Markov chains [243].

4.1.2 Travel time estimation with interval detectors

Interval detectors are more recent than point detectors, and in the past few years research on the use of these sensors has flourished considerably. The reason is that interval detectors provide travel time data directly and offer more possibilities for ATIS.

4.1.2.1 Probe vehicles

Probe vehicles equipped with GPS systems and/or cell-phones are able to collect position, speed and time stamp data every few seconds [122, 246]. This introduces a wide range of new possibilities into travel time or space mean speed estimation. Probe vehicles are theoretically able to provide all the data needed to calculate the space mean speed because the vehicles can be tracked at all times. However, in real situations the use of this type of detector entails a couple of practical inconveniences that complicate the estimation of travel time. The most important is that, it is usually impossible to track all the vehicles in a traffic network and therefore the formula for space mean speed can not be directly calculated in most cases. This problem has been partly alleviated with the use of taxi or bus fleets [59, 164] and with the advance and popularization of wireless communication systems and smartphones [7, 246]. However, having access to a probe vehicle sample which is small and not representative of the whole population is still quite common. Given this situation, in the past few years, a series of methods, whose main objective is to estimate travel time using data from a insufficient or sparse probe vehicle sample, have been published.

To begin with, a few different **statistical approaches** have been proposed. The most simple approximation is a weighted average that combines

real time and historical data [164]. A slightly more complex model is presented by El Esawey and Sayed [59], where a Bayesian conjugate scheme is used with the same purpose. Finally, in [98], a complex statistical regression model is proposed that makes use of the correlations between links in order to generalize the low frequency GPS probe vehicle data.

Besides statistical methods, another approximation is the use of **fuzzy logic** [116, 122], that is an extension of regular set theory, in which each element is associated to a fuzzy set with a degree of membership. The objective is to assign the individual trajectories of probe vehicles to different fuzzy driving patterns and fuzzy traffic situations and to derive the mean travel time of the whole population from this information.

Finally, different **probabilistic graphical models** such as Markov chains [169] and Dynamic Bayesian Networks [88, 89] have been built from probe vehicle data, in order to model the spatial and/or temporal evolution of several traffic variables or parameters and obtain the probability distribution of travel time from this information.

It must be said that although the number of studies that use probe vehicles is still quite limited, the popularization of GPS enabled devices and smart phones promises an increasing interest in this type of models. In this line another factor that should be taken into account when working with this type of devices, apart from the sample size, is the noise that could accompany the measurements obtained from them due to the small sample sizes, loss of signal, positioning errors, etc. [7, 246].

4.1.2.2 AVI detectors

Another type of interval detector that has gained popularity in the past few years is the AVI detector. We recall that these detectors identify the vehicles at the beginning and end of the study section and infer travel time values from this data. Some of these detectors, such as license plate matching video cameras or closed toll highways, are able to collect the travel times of all the vehicles that travel in the surveilled section. However, in many cases, the detectors only capture the travel times of a sample of the whole traffic population which is not always sufficiently large.

In this second case, for example when using electronic toll collection tags or roadside beacons, directly calculating the average of the observed travel times is not always a good solution because the sample might not represent the whole population adequately. In this case, some authors propose more suitable **statistical methods** that combine current and historical data in different manners [132, 143, 201, 202].

Additionally, even when the sample of detected vehicles is big enough, when using data from AVI detectors, unlike with GPS enabled vehicles, a problem arises. This obstacle is the difficulty in differentiating noisy and unusual data from valid data. Extremely short and long observations should be removed from the data base to obtain more reliable estimations, but the

identification of non-valid data is not always immediate [152]. Because of this, some effort is put into filtering invalid registers such as travel times of vehicles that stop midway, duplicate entries, and observations of vehicles that travel faster than permitted [54, 55, 132, 152, 193, 195, 202].

4.1.3 Estimation with fusion of different data sources

Most of the studies in the literature use only one kind of detector data as input to the model. However, lately, fusion of different types of sensors has been introduced into the travel time estimation field to increase reliability of travel time estimates [35] and to reduce sensing costs [62].

Although in the literature the term 'data fusion' is used with various meanings, we will focus on data fusion from different types of traffic sensors, also denominated multi-sensor fusion.

We will distinguish between two distinct types of fusion algorithms. The first type directly accepts input data from different sources and constructs a unique estimation model. The second type of fusion consists in constructing an estimation model for each data source by using one of the methods presented in the previous sections and, finally fusing these estimates by different techniques. A graphical example of these two fusion methodologies can be seen in Figure 4.1.

In the **first type of fusion**, the direct fusion of data from different sources (Figure 4.1, Approach 1), methodologies such as neural networks, state space models or traffic theory based models are proposed in the travel time estimation literature.

Cheu et al. [32], Liang and Ling-Xiang [123] and Bachmann [6], present several feed forward neural networks with some of the input nodes corresponding to data from loop detectors and the remainder of input nodes corresponding to data obtained from probe vehicles.

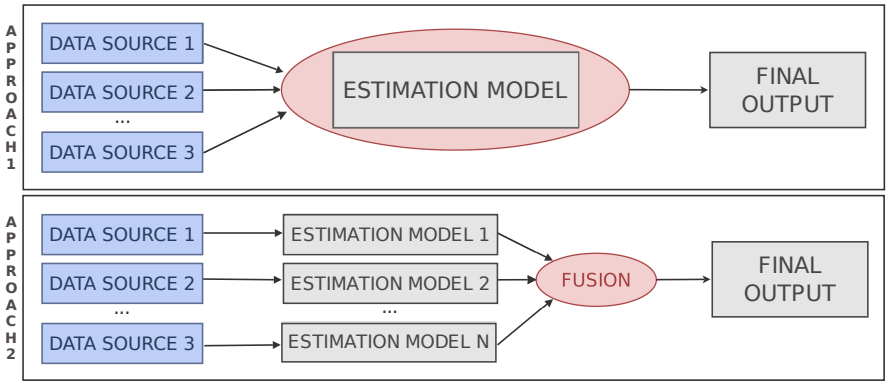


Fig. 4.1. Travel Time Estimation from Different Data Sources

Another way of fusing data from different sensors is using state space models. These models consist of two equations that describe the evolution of a non-observable state variable and are widely used for time series estimation and prediction. The state equation defines the theoretical evolution of the system while the measurement equation defines the relation between the observations and the true non-observable state. Kalman filters are recursive methods that are widely used to solve these dynamic state space models and different variations of these filters exist depending on the linearity of the equations involved. In the travel time estimation literature, three different types of state space solutions are proposed that accept inputs from different sensors [6, 37, 146].

Finally, models based on traffic theory are not so common when the data is provided by different sources, because it is not easy to find models that accept inputs from various sensors [39]. However, some isolated cases can be seen in [140], where shock wave theory is used to better estimate travel time in an urban stretch from AVI cameras and probe vehicle data; in [85], where a g-factor approach is used to combine data from single and double loop detectors and in [39], where the Moskowitz formula from kinematic wave theory is applied fusing data from loop detectors and GPS provided cell phones.

The **second type of fusion** algorithms estimate travel time separately for each available sensor and then fuse these estimations using linear combinations, Bayesian theory, evidential theory, fuzzy theory or historical and real time profiles (Figure 4.1, Approach 2).

One of the first proposals for this type of fusion was presented in the ADVANCE project by Boyce et al. [14]. In this work, 5000 probe vehicles were used to construct default historical profiles by using an asymmetric network equilibrium model. In the cases where the actual traffic state deviated significantly from the historical profiles, the travel time estimations are updated by data obtained from single loop detectors in real time.

Another way to combine estimations from different sensors is by weighted linear combination or weighted average. The weights can be calculated in different manners, but they are usually built by measuring the reliability of the estimations from each individual source and giving more weight to the most reliable sources. Some ideas for calculating these weights are proposed by Bachmann [6], Choi [35] and Choi and Chung [36].

Bayesian theory is also a common technique for this first type of fusion. The credibility of each source is measured by using probabilistic functions and based on these values, a final travel time estimation value is obtained by using the simple Bayes Rule. This technique has been used by Soriguera and Robusté [191] in order to fuse several different travel time estimation values extracted from loop detector and toll ticket data.

In relation to this, another methodology that is often used for fusion of estimators is evidential theory or Dempster-Shafer theory, and several attempts to apply this theory to estimation of travel time can be seen in [60, 61, 189]. Evidential theory is a generalization of Bayesian probability theory and it

permits the treatment of ignorance, which is not contemplated in Bayesian theory. Each of the possible discrete states or combination of states is assigned a credibility measure on the basis of a belief function (m_i), defined differently for each source i and that indicates the credibility or degree of trust that the source has for each possible output. Different methods have been proposed to calculate these mass functions in [60, 61, 189] and once this is done, Dempster-Shafer theory provides a simple formula that permits the fusion of two sources of information based on the orthogonal sum of the belief functions.

A last typical approach in data fusion is the use of fuzzy set theory which allows the introduction of vagueness into the model [105]. However, in travel time estimation, this methodology seems to be used only by Soriguera et al. [189].

4.2 Travel time prediction models

As defined in Chapter 2, in travel time prediction algorithms, the objective is to use the current and past traffic and contextual data to forecast the travel time in future time intervals [13]. Because of its utility for travelers, the prediction of this traffic variable has become a very recurrent topic in the literature of intelligent transportation systems and a vast portfolio of different methods has been presented with this objective [217].

Estimation methods presented in the previous section are more dependent on the specific characteristics of each data source and because of this, the classification has been given based on data sources. However, the data source or typology of the input data is not so relevant in prediction methods, because the data can be translated from one format to another using estimation methods. Because of this, it is interesting to categorize the prediction methods based on the type of model applied and not on the data source. A taxonomy of these methods is schematically represented in Figure 4.2 and will be developed in the next sections.

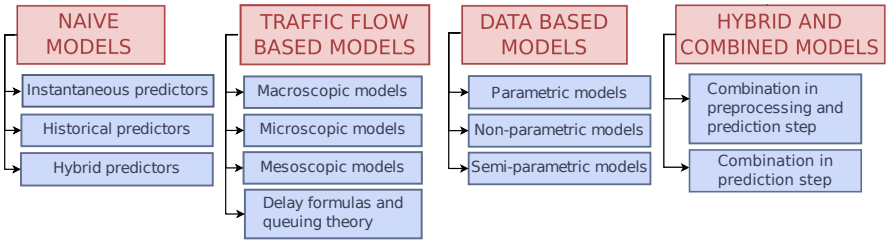


Fig. 4.2. Travel Time Prediction Models Taxonomy

4.2.1 Naive models

As the name indicates, these are the most simple and ad hoc travel time prediction methods. They do not need any training or estimation of parameters and are very fast. Nevertheless, they make some very restrictive assumptions that are not fulfilled in many situations [219]. They are mainly used in commercial ATIS because of their simplicity [87, 143, 195] but in the scientific literature they are usually only used as a baseline for comparison with other more complex methods. These methods are divided into instantaneous, historical and hybrid methods, and a summary including their advantages and drawbacks is available in Table 4.3.

4.2.1.1 Instantaneous predictors

These models assume that the traffic conditions, and therefore travel time, will remain constant indefinitely [217]. Based on this, the idea of instantaneous predictors is to simply output the most up-to-date travel time estimation available.

The assumption that the traffic state will not suffer any changes becomes weaker as the prediction time horizon increases and, therefore, these models are not reliable in most situations [184]. Because of this, instantaneous predictors provide an adequate comparison baseline, and any new travel time prediction method, in order to be considered, should be able to obtain better results.

4.2.1.2 Historical predictors

These models assume that the travel time at a certain time interval is very similar to the travel times collected at the same time in the past. The most common is to simply average all the historical travel times collected in the given time interval [184]. Slightly more complex methods reduce the historical set by filtering by weekday, month or other characteristics [87, 232] or weight the historical average according to the similarity of the current situation with the historical profile [87].

These methods depend greatly on the similarity in traffic conditions between current and past days and this does not always occur, especially in non-recurrent congestion situations. However, historical estimators are generally more accurate for long term prediction than the instantaneous predictors [184].

4.2.1.3 Hybrid methods

This type of model combines historical and instantaneous methods in a simple way and with no need for parameter estimation.

Table 4.3. Naive Travel Time Prediction Methods

Method Type	Advantages	Disadvantages	Specific Methods
Instantaneous	-Simplicity -Can give good results for short prediction horizons	-Assumptions are rarely fulfilled -Weak when the prediction horizon increases	Instantaneous predictors [184, 217]
Historical	-Simplicity -Can give good results for long prediction horizons	-Similarity with past conditions is required -Not good for non-recurrent congestion	Historical predictors [87, 184, 232]
Combination methods	The assumptions are not so strong	The parameters must be determined manually	-Exponential Filtering [233] -Switch Models [184]

A first example is the **weighted average** between an instantaneous and a historical predictor presented by Wunderlich et al. [233], where the weights are pre-specified by the researcher. A second approach is proposed by Schmitt and Jula [184] who applies a **switch model** between the two naive predictors. The instantaneous predictor is applied for short-term prediction while the historical predictor is used for longer term prediction.

4.2.2 Traffic theory based models

Traffic theory based approaches usually focus on recreating the traffic conditions in the future time intervals and then deriving travel times from the predicted traffic state and variables values [219]. The most recurrent models are simulation tools which can be divided into three main categories: macroscopic, microscopic and mesoscopic simulation [217]. Moreover, apart from simulation models, there are some other methods that must be taken into account: delay formulas and queue theory.

Traffic theory based models are very advantageous for ATMS and ATIS because they give very detailed information about the location and causes of delays on a road network, and they provide useful means for decision making [217]. Furthermore, they allow the representation of crucial components in traffic modeling such as traffic lights, intersections, lanes etc.

The main drawback of these models is that the traffic condition or traffic flow that they recreate must be very similar to the real traffic situation in order to obtain accurate travel time predictions. However, this is fairly difficult in general cases, considering that simulation models are based on simplifications of the real situation and are limited to a restricted number of variables and possibilities. Furthermore, these methods are in general computationally very intensive and a high knowledge of traffic theory is necessary for their application. More specific information about each type of traffic theory based method can be looked up in Table 4.4.

4.2.2.1 Macroscopic simulation models

The models apply equations from fluid flow theory to model the traffic by simulating aggregated traffic variables such as flow, density and mean speed in the future time intervals. There are many equations and general relations between traffic variables that can be used for macroscopic simulation and they are categorized based on the order of the mathematical equation. These models do not generally output travel time values and therefore, these will have to be inferred using estimation methods such as the ones presented in Section 4.1.

An example of a macroscopic simulator software is METANET [154]. Moreover, some other applications of macroscopic models to travel time prediction can be seen in [19, 47]. However, it must be said that this type of model is not frequently used in ATIS and they are not useful in some cases such as urban traffic prediction.

4.2.2.2 Microscopic simulation models

There are different ways to represent the distribution of traffic in a traffic network. Two typical ways are Origin-Destination (O-D) matrices, which represent the traffic flow from every possible origin to every destination, or turning volumes that represent the percentage of the traffic that turns in each direction in an intersection. Microscopic models take predictions of these O-D matrices or turning volumes as input and simulate trajectories of individual vehicles in the future time intervals, taking into account factors such as interactions between vehicles, driver behavior, lane changing, etc. [219].

Some of the models used in microscopic simulation are **car-following models** [15] and **cellular automaton models** [144]. The first are time continuous ordinary differential equations which represent the behavior and trajectory of each vehicle depending on the vehicle in front. On the contrary, the second are simpler models that discretize the time and study road into small cells and move the vehicles along the cells by following some predefined rules for lane-changing and acceleration, among other factors.

As opposed to the macroscopic models, in these cases, travel times can be derived directly [219] but there is an additional task of predicting O-D matrices or turning volumes. Some microscopic simulation softwares are CORSIM [77], PARAMICS [20], and INTEGRATION [214] and some examples of their use for travel time prediction for ATIS systems can be seen in [128, 138].

4.2.2.3 Mesoscopic simulation models

These models combine the features of microscopic and macroscopic simulations models. They simulate individual vehicles, but describe their behavior and interactions based on general macroscopic relationships [217]. They are mostly used in cases of large networks where the microscopic simulation is infeasible.

Some examples of mesoscopic simulators are CONTRAM [203], which groups the vehicles into platoons and assigns the same behavior to the whole platoon, DynaMIT [9], which divides the road into cells and assigns a behavior to each cell and DynaSMART [97], where the vehicles are represented individually but the speed in each link is determined by a macroscopic speed-density function. Similar to other simulation models, mesoscopic models are mainly used for evaluation purposes and traffic management, however, they can also be useful for travel time prediction for ATIS systems. Some examples of this latter purpose can be seen in [92, 155].

4.2.2.4 Delay formulas and queuing theory

To finish with traffic theory based travel time prediction models, delay formulas and queue theory [23, 253] must be mentioned. They are basically estimation methods, but they become predictive methods when the input variables are predicted values.

Table 4.4. Traffic Flow Based Travel Time Prediction Methods

Method Type	Advantages	Disadvantages	Specific Methods
Macroscopic Simulation Models	Good for prediction in big networks	Travel time is not obtained directly -Individual details are overlooked	METANET [154]
Microscopic Simulation Models	Very detailed information can be obtained	-Computationally intensive -O-D matrices must be predicted	-CORSIM [77] -PARAMICS [20] -INTEGRATION [214]
Mesoscopic Simulation Models	-Faster than microscopic models -More detailed than macroscopic models	They inherit some of the disadvantages of microscopic and macroscopic models	-CONTRAM [203] -DynaMIT [9] -DynaSMART [97]
Delay Formulas and Queuing theory	Good for specific situations (intersections, congestion etc.)	Not used much for ATIS	Specific mathematical formulas for delays and queues [23, 56, 127, 253]

These models are mostly used for delay prediction in urban roads in more specific situations of congestion or signalized intersections. Because of this, they must be used in combination with another method that predicts the cruising time for the rest of the link [127]. They are widely used for optimization of signal timing and traffic management in general [56], but are also useful to obtain predictive values for ATIS, especially in the case of urban sites [253].

4.2.3 Data based models

In data based models, the function that relates the explicative variables with the target variable (f) is not obtained from traffic theory identities and relations, but instead, it is determined by the data itself by using statistical and machine learning techniques [219].

The main advantage of these methods is that expertise in traffic theory is not required. The downsides are that usually a lot of data is needed, which is not always available, and that the models are strongly linked to the data and consequently to a certain study site [219]. Because of this, they are not always successfully transferable to other sites. An outline of these methods is presented in Table 4.5.

4.2.3.1 Parametric models

In these models the ensemble of parameters that must be estimated to completely define f is predefined and set in a finite dimensional space [80]. In the case of travel time prediction, this means that the structure of f is fully predetermined by the researcher but, however, some parameters will be determined using the data.

The most typical parametric model is **linear regression**, where the target variable is a linear function of the explanatory or input variables:

$$TT(t+k) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_n X_n \tag{4.1}$$

The use of different sets of input variables and diverse techniques to estimate these parameters $(\beta_0, \beta_1, ..., \beta_n)$ define the different linear regression models. Some possible input variables are traffic observations from current and past time intervals [111, 148] or more elaborate inputs such as historical and instantaneous predictors [58]. Adding context information such as the departure moment, the day of the week and the weather can also be beneficial [111]. In travel time prediction, the parameter learning techniques can also vary from the common least squares approach [111, 148] to more complex entropy minimization methods [58].

The next type of parametric model is the **Bayesian Net**. The most simple example is the Naive Bayes model [115, 248] where it is assumed that the

explanatory variables are conditionally independent, given the target variable. In some other more complex cases, more elaborate relations between the variables are found by using the data [22, 95].

A third group of parametric model for travel time prediction consists of **time series models**. Among these, the most common are the state space models that have been mentioned in previous sections and are typically based on the following equations:

State Equation: $TT(t+1) = \Phi(t)TT(t) + w(t)$

Measurement Equation: $Y(t) = TT(t) + v(t)$

where TT is the hidden travel time variable to predict, Y corresponds to the travel time observations collected by the traffic sensors, Φ is a dynamic parameter that can be set using different methodologies and w and v are white noise errors with zero mean and variances that change with time. When linearity and normality conditions are fulfilled or assumed, this model can be solved by using a regular Kalman filter [30, 34, 110, 240]. In addition, some enhancements to this state space model are presented by Zhu et al. [255] and Vanajakshi et al. [222], where the travel times of neighboring links are also taken into account, and by Jula et al. [101], where the historical traffic changes in the target link are included into the model.

Another type of time series model, which can also be formulated as a state space model, is the ARMA model, which is a combination of autoregressive (AR) and moving average (MA) models. The general formulation of an ARMA(p,q) is:

$$TT(t) - \sum_{i=1}^{p} \phi_i TT(t-i) = Z(t) + \sum_{j=1}^{q} \theta_j Z(t-j) \tag{4.2}$$

where the target variable $TT(t)$, in our case travel time at departure time interval t, is represented as a linear function of this same variable in previous time intervals $(TT(t-1), ..., TT(t-p))$ and a set of white noise variables $(Z(t), ...Z(t-q))$. The parameters are represented by $(\phi_1, .., \phi_p, \theta_1, ..., \theta_q)$ and can be estimated using different methods [239, 245]. A couple of modifications of the ARMA model are presented in [245] and [76]. In the first study, a generalization of the ARMA model denominated ARIMA that is useful to deal with non stationary time series is used. Furthermore, a dummy variable is included into the model to account for incidents and rapidly varying traffic conditions. In the second one, a seasonal component is added, obtaining a structure denominated SARIMA.

Finally, more unusual time series methodologies applied to travel time prediction are non-linear time series models [96] and Dynamic Bayesian Networks [88], which are a subtype of Bayesian Networks specifically designed for time series.

4.2.3.2 Non-parametric models

In this case, the structure of the model is not predefined and therefore the shape of f is also obtained from the data. Consequently the term non-parametric does not mean that there are no parameters to be estimated, but on the contrary, it means that the number and typology of the parameters is unknown a priori and possibly infinite [217].

There are many methodologies for non-parametric regression, and the most recurrent in the literature of travel time prediction are **artificial neural networks**. Many different types of neural networks have been applied from regular multilayer feed forward neural networks [106, 111, 120, 148, 156, 230, 245] to more complex spectral basis neural networks [158], counter propagation networks [52], generalized regression networks [100] and recurrent neural networks [53, 151, 220] and different input variables are used in each case depending on the availability. The training of the neural networks is commonly carried out by different variations of the back propagation algorithm [111, 158, 220], although depending on the network type, other types of techniques might be preferable [52, 106, 220].

Another option for travel time prediction is using **regression trees** [111, 148]. In travel time prediction, regression trees are trained by using top-down iterative methods, where at each iteration a set of new branches of the tree is created by choosing the explanatory variable that best divides the dataset. The choice of the explanatory variable and the division thresholds is made by using a pre-specified criterion, for example the Gini Impurity or the Information Gain.

A third non-parametric approach that gives very accurate results is the **local regression approach**. The main idea of these methods is to choose a set of historical data instances which have similar characteristics to the current situation and then obtain the prediction by using a model constructed with these chosen data points [148]. Different local regression models appear depending on the type of technique used to select the set of similar historical points and also depending on the methodology chosen to fit the model [27, 38, 148, 187, 199, 205].

Finally, a few researchers [134, 232, 245] have used **Support Vector Regression (SVR)** techniques to find travel time in the future. This approach consists on mapping the input dataset into a higher dimensional space with the help of a kernel function and finding the flattest linear function that relates these modified input vectors and the target variable with an error smaller than a predefined ϵ. This linear function is mapped again into the initial space to obtain a final non-linear function that is used to predict travel time. Some of the most common kernel functions used in this case are radial basis kernels and linear kernels [232, 245].

Table 4.5. Data Based Travel Time Prediction Methods

Method Type	Advantages	Disadvantages	Specific Methods
Parametric models	Visual and easy to understand	Too simple structures may not represent the data well	-Linear Regression [58, 111, 148] -Bayesian Nets [22, 95, 115, 248] -Time Series Models [30, 34, 76, 88, 96, 101, 110, 222, 239, 240, 245, 255]
Non-Parametric models	Underlying complex, nonlinear structures can be found	More data than for parametric models is needed	-Neural Networks [52, 53, 100, 106, 111, 120, 148, 151, 156, 158, 220, 230, 245] -Decision Trees [111, 148] -Local Regression Methods [27, 38, 148, 187, 199, 205] -Support Vector Regression [134, 232, 245]
Semi Parametric models	-Not as simple as parametric models -Valid for situations where non-parametric models suffer from the curse of dimensionality	Structure is somewhat predefined and may not represent the data well	Varying Coefficient Models [78, 176, 184, 252]

4.2.3.3 Semi-parametric models

This last case is a combination of parametric and non-parametric regression. The idea is to loosen some of the strong assumptions of the parametric model to obtain a more flexible structure [180]. In the case of travel time prediction, semi-parametric models are presented in the form of **varying coefficient regression models** [81]. Travel time is defined as a linear function of the naive historical (T_h) and instantaneous predictors (T_i), but the parameters vary depending on the departure time interval (t) and prediction horizon (Δ) [78, 176, 184, 252]:

$$TT(t, \Delta) = \alpha(t, \Delta)T_h + \beta(t, \Delta)T_i \tag{4.3}$$

In this approach, the structure of f is defined as a linear function with respect to T_h and T_i, which corresponds to a parametric model. However, the parameters α and β are defined as smooth functions of departure time and prediction horizon and have previously unknown structures, which corresponds to non-parametric models.

An enhancement of this method is presented in [93], where a log-linear regression model with varying coefficient is applied.

4.2.4 Combined and hybrid models

These last models are denominated hybrid or combination models because they combine several models of the same or different type. The objective is to enhance the performance of each of the participant models. In this context, we will define two different types of combinations:

4.2.4.1 Combination in preprocessing and prediction step

This type of combination consists in the consecutive use of two methods. A first method, generally data based, is applied to pre-process, simplify or group the input data. Then, a second method is used to obtain the predictive travel time values.

A first common approach is to use methods such as clustering [117], principal component analysis [53, 244] or rough set theory [31] in order to **reduce the number of features** and obtain a new and simplified set of input data. These new input vectors are later introduced in models such as neural networks [117] or support vector regression [31] to calculate the predictions.

Another typical procedure is to initially apply clustering methods to **identify different traffic states or situations** and then, to **build specific models for each case**. As we have already commented in previous sections, in this dissertation we will take this type of combined models as a starting point that will lead to the methodological contributions that we present in Chapters 6 and 7.

As examples, Zou et al. [256] combine a rule based clustering method with several neural networks, Yildirimoglu and Geroliminis [244] present a Gaussian Mixture Model for the clustering task and stochastic congestion map models and speed profiles to provide the predictions and, finally, Elhenawy et al. [63] apply a K-means algorithm to cluster the data and a set of genetic programming algorithms to find the predictive function for each cluster.

4.2.4.2 Combination in prediction step

This second type of combination fuses several methods directly in the prediction step in order to obtain more reliable forecasts.

A first typical example is the use of **meta-models**. In this case, several models of the same type are combined by methodologies such as boosting [121], bagging [134] or Bayesian combination [216, 218]. Some examples for travel time prediction can be seen with combinations of decision trees [134] and neural networks [121, 216].

A second strategy is based on the **combination of traffic theory models with data based models**. In this line, Liu et al. [126] incorporate queue theory equations into a data based state space model to capture the dynamics of traffic more adequately. Furthermore, Hofleitner et al. [88] prove that the inclusion of a theoretical density model, which models the distribution of probe vehicles in a link, improves the predictions of their data based Dynamic Bayesian Network. Also, in [89], a Dynamic Bayesian Network is used to estimate some parameters of a predictive model based on traffic flow theory. To finish, in [235] and [57], data based models such as SARIMA time series models or CART regression trees are combined with macroscopic traffic flow and queuing theory formulas to predict the travel time in incident situations.

Finally, a third approach is to **combine different types of data based models** in the prediction phase. In [242], a neural network is used to fuse the predictions obtained from an instantaneous naive model, two variations of the exponential smoothing model and an ARIMA model, with the objective of obtaining reliable forecasts when data is collected at irregular time intervals. Li and Chen [118] propose a combination of K-means clustering, decision trees and neural networks to predict travel time on a freeway with non-recurrent congestion. Another proposal is to use another data based method in the training process of a neural network. In [220] and [126] extended Kalman filters are used to train neural networks. Moreover, in [100] and [254] radial basis neural networks are used and these neural networks make use of clustering or other data based methods in the training phase to obtain the centres of the hidden nodes.

4.3 Evaluation of travel time estimation and prediction models

In order to assess the quality and reliability of the travel time models presented in the previous sections, a complete and thorough evaluation process is necessary. This process should take into account the effect of the numerous factors that affect travel time and should provide a complete and quantified view of the reliability of the model enabling the comparison with other models.

4.3.1 Factors affecting travel time estimation and prediction

There are many factors that affect the reliability or quality of travel time estimation and prediction models and in this section they will be divided into two main groups.

The first group of factors will be directly related to the characteristics of the data. On the one hand, although some travel time modeling methods such as support vector regression are more robust in the presence of noisy or dirty data [232], all methods are, to some extent, negatively influenced by bad-quality data. Consequently, it is very recommendable to apply data cleansing methods such as low pass filters [111, 251], threshold methods [128], wavelet transformation methods [247], rough sets [31], outlier detection methods [247], missing data treatment techniques [128, 185, 199] and other ad hoc filters [41, 132, 152, 252] before passing to the estimation or prediction step. On the other hand, aggregation and simplification of the data is commonly applied to reduce storing costs and computational burden and this leads to an important information loss if it is not carried out correctly. Consequently, in the past few years, some researchers such as Bigazzi et al. [12] and Oh et al. [149] have studied the effect of aggregation in different applications while some others have aimed to find the optimal aggregation intervals, such as Park et al. [157].

The second group of factors are those related to non-recurrent congestion, which augment travel time variability and complicate and deteriorate the modeling process, especially in the case of prediction. They include adverse weather conditions, traffic incidents, characteristics of road geometry, bad road conditions and several other factors that cause variations in the demand or capacity of traffic flow, which are different to the typical traffic patterns, and therefore seriously affect the accuracy of the travel time model [91]. Different studies have focused on measuring the impact of these factors in travel time or traffic in general, especially the effect of adverse weather conditions [210] and incidents [209]. However, given the unpredictability of many of these factors and the difficulty in quantifying their impact in real situations, in part due to the challenge of obtaining significant data, most of the travel time models do not include their influence and therefore yield bad solutions in their presence. As such, only a few studies such as [57, 235, 244, 245] are available that attempt to predict travel time on variable traffic or incident conditions.

4.3.2 Performance measures for travel time models used in ATIS systems

For ATIS systems, a complete and adequate evaluation framework is presented in [219] where the reliability of a model is assessed in terms of its accuracy, validity, robustness and adaptiveness. The consideration of all these characteristics implies that an acceptable travel time estimation or prediction model for ATIS systems should output accurate results, similar to the real values, while adapting itself and performing robustly in different traffic situations and in the presence of missing or erroneous data.

However, most of the authors concentrate solely on assessing the accuracy of the model by using different error measuring statistics, mostly Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error (MAPE) [217]. These error measures only capture the average error that the methods commit throughout the study period and are therefore not adequate to evaluate the robustness of the model in different situations or to assess the effect of the factors mentioned above.

This evaluation can be slightly improved if the error measuring statistics are calculated separately for different time periods or even for different traffic situations. Moreover, the most complete evaluation procedure is the study of the distribution of the error [109], instead of focusing only on mean values.

4.4 Discussion

As shown throughout the chapter, many models have been published for both travel time prediction and travel time estimation. However, there is a lack of complete comparative studies that evaluate the goodness of different models. Because of this, at the moment, there is not enough information to choose one *best method* for travel time estimation or prediction. Moreover, the existence of such a model is highly questionable and it is more realistic to assume that each situation and study site probably requires the use of a different type of method.

In spite of this, based on the results provided by the authors, some information concerning the number of studies that use real/simulated data as input and for evaluation, the frequency in which the authors validate their proposals in each type of study site, and the number of studies that compare their approaches with methods from other categories can be extracted and summarized (see Tables 4.6 and 4.7). In addition, in the case of prediction, the number of proposals that perform one-step-ahead prediction and those that forecast multiple steps ahead into the future are compared. It must be noted that not all publications provide all the information and some fit into more than one option, so the counts provided in the table do not always coincide with the total number of publications of each type.

In the following paragraphs, the results provided in these tables are analyzed and some meaningful conclusions are extracted. Moreover, a series of general recommendations are given for the selection of one method over another, including some general headings about the accuracy and efficiency of the methods.

To begin with, it can be concluded that the choice of a specific method is very closely related to the **typology, quantity and quality of the available data**. As can be seen in Table 4.6, this is obvious for estimation methods, which directly depend on the type of traffic sensor available, but it is also relevant in the case of prediction. For example, regarding the quantity of data, statistical and machine learning estimation and prediction methods, especially non-parametric ones, generally require considerable amounts of data that might not always be available, especially in urban environments. As a consequence, their application is mostly centered on short freeway segments. On the contrary, traffic theory based methods are not so data intensive and are more frequently applied in urban networks, where data sources are scarcer.

It can also be observed in the table that most of the authors attempt to use real data to validate their proposals, especially in the case of prediction. Due to the increase and popularization of the diverse traffic sensors, the acquisition of real data is usually not a problem in general contexts. In this manner, simulated data is commonly used for more specific traffic contexts such as incidents, road works, congestion, urban networks, etc. for which it might be complicated to obtain a sufficient amount of data. Moreover, simulated data acquires more relevance in estimation models because of the difficulty in obtaining ground truth travel time measurements for validation.

In relation to the **popularity** of the different models, to begin with, it seems that methods based on traffic flow theory and simulation are more popular for estimation tasks than for prediction. This is expected because, as commented in previous sections, many of these methods are essentially estimation models that become predictive when the input variables are forecasted values. In addition, the computational burden of some traffic flow based models make them complicated to use for on-line predictive tasks.

In relation to data based models, it can be seen in Tables 4.6 and 4.7 that the most recurrent are artificial neural networks and time series models, especially state-space formulations. Neural networks are popular because they are able to find complex non-linear relations between the traffic variables. The main drawbacks of these methods are the cost and difficulty of their calibration and training process and their limited interpretability or *black box* nature. On the contrary, state space models are more simplistic but are fast and enable the combination of historical information with real time data. Furthermore, some more complex versions provide the opportunity to incorporate traffic flow equations.

To finish with this short overview of the popularity of the methods, we must say there is a clear uptrend in the use of fusion or combined models in the past few years. The main reason for this is that these models are the

Table 4.6. Summary of characteristics of the reviewed travel time estimation methods.

Method	# of publications	Input data: Real/Simulated	Ground truth data: Real/Simulated	Urban/Highway	Section/Network	Comparison with other methods
Single loop detectors						
Traffic theory based models	8	5/6	4/7	-/6	6/1	5
Data based models	5	4/-	4/-	2/2	4/-	-
Hybrid models	1	1/-	1/-	-/1	1/-	-
Double loop detectors						
Trajectory methods	5	2/3	2/3	-/5	5/-	-
Vehicle re-identification methods	6	5/1	5/1	1/5	5/1	1
Traffic theory based models	4	2/2	2/2	1/3	4/-	2
Data based models	2	2/-	2/-	1/1	2/-	-
AVI detectors						
Statistical models	4	4/-	4/-	3/1	4/-	3
Probe vehicles						
Statistical models	3	2/2	2/2	3/-	2/1	-
Fuzzy logic	2	2/-	2/-	-/1	2/-	-
Probabilistic graphical models	3	3/1	3/1	3/-	1/2	2
Heterogeneous sources						
Type 1 fusion						
Neural Networks	3	2/2	2/2	2/1	2/1	2
State-space models	3	1/3	1/3	1/2	3/-	3
Traffic theory based models	3	3/-	3/-	1/2	3/-	2
Type 2 fusion						
Weighted linear combination	3	3/2	2/1	-/2	3/-	2
Bayesian theory	1	1/-	1/-	-/1	1/0	1
Dempster-Shafer theory	3	3/-	2/-	1/2	3/-	2
Fuzzy logic	1	1/-	-/-	-/1	1/-	1
Network equilibrium model	1	1/-	1/-	1/-	-/1	-

perfect framework to include data from different sources and to take advantage and combine the strengths of different types of travel time estimation and prediction methods.

Next, we study the **study sites** where the different proposals are evaluated. It is evident that most travel time prediction and estimation studies are situated in freeway segments (see Tables 4.6 and 4.7). This is especially noticeable in the case of data based models, such as linear regression, time series models or neural networks, which are very popular in the literature but have generally only been validated in highway sections. The extension to ur-

Table 4.7. Summary of characteristics of the reviewed travel time prediction methods.

Method	# of publications	Input and ground truth: Real/Simulated	One-step/Multi-step	Urban/Highway	Section/Network	Context information	Comparison with other methods
Naive methods							
Instantaneous	16	16/-	5/11	3/14	14/2	-	16
Historical	16	14/2	6/10	1/13	12/3	✓	14
Hybrid	2	1/1	1/1	-/2	1/1	-	1
Traffic theory based models							
Macroscopic simulation	2	2/-	2/-	1/2	2/-	✓	0
Microscopic simulation	2	2/-	-/2	2/1	2/-	✓	0
Mesoscopic simulation	2	2/-	1/1	1/1	-/2	✓	1
Delay formulas and Queuing theory	4	2/2	1/2	2/-	4/-	-	1
Data based models							
Parametric models							
Linear regression	3	2/1	-/3	-/3	2/1	✓	2
Bayesian networks	4	1/2	-/4	3/1	1/2	✓	1
Time series models	12	10/3	7/5	4/8	10/2	✓	4
Non-parametric models							
Neural Networks	13	11/1	3/10	4/8	12/-	✓	6
Regression trees	2	2/-	-/2	-/1	2/-	✓	2
Local regression	6	5/1	-/6	1/4	4/1	-	4
Support vector regression	3	3/-	-/3	1/2	2/1	✓	3
Semi-parametric models							
Varying coefficient models	4	4/-	-/4	-/4	4/-	-	4
Combined and hybrid models							
Combination in preprocessing and prediction step							
Dimensionality reduction	4	4/-	3/1	1/3	4/-	✓	0
Combination of models for different situations	3	3/-	1/2	-/3	3/-	-	2
Combination in prediction step							
Meta-models	4	3/1	1/3	2/1	2/2	✓	2
Combination of traffic theory and data based models	5	5/-	3/2	3/2	3/2	✓	4
Combination of data based models	6	5/1	4/1	3/2	5/-	✓	3

ban sites and entire networks is a topic that has gained a lot of interest in the recent years and is certainly a requirement for future models aimed at ATIS. In this line, it seems that traffic flow and simulation models together with delay formulas and queuing theory provide adequate means to model the complex traffic of urban environments. Based on this, the combination of data based models with this type of formulas (see Section 4.2.4.2) appears as a promising future line.

Contextual information is very important for travel time prediction and this is why many methods include this type of information as input (see Table 4.7). However, the typology of context information varies from model to model. Models based on traffic flow generally include information about the characteristics of the study site such as number of lanes, number and situation of intersections, situation and cycle length of traffic lights, etc. The reason is that in order to correctly model the dynamics of traffic flow it is necessary to have detailed information about the settings of the study site. On the contrary, data based models normally focus on weather information, information about incidents, calendar information, etc. and only a few models include information concerning the characteristics of the study site.

Including information about the characteristics of the study site contributes to more general models that can be transferred to other study sites more easily. Contrarily, data related to the weather, incidents or calendar issues is valuable to predict recurrent and non-recurrent congestion, and can be useful to forecast travel time in these situations. Because of this, in our opinion, both types of contextual information should be considered in all model types in the future.

Another factor studied in Table 4.7 is whether the proposed models forecast **one-step-ahead** (generally 5 minutes) **or** if the are able to provide **multi-step predictions**. As commented in previous sections, the prediction horizon varies a lot over models. It can be seen that while some simple methods such as state space models or instantaneous predictors are more useful for one step predictions, other methods such as regression trees, neural networks or local regression methods usually aim at forecasting for longer horizons, which is more useful for ATIS. In addition, not all authors explicitly declare the prediction horizon used in their experimentation, which greatly complicates the reproducibility and comparison between methods. In all cases it is evident that longer prediction horizons lead to lower accuracy values. Furthermore, more than one study supports that real time information is valuable for short time prediction, but as the horizon increases, the inclusion of historical data leads to more accurate predictions. Long term prediction is an important topic for ATIS and therefore, all these factors should be taken into account in future works.

With regards to **accuracy**, it is difficult to extract conclusive evidence from the data provided in most studies. On top of that, no complete comparative studies are available. In the few cases where comparisons are made, only the most simple and naive models are used and researchers do not usu-

ally compare their methods with other types of state-of-the-art models (see the last columns of Tables 4.6 and 4.7). Furthermore, the methods are evaluated in different traffic contexts and study sites, and there is no standard validation framework. In this context, accuracy results should be handled and interpreted with caution. It should be said that this problem would be greatly alleviated with the publication of a complete comparative study or the creation of a repository where the authors could upload their algorithms, include datasets for evaluation and create standard evaluation methodologies, which are not available at present.

In spite of this, it is clear that for normal traffic situations in highway environments, where abundant real time traffic information is available, simple models such as linear regression or naive models in the case of prediction and trajectory methods in the case of estimation provide good solutions to model travel time. For more complex situations, such as congested roads, incident situations or urban contexts, models that are able to model the non-linear relations between traffic variables are necessary. In this manner, neural networks, local regression techniques and models based on traffic flow seem to be the most valuable methods according to various authors. However, many of these models require a great effort of calibration and parameter selection and this preliminary step greatly affects the accuracy of the methods. As a last recommendation, in order to combine the capabilities and strengths of both simple and complex models, it is interesting to use combination models such as those presented in Section 4.1.3 and 4.2.4, as they have shown promising accuracy values in recent works.

Finally, the **efficiency** issue is another factor to take into account when choosing one specific technique over another. Generally, this factor is completely ignored by the authors and is almost never mentioned in the publications. This makes it impossible to extract any detailed conclusions. However, we can conclude that models based on traffic theory can be very useful to describe congested situations or urban traffic in detail but they are generally complicated and computationally expensive models. Because of this, their application to entire networks might not be efficient, especially for real-time estimation or prediction purposes. This is especially noticeable for microscopic simulation models. Contrarily, the training of data based models, especially non-parametric ones, is usually computer intensive, but once the model is constructed, they are very useful to give prediction and estimation values in an on-line fashion. Nevertheless, many of these models depend on the similarity of the actual traffic situation with historical patterns and, therefore, it is necessary to re-train the model in a periodic manner to capture the changes that may occur in the underlying traffic dynamics. Furthermore, in the case of prediction, many data based models require the training of a separate model for each prediction horizon. Based on this, when using data based models, it should not be forgotten that, depending on the quantity of data and the complexity of the models, the training of these models can be highly expensive in computational terms.

To sum up, although no complete and objective comparative study is available, depending on the data resources, the study site and traffic situation we are dealing with and the efficiency requirements, some methods are more recommendable than others. Furthermore, hybrid or combination models seem like an interesting future line because they have reported good accuracy values and enable the fusion of different types of models, data, etc. Nonetheless, it is surprising to note that many of the popular and most recent ATIS are based on the most simple and naive travel time estimation and prediction models available. The main reason for this is that the more complex models published in the literature are generally not appropriately adapted to the practical requirements of an ATIS, such as dealing with noisy or missing data, making predictions for an entire network, using data from different sources and types, etc. Although in some situations simple models may give acceptable results, as commented above, the use of more elaborate models is highly recommendable for many more complex traffic situations. Nevertheless, for this to be viable, the models should be directly applicable and should meet with all the requirements of ATIS. This is not the case for most published models that are only applied in short highway segments, with no problems of missing data or lack of traffic sensors, and no large or unexpected variability in traffic conditions.

4.5 Conclusions and future work

Travel time is a very useful traffic variable, especially for ATIS, and the modeling of this traffic variable has been a recurrent topic in the scientific literature. However, the only reviews available for this topic are restricted to predictive models [217, 224]. Furthermore, these reviews do not focus their attention on travel time but survey the predictive models for all the traffic variables. In this chapter, an extensive survey of all the necessary concepts when modeling travel time is performed and a complete and innovative taxonomy of the existing methods for estimation and prediction of travel time is presented.

Moreover, although research on travel time modeling has been very extensive in the past few years, there are still many directions that future studies should follow in order to fill in the gaps present in the available literature. As such, one important future research line is aimed at enhancing the existing methods in order to adapt them to the practical requirements of ATIS. Some of the most relevant of these requirements are enumerated in the following paragraphs:

- **Non Recurrent Congestion:** As seen in Section 4.3, most travel time prediction models are not accurate in situations when traffic changes abruptly and unexpectedly. Furthermore, it is not reliable to assume that the accurate prediction of this type of congestion is always possible and, even if it were, it would be very difficult to predict its cause, its typology and its impact on traffic. However, an effort must be made in this direction and predictive

models developed in the future should at least be capable of rapidly detecting non-recurrent congestion. In addition, they should be flexible enough to immediately adapt to unexpected traffic situations as in [26, 57, 235, 244, 245].

- **Fusion of Data Sources:** Both in travel time prediction and estimation, very few methods are able to incorporate data from different types of sensors, and in real situations the data available from a unique type of sensor might not be sufficient to represent the traffic state adequately. Moreover, the use of interval detectors is not very common, but recently, the access to this type of rich and valuable data has been simplified and many ATIS have started to employ their users as probe vehicles, especially for modeling urban networks. In fact, in addition to collecting mobility data from them, the individual characteristics and behaviors of the drivers can be considered and included in the models. Furthermore, some ATIS such as Waze [139], have started to demand information such as the existence of a delay or an accident, its cause, its location, information about roads or maps, etc. from their users in an interactive manner to improve their models. All this rich type of information should be considered in future studies of travel time estimation and prediction.
- **Context Information:** Apart from pure traffic information, contextual data can be very valuable, especially in the case of prediction. Factors such as meteorology and special events are strong indicators of the traffic situation but they are generally not included in the models. This information is quite easy to obtain and should certainly be contemplated in future travel prediction models. Furthermore, as shown in [227], different areas might have different travel behaviours and characteristics which should be included in the modelling process.
- **Extension to entire road networks:** It can be observed that most of the presented travel time estimation and prediction methods are restricted to short road segments, or predefined trajectories and are not usually extended to road networks or urban roads. This is certainly an issue that should be considered in future works because an ATIS should be able to provide travel time measurements and prediction throughout the whole network. However, the adaptation of the travel time models to a whole road network is not always straightforward because the impact of adjacent road sections and other contextual factors, together with the issue of missing data in certain zones have to be considered in order to adequately capture the dynamics of the whole network.
- **Confidence Intervals:** Although there are some exceptions, most of the existing travel time models are centered on obtaining a unique mean travel time value. However, on occasions these values might not be accurate and so, a measure of the reliability of the travel time prediction or estimation is very helpful. Confidence intervals are especially useful for this purpose but there are very few studies that provide them. Future models should be able to quantify the reliability of their predictions and estimations in order to give more complete information to the travelers.

Fusion models and combination or hybrid models such as those presented in Sections 4.1.3 and 4.2.4 seem like a promising research direction for travel time prediction and estimation. On the one hand, they enable the combination of flexible models for unexpected situations with more rigid models for common traffic situations, which might be especially useful for predictive purposes. On the other hand, they can easily include data from different sensors as can be seen in Figure 4.1, which is necessary in both prediction and estimation. Furthermore, they can be designed to choose the most adequate option from a set of models in each situation, which will improve the accuracy of estimations and predictions and will permit a better description of the dynamics of a whole network.

However, to build such a combination model a deeper knowledge of the characteristics of the existing models is necessary, which leads to the last future research direction:

- **Development of a complete comparative study of travel time estimation and prediction methods**: This study, rather than aimed at finding one *best method*, should be aimed at providing a complete description of the methods and their qualities with the objective of facilitating the choice of one method in each situation.

Part III

Methodological contributions to time series data mining

5

The effect of time series clustering in travel time prediction

As we have concluded in the discussion of Chapter 4, fusion or hybrid models seem to be one of the most promising future research lines in both travel time prediction and estimation. In this case, we focus on the task of prediction and take the combined or hybrid models designed for this purpose as the baseline or starting point of this chapter and the rest of the dissertation.

Note that among the combined travel time prediction models proposed in the literature (see Section 4.2.4), an interesting data driven approach put forward only by a few authors consists in initially dividing the available traffic data into several clusters, and then building a different model for each group. This strategy enables the construction of a set of more specific models, each of which will be used to represent one of the clusters.

Particularly, in [244] a time series clustering approach is applied, where each instance to be clustered consists of the sequence of velocity measurements collected over an entire day. In most road networks, the presence of different daily patterns or *day types* is evident. As such, it seems logical to think that the application of a time series clustering pre-process is probably beneficial. Moreover, this approach has been frequently applied in other areas such as energy load forecasting [4, 113, 133].

Note that, to the best of our knowledge, in the area of travel time prediction only one publication adopts the strategy of time series clustering preprocessing and, to this point, the effect and possible benefits of this approach in the task of travel time prediction have not been extensively analyzed. For this reason and as a first step, in this chapter, we will present a preliminary analysis of the effect of time series clustering on several travel time prediction models. The importance of this preliminary analysis is crucial because it will lead to the two major contributions to time series data mining included in the next two chapters, aimed at solving two of the problems or difficulties that arise when applying these combined models.

The rest of the chapter is organized as follows. In Section 5.1 the travel time prediction models used in this chapter are introduced. In Section 5.2, the appearance of the paradigm of time series clustering in the travel time pre-

diction literature is analyzed. In Section 5.3 we propose a strategy to analyze the effect of time series clustering in travel time prediction models. The experiments and results are shown in Section 5.4 and, finally, some conclusions are presented in Section 5.5 which will serve as a motivation for the following chapters.

5.1 Short-term travel time prediction on highways

As introduced in Chapter 2, the objective of most travel time prediction models is to forecast the mean travel time for trajectories that will start at the moment the prediction is made (present) or in a given time interval in the future. For this purpose, traffic and contextual data available in the present, together with data from the past, is generally used as input (see Equation 2.2 for a mathematical definition). Departing from this general definition of travel time prediction models, in this chapter, we will restrict the analysis to models of the type:

$$TT_R(t+\Delta) = f_\Delta(t+\Delta, TT(t), TT(t-1), TT(t-2)) \quad \Delta = 0, 1, 2, 3, ... \qquad (5.1)$$

where the only variables used as input to the model are the time $(t+\Delta)$, and the travel time at the moment the prediction is made $(TT(t))$ and at the two previous time intervals $(TT(t-1), TT(t-2))$.

As explained in Chapter 4, f can be defined in different manners but, in this chapter, we focus on several popular data driven models that learn the shape of this function from a training dataset. Since these models have already been presented in Section 4.2.3, we will only mention some specific details associated to each of them:

- **Historical predictor:** the historical travel times collected at the same time are simply averaged to provide a prediction.
- **Linear regression:** The parameters of the model are learned by using the least squares estimator.
- **Regression tree:** CART trees are trained with the *rpart* package in R [204], with the parameters set in their default mode.
- **Support vector regression:** The SVR models are learned by using the *e1071* package in R [137] and selecting the radial basis kernel. All the remainder parameters of the *svm* function are left in their default settings.
- **Feed forward neural network:** Multilayer feed forward neural networks are trained with 4 input nodes, one output node and one hidden layer with 5 nodes. The training of the neural networks is carried out by using the *nnet* package in R [177] which uses the back propagation algorithm to obtain the weights that connect the different nodes. The parameters for the *nnet* function are all left in their default settings, except *linout*, which

is set to true to perform regression instead of classification, *skip*, which is set to *TRUE* to allow skip-layer connections from input to output, and *maxit*, which is set to 1000.

Note that, except for the historical predictor, a different model will have to be trained for each prediction horizon Δ.

5.2 Time series clustering and travel time prediction

In travel time prediction, the use of clustering appears only in a few works. These proposals initially apply clustering methods to identify different traffic states or patterns and subsequently build specific models for each case. For example, in [256] a rule based clustering method is combined with several neural networks. In [244] the data is clustered by using a Gaussian Mixture Model and, then, travel time predictions are provided by means of a combination of stochastic congestion map models and speed profiles. Finally, in [63], a K-means clustering algorithm is applied together with a set of genetic programming algorithms.

Although traffic data has a clear temporal character, in most of these studies time series clustering is not applied but, instead, the authors transform the data to enable the application of regular clustering techniques. For example, in [256] the traffic information collected in each time interval is considered as an independent data instance and the temporal correlation of the measurements is ignored. Also, in [63], the authors apply a rolling window and divide the time series of each day into different very short subsequences, each of which contains only 4 data points. Both of these approaches omit the presence of long term patterns, and only in [244] is the time series corresponding to each day considered as an entire instance. In this latter case, the clustering process provides a set of *day types* that have an overall similar behavior. This approach, which we have referred to as time series clustering, is the one we will analyze in this chapter and in the remainder of the dissertation.

Given the presence of different daily patterns in most roads, it is not far fetched to think that building a different model for each day type will result in a better overall performance. However, the use of time series clustering schemes in the area of travel time prediction is almost non-existent and so, no empirical analysis has been performed that will support this fact. Consequently, our goal is to analyze when, by applying a time series clustering pre-process, the travel time prediction models shown in Section 5.1 are indeed improved. Recall that in order to apply time series clustering to a given database, the practitioner must typically select a distance measure. As commented in Chapter 3, a vast portfolio of distance measures, specific to time series have been presented in the past few years. In spite of this, in the travel time prediction literature, to the best of our knowledge, the use of different time series distance measures is not present in any work and the great

majority simply use Euclidean distance. To demonstrate that the use of other distance measures can be beneficial, in this study we compare the performance of Euclidean distance with that of another very popular time series distance measure, Dynamic Time Warping [10] (see Section 3.2), which obtains better results in more than one occasion.

5.3 Comparison of procedures with and without clustering

In order to evaluate the effect of time series clustering when performing travel time prediction, we will analyze and compare the results obtained by two different procedures.

The first procedure (P1) is the common scheme that includes no clustering process and is used by most of the authors in the travel time prediction literature. In this case, for each prediction horizon Δ, all the available historical data is used together to build a unique model of each of the types shown in Section 5.1.

The second procedure (P2) is the one that we want to analyze in this chapter. As can be seen in Figure 5.1, in the training process of P2, before building the travel time prediction models, a time series clustering pre-process is applied. As such, the different days in the training set are clustered into "day types" based on their similarity and by using the K-medoids algorithm with a pre-defined K and a given distance measure d. Once the data is clustered, a separate model ($M_{i,\Delta}$) is trained for each cluster and prediction horizon. The idea is that different patterns will be represented by different, more specific, models that will yield more accurate results.

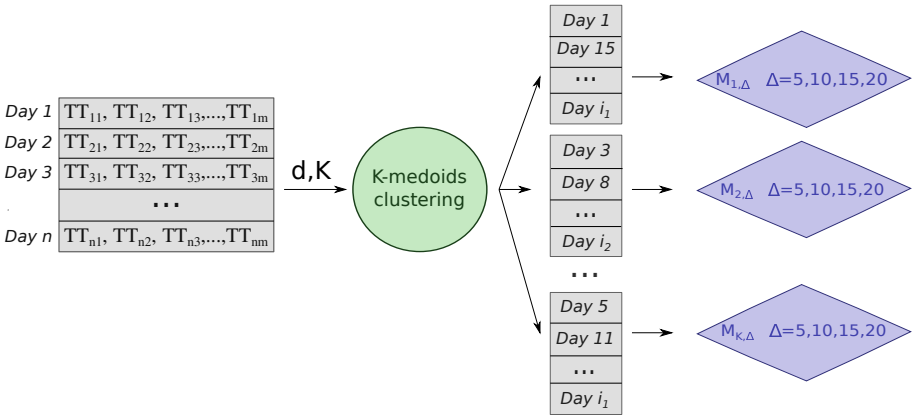


Fig. 5.1. The training phase of the P2 procedure.

The prediction of new travel time values using the models issued by P2 is slightly more complex than with P1. In this case, there is one model for each cluster and prediction horizon so, each time a new day begins, the first step is to decide to which cluster it belongs. In real travel time prediction problems, the data is collected continuously and so, the cluster assignment must be carried out online, using only the data collected up until the given timestamp in which the prediction is issued.

In this chapter, we apply a naive approach based on 1NN to solve this problem. Each time a new data point arrives, we compare the sequence collected until that moment with the representative or medoid of each cluster, and we assign the label of the nearest cluster. Of course, this comparison is carried out using the same distance measure d applied in the training phase. Once this is done, travel time values are predicted with the model trained for the chosen cluster and the desired prediction horizon.

5.4 Experimentation

In this section we summarize the experimentation performed to analyze and compare the performances of P1 and P2.

5.4.1 The data

Kaggle [1] is a platform which hosts data mining competitions proposed by diverse enterprises and researchers. In these experiments, we use the data presented by the The NSW Roads and Traffic Authority (RTA) [2] for a competition held in 2010 whose objective was to predict travel time values for the M4 freeway in Sydney. For this, the organization provided a set of travel time measurements collected, every 3 minutes, from February 2008 to mid-November 2010 in 61 consecutive sections of this freeway, 30 of which were eastbound and 31 were westbound. The data until July of 2010 was as historical data, and is the one used in this study. However, missing values because of lack of traffic or sensor failures are present in the data and, before applying the P1 and P2 procedures, the days with more than 10% of missing values have been eliminated.

From the 61 sections available, 8 are analyzed in this preliminary study. These routes are situated both in the eastbound and westbound directions, at the beginning, middle and end of the whole route and have been selected because they show different patterns and characteristics.

[1] Kaggle: http://www.kaggle.com
[2] The NSW Roads and Traffic Authority: http://www.transport.nsw.gov.au

5.4.2 Experimental setup

In the experimentation presented within this chapter, both procedures P1 and P2 are applied, as explained in Section 5.3, separately to the data from each section. Moreover, the prediction horizon Δ is chosen to be 5,10,15 or 20. Since the duration of each time interval is 3 minutes, this means that we focus on 15, 30, 45 and 60 minute ahead predictions.

Additionally, when applying the K-medoids algorithm, since the number of clusters K is unknown a priori, a set of different values $(2, 3, 4, 5)$ are tried out separately in different experimentation rounds. Finally, the effect of the two distance measures (d) introduced in Section 5.2 will also be examined separately. The silhouette is a well known clustering validity index that takes values from -1 to 1. The larger this value, the better the clustering result. As such, for each combination of K and d, 15 different random initializations of the K-medoids algorithm are tested in each case and the solution with the highest silhouette index is selected with the objective of avoiding local optima. Additionally, the silhouette values obtained from this process will later be used to study the effect that the quality of the clustering has on the travel time prediction results issued by P2.

Finally, in order to validate the performance of procedures P1 and P2, a train/test methodology is used with the idea of respecting the temporal character of the data. The first 70% of the days in the database are used for training and the remaining 30% will be used for testing. This results in 344 days for training and 147 days for testing. In this context, the performance of the different models is measured by means of the Mean Absolute Percentage Error (MAPE) obtained for the test instances:

$$MAPE = \frac{1}{n} \sum_{1}^{n} \frac{|TT - \hat{TT}|}{TT} \cdot 100 \tag{5.2}$$

where TT is the true travel time and $\hat{TT}$ is the predicted travel time.

5.4.3 Results

In Table 5.1, we show the results obtained from the experimentation. For P2, we only show the results for the combination of K and d that obtains the best mean MAPE result for all prediction horizons (Δ) and model types. The distance measure (d) and number of clusters (K) used to obtain these results are shown above the results as well as the silhouette obtained by the selected clustering.

From these results, the first obvious conclusion that we reach is that the most suitable parameters d and K are very specific to each database. Indeed, as can be seen in Figure 5.2, the results can change drastically depending on these two parameters. Specifically for the data used in this study, it can be seen in Table 5.1 that DTW seems to obtain better results than ED in many cases. However, in the travel time prediction literature, there are no studies

Table 5.1. Summary of MAPE results provided by P1 and P2 for different predictions horizons (Δ). For each route, the results in bold show the lowest MAPE result for each type of method and prediction horizon.

	Section=8, K=2, d=ED, Sil.=0,12								**Section=32, K=5, d=ED, Sil.=0,28**							
Method	**$\Delta=5$**		**$\Delta=10$**		**$\Delta=15$**		**$\Delta=20$**		**$\Delta=5$**		**$\Delta=10$**		**$\Delta=15$**		**$\Delta=20$**	
	P1	**P2**	**P1**	**P2**	**P1**	**P2**	**P1**	**P2**	**P1**	**P2**	**P1**	**P2**	**P1**	**P2**	**P1**	**P2**
HP	7,45	**6,03**	7,45	**6,17**	7,45	**6,30**	7,45	**6,44**	3,10	**1,33**	3,10	**1,35**	3,10	**1,36**	3,10	**1,36**
LR	**4,34**	4,35	5,96	**5,83**	7,13	**6,86**	8,03	**7,64**	0,96	**0,82**	1,00	**0,88**	1,04	**0,92**	1,13	**0,96**
RT	5,27	**4,82**	6,12	**5,42**	5,88	**5,67**	6,55	**5,88**	**0,89**	1,10	**0,95**	1,10	**1,01**	1,24	1,48	**1,44**
NN	4,27	**4,05**	5,95	**5,76**	8,58	**6,82**	7,99	**7,48**	0,95	**0,75**	1,13	**0,84**	1,02	**0,91**	1,25	**0,91**
SVR	**3,76**	3,78	4,61	**4,59**	5,26	**5,22**	5,86	**5,77**	**0,83**	0,96	**0,96**	1,00	1,05	**1,01**	1,18	**1,04**
	Section=19, K=4, d=DTW, Sil.=0,74								**Section=35, K=5, d=DTW, Sil.=0,40**							
Method	**$\Delta=5$**		**$\Delta=10$**		**$\Delta=15$**		**$\Delta=20$**		**$\Delta=5$**		**$\Delta=10$**		**$\Delta=15$**		**$\Delta=20$**	
	P1	**P2**	**P1**	**P2**	**P1**	**P2**	**P1**	**P2**	**P1**	**P2**	**P1**	**P2**	**P1**	**P2**	**P1**	**P2**
HP	192,2	**4,82**	192,2	**6,75**	192,2	**9,08**	192,2	**10,65**	2,47	**1,70**	2,47	**1,76**	2,47	**1,84**	2,47	**1,92**
LR	18,28	**2,43**	29,62	**3,19**	39,57	**4,14**	46,80	**5,01**	**1,05**	1,34	**1,32**	1,59	**1,60**	1,73	1,81	**1,80**
RT	3,03	**2,47**	3,10	**2,57**	3,12	**2,66**	3,15	**2,87**	1,80	**1,06**	1,84	**1,10**	2,00	**1,15**	2,58	**1,23**
NN	6,34	**2,40**	29,18	**2,62**	6,11	**2,73**	4,76	**3,33**	**1,00**	1,27	**1,20**	1,55	1,56	**1,44**	**1,63**	1,73
SVR	9,99	**2,31**	9,77	**2,60**	9,21	**2,83**	7,94	**2,94**	**0,82**	**0,82**	**0,88**	0,89	0,95	**0,93**	1,00	**0,97**
	Section=27, K=5, d=DTW, Sil.=0,82								**Section=36, K=5, d=DTW, Sil.=0,60**							
Method	**$\Delta=5$**		**$\Delta=10$**		**$\Delta=15$**		**$\Delta=20$**		**$\Delta=5$**		**$\Delta=10$**		**$\Delta=15$**		**$\Delta=20$**	
	P1	**P2**	**P1**	**P2**	**P1**	**P2**	**P1**	**P2**	**P1**	**P2**	**P1**	**P2**	**P1**	**P2**	**P1**	**P2**
HP	2,55	**0,72**	2,55	**0,72**	2,55	**0,75**	2,55	**0,77**	3,56	**1,18**	3,56	**1,22**	3,56	**1,26**	3,56	**1,29**
LR	1,16	**0,62**	1,19	**0,65**	1,25	**0,68**	1,38	**0,71**	1,39	**0,85**	2,10	**1,08**	2,71	**1,24**	3,13	**1,35**
RT	0,59	**0,45**	0,59	**0,46**	0,63	**0,46**	0,69	**0,47**	1,07	**0,83**	1,30	**0,89**	1,57	**0,98**	1,74	**1,01**
NN	1,08	**0,62**	1,17	**0,56**	1,33	**0,57**	1,36	**0,65**	1,24	**0,81**	1,97	**1,02**	2,63	**1,22**	2,91	**1,33**
SVR	0,73	**0,55**	0,95	**0,55**	1,14	**0,57**	1,32	**0,59**	1,38	**0,88**	1,53	**0,94**	1,65	**0,98**	1,74	**1,02**
	Section=29, K=2, d=DTW, Sil.=0,88								**Section=46, K=4, d=DTW, Sil.=0,31**							
Method	**$\Delta=5$**		**$\Delta=10$**		**$\Delta=15$**		**$\Delta=20$**		**$\Delta=5$**		**$\Delta=10$**		**$\Delta=15$**		**$\Delta=20$**	
	P1	**P2**	**P1**	**P2**	**P1**	**P2**	**P1**	**P2**	**P1**	**P2**	**P1**	**P2**	**P1**	**P2**	**P1**	**P2**
HP	4,44	**2,35**	4,44	**2,39**	4,44	**2,47**	4,44	**2,52**	7,87	**5,02**	7,87	**5,14**	7,87	**5,11**	7,87	**5,39**
LR	1,52	**0,93**	1,99	**1,10**	2,25	**1,19**	2,44	**1,25**	**4,34**	4,48	5,70	**5,67**	6,73	**6,55**	7,51	**7,28**
RT	2,04	**1,45**	**1,55**	1,87	**1,44**	2,01	**1,38**	1,97	3,94	**3,64**	4,89	**4,53**	5,72	**4,80**	6,30	**5,57**
RR	**1,14**	1,31	2,49	**1,74**	1,99	**1,18**	1,72	**1,22**	4,00	**3,77**	**5,52**	**5,52**	6,64	**5,80**	7,43	**7,22**
SVR	1,09	**0,87**	1,18	**0,96**	1,24	**1,07**	1,31	**1,21**	3,47	**3,35**	3,88	**3,81**	4,12	**4,07**	**4,18**	4,23

which attempt to use different types of distance measures and most of them focus on rigid distances such as ED. Indeed, most studies try to reduce the problem to a common clustering problem without taking into account that

temporal data has very specific characteristics that can only be dealt with specific distance measures [228]. With this, we conclude that clustering can be beneficial but, the selection of an adequate K and, especially, a suitable distance measure must be considered.

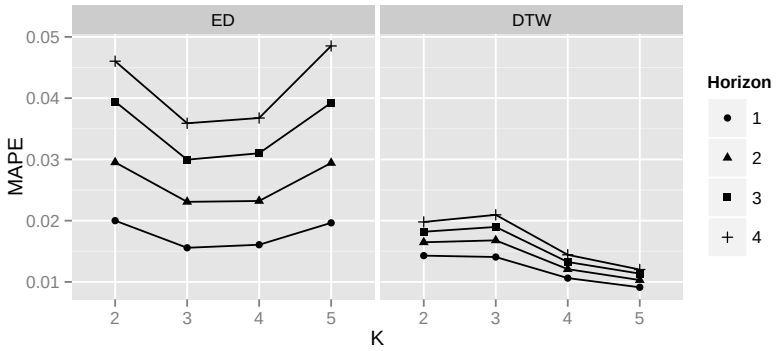


Fig. 5.2. MAPE obtained for all methods in route 36 for the two distance measures, different values of K and different prediction horizons.

The second fact that we would like to underline is that naive historical predictors seem to benefit more from the clustering process than other more complex models. With the historical predictor, P2 obtains much better results than P1 in all cases. This is especially interesting for long term prediction, where extremely simple historical models are on occasions competitive with some other more complex models (see results for section 8 and 32, 36 and 46).

As a final point, it seems reasonable to think that the quality of the clustering has an effect on the results issued by P2. In this sense, the silhouette can be used as a predictor of the performance of P2 to a certain degree. If this index is close to 1 (see sections 19, 27 and 29 in Table 5.1), the results for P2 are generally better than those for P1, specially as the prediction horizon increases. As the silhouette becomes lower, the results of P2 become worse and P2 does not always improve the results of P1. When it does, it is mostly in the case of longer prediction horizons and the difference is generally smaller that that obtained for sections with higher silhouette values (see sections 8, 35 and 46 in Table 5.1). Since the computational burden of P2 is greater than that of P1, in these last cases, the clustering process should not be carried out lightly.

Also, although the silhouette can give us some clues about the performance of P2, the clustering with the highest silhouette does not always provide the best results for P2, especially when the silhouette values are low. For example, it must be noted that the selection of many clusters, in some cases, indicates the presence of many noisy days in the data. Isolating these series in separate

clusters accounts for better results in P2 than in P1, but the quality of the clustering, in these cases, is not good. This happens for example in section 32, where the silhouette is not very high but the results for P2 outperform the ones in P1 in many cases. As such, this index alone should not be used to decide which clustering will provide the best result for P2, and visual inspection of the data or application of other quality measures is highly recommendable.

5.5 Conclusions

In this chapter we have analyzed the effect of applying a time series clustering pre-process for short term travel time prediction. The first conclusion that we extract from this analysis is that, as expected, time series clustering can be beneficial for travel time prediction. Indeed, it has been shown in our experimentation that this type of clustering is especially beneficial when using naive methods based on historical means. Historical models are simple and efficient and so this conclusion should be taken into account especially for longer term predictions, when historical predictors show their best performance.

However, even if time series clustering shows some benefits, it is clear from our experimentation that it should not be applied lightly, without a previous profound study of the data. The clustering process adds another source of error to the model and is computationally more expensive. Therefore, if there are no clear patterns present in the data or the quality of the clustering is bad, it may not be worth it or may even affect the results negatively. In this line, the distance measure applied strongly affects the travel time predictions and, therefore, it should be individually and carefully selected for each database. In view of this, in Chapter 6, a method that automatically selects the most suitable distance measure(s) from a set of candidates is proposed.

Additionally, the assignment of new days to a given cluster can be understood as a problem of early classification of time series [237]. In this chapter, a naive approach has been adopted to solve this issue, but in Chapter 7 a more sophisticated early classification method will be presented and validated.

6

A contribution to distance measure selection in time series clustering

In the past few years, clustering has become a popular task associated with time series. However, the clustering of time series databases typically requires the definition of a distance measure which will estimate the level of similarity or dissimilarity between time series. As demonstrated on more than one occasion in the literature, the choice of a suitable distance measure is crucial to the success of the clustering process. Particularly, in the specific case of travel time prediction, we have shown in Chapter 5 that, effectively, the selection of a suitable distance measure has a deep impact on the obtained forecasts.

Previous experiments [228] suggest that the specific characteristics of each database make some distance measures more suitable than others. However, the choice of an adequate similarity measure is not easy because it is necessary to find a relationship between the characteristics of the time series databases and the properties of the different distance measures.

To the best of our knowledge, in the context of time series clustering, no formal methodology is available in the literature that supports the choice of one distance measure over another. As such, the common strategy is to experiment with a set of different distances and to choose the best performer. Due to the large number of available time series distance measures and the time complexity of many of them, this strategy is computationally very expensive and intractable in practice.

With the objective of simplifying this task, we propose a multi-label classification framework that provides the means to automatically select the most suitable distance measures for clustering a time series database. This classifier is based on a novel collection of characteristics that describe the main features of the time series databases and provide the predictive information necessary to discriminate between a set of distance measures. In order to test the validity of this classifier, we conduct a complete set of experiments using both synthetic and real time series databases and a set of 5 common distance measures. The positive results obtained by the designed classification framework indicate that the proposed methodology is useful to simplify the process of distance selection in time series clustering tasks.

This chapter is organized as follows. In Section 6.1 we present the set of features that will be used to describe the time series databases. Section 6.2 introduces our classification framework, and Section 6.3 presents a method to define the class labels of the training set that will be used to build this classifier. We empirically validate our ideas in Section 6.4, and Section 6.5 provides a summary of the most relevant results. Finally, in Section 6.6 we extract the main conclusions and propose some insights on future work.

6.1 Time series database characterization

In this section we define a novel set of features that quantifies certain characteristics of time series databases and will be used as predictive information for choosing a suitable distance measure. A summary of these characteristics is shown in Table 6.1. It should be noted that these characteristics do not describe a single time series but an entire database of time series. For this reason, for features describing a single series, we use a set of statistics to generalize the result to the entire database: we use the mean, the standard deviation and the 5% and 95% percentiles in most cases, together with some additional statistics for some particular features, which we comment on separately in each case.

The databases considered in this chapter only contain series of the same length so we will not study the cases with series of different lengths or sampling rates in detail. However, the calculation of the features can be extended directly to databases of this type in most cases. If specific techniques are necessary, we will make an explicit mention.

6.1.1 Dimensionality of the database

It has been empirically demonstrated in the past few years that, in time series classification, the dimensionality of the time series has an impact on the accuracy of some similarity measures. For example, the accuracy of rigid distance measures such as Euclidean Distance (ED) improves as the number of time series in the database increases [228]. Besides, not all distance measures provide good results when working with very long time series [64]. These claims have not been assessed or analyzed in the clustering framework, but could also be relevant in some cases. As such, the number of time series in the database and the average length of these sequences are the first two variables that we will use to describe a time series database.

6.1.2 Shift

It is very common that the time series in a database are shifted in the time axis and, as commented in Section 3.2, not all distance measures are able to

deal with this type of similarity. For this reason, quantifying the level of *shift* in a time series database can be highly relevant.

To calculate the shift present in a database of time series $D = \{X_1, X_2, ..., X_N\}$, where X_i is a time series, we calculate the correlation of each pair of time series in the database at different lags and select the time lag in which this correlation yields its maximum. In this case, we define the correlation value of two series $X = \{x_0, x_1, ..., x_{n-1}\}$, $Y = \{y_0, y_1, ..., y_{n-1}\}$ at lag t using a slightly modified version of the common cross correlation formula, which avoids penalization at higher lags and considers both the positive and negative lags:

$$r_t(X,Y)=\begin{cases}\dfrac{\frac{1}{n-t}\sum_{i=0}^{n-1-t}(x_i-\bar{x}_+)(y_{i+t}-\bar{y}_+)}{\sqrt{Var(x_+)}\sqrt{Var(y_+)}} & \text{if } t=0,1,2,3,...\\ \dfrac{\frac{1}{n-t}\sum_{i=0}^{n-1-t}(y_i-\bar{y}_-)(x_{i+t}-\bar{x}_-)}{\sqrt{Var(y_-)}\sqrt{Var(x_-)}} & \text{if } t=-1,-2,-3,...\end{cases} \tag{6.1}$$

where $\bar{x}_+$ and $\bar{y}_+$ are the mean values of $X_+ = \{x_0, x_1, ..., x_{n-1-t}\}$ and $Y_+ = \{y_t, y_{t+1}, ..., y_{n-1}\}$ respectively, and $Var(x_+)$ and $Var(y_+)$ refer to the sample variances of these series. For negative lags, the same is applied for series $X_- = \{x_t, x_{t+1}, ..., x_{n-1}\}$ and $Y_- = \{y_0, y_1, ..., y_{n-1-t}\}$, respectively. It must be noted that Equation 6.1 is only directly applicable for series that are sampled in the same timestamps. Nevertheless, this calculation may be extended by applying more specific methods that obtain correlation values between unevenly sampled series [173].

Based on equation 6.1, we can easily compute the maximum correlation value and its corresponding lag value:

$$r_{max}(X_i, X_j) = \max_t(r_t(X_i, X_j)) \tag{6.2}$$

$$lag(X_i, X_j) = \operatorname*{argmax}_t(r_t(X_i, X_j)) \tag{6.3}$$

At this point, it must be stressed that, while it is interesting to quantify the *shift* between a pair of similar time series, this operation does not make sense for series with completely different shapes. In this context, a simple method must be provided that attempts to discard the correlation and lag values obtained from dissimilar pairs of time series. The solution we propose is to simply consider the pairs of series with a high enough correlation value, and to provide the average of their absolute lag values as an approximation of the total shift level present in the database:

$$\text{Shift level} = \left(\frac{1}{|S_{\delta^*}|}\sum_{(X_i,X_j)\in S_{\delta^*}}|lag(X_i, X_j)|\right)\times\frac{100}{l} \tag{6.4}$$

where l is the mean length of the time series of the database. As can be seen in the formula, we define the shift value of a database as a percentage of l, directly enabling the comparison of the values obtained for different datasets. Also, we define S_{δ^*} as follows:

$$S_{\delta^*} = \left\{ (X_i, X_j) \mid i > j \ \wedge \ \frac{r_{max}(X_i, X_j) - r_*}{r^* - r_*} > \delta^* \right\} \tag{6.5}$$

$$r^* = \max_{X_i, X_j \in D} r_{max}(X_i, X_j) \tag{6.6}$$

$$r_* = \min_{X_i, X_j \in D} r_{max}(X_i, X_j) \tag{6.7}$$

We note that in the calculation of S_{δ^*}, each pair of series in D is only considered once. The reason for this is that the value of r_{max} is equal for (X_i, X_j) and (X_j, X_i), and the value of lag only differs in the sign (see equations 6.1,6.2 and 6.3). As can be seen in equation 6.4, in the calculation of the shift feature, only the absolute phase difference between two series is of interest and, as such, it is not necessary to check the pairwise correlations between the series in both senses.

In order to define the threshold δ^* of equation 6.5, we order the normalized correlation coefficients between the series in the database into an ascending sequence R. We recall that this study is set in a clustering framework and therefore the existence of several underlying groups in the databases is assumed. The series from the same cluster are supposed to be similar to each other, whereas those that belong to different clusters are dissimilar in some sense. In this context, if all the clusters are sufficiently different from each other, we will observe a clear jump that will separate the high correlations (intra-cluster) from the low correlations (inter-cluster), as shown in Figure 6.1.

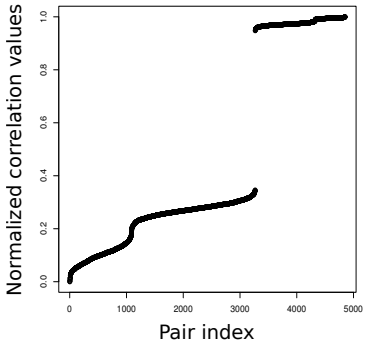


Fig. 6.1. Normalized correlations between pairs of series in a 3-cluster database.

Based on this, the first step is to calculate the first point after this jump occurs:

$$\delta = \max\{\, \hat{r}_i \in R \mid \ (\hat{r}_i - \hat{r}_{i-1}) > 0.01\} \tag{6.8}$$

If the clusters are different enough from each other, δ^* will be directly defined by δ. However, the clusters in a time series database are not always very different from each other and in some cases the jump may be situated at very low correlation values or may not be present at all. Also, it is possible that the number of correlation values above this jump is too low to obtain significant information. As such, if δ is lower than 0.95, or if the pairs of series with a correlation value above δ are less than 5% of all the pairs of series, we set δ^* to a fixed value of 0.95:

$$\delta^* = \begin{cases} \delta & \text{if } \delta \geq 0.95 \text{ and } |S_\delta| \geq 0.05 \cdot \frac{N(N-1)}{2} \\ 0.95 & \text{otherwise} \end{cases} \tag{6.9}$$

The previous formulation includes some predefined thresholds and parameters such as the lower limit of δ^*, set to a fixed value of 0.95. It must be noted that we have set these parameters, and any others that appear in the formulations of the other characteristics defined in this section, based on the preliminary experiments explained later in Section 6.1.10.

It is easy to prove that the correlation, as formulated in Equation 6.1, between two identical but shifted linear series is always 1 and does not depend on the phase difference. Because of this, the procedure explained above is not useful to find the lag between two linear series. We consider a series linear if the Adjusted Root Mean Squared Error (ARMSE) of its fitted linear regression is lower than a user defined threshold, 0.1 in this case. If both of the series that are being compared (X and Y) are linear, the lag between them will be given by the median (med) of the difference between their fitted linear regressions $\hat{X} = \{a \cdot t + b, \ \ t = 0, ..., n-1\}$ and $\hat{Y} = \{a' \cdot t + b', \ \ t = 0, ..., n-1\}$:

$$lag(X, Y) = med(\{(a - a')t + (b - b'), \ \ t = 0, 1, ..., n-1\}) \tag{6.10}$$

The correlation between the two original series controls whether they come from the same linear trend or not. As in the previous case, the correlation between a pair of series must be higher than δ^* in order to be considered, so we will only keep the pairs of linear series with a similar slope. In the special case of constant series, the correlation can not be calculated. However, this is not problematic because constant series are invariant to shift, and do not give any useful information about the shift level of the database. As such, we will discard them before calculating the shift level of the database.

6.1.3 Correlation

Another feature that can be used to describe a time series database is the 95% percentile of the $r_{max}(X_i, X_j)$ values obtained in the calculation of the

shift (see Section 6.1.2). This feature analyzes the level of correlation present in the database after removing the shift. It is a quite general feature that can be affected by various factors. For example, factors such as noise, outliers or local warp present in the database will be responsible for lower correlation values. Only some of the distance measures are able to deal with these issues and, therefore, this characteristic can be useful to discriminate between them.

In the same manner, the 5% percentile, the mean and standard deviation of the $r_{max}(X_i, X_j)$ values can be calculated. Contrary to the 95% percentile, these values provide information about the level of similarity between the clusters present in the database.

6.1.4 Seasonality

A time series has a seasonal component if it contains a pattern that repeats itself periodically. It is a basic characteristic of time series and it is interesting to study its influence in the performance of different distance measures. The methodology used to quantify this characteristic is based on the framework used in [229].

The first step is to find out whether the time series contain a seasonal component or not. If the collection frequency of the series is unknown, we use the spectral density function to find the frequency which has the greatest contribution to the variance of the series. If the contribution of this maximum value is higher than a user specified threshold (10, in this case), then the series has a seasonal component in that frequency. If no such frequency is found, we declare the seasonal component of the series null.

Finally, if the seasonal component is not null, we decompose the series using the Seasonal-Trend decomposition procedure based on local polynomial regression (STL) [40] and then quantify the seasonality in the following manner:

$$\text{Seasonality level} = 1 - \frac{Var(X_t - S_t - T_t)}{Var(X_t - T_t)} \tag{6.11}$$

where X_t is the time series and S_t and T_t are its seasonal and trend components respectively.

To characterize an entire database, in addition to the basic statistics (mean, standard deviation and 5% and 95% percentiles) of all the seasonality levels, we save the percentage of series in the database that have a seasonality level higher than 0.5 as a feature. This last feature represents the number of series which have an important component of seasonality.

6.1.5 Trend

The trend of a time series is described as the long term behavior that does not include seasonal or random effects. The first way to characterize the trend

level of a time series is by following the same methodology presented in the previous section:

$$\text{Trend level} = 1 - \frac{Var(X_t - S_t - T_t)}{Var(X_t - S_t)} \tag{6.12}$$

This value will be calculated for each series and summarized using the four basic statistics (mean, standard deviation and the 5% and 95% percentiles).

Moreover, to describe the trend more in detail, we propose the following additional features: number of local maxima (peaks) and minima (valleys), mean distance between the peaks (and valleys) in the time domain normalized by the length of the time series, mean distance between the values of the peaks (and valleys) normalized by the maximum and minimum values of the series, number of jumps higher than $0.1 *$ (max value of series $-$ min value of series) and mean size of jumps normalized by the maximum and minimum values of the series. We identify the peaks and valleys in the time series by means of the *findPeaks* and *findValleys* functions in the *quantmod* package of R [181]. These functions are based on finding the timestamp in which the trend of the series changes from positive to negative, and outputting the next timestamp after the peak/valleys occurs. In order to obtain the exact peak value, we take the points immediately previous to those given by the *findPeaks* and *findValleys* functions.

To describe the entire database, we only consider the mean values in this case to avoid an excessive number of characteristics.

6.1.6 Noise

The amount of noise in the time series database may be relevant when choosing one distance over another, because some measures have a more robust behavior when noise is present [64]. To quantify the noise in a database, the idea is to remove it from each time series by applying several techniques reviewed in [108], and to calculate the standard deviation of this removed part. After calculating the noise level for each time series, we obtain an overall noise value by using the basic statistics (mean, standard deviation and the 5% and 95% percentiles). Additionally, we also save the median in order to reduce the effect of series with a strongly deviating noise level.

The first technique we have used to quantify the noise level of a time series database is a moving average filter. This method moves a fixed-sized window along the series and substitutes each point with the average of the values in the window. In order to try to find a balance between eliminating local fluctuations and over-smoothing the series, we have chosen the window size to be 3% of the length of the series in the database. For short series, where %3 of the length is less than 2 points, we set the window size to a fixed value of 2 points.

The second technique is an exponential moving average filter. In this case, each point is replaced by the weighted average of all its previous points in

the time series. The weights are defined by an exponential function that decreases as the data points get older. We have chosen a parameter of 0.7 after preliminary experiments.

The third method is called linear Fourier smoothing. The idea is to calculate the Discrete Fourier Transform of the time series and to eliminate the components that correspond to frequency values higher than a user predefined cut-off frequency. By using the inverse transformation, we obtain the smoothed time series and the noise can be quantified. In this case, we remove components referring to the highest 10% of the frequencies to try to smooth only the fast fluctuations in the time series.

The last two denoising techniques applied are variations of the nonlinear wavelet shrinkage, which is based on the Discrete Wavelet transform. Unlike the Fourier Transform, the Wavelet transform uses wavelet functions, in this case D2 Daubechies and C6 Coiflets, which are wave-like oscillations but of finite length. We have implemented this denoising method with the *wavshrink* function of the *wmtsa* package [45] in R by modifying only the parameter corresponding to the choice of the wavelet functions. With its default parameters, this R function calculates the decimated discrete wavelet transform of the series and then, the coefficients whose magnitude is below a threshold *delta* are set to 0. This noise threshold is calculated by using the universal threshold function which is defined so that if the original time series only contained Gaussian noise, then the hard thresholding function would correctly eliminate all the noise contained in the series. Once the shrinkage is performed, by means of a synthesis operation, the new wavelet coefficients are transformed again into the time domain, obtaining the new de-noised series.

6.1.7 Outliers

Outliers, or data points that deviate notably from the rest of the data, greatly affect some distance measures [64] and should be included in the characterization of time series databases. For this reason, we calculate the proportion of outliers in each series with two different methods. Next, we generalize the results to the entire database as in the case of the noise.

The first method consists of building a box-plot for each time series and calculating the percentage of points that lie outside the whiskers of the plot. In order to remove the effect of trend and seasonality in the distribution of the data, the application of this technique must be performed after the series is detrended and deseasonalized using the STL method as explained in Section 6.1.4. Standard box-plots are designed to represent symmetric distributions and time series do not always fulfill this condition. Therefore, we use the adjustment proposed in [94] and implemented in the *robustbase* package [179] in R to build the plots. This approach makes use of the medcouple coefficient (MC) to measure the skew and modify the limits and whiskers of the plot to better capture the behavior of a non-symmetric distribution. Precisely, the MC coefficient takes values between -1 and 1, where the positive

values correspond to right-skewed distributions and the negative values to those that are left-skewed. Symmetric distributions take a MC value of 0. In the *robustbase* package, the whiskers of the box-plot are defined based on an exponential function of the MC coefficient. Specifically, all points outside the intervals shown in Equation 6.13 will be catalogued as outliers.

$$\begin{cases} \left[Q1 - 1.5e^{4MC} \cdot IQR\,,\, Q3 + 1.5e^{3MC} \cdot IQR\right] & if \quad MC \geq 0 \\ \left[Q1 - 1.5e^{3MC} \cdot IQR\,,\, Q3 + 1.5e^{4MC} \cdot IQR\right] & if \quad MC < 0 \end{cases} \tag{6.13}$$

where Q1 and Q3 are the first and third quartiles respectively, MC is the medcouple coefficient and IQR is the interquartile range. For symmetric distributions, Equation 6.13 is reduced to the formulation of a common box-plot.

The second method is an adaptation of the Local Outlier Factor (LOF) method presented in [18] to time series data. LOF is based on calculating the local density of each point by using the proximity to its neighbors and finding points that have a notably lower density than their neighbors. The method builds a ranking of the data points, and the outliers are identified as the points that have high LOF factors. This algorithm is defined for regular datasets and, therefore, a neighborhood is defined as a set of points that have similar values. However, in time series, the points are temporally correlated with each other and the definition of neighborhood must capture this behavior. As such, we have combined the LOF method with a sliding window of size 11. In each window, the LOF technique is applied by using the *DMwR* package [206] in R and we identify the points with a score higher than 5 as outliers for that window. In order to discard jumps or sudden changes, we finally consider a point an outlier if we have identified it as such in the majority of the windows that contain it (90% in this case, based on exploratory trials).

6.1.8 Skewness and kurtosis

These two characteristics are related to the shape of the probability distribution of the data and can provide some insights about the shape of the time series. Skewness is a measure of the asymmetry of the data around the mean value and is calculated as follows:

$$\text{Skewness} = \frac{\sum_{i=0}^{n-1} (x_i - \bar{x})^3}{n\sigma^3} \tag{6.14}$$

$\bar{x}$ is the mean of the time series and σ is the standard deviation. On the contrary, kurtosis is a measure of how peaked or flat the probability distribution of the data is:

$$\text{Kurtosis} = \frac{\sum_{i=0}^{n-1} (x_i - \bar{x})^4}{n\sigma^4} - 3 \tag{6.15}$$

These features only represent one time series and, so, we generalize them in the usual way (mean, standard deviation and 5% and 95% percentiles).

Table 6.1. Summary of all the characteristics. %95Q and %5Q correspond to the %5 and %95 quantiles.

Characteristic	**Statistics**	**Quantification method/s**	**Number of variables**
Dimension of database		Number of series in the database	2
		Mean length of series in database	
Shift	Mean	Method based on cross correlations	1
Correlation	Mean, St Dev, %95Q, %5Q	Cross-correlations between series	4
Seasonality	Mean, St Dev, %95Q, %5Q	STL decomposition	5
	% of series with high seasonality		
Trend	Mean, St Dev, %95Q, %5Q	STL decomposition	4
Trend	Mean	Number of peaks	8
		Mean distance in time between peaks	
		Mean distance between values of peaks	
		# of valleys	
		Mean distance in time between valleys	
		Mean distance between values of valleys	
		# of jumps	
		Mean size of jumps	
Noise	Mean, St Dev, %95Q, %5Q, Median	Moving average	25
		Exponential Smoothing	
		Fourier smoothing	
		Non-linear Wavelet Shrinkage with D2	
		Non-linear Wavelet Shrinkage with C6	
Outliers	Mean, St Dev, %95Q, %5Q, Median	Adjusted box-plots	10
		LOF adapted to time series	
Skewness	Mean, St Dev, %95Q, %5Q	Skewness formula	4
Kurtosis	Mean, St Dev, %95Q, %5Q	Kurtosis formula	4
Autocorrelation	Mean, St Dev, %95Q, %5Q	Box-Pierce statistic	4
Total			**71**

6.1.9 Autocorrelation

We obtain the level of autocorrelation of a single series X, from a modification of the Box-Pierce Statistic [229]:

$$AC(X) = \frac{1}{h}\sum_{i=1}^{h} r_i(X)^2 \tag{6.16}$$

where $r_i(X)$ is the autocorrelation value of the series at lag i, and h is chosen to be 15% of the length of the series. As the randomness in the series grows, the temporal correlation becomes weaker and the Box-Pierce statistic provides lower values. We generalize the autocorrelation values as usual (mean, standard deviation and 5% and 95% percentiles).

6.1.10 Parameter tuning in the definition of the characteristics

As commented previously, most of the characteristics presented above to describe time series databases require the definition of some parameters or thresholds. We have chosen these parameters after a set of exploratory experiments carried out by using the Synthetic control [162] and the CBF [103] synthetic databases. Firstly, we have modified the characteristics of these two databases by tuning them synthetically and univariately and obtaining a set of databases with different and previously known features. The details on how to artificially tune the features of these time series databases can be studied in Appendix A. Secondly, we have measured the characteristics of these artificially generated databases by using the methods proposed in the previous sections, using some different parameter combinations. In a last step, we choose the parameters that yield the most accurate correspondence between the synthetically introduced *ground truth* characteristics and those that are measured. The combination of parameters that we have considered is not exhaustive and thus, the method does not ensure optimality in any way. In this context, more sophisticated parameter search strategies could be applied, which could improve the results.

6.2 Automatic distance selection for time series databases

In this section we use the characteristics defined in Section 6.1 to create a multi-label classifier that is able to automatically choose the most suitable distances from a set of candidates. The proposed classification framework is shown in Figure 6.2: the predictive variables are defined by the features described in Section 6.1 and the class labels declare which distance measures, from the set of options, are the most appropriate for clustering each database.

The reason for choosing a multi-label framework is that if we compare the behavior of different distance measures when clustering a time series database, it is common to find measures that yield equally good clustering results. Instead of providing only one solution, the multi-label framework enables the selection of the entire set of most suitable distance measures for each time series database. This increases the value of the distance selection tool because it allows the user to choose between equally good solutions based on the computational efficiency of the distance measures or other possible reasons.

After various preliminary tests, we have chosen two algorithms to train the proposed multi-label classifier, although any other choice could be used within the framework directly. The first is based on the *Classifier Chain algorithm (CC)* [172], which decomposes the multi-label problem into an ordered chain of binary classification problems [250]. Each of these binary classification problems corresponds to one of the class labels, and incorporates the class predictions from all the previous classifiers in the chain into its feature space. Since the single stand-alone CC classifier is strongly affected by the order of the labels, an ensemble approach is used in this case, as proposed in [172]: the *Ensemble Classifier Chain (ECC)* method. The second multi-label classifier used in this chapter is the *Random-k-labelsets classifier (RkL)* [208], which transforms the multi-label problem into a multi-class framework.

Both these methods are high-order problem transformation methods, because they transform the multi-label problem into a set of simpler more common classification tasks, which consider complex relations between all the labels. As such, they require the selection of an algorithm that solves the underlying classical classification problems. Pruned C4.5 trees [165] have been

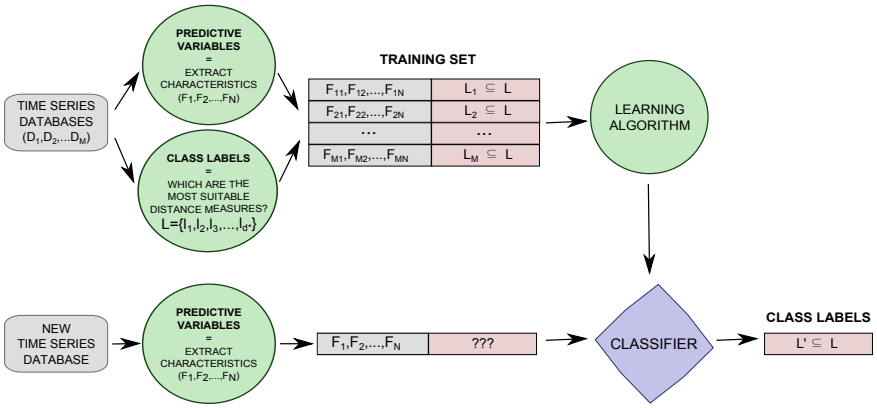


Fig. 6.2. Multi-label classification framework for the selection of the most suitable distance measure to cluster a time series dataset. If $d*$ candidate distance measures are considered and L is the label set that includes all these measures, a subset L_i of L is assigned to each time series database which represents the most suitable measures for the given dataset.

chosen because of their inherent feature selection ability, which adapts perfectly to this framework because it provides the means to automatically select a relevant set of features for each underlying classification problem.

In order to train these multi-label classifiers, we need the predictive variables and class labels of the training set of time series databases. We can obtain the predictive variables directly by simply calculating the features introduced in Section 6.1. However, the definition of the class labels is not so simple.

In a typical classification task, the class values of the set of training instances are usually provided by a set of experts or by the data generation process itself. In our case, recall that the class labels declare which distance measures are the most suitable to cluster each time series database. We do not have the labels of any publicly available time series datasets and, consequently, we will have to calculate them for a set of training instances by assessing and comparing the performance of the candidate set of distance measures when clustering them. This can be done if the ground truth clustering is known in advance, which is the case for the databases used in this study. However, to the best of our knowledge, there is no complete and statistically sound framework to perform this evaluation in the time series clustering context. In the following section, we propose a procedure to carry out this task.

6.3 Selection of the most suitable distance set

As mentioned in the previous section, in order to build the training set, we must define the class labels for each time series database by selecting the most suitable distance measures from the set of options. For this, we will replicate a common clustering procedure where different clustering solutions will be built, evaluated and compared. In the following sections, these three steps are explained in more detail:

6.3.1 Clustering phase

In this first phase we cluster each training time series database in various ways by combining different options of distance measures, parameter settings and clustering algorithms (see Figure 6.3):

- **Distance measures**: We select a set of candidate distance measures from which the most suitable will be chosen. Note that, the larger the candidate distance set, the more complex the multi-label classification problem will become.
- **Parameters for the distance measures:** If the distance measures require the selection of a parameter, we perform the clustering separately for each parameter option.

- **Clustering algorithms:** We use several clustering algorithms to mitigate the influence that these may have on the performance of the distance measures. We choose the partitional K-medoids clustering algorithm with 15 different initializations and the agglomerative hierarchical algorithm with different options of linkage criteria: maximum or complete linkage, minimum or single linkage, mean linkage (UPGMA) or Ward's criterion.

Given the characteristics of the databases used in this study, the number of clusters is known beforehand and will be used directly.

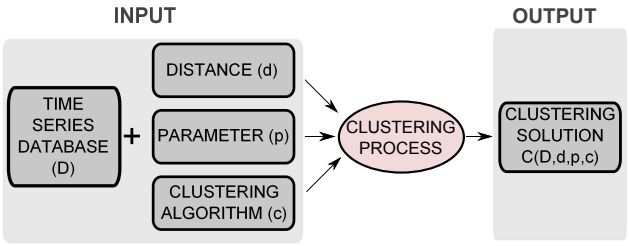


Fig. 6.3. Clustering phase of the most suitable distance measure selection process.

6.3.2 Evaluation phase

In this second phase, we evaluate all the clusterings performed in the previous stage in order to enable the comparison between them. As aforementioned, the ground truth clustering for all the databases that have been used in this study is known. This enables the calculation of the $\mathcal{F}$-measure for clustering, which is used as an evaluation metric in this process [226]:

$$\mathcal{F}(C^*, C) = \frac{1}{N} \sum_{i=1}^{k^*} N_i \max_{j=1,\ldots,k} \left\{ \frac{2|C_i^* \cap C_j|}{|C_i^*| + |C_j|} \right\} \tag{6.17}$$

where N is the number of series in the database, N_i is the number of time series in the i-th cluster of the ground truth solution, k^* is the number of clusters present in the ground truth clustering (C^*) and k is the number of clusters in the clustering that is to be evaluated (C). C_i^* and C_j represent the i-th and j-th cluster in the ground truth clustering and the clustering that is being evaluated respectively. In this study k^* and k are always equal.

We group the results as shown in Figure 6.4. For each database (D) and each combination of distance measure (d) and parameter (p), we obtain an array of $\mathcal{F}$-values that includes 15 values relative to the K-medoids algorithm ($\{c_1, ..., c_{15}\}$) and 4 values obtained from the variations of the hierarchical clustering algorithm ($\{c_{16}, ..., c_{19}\}$).

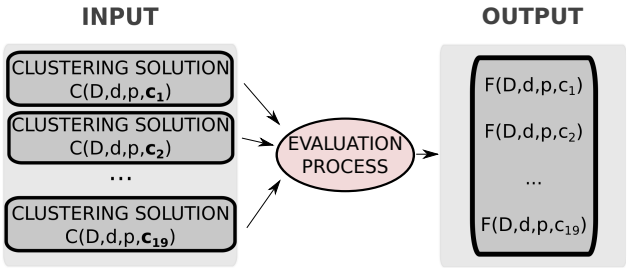


Fig. 6.4. Evaluation phase of the most suitable distance measure selection process.

6.3.3 Comparison and selection phase

In this last phase, the aim is to choose the most suitable distances for the time series database. The different parameter options have a deep impact on the performance of the distance measures. In this context, before comparing the results of different measures, the most appropriate parameter has to be identified for each case. For each distance measure, we analyze all its $\mathcal{F}$-arrays (one for each parameter) together, comparing their 19 values row by row. The most suitable parameter is that which has the best $\mathcal{F}$-value for the highest number of rows. If there is a tie, we choose one of the winning parameter options randomly.

Once we have selected the best parameter for each distance measure, we discard all the other options and can focus on comparing the 5 different distance measures. In order to do this, we compare the $\mathcal{F}$-value arrays associated to each distance measure using various consecutive statistical tests (see Figure 6.5).

First, we apply the Friedman rank sum test [66] to discover if there are any overall statistically significant differences between the performances of the distance measures. If the null hypothesis for this test is not rejected (p-value>

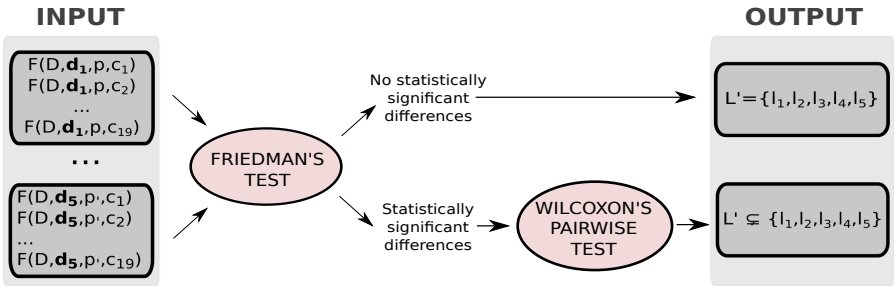


Fig. 6.5. Distance measure selection and label creation.

α), no significant differences can be assumed and we consider all the candidate distance measures equally suitable for the given time series database.

In contrast, if the null hypothesis is rejected (p-value$\leq \alpha$), we identify the first distance in the ranking built by the test, d, as one of the most suitable. In this case, the process continues with a comparison of every pair of distances using the Wilcoxon signed rank test [231]. In order to account for the multiple comparisons and control the family-wise error rate, we modify the p-values obtained from the Wilcoxon tests following the Holm post-hoc method. Finally, using these modified p-values, we determine the distance measures whose performance is statistically equal to d-s and add them to the set of most suitable distance measures. We set the significance level α to 0.05 in all the statistical tests applied in this phase.

6.4 Experimental setup

In this section, the experiments carried out are summarized. The objective is to evaluate the accuracy of the automatic distance measure selection tool, built as explained in Section 6.2.

6.4.1 The data

In this chapter we use three types of time series databases: datasets from the UCR archive, synthetic datasets, created specifically for this contribution, and datasets built from the PAMAP dataset.

Notice that z-normalization is a procedure which is typically applied before clustering or classifying time series data, with the aim of removing scale differences that could exist within the classes or clusters. However, in this case, we have not z-normalized any of the datasets given the small, or almost non-existent, scale difference between the series therein.

UCR datasets

Specifically, the 46 databases from the UCR time series database archive [103]. This archive collects the majority of the publicly available synthetic and real time series databases and, since its creation, has provided the baseline for evaluating new classification, clustering and indexing proposals.

We have calculated the characteristics introduced in Section 6.1 for all these 46 databases to be used as predictive information. Furthermore, one of the advantages of the UCR archive is that the ground truth clustering of the databases is provided together with the data. This enables the calculation of the class variables, as explained in Section 6.3.

Synthetic datasets

Recall that each time series database becomes an instance in our classification framework and the number of databases in the UCR archive is not very large. In view of this, we have complemented the set of databases with 555 synthetic datasets, generated specifically for this study. The details for the generation of the databases are included in Appendix A.

As can be seen there, the synthetic databases used in this study are divided into 8 distinct groups, depending on the baseline functions used to generate the series therein. The baseline functions range from simple piecewise constant functions or linear series to more complex ARMA formulations, combinations of sinusoidal functions, random functions, etc. From each type of database, several different instances have been generated by synthetically modifying and tuning the dimension of the database, the level of noise, outliers, shift and local warp. The idea is to introduce databases with different characteristics that will cover the feature space and will enable the construction of a general classifier, applicable to new databases.

Once we have generated the 555 synthetic databases, in order to use them as training instances of the classification framework presented in Section 6.2, the characteristics described in Section 6.1 have been calculated for each of them. Additionally, since the ground truth clustering of these artificial databases is directly obtained from the database generation process (see supplementary material mentioned previously for details), we obtain their corresponding class labels as explained in Section 6.3.

PAMAP database

The PAMAP database [174, 175] consists of the sensorial monitorization of 8 subjects while they perform several physical activities. The database is divided into 2 different parts, one corresponds to outside activities, such as running or rope jumping, and the other to inside activities, such as ironing or vacuuming. Each part consists of a set of multivariate streams of data (one stream for each subject), which record the measurements obtained during the experimentation by means of 3 Colibri wired IMUs (inertial measurement units) situated on the hand, chest and shoe of the subjects and a Heart-Rate monitor. In total, for each subject, there are two sets of streams of 45 variables, one for the outside activities and one for those activities performed inside. The specific activity that the subject is performing in each instant is also provided together with the sensor data, and this information will be used to define the ground truth clustering, necessary to evaluate the results obtained by our method.

In this chapter we focus on the outdoor activity data, which is available for 7 of the 8 subjects. The total number of activities that the subjects perform in this case is also 7. For the sake of simplicity, from the 45 variables collected in the data streams, we choose 3 variables (z-accelerometer on hand, chest and shoe) and analyze them separately: from each of the selected variables we will

obtain a different time series database. This choice has been made based on [168] but, of course, choosing other variables is also possible. In fact, selecting the most representative variable or variables, or applying the method to the multivariate stream itself, could be an interesting subject of study, but its outside the scope of this experimentation.

In order to be able to cluster these databases, we first have to pre-process the streams to obtain a set of finite-length time series, each one corresponding to one of the classes (defined by the different activities). As such, we separate the sections corresponding to each activity and each subject, and we segment them using a sliding window, as proposed in [175] and [168]. The width of the window is chosen to be 1000 data points (10 seconds) and we allow 50% overlap between the windows as shown in [175].

After applying this pre-process to the three selected variables separately, we obtain three databases with 2923 series of length 1000 each. We calculate the characteristics of these databases, as shown in Section 6.1. Additionally, we also obtain the ground truth labels for each dataset by using the information about the ground truth clustering, and applying the process shown in Section 6.3.

6.4.2 The distance measures

To validate our proposal, from all the time series similarity measures available in the literature, we have chosen the five explained in detail in Section 3.2: Euclidean distance (ED), Dynamic Time Warping (DTW), Edit Distance for Real Sequences (EDR), TQuest and the Fourier coefficient based distance. We have made this selection with the idea of including measures with different characteristics and considering their performance in previous publications. However, the distance selection method that is proposed in this chapter could be applied to other similarity measures, including metric learning proposals.

All the distance measures introduced above, except ED, require the selection of a parameter. In these experiments, we have considered the options summarized in Table 6.2 for each distance. We have defined the range of the parameters based on the experimentation shown in [228], but have augmented the step size in all cases because of the extra computational effort when working with the large number of databases included in this study.

6.4.3 Evaluation of the multi-label classifier.

In order to validate the performance of our proposal, we perform a complete set of experiments that we will explain in the following paragraphs.

To begin with, we have selected five different evaluation scenarios, summarized in Table 6.3, and have studied the behavior of our method in each of these contexts.

In the first scenario we only use the synthetic databases to evaluate the multi-label classifiers, within a 10x5-fold cross validation framework. In the

Table 6.2. Parameter options for the distance measures. l is the length of the time series in the database, and avg and sd are the average and the standard deviation of the series in the dataset.

Distance Measure	**Parameter**	**Min. value**	**Max. value**	**Step size**
DTW	Window size (w)	$0.04 \cdot l$	$0.32 \cdot l$	$0.04 \cdot l$
EDR	Threshold (ϵ)	$0.20 \cdot sd$	sd	$0.20 \cdot sd$
TQuest	Threshold (τ)	$avg - sd$	$avg + sd$	$0.40 \cdot sd$
F	Number of coefficients (f)	$0.04 \cdot l$	$0.32 \cdot l$	$0.04 \cdot l$

second scenario we use both the synthetic databases and the databases from the UCR archive together, also within a 10x5-fold cross validation scheme. The third scenario concentrates on the databases of the UCR archive and discards the synthetic databases. Finally, the fourth and fifth scenario apply a train/test validation methodology where one group of databases (synthetic or UCR) is used for training and the other for testing.

Table 6.3. Summary of validation scenarios.

Evaluation Scenario	**Training time series databases**	**Testing time series databases**	**Validation Technique**
Experiment 1	Synthetic	Synthetic	10x5-fold CV
Experiment 2	Synthetic + UCR	Synthetic + UCR	10x5-fold CV
Experiment 3	UCR	UCR	10x5-fold CV
Experiment 4	Synthetic	UCR	Train/Test
Experiment 5	UCR	Synthetic	Train/Test

To provide a baseline for comparison, we have included three additional naive multi-label classifiers in this first experimentation round. The first is a *Random Classifier (RC)* which divides the problem into 5 independent binary classification problems, each one associated to a distance measure. Each of these binary classifiers will output a value of 0 (distance not present in the label set) or 1 (distance present in the label set) randomly but respecting the univariate probabilities of presence/absence observed in the training set for each distance. The performance values of this classifier can be exactly calculated if the class distribution of the testing set is assumed to be equal to that of the training set. However, this condition does not necessarily hold in all cases. Based on this, and given the randomness of RC, we have simulated this classifier 1000 times and then averaged the performance values of all rounds

in order to provide reliable results. The second one is denominated *zeroR*, following the notation in MEKA, and uses a constant classifier that simply selects the most common label as the base classifier of the RkL algorithm. The third classifier, called *Majority label set (MLS)*, simply assigns the most common label set in the training set to all the test instances.

We have evaluated the performance of all these classifiers using three metrics. The first is the Exact Match (EM) metric, which measures the proportion of correctly classified instances:

$$\text{Exact Match} = \frac{1}{N_{Te}} \sum_{i=1}^{N_{Te}} I(\hat{L}_i = L_i) \tag{6.18}$$

where N_{Te} is the total number of instances in the testing set, L_i is the true set of class labels for instance i and $\hat{L}_i$ is the predicted set of labels for this same instance. I takes a value of 1 if the condition is true and 0 otherwise. It must be noted that EM is the strictest among the evaluation metrics in the multi-label framework because it only considers the instances whose predicted label set is identical to the ground truth label set [250].

The second metric is the accuracy, which is not as strict as the EM metric and is calculated as [250]:

$$\text{Accuracy} = \frac{1}{N_{Te}} \sum_{i=1}^{N_{Te}} \frac{|L_i \bigcap \hat{L}_i|}{|L_i \bigcup \hat{L}_i|} \tag{6.19}$$

The third performance measure we have chosen is the macro F1-measure calculated as [172]:

$$F1_{macro} = \sum_{i=1}^{L} F1(i) \tag{6.20}$$

where L is the maximum number of possible labels and F1(i) is the F1-score calculated for label i as:

$$F1 = \frac{2 \cdot TP}{2 \cdot TP + FN + FP} \tag{6.21}$$

where TP is the number of true positives and FP and FN are the number of false positives and false negatives, respectively.

As can be seen in Equations 6.18 and 6.19, the exact match and the accuracy are example-based metrics, where the performance of the classifier is calculated individually for all the instances in the database and then averaged to obtain a global value [250]. On the contrary, the F1-macro measure is a label-based metric, where the performance is evaluated for each label separately and then averaged across all the class labels. By using these two types of measures we will analyze the classifiers from two different points of view.

As additional metrics, and in order to study the behavior of some classifiers more specifically, we calculate the multi-label precision and recall [250]:

$$\text{Precision} = \frac{1}{N_{Te}} \sum_{i=1}^{N_{Te}} \frac{|L_i \bigcap \hat{L}_i|}{|\hat{L}_i|} \tag{6.22}$$

$$\text{Recall} = \frac{1}{N_{Te}} \sum_{i=1}^{N_{Te}} \frac{|L_i \bigcap \hat{L}_i|}{|L_i|} \tag{6.23}$$

The comparison of these two metrics will provide some insights into the relation between the cardinality of the predicted class labels and the true class labels.

Note that we have not used the databases obtained from PAMAP in this first set of experiments. The reason is that the objective is to evaluate the overall performance of the method, and the small number of databases obtained from the PAMAP data (only 3), do not allow to yield conclusive results. However, some additional experiments have been performed using the PAMAP databases, with the objective of applying our method to databases different from the typical benchmark datasets. These experiments do not require extensive explanations and will be directly introduced in Section 6.5.

In all the proposed experiments, we have implemented the RkL and ECC classifiers proposed in Section 6.2 with MEKA [171], a multi-label extension to WEKA. In the case of RkL, we employ the parameter combination recommended in [172]: $k = 3$ and $m = 2L$, where L is the number of possible labels. Finally, for the ECC classifier, since the training sets are not very large, we set the only parameter, the number of iterations, to 50 as proposed in [172]. In both these classifiers, we use a unique threshold for all labels by leaving the default option $PCut1$ in MEKA.

With respect to the parameters of the pruned C4.5 trees, in Experiments 3 and 5, we directly adopt the default parameters provided by the J48 function in WEKA, which implements this type of model. In these cases, the pruning strategy applied by WEKA is subtree raising with the confidence level set on a fixed value of 0.25. In order to obtain more generalizable results, *reduced error pruning* can be applied, which holds out part of the training set to estimate this confidence factor. However, this implies that the initial tree structure is learned with only part of the training set. In Experiments 3 and 5, the training set is already quite small so this option is not appropriate. However, in all the rest of the experimentation, we have enabled this option. All the other parameters have been left in default mode.

6.5 Results

In this section, we summarize the results obtained from the experimentation.

Table 6.4. Exploratory analysis of label distribution in the synthetic and UCR databases.

	UCR datasets	**Synthetic datasets**
Number of labels per instance		
Mean	1.370	1.986
Standard Deviation	0.771	1.228
Presence of each class label (%)		
DTW	50.00	68.30
EDR	45.70	69.20
TQUEST	19.60	7.40
ED	10.90	22.00
F	10.90	3.17
Concurrent presence of two class labels (%)		
ED & F	8.70	21.40
DTW & TQUEST	6.50	6.50
DTW & F	6.50	26.30
ED & DTW	4.30	19.80
ED & EDR	2.20	18.20
ED & TQUEST	2.20	3.60
EDR & TQUEST	2.20	6.70
EDR & F	2.20	24.90
TQUEST & F	2.20	4.30
DTW & EDR	1.30	41.40

6.5.1 Exploratory analysis of the data

Before dealing with the results obtained from the experimentation, it is important to have some information about the characteristics of the databases that have been used. The goal is to analyze the features and the labeling and to find the similarities and differences that could exist between the two types of databases.

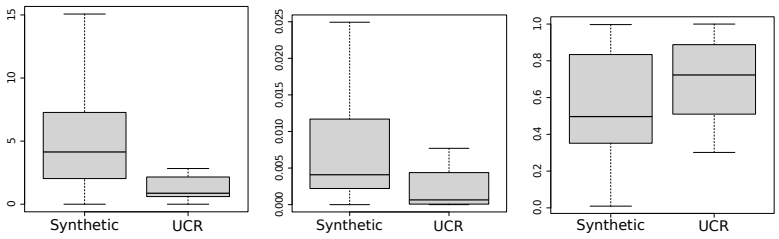
To begin with, in Table 6.4, some statistics and percentages are shown that provide information about the label distribution in the synthetic databases and the databases from the UCR archive.

It is obvious that the distribution of labels differs from one set of databases to the other. While the synthetic databases tend to have more than one label, the databases from the UCR archive generally admit only one distance as the most suitable. As evidence of this, the mean number of labels per instance is much closer to one in the case of the UCR distances than with the synthetic databases. Furthermore, the standard deviation is much lower in the UCR databases, and the concurrent appearances of pairs of distances are much less frequent in this case. Taking into account that the distance selection process is based on statistical tests, this suggests that many of the databases from the

UCR archive have very specific characteristics that only fit with the properties of one distance measure. For example:

- Many of the databases from the UCR archive have very similar clusters, some are even very difficult to discern by visual criteria. By setting very small values to the ϵ parameter of the EDR distance, this measure is able to magnify the small differences between the series, becoming an ideal distance measure for this type of databases.
- TQuest is among the most suitable distance measures much more frequently with the UCR databases than with the synthetic databases. As commented initially, this distance measure is not usually suitable for general databases and only provides good results in some specific cases where a certain threshold has a special relevance.

In relation to the feature space, after an exploratory analysis, it can be said that the synthetic databases cover a larger part than the databases from the UCR. As can be seen in Figures 6.6a and 6.6b, the noise and outlier levels are restricted to very small values in the databases of the UCR. Also, as commented previously, there is a large presence of databases with very similar clusters in the UCR archive. This results in a much higher mean correlation value in the databases of the UCR in comparison with the synthetic databases (see, Figure 6.6c).



(a) Noise level. (b) Outlier level. (c) Correlation level.

Fig. 6.6. Comparison of the mean noise level, mean outlier level and mean correlation level in the synthetic and UCR databases. Noise level obtained by the moving average method and outlier level calculated using the adjusted box-plot method.

Finally, variables related to trend, such as number of peaks, number of jumps, kurtosis and skewness, are generally higher in the synthetic databases. However, these variables are strongly influenced by the presence of randomness, noise and outliers and it is complicated to extract additional conclusions from exploratory univariate plots.

6.5.2 Evaluation of the classifiers

We will now focus on the results obtained by the classifiers proposed in Section 6.2: in Table 6.5, the exact match, accuracy and F1-macro values are provided for the 5 proposed evaluation scenarios.

The first clear conclusion is that, in the first two validation scenarios, the classifiers that use the characteristics introduced in Section 6.1 clearly outperform the other three naive classifiers for all the three performance measures. Furthermore, the high performance values obtained confirm the usefulness of the two classifiers as automatic distance selection tools.

In the third validation experiment, the RkL and the ECC classifiers also obtain superior results for all performance measures except the exact match, which is slightly superior for the MLS classifier. The reason for this is that, the labels of the databases in the UCR archive are very unbalanced. The most common labelset in the UCR archive only contains one label, the EDR distance, and this choice appears in almost a third of the databases. As such, the MLS classifier obtains good results in exact match, but very low values for

Table 6.5. Exact Match, Accuracy and F1-macro values for the multi-label classifiers.

	EXACT MATCH				
	ECC	**RkL**	**RC**	**MLS**	**zeroR**
Experiment 1	0.45	0.46	0.10	0.25	0.17
Experiment 2	0.42	0.44	0.10	0.24	0.16
Experiment 3	0.28	0.28	0.13	0.29	0.11
Experiment 4	0.07	0.13	0.09	0.30	0.09
Experiment 5	0.23	0.23	0.10	0.23	0.17
	ACCURACY				
	ECC	**RkL**	**RC**	**MLS**	**zeroR**
Experiment 1	0.70	0.71	0.40	0.41	0.56
Experiment 2	0.68	0.69	0.39	0.41	0.55
Experiment 3	0.45	0.44	0.28	0.36	0.41
Experiment 4	0.39	0.40	0.32	0.37	0.44
Experiment 5	0.48	0.48	0.30	0.39	0.56
	F1-macro				
	ECC	**RkL**	**RC**	**MLS**	**zeroR**
Experiment 1	0.71	0.64	0.40	0.16	0.33
Experiment 2	0.69	0.62	0.39	0.16	0.32
Experiment 3	0.30	0.26	0.27	0.12	0.23
Experiment 4	0.38	0.30	0.31	0.13	0.26
Experiment 5	0.33	0.30	0.31	0.16	0.33

the F1-macro measure, because it only guesses the label of the EDR distance. This same behavior also appears in scenario 4, where the MLS classifier obtains very high exact match values but low values in other performance measures.

In the fourth experiment, the classifiers obtained by the ECC and RkL algorithms obtain low exact match values. A first possible explanation for this phenomenon is that there are large differences between the feature space of the databases from the UCR and the synthetic databases (see Section 6.5.1). In view of this, the results obtained are somewhat as expected.

Moreover, some additional conclusions can be obtained if the precision and recall values are studied (see Table 6.6). In the fourth validation scenario, the classifiers obtained from the RkL and ECC algorithms obtain much higher values of recall than precision, which suggests that these classifiers tend to assign more labels than necessary. This behavior is expected because, as observed in Section 6.5.1, the databases from the UCR generally admit fewer class labels than the synthetic databases. In this context, although the accuracy and precision values for the ECC and RkL classifiers do not differ too much from those obtained from the MLS classifier, the exact match values become worse on account of the excess of labels. Nevertheless, given the high results that the ECC and RkL classifiers obtain for the F1-macro and accuracy measures in this experiment, we can say that, although they are more conservative, these classifiers are still useful to reduce the size of the candidate distance set.

Finally, in the fifth scenario, the ECC and RkL methods obtain exact match values which are similar (and slightly superior in the case of ECC) to the MLS classifier, and very superior to those obtained by the Random and

Table 6.6. Precision and Recall values for the multi-label classifiers.

	PRECISION				
	ECC	**RkL**	**RC**	**MLS**	**zeroR**
Experiment 1	0.79	0.80	0.54	0.69	0.68
Experiment 2	0.77	0.78	0.53	0.67	0.67
Experiment 3	0.53	0.51	0.33	0.46	0.46
Experiment 4	0.46	0.47	0.36	0.46	0.48
Experiment 5	0.67	0.70	0.45	0.68	0.69
	RECALL				
	ECC	**RkL**	**RC**	**MLS**	**zeroR**
Experiment 1	0.85	0.84	0.60	0.41	0.81
Experiment 2	0.84	0.82	0.59	0.41	0.81
Experiment 3	0.58	0.55	0.41	0.36	0.66
Experiment 4	0.74	0.67	0.56	0.37	0.76
Experiment 5	0.60	0.58	0.41	0.39	0.81

ZeroR classifiers. With respect to the other metrics, only the zeroR classifier obtains slightly higher results than the ECC and RkL classifiers.

The reason for this is that, in all the validation scenarios, the zeroR classifier always tends to assign the two most common distances, DTW and EDR, to all the test instances. Given the unbalanced nature of the label sets (see Table 6.5.1), especially those of the UCR databases, the F1-macro score and the accuracy are quite high for this classifier in all cases. However, the exact match value is very low in all cases. If we study Table 6.6, we can see that the recall values are generally very high for the zeroR classifier. In contrast, the precision values are very similar to or even lower than those obtained by other classifiers such as RkL or ECC. The big difference between these two performance scores implies that this classifier always tends to assign more labels than necessary. Moreover, since the two distance measures that the zeroR classifier systematically selects are the most expensive in terms of computational cost, we can say that this classifier is not very useful as a distance selection tool.

In summary, from the results shown, it may be concluded that the characteristics proposed to describe the time series databases are useful to discriminate between distance measures. This is deduced from the fact that the proposed classifiers obtain overall good performances for all evaluation scores in comparison to the naive classifiers, which obtain better results only on some occasions and for some specific evaluation scores. Furthermore, in view of the results obtained we can say that the proposed framework is useful to automatically select the most suitable distance measure for new time series databases.

6.5.3 Visual assessment of the results

As a second attempt to validate our method, we focus on the databases from the UCR and visually represent some of the obtained results. To do this, first, we obtain a label prediction for each dataset in the UCR archive using a ECC multi-label classifier trained using all the synthetic sets and all the UCR datasets except the one of interest. Then, we plot the obtained results in different manners, that will be explained directly in Section 6.5.

The Texas sharpshooter plot [8] was designed to evaluate distance measures in the context of time series classification. The aim of this plot is to assess when a new distance measure is better than a given baseline (ED, generally), and, additionally, if the databases in which it performs better can be identified in advance. For this, the expected gain in accuracy obtained by the proposed distance is calculated by using only a training set and then compared to the true gain, obtained by using a testing set.

In our case, we do not propose a unique distance measure, but a distance measure selecting tool. Furthermore, we focus on clustering, where there are no training and testing sets. In this context, the Texas sharpshooter plot is

not directly applicable. However, we retain the underlying idea, take the ED as the baseline distance and calculate these two values for each database:

$$F_{obtained} = \frac{\begin{array}{c}\text{Best F-measure obtained by the distance} \\ \text{measures selected by the multi-label classifier}\end{array}}{\begin{array}{c}\text{F-measure obtained by using} \\ \text{the ED}\end{array}} \tag{6.24}$$

$$F_{true} = \frac{\begin{array}{c}\text{Best F-measure obtained by the true} \\ \text{most suitable distance measures}\end{array}}{\begin{array}{c}\text{F-measure obtained by using} \\ \text{the ED}\end{array}} \tag{6.25}$$

Note that the true most suitable distance measure set is obtained as explained in Section 6.3. Once we have obtained these values, we plot them and obtain the following conclusions:

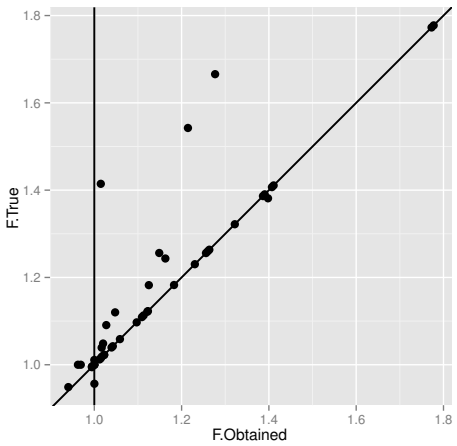


Fig. 6.7. Variation of the Texas sharpshooter plot for the databases in the UCR archive.

- **Points close to the diagonal:** The points that lie close to the diagonal represent databases for which our method obtains a F-measure similar to the F-value obtained by the most suitable distance measure(s). Specifically, for the databases that are situated exactly on the diagonal, our method has provided at least one of the most suitable distances among its choices. Note that there are a few databases that obtain a higher F.Obtained value than the F.True value (points below the diagonal). The reason for this is that the most suitable distance sets are selected by means of statistical tests (see Section 6.3) whereas in Figure 6.7 we focus only on the individual highest

F-values. There are only three cases that deviate notably from the diagonal: the *FaceFour*, *TwoPatterns* and *fish* databases. In the first case, our classifier chooses the Fourier distance instead of the EDR distance, which is the most suitable for this dataset. The database is quite noisy and both these distance measures are able to deal with noise, to some extent. Furthermore, the shift in the *FaceFour* dataset is very small, which is possibly why the multi-label classifier chooses the F distance over the EDR distance. In the case of *Two patterns*, the classifier chooses the EDR distance when the DTW distance is the most suitable. In this database, some sections of the series are noisy (generated by a Gaussian distribution), which is probably why the EDR distance was chosen. However, these noisy sections are alternated with piecewise constant functions, which are essentially the parts which provide relevant information about the different classes. This behavior can not be detected by the common denoising methods and so, it is difficult for the multi-label classifier to predict the correct class label. Finally, in the *fish* database, the DTW distance is chosen by the classifier, when the EDR is the most suitable. In this case, it is difficult to extract any intuitive reasons for the choice made by the classifier as it is probably due to a complex combination of features.

- **Points to the right of the vertical line:** The datasets that lie to the right of the vertical line are those in which our method has provided useful information, resulting in an improvement over the ED distance. It is obvious that most of the UCR databases shown in the plot lie in this area.
- **Points to the left of the vertical line:** These are the cases in which the ED performs better than any of the distance measures selected by our multi-label classifier. For the 5 databases that lie in this area, the accuracy obtained by the selected distance is not much lower than that obtained by ED. However, the ED is computationally very cheap, so using a more complex measure to cluster our database when ED provides good results, is a waste of computational effort. For two of the databases in this area, our classifier chooses the Fourier distance, which is not computationally very expensive. However, in the other three databases, DTW or EDR are chosen. In these cases, clustering the databases with the distances chosen by our classifier will yield similar accuracy results, but will result in a significant increase in computational cost compared to applying ED.

To finish with our visual analysis, we show two examples (Figures 6.8 and 6.9) of the improvement that can be obtained by using the distances proposed by our classifier instead of the baseline ED. In the *CBF* database, the shifting and warping between the series in the same class is quite large, and in this context, DTW gives much better results than ED. In the *DiatomSizeReduction* database, the similarity between the clusters makes EDR the most suitable distance measure because, by choosing a very small epsilon, it is able to magnify the tiny differences between the clusters, that ED is not able to capture correctly.

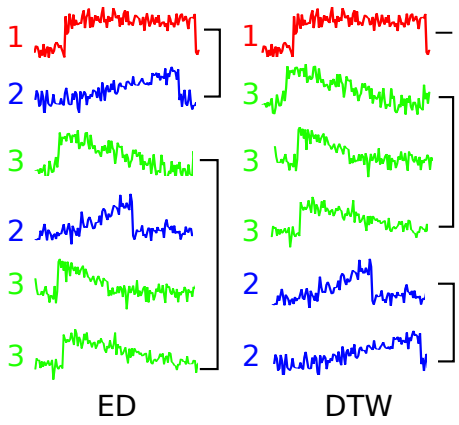


Fig. 6.8. The cluster solution obtained for a subset of the *CBF* dataset by using ED and the distance selected by our multi-label classifier. The true cluster of each series is represented by the number on its left and the clustering obtained by the distance measure is represented by the lines on the right.

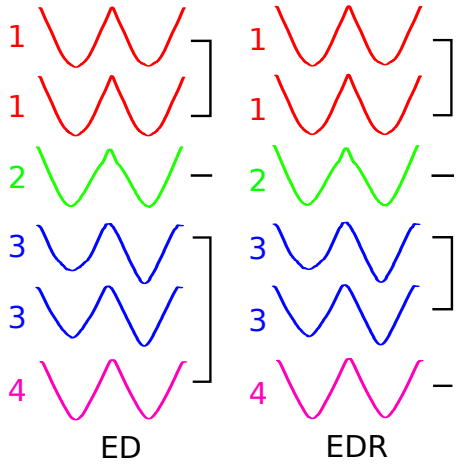


Fig. 6.9. The cluster solution obtained for a subset of the *DiatomSizeReduction* dataset by using ED and the distance selected by our multi-label classifier.

6.5.4 Performance of the method in the PAMAP database case study

Finally, in order to show the applicability of our method in a real context, we perform some final experiments using the PAMAP dataset [174, 175]. Note that the objective of this experiment is simply to show a supplementary example of the usage and performance of the proposed method with a set of unprocessed data. After calculating the characteristics and class values for the three databases created from the PAMAP data, we introduce the ob-

tained predictive variables into an ECC multi-label classifier, trained by using all the synthetic databases and the UCR databases and we analyze the differences between the predicted class labels and the true most suitable distance set directly.

The label predictions obtained for the three databases (z-accelerometer on hand, chest and shoe) built from the PAMAP data are shown in Table 6.7. Together with this, the true most suitable distance measures for each database are also shown. Additionally, in Figure 6.10, we plot the results by using the variation of the Texas sharpshooter plot, introduced in Section 6.5.3.

Table 6.7. Predicted and true most suitable distance sets for the three databases built from the PAMAP data.

	Predicted most suitable distance measures	**True most suitable distance measures**
z-acc. in hand	DTW, EDR	DTW
z-acc. in chest	DTW, EDR	DTW
z-acc. in shoe	DTW, EDR, F	DTW, F

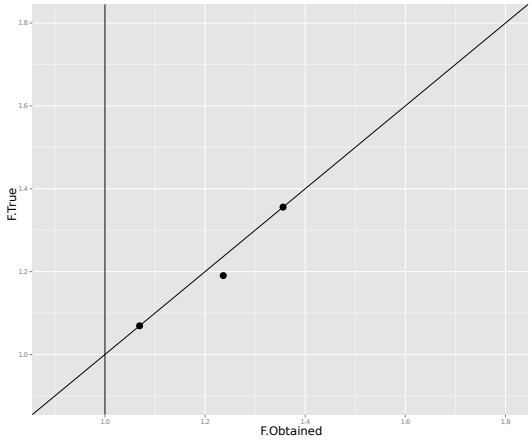


Fig. 6.10. Variation of the Texas sharpshooter plot for the databases obtained from the PAMAP database.

As can be seen in Table 6.7, our method always includes all the most suitable distances among its choices. The improvement over ED is clearly shown in the variation of the Texas sharpshooter plot shown in Figure 6.10, because all the points are situated to the right of the vertical line. Furthermore, the exact

match between the predicted distances and the true most suitable distances is almost obtained, and only EDR is chosen erroneously (false positive). It is probable that this distance is chosen due to the large amount of shift and noise present in the PAMAP data. Additionally, note that although EDR does not obtain the best values overall (recall that true labels are obtained using statistical tests), for the database obtained from the z-accelerometer situated on the hand, this distance obtains the best individual F-measure, which is why one of the points in Figure 6.7 appears below the diagonal line.

6.6 Conclusions and future work

In this chapter, a multi-label classifier for the automatic similarity measure selection has been proposed for the task of clustering time series databases. The classifier receives a set of characteristics that describe the database as input and returns the set of most suitable distance measures from a set of candidates. The positive results obtained in the experimentation for various multi-label classification performance measures demonstrate that this tool is useful to simplify the distance measure selection process, crucial to the time series database clustering task.

An important by-product of this chapter is the introduction of the labeling process introduced in Section 6.3. With the definition of this process, we have proposed a distance measure evaluation method based on statistical tests for the task of clustering. We believe that, a method of this type has not been proposed before.

The first obvious future research direction is to include new distance measures in the proposed framework. In this line, a more extensive study could be performed introducing new features, that would describe other aspects of the time series databases that have not been considered in this chapter. For this purpose, some of the features presented in [68] could be considered.

Another proposal for future work includes an optimization of the temporal costs associated with the calculation of the characteristics. Some of the features introduced in this study, such as the shift, are computationally quite expensive to calculate, which could be an inconvenience when working with particularly large databases. Since only means, medians, standard deviations and other general statistics are calculated, strategies such as sampling the time series database could be applied to reduce this computational cost. In the same line, reducing the number of parameters associated to the characteristics could also improve the applicability of the proposal.

Finally, some insights into the definition of the parameters of the distance measures have been included throughout the chapter, but no extended experimentation has been carried out on this topic. Studying the relationship between the characteristics of the databases and the parameters that define each distance could be useful to simplify the selection of a distance measure even more.

7

A contribution to early classification of time series

We have seen in Chapter 5 that, in travel time prediction, using a combined model based on time series clustering shows an improvement in performance in comparison to the general model trained with all the data, especially when using naive historical travel time prediction models. However, we have also seen that, in order to predict the travel time values for new incoming data points, the days they belong to must be assigned to one of the existing clusters or "day types". Since the objective is to provide online travel time predictions, these assignments must be made using only the data available at the time the prediction is made.

In Chapter 5 we have used a naive 1NN based strategy to make this class assignment, but this method entails several inconveniences. The main problem is that the class allocations are made every time a new data point arrives, and are based simply on the proximity to the cluster centroids. As such, these models do not consider the typology of the classes and can not measure the quality or accuracy of the predicted outcomes. Note that, if the class assignments are erroneous, the quality of the travel time predictions might deteriorate, because we will be using a model specifically trained for another, and possibly very distinct, traffic pattern.

Bearing all this in mind, we can say that we are interested in using specific models for each pattern whenever possible, but when the risk of making a mistake in the class assignment is high, we prefer to be safe and to simply use a general model trained with all the data.

It is easy to imagine that the earliness of the class predictions may affect their accuracy. As a general rule, the more data points that are available, the more accurate the class predictions become because the information about the time series is more complete. In view of this, the aim is to find a trade-off between two conflicting objectives: the accuracy of the predictions and their earliness. Class predictions should be given as early as possible while maintaining a satisfactory level of accuracy [237]. In the literature of time series data mining this task is denominated **early classification of time series**.

This problem arises in contexts where the data is collected over time and it is preferable, or even necessary, to predict the class labels of the time series as early as possible. For example, in medical informatics, the patient's clinical data is monitored and collected over time, and the early diagnosis of some diseases is highly correlated with a positive prognosis. For example, for *tamponade*, a critical medical condition that is best detected by examining photoplethismography (PPG) time series, researchers have long noted "the critical role of early diagnosis" [21], with just tens of seconds separating fatal from benign outcomes [194].

In this chapter, we present an **E**arly **C**lassification framework for time series based on class **DI**scriminativeness and **RE**liability of predictions (EC DIRE). As its name implies, in its training phase, the method analyzes the discriminativeness of the classes over time and selects a set of time instants at which the accuracy for each class begins to exceed a pre-defined threshold. This threshold is defined as a percentage of the accuracy that would be obtained if the full length series were available and its magnitude is selected by the user. The information obtained from this process will help us decide in which instants class predictions for new time series will be carried out, and it will be useful to avoid premature predictions. Furthermore, ECDIRE is based on probabilistic classifiers (PC) so, the reliability of the predicted class labels is controlled by using the class probabilities extracted from the classification models built in the training process.

The rest of the chapter is organized as follows. In Section 7.1, we introduce the problem of early classification in more detail and summarize the related work. Section 7.2, presents the main contribution of the chapter: the ECDIRE framework. In Section 7.3, we briefly introduce Gaussian Process classification models, the specific probabilistic models used in this chapter, and present an adaptation for the task of time series classification. In Section 7.4, the experimentation and validation of this method is carried out using a set of benchmark time series databases and, in Section 7.5, we present a case study on a real-world problem regarding the early identification of bird calls in a biodiversity survey scenario. Finally, in Section 7.6, we summarize the main conclusions and propose some future research directions.

7.1 Early classification of time series: related work

As shown in Chapter 3, time series classification [236] is a supervised learning problem where the objective is to predict the class membership of time series as accurately as possible. Due to its popularity, over the years, many extensions of the time series classification problem have been studied and early classification of time series [237] is one of the most notable and that which will be studied in this chapter. This problem consists of predicting the class labels of the time series as early as possible, preferably before the full sequence length is available.

This problem appears in many domains in which data arrives in a streaming manner and early class predictions are important. The most obvious applications are in medical informatics where the patient is monitored and tested, the clinical data is obtained over time and diseases must be identified as soon as possible. Other examples range from online fault detection in an industrial plant [17] to early detection of chemical spills or leaks by using data collected continuously from a set of sensors [82].

The earlier a prediction is made, fewer data points from the time series will be available and, in general, it will be more difficult to provide accurate predictions. However, as commented previously, in early classification of time series, the class predictions should be carried out as early as possible. In this context, the key to early classification of time series is not only to maximize accuracy, but rather to find a trade-off between two conflicting objectives: accuracy and earliness. The aim is to provide class values as early as possible while ensuring a suitable level of accuracy.

The solutions proposed in the literature are varied. Some methods simply learn a model for each early timestamp and design different mechanisms to decide which predictions can be trusted and which should be discarded. To classify a new time series, predictions and reliability verifications are carried out systematically at each timestamp and the class label is issued at the first timestamp in which a reliable prediction is made. For example, in Ghalwash et al. [72], a hybrid Hidden Markov/Support Vector Machine (HMM/SVM) model is proposed and a threshold is set on the probability of the class membership to try to ensure the reliability of the predicted labels. In Hatami and Chira [82], an ensemble of two classifiers is proposed which must agree on the class label in order to provide a prediction. Lastly, Parrish et al. [160] present a method based on local Quadratic Discriminant Analysis (QDA). In this case, the reliability is defined as the probability that the predicted class label using the truncated time series and the complete series will be the same. At each timestamp, the method checks if this probability is higher than a user pre-defined threshold and gives an answer only if this is so.

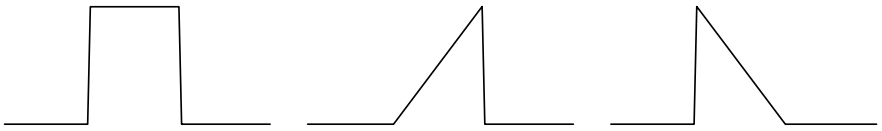
A slightly different strategy can be found in Ghalwash et al. [71], Xing et al. [238] and He et al. [83], where the authors propose methods to discover a set of shapelets which are useful to discriminate between the classes early in time. The concept of *shapelet* was first introduced by Ye and Keogh [241] and refers to a subsequence or subpattern of a time series that can be used to represent and identify a given class. To classify a new time series in an environment where the data is collected over time, each time a new data point arrives, the series is compared to the library of shapelets and is classified if a match is found [71]. The matching condition depends on the required reliability level and is determined in the training process.

Finally, we must mention the Early Classification on Time Series (ECTS) method presented by Xing et al. [237]. This work formally defines the problem of early classification for the first time, and analyzes the stability of the nearest neighbor relationships in the training set as the length of the series is

truncated. In its training phase, the ECTS method calculates the Minimum Prediction Length (MPL) for each time series in the training set. This value represents the earliest timestamp in which the given training series begins to provide reliable nearest neighbor class predictions. To predict the class label of a new unclassified time series, at each early timestamp t, all the training series are truncated to the corresponding length and a simple nearest neighbor classifier (1NN) is applied. However, once the nearest neighbor is found, if its MPL is larger than t, the prediction is considered unreliable, no class label is provided and the method simply waits until the next data point arrives.

In this chapter, we will design an early classification method that will focus on two aspects. Firstly, note that a clear disadvantage of the early classification methods in the literature is that, when trying to predict the class label of a new time series, forecasts must be made and checked at all timestamps. Only the ECTS method can avoid a few predictions at the initial early instants, when the MPLs of absolutely all the training series are larger than the given timestamp. This results in many unnecessary calculations that could be avoided if the typology of the different classes were taken into account. The synthetic CBF database [103] is a clear example of the disadvantage of issuing and checking the predictions at each timestamp. This database is identified by three shapes, each of which represents a class (see Figure 7.1). The series that belong to each class are obtained by translating, stretching and compressing an initial shape and adding noise at different levels. As can be seen, the three shapes are identical until one point and, thus, it makes no sense to make predictions until this time has arrived. In order to deal with this issue, our method incorporates a mechanism to initially decide, for each class, after which time instant it makes sense to make predictions and when we can avoid unnecessary calculations. Additionally, as will be shown in Section 7.5, this mechanism will also provide a direct way to apply ECDIRE to the problem of early pattern identification in streams of unknown and possibly infinite length.

Secondly, our method is based on probabilistic models and includes a procedure that controls the reliability of the class forecasts. We note that some proposals such as the ECTS method, which is based on the NN classifier, simply lack a strategy to control the reliability of the issued class labels. As such, for example, if an outlier series arrives, it will inevitably be assigned to one of the classes. More adequate solutions such as Ghalwash et al. [71; 72], Hatami



(a) Cylinder. (b) Bell. (c) Funnel.

Fig. 7.1. The idealized shapes that define the CBF synthetic database. The actual items are distorted by random noise and warping.

and Chira [82], Parrish et al. [160], Xing et al. [238] and He et al. [83] try to uncover those situations by means of different mechanisms. However, most of these methods are not able to provide a quantitative and interpretable measure of the uncertainty of their predictions. As commented in Ghalwash et al. [71], the ability of providing such measures is a desirable property of early classification methods because it helps the users to assess the quality of the responses they receive. Our early classification method includes a strategy to discard unreliable responses, and it directly provides a quantitative measure of the goodness of the issued predictions.

7.2 Early Classification framework for time series based on class DIscriminativeness and REliability of predictions (ECDIRE)

As previously commented, early classification has to deal with two conflicting objectives: earliness and accuracy. In this section, we introduce the ECDIRE method, which takes these two objectives into account.

7.2.1 Learning phase

The learning stage of the ECDIRE method is divided into three steps. In the first step, we focus on designing a mechanism that will identify the timestamps from whence it makes sense to make forecasts, for each class. In the second step we will design a reliability condition that will control the goodness of the predictions and will directly provide a measure for the uncertainty of the predictions. Finally, once this is done, we train a set of probabilistic classifiers (PC), which will be used to issue the class predictions for the new time series.

Step 1: Analysis of the discriminativeness of the classes

In the first step of our early classification framework, we propose analyzing the database and the classes therein with the objective of deciding from which instant on it makes sense to make predictions, and when we can avoid unnecessary calculations when predicting the class label of a new time series.

We propose building a timeline that describes in which moment each class becomes *safe*. This concept of safety refers to the moment in which we can begin to discriminate a certain class from all the rest of the classes with a certain degree of accuracy.

Figure 7.2 shows an example timeline with four relevant timestamps: t_1,t_2,t_3 and t_4. These instants are ordered in time and, in each of them, a class or set of classes becomes safe. For example, at instant t_3 we can begin to discriminate class C_5 from the rest of the classes. If we consider the classes that have appeared earlier in the timeline, we can conclude that at timestamp

t_3, classes C_2, C_4, C_1 and C_5 can be predicted accurately. As will be shown in the next steps, predictions will only be provided at the timestamps that appear in the timeline or later and, by means of this mechanism, a large number of calculations can be avoided when making predictions for new time series.

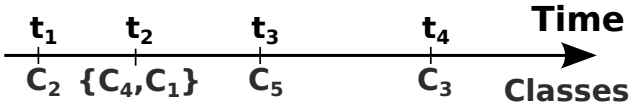


Fig. 7.2. An example timeline built to represent at which instant each class becomes safe.

The problem is now reduced to finding a method to discover the timestamps which constitute the timeline of each database. In order to do that, at each possible timestamp, a multi-class PC, which considers all the classes in the training set, is trained. As will be explained in Section 7.3.1, in this chapter Gaussian Process classifiers are used, but any other classifier that provides probabilistic outputs could be used. Then, we check at which instant the accuracy becomes good enough for each class. Training a classifier at a given early time instant simply implies learning the model using only the data points collected up to that instant. We recall that in early classification problems the time series are collected over time and, so at each early timestamp, only the corresponding first points of each series are available.

In this context, we will analyze the overall trends that can be observed in the accuracy of the classifiers as more data is available. At the beginning, the data is scarce and, with every new additional data point, the accuracy of the classifiers generally increases rapidly. However, as time goes by, the number of available data points becomes larger and the improvement in accuracy becomes slower and tends to stabilize (see Figure 7.3a). On some occasions, the accuracy might even drop slightly after a point due to noise that the additional data may introduce (see Figure 7.3b). In these two cases, which are the most common, it is possible to classify the time series early in time without downgrading the accuracy that would be obtained if the full series were available ($full\_acc.$). A third possible case occurs when the behavior of the accuracy is strictly monotonous (see Figure 7.3c). In this case it will not be possible to obtain the accuracy achieved with the full series earlier in time and, if early predictions are desired, the accuracy requirements will have to be downgraded.

In practice, although they are more uncommon, there are other possible trends that the accuracy may follow. Nevertheless, these can essentially be reduced to these three cases. For example, the case where the accuracy is constant can be considered equivalent to the first case, and the strictly decreasing case can be included within the second trend type, because the practical implications regarding early classification are the same.

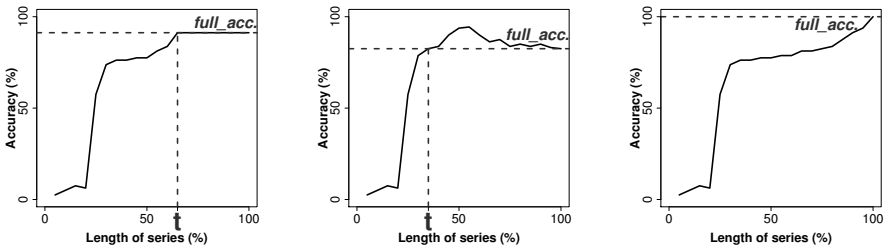


Fig. 7.3. Illustrative examples of the most common behaviors of the accuracy of a classifier as the number of available data points increases. The x axis refers to the percentage of the series available at the time of prediction and the y axis refers to the accuracy obtained by the classifier.

Also note that when we reduce the possible behaviors of the accuracy into these three cases, we are talking about the overall trend of the accuracy over time. In practice, the raw behavior of the accuracy tends to be noisy, with peaks and outliers that disturb these perfect trends. As such, when choosing the timestamps at which we can make early predictions, we require some sort of stability in the behavior of the accuracy. In the following paragraphs the procedure followed to build the timeline is explained in more detail.

As we have mentioned, in a first step, a multi-class PC is trained within a stratified 10x5-fold cross validation framework, following the recommendations of Rodríguez et al. [178]. This is done by using the full length time series and considering all the classes. Once this is carried out, the accuracy of the model is estimated separately and locally within each class. Subsequently, the classes that yield an accuracy lower than $1/|C|$ are identified, C being the set of different classes in the database. If a random classifier, which assumes that the classes are equally distributed, were applied to the training data, the accuracy that would be obtained within each class would tend to $1/|C|$. In this context, the predictions obtained for the identified classes are worse than those that would be obtained by the random classifier. If we consider this random classifier the baseline, these class values can not be predicted with enough accuracy, even when the whole series is available. Because of this, their corresponding labels will never be given as a prediction in our early classification framework.

Of course, if desired, this condition could be modified or removed by the user. For example, note that, if the classes are unbalanced, the baseline presented above is stricter with smaller classes than with larger ones. In this case, the class frequencies in the training set could be used to define a different baseline accuracy value for each class. Of course, this could result in very low baselines for some classes, which might result in more unreliable class pre-

dictions. In this sense, one could also consider $1/|C|$ a very low baseline and require higher accuracy values for all classes. Finally, these baseline accuracy values could also be set manually using some sort of domain knowledge or based on domain specific demands.

For the rest of the classes (c), a percentage ($perc\_acc_c$) of the accuracy obtained with the full length series ($full\_acc_c$) is chosen and, with this, the desired level of accuracy is defined for each class as $acc_c = perc\_acc_c * full\_acc_c$. The goal will be to preserve this accuracy while making early predictions. With this information, the timeline can be built by following these steps:

1. Select a wide set of early timestamps. In our case, they are defined as a sequence of percentages of the length of the series in the database: $E = \{e_1\%, e_2\%, e_3\%, \ldots, 100\%\}$
2. For each early timestamp $e_i \in E$:
   a) Truncate the time series in the training set to the length specified by e_i, starting from the first point.
   b) Train an early PC within a stratified 10x5-fold cross validation framework using these truncated sequences.
   c) For each class (c), save the mean of the accuracies obtained in all the repetitions and folds ($acc_{e_i,c}$). For a more conservative solution, choose the minimum or a low percentile of the accuracies obtained in all the repetitions and folds.
3. For each class c, choose the timestamp $e_c^* \in E$ as:

$$e_c^* = \min\{e_i \in E \mid \forall j \geq i \ , \ acc_{e_j,c} \geq acc_c\} \tag{7.1}$$

These e_c^* values are the timestamps that appear in the timeline and will be crucial to avoid useless predictions because, as will be shown in the next sections, forecasts will only be made at these instants or later. Furthermore, the timeline provides valuable information about each class and, thus, incorporating it into the ECDIRE method improves the interpretability of the model, which is a desirable property in early classification [71]. Finally, as we will explain in Section 7.5, the timeline obtained in this first step provides the means to directly apply the ECDIRE method to another scenario: the early detection of significant patterns in streams of unknown length (possibly infinite).

Step 2: Prediction reliability

In this step, we concentrate on the second crucial aspect of our method: designing a mechanism to control the reliability of the class predictions issued by the classifiers. As commented previously, in this work we use classifiers which provide probabilistic outputs. These values can be used to ensure the reliability of the class forecasts and to provide a quantitative measure of the quality of our predictions. The idea is to set a threshold to the differences

between the probabilities assigned to the different classes with the aim of requiring a sufficiently large difference between the predicted class and the rest of the classes.

A separate threshold ($\theta_{t,c}$) is defined for each class (c) and early timestamp (t) and, for this, we use the set of classifiers built within the 10x5-fold cross validation framework presented in the 2(b) item of the previous step. We recall that each classifier corresponds to one early timestamp and, since a different threshold is sought for each early time instant, we analyze each one separately.

First, the predicted class probabilities of all the correctly classified test instances are extracted from all the folds and repetitions and they are grouped depending on the class of the test instance. Within each group, the differences between the predicted class probability of the pertinent class and the rest of the classes will be computed and the minimum of these differences will be saved for each instance. These values represent the distance in terms of differences in class probabilities of the winning class and the next most probable class for all the test instances in the group. Finally, to define the threshold for the given early timestamp and class, the minimum of all these values is taken within each group. It must be noted that choosing the minimum in this last step leads to a very loose threshold that will only discard very uncertain predictions. For a more conservative threshold or when many outlier series are present, the mean or median could be chosen by the user in this last step (see example in Section 7.5).

Step 3: Training the set of classifiers

In this last step, we will train a set of PCs, which will later be used to classify new time series. To begin with, a classifier is built for each timestamp that appears in the timeline constructed in the previous section. Contrary to Step 1, where the PCs are learnt within a 10x5-fold cross validation scheme, in this case, the models are built by using all the training instances. Additionally, if the last timestamp in the timeline (e_k^*) does not coincide with 100% of the length of the series, the ensemble of classifiers will be complemented with classifiers built at all the posterior timestamps in E, namely in $T = \{e_i \in E : (e_i > e_k^*)\}$. This means that, after timestamp e_k^*, our method becomes similar to those in the literature but, until this point, many unnecessary models will be discarded. As for the models built in Step 1, in this case, all the classifiers will also consider all the classes in the database.

In Figure 7.4, an example of the ECDIRE model obtained for a database with four classes is represented. In this case, the last timestamp in the timeline is e_3^*. As can be seen, PCs are only available at the timestamps that appear in the timeline and later. Furthermore, in each case, only the classes which are safe are considered.

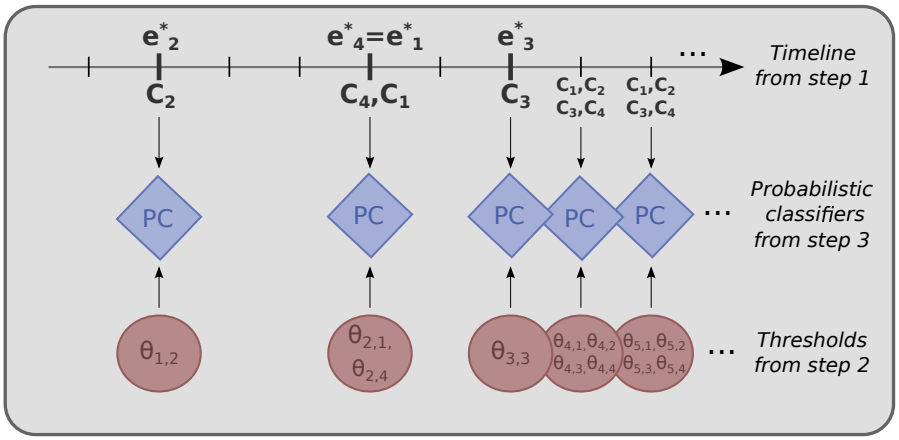


Fig. 7.4. An example of the ECDIRE model obtained from the three steps that conform the learning phase.

7.2.2 Prediction phase

In this phase, the objective is to make early class predictions for new unclassified time series. As commented previously, PC models are only available at the timestamps that appear in the timeline obtained in Step 1 and at all the timestamps after the last point in the timeline, so predictions will only be made at these time instants. Furthermore, although the classifiers are built including all the classes, only those that are safe will be considered at each instant. If the classifier assigns a non-safe label to a given test series, the answer will be ignored and the time series will continue in the process and wait until the next time step, when more data will be available.

Moreover, the series that are assigned to a safe class must pass the test of the prediction reliability. The differences between the predicted class probabilities obtained from the classification will be compared to the thresholds extracted in Step 2. Only if they are larger, will a final class be given. If not, the series will not be classified and will continue in the process and wait until enough new data points are available.

Finally, the class probabilities obtained for the classified test instances can be used as an estimate of the uncertainty of the prediction. As commented previously and as stated by Ghalwash et al. [71], the availability of measures of this type is a desirable property for early time series classification problems, and providing these values may be useful and informative for users.

7.3 Probabilistic classifiers in the ECDIRE framework

Based on the general design of our early classification framework, any classification algorithm with probabilistic outputs could be used to build the clas-

sifiers in Sections 7.2.1. In this case, we have chosen Gaussian Process (GP) models.

GP models are probabilistic models and, unlike other kernel based models such as SVMs, they output fully probabilistic predictions, which are directly applicable in our framework. Although they have been extensively applied in many domains, they are not commonly used in time series classification and, to the extent of our knowledge, have never been applied to the task of early classification. However, these models are well known for their ability to obtain good generalization results in the presence of few labeled instances [197], which is common when working with time series. Also regarding this, when working with GP models, the parameters of the kernel function can be learnt from the data within the Bayesian inference framework, whilst in SVMs this is generally done by using cross validation. Finally, in some cases, these models have shown superior performance for time series classification compared to other kernel based models such as SVMs [197].

7.3.1 Gaussian Process classification

A Gaussian Process (GP) [170] is an infinite collection of random variables that, when restricted to any finite set of dimensions, follows a joint multivariate Gaussian distribution. A GP is completely defined by a mean function $m(x)$ and a covariance function $k(x, x')$:

$$f(x) \sim GP(m(x), k(x, x')) \tag{7.2}$$

In machine learning, GPs are used as priors over the space of functions to deal with tasks such as regression and classification by applying Bayesian inference. The idea is to calculate the posterior probability of the classification or regression function, departing from a GP prior and applying Bayes' rule.

In this chapter, we are interested in the use of GPs to solve classification tasks. The idea is similar to that of the logistic or probit regression. First, a continuous latent function f with a GP prior is introduced, which is a linear combination of the input variables. This function is then composed with a *link* function (σ) such as the logistic or probit function which transfers the values to the $[0, 1]$ interval. The resultant function π is the function of interest, and a prediction for a given test instance can be obtained by calculating [170]:

$$\bar{\pi}_* = p(y_* = +1|X, \mathbf{y}, \mathbf{x}_*) = \int \sigma(f_*) p(f_*|X, \mathbf{y}, \mathbf{x}_*) df_* \ , \tag{7.3}$$

where

$$p(f_*|X, \mathbf{y}, \mathbf{x}_*) = \int p(f_*|X, \mathbf{x}_*, \mathbf{f}) p(\mathbf{f}|X, \mathbf{y}) d\mathbf{f}, \tag{7.4}$$

and f_* is the value of the latent variable for the test instance and $\{\mathbf{x}_*, y_*\}$ and $\{X, \mathbf{y}\}$ are the input and class values of the test instance and training instances respectively. Contrary to the case of regression, this problem

is not analytically tractable because the class variables are discrete and a non-Gaussian likelihood appears when calculating the posterior of the latent variable ($p(\mathbf{f}|X,\mathbf{y})$). In any case, many methods have been proposed to find a solution to this inference problem, and some examples and relevant references can be found in Rasmussen and Williams [170].

Specifically the proposal shown in Girolami and Rogers [74], which is valid for both binary and multinomial classification, has been applied in this case to train the GP models. Departing from a probit regression model, these authors apply a data augmentation strategy, combined with a variational and sparse Bayesian approximation of the full posterior. They first introduce a set of m_k variables, similar to the f introduced above but associated to each one of the possible class values k. Additionally, they augment the model with a set of auxiliary latent variables $y_k = m_k + \epsilon$, where $\epsilon \sim N(0,1)$. In probit regression, this strategy enables an exact Bayesian analysis by means of Gibbs sampling, but the authors additionally propose a variational and sparse approximation of the posterior which improves the efficiency of the method. The details of this numerical approximation can be studied in the cited reference [74].

7.3.2 Gaussian Process for time series classification

A straightforward way to apply GP classification to temporal data is to include the raw time series directly into the model as input vectors. In this case, the covariance function could be defined by any commonly used kernel function (see [170]). The main drawback of this approach is that these common kernels are not specifically designed for time series and, as such, they are not designed to deal with shifts, warps in the time axis, noise or outliers, features that are commonly present in temporal databases. Indeed, many of these kernels are based on the Euclidean distance (ED), and it has been proven in the past few years that this distance measure does generally not provide the best results when working with time series databases [228].

In view of this and as shown in Pree et al. [163] and Wang et al. [228], it seems interesting to be able to include different time series distance measures into the classification framework. A simple way of including different distance measures into a GP classification framework is to directly replace the ED that appears in other common kernel functions by the distance measure of interest [163]. However, this solution is problematic for many time series distance measures because the resultant kernel function does not necessarily create positive semi-definite covariance matrices, which are a requirement when working with GP models [170]. Thus, to deal with this issue, we propose the methodology shown in Figure 7.5.

In a preliminary step, a distance matrix is calculated by using any distance d of interest. Each position of this matrix holds the distance (d_{ij}) between two series TS_i and TS_j. This new feature matrix will be the input to the GP classification framework and, now, any typical kernel function may be used and the problem of non-positive semi-definite covariance matrices will not be

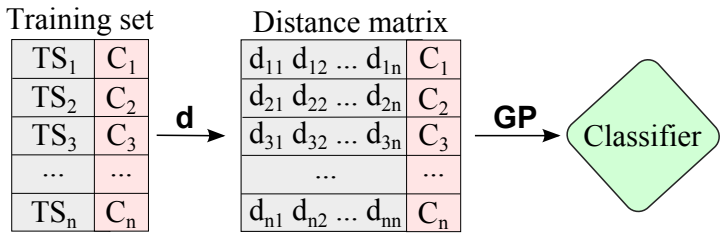


Fig. 7.5. Classification framework

encountered. Additionally, this procedure also allows us to combine different distance measures by concatenating the distance matrices obtained from each of them [102] or by combining them using some operation such as a weighted sum, used in Stathopoulos and Jones [197].

This idea of building classifiers by using distance matrices was first introduced by Graepel et al. [75] and can be used to easily introduce different distance measures into common classification models. Some applications of this methodology to time series databases can be found in Stathopoulos and Jones [197] and Kate [102], where the popular Dynamic Time Warping distance is included into GP models and SVMs, respectively.

7.4 Experiments

In this section, the performance of the ECDIRE method is evaluated in terms of accuracy and earliness. The complete code of the experimentation is readily available in the Downloads section of our web page [1].

7.4.1 The Data

In this chapter, all the time series databases from the UCR archive available at the time of the experimentation are used for validation (see Section 6.4.1 for more information on these datasets). Some datasets from this archive have also been used to validate most previous early time series classification proposals, but, to the best of our knowledge, no previous early classification method has been validated with so many databases.

7.4.2 Parameters of the early classification framework

The early classification framework presented in this chapter requires the definition of two parameters. First of all, the number of early timestamps that

[1] http://www.sc.ehu.es/ccwbayes/isg/index.php?option=com_remository&Itemid=13&func=startdown&id=23

ECDIRE considers have to be chosen. In this case, in order to avoid excessive computational costs, we have chosen E={5%,10%,15%,...,85%,90%,95%, 100%}. However, choosing a finer granularity could result in earlier predictions.

Additionally, recall that the desired level of accuracy (acc_c) must be tuned by the user. This parameter was defined as a percentage of the full accuracy ($full\_acc_c$) for each class. In this case, 100% of $full\_acc_c$ is chosen for all classes. Namely, $perc\_acc_c = 100\%, \quad \forall c \in C$

7.4.3 Parameters and implementation of the GP models

As commented in Section 7.3.1, the GP classifiers in the ECDIRE method have been trained following the method proposed in Girolami and Rogers [74] and, specifically, based on its implementation available in the *vbmp* package of R [112]. This package is designed for multinomial GP classification and only works with datasets that have more than 2 classes. In view of this, we have completed the code in order to also enable binary classification by following the formulation shown by Girolami and Rogers [74]. The parameters for the *vbmp* function have all been set to their default values except the convergence threshold, which has been lowered to a stricter value (10^{-8}), and the maximum iteration number (set to 24), which has been lowered in order to reduce computational costs. Furthermore, after a set of preliminary experiments, the inner product kernel has been chosen as the covariance function. In this case, the hyperparameter of the kernel function is a scaling factor which assigns a weight to each of the dimensions of the input data. We recall that each dimension of the input distance matrix corresponds to one of the series in the training set (see Figure 7.5), so the hyperparameter is initialized to a vector of ones, giving equal importance to all the dimensions. However, the hyperparameter estimation option (*bThetaEstimate*) is set to TRUE in order to remove the effect of any irrelevant or outlier series.

In some cases, depending on the dimension of the training data and the magnitude of its values, the hyperparameter estimation method implemented in this package results in some numerical errors. This is the case of the *50words* and the two *NonInvasiveFatalECG_Thorax* (NIFECG_Thorax) datasets. These errors are due to the difficulties that arise when dealing with very small values and, hence, scaling the training set with a big enough scaling factor will enable the estimation of the hyperparameter. However, a different and adequate scaling factor has to be sought for each database. Since searching for a method to automatically find this factor is outside the scope of this dissertation, we have chosen to use a fixed and uninformative hyperparameter in these three databases, a vector of ones, by setting *bThetaEstimate* to FALSE.

The distance measure used to built the input matrix is another parameter to be chosen when building the GP classifiers. As explained in Section 7.3.2, any time series distance could be used. Nevertheless, the evaluation of the

performance of the different time series similarity measures is not the objective at hand and, with this in mind, the basic ED has been used in the experiments presented.

7.4.4 Comparison to other early classification methods

In order to validate our proposal, we have compared its performance with the 1NN algorithm which uses the full length series and 3 state-of-the-art early classification methods with available source codes [2] [3].

The first early classification method is the ECTS method [237]. We compare our method to two variants of this proposal, the strict (ECTS1) and the loose (ECTS2) version, setting the only parameter (*minimumSupport*) to 0, as the authors have shown in their experiments. The second method used for comparison is the Early Distinctive Shapelet Classification (EDSC) method [238]. The authors propose two variants that show similar results in their experimentation but, in this experimentation, only the one based on the Chebyshev Inequality is considered because of code availability. The parameters are set to the same values shown in their experimentation. The third method is the reliable early classification method proposed by Parrish et al. [160]. The values for the reliability threshold τ are taken from the experimentation shown by the authors: 0.001, 0.1, 0.5 and 0.9 (RelClass1,2,3 and 4, respectively from now on). Furthermore, in this last method, the local discriminative Gaussian dimensionality reduction has been enabled because it reduces computational costs and can reduce noise and yield higher accuracy values [160].

7.4.5 Evaluation method

The databases from the UCR archive used in this study are provided with pre-specified training and testing sets. In order to enable reproduction of the results, it is common in the literature to directly use these training and testing sets in the experimentation and, in view of this, these sets have also been respected in this case. As such, the evaluation of the classification framework has been done following a train/test validation methodology.

Two evaluation measures are given, each one corresponding to one of the conflicting objectives of the early classification problem. The accuracy is calculated in terms of the percentage of the correctly classified test instances:

$$Accuracy = \frac{1}{N}\sum_{i=1}^{N} I(\hat{C}_i = C_i) \tag{7.5}$$

C_i being the true class value for test instance i and $\hat{C}_i$ its predicted value. $I(\cdot)$ takes a value of 1 if the condition is true, and 0 otherwise.

[2] **ECTS and EDCS:** http://zhengzhengxing.blogspot.com.es/p/research.html
[3] **RelClass**: http://www.mayagupta.org/publications/Early_Classification_For_Web.zip

The earliness is measured by calculating the mean of the time instances t_i in which the test examples are classified. In order to give a more interpretable result, this value is normalized by the full length of the series in the database (L) and given as a percentage:

$$Earliness = \frac{1}{N} \sum_{i=1}^{N} \frac{t_i}{L} \times 100 \tag{7.6}$$

All the early classification methods are evaluated by using these two measures. However, in the case of the full 1NN method, as it always uses the full length of the series, we only provide the metric regarding the accuracy.

Apart from giving the raw values of these two performance measures, in order to assess the statistical differences between the different methods in each objective, we perform a set of statistical tests for each objective. First, the Friedman test is applied to conclude if there are any overall statistical differences and to obtain a ranking of the methods. Then, the Holm posthoc procedure [51] is applied to make pairwise comparisons between the methods in each objective. The results will show if there are statistically significant differences between each pair of methods in each of the objectives.

Finally, for a more global evaluation and comparison between our proposal and the other state-of-the-art early classification methods, we take the multiobjective nature of the early classification problem into account. The Pareto optimality criterion states that a solution dominates another solution if it obtains better results in at least one of the objectives while not degrading any of the others. Based on this criterion, by counting the times in which our method dominates the others and vice versa, we provide a multiobjective comparison of the ECDIRE method with the rest of the early classification methods considered in this experimentation.

7.4.6 Results

In Tables 7.1 and 7.2 the accuracy and earliness results obtained from the experimentation can be observed respectively. It must be noted that the results for the *StarlightCurves*, *NIFECG_Thorax1* and *NIFECG_Thorax2* databases are ommited for the EDSC method, because after more than a month and a half of computation, the method was not capable of providing any results. The results issued from Friedman's test and Holm's method are summarized in Figures 7.6 and 7.7. These tests are performed by removing the *StarlightCurves*, *NIFECG_Thorax1* and *NIFECG_Thorax2* databases due to the lack of results for the EDSC method.

If we analyze the **accuracy** results, it can be seen that the ECDIRE method obtains the best results in 17 databases, only preceded by the RelClass4 method which obtains the best accuracy in 18 databases. The worst values are obtained by the ECTS and the ECDS methods which beat all the other methods in only 3 cases. Furthermore, it should be noted that, contrary

Table 7.1. Accuracy values for ECDIRE, strict and loose versions of the ECTS, EDSC, RelClass with 4 parameter settings and the full 1NN method. For the ECDIRE method the percentage of classified series are included between parentheses. The method/s with the highest accuracy in each database are shown in bold.

Dataset	ECDIRE	ECTS1	ECTS2	EDSC	RelClass1	RelClass2	RelClass3	RelClass4	1NN
50words	0.55 (100%)	0.58	0.57	0.42	0.65	0.65	**0.66**	**0.66**	0.61
Adiac	0.63 (100%)	0.61	0.40	0.13	0.62	0.63	0.63	**0.64**	0.61
Beef	0.47 (96.67%)	0.53	0.50	0.20	0.40	0.40	0.57	**0.67**	0.53
CBF	**0.89 (100%)**	0.85	0.85	0.88	0.35	0.46	0.80	0.87	0.85
ChlorineConcentration	0.58 (100%)	0.62	0.62	0.53	**0.82**	**0.82**	**0.82**	**0.82**	0.65
CinC_ECG_torso	0.71 (89.06%)	0.87	0.87	0.56	0.80	0.84	0.85	0.86	**0.90**
Coffee	**0.89 (96.43%)**	0.79	0.75	0.39	0.71	**0.89**	**0.89**	**0.89**	0.75
Cricket_X	0.58 (100%)	0.57	0.56	0.46	0.60	**0.62**	0.61	0.61	0.57
Cricket_Y	0.62 (100%)	0.63	0.63	0.55	0.65	**0.68**	**0.68**	**0.68**	0.64
Cricket_Z	0.60 (100%)	0.59	0.59	0.43	**0.66**	**0.66**	**0.66**	**0.66**	0.62
DiatomSizeReduction	0.85 (97.39%)	0.80	0.80	0.85	**0.94**	**0.94**	**0.94**	**0.94**	0.93
ECG200	0.87 (100%)	**0.89**	**0.89**	0.82	0.88	**0.89**	**0.89**	**0.89**	0.88
ECGFiveDays	**0.94 (99.65%)**	0.62	0.62	0.76	0.57	0.51	0.52	0.77	0.80
FaceAll	**0.88 (100%)**	0.74	0.76	0.67	0.69	0.69	0.69	0.69	0.71
FaceFour	0.64 (75.00%)	0.77	0.82	0.78	0.84	0.83	0.83	**0.86**	0.78
FacesUCR	0.74 (99.17%)	0.72	0.71	0.68	0.76	0.76	**0.77**	**0.77**	**0.77**
fish	**0.80 (100%)**	0.75	0.75	0.62	**0.80**	0.79	0.79	0.79	0.78
Gun_Point	0.92 (100%)	0.87	0.87	**0.95**	0.91	0.91	0.91	0.91	0.91
Haptics	**0.45 (96.75%)**	0.37	0.37	0.34	0.30	0.41	0.41	0.40	0.37
InlineSkate	0.26 (96.55%)	0.33	0.33	0.21	0.25	0.28	0.27	0.27	**0.34**
ItalyPowerDemand	0.93 (100%)	0.94	0.94	0.87	0.67	0.79	0.85	**0.95**	0.96
Lighting2	0.54 (88.52%)	0.70	0.70	**0.79**	0.62	0.64	0.62	0.67	0.75
Lighting7	0.53 (97.26%)	0.58	0.58	0.62	**0.70**	0.68	0.68	0.67	0.58
MALLAT	0.80 (99.70%)	0.85	0.85	0.64	0.44	0.70	0.73	0.80	**0.91**
MedicalImages	**0.73 (100%)**	0.68	0.68	0.64	0.65	0.67	0.67	0.68	0.68
MoteStrain	0.85 (100%)	**0.88**	**0.88**	0.80	0.58	0.58	0.58	0.58	0.88
OliveOil	0.40 (100%)	**0.90**	**0.90**	0.77	0.43	0.80	0.77	0.83	0.87
OSULeaf	0.55 (100%)	0.49	0.49	**0.59**	0.49	0.49	0.48	0.48	0.52
SonyAIBORobotSurface	**0.84 (99.00%)**	0.69	0.69	0.74	0.81	0.79	0.79	0.78	0.70
SonyAIBORobotSurfaceII	0.74 (100%)	0.84	0.85	0.83	0.87	**0.88**	**0.88**	**0.88**	0.86
StarLightCurves	**0.96 (100%)**	0.85	0.15	NA	0.94	0.95	0.95	0.95	0.85
SwedhLeaf	**0.86 (100%)**	0.79	0.78	0.39	0.84	0.83	0.83	0.83	0.79
Symbols	0.74 (91.86%)	0.83	0.81	0.44	0.47	0.67	0.71	0.79	**0.90**
synthetic_control	0.96 (99.67%)	0.89	0.88	0.88	0.84	0.97	**0.98**	**0.98**	0.88
Trace	0.82 (100%)	0.74	0.74	0.84	0.76	0.84	**0.86**	**0.86**	0.76
TwoLeadECG	**0.86 (99.82%)**	0.73	0.73	0.84	0.73	0.72	0.72	0.72	0.75
Two_Patterns	0.87 (100%)	0.86	0.86	0.81	**0.93**	**0.93**	**0.93**	**0.93**	0.91
uWaveGestureLibrary_X	**0.78 (100%)**	0.73	0.73	0.37	0.75	0.75	0.75	0.75	0.74
uWaveGestureLibrary_Y	**0.70 (100%)**	0.63	0.63	0.31	0.60	0.67	0.68	0.68	0.66
uWaveGestureLibrary_Z	**0.71 (100%)**	0.65	0.65	0.42	0.71	0.71	0.71	0.71	0.65
wafer	0.98 (100%)	0.99	0.99	0.98	0.97	0.99	0.99	**1.00**	**1.00**
WordsSynonyms	0.51 (97.49%)	0.59	0.59	0.42	0.64	**0.65**	**0.65**	**0.65**	0.62
yoga	**0.85 (99.83%)**	0.81	0.81	0.68	0.82	0.83	0.83	0.83	0.83
NIFECG_Thorax1	**0.90 (100%)**	0.82	0.81	NA	0.87	0.87	0.87	0.87	0.83
NIFECG_Thorax2	**0.92 (100%)**	0.88	0.88	NA	0.88	0.88	0.88	0.88	0.88

to the other methods, in the ECDIRE method, it is common that at the end of the process some series remain unclassified, due to the unreliability of their class label (see values between parentheses in Table 7.1). In this work, we choose the worst possible validation scenario for our method and include un-

classified examples as incorrectly classified instances. This includes the series that, at the end of the prediction process, do not pass the reliability tests but also includes those that are assigned with one of the initially discarded classes (see Step 1 of Section 7.2.1). However, it is possible that in some cases, the cost of making an error is larger than that of not providing a class value. Also, in some real contexts, outliers series that do not belong to any class could be present. In these cases, if a cost sensitive measure were applied, the results of the ECDIRE method regarding the committed error would further improve.

Figure 7.6 enables us to conclude that, with respect to **accuracy**, the best results are obtained by the RelClass2, RelClass3 and RelClass4 methods, the Full 1NN method and the ECDIRE method. The results of all these methods can not be said to be significantly different from each other. Moreover, the worst accuracy results are obtained by the EDSC, ECTS2 and RelClass1 methods.

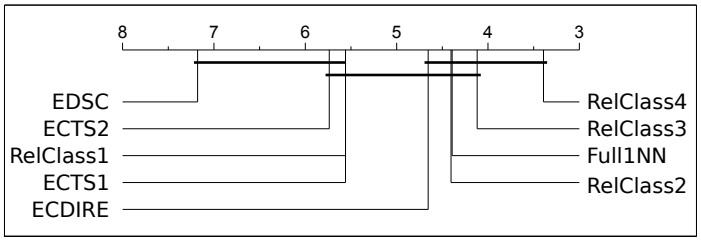


Fig. 7.6. Results from statistical tests for accuracy. The ranking is issued from the Friedman test and shows the goodness of the methods from better to worse (right to left). The bold lines show the results issued from the Holm procedure, and joins pairs of methods that do not yield statistically significant differences, for a significance level of $\alpha = 0.05$.

If the results for **earliness** are observed in Table 7.2, it can be seen that ECDIRE obtains the lowest value in 24 databases out of 45, with the other methods trailing far behind. Additionally, our method obtains the best value in the ranking issued by Friedman's test (see Figure 7.7). If we observe the results from the pairwise statistical tests, all the methods obtain significantly worse values in earliness than ECDIRE, except EDSC and RelClass1.

Although RelClass1 and EDSC obtain good results in terms of earliness, it must be said that they are among the worst with regards to accuracy, which implies that they strongly lean towards one of the objectives (earliness) and do not seek the tradeoff between earliness and accuracy, which is the final objective of early classification. Specifically, in some databases such as *CBF*, *ECGFiveDays*, *ItalyPowerDemand*, *MALLAT* and *Symbols*, where all other methods easily obtain very high accuracy values, the RelClass1 gravely compromises the accuracy by choosing extremely low earliness values. The

Table 7.2. Earliness values for ECDIRE, strict and loose versions of the ECTS, EDSC and RelClass with 4 parameter settings. The method/s with the lowest earliness value in each database are shown in bold.

Dataset	ECDIRE	ECTS1	ECTS2	EDSC	RelClass1	RelClass2	RelClass3	RelClass4
50words	**41.84**	77.10	72.86	70.14	84.95	90.79	92.20	95.45
Adiac	**40.71**	64.05	59.09	87.94	91.20	95.41	96.04	98.03
Beef	68.79	77.67	76.50	91.94	**00.21**	07.30	25.70	71.97
CBF	26.64	71.5	71.50	35.03	**00.80**	06.71	35.02	52.02
ChlorineConcentration	**16.57**	67.01	66.07	42.67	96.02	97.29	97.59	98.32
CinC_ECG_torso	44.97	59.91	**00.58**	51.81	45.96	54.12	56.58	64.62
Coffee	85.00	84.93	83.94	58.38	**05.53**	30.96	38.44	63.72
Cricket_X	**46.83**	73.26	71.80	62.17	62.71	76.01	78.68	84.86
Cricket_Y	**40.38**	67.47	66.49	55.55	67.90	79.99	82.36	87.90
Cricket_Z	**46.95**	69.21	67.86	67.24	72.05	78.57	80.36	84.60
DiatomSizeReduction	15.34	**14.88**	**14.88**	23.56	25.35	31.64	33.49	40.59
ECG200	48.25	77.13	60.11	**25.81**	41.71	63.70	68.81	84.77
ECGFiveDays	83.60	63.82	63.82	50.95	**00.74**	01.63	15.84	73.24
FaceAll	**56.38**	68.25	63.85	42.44	92.33	95.54	96.27	97.92
FaceFour	27.50	89.51	72.26	40.62	**24.58**	31.25	34.22	45.14
FacesUCR	**58.26**	89.24	87.21	64.33	82.89	90.80	92.71	96.65
fish	58.23	65.81	60.94	**54.46**	75.39	83.34	85.42	90.51
Gun_Point	**32.47**	46.92	46.92	46.20	62.14	69.42	71.33	79.67
Haptics	71.21	93.90	93.87	28.07	**02.12**	43.45	57.89	78.91
InlineSkate	**34.04**	86.42	85.08	50.02	72.41	85.10	87.31	92.70
ItalyPowerDemand	67.90	79.33	79.33	75.99	**05.09**	27.10	35.92	64.19
Lighting2	**07.50**	89.01	89.01	63.97	37.91	57.89	61.16	71.89
Lighting7	**27.11**	86.98	86.97	81.65	76.65	83.74	85.23	89.72
MALLAT	45.13	69.32	69.32	58.58	**13.97**	35.92	44.01	67.43
MedicalImages	**32.96**	54.85	53.87	45.64	75.04	86.64	88.96	93.78
MoteStrain	**15.08**	84.86	79.06	41.42	82.72	89.36	90.94	95.49
OliveOil	30.00	87.34	87.34	37.45	**00.33**	12.09	18.76	43.18
OSULeaf	**48.99**	78.20	76.59	65.33	93.58	96.44	97.10	98.48
SonyAIBORobotSurface	60.08	68.49	68.49	37.91	**36.51**	53.54	57.7	76.59
SonyAIBORobotSurfaceII	**17.66**	55.81	54.54	35.17	50.97	66.97	70.86	80.31
StarLightCurves	**29.41**	82.83	82.25	NA	85.08	88.86	90.02	93.59
SwedhLeaf	**41.78**	77.63	76.27	68.92	84.96	90.57	91.96	95.44
Symbols	24.65	46.23	51.30	78.17	**10.08**	38.18	45.82	61.85
synthetic_control	60.17	89.97	87.88	55.60	**37.66**	65.83	71.54	85.11
Trace	**41.10**	51.98	50.72	44.97	49.98	71.31	77.82	86.32
TwoLeadECG	68.44	64.43	64.43	**48.49**	73.63	81.40	83.63	89.57
Two_Patterns	96.35	86.79	86.52	**64.32**	86.87	90.72	91.82	94.80
uWaveGestureLibrary_X	84.01	86.90	85.98	79.60	**78.92**	88.04	90.09	94.90
uWaveGestureLibrary_Y	97.09	86.91	86.29	81.32	**56.14**	77.86	81.96	90.58
uWaveGestureLibrary_Z	**74.84**	85.98	85.03	76.83	77.26	89.46	91.80	96.39
wafer	**11.37**	44.38	44.38	27.59	12.65	25.48	30.75	49.60
WordsSynonyms	**63.56**	83.40	82.51	77.86	85.48	90.19	91.40	94.60
yoga	100.0	70.74	69.41	**52.15**	78.67	85.62	87.28	91.65
NIFECG_Thorax1	**62.54**	81.54	78.22	NA	89.39	92.59	93.47	95.95
NIFECG_Thorax2	**54.17**	79.83	76.84	NA	86.97	91.06	92.16	95.33

only database where the accuracy of the ECDIRE method drops so drastically in comparison to the other methods is *OliveOil*.

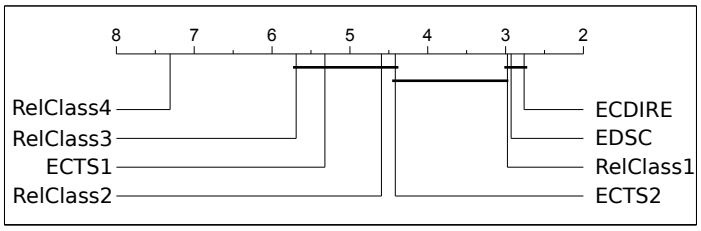


Fig. 7.7. Results from statistical tests for earliness. The ranking is issued from the Friedman test and shows the goodness of the methods from better to worse (right to left). The bold lines show the results issued from the Holm procedure, and joins pairs of methods that do not yield statistically significant differences, for a significance level of $\alpha = 0.05$.

To conclude, in Table 7.3 we provide a summary of the domination counts for the pairwise comparison between ECDIRE and the rest of the early classification proposals considered in this study. As commented previously, this last table gives us a multi-objective comparison of the methods, and is helpful to assess the overall goodness of our method. Based on the results shown in Table 7.3, it can be said that ECDIRE dominates the other methods much more often than it is dominated by them. From this, we conclude that our proposal improves the results obtained by the other state-of-the-art methods in many cases, and is thus a good solution for performing reliable and early classification of time series.

Table 7.3. Summary of domination counts for ECDIRE versus ECTS, EDSC and RelClass. The first number corresponds to the number of times ECDIRE dominates the other method, the second number refers to the times the comparison method dominates ECDIRE and the third number counts the cases in which the Pareto optimality criterion gives us no information (draws).

Compared methods	**Domination counts**
ECDIRE vs. ECTS (strict)	20/0/25
ECDIRE vs. ECTS (loose)	20/1/24
ECDIRE vs. EDSC	17/1/24
ECDIRE vs. RelClass1 ($\tau = 0.001$)	16/4/25
ECDIRE vs. RelClass2 ($\tau = 0.1$)	15/3/27
ECDIRE vs. RelClass3 ($\tau = 0.5$)	15/4/26
ECDIRE vs. RelClass4 ($\tau = 0.9$)	17/3/25

7.5 Case study: Early classification of bird songs

In addition to the experiments shown for the UCR archive, and taking into account the ubiquity of streaming data, we consider it imperative to show how our system could be applied to this type of data, which is of unknown and possibly infinite length. As a case study, we have used a small size dataset that does not allow forceful statistically significant claims. Our intention is simply to show the utility of our ideas outside of a lab setting.

In various parts of the world, automatic audio and video monitoring stations are set up in an attempt to record targeted species. For example, in Myanmar (Burma), the Myanmar Jerdon's babbler, *Chrysomma altirostre* [44], was believed to be extinct, but in 2015 a small population was discovered [50]. Discovery of such tiny populations is often made with a targeted search, placing ruggedized video monitoring stations in the field. Since such devices have limited computational resources, especially memory and battery-life, they are often designed to be triggered by sound. A simple volume threshold is not suitable because it is easily tricked by wind noise or non-target species. Clearly, it would be better for the video recording to be triggered by the target species call. This appears to offer a paradox, how can one get a recording of a bird that may be extinct? In many cases, it is possible to get the recoding of a closely related species. For example, in the above case, there is a closely related species, the Sind Jerdon's babbler (*Chrysomma altirostre scindicum*) that is common in the Indus basin of Pakistan.

Given this, we can approximately learn the typical call of the target species, and use it to trigger image/video recording. This is a natural domain for early classification; we would like to begin recording as soon as possible, before the bird can flit away.

Note that this problem setting differs slightly from the common early classification problem. In this case, we have a unique stream of data of unknown and possibly infinite length, and the aim is to detect certain significant patterns within the stream as early as possible. Nevertheless, the specific design of the ECDIRE framework provides a direct extension to this very common problem setting, unlike most early classification methods proposed in the literature.

To demonstrate this, we performed the following experiment. We select two different birds from the same species, namely the White Crowned Sparrow species (*Zonotrichia leucophrys pugetensis*) from the Xeno-Canto database [234]. Although the calls of birds from the same species are similar to each other, there are clear differences between the calls of each individual bird. As such, provided a stream containing forest noise and some occasional bird calls, the goal is to try to detect and identify each bird call as early in time as possible.

We exploit the fact that we can convert high-dimensional (44,100Hz) audio into a low dimensional Mel-Frequency Cepstral Coefficients (MFCC) space (100Hz). It has long been known that the MFCC space is amenable to bird

song classification [107, 213]. While most classification algorithms attempt to use up to twelve coefficients, for simplicity and visual intuitiveness we consider only one here. As Figure 7.8 shows, a single MFCC coefficient does suggest that little interclass variability is present, which bodes well for the task at hand.

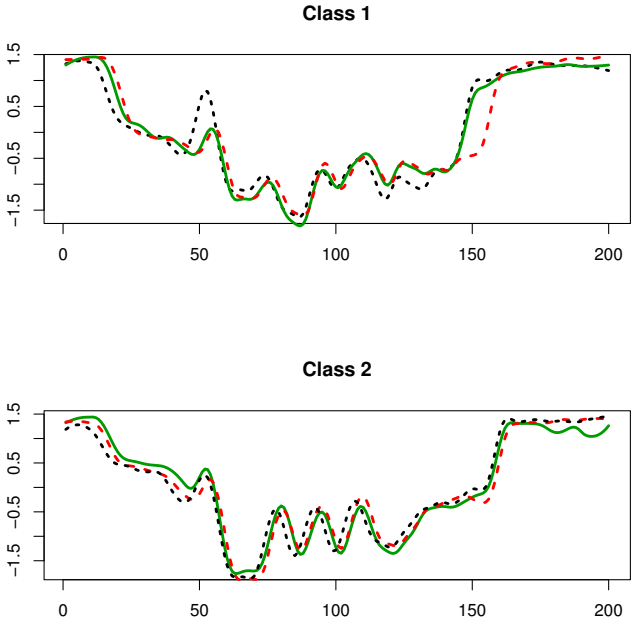


Fig. 7.8. Three example calls from each class.

To apply the ECDIRE method, we build a training set consisting of 5 and 8 calls from each of the birds of length 200, respectively. As shown in Section 7.2.1, we apply a cross validation process to this data and build the corresponding timeline. From this process, we obtain one safe timestamp for each class, with values of 15% (t_1) and 20% (t_2) of the length of the bird calls, respectively. Additionally, we calculate the reliability thresholds as explained in Section 7.2.1 for these early timestamps.

Now, we use the information obtained from this process to obtain class predictions of a new testing stream of data. In this experiment, the length of the stream is of 1600 points, but the method could be applied identically to a stream of unknown or even infinite length. As previously commented, this stream will contain a few bird calls and background noise from the forest in between the calls. Although the ECDIRE method is initially defined to classify finite time series, the set of *safe* timestamps for each class provides a direct way to apply it to this other problem setting. This is because the *safe*

timestamps directly define suitable sizes for the sliding windows that will be used to analyze the incoming stream.

Thus, to identify the call of the first bird, we analyze the stream with a sliding window of size t_1 (15% of the length of the pattern, thus 30 data points). Every time a new data point arrives, we slide the window one position, and, by using a GP classifier built with all the training data for the corresponding earliness value t_1 (see Section 7.2.1), we obtain a class prediction for the data section enclosed within. Note that since the test stream is 1600 data points long and the sliding window is 30 points wide, we will perform 1571 class predictions. To finish, as shown in Section 7.2.2, we apply the reliability test and decide whether we make a final prediction or choose to abstain. In this case, given the high level of noise present in the incoming testing stream, instead of using the minimum to calculate the reliability thresholds, we decide to use a stricter threshold and calculate the mean of the differences between the posterior probabilities of the target class and the next most probable class obtained in the training phase. In Figure 7.9 we show the results obtained for the first bird.

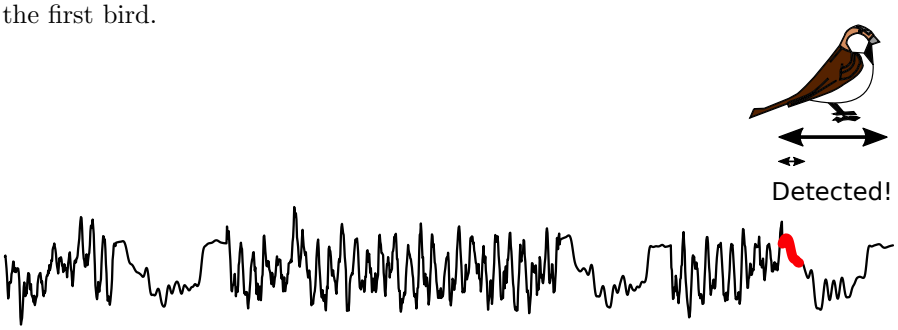


Fig. 7.9. Detection and identification of the first bird by using the ECDIRE method, using the mean to calculate the reliability threshold. The presence of this bird in the stream is represented by an image of a bird, and the first window in which the bird is identified is shown by a bold line. The arrows indicate the length of the call and the portion used to detect it, respectively (from top to bottom).

It can be seen in Figure 7.9 that the first bird is identified perfectly and very early in time, with a window size of only 15% of the length of the pattern. Additionally, for this bird, our method issues no false positives. Given the large number of class predictions that we carry out, these results are indeed impressive.

For the second bird, we follow the same procedure, but take the sliding window of size t_2 (20% of the length of the patterns, thus 40 data points), provided by the ECDIRE method. Note that with this setting, in this case, we make 1561 class predictions. By using the mean difference between the posterior probabilities of the target class and the next probable class obtained

in the training phase to calculate the reliability threshold, we obtain the result shown in Figure 7.10.

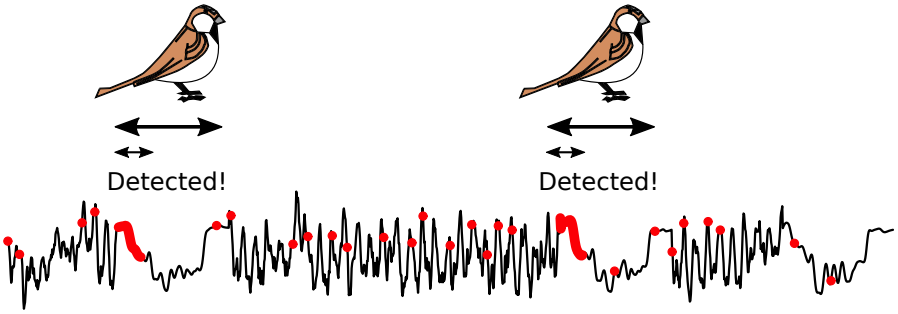


Fig. 7.10. Detection and identification of the second bird by using the ECDIRE method, using the mean to calculate the reliability threshold. The presence of this bird in the stream is represented by an image of a bird, and the first window in which the bird is identified is shown by a bold line. The arrows indicate the length of the call and the portion used to detect it, respectively (from top to bottom). False positives are represented by round points in the stream. Note than when the method issues a false positive, it usually commits the same mistake in several consecutive windows. In the image, for the sake of clarity, only the first window is marked.

This bird is also identified correctly in all cases, with a window size of only 20% of the length of the bird call. However, contrary to the first case, for this bird we obtain many false positives. The results obtained are still better than those that would be obtained by a simple volume threshold, which would not be able to distinguish between birds and would constantly be triggered by the background forest noise, but they could be improved by better tuning the reliability threshold. For example, if we take the median instead of the mean, we obtain a stricter reliability threshold and, thus, the results shown in Figure 7.11. In this case, the number of false positives is much lower but the method fails to identify the bird on one of its appearances.

We insist on the fact that the aim of this section is to simply demonstrate the utility and direct application of ECDIRE to a problem of this type, and that the small size of the training set does not allow us to make any claims regarding the results. Specifically, recall that the reliability thresholds are calculated based on the differences between the posterior probabilities obtained in the training process. The extremely small number of training instances make the mean and median values not very robust in this example, which is why the results change so drastically. However, with a larger training set, we would be able to better tune the reliability thresholds in order to remove the false positives while retaining the correct class identifications. Optimally tuning the reliability threshold based on the level and type of noise in the

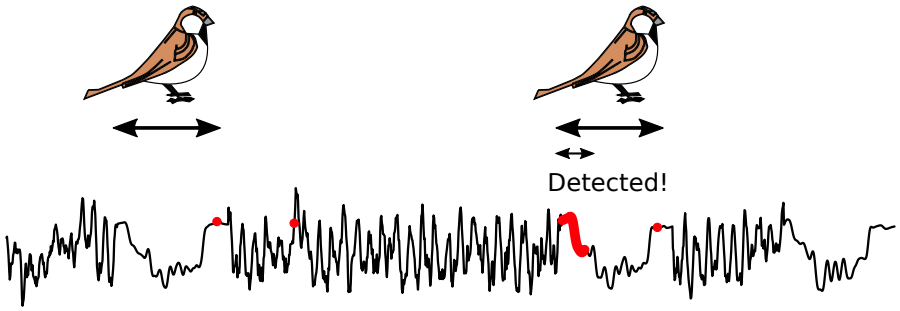


Fig. 7.11. Detection and identification of the second bird by using the ECDIRE method, using the median to calculate the reliability threshold. The presence of this bird in the stream is represented by an image of a bird, and the first window in which the bird is identified is shown by a bold line. The arrows indicate the length of the call and the portion used to detect it, respectively (from top to bottom). False positives are represented by round points in the stream. Note than when the method issues a false positive, it usually commits the same mistake in several consecutive windows. In the image, for the sake of clarity, only the first window is marked.

incoming stream could be an interesting area of study, which is proposed as future work.

7.6 Conclusions and Future Work

In this chapter, we have proposed a method for early classification of time series based on probabilistic classifiers. Our proposal incorporates two strategies that deal with two of the main issues that arise with other early classification methods published in the literature. On the one hand, in order to avoid excessive calculations in the prediction phase, the discriminativeness of the classes is analyzed at different time intervals and a methodology is proposed to identify the time instant at which we can begin to make accurate predictions for each class. This procedure allows a direct application to infinite streams of data, which had not been dealt with in other early classification works. On the other hand, the goodness of the issued class labels is controlled, discarding predictions that do not yield a sufficiently large class probability.

Contrary to the other early classification methods used for comparison, the proposal fulfills the three desirable characteristics for early classification methods recommended by Ghalwash et al. [71]. First, the method obtains competitive results with respect to the **earliness** of the prediction in comparison with other state-of-the-art early classification methods. Indeed, the few methods that obtain similar results in earliness are weaker than ECDIRE with respect to accuracy. When evaluating the methods from a multi-objective point of view, it can be concluded that the ECDIRE method dominates the

rest of the methods much more frequently than it is dominated. All this proves the usefulness of the method to obtain reliable and early classifications of time series.

Secondly, with regards to **understandability**, this method is able to discover at which time instant each class can be reliably discriminated from other classes. This information is valuable and may help the user to understand the underlying patterns in the data. Additionally, the method can be applied by simply setting a grid of early timestamps to consider and the $perc\_acc_c$ parameter. These parameters can easily be understood and fixed by any user, expert or otherwise. More experienced users may modify other parameters such as the distance measure d, the kernel function, etc. but this is not a requirement. In this line, an obvious proposal for future work consists of studying the behavior of different distance measures and kernels within this framework.

As a last point, a straightforward **measure for uncertainty** of the predictions issued by our framework is provided by the probabilities extracted from the classifiers employed within the framework. A more informative uncertainty measure could incorporate additional knowledge acquired in the learning phase, such as information about the discriminativeness of the classes in the database in different time intervals. The design of this measure is proposed for future elaboration.

8

Automatic distance measure selection and early classification for the problem of travel time prediction

In this last chapter, we return to the task of travel time modeling, and specifically to the problem of travel time prediction. We focus again on the combined travel time prediction models based on time series clustering pre-processes. Recall that these models initially apply a time series clustering process to identify the different long term patterns in the data (in our case, daily patterns). Then, a different travel time prediction model is used for each cluster.

As concluded in Chapter 5, these combined models give rise to a couple of non-trivial problems: the selection of a suitable distance measure to cluster the database, and the assignment of new incoming days to specific clusters using incomplete information. These two general time series data mining problems have been addressed in Chapters 6 and 7 respectively, and a possible solution has been proposed for each of them.

Now, in this chapter, the aim is to analyze the performance of these two time series data mining contributions, within the particular scenario of travel time prediction. Note that, as seen in Chapter 5, the naive historical predictor is the model that benefits the most from applying time series clustering pre-processes. As such, in this chapter, we will restrict our analysis to this prediction model. Accordingly, the intention is not to propose a model that is competitive with the state-of-the-art. Instead, we just want to analyze the effect of applying the two main methodological contributions of this dissertation to this specific problem.

8.1 Summary of the proposal

The strategy followed in this chapter is very similar to that applied in Chapter 5. However, in this case we want to validate the two contributions from Chapters 6 and 7, so we compare the performances of five different procedures.

The first procedure (P1) is the baseline procedure, where no time series clustering is performed, and the historical predictor (see Section 4.2.1) is applied directly. This procedure is equivalent to P1 from Chapter 5.

The other four procedures (P2, P3, P4, P5) are more complex and consist of two phases, the learning phase and the prediction phase, which are explained in detail in the following sections.

8.1.1 Learning phase

The learning phase of P2, P3, P4 and P5 consists of three steps, and is carried out using only a training set of historical travel time measurements, arranged in daily time series (see Figure 8.1). In the first step, a distance measure is selected. In the second step, by using the selected distance measure (d) and the K-medoids algorithm, the days that conform the historical data are clustered into a set of different *daily patterns*. Finally, based on the clustered training set, we build an early classifier.

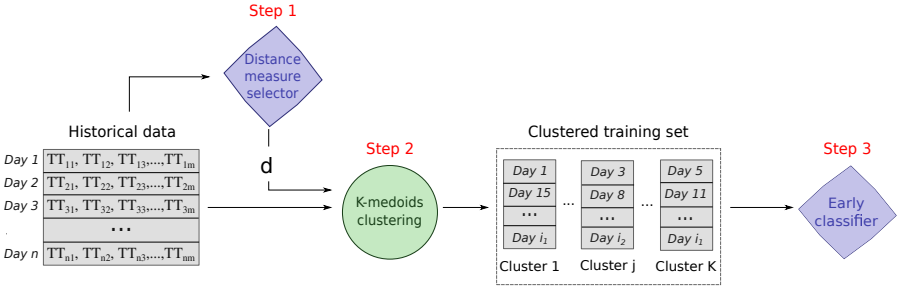


Fig. 8.1. Summary of the steps in the learning phase of the four non-naive procedures.

The difference between the four non-naive procedures lies in two aspects: the method we use to select the distance measure, and the type of early classifier we build. In Table 8.1 the particular choices for each procedure are shown. As commented in previous chapters, ED is the acronym used for the Euclidean distance. ADMS refers to the automatic distance measure selector proposed in Chapter 6. The 1NN early classification approach consists of comparing the

Table 8.1. Distance measure selector and early classifier used in the four non-naive procedures.

Procedure	Distance Measure Selector	Early Classifier
Procedure 2 (P2)	Always use ED	1NN
Procedure 3 (P3)	ADMS	1NN
Procedure 4 (P4)	Always use ED	ECDIRE
Procedure 5 (P5)	ADMS	ECDIRE

available data from the incoming day with the representatives or medoids of the different clusters, using the distance measure selected in each case (see Chapter 5). Finally, ECDIRE is the early classification method introduced in Chapter 7.

Note that the 1NN early classifier does not require any real training, only the identification of the medoids obtained in the clustering phase. On the contrary, the ECDIRE early classifier must be trained as shown in Section 7.2.1.

8.1.2 Prediction phase

Given a prediction time (t) and a forecast horizon (Δ), the objective of this phase is to provide a prediction of the travel time at time $t+\Delta$, by using the clustered historical dataset and the early classifier obtained in the learning phase. This will be done in two steps, which are shown graphically in Figure 8.2 for the 1NN early classifier and in Figure 8.3 for ECDIRE.

Suppose, instant $t+\Delta$ belongs to day D. If t and $t+\Delta$ do not belong to the same day (for example when we want a 6-hour ahead prediction at 22:00), at the time we make the travel time prediction (t) we do not have any travel time information about day D. In this case, we can not assign D to any day type, so we directly use the basic historical model that uses all the training data to provide a travel time prediction at time $t+\Delta$. On the contrary, if t and $t+\Delta$ belong to the same day D, then we try to assign D to a particular day type. This will be done by introducing the travel time measurements available from that day ($TT_{D,1}, TT_{D,2}, ..., TT_{D,t}$) into the early classifier built in the learning step. The procedure will differ depending on the type of early classifier chosen in each case.

In the case of 1NN, the available data from day D is compared with the medoids obtained from the clustering process, and the closest cluster is selected. Accordingly, if t and $t+\Delta$ belong to the same day, we will always obtain a class prediction. As such, only the training data that belongs to that

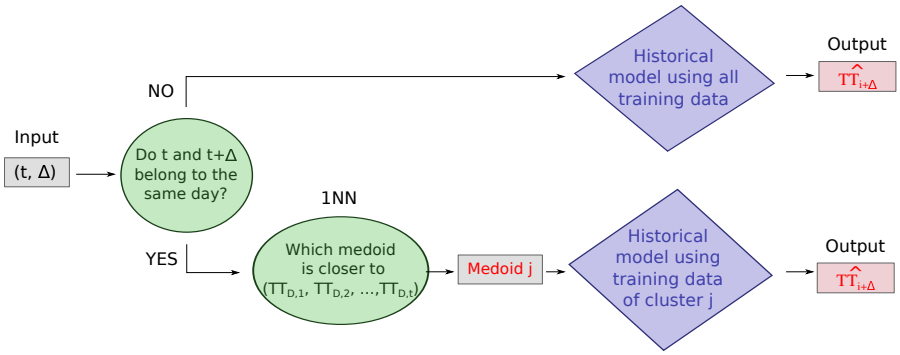


Fig. 8.2. Summary of the prediction phase when using the 1NN early classifier.

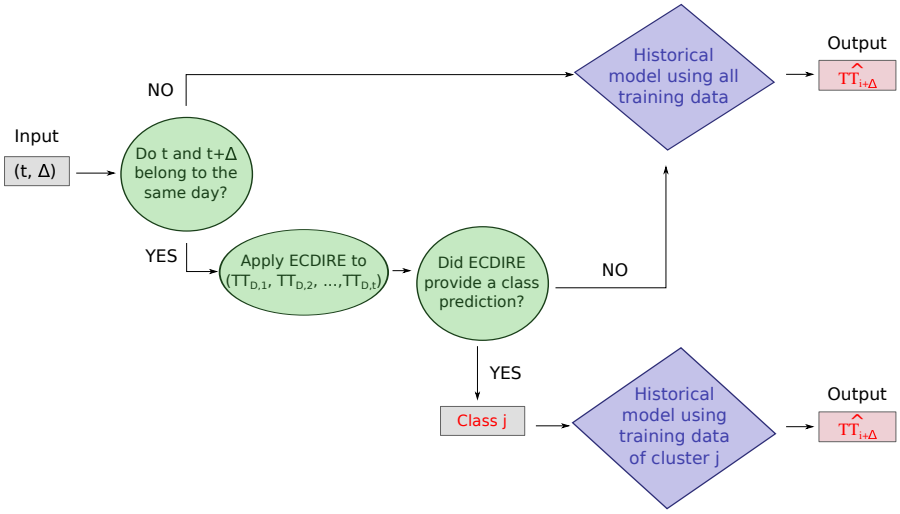


Fig. 8.3. Summary of the prediction phase when using the ECDIRE early classifier.

cluster will be used within the historical model to provide a prediction of travel time at time $t + \Delta$.

However, in the case of ECDIRE, as explained in Chapter 7, class predictions will only be issued when they are reliable. So, in some cases, the ECDIRE classifier will abstain and we will have to use the basic historical predictor to provide the travel time predictions. In some other cases, a class output will be provided by ECDIRE and hence, the travel time predictions will be obtained by using the historical model that considers only the training data from the corresponding cluster. The reader can access Figure 7.4 and Sections 7.2.1 and 7.2.1 for more information on why ECDIRE does not always provide a class prediction.

Finally, we must emphasize that, although the basic historical model is not affected by the prediction horizon (Δ), when we combine it with a time series clustering pre-process and an early classifier, the results for different prediction horizons usually vary. This is because the instant in which the travel time prediction is made (t) determines the quantity of travel time data ($TT_{D,1}, TT_{D,2}, ..., TT_{D,t}$) that is available from the current day. This information is used to find out the type of day which D belongs to and, in this context, different prediction horizons will have to be evaluated separately.

8.2 Experimentation

In this section, we extend the experiments shown in Chapter 5, with the aim of testing whether the proposed automatic distance selection method (Chapter 6) and the ECDIRE early classifier (Chapter 7) are useful for the problem of

travel time prediction. For this, we evaluate and compare the performance of the four non-naive procedures explained in the previous section.

8.2.1 The data

The data used in this experimentation is once again obtained from the RTA competition held at Kaggle. Recall that the data consists of travel time measurements taken every 3 minutes from February 2008 to July 2010 (see Chapter 5 for more information on this data). The data is pre-processed in the same manner as in the experimentation carried out in Chapter 5. However, in this case, we consider all the road sections available, 59 in total (all except sections 18 and 57, for which not all the historical data is provided).

As in Chapter 5, the datasets are also divided into train/test validation sets following a proportion of 70%/30% and respecting the chronological order. This results in a database of 344 days for training purposes and 147 for testing for each road section.

8.2.2 Experimental setup

In the following sections the parameter selection and other choices made during the learning and prediction phases are explained in detail, as well as the evaluation measures used in the experimentation.

Setup for the learning phase

In the first step of the learning phase, a distance measure is selected to cluster the training set. For procedures 2 and 4, the Euclidean Distance (ED) is used directly, so no further explanations are necessary. On the contrary, for procedures 3 and 5, we use the multi-label classifier that automatically selects the distance measures (ADMS, from now on) proposed in Chapter 6. This classifier is trained using the ECC algorithm [172] and the UCR and synthetic databases (see Section 6.4.1), and it considers the same 5 distance measures introduced in the mentioned chapter (ED, DTW, EDR, TQuest and the Fourier Coefficient based distance) with the same parameter values. The reason why we don't use any traffic data for training the ADMS is the lack of ground truth clustering results for these databases (see Section 6.3 for an explanation on why this information is necessary).

In order to apply the ADMS, we calculate the characteristics of the training sets obtained from Kaggle as explained in Section 6.1 for the 59 road sections, and we introduce each of them as input into the classifier, obtaining the most suitable distance set for each route. From this set of distances we choose only one by using the silhouette and Dunn indices obtained from the different clustering results and following a method similar to that proposed in [4]. These two measures are well known cluster validation indices, and the measure

with the best sum of ranks for both indices is chosen as the most suitable. If there are ties, one option is selected uniformly at random. Furthermore, the parameter for the selected distance measure and the number of clusters are also picked in the same way. The number of clusters considered are 2, 3, 4 and 5, as in Chapter 5.

The second step in the training phase consists in clustering the training set. As commented above, the clustering of the training sets is carried out with the K-medoids algorithm. As in Chapter 5, this algorithm is always applied with 15 different initializations, and the solution with the highest silhouette is selected in order to avoid local minima.

Finally, we also need to specify the parameters used to train the early classifiers. As commented previously, the 1NN method requires no real training, so no parameters must be specified. In the construction of the ECDIRE classifier, all the parameters are defined as in Chapter 7, except the threshold for the reliability test. Given the large quantity of noise and outlier series present in the data, instead of using the minimum of the differences between the two largest posterior probabilities obtained in the training phase (see Section 7.2.1), we try different quantiles {20%, 30%, 40%, 50%, 60%, 70%, 80%}.

Additionally, we have observed that in the presence of very unbalanced classes, the probabilistic GP classifiers within ECDIRE tend to assign all or most of the predictions to the majority class, especially when the series in different clusters are very similar to each other at the beginning and the class predictions are made early in time. Given the design of ECDIRE (see Chapter 7) this precipitates the prediction of the majority class, preventing the consideration of the other classes. To avoid this, we have added a naive constraint to the construction of the timeline of ECDIRE: If in a given timestamp, the accuracies of all the classes except one are below 0.1, then we will continue to the next timestamp and no classes will be added to the timeline.

Setup for the prediction phase

First, we must provide some specifications about the early classifiers used in the prediction phase. Regarding the 1NN classifier, to reduce the computational cost and to enable a fair comparison with the ECDIRE method, which can only output class predictions with a granularity of 5% of the length of the series (see Chapter 7), the 1NN comparisons will not be performed every time a new data point arrives, but instead they will be carried out with a frequency of 5% of the length of the series.

Furthermore, in the case of both 1NN and ECDIRE, when calculating the distances between the training series and the partial time series (subsequences) obtained from new incoming days, in the case of the elastic measures (EDR and DTW), we do not truncate the training series directly at the size of the incoming subsequence before the comparison. With the aim of respecting the elastic nature of the distance measures also in these partial comparisons, we add an extra window of data points to the training series. The length of this

window is obtained by calculating all the possibilities and choosing the best choice, as explained in [207].

Finally, regarding the calculation of the travel time predictions, since the historical model has no parameters, we only have to select a set of prediction horizons. Contrary to the experimental setup of Chapter 5, in this case we consider more prediction horizons, namely, 15-minute, 30-minute, 45-minute, 60-minute, 90-minute, 2-hour, 3-hour, 6-hour and 12-hour ahead predictions.

Evaluation method

As in Chapter 5, the Mean Absolute Percentage Error (MAPE) (see Equation 5.2) is used as a performance evaluation measure. Indeed, we take P1 as the baseline and for each procedure (Pi) and route (R) we calculate the following value:

$$diffMAPE_{Pi,R} = \frac{MAPE_{P1,R} - MAPE_{Pi,R}}{MAPE_{P1,R}} * 100 \tag{8.1}$$

The objective is to obtain large positive $diffMAPE$ values.

Finally, in order to obtain additional information on the performance and behavior of the procedures, we also calculate the following values:

- $H_{\Delta,R}$=percentage of points (t, Δ) for which no early class prediction is provided by ECDIRE and the basic historical predictor needs to be used (see Figure 8.3).
- E_R=Mean time of the day in which the ECDIRE method provides a class prediction for the testing days in route R. This measure is similar to the mean earliness used in Chapter 7.
- M_R=Percentage of testing days from route R that are never classified early with the ECDIRE method because of lack of reliability.

8.2.3 Results

First, in Tables 8.2, 8.3, 8.4, 8.5 and 8.6 we show the $diffMAPE$ results obtained for all the routes. For the sake of clarity, we only show the results for the 15-minute, 45-minute, 90-minute and 6-hour prediction horizons and the 20% quantile for the reliability threshold. However, the results of the rest of the prediction horizons and quantiles follow the same trend.

Considering the results shown in these tables, firstly, we note that, overall, the distance measures selected by the ADMS do not seem to yield better clustering solutions than those obtained by ED. Indeed, note that the results vary greatly from route to route. Additionally, note that as the prediction horizon increases, the utility of applying clustering diminishes, and it is better to simply use P1. Secondly, if we compare the results obtained by both early classification methods considered, overall, we note that the results of

ECDIRE are usually closer to 0 than those of 1NN. Indeed, for some routes, and especially as the prediction horizon increases, the $diffMAPE$ values are exactly 0, which means that no improvement or deterioration is obtained in comparison to P1. On the contrary, the results for 1NN are much more extreme. When the prediction horizons are small, 1NN sometimes obtains very good results. However, as the prediction horizon increases, the results tend to get worse and worse.

Table 8.2. $diffMAPE$ results for the routes 1-10 and for different prediction horizons (Δ). The values between parentheses correspond to $H_{\Delta,R}$ values. Finally, the last column identifies the distance measure used in each case

R		Δ=15min	Δ=45min	Δ=90min	Δ=6h	E_R	M_R	d
1	P2	8.01	4.62	3.23	-1.26	-	-	ED
	P3	5.56	5.01	4.82	2.28	-	-	EDR
	P4	-2.28 (92.95)	-2.00 (93.71)	-1.78 (94.86)	0.00(100.00)	19:12	36.73	ED
	P5	1.88 (70.78)	1.74 (72.47)	1.65 (75.01)	1.23 (90.15)	15:08	80.95	EDR
2	P2	16.87	-11.52	-32.89	-17.23	-	-	ED
	P3	20.49	-7.46	-20.43	-6.02	-	-	F
	P4	-0.03 (86.70)	-0.03 (88.15)	-0.04 (90.30)	0.00(100.00)	19:12	69.39	ED
	P5	0.06 (87.25)	0.02 (89.04)	0.01 (91.72)	0.00(100.00)	20:14	85.71	F
3	P2	28.19	17.17	10.85	8.03	-	-	ED
	P3	-7.42	-18.52	-34.98	-33.99	-	-	DTW
	P4	3.86 (79.25)	0.43 (81.32)	-0.19 (84.40)	0.00 (99.93)	18:47	99.32	ED
	P5	0.30 (82.40)	-0.09 (83.75)	0.11 (85.78)	-0.06 (97.89)	17:16	64.63	DTW
4	P2	16.24	13.48	3.94	-1.09	-	-	ED
	P3	-2.29	-2.47	-1.61	-1.05	-	-	EDR
	P4	-0.58 (84.84)	-0.64 (86.82)	-0.68 (89.78)	0.00(100.00)	19:58	95.24	ED
	P5	-0.28 (85.20)	-0.16 (86.48)	-0.10 (88.40)	0.00 (99.87)	18:00	61.22	EDR
5	P2	9.92	3.32	-0.16	-1.10	-	-	ED
	P3	-0.11	-12.42	-20.01	-29.40	-	-	F
	P4	0.39 (95.95)	0.34 (96.39)	0.35 (97.05)	0.00(100.00)	19:12	21.09	ED
	P5	-0.13 (87.33)	-0.17 (88.70)	-0.17 (90.77)	0.00(100.00)	19:12	65.99	F
6	P2	12.82	-89.56	-92.81	-133.98	-	-	ED
	P3	-22.28	-23.68	-30.80	-40.48	-	-	DTW
	P4	0.00(100.00)	0.00(100.00)	0.00(100.00)	0.00(100.00)	-	0.00	ED
	P5	14.53 (70.81)	8.98 (72.89)	2.11 (76.02)	0.10 (94.78)	16:48	100.00	DTW
7	P2	4.43	4.58	6.04	7.59	-	-	ED
	P3	-5.05	-11.56	-11.44	-10.28	-	-	F
	P4	0.12 (90.50)	0.08 (91.53)	0.04 (93.09)	0.00(100.00)	19:12	49.66	ED
	P5	0.09 (88.92)	0.09 (90.12)	0.08 (91.91)	0.00(100.00)	19:12	57.82	F
8	P2	-4.23	-11.76	-114.45	-231.94	-	-	ED
	P3	-43.38	-72.41	-77.06	-88.63	-	-	DTW
	P4	3.21 (88.14)	0.81 (89.16)	-0.05 (90.70)	0.00 (99.90)	18:00	48.98	ED
	P5	0.00(100.00)	0.00(100.00)	0.00(100.00)	0.00(100.00)	-	0.00	DTW
9	P2	-7.02	-27.70	-62.96	-27.74	-	-	ED
	P3	-34.45	-48.23	-67.36	-89.24	-	-	DTW
	P4	-0.18 (86.89)	-0.13 (88.38)	-0.11 (90.58)	0.00(100.00)	19:23	71.43	ED
	P5	0.00(100.00)	0.00(100.00)	0.00(100.00)	0.00(100.00)	-	0.00	DTW
10	P2	-6.73	-7.83	-7.57	3.72	-	-	ED
	P3	-22.55	-26.90	-26.44	-31.43	-	-	EDR
	P4	-0.07 (83.31)	-0.07 (85.12)	0.00 (87.82)	0.00(100.00)	19:12	87.07	ED
	P5	0.00 (95.30)	-0.02 (95.81)	0.02 (96.58)	0.00(100.00)	19:12	24.49	EDR

Table 8.3. $diffMAPE$ results for the routes 11-24 and for different prediction horizons (Δ). The values between parentheses correspond to $H_{\Delta,R}$ values. Finally, the last column identifies the distance measure used in each case

R		Δ=15min	Δ=45min	Δ=90min	Δ=6h	E_R	M_R	d
11	P2	-50.19	-55.31	-43.93	-47.72	-	-	ED
	P3	-15.63	-41.14	-17.83	-16.66	-	-	DTW
	P4	-0.16 (90.73)	-0.12 (92.43)	0.06 (94.99)	0.00(100.00)	21:04	81.63	ED
	P5	0.23 (96.25)	0.04 (98.12)	0.00(100.00)	0.00(100.00)	22:48	89.80	DTW
12	P2	12.15	11.29	18.87	23.69	-	-	ED
	P3	-53.93	-65.45	-99.41	-113.97	-	-	DTW
	P4	0.11 (91.59)	0.09 (93.65)	0.06 (96.00)	0.00(100.00)	21:45	98.64	ED
	P5	0.00(100.00)	0.00(100.00)	0.00(100.00)	0.00(100.00)	-	0.00	DTW
13	P2	10.10	6.46	7.84	6.86	-	-	ED
	P3	-21.41	-50.64	-52.20	-74.84	-	-	DTW
	P4	3.72 (77.03)	0.05 (79.10)	0.15 (82.22)	0.00 (99.83)	18:15	99.32	ED
	P5	0.00(100.00)	0.00(100.00)	0.00(100.00)	0.00(100.00)	-	0.00	DTW
14	P2	1.13	-6.78	-33.52	-53.80	-	-	ED
	P3	9.38	11.27	6.99	3.89	-	-	TQuest
	P4	0.57 (80.30)	-0.66 (82.00)	-0.12 (84.56)	0.00 (99.83)	18:00	81.63	ED
	P5	0.00(100.00)	0.00(100.00)	0.00(100.00)	0.00(100.00)	-	0.00	TQuest
15	P2	-1.71	-45.06	-67.07	-101.94	-	-	ED
	P3	-85.79	-49.03	-73.45	-21.04	-	-	DTW
	P4	4.10 (81.58)	0.20 (83.17)	-0.13 (85.56)	0.00 (99.84)	18:00	76.19	ED
	P5	-0.02 (99.22)	0.00 (99.30)	0.00 (99.43)	0.00(100.00)	19:12	4.08	DTW
16	P2	-6.75	-60.66	-30.48	-25.85	-	-	ED
	P3	-4.31	-4.31	-4.06	-3.31	-	-	EDR
	P4	-0.02 (85.07)	0.02 (86.70)	0.02 (89.13)	0.00(100.00)	19:13	78.23	ED
	P5	-0.32 (91.25)	-0.25 (93.24)	-0.15 (96.22)	0.00(100.00)	21:36	95.24	EDR
17	P2	-1.27	0.67	-5.05	-1.65	-	-	ED
	P3	6.35	-9.91	-8.50	3.14	-	-	EDR
	P4	-0.29 (85.43)	-0.05 (87.02)	-0.02 (89.38)	0.00(100.00)	19:12	76.19	ED
	P5	2.32 (16.08)	1.92 (18.17)	1.80 (21.28)	1.70 (39.95)	03:36	100.00	EDR
19	P2	93.84	93.84	88.87	69.49	-	-	ED
	P3	93.84	93.84	88.87	69.49	-	-	EDR
	P4	3.99 (95.83)	1.99 (97.91)	0.00(100.00)	0.00(100.00)	22:48	100.00	ED
	P5	13.28 (86.10)	11.32 (88.14)	8.40 (91.21)	0.00(100.00)	20:24	97.96	EDR
20	P2	-20.86	-46.10	-49.21	-49.42	-	-	ED
	P3	-1.30	-1.30	-1.03	-0.62	-	-	EDR
	P4	2.95 (77.51)	0.48 (79.58)	-0.14 (82.69)	0.00 (99.86)	18:22	99.32	ED
	P5	-1.20 (9.11)	-1.10 (11.17)	-0.92 (14.26)	-0.62 (32.69)	01:40	(98.64)	EDR
21	P2	-217.42	-231.52	-260.77	-262.51	-	-	ED
	P3	-0.69	-0.82	-0.52	-0.50	-	-	EDR
	P4	-0.08 (99.84)	-0.11 (99.85)	0.00 (99.87)	0.00(100.00)	18:00	0.68	ED
	P5	-0.63 (8.70)	-0.57 (10.75)	-0.48 (13.81)	-0.36 (32.15)	01:25	(97.96)	EDR
22	P2	18.27	16.06	11.92	8.21	-	-	ED
	P3	7.47	7.17	6.51	4.84	-	-	EDR
	P4	-0.22 (91.83)	-0.09 (92.78)	-0.05 (94.20)	0.00(100.00)	19:30	45.58	ED
	P5	6.35 (13.97)	5.75 (16.02)	4.99 (19.09)	4.81 (37.29)	02:40	(97.96)	EDR
23	P2	-23.43	-75.34	-84.83	-103.01	-	-	ED
	P3	-235.59	-328.60	-633.65	-1290.49	-	-	DTW
	P4	14.34 (57.85)	14.29 (59.87)	14.29 (62.90)	14.68 (73.69)	13:22	97.28	ED
	P5	0.00(100.00)	0.00(100.00)	0.00(100.00)	0.00(100.00)	-	0.00	DTW
24	P2	5.46	6.45	-32.77	-53.36	-	-	ED
	P3	2.27	1.65	1.23	-0.81	-	-	EDR
	P4	3.38 (80.82)	2.88 (82.90)	2.11 (86.01)	0.00(100.00)	19:12	100.00	ED
	P5	4.88 (81.86)	5.72 (83.14)	6.49 (85.06)	1.34 (96.53)	16:41	61.22	EDR

Table 8.4. $diffMAPE$ results for the routes 25-37 and for different prediction horizons (Δ). The values between parentheses correspond to $H_{\Delta,R}$ values. Finally, the last column identifies the distance measure used in each case

R		Δ=15min	Δ=45min	Δ=90min	Δ=6h	E_R	M_R	d
25	P2	33.59	33.86	23.68	7.97	-	-	ED
	P3	58.08	57.81	48.72	27.24	-	-	DTW
	P4	23.23 (11.51)	18.37 (13.60)	12.52 (16.73)	9.50 (35.41)	02:30	100.00	ED
	P5	39.44 (5.84)	36.65 (7.93)	30.53 (11.06)	18.01 (29.79)	01:12	100.00	DTW
26	P2	30.75	36.51	32.15	23.49	-	-	ED
	P3	32.74	30.05	28.17	23.39	-	-	DTW
	P4	2.10 (90.54)	1.02 (92.56)	-0.16 (95.59)	-0.27 (99.52)	21:27	96.60	ED
	P5	32.53 (11.20)	30.11 (13.29)	27.31 (16.42)	18.55 (35.11)	02:25	100.00	DTW
27	P2	69.74	69.78	62.48	43.92	-	-	ED
	P3	20.45	-82.89	11.26	-2.00	-	-	EDR
	P4	0.59 (95.54)	0.00 (97.60)	-0.16 (99.67)	0.00 (99.93)	22:43	98.64	ED
	P5	7.54 (11.06)	1.92 (13.15)	-3.10 (16.29)	-3.96 (34.97)	02:24	100.00	EDR
28	P2	14.22	14.22	6.72	-3.40	-	-	ED
	P3	-131.45	-195.39	-149.20	-151.96	-	-	EDR
	P4	11.46 (14.73)	8.01 (16.69)	3.71 (19.63)	-3.02 (37.20)	01:58	93.88	ED
	P5	-3.79 (11.06)	-7.41 (13.15)	-11.59 (16.29)	-11.55 (34.97)	02:24	100.00	EDR
29	P2	-4.98	-14.16	-14.46	-6.78	-	-	ED
	P3	28.30	-1.69	17.88	1.37	-	-	EDR
	P4	0.61 (86.33)	-0.11 (87.52)	-0.09 (89.31)	0.00 (99.88)	18:03	57.14	ED
	P5	15.04 (11.06)	10.90 (13.15)	5.02 (16.29)	0.09 (34.97)	02:24	100.00	EDR
30	P2	-27.17	-43.87	-50.30	-26.63	-	-	ED
	P3	12.40	-25.37	-28.40	-228.47	-	-	EDR
	P4	29.20 (11.34)	27.73 (13.42)	26.00 (16.56)	24.90 (35.24)	02:27	100.00	ED
	P5	-0.73 (95.37)	-0.63 (97.41)	-0.43 (99.47)	-0.10 (99.72)	22:40	(97.96)	EDR
31	P2	1.63	2.17	1.44	13.77	-	-	ED
	P3	-18.72	-18.72	-16.55	-21.15	-	-	EDR
	P4	8.85 (50.79)	8.82 (52.87)	9.04 (55.98)	0.99 (74.63)	11:55	99.32	ED
	P5	-14.28 (63.23)	-13.64 (65.21)	-12.84 (68.16)	-7.19 (85.79)	14:28	94.56	EDR
32	P2	26.12	26.09	22.34	13.50	-	-	ED
	P3	32.42	32.35	37.20	34.26	-	-	EDR
	P4	5.42 (80.96)	4.17 (83.04)	2.84 (86.15)	0.00(100.00)	19:13	100.00	ED
	P5	0.06 (99.90)	0.06 (99.92)	0.06 (99.94)	0.00(100.00)	20:24	0.68	EDR
33	P2	25.97	25.97	15.53	5.79	-	-	ED
	P3	29.97	29.67	17.09	7.44	-	-	EDR
	P4	23.36 (12.08)	18.83 (14.06)	12.74 (17.01)	5.44 (34.58)	01:29	94.56	ED
	P5	13.40 (11.06)	9.26 (13.15)	6.87 (16.29)	5.74 (34.97)	02:24	100.00	EDR
34	P2	42.37	40.54	30.65	-60.94	-	-	ED
	P3	15.65	14.47	4.88	2.85	-	-	EDR
	P4	36.87 (8.57)	33.53 (10.66)	28.55 (13.79)	22.00 (32.18)	01:51	100.00	ED
	P5	2.03 (11.06)	-0.52 (13.15)	-0.95 (16.29)	-0.29 (34.97)	02:24	100.00	EDR
35	P2	-22.78	-22.94	-34.05	-70.98	-	-	ED
	P3	-168.91	-173.77	-181.68	-155.21	-	-	EDR
	P4	22.05 (5.84)	19.09 (7.93)	14.11 (11.06)	10.05 (29.79)	01:12	100.00	ED
	P5	6.49 (11.06)	4.66 (13.15)	2.80 (16.29)	2.92 (34.97)	02:24	100.00	EDR
36	P2	-20.07	-111.06	-111.87	-152.54	-	-	ED
	P3	21.89	21.81	21.47	19.20	-	-	EDR
	P4	0.00(100.00)	0.00(100.00)	0.00(100.00)	0.00(100.00)	-	0.00	ED
	P5	0.00(100.00)	0.00(100.00)	0.00(100.00)	0.00(100.00)	-	0.00	EDR
37	P2	37.14	21.62	23.98	23.27	-	-	ED
	P3	4.97	4.97	1.59	0.68	-	-	EDR
	P4	1.91 (41.92)	-0.17 (43.80)	-0.08 (46.61)	0.08 (63.58)	08:24	90.48	ED
	P5	1.02 (13.77)	0.45 (15.81)	0.06 (18.87)	0.59 (37.05)	02:37	97.96	EDR

Table 8.5. $diffMAPE$ results for the routes 38-50 and for different prediction horizons (Δ). The values between parentheses correspond to $H_{\Delta,R}$ values. Finally, the last column identifies the distance measure used in each case

R		Δ=15min	Δ=45min	Δ=90min	Δ=6h	E_R	M_R	d
38	P2	25.52	24.47	-13.30	-57.20	-	-	ED
	P3	-3.66	-17.25	-59.49	-32.60	-	-	EDR
	P4	41.85 (29.23)	37.40 (31.24)	25.47 (34.27)	-1.24 (52.45)	06:20	97.28	ED
	P5	28.92 (16.08)	28.92 (18.17)	28.84 (21.28)	0.50 (39.95)	03:36	100.00	EDR
39	P2	-50.86	-113.66	-135.60	-93.97	-	-	ED
	P3	-0.46	0.34	-1.75	14.58	-	-	EDR
	P4	-0.12 (99.20)	-0.12 (99.23)	-0.12 (99.27)	-0.09 (99.52)	09:36	1.36	ED
	P5	-2.98 (11.61)	-2.89 (13.70)	-2.40 (16.83)	-1.05 (35.42)	02:31	100.00	EDR
40	P2	22.48	17.77	-35.56	-9.25	-	-	ED
	P3	2.19	2.02	0.56	23.11	-	-	EDR
	P4	0.01 (48.86)	0.01 (50.68)	0.09 (53.41)	0.00 (69.76)	09:49	87.76	ED
	P5	0.00 (11.06)	-0.39 (13.15)	-0.53 (16.29)	0.21 (34.97)	02:24	100.00	EDR
41	P2	53.86	50.67	56.58	45.16	-	-	ED
	P3	1.02	0.96	1.25	1.80	-	-	EDR
	P4	0.02 (52.88)	0.06 (54.53)	0.08 (57.00)	0.12 (71.96)	09:36	79.59	ED
	P5	0.46 (81.97)	0.37 (83.95)	0.22 (86.91)	0.00(100.00)	19:14	94.56	EDR
42	P2	47.97	47.43	48.59	34.31	-	-	ED
	P3	-3.85	-8.22	-14.72	-43.75	-	-	EDR
	P4	0.51 (50.86)	0.40 (52.58)	0.19 (55.16)	0.01 (70.76)	09:36	82.99	ED
	P5	2.44 (5.84)	2.27 (7.93)	1.94 (11.06)	1.91 (29.79)	01:12	100.00	EDR
43	P2	10.91	-4.22	-8.87	-6.32	-	-	ED
	P3	-9.75	-5.44	-15.61	-13.01	-	-	F
	P4	0.05 (47.64)	0.02 (49.47)	0.00 (52.22)	0.00 (68.84)	09:36	88.44	ED
	P5	0.02 (94.36)	0.02 (94.56)	0.02 (94.86)	0.03 (96.64)	09:36	9.52	F
44	P2	-252.35	-265.77	-433.70	-120.95	-	-	ED
	P3	7.67	-24.11	-24.08	-21.83	-	-	EDR
	P4	-0.16 (70.81)	-0.08 (72.89)	-0.09 (76.02)	-0.23 (94.78)	16:48	100.00	ED
	P5	-0.24 (97.38)	-0.23 (97.47)	-0.15 (97.59)	-0.15 (98.36)	08:24	4.08	EDR
45	P2	-248.17	-261.38	-457.13	-156.93	-	-	ED
	P3	15.76	11.81	-8.49	-11.23	-	-	EDR
	P4	10.49 (36.89)	2.55 (38.97)	-0.59 (42.08)	-0.39 (60.83)	08:39	100.00	ED
	P5	-0.10 (91.13)	-0.02 (91.44)	-0.02 (91.91)	-0.04 (94.72)	09:36	14.97	EDR
46	P2	25.61	24.99	27.53	2.36	-	-	ED
	P3	9.95	4.17	-16.47	-12.53	-	-	EDR
	P4	-0.17 (71.59)	-0.03 (73.50)	-0.06 (76.38)	0.14 (93.60)	16:22	91.84	ED
	P5	-2.36 (41.21)	-1.70 (43.27)	-0.78 (46.36)	-0.88 (64.83)	09:31	98.64	EDR
47	P2	19.33	13.24	-16.41	-35.62	-	-	ED
	P3	13.89	-6.14	-23.70	-10.67	-	-	DTW
	P4	-1.00 (53.47)	-1.61 (55.40)	-1.62 (58.30)	-1.67 (75.78)	11:49	93.20	ED
	P5	0.00(100.00)	0.00(100.00)	0.00(100.00)	0.00(100.00)	-	0.00	DTW
48	P2	17.73	15.14	-5.75	-33.99	-	-	ED
	P3	2.71	-5.36	-28.62	-22.50	-	-	EDR
	P4	0.34 (49.83)	-0.10 (51.67)	-0.15 (54.44)	0.04 (71.19)	10:17	89.12	ED
	P5	-0.16 (81.44)	-0.15 (82.09)	-0.14 (83.07)	-0.08 (88.96)	09:36	31.29	EDR
49	P2	10.65	6.67	1.38	5.87	-	-	ED
	P3	9.09	9.09	9.08	7.71	-	-	EDR
	P4	-0.59 (49.72)	-0.03 (51.49)	0.11 (54.13)	0.56 (70.10)	09:37	85.03	ED
	P5	0.29 (96.82)	0.11 (98.41)	0.00(100.00)	0.00(100.00)	22:48	76.19	EDR
50	P2	32.71	25.11	6.60	-14.64	-	-	ED
	P3	3.31	3.31	2.04	1.78	-	-	EDR
	P4	-0.58 (42.69)	-1.49 (44.76)	-1.39 (47.88)	-0.91 (66.66)	10:03	100.00	ED
	P5	0.52 (91.94)	0.43 (93.79)	0.28 (96.54)	0.00(100.00)	21:37	88.44	EDR

Table 8.6. $diffMAPE$ results for the routes 51-61 and for different prediction horizons (Δ). The values between parentheses correspond to $H_{\Delta,R}$ values. Finally, the last column identifies the distance measure used in each case

R		Δ=15min	Δ=45min	Δ=90min	Δ=6h	E_R	M_R	d
51	P2	25.90	18.86	1.67	-17.04	-	-	ED
	P3	15.36	-17.54	-16.31	-27.35	-	-	EDR
	P4	2.08 (52.24)	0.70 (54.24)	0.52 (57.24)	-0.15 (74.28)	11:56	96.60	ED
	P5	-7.95 (72.04)	-8.11 (73.20)	-5.35 (74.95)	-0.67 (83.32)	11:46	55.78	EDR
52	P2	30.55	10.90	4.56	-50.96	-	-	ED
	P3	-27.08	-89.97	-114.94	-188.13	-	-	F
	P4	-0.16 (61.71)	-0.30 (63.06)	-0.37 (65.08)	-0.53 (77.23)	09:36	64.63	ED
	P5	0.80 (63.15)	0.35 (64.50)	0.24 (66.51)	-0.10 (78.53)	10:07	64.63	F
53	P2	30.33	-1.32	6.45	-10.14	-	-	ED
	P3	-37.53	-62.29	-112.06	-149.67	-	-	F
	P4	-0.33 (79.25)	-0.30 (81.32)	-0.30 (84.40)	-0.01 (99.93)	18:47	99.32	ED
	P5	-0.48 (68.25)	-0.95 (69.42)	-0.94 (71.17)	-0.36 (81.62)	10:09	55.78	F
54	P2	25.54	25.29	12.52	1.04	-	-	ED
	P3	25.49	25.23	12.53	0.97	-	-	F
	P4	1.00 (42.02)	0.40 (44.06)	0.37 (47.10)	0.37 (65.51)	09:36	97.96	ED
	P5	1.01 (43.10)	0.41 (45.11)	0.38 (48.11)	0.38 (66.19)	09:40	96.60	F
55	P2	12.33	12.46	1.09	-6.97	-	-	ED
	P3	27.67	26.28	21.10	10.47	-	-	DTW
	P4	6.84 (77.33)	3.02 (78.05)	-0.97 (79.14)	-1.74 (85.75)	08:24	35.37	ED
	P5	0.00(100.00)	0.00(100.00)	0.00(100.00)	0.00(100.00)	-	0.00	DTW
56	P2	1.49	1.42	1.46	1.31	-	-	ED
	P3	46.78	45.61	42.96	31.61	-	-	DTW
	P4	-0.05 (95.17)	0.00 (95.69)	0.03 (96.48)	0.00(100.00)	19:12	25.17	ED
	P5	-0.22 (65.81)	-0.21 (66.82)	-0.19 (68.33)	-0.09 (77.41)	06:49	48.30	DTW
58	P2	-7.13	-19.34	-21.75	-16.17	-	-	ED
	P3	-0.40	4.00	-7.29	-4.22	-	-	DTW
	P4	-0.18 (88.16)	-0.05 (89.50)	-0.01 (91.51)	0.00(100.00)	19:24	64.63	ED
	P5	-0.20 (87.04)	-0.06 (88.64)	-0.01 (91.01)	0.00(100.00)	19:45	76.87	DTW
59	P2	10.92	9.89	9.77	9.14	-	-	ED
	P3	4.42	4.35	4.25	1.97	-	-	EDR
	P4	0.01 (96.77)	0.00 (98.23)	-0.01 (99.71)	0.00(100.00)	22:41	70.07	ED
	P5	2.01 (66.73)	1.76 (68.41)	1.47 (70.91)	0.55 (85.36)	13:51	80.27	EDR
60	P2	8.56	0.20	1.56	10.46	-	-	ED
	P3	-11.67	-15.49	-19.25	-56.08	-	-	DTW
	P4	-0.02 (84.68)	-0.02 (86.35)	-0.02 (88.84)	0.00(100.00)	19:13	80.27	ED
	P5	0.01 (83.96)	0.01 (85.70)	0.01 (88.30)	0.00(100.00)	19:12	83.67	DTW
61	P2	-27.31	-48.78	-44.97	-39.81	-	-	ED
	P3	4.55	4.31	4.20	1.64	-	-	EDR
	P4	5.29 (28.05)	4.77 (29.74)	4.35 (32.28)	3.43 (47.38)	02:24	80.95	ED
	P5	2.55 (64.91)	2.40 (66.79)	2.22 (69.61)	1.54 (85.79)	14:29	90.48	EDR

Now, taking these general conclusions into account, and by means of some plots, we analyze two aspects in more detail: the effect of using the ADMS vs. ED, and the impact of using ECDIRE vs. 1NN.

8.2.3.1 The effect of using ADMS vs. ED

Based on the results of Tables 8.2, 8.3, 8.4, 8.5 and 8.6, it does not seem that applying the ADMS is very beneficial. However, in order to see if there are any general trends, we compare the results obtained by P2 and P3 (see Figure 8.4), and P4 and P5 (see Figure 8.5) considering all the routes together

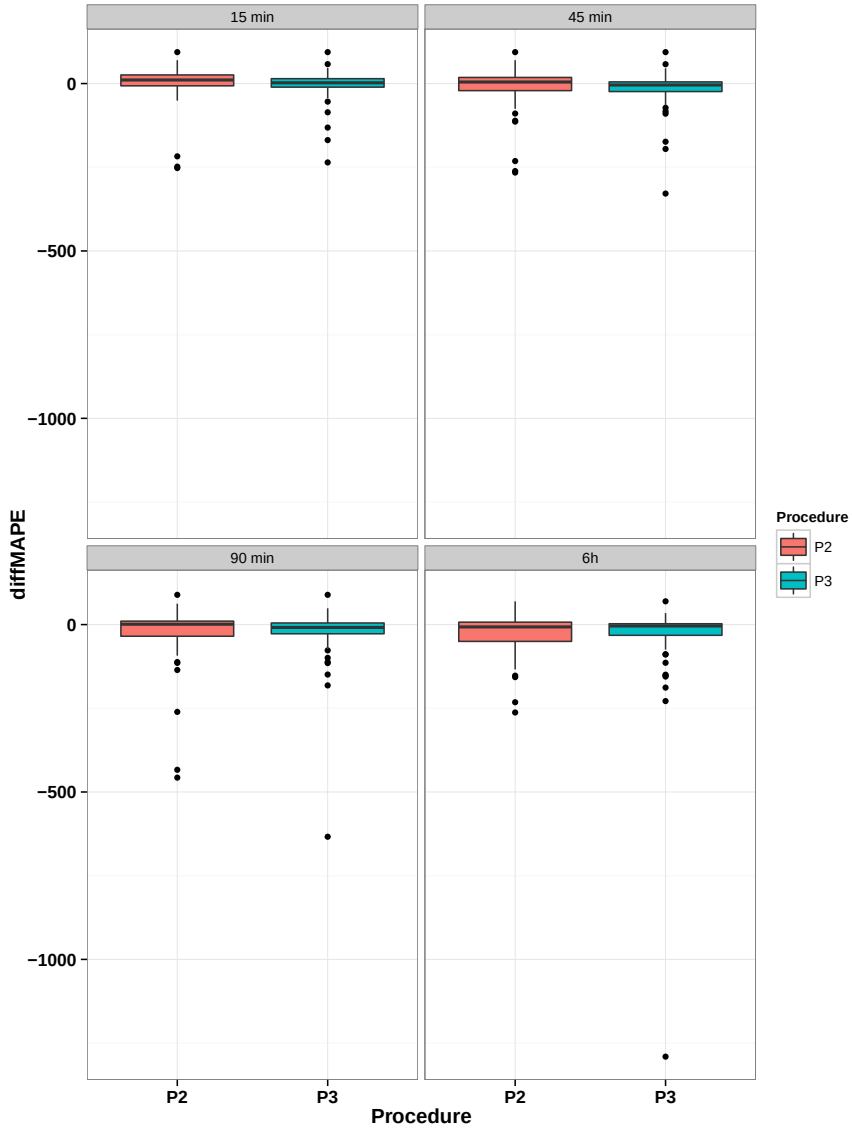


Fig. 8.4. Differences between using ED or applying the ADMS to select the distance measure when the early classification is performed using 1NN.

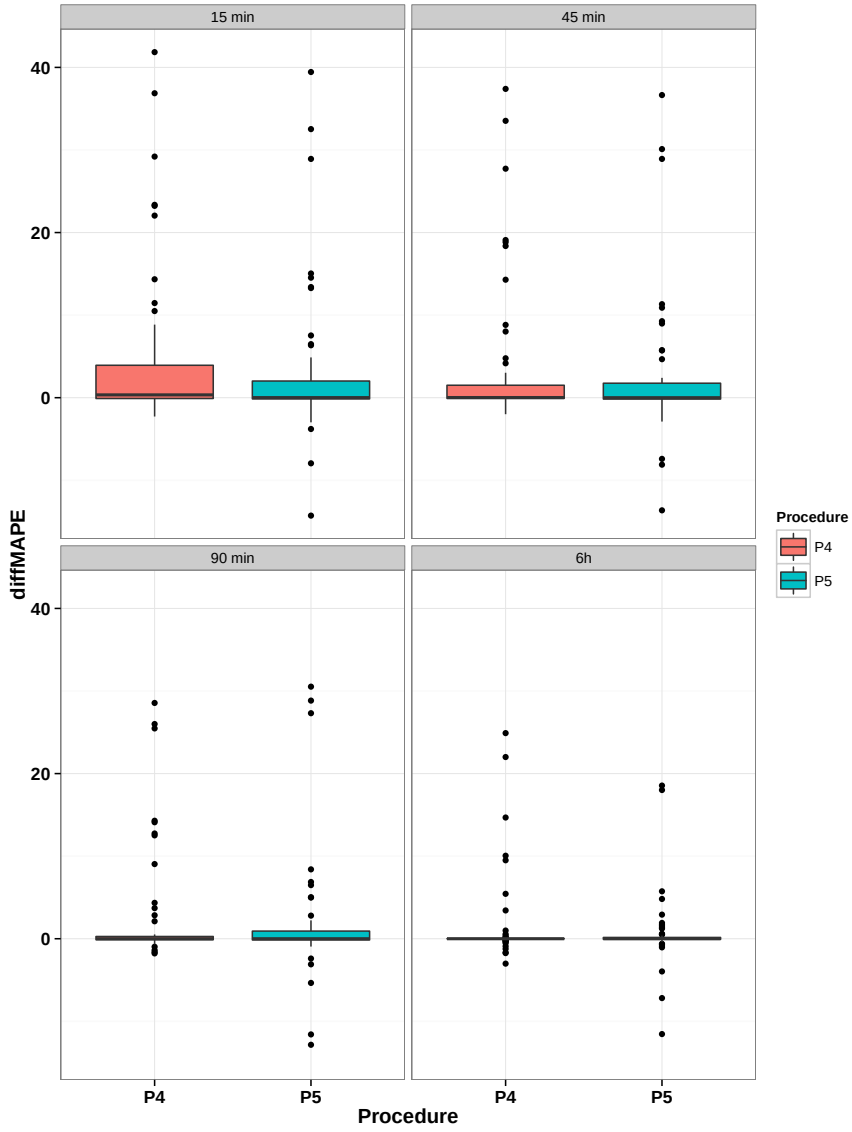


Fig. 8.5. Differences between using ED or applying the ADMS to select the distance measure when the early classification is performed using ECDIRE.

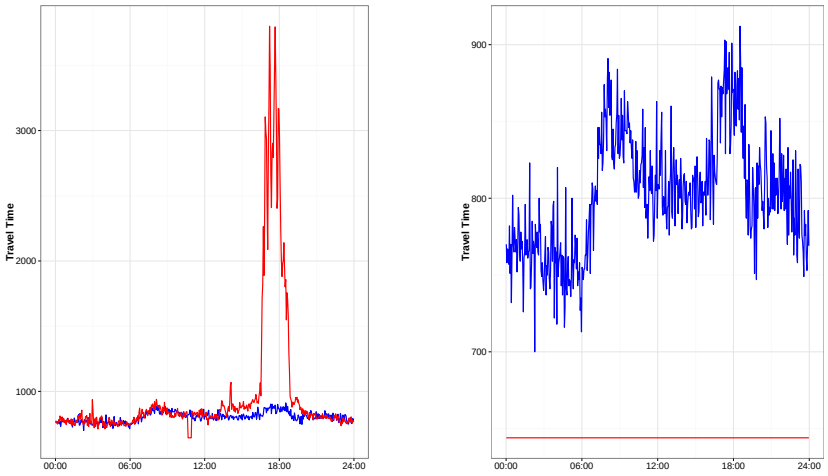
and plotting the distribution of the $diffMAPE$ values using box-plots. With these box-plots we confirm that, indeed, there is no evidence that the distance measures selected by the ADMS have a better performance than ED overall. The results depend strongly on the route at hand, and the specific clustering result obtained in each case.

Now, in order to show some of the typical behaviors that appear in the different routes, we analyze the results obtained for three example routes in more detail. These specific routes have been chosen because they show very different behaviors and are illustrative examples of the patterns that appear. In the first example, the results for the distance selected by the ADMS are better than those obtained by ED. In the second example, ED obtains better results than the distance measure selected by ADMS. Finally, in the third case, neither of the distance measures obtains good results, and using P1 is better than using any of the four non-naive procedures.

In Route 1 and when using the 1NN as an early predictor (see result for P2 in Table 8.2), the ED has a very good performance for the first prediction horizons, but subsequently, the $diffMAPE$ values deteriorate rapidly, becoming negative in the last prediction horizons. When using ECDIRE with ED the results are always negative (see result for P4 in Table 8.2). Contrarily, the EDR distance measure does not obtain such good results at the beginning but does not deteriorate as much either (see results for P3 and P5 in Table 8.2).

To better understand why this happens, in Figure 8.6 we compare the cluster medoids obtained in the clustering process when using the distance measure selected by the ADMS (EDR) and the ED. Each line represents a cluster medoid, which consists of the travel time measurements collected over an entire day. Note that the two clusters identified by ED can not be distinguished until more than half of the sequence is available. Then, one of the patterns changes very drastically. In this context, when the prediction horizons are small and very recent measurements are available, the 1NN method is able to detect the peak of traffic without much delay. However, as the prediction horizon increases, the identification of the clusters obtained by ED becomes impossible. This is reflected, for example, in the fact that the ECDIRE method with ED issues no class predictions in more than 90% of the cases (see $H_{\Delta,R}$ values in Table 8.2) and also in the mean earliness value, which ascends to 19:12 in this case. Contrarily, EDR separates two clusters that are very distinct from each other (see Figure 8.6b) and which can be discriminated earlier in time (mean earliness value is 15:08).

This behavior is repeated in other routes such as 14, 22, 24, 29 or 36. Threshold based distances such as EDR or TQuest tend to isolate very rare patterns into separate clusters. These patterns do not usually appear again in the testing days, and are therefore not useful to obtain good travel time predictions. Applying the EDR distance is a way of identifying and separating these series automatically. Furthermore, when these patterns are very different from the rest of the patterns, as in the case of Route 1, they are usually never

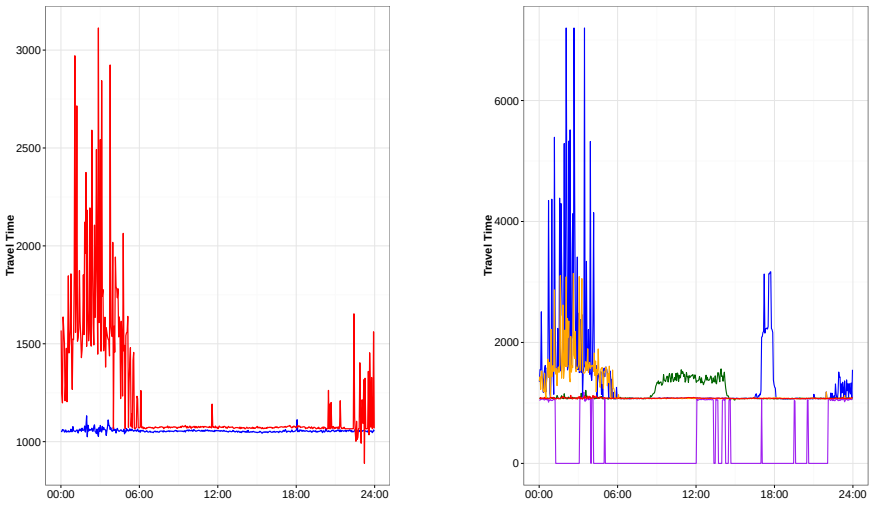


(a) ED. (b) ADMS (EDR).

Fig. 8.6. Cluster representatives for the ED and the EDR for Route 1.

selected by the early classifiers, and are thus discarded from the travel time prediction phase in many cases, resulting in lower errors.

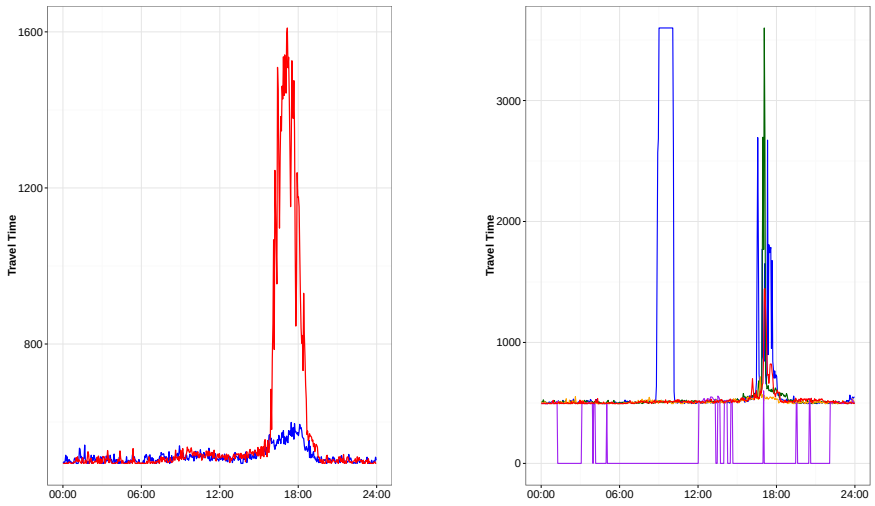
However, if the isolated patterns are not very different from each other, the behavior can be completely opposite. As a second example we analyze the results obtained in Route 28 (see first four rows of Table 8.4). In this case, ED obtains better results than the distance measure selected by ADMS (EDR). Indeed, if we use EDR in this route, the results are worse than applying the basic historical predictor. In this case, the two patterns obtained by ED are easily identifiable from the beginning of the sequence (see Figure 8.7a). EDR, on the contrary, identifies 5 clusters, some of them very extreme and rare, but which can be easily confused with the others at the beginning of the day (see Figure 8.7b). In this case, if we apply the 1NN method with the EDR distance, the cluster assignments become very volatile and random, resulting in very bad travel time predictions (see results for P2 in Table 8.4). The ECDIRE method, as will be explained later, slightly mitigates this by controlling the reliability of the predictions and initially discarding the very rare patterns (see results for P5 in Table 8.4). Similar behaviors can be seen for example in Routes 6, 7, 12, 13 or 23.



(a) ED. (b) ADMS (EDR).

Fig. 8.7. Cluster representatives for the ED and the EDR for Route 28.

Finally, in some cases, the clustering pre-process is not beneficial with either ED or the distance measure selected by the ADMS. Observe for example the results and cluster medoids obtained for Route 9 in Table 8.2 and Figure 8.8, respectively. The $diffMAPE$ values are always negative for all non-naive procedures. With ED, the only difference between the clusters is the size of the peak. In this sense, the clusters can not be identified until very late in time (the E_R value for P4 on this route is 19:23) , and even then, in practice it is difficult to discern between them, because the border between the clusters is not very clear, and in many occasions no reliable predictions can be made (note that M_R=71.43% for P4, so the number of unclassified days in the testing set is almost 30%). With DTW, there are five clusters. One of them (the purple line) is a rare pattern which can be identified quite easily. However, the rest of the patterns are very similar to each other during a large part of the day, and can thus be confused easily. Since some of these patters are quite extreme, making errors in the clusters assignations results in large prediction errors. In both these cases, it it better to simply use the basic historical predictor.



(a) ED. (b) ADMS (DTW).

Fig. 8.8. Cluster representatives for the ED and the DTW for Route 9.

In essence, with the observed results it is difficult to make conclusive statements about the goodness of using the ADMS, because the behavior varies a lot from route to route. However, there are several factors that we would like to underline:

- It is noticeable that, when the prediction horizon is very long, time series clustering ceases to be beneficial, because the patterns change very abruptly and it is risky to lean towards specific patterns.
- The objective of the ADMS is to select the most suitable distance measure for clustering a time series database. However, in this case, the final objective is not clustering, but prediction. In this context, we wonder if some clustering solutions are more suitable than others for the task of forecasting. If this were so, it could be interesting to train the ADMS taking this final objective (forecasting) into account.
- The ADMS does not only provide one distance measure, but a set of the most suitable distances. In this case, we have only selected one of them by using the silhouette and Dunn indices as explained in Section 8.2.2. Furthermore, the most adequate parameter and the number of clusters for the selected distance measure has also been chosen using these indices. However, it is questionable whether this method is adequate, especially if we are comparing clustering results obtained from different distance measures.

- Recall that the automatic distance selector has been trained using the synthetic datasets and the datasets from the UCR. ED is not frequently part of the most suitable distance set of these time series datasets, and consequently it is never predicted in this case (see column d of Table 8.2). However, it seems that in this case it is indeed a suitable distance measure on more than one occasion. As such, it may be necessary to train the automatic distance measure selector with databases which are more similar to these from Kaggle.

8.2.3.2 The effect of using 1NN vs. ECDIRE

In this section, we analyze the performance of the two early classifiers considered in more detail: ECDIRE and 1NN. If we compare the performances of P3 and P4 (see Figures 8.9), and P3 and P5 (see Figure 8.10)) using box-plots to find any overall trends as in the previous section, we confirm that, overall, ECDIRE obtains a slight improvement over the baseline in the first prediction horizons and, as the prediction horizon increases, the results tend towards the baseline. On the contrary, in the case of 1NN, the predictions are quite good in the first prediction horizons but, as the prediction horizon increases, the results tend to deteriorate, rapidly becoming worse than the baseline. Also, in the case of 1NN, there are quite a few very extreme negative values in all prediction horizons. These are represented as outlier points in the box-plots and correspond to some routes (i.e. 21, 23, 44, 45) in which the predictions obtained by P2 and P3 are always much worse than the baseline.

If we analyze the results and the data more in depth, we identify three different causes which explain these results. We explain these three causes in detail in the following paragraphs and, additionally, we also provide a graphical and illustrative example for each of them.

To begin with, in many routes the series in different clusters have a very similar and constant trend in a substantial part of the series. Then, in the afternoon, one or more patterns change drastically and unpredictably. In this context, the cluster assignments become more and more difficult as the prediction horizon increases because no recent data is available. When t and $t+\Delta$ belong to the same day, 1NN always issues a class prediction, regardless of the quality (see Figure 8.2), so on many occasions it makes mistakes, resulting in very bad travel time predictions. On the contrary, ECDIRE analyzes the classes and only provides class predictions when it is presumably safe to do so. In this sense, it is much more conservative than 1NN and the results are closer to 0. If we observe the $H_{\Delta,R}$ values in Table 8.2 we can see that they are generally very high so, for many travel time predictions, the whole training set is used and the model does not lean towards any specific model. This is influenced by the fact that in many cases the class predictions are not issued by ECDIRE until late afternoon, sometimes, after peak traffic is over (see E_R values in Table 8.2).

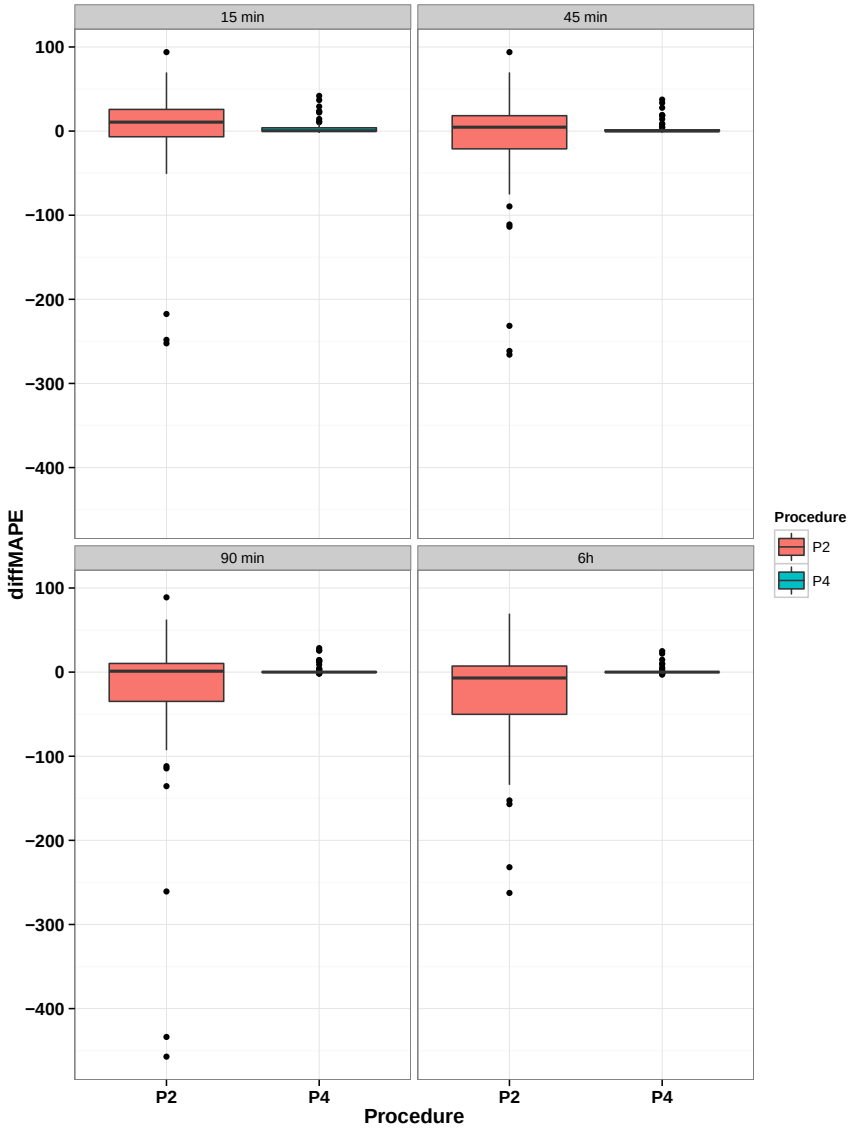


Fig. 8.9. Differences between using 1NN or ECDIRE as an early classifier, when the distance measure used is ED.

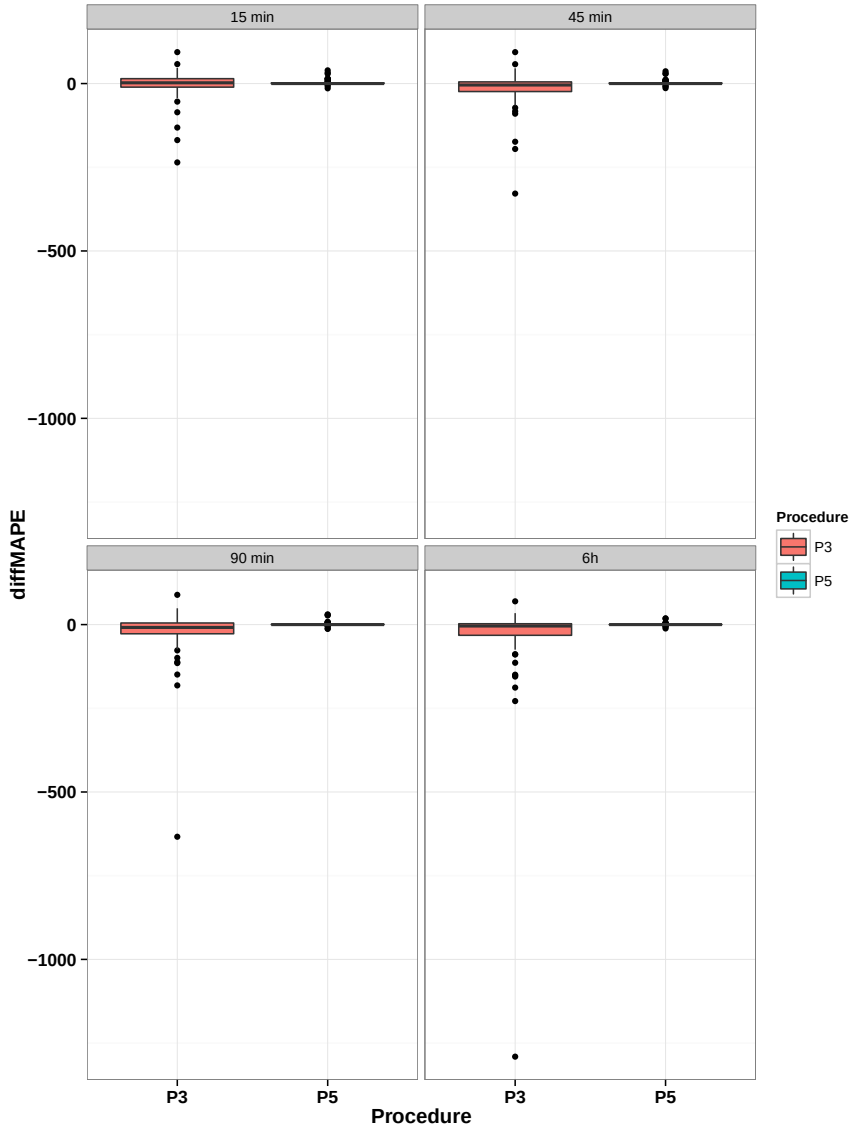


Fig. 8.10. Differences between using 1NN or ECDIRE as an early classifier, when the distance measure used is selected by the ADMS.

A clear example of this can be seen in Route 15. When using the ED, three patterns are identified which are identical until almost 18:00. Then two of the patterns (the blue and red lines) change suddenly and drastically. In this case, as the prediction horizon increases, the predictions for 1NN become worse and worse, because the classes can not be discriminated reliably from each other and the classifications of the 1NN method become almost random (see results for P2 in Table 8.3). On the contrary, ECDIRE is much more conservative (see results for P4 in Table 8.3). The mean earliness value is 18:00, which indicates that the series are not classified until late in the afternoon. Furthermore, many series are never classified due to reliability conditions (M_R value for P4 is 76.19%). All this results in very high $H_{\Delta,R}$ values, that tend to 100% as the prediction horizon increases and so, when the prediction horizon becomes large enough, P4 becomes identical to P1.

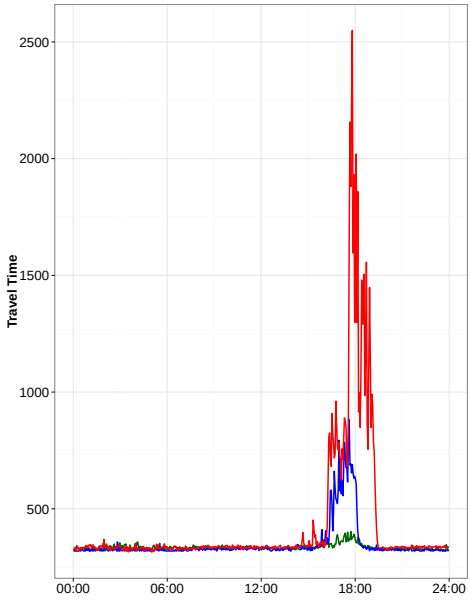


Fig. 8.11. Cluster medoids obtained with the ED for route 15.

A second factor to take into account is that, as commented previously, sometimes, the clustering process tends to isolate some series with very specific characteristics (very extreme patterns, erroneous values, noisy patterns) into separate clusters. These patterns usually appear in very few series and are not repeated in the testing set. However, if they are similar to the other patterns in part of the sequence, they can cause large errors if we use the 1NN naive early classification method. The ECDIRE method tends to discard these patterns because of their unreliability, so the errors do not become very large.

Route 23 illustrates this last situation. As can be seen in Figure 8.12, the ED identifies five different patterns. Clusters 1 and 2 (blue and red lines, respectively), which have the least extreme values, take almost all the training set, although cluster 1 is larger. On the contrary, classes 3, 4 and 5 only contain one series (the representative itself). However, note that all the clusters are very similar in more than the first half of the series, except that represented by the purple line. In this context, the cluster assignments with 1NN are carried out almost randomly. Since mistakenly assigning one of the 3 extreme clusters to a series results in very large MAPE errors, the behavior of P2 becomes very bad and degenerates even more as the prediction horizon increases (see Table 8.3). On the contrary, the ECDIRE method, initially discards the prediction of classes 3, 4 and 5 because of lack of reliability. With respect to classes 1 and 2, based on the timeline built and the reliability condition, the class assignments are carried out much later in time: the mean earliness for P4 is 13:22 and some series are not predicted until 95% of the series is already collected. All this results in a more conservative prediction that, in cases like this, obtains better results (see results for P4 in Table 8.3). Similar behaviors can also be seen in routes 6, 7, and 8, among others

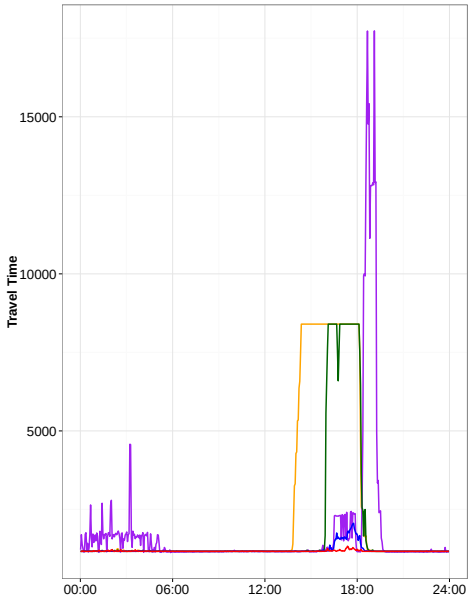


Fig. 8.12. Cluster medoids obtained with the ED for route 23.

Finally, as commented previously, in some cases, the clustering result is not helpful at all. In these cases both P2 and P3 (the procedures based on 1NN) usually obtain very bad results. However, ECDIRE, based on reliability

requirements, is able to detect this and barely provides any class predictions. In this sense P4 and P5 become almost identical (or even identical in some cases) to P1 and the $diffMAPE$ results tend to 0.

This can be seen in Route 8, for example, where the $diffMAPE$ results for P3 are always negative and very large. However, the values for P5 are always 0. Note that in this case no early classification is applied at all in P5: the $H_{\Delta,R}$ values, are 100% in all cases (see Table 8.2). If we observe the cluster medoids for Route 8 and distance DTW (see Figure 8.13) we observe that they are almost identical in shape, but one has a higher peak than the other. Cluster 1 contains series with a smaller peak, and Cluster 2 contains series with a larger peak. However, in practice, the border between these two clusters is not clear at all, which makes it very difficult or even impossible to discern between them. In view of this, ECDIRE opts not to lean towards any of the patterns and P5 becomes identical to P1.

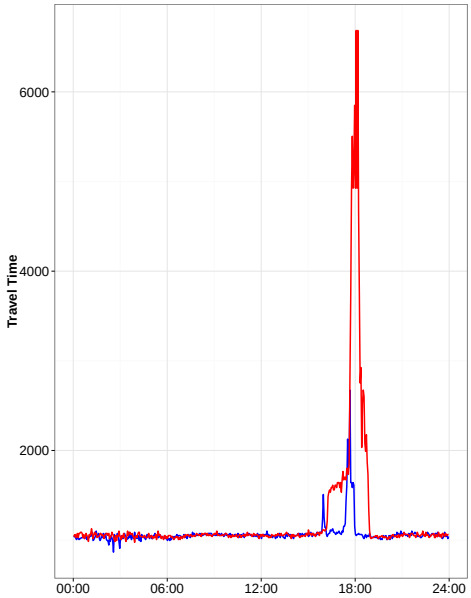


Fig. 8.13. Cluster medoids obtained with the DTW distance for route 8.

To conclude, although the ECDIRE method shows some benefits in comparison to 1NN, it also has some drawbacks:

- The ECDIRE method still slightly tends towards larger classes, especially if the patterns are similar at the beginning. The naive method we have proposed to mitigate this trend is certainly not optimal and, furthermore, it is not adequate when certain classes always yield very low accuracy clas-

sification results, even when the whole sequence is available. In this sense, specific and more adequate methods to deal with unbalanced datasets should be analyzed.

- The ECDIRE method is sometimes too conservative and makes the class assignments very late in time, sometimes too late to obtain helpful information for the task of travel time prediction, because peak traffic is already over. In this sense, it would be better if it could adapt more rapidly to the incoming patterns. A possible solution could consist in lowering the reliability conditions and giving more importance to the earliness of the predictions than to their accuracy. But then, we could encounter the same problems that arise with 1NN.

The analysis of all these results and the identification of these different behaviors gives way to many future research directions, mainly directed to adapting the proposals of Chapters 6 and 7 to this specific problem and this particular database. These are specified in the following section.

8.3 Conclusions and future work

In this chapter we have applied the contributions shown in Chapters 6 and 7 to the problem of travel time prediction. We have analyzed their performance in this particular environment and we have discovered specific behaviors and characteristics of the data that can lead to the design of some adaptations in the proposed methods.

To begin with, we must emphasize the importance of the pre-process of the training data. Since the travel time patterns within a specific road can change permanently due to roadworks, new lanes, speed modifications, restrictions, etc. we think it is necessary to build a representative training set which contains patterns which are similar to the new incoming testing days before applying any travel time prediction model.

Also, we have seen that even if time series clustering is helpful in some cases, for very long term prediction horizons it is not beneficial. In this context, it could be useful to find out exactly when this procedure ceases to be useful.

Next, we must question the validity of the silhouette and Dunn indices for the specific task of distance measure and parameter selection within a forecasting problem. Particularly, we should analyze if these indices allow a fair comparison when the clustering results are obtained with different distance measures and for the purpose of travel time forecasting. If not, we could try to propose new measures of evaluation which take into account that different distance measures are to be compared and which consider that the final objective is forecasting.

Also in this sense, the training of the ADMS does not consider that the final objective is travel time forecasting. As such, we could consider training this

multi-label classifier directly with the traffic databases, using the predictive power as the evaluation measure for the clustering results.

With respect to the ECDIRE early classifier, we have seen that one of the principal problems that arises when it is applied to traffic data is the presence of very unbalanced datasets. As such, an obvious research direction consists in trying to analyze and include specific methods, such as those reviewed in [84], to deal with this kind of datasets in our framework.

The optimization of several parameters of ECDIRE could also be another future line of study. Particularly, depending on the characteristics of the database, the threshold that controls the reliability test could be adapted.

Finally, in this chapter we have only included the historical models as travel time predictors. However, the introduction of other models and the combination of different types of predictors for different traffic patterns could maybe improve the results.

Part IV

Conclusions and future work

9

Conclusions and future work

In this last chapter we summarize the main conclusions obtained from all the contributions that appear in this dissertation. Additionally, we also enumerate some general future research directions that have emerged from the work presented. These directions, along with more detailed and specific conclusions are included in each chapter separately. Finally, in the last part of this section, the publications that have resulted from this dissertation are shown.

9.1 Conclusions

This dissertation has been devoted to working on several time series data mining problems: the selection of a suitable distance measure for clustering time series databases and the problem of early classification of time series. However, the starting point to or inspiration for these problems has been the problem of travel time modeling.

As such, as a first contribution, a complete review of the literature on travel time modeling has been provided, focusing on the problems of estimation and prediction. Although these two tasks are frequently confused in the literature, we have shown in Chapter 2 that they are separate concepts that require different modeling strategies. Estimation models largely depend on the type of data or sensors available on the trajectory under study. Contrarily, prediction models differ mainly in the type of strategy used to build the model. Based on this, a separate and detailed taxonomy of the existing approaches for travel time prediction and estimation has been provided in Chapter 4.

From this literature review and taxonomy, the main conclusion that we have extracted is that most of the models published in the literature are not well adapted to the practical requirements of ATIS. To begin with, most of them are only applied over small road sections and there are not many publications that extend their methodologies to entire road networks. Furthermore, the type of data used in the proposed models is quite restrictive: generally only one type of traffic sensor is used and contextual information such as weather

is almost never considered. Also, not many effective solutions are provided to deal with non-recurrent congestion, a crucial issue in travel time modeling. Finally, the great majority of proposals focus on providing punctual travel time estimations and dismiss the fact that confidence intervals or the like allow the users to assess the goodness of the predictions.

In this context, combined or hybrid models seem to be the most promising future research direction, due to their adaptability, their capacity to combine different sources of data and their ability to represent diverse traffic patterns adequately. If we focus on prediction, a specific type of combined models are those which, in order to identify different traffic patterns or behaviors, apply clustering pre-processes to the data and then apply a (distinct) travel time model to each pattern. Particularly, a not very common approach in the travel time prediction literature is applying the paradigm of time series clustering, which has some specific characteristics, such as the need for special distance measures for time series. In time series clustering, instead of clustering regular data points, the objective is to group entire sequences, for example, measurements of travel time taken throughout an entire day.

Since the presence of different daily traffic patterns is unquestionable in most study sites, this method seems to be a reasonable approach. However, the real benefits of time series clustering in travel time prediction have not yet been demonstrated in the literature. As such, in Chapter 5, we have performed some preliminary experiments, using some basic distance measures, that prove that, indeed, clustering can be beneficial for travel time prediction purposes.

Nevertheless, even if applying time series clustering pre-processes does sometimes yield improved travel time predictions, it is very important to underline that the specific distance measure chosen to quantify the similarity between the time series is a crucial aspect of the process, and thus has a deep impact on the obtained results. Due to the increasing popularity of time series data mining tasks, in the past few years, a vast quantity of time series distance measures have been published. Also, it has been demonstrated in the literature that there is not one best distance measure that is adequate for all time series databases. In fact, the performance of the different existing time series distance measures depends strongly on the specific characteristics of each database. In this context, the selection of the most suitable distance measure for clustering a time series database is not a trivial issue.

This is exactly what we have focused on in the second main contribution of this dissertation. In Chapter 6, we propose a multi-label classifier which takes a set of characteristics of the time series database as input, and outputs the most suitable distance measures(s) from a set of candidates. The experiments performed have shown that this approach is useful to simplify the distance selection process in a time series database clustering task.

Finally, recall that when using the combined travel time model which we have considered initially, a second problem that we encountered was that it is necessary to assign the new incoming sensor measurements to a given cluster, this is in order to find out which model we must use in the given case. This

assignment must be done online, with the data available at the moment and as early as possible.

This is exactly where the problem of early classification comes into play. Early classification is a supervised learning task, where the objective is to predict the class labels of time series as soon as possible, before the whole time series is available, but respecting a pre-defined level of accuracy. In Chapter 7, the third important contribution of this dissertation is presented: we provide a method for early classification of time series based on probabilistic classifiers that improves the results obtained by previous methods of the state-of-the-art by dealing with some of the flaws of previously published techniques.

Finally, after presenting these two contributions on time series data mining, in Chapter 8, we have returned to the problem of travel time prediction and have analyzed the performance and usefulness of these two proposals within this specific problem. From this last work, we extract some conclusions and many future research lines, mostly directed at adapting the automatic distance measure selector and the proposed early classifier to the particular scenario and type of data issued from the problem of travel time prediction.

To sum up, in this dissertation we have departed from an analysis of the problem of travel time modeling which has led the two main methodological contributions to time series data mining: a method to automatically select the most suitable distance measure/s to cluster a database, and an early classifier based on probabilistic classifiers.

9.2 Future work

The contributions presented in this dissertation have led to many new research directions, which we can divide into two groups: methodological research proposals and applications or solutions to real-world problems.

Regarding the methodological research proposals we underline the following points:

- As we have shown in Chapter 3, in the past few years, the research community has focused on creating new distance measures for time series. Still, most of these are useful only for univariate sequences. In this context, the creation of similarity measures for multi-variate time series is an obvious future research direction. Specifically, the use of meta-distances or combinations of distances could be an interesting path to follow in this sense.
- Also, in Chapter 3, we have seen that model-based time series distance measures fit a model to each time series and then calculate the distance between the models (usually, by computing the Euclidean distance between its coefficients). Most model-based distances proposed in the literature are based on simple parametric models, such as ARMA models and it is an open question to find out if distances based on more complex models, such

as Gaussian processes, could be created, and if these could be useful in practice.

- In Chapter 6, we saw that many of the time series distance measures available in the literature require the selection of one or more parameters. Analyzing the effect of these parameters on the performance of the distance measures may be helpful to simplify the distance measure selection process even more. Indeed, we consider the possibility of finding a relation between the parameters of a distance measure and certain characteristics of the database. As an example, the level of shift available in a database could be used to determine the window size used in the DTW distance, or the correlation level present in the database could be decisive when choosing an ϵ value for the EDR distance.
- In Chapter 7, we introduced the problem of early classification and proposed a method to solve this task, using probabilistic classifiers. The proposed method is able to deal with multi-variate time series by using specific distance measures designed for this type of sequences. However, it could be interesting to provide more specific early classification solutions for this type of temporal data.
- Although the method proposed in Chapter 7 includes a reliability test aimed at discarding outlier series, adjusting the parameters of this method automatically, by using some knowledge about the database, could be very useful. Also, this method could be extended to perform the task of detecting outliers or unusual series.
- In essence, early classification is a multi-objective optimization problem which tries to find a trade-off between two conflicting objectives: earliness and accuracy. However, it is not common to tackle the problem using optimization techniques. As such, we propose analyzing the problem from this point of view.
- In Chapter 8 we have mentioned that the early classifier proposed in Chapter 7 shows very extreme behaviors in the presence of very unbalanced classes, especially if the time series in different classes are very similar to each other in a large section of their length. As such, as future work, we propose an enhancement of this early classifier that takes this factor into account.
- In Chapter 8 we have used the silhouette and the Dunn index to evaluate and compare different clustering results, even those obtained using different distance measures. However, it remains to be proved if these measures are the most suitable ones for this purpose, especially if the final objective is forecasting the future values of time series (travel time in this case). Indeed, the creation of new and more appropriate measures could be an interesting direction to follow.

From the second group of future research directions we propose the following topics:

- Given the large number of travel time modeling problems available in the literature and the proliferation of this area of study in the past few years (see Chapter 4), we think that it would be very helpful for the community to perform a complete empirical comparison of the available methods. This could provide conclusive facts about the advantages and disadvantages of each method and would help the users choose an adequate model for each study site and traffic situation. This would also simplify the construction of hybrid methods that combine different model types for different traffic patterns.
- As shown in Chapter 4, there is a great number of complex travel time prediction and estimation models available in the literature. However, we think that it could be interesting to study the real improvement that is obtained with these models, in comparison with other naive models such as historical means or instantaneous predictors, depending on the real-time and historical data available.
- Finally, the contributions shown in Chapters 6 and 7 could be useful in other application scenarios. As an example, the problem of early classification appears naturally in the control of insuline administration for patients with the Diabetes Mellitus disease. Diabetes Mellitus is a chronic disease related to the inability of the pancreas to produce insulin, or the resistance of the organism to react to insulin [86]. The result is an increase in the glucose level in the blood that must usually be compensated with the external administration of insulin. However, optimally controlling the insulin doses is not a simple task, due to all the factors that affect the glucose level in the blood (glycemia) of a patient [86]. As such, given the patients temporal measurements of glycemia, sport activities, food intakes, etc. early prediction could be used to calibrate the insulin doses and, especially, detect irregularities or abnormalities as soon as possible because they may result in immediate danger or complications to the patient's health.

9.3 Publications

The research work carried out during this thesis has produced the following publications and submissions:

9.3.1 Referred journals

- **U. Mori**, A. Mendiburu, Eamonn Keogh & J.A. Lozano (2015). Reliable early classification of time series based on discriminating the classes over time. *Data Mining and Knowledge Discovery*. Submitted (Major Revision).
- **U. Mori**, A. Mendiburu, & J.A. Lozano (2015). TSdist: An R package for time series similarity calculation. *Journal of Machine Learning Research*. Submitted.

- **U. Mori**, A. Mendiburu, & J.A. Lozano (2015). Similarity measure selection for clustering time series databases. *IEEE Transactions on Knowledge and Data Engineering*. In press.
- **U. Mori**, A. Mendiburu, M. Álvarez & J.A. Lozano (2014). A Review of Travel Time Estimation and Forecasting for Advanced Traveller Information Systems. *Transportmetrica A: Transport Science*, 11(2): 119:157.

9.3.2 Conference communications

- **U. Mori**, A. Mendiburu, & J.A. Lozano (2015).A preliminary analysis on the effect of time series clustering on short term travel time prediction models. *In Proceedings of the International Work Conference on Time Series (ITISE 2015), Granada, Spain, 1-3 July. Pp. 645-656*
- **U. Mori**, A. Mendiburu, & J.A. Lozano (2013). Time Series Database Characterization and Similarity Measure Selection. *In Actas del VII Simposio de Teoría y Aplicaciones de Minería de Datos (TAMIDA 2013), Madrid, Spain, 17-20 September. Pp. 1427-1436.*

Part V

Appendices

A

Generation of synthetic time series databases with specific characteristics

In this appendix the generation process of the synthetic databases used in Chapter 6 is explained in detail. With this information and the code available in the Downloads section of our webpage, the readers can easily generate the databases used in this chapter. Furthermore, the explanations included in this appendix and the attached code enable the generation of new databases with different characteristics that can be used in other research projects.

A.1 Overall synthetic database generation process

In this first section, the overall process for the generation of the synthetic databases used in Chapter 6 is introduced. It must be noted that these databases are designed to be used for time series clustering tasks. As such, they contain several underlying groups or clusters, which are intentionally and artificially generated. This fact will be one of the main points of the generation process and will be explained in the following paragraphs.

To begin with, the synthetic databases presented in this appendix can be divided into 8 distinct groups (see second column, **'Database type'** in Table A.1). In the experimentation of Chapter 6, for each type of database we have generated various different examples by modifying the level of noise, the level of outliers, the mean level of shift and the mean level of local warp within the ranges shown in Table A.1. In order to avoid homogeneity, we only generate one example database for each combination of categories and each database type. Furthermore, we discard the databases with 0 values for all options because they are too simple and unrealistic. An example of this generation process for the databases of type CBF can be seen in Algorithm 1.

Database type	Noise level	Outliers level	Shift level	Warp level
Armas ($k^* = 8$) Synthetic control ($k^* = 6$)	None=0 Low={1,2} Med={3,4} High={5,6}	None=0 Low={1,2} Med={3,4} High={5,6}	None=0 Low={1,...,5} Med={6,...,12} High={13,...,30}	None=0
Sines ($k^* = 5$) Rational ($k^* = 4$) Seasonal ($k^* = 4$)	None=0 Low={1,2} Med={3,4} High={5,6}	None=0 Low={1,2} Med={3,4} High={5,6}	None=0 Low={1,...,5} Med={6,...,12} High={13,...,18}	None=0
CBF ($k^* = 3$)	None=0 Low={1,2,3} High={3,4,5}	None=0 Low={1,2,3} High={3,4,5}	None=0 Low={1,...,5} High={6,...12}	None=0 Low={1,...,5} High={6,...,10}
Two Patterns ($k^* = 4$) Kohler & Lorenz ($k^* = 5$)	None=0 Low={1,2,3} High={3,4,5}	None=0 Low={1,2,3} High={3,4,5}	None=0 Low={1,...,5} High={6,...,10}	None=0 Low={1,...,5} High={6,...,8}

Table A.1. Characteristic options for the generation of the synthetic databases.

Algorithm 1 Generation of the synthetic databases of type CBF.

for *noise* in $\{none, low, high\}$ **do**
  for *outliers* in $\{none, low, high\}$ **do**
    for $shift$ in $\{none, low, high\}$ **do**
      for *warp* in $\{none, low, high\}$ **do**
        if $noise \neq none$ **or** $outlier \neq none$ **or** $shift \neq none$ **or** $warp \neq none$ **then**
          $noise\_level \leftarrow$ value randomly chosen inside *noise* category.
          $outlier\_level \leftarrow$ value ranmdoly chosen inside *outlier* category.
          $shift\_level \leftarrow$ value randomly chosen inside $shift$ category.
          $warp\_level \leftarrow$ value randomly chosen inside *warp* category.
          $dimension \leftarrow$ value randomly chosen from the second column of Table A.2.
          $proportions \leftarrow$ value randomly chosen from the third column of Table A.2.
          Generate and save database of type CBF with selected options.
        end if
      end for
    end for
  end for
end for
return 80 databases of type CBF

If this same process is repeated for all the database types, it results in a total of 555 synthetic databases ($2 \times 4 \times 4 \times 4 \times 1 + 3 \times 4 \times 4 \times 4 \times 1 + 1 \times 3 \times 3 \times 3 \times 3 + 2 \times 3 \times 3 \times 3 \times 3 - 8$). Nevertheless, in order to be able to apply Algorithm 1, some additional points must be specified.

First, the meaning of the variables *dimension* and *proportions* must be explained (see lines 10 and 11 of Algorithm 1). The first one represents the dimension of the database and is specified by the number of series that it contains and their respective length ($N \times L$). To define the second variable we must recall that we are working in a clustering environment. Because of this, each synthetic dataset will be generated is such a way that it will contain a predefined and previously known set of clusters. The series belonging to the same cluster will have similar shapes. In this context, the *proportion* variable represents the proportion of series belonging to each cluster.

Once this is explained, we must define a method to generate a database of a given type and with some specific characteristics (see line 12 of Algorithm 1). Each type of synthetic database included in this appendix is defined by k^* predefined shapes, each shape representing one cluster. For example, in the classical Cylinder-Bell-Funnel (CBF) database [103], the three shapes that can be seen in Figure 7.1 are used to represent three different clusters. For the details about the basic shapes of the other types of synthetic database see Section A.2. In this context, to create a synthetic database of a certain type (e.g., CBF) and with some specific characteristics, first the dimension and the proportion of series in each cluster are chosen randomly from the set of predefined options (see Table A.2). Based on this, we replicate the k^* shapes that define that type of synthetic database (the shapes in Figure 7.1 in the case of CBF) accordingly. Next, the noise, outliers, shift and warp will be added to the database in the following manner:

- **Noise:** The noise is introduced separately in each series of the database by adding random values issued from a normal distribution of mean 0. The standard deviation (σ) of this distribution is defined by the level of noise normalized by the maximum (max) and minimum (min) values of the series:

$$\sigma = \frac{\text{noise level}}{max - min} \tag{A.1}$$

- **Outliers:** The introduction of outliers will be carried out independently for each time series in the database. The selected outlier level will represent the proportion of points in the series that will become outliers. Given a specific outlier level, the corresponding number of points are selected randomly from the series and interchanged with points randomly chosen from other series in order to convert them into outliers.

- **Shift:** Contrary to previous features, the introduction of shift in a time series database can not be carried out separately for each time series because it is a global variable. Each series must be shifted in a specific manner so that, overall, a pre-specified mean level of shift is obtained. The proposed solution is to shift each time series X_i following a random variable P_i that takes random integer values in the interval $[-m, m]$, m being a positive integer. If this procedure is followed, the mean level of shift (sh) introduced in the database can be quantified by calculating the expectation of the mean phase difference between all pairs of series in the database:

$$sh = \frac{2}{N(N-1)} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \mathbb{E}(|P_i - P_j|) = \frac{4(m+1)m}{3(2m+1)} \tag{A.2}$$

where N is the total number of series in the database. Now, we are interested in the inverse problem: given a level of mean shift (sh), we want to find the maximum value m to which we can shift the series. This can be obtained by simply calculating the inverse of Equation A.2:

$$m = \frac{3 \cdot sh - 2 + \sqrt{9 \cdot sh^2 + 4}}{4} \tag{A.3}$$

With this equation and the procedure explained above, the shift can be introduced in a controlled manner in synthetic time series databases. The only inconvenience of Equation A.3 is that, for most values of sh, this inverse function does not return integer values. This is solved by simply truncating the result.

- **Warp or local scaling**: Given the difficulty of introducing warp in general time series databases, it has only been considered for the CBF [103], Two Patterns [70] and Kohler & Lorenz [108] database types, where it can be done in a simple way. At least one of the shapes in these types of synthetic databases is a piecewise function and, in this case, the local scaling is introduced by modifying the limits of the pieces in both directions in a random way and inside a given interval $[-m, m]$, where m is chosen from the options shown in the last column of Table A.1.

The code used to generate the databases and the databases themselves are available together with this document in the Downloads section of our website [1].

A.2 Additional specifications of the synthetic database generation

In this section, some additional specifications needed to completely determine the generation process of the synthetic databases are given. To begin with, we recall that in the generation of the synthetic databases the dimension of the database and the proportion of series in each cluster were chosen randomly from a set of options. In Table A.2 we summarize these options.

[1] http://www.sc.ehu.es/ccwbayes/isg/index.php?option=com_content&view=article&id=122&Itemid=100#timeseries

Database type	Dimension	Proportions
Armas (K=8)	70x70, 100x70, 150x70, 200x70, 300x70, 500x70,	(1/8,1/8,1/8,1/8,1/8,1/8,1/8,1/8) (1/4,1/8,1/8,1/16,3/16,1/8,1/8,0) (1/5,1/5,0,0,1/5,0,1/5,1/5) (1/4,0,1/3,1/8,1/8,0,1/6,0)
Synthetic control (K=6)	70x100, 100x100, 150x100, 200x100, 300x100, 500x100,	(1/6,1/6,1/6,1/6,1/6,1/6) (1/6,1/12,1/12,1/6,1/4,1/4) (1/3,1/3,1/12,1/12,1/12,1/12)
Sines (K=5) Kohler & Lorenz (K=5)	70x150, 100x150, 150x150, 200x150, 300x150, 70x200, 100x200, 150x200,	(1/5,1/5,1/5,1/5,1/5) (1/10,1/5,1/10,2/5,1/5) (1/15,2/5,1/15,2/5,1/15) (2/5,1/10,1/5,1/10,1/5)
CBF (K=3)	200x200, 300x200, 70x300, 100x300, 150x300, 200x300,	(1/3,1/3,1/3) (1/6,1/3,1/2) (2/3,1/6,1/6)
Two Patterns (K=4) Rational (K=4) Seasonal (K=4)	70x500, 100x500	(1/4,1/4,1/4,1/4) (1/3,1/3,1/6,1/6) (1/8,1/4,1/2,1/8)

Table A.2. Possible values for the *dimension* and *proportion* variables.

Furthermore, in order to fully describe the different synthetic database types (see 'Database Type' in Table A.1) the shapes that define each of them are specified:

- **ARMAs:** 8 series obtained from different initializations of an ARMA(3,2) process with coefficients AR=(1,-0.24,0.1) and MA=(1,1.2) conform this synthetic database.
- **Synthetic control:** This time series database is an example of a synthetic database that was first introduced by Pham and Chan [162]. It is based on 6 basic shapes:

$$f_1(t) = 80 + r(t)$$
$$f_2(t) = 80 + 15\sin(\frac{2\pi t}{T})$$
$$f_3(t) = 80 + 0.4t$$
$$f_4(t) = 80 - 0.4t$$
$$f_5(t) = \begin{cases} 80 & t \leq \lfloor \frac{L}{2} \rfloor \\ 90 & \lfloor \frac{L}{2} \rfloor < t \end{cases}$$
$$f_6(t) = \begin{cases} 90 & t \leq \lfloor \frac{L}{2} \rfloor \\ 80 & \lfloor \frac{L}{2} \rfloor < t \end{cases}$$

where r(t) is a random value issued from a $\mathcal{N}(0,3)$ distribution, L is the length of the series and T is the period and is defined as a third of the length of the series.

- **Sines:** This type of synthetic database is defined by 4 different shapes that are based on the function $f(t) = sin(\frac{2\pi t}{T}) \cdot r$, where r is a random number between 1 and 1.1 and $t = 1, ..., L$. The first shape is obtained by generating instances of f itself. The second shape is obtained by fitting Local Polynomial Regressions (LOESS) to instances of f using polynomials of degree 1 and a smoothing parameter of 0.25. To build the third shape, a sample of the f function is taken, but the values higher than τ are modified into a constant value of τ, τ being a random number between 0.9 and 0.99. The fourth shape is the negative counterpart of the previous case. Finally, the fifth shape takes an instance of f and adds a random number between 0.2 and 0.3 to a part of the series. The objective of introducing this synthetic database type is to create databases with very similar clusters.
- **Rational:** This database type is generated following the same idea as in the previous case and consists of 4 shapes. The first shape is defined by the values of $f(x) = \frac{x}{x^2+1} \cdot r$ in L equidistant points in the interval $[-20, 20]$, r being a random number between 1 and 1.1 and L the length of the series. The second shape is defined by the function $g(x) = \frac{x}{x^2+2} \cdot r$, also in the interval $[-20, 20]$. The third is a smooth LOESS approximation of f with polynomials of degree 1 and a smoothing parameter of 0.4, and the fourth is a cubic spline interpolation.
- **Seasonal:** This type of database consists of 4 shapes based on adding sinusoidal functions:

$$
\begin{aligned}
&f_1(t) = \sin(\frac{2\pi t}{T}) + \frac{1}{2}\sin(\frac{4\pi t}{T} - 2) + \frac{1}{3}\sin(\frac{4\pi t}{T} - 2) + \frac{1}{3}\cos(\frac{8\pi t}{T} - 2) + \\
&\sin(\frac{10\pi t}{T} - 6) \\
&f_2(t) = \frac{3}{2}\sin(\frac{2\pi t}{T}) + \frac{1}{2}\cos(\frac{12\pi t}{T} - 1) + \frac{1}{6}\cos(\frac{4\pi t}{T} - 7) + \sin(\frac{4\pi t}{T} - 9) \\
&f_3(t) = 1.2\cos(\frac{6\pi t}{T} + 7) + 0.9\sin(\frac{2\pi t}{T} - 2) + \frac{1}{3}\cos(\frac{2\pi t}{7T}) \\
&f_4(t) = 0.24\sin(\frac{4.4\pi t}{T} + 7) + \frac{1}{2}\cos(\frac{2\pi t}{T} - 5) + \frac{1}{3}\cos(\frac{4\pi t}{7T})
\end{aligned}
$$

where T is a half of the series length and $t = 1, ..., L$. The objective of mixing many sinusoidal forms is to obtain time series with many peaks.

- **CBF:** This type of synthetic database is also available in the UCR repository. Three basic shapes are defined as follows:

$$f_1(t) = \begin{cases} 0 & t \leq a \\ 6 + r(t) & a < t \leq b \\ 0 & t \geq b \end{cases} \qquad f_2(t) = \begin{cases} 0 & t \leq a \\ (6 + r(t)) \cdot \frac{t-a}{b-a} & a < t \leq b \\ 0 & t \geq b \end{cases}$$

$$f_3(t) = \begin{cases} 0 & t \leq a \\ (6 + r(t)) \cdot \frac{b-t}{b-a} & a < t \leq b \\ 0 & t \geq b \end{cases}$$

where a is initially defined as $\lfloor \frac{L}{3} \rfloor$ and b as $\lfloor \frac{2L}{3} \rfloor$, being L the length of the series.

- **Two Patterns:** This type of synthetic database was introduced by Geurts [70] and an example is included in the UCR repository. The idea is two combine two patterns *us* and *ds* as follows:

$$f_1(t) = \begin{cases} r(t) & 0 \leq t \leq t_1 \\ us(t - t_1, l_1) & t_1 < t \leq t_1 + l_1 \\ r(t) & t_1 + l_1 < t \leq t_2 \\ us(t - t_2, l_2) & t_2 < t \leq t_2 + l_2 \\ r(t) & t \geq t_2 + l_2 \end{cases}$$

$$f_2(t) = \begin{cases} r(t) & 0 \leq t \leq t_1 \\ us(t - t_1, l_1) & t_1 < t \leq t_1 + l_1 \\ r(t) & t_1 + l_1 < t \leq t_2 \\ ds(t - t_2, l_2) & t_2 < t \leq t_2 + l_2 \\ r(t) & t \geq t_2 + l_2 \end{cases}$$

$$f_3(t) = \begin{cases} r(t) & 0 \leq t \leq t_1 \\ ds(t - t_1, l_1) & t_1 < t \leq t_1 + l_1 \\ r(t) & t_1 + l_1 < t \leq t_2 \\ us(t - t_2, l_2) & t_2 < t \leq t_2 + l_2 \\ r(t) & t \geq t_2 + l_2 \end{cases}$$

$$f_4(t) = \begin{cases} r(t) & 0 \leq t \leq t_1 \\ ds(t - t_1, l_1) & t_1 < t \leq t_1 + l_1 \\ r(t) & t_1 + l_1 < t \leq t_2 \\ ds(t - t_2, l_2) & t_2 < t \leq t_2 + l_2 \\ r(t) & t \geq t_2 + l_2 \end{cases}$$

where,

$$us(t, l) = \begin{cases} -5 & 0 \leq t \leq \frac{l}{2} \\ 5 & t_1 < t \leq t_1 + l_1 \end{cases} \qquad ds(t, l) = \begin{cases} 5 & 0 \leq t \leq \frac{l}{2} \\ -5 & t_1 < t \leq t_1 + l_1 \end{cases}$$

where, t_1 is initially set in $\lfloor \frac{L}{3} \rfloor$, t_2 in $\lfloor \frac{2L}{3} \rfloor$, l_1 and l_2 both take a value of $0.1 \cdot L$ and $r(t)$ is a random value extracted from a $\mathcal{N}(0,1)$ distribution.

- **Kohler & Lorenz:** This last synthetic database was created by making some modifications on the synthetic series presented in Köhler and Lorenz [108]. The following 5 shapes are defined:

$$f_1(t) = 13 \cdot r(t) \sin(5 \cdot r(t) \frac{t}{T})$$

$$f_2(t) = \begin{cases} \frac{-t^2}{100} & t \leq \lfloor \frac{L}{2} \rfloor \\ \frac{t^2}{530} & \lfloor \frac{L}{2} \rfloor > t \end{cases}$$

$$f_3(t) = \begin{cases} 5 & t \leq \lfloor \frac{L}{3} \rfloor \\ -12 & \lfloor \frac{L}{3} \rfloor < t \leq \lfloor \frac{2L}{3} \rfloor \\ 8 & t \geq \lfloor \frac{2L}{3} \rfloor \end{cases}$$

$$f_4(t) = \begin{cases} 15 \cdot r(t) \sin(15 \cdot r(t) \frac{t}{T}) & t \leq \lfloor \frac{L}{8} \rfloor \\ r(t) \sin(7 \cdot r(t) \frac{t}{T}) & \lfloor \frac{L}{8} \rfloor < t \leq \lfloor \frac{2L}{8} \rfloor \\ 10 & \lfloor \frac{2L}{8} \rfloor < t \leq \lfloor \frac{3L}{8} \rfloor \\ -2 & \lfloor \frac{3L}{8} \rfloor < t \leq \lfloor \frac{4L}{8} \rfloor \\ \frac{(10t/L)^2}{2} & \lfloor \frac{4L}{8} \rfloor < t \leq \lfloor \frac{5L}{8} \rfloor \\ \frac{-(2t/L)^2}{2} & t \geq \lfloor \frac{5L}{8} \rfloor \end{cases}$$

$f_5(t) =$ series generated from an ARMA(3,2) process with parameters AR=(1,-0.24,0.1) and MA=(1,1.2)

where L is the length of the series, T is 10% of the length of the series and $r(t)$ is a random number obtained from a $\mathcal{N}(0, 0.1)$ distribution.

B

TSdist: An R package for time series similarity measures calculation

As commented in Section 3.2, distance measures are crucial aspects of many time series data mining tasks. `R` is a popular programming language and a free software environment for statistical computing, data analysis and graphics [167] which can be extended by means of *packages*, contributed by the users themselves. In this context, a few `R` packages such as `dtw` [73], `pdc` [16], `proxy` [136], `longitudinalData` [69] and `TSclust` [141] provide implementations of some time series distance measures. However, many of the most popular distances reviewed by Esling and Agon [64], Liao [124] and Wang et al. [228] are not available in these packages.

In this chapter, the `TSdist` package [142] for the `R` statistical software is presented. In addition to providing wrapper functions to all the distance measures implemented in the previously mentioned packages, `TSdist` provides the implementation of a new set of distance measures designed for time series. These have been selected because they are considered useful in recent reviews on the topic [64, 124, 228]. In this manner, and to the best of our knowledge, this package provides the most up-to-date coverage of the published time series distance measures in `R`.

B.1 Design and Implementation

As can be seen in Figure B.1, the core of the `TSdist` package consists of three types of functions. To begin with, the functions of the type `MethodDistance` can be used to calculate distances between pairs of numerical and univariate vectors. Of course, `Method` must be substituted by the name of a specific distance measure. These functions conform the basis of the package and all the rest of the functions are built around them. Most of them are implemented exclusively in `R` language but, for the Edit based distances, the internal routines are implemented in C language, for reasons of computational efficiency.

In the next level, the wrapper function called `TSDistances` enables the calculation of distance measures between univariate time series objects of type

`ts`, `xts` [182] and `zoo` [249]. This function just takes care of the conversion of data types and then makes use of the desired `MethodDistance` function.

Finally, the `TSDatabaseDistances` function is designed to build distance matrices by calculating all the pairwise distances between the series in a time series database. Upon loading the `TSdist` package, the `TSDistances` function is automatically included in the `pr_DB` database, which is a list of similarity measures defined in the `proxy` package [135]. This directly enables the use of the `dist` function, the baseline `R` function to calculate distance matrices, with the dissimilarity measures defined in the `TSdist` package. This is the general strategy followed by the `TSDatabaseDistances` function and, only for a few special measures, the distance matrix is calculated in other ad-hoc manners for efficiency purposes. As an additional capability of the `TSDatabaseDistances` function, the distance matrices can not only be calculated for a single database, but also for two separate databases. In this second case, all the pairwise distances between the series in the first database and the second database are calculated. This last feature is especially useful for classification tasks where train/test validation frameworks are frequently used.

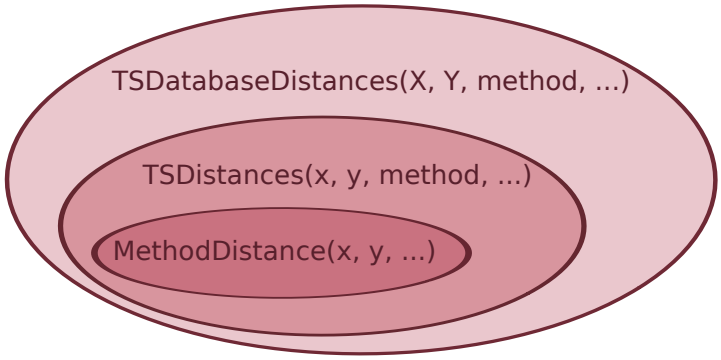


Fig. B.1. Structure and organization of the TSdist package.

B.2 Summary of distance measures included in TSdist

In Table B.1, a summary of the distance measures included in `TSdist` is presented. Since the package includes wrapper functions to distance measures hosted in other packages, the original package is also cited in the table.

As can be seen, the distance measures implemented specifically for `TSdist` complement the set of measures already included in other packages, contributing to a more thorough coverage of the existing time series distance measures. As the most notable example, edit based distances for numeric time series have been introduced, which was a completely overlooked category of distance measures in previous `R` packages.

Table B.1. Summary of distance measures for time series implemented in R.

	proxy	long.Data	TSclust	dtw	pdc	TSdist
Shape-based distances						
L_p distances	✓					
Frechet distance		✓				
Dynamic Time Warping				✓		
Keogh.LB for DTW						✓
Short Time Series Distance						✓
DISSIM						✓
Cross-correlation based						✓
(Partial) Autocorrelation based			✓			
Pearson correlation based			✓			
Edit based distances						
Edit Distance for Real Sequences						✓
Edit Distance with Real Penalty						✓
Longest Common Subsequence						✓
Feature-based distances						
Fourier Decomposition based						✓
TQuest						✓
Wavelet Decomposition based			✓			
(Integrated) Periodogram based			✓			
SAX representation based			✓			
Spectral Density based			✓			
Structure-based distances						
Piccolo distance			✓			
Maharaj distance			✓			
Cepstral based distances			✓			
Compression based distances			✓			
Complexity invariant distance			✓			
Permutation distribution based distance					✓	
Prediction based						
Non Parametric Forecast based			✓			

B.3 User interface by example

The `TSdist` package is available from the CRAN repository, where the source files for Unix platforms and the binaries for Windows and some OS-X distributions can be downloaded. For more information on software pre-requisites and detailed instructions on the installation process of `TSdist`, see the README file included in the inst/doc directory of the package.

The most common usage of time series distance measures is within clustering and classification tasks, and all the measures included in this package can be useful within these two frameworks. As a support for these two tasks, the `TSdist` package includes two functions (`OneNN` and `KMedoids`) which implement the 1-NN classifier and the K-medoids clustering algorithm, respectively. In the following paragraphs, we provide a complete example that shows how the package could be used to solve a time series classification problem.

Suppose we want to classify the series in the `example.database2` database (included in `TSdist`), which contains 100 series from 6 very different classes. In order to simulate a typical classification framework, we divide the database into two sets by randomly selecting 30% of the series for training purposes and 70% for testing [1]. Then, we apply the 1-NN classifier to the testing set with different distance measures and analyze the classification error in each case:

```
> OneNN(train, trainclass, test, testclass, "euclidean")$error

[1] 0

> OneNN(train, trainclass, test, testclass, "acf")$error

[1] 0.4142857

> OneNN(train, trainclass, test, testclass, "dtw")$error

[1] 0
```

If the selected distance measure requires the definition of any parameters, these should be included at the end of the call:

```
> OneNN(train, trainclass, test, testclass, "tquest", tau=85)$error

[1] 0.2571429
```

As can be seen, the best results are provided by the DTW and Euclidean distances, which yield a perfect classification for this toy database. However, previous experiments show that there is no "best" distance measure which is suitable for all databases, [228]. In this context, a specific distance measure must be selected, in each case, in order to obtain satisfactory results. The large number of distance measures included in `TSdist` and the simple design of this package allows the user to try different distance measures directly, simplifying the distance measure selection process considerably.

For more detailed information on the databases and functions included in the `TSdist` package, and a more complete set of examples, the reader can access the help pages or the manual of the `TSdist` package and the vignette included within.

[1] The code to load and prepare the data is available in the documentation of the `OneNN` function.

References

[1] Abbott-jard, M., Shah, H., and Bhaskar, A. (2013). Empirical evaluation of Bluetooth and Wifi scanning for road transport. In *Proceedings of the Australasian Transport Research Forum*, number October, pages 1–14.

[2] Abdulhai, B. and Tabib, S. M. (2003). Spatio-temporal inductance-pattern recognition for vehicle re-identification. *Transportation Research Part C*, 11(3-4):223–239.

[3] Agrawal, R., Faloutsos, C., and Swami, A. (1993). Efficient Similarity Search in Sequence Databases. In *Proceedings of the 4th International Conference of Foundations of Data Organization and Algorithms*, volume 5, pages 69–84.

[4] Alvarez, F. M., Troncoso, A., Riquelme, J. C., and Ruiz, J. S. A. (2011). Energy Time Series Forecasting Based on Pattern Sequence Similarity. *IEEE Transactions on Knowledge and Data Engineering*, 23(8):1230–1243.

[5] Aß falg, J., Kriegel, H.-p., Kröger, P., Kunath, P., Pryakhin, A., and Renz, M. (2006). Similarity Search on Time Series based on Threshold Queries. In *Proceedings of the 10th international conference on Advances in Database Technology*, pages 276–294.

[6] Bachmann, C. (2011). *Multi-Sensor Data Fusion for Traffic Speed and Travel Time Estimation*. PhD thesis, University of Toronto.

[7] Bar-Gera, H. (2007). Evaluation of a cellular phone-based system for measurements of traffic speeds and travel times: A case study from Israel. *Transportation Research Part C: Emerging Technologies*, 15(6):380–391.

[8] Batista, G. E., Wang, X., and Keogh, E. J. (2011). A Complexity-Invariant Distance Measure for Time Series. In *Proceedings of the SIAM International Conference on Data Mining*, pages 699–710.

[9] Ben-Akiva, M., Bierlaire, M., Koutsopoulos, H., and Mishalani, R. (1998). DynaMIT : a simulation-based system for traffic prediction. In *DACCORD Short Term Forecasting Workshop*, pages 1–12.

[10] Berndt, D. J. and Clifford, J. (1994). Using Dynamic Time Warping to Find Patterns in Time Series. In *AAAI-94 Workshop on Knowledge Discovery in Databases.*, pages 359–370.

[11] Bhaskar, A. and Chung, E. (2013). Fundamental understanding on the use of Bluetooth scanner as a complementary transport data. *Transportation Research Part C: Emerging Technologies*, 37:42–72.

[12] Bigazzi, A. Y., Siri, H., and Bertini, R. L. (2009). Effects of Temporal Data Aggregation on Performance Measures and Other Intelligent Transportation Systems Applications. *Transportation Research Record: Journal of the Transportation Research Board*, 2160:96–106.

[13] Bovy, P. and R. Thijs (2000). *Estimators of Travel Time for Road Networks*. Delft University Press, Delft, The Netherlands.

[14] Boyce, D., Rouphail, N., Buren, W. V., South, S., and Kirson, A. (1993). Estimation and Measurement of Link Travel Times in the ADVANCE Project. In *IEEE - IEE Vehicle Navigation & Information Systems Conference*, pages 62–66.

[15] Brackstone, M. and Mcdonald, M. (2000). Car-following : a historical review. *Transportation Research Part F*, 2(1999):181–196.

[16] Brandmaier, A. M. (2015). pdc: An r package for complexity-based clustering of time series. *Journal of Statistical Software*.

[17] Bregón, A., Simón, M. A., Rodríguez, J. J., Alonso, C., Pulido, B., and Moro, I. (2006). Early Fault Classification in Dynamic Systems Using Case-Based Reasoning. In *Proceeding CAEPIA'05 Proceedings of the 11th Spanish association conference on Current Topics in Artificial Intelligence*, pages 211–220.

[18] Breunig, M. M., Kriegel, H.-p., and Ng, R. T. (2000). LOF: Identifying Density-Based Local Outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 93–104.

[19] Calvert, S. C., van Lint, J. W. C., and Hoogendoorn, S. P. (2010). A hybrid travel time prediction framework for planned motorway roadworks. In *Proceedings of the International IEEE Conference on Intelligent Transportation Systems*, pages 1770–1776. Ieee.

[20] Cameron, G. D. and Duncan, G. I. (1996). PARAMICS, Parallel microscopic simulation of road traffic. *The Journal of Supercomputing*, 10(1):25–53.

[21] Carrillo, E., Heniford, B., Dykes, J., McKenzie, E., Polk, H., and Richardson, J. (1997). Cardiac herniation producing tamponade: the critical role of early diagnosis. *The Journal of Trauma*, 43:19–23.

[22] Castillo, E., Nogal, M., Menéndez, J. M., Sánchez-cambronero, S., and Jiménez, P. (2012). Generalized Beta-Gaussian Bayesian Networks. *IEEE Transactions on Intelligent Transportation Systems*, 13(2):565–581.

[23] Celikoglu, H. B. (2007). A dynamic network loading process with explicit delay modelling. *Transportation Research Part C: Emerging Technologies*, 15(5):279–299.

[24] Celikoglu, H. B. (2013a). Flow-Based Freeway Travel-Time Estimation : Dynamic Path Loading. *IEEE Transactions on Intelligent Transportation Systems*, PP(99):1–10.

[25] Celikoglu, H. B. (2013b). Reconstructing freeway travel times with a simplified network flow model alternating the adopted fundamental diagram. *European Journal of Operational Research*, 228(2):457–466.

[26] Chang, G.-l. and Park, S. Y. (2011). Intelligent transportation system field demonstration: intergration of variable speed limit control and travel time estimation for a recurrently congested highway. *Transportation Research Record*, 2243:55–66.

[27] Chang, H., Park, D., Lee, S., Lee, H., and Baek, S. (2010). Dynamic multi-interval bus travel time prediction using bus transit data. *Transportmetrica A: Transport Science.*, 6(1):19–38.

[28] Chen, L., Ozsu, M. T., and Oria, V. (2005). Robust and Fast Similarity Search for Moving Object Trajectories. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, pages 491–502.

[29] Chen, M. and Chien, S. I. J. (2000). Determining the Number of Probe Vehicles for Freeway Travel Time Estimation by Microscopic Simulation. *Transportation Research Record: Journal of the Transportation Research Board*, 1719:61–68.

[30] Chen, M. and Chien, S. I. J. (2001). Dynamic Freeway Travel-Time Prediction with Probe Vehicle Data: Link Based Versus Path Based. *Transportation Research Record: Journal of the Transportation Research Board*, 1768:157–161.

[31] Chen, Y., van Zuylen, H. J., and Qipeng, Y. (2010). Travel Time Prediction on Urban Networks Based on Combining Rough Set with Support Vector Machine. In *Proceedings of the 2010 International Conference on Logistics Systems and Intelligent Management*, pages 586–589.

[32] Cheu, R. L., Lee, D.-H., and Xie, C. (2001). An Arterial Speed Estimation Model Fusing Data from Stationary and Mobile Sensors. In *Proceedings of the IEEE International Conference on Intelligent Transportation System*, pages 573–578.

[33] Cheu, R. L., Xie, C., and Lee, D.-H. (2002). Probe Vehicle Population and Sample Size for Arterial Speed Estimation. *Computer-Aided Civil and Infrastructure Engineering*, 17:53–60.

[34] Chien, S. I. J., Liu, X., and Ozbay, K. (2003). Predicting Travel Times for the South Jersey Real-Time Motorist Information System. *Transportation Research Record: Journal of the Transportation Research Board*, 1982(03):32–40.

[35] Choi, K. (1999). Data Fusion Methodology for Link Travel Time Estimation for Advanced Traveler Information System. *KSCE Journal of Civil Engineering*, 3(1):1–14.

[36] Choi, K. and Chung, Y. (2002). A Data Fusion Algorithm for Estimating Link Travel Time. *Journal of Intelligent Transportation Systems*, 7(3):235–260.

[37] Chu, L., Oh, J., and Recker, W. (2005). Adaptive Kalman Filter Based Freeway Travel time Estimation. In *Transportation Research Board 84th Annual Meeting*, pages 1–21.

[38] Chung, E. and Kuwahara, M. (2005). Performance Evaluation of An Adaptive Travel Time Prediction Model. In *Proceedings of the 8th International IEEE Conference on Intelligent Transportation Systems*, pages 1000–1005.

[39] Claudel, C. G., Hofleitner, A., Mignerey, N. D., and Bayen, A. M. (2009). Guaranteed bounds on highway travel times using probe and fixed data. In *88th Annual Meeting of the Transportation Research Board*, pages 1–19.

[40] Cleveland, R. B., Cleveland, W. S., McRae, J. E., and Terpenning, I. (1990). STL: A Seasonal-Trend Decomposition Procedure Based on Loess. *Journal of Official Statistics*, 6(1):3–73.

[41] Coifman, B. (1998). A New Algorithm for Vehicle Reidentification and Travel Time Measurement on Freeways. In *Applications of Advanced Technologies in Transportation, American Society of Civil Engineers*, pages 167–174.

[42] Coifman, B. (2002). Estimating Travel Times and Vehicle Trajectories on Freeways Using Dual Loop Detectors. *Transportation Research: Part A*, 36(4):351–364.

[43] Coifman, B. and Krishnamurthy, S. (2007). Vehicle reidentification and travel time measurement across freeway junctions using the existing detector infrastructure. *Transportation Research Part C*, 15(3):135–153.

[44] Collar, N. J., Andreev, A., Chan, S., Crosby, M., Subramanya, S., Tobias, J., and (Eds) (2001). Chrysomma altirostre. In *Threatened Birds of Asia: The BirdLife International Red Data Book*, pages 2112–2119. BirdLife International.

[45] Constantine, W. and Percival, D. (2012). wmtsa: Wavelet Methods for Time Series Analysis.

[46] Cortés, C. E., Lavanya, R., Oh, J.-S., and Jayakrishnan, R. (2001). A general purpose methodology for link travel time estimation using multiple point detection. *Transportation Research Record: Journal of the Transportation Research Board*, 1802:181–189.

[47] Cremer, M. (1995). On the Calculation of Individual Travel Times by Macroscopic Models. In *Proceedings of the Vehicle Navigation and Information Systems Conference*, pages 187–193.

[48] Dailey, D. J. (1993). Travel Time Estimation Using Cross Correlation Techniques. *Transportation Research Part B*, 26(2):97–107.

[49] Dailey, D. J. (1997). Travel Time estimates using a series of single-loop volume and occupancy measurements. In *Transportation Research Board 76th Annual Meeting*, number 970378.

[50] Dell'Amore, C. (2015). 'Extinct' Bird Rediscovered in Myanmar, Surprising Scientists.

[51] Demšar, J. (2006). Statistical Comparisons of Classifiers over Multiple Data Sets. *The Journal of Machine Learning Research*, 7:1–30.

[52] Dharia, A. and Adeli, H. (2003). Neural network model for rapid forecasting of freeway link travel time. *Engineering Applications of Artificial Intelligence*, 16(7-8):607–613.

[53] Dia, H. (2001). An object-oriented neural network approach to short-term traffc forecasting. *European Journal Of Operational Research*, 131.

[54] Dion, F. and Rakha, H. (2003). Estimating Spatial Travel Times using Automatic Vehicle Identification Data. In *Transportation Research Board Annual Meeting*, pages 1–30.

[55] Dion, F. and Rakha, H. (2006). Estimating dynamic roadway travel times using automatic vehicle identification data for low sampling rates. *Transportation Research Part B*, 40(9):745–766.

[56] Dion, F., Rakha, H., and Kang, Y.-S. (2004). Comparison of delay estimates at under-saturated and over-saturated pre-timed signalized intersections. *Transportation Research Part B*, 38(2):99–122.

[57] Domenichini, L., Salerno, G., Fanfani, F., Bacchi, M., Giaccherini, A., Costalli, L., and Baroncelli, C. (2012). Travel Time in Case of Accident Prediction Model. *Procedia - Social and Behavioral Sciences*, 53:1078–1087.

[58] Du, L., Peeta, S., and Kim, Y. H. (2012). An adaptive information fusion model to predict the short-term link travel time distribution in dynamic traffic networks. *Transportation Research Part B*, 46(1):235–252.

[59] El Esawey, M. and Sayed, T. (2010). Travel Time Estimation in Urban Networks Using Buses as Probes. In *Annual Conference of the Transportation Association of Canada*, pages 1–22.

[60] El Faouzi, N.-E., Klein, L. A., and De Mouzon, O. (2009). Improving Travel Time Estimates from Inductive Loop and Toll Collection Data with Dempster-Shafer Data Fusion. *Transportation Research Record: Journal of the Transportation Research Board*, 2129:73–80.

[61] El Faouzi, N.-E. and Lefevre, E. (2006). Classifiers and Distance-Based Evidential Fusion For Road Travel Time Estimation. In *Proceedings of SPIE, The International Society for Optical Engineering*, pages 1–15.

[62] El Faouzi, N.-E., Leung, H., and Kurian, A. (2011). Data fusion in intelligent transportation systems : Progress and challenges âĂŞ A survey. *Information Fusion*, 12(1):4–10.

[63] Elhenawy, M., Chen, H., and Rakha, H. a. (2014). Dynamic travel time prediction using data clustering and genetic programming. *Transportation Research Part C: Emerging Technologies*, 42:82–98.

[64] Esling, P. and Agon, C. (2012). Time-series data mining. *ACM Computing Surveys*, 45(1):1–34.

[65] Fei, X., Lu, C.-C., and Liu, K. (2011). A bayesian dynamic linear model approach for real-time short-term freeway travel time prediction. *Transportation Research Part C*, 19(6):1306–1318.

[66] Friedman, M. (1940). A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11(1):86–92.

[67] Fu, T.-C. (2011). A Review on Time Series Data Mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181.

[68] Fulcher, B. D. and Jones, N. S. (2014). Highly comparative feature-based time-series classification. *IEEE Transactions on Knowledge and Data Engineering*, 26(12):1–20.

[69] Genolini, C. (2014). *longitudinalData: Longitudinal Data*. R package version 2.2.

[70] Geurts, P. (2002). *Contributions to decision tree induction: bias/variance tradeoff and time series classification.* PhD thesis, University of Liege, Belgium.

[71] Ghalwash, M. F., Radosavljevic, V., and Obradovic, Z. (2014). Utilizing temporal patterns for estimating uncertainty in interpretable early decision making. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '14*, pages 402–411, New York, New York, USA. ACM Press.

[72] Ghalwash, M. F., Ramljak, D., and Obradovic, Z. (2012). Early classification of multivariate time series using a hybrid HMM/SVM model. In *IEEE International Conference on Bioinformatics and Biomedicine*, pages 1–6.

[73] Giorgino, T. (2009). Computing and Visualizing Dynamic Time Warping Alignments in R: the dtw Package. *Journal of Statistical Software*, 31(7).

[74] Girolami, M. and Rogers, S. (2006). Variational Bayesian Multinomial Probit Regression with Gaussian Process Priors. *Neural Computation*, 18:1790–1817.

[75] Graepel, T., Herbrich, R., Bollmann-sdorra, P., and Obermayert, K. (1998). Classification on Pairwise Proximity Data. In *NIPS*, pages 438–444. The MIT Press.

[76] Guin, A. (2006). Travel Time Prediction using a Seasonal Autoregressive Integrated. In *Proceedings of the 2006 IEEE Intelligent Transportation Systems Conference*, volume 285, pages 493–498.

[77] Halati, A., Lieu, H., and Walker, S. (1997). CORSIM- Corridor Traffic Simulation Model. In *Traffic Congestion and Traffic Safety in the 21st Century: Challenges, Innovations, and Opportunities*, pages 570–576.

[78] Hall, E. and Petty, K. (2005). A Travel Time Prediction Algorithm Scalable to Freeway Networks with Many Nodes with Arbitrary Travel Routes. *Transportation Research Record: Journal of the Transportation Research Board*, 1935:147–153.

[79] Hall, F. L. (2001). Chapter 2: Traffic Stream Characteristics. In *Traffic Flow Theory. A State-of-the-Art Report.*

[80] Hardle, W., Müler, M., Sperlich, S., and Werwatz, A. (2004). *Nonparametric and Semiparametric Models.* Springer-Verlag, Berlin, Heidelberg.

[81] Hastie, T. and Tibshirani, R. (1993). Varying-Coefficient Models. *Journal of the Royal Statistical Society. Series B (Methodological)*, 55(4):757–796.

[82] Hatami, N. and Chira, C. (2013). Classifiers With a Reject Option for Early Time-Series Classification. In *IEEE Symposium on Computational Intelligence and Ensemble Learning (CIEL)*, pages 9–16.

[83] He, G., Duan, Y., Peng, R., Jing, X., Qian, T., and Wang, L. (2015). Early classification on multivariate time series. *Neurocomputing*, 149:777–787.

[84] He, H. and Garcia, E. a. (2009). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284.

[85] Hellinga, B. R. (2002). Improving freeway speed estimates from single-loop detectors. *Journal of Transportation Engineering*, 128(1):58–67.

[86] Hidalgo, J. I., Colmenar, J. M., Risco-Martin, J. L., Cuesta-Infante, A., Maqueda, E., Botella, M., and Rubio, J. A. (2014). Modeling glycemia in humans by means of Grammatical Evolution. *Applied Soft Computing Journal*, 20:40–53.

[87] Hoffmann, G. and Janko, J. (1990). Travel Time as a Basic Part of the LISB Guidance Strategy. In *Proceedings of the IEEE Road Traffic Control Conference*, pages 6–10.

[88] Hofleitner, A., Herring, R., Abbeel, P., and Bayen, A. (2012a). Learning the Dynamics of Arterial Traffic From Probe Data Using a Dynamic Bayesian Network. *IEEE Transactions on Intelligent Transportation Systems*, 13(4):1679–1693.

[89] Hofleitner, A., Herring, R., and Bayen, A. (2012b). Arterial travel time forecast with streaming data: A hybrid approach of flow modeling and machine learning. *Transportation Research Part B: Methodological*, 46(9):1097–1122.

[90] Hoh, B., Iwuchukwu, T., Jacobson, Q., Work, D., Bayen, A. M., Herring, R., Herrera, J.-C., Gruteser, M., Annavaram, M., and Ban, J. (2012). Enhancing Privacy and Accuracy in Probe Vehicle-Based Traffic Monitoring via Virtual Trip Lines. *IEEE Transactions on Mobile Computing*, 11(5):849–864.

[91] Hojati, A. T., Ferreira, L., and Charles, P. (2009). Assessing the major causes of travel time reliability on urban freeways. In *32nd Australasian Transport Research Forum (ATRF)*, pages 1–10.

[92] Hu, T.-Y., Tong, C.-C., Liao, T.-Y., and Ho, W.-M. (2012). Simulation-Assignment-Based Travel Time Prediction Model for Traffic Corridors. *IEEE Transactions on Intelligent Transportation Systems*, 13(3):1277–1286.

[93] Huang, L. and Barth, M. (2008). A Novel Loglinear Model for Freeway Travel Time Prediction. In *Proceedings of the 2008 IEEE Intelligent Transportation Systems Conference*, pages 210–215.

[94] Hubert, M. and Vandervieren, E. (2008). An adjusted boxplot for skewed distributions. *Computational Statistics & Data Analysis*, 52(12):5186–5201.

[95] Inrix (2006). Dynamic Predictive Traffic Service. Technical report.

[96] Ishak, S. and Al-Deek, H. (2003). Statistical Evaluation of I-4 Traffic Prediction System. *Transportation Research Record: Journal of the Transportation Research Board*, 1856:16–24.

[97] Jayakrishnan, R., Mahmassani, H. S., and Hu, T.-Y. (1994). An evaluation tool for advanced traffic information and management systems in urban networks. *Transportation Research Part C*, 2(3):129–147.

[98] Jenelius, E. and Koutsopoulos, H. N. (2013). Travel time estimation for urban road networks using low frequency probe vehicle data. *Transportation Research Part B: Methodological*, 53:64–81.

[99] Jiang, G., Gang, L., and Cai, Z. (2006). Impact of Probe Vehicles Sample Size on Link Travel Time Estimation. In *Proceedings of the 2006 IEEE Intelligent Transportation Systems Conference*, pages 505–509.

[100] Jiang, G. and Zhang, R. (2001). Travel-Time Prediction for Urban Arterial Road : A Case on China. In *Proceedings of the IEEE International Vehicle Electronics Conference*, number 142, pages 255–260.

[101] Jula, H., Dessouky, M., and Ioannou, P. a. (2008). Real-Time Estimation of Travel Times Along the Arcs and Arrival Times at the Nodes of Dynamic Stochastic Networks. *IEEE Transactions on Intelligent Transportation Systems*, 9(1):97–110.

[102] Kate, R. J. (2015). Using dynamic time warping distances as features for improved time series classification. *Data Mining and Knowledge Discovery*.

[103] Keogh, E., Zhu, Q., Hu, B., Y., H., Xi, X., and Wei, L. Ratanamahatana, C. A. (2011). The UCR Time Series Classification/Clustering Homepage.

[104] Kesting, A. and Treiber, M. (2008). Calculating travel times from reconstructed spatiotemporal traffic data. In *Proceedings of the 4th International Symposium on Networks for Mobility.*, pages 1–10.

[105] Khaleghi, B., Khamis, A., Karray, F. O., and Razavi, S. N. (2013). Multisensor data fusion: A review of the state-of-the-art. *Information Fusion*, 14(1):28–44.

[106] Khosravi, A., Mazloumi, E., Nahavandi, S., Member, S., Creighton, D., and Lint, J. W. C. H. V. (2011). Prediction Intervals to Account for Uncertainties in Travel Time Prediction. *IEEE Transactions on Intelligent Transportation Systems*, 12(2):537–547.

[107] Kogan, J. a. and Margoliash, D. (1998). Automated recognition of bird song elements from continuous recordings using dynamic time warping and hidden Markov models : A comparative study. *Journal of the Acoustical Society of America*, 103(4):2185–2196.

[108] Köhler, T. and Lorenz, D. (2005). A comparison of denoising methods for one dimensional time series. Technical report.

[109] Krishnan, R. and London, I. C. (2008). Short-term travel time prediction : An overview of methods and recurring themes. In *Proceedings of the Transportation Planning and Implementation Methodologies for Developing Countries Conference (TPMDC)*, pages 1–18.

[110] Kuchipudi, C. M. and Chien, S. I. J. (2003). Development of a Hybrid Model For Dynamic Travel Time Prediction. In *Transportation Research Board, the 82nd Annual Meeting*, number October 2002, pages 1–28.

[111] Kwon, J., Coifman, B., and Bickel, P. (2000). Day-to-Day Travel Time Trends and Travel Time Prediction from Loop Detector Data. *Trans-*

portation Research Record: Journal of the Transportation Research Board, 1717:120–129.

[112] Lama, N. and Girolami, M. (2014). vbmp: Variational Bayesian Multinomial Probit Regression. R package version 1.34.0.

[113] Lamedica, R., Prudenzi, A., Sforna, M., Caciotta, M., and Orsolini, V. (1996). A neural network based technique for short-term forecasting of anomalous load periods. *IEEE Transactions on Power Systems*, 11(4):1749–1756.

[114] Leduc, G. (2008). Road Traffic Data : Collection Methods and Applications. Technical report, Institute for Prospective Technological Studies, JRC European Commission.

[115] Lee, H., Chowdhury, N. K., and Chang, J. (2008). A New Travel Time Prediction Method for Intelligent Transportation Systems. In *Proceedings of the 12th international conference on Knowledge-Based Intelligent Information and Engineering Systems, Part I*, pages 473–483.

[116] Lee, S.-H., Viswanathan, M., and Yang, Y.-K. (2006). A Hybrid Soft Computing Approach to Link Travel Speed Estimation. In *Fuzzy Systems and Knowledge Discovery*, pages 794–802.

[117] Lee, Y. (2009). Freeway Travel Time Forecast Using Artifical Neural Networks With Cluster Method. In *Proceedings of the 12th International Conference on Information Fusion*, pages 1331–1338.

[118] Li, C.-S. and Chen, M.-C. (2014). A data mining based approach for travel time prediction in freeway with non-recurrent congestion. *Neurocomputing*, 133:74–83.

[119] Li, L., Chen, X., Li, Z., and Zhang, L. (2013). Freeway Travel-Time Estimation Based on Temporal -Spatial Queuing Model. *IEEE Transactions on Intelligent Transportation Systems*, PP(99):1–6.

[120] Li, R. and Rose, G. (2011). Incorporating uncertainty into short-term travel time predictions. *Transportation Research Part C*, 19(6):1006–1018.

[121] Li, Y., Fujimoto, R. M., and Hunter, M. P. (2009). Online Travel Time Prediction Based on Boosting. In *Proceedings of the 12th International IEEE Conference on Intelligent Transportation Systems*, pages 759–764.

[122] Li, Y. and McDonald, M. (2002). Link Travel Time Estimation Using Single GPS Equipped Probe Vehicle. In *The IEEE 5th International Conference on Intelligent Transportation Systems*, number September, pages 932–937.

[123] Liang, Z. and Ling-Xiang, Z. (2006). Link Travel Time Estimation Model Fusing Data and Stationary Detector Based on BP Neural Network. In *Proceeding of the International Conference on Communications, Circuits and Systems*, volume 00, pages 2146–2149.

[124] Liao, T. W. (2005). Clustering of time series data: a survey. *Pattern Recognition*, 38(11):1857–1874.

[125] Lim, S. and Lee, C. (2011). Data fusion algorithm improves travel time predictions. *IET Intelligent Transport Systems*, 5(4):302.

[126] Liu, H., van Lint, H., van Zuylen, H., and Zhang, K. (2006a). Two distinct ways of using Kalman filters to predict urban arterial travel time. In *Proceedings of the 2006 IEEE Intelligent Transportation Systems Conference*, pages 845–850. Ieee.

[127] Liu, H., van Zuylen, H. J., van Lint, H., Chen, Y., and Zhang, K. (2005). Prediction of Urban Travel Times with Intersection Delays. In *Proceedings of the 8th International IEEE Conference on Intelligent Transportation Systems*, pages 1062–1067.

[128] Liu, Y., Lin, P.-W., Lai, X., Chang, G.-L., and Marquess, A. (2006b). Developments and Applications of a Simulation-Based Online Travel Time Prediction System for Ocean City , Maryland. *Transportation Research Record: Journal of the Transportation Research Board*, 1959:92–104.

[129] Long, J., Gao, Z., and Szeto, W. Y. (2011). Discretised link travel time models based on cumulative flows: Formulations and properties. *Transportation Research Part B*, 45(1):232–254.

[130] Longfoot, J. (1991). An automatic Network Travel Time System-ANTSS. In *Proceedings of the 1991 Vehicle Navigation and Information Systems Conference. Vol. 2.*, pages 1053–1061.

[131] Lucas, D. E., Mirchandani, P. B., and Verma, N. (2004). Online travel time estimation without vehicle identification. *Transportation Research Record: Journal of the Transportation Research Board*, 1867:193–201.

[132] Ma, X. and Koutsopoulos, H. N. (2008). A New Online Travel Time Estimation Approach using Distorted Automatic Vehicle Identification Data. In *Proceeding of the 11th International IEEE Conference on Intelligent Transportation Systems*, pages 204–209.

[133] Martin-Merino, M. and Roman, J. (2006). A New SOM Algorithm for Electricity Load Forecasting. *Neural Information Processing*, pages 995–1003.

[134] Mendes-Moreira, J., Jorge, A. M., Freire de Sousa, J., and Soares, C. (2012). Comparing state-of-the-art regression methods for long term travel time prediction. *Intelligent Data Analysis*, 16:427–449.

[135] Meyer, D. and Buchta, C. (2013). proxy: Distance and Similarity Measures.

[136] Meyer, D. and Buchta, C. (2015). *proxy: Distance and Similarity Measures*. R package version 0.4-14.

[137] Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., Leisch, F., Chang, C.-C., and Lin, C.-C. (2014). e1071: Misc Functions of the Department of Statistics. R package version 1.6-4.

[138] Miska, M., Muller, T. H., and van Zuylen, H. (2005). Online Travel time Prediction with Real-time Microscopic Simulation. In *Proceedings of 84th Annual Meeting of the Transportation Research Board*, volume 4791, pages 1–17.

[139] Mobile, W. (2009). Waze.

[140] Monteil, J., Mehran, B., and Kuwahara, M. (2009). A Data Fusion Technique To Estimate Travel Time From Sparse AVI and Probe Data on

Urban Streets. In *Proceedings of Infrastructure Planning, Japan Society of Civil Engineers*.

[141] Montero, P. and Vilar, J. A. (2014). Time series clustering utilities.

[142] Mori, U., Mendiburu, A., and Lozano, J. A. (2015). *TSdist: Distance Measures for Time Series data.* R package version 2.1.

[143] Mouskos, K. C., Niver, E., and Pignataro, L. J. (1998). Transmit System Evaluation. Technical report, New Jersey Institute of Technology, Newark, New Jersey.

[144] Nagel, K. (2002). Cellular Automata Models for Transportation Applications. *Lecture Notes in Computer Science*, 2493:20–31.

[145] Nam, D. H. and Drew, D. R. (1996). Traffic Dynamics: Method for Estimating Freeway Travel Times in Real Time from Flow Measurements. *Journal of Transportation Engineering*, 122(3):185–191.

[146] Nanthawichit, C., Nakatsuji, T., and Suzuki, H. (2003). Application of Probe Vehicle Data for Real-Time Traffic State Estimation and Short-Term Travel Time Prediction on a Freeway. In *Transportation Research Board 2003 Annual Meeting*, volume 5890, pages 1–17.

[147] Ndoye, M., Totten, V. F., Krogmeier, J. V., and Bullock, D. M. (2011). Sensing and Signal Processing for Vehicle Reidentification and Travel Time Estimation. *IEEE Transactions on Intelligent Transportation Systems*, 12(1):119–131.

[148] Nikovski, D., Nishiuma, N., Goto, Y., and Kumazawa, H. (2005). Univariate short-term prediction of road travel times. In *Proceedings of the 2005 IEEE Intelligent Transportation Systems*, pages 1074–1079. Ieee.

[149] Oh, C., Ritchie, S., and Oh, J.-S. (2005). Exploring the Relationship Between Data Aggregation and Predictability to Provide Better Predictive Traffic Information. *Transportation Research Record*, 1935(1):28–36.

[150] Oh, J.-S., Jayakrishnan, R., and Recker, W. (2003). Section travel time estimation from point detection. In *82nd Annual Meeting of the Transportation Research Board*, pages 1–13.

[151] Ohba, Y., Koyama, T., and Shimada, S. (1997). Online-Learning Type Of Traveling Time Prediction Model in Expressway. In *Proceedings of the IEEE Conference on Intelligent Transportation System*, pages 350 – 355.

[152] Ohba, Y., Ueno, H., and Kuwahara, M. (1999). Travel Time Calculation Method for Expressway Using Toll Collection System Data. In *Proceedings of the 1999 IEEE/IEEJ/JSAI International Conference on Intelligent Transportation Systems*, pages 471–475.

[153] Palacharla, P. V. and Nelson, P. C. (1999). Application of fuzzy logic and neural networks for dynamic travel time estimation. *International Transactions in Operational Research*, 6:145–160.

[154] Papageorgiou, M., Papamichail, I., Messmer, A., and Wang, Y. (2010). Traffic Simulation with METANET. In Barcelo, J., editor, *Fundamentals of Traffic Simulation*, volume 145 of *International Series in Operations Research & Management Science*, pages 399–430. Springer New York, New York, NY.

[155] Park, B. B., Smith, B. L., Lee, J., Pampati, D., Ben-Akiva, M., and Balakrishnan, R. (2006). Evaluation of DynaMIT âĂŞ A Prototype Traffic Estimation and Prediction System. Technical Report June.

[156] Park, D. and Rilett, L. R. (1999). Forecasting Freeway Link Travel Times with a Multilayer Feedforward Neural Network. *Computer-Aided Civil and Infrastructure Engineering*, 14(5):357–367.

[157] Park, D., Rilett, L. R., Gajewski, B. J., Spiegelman, C. H., and Choi, C. (2009). Identifying optimal data aggregation interval sizes for link and corridor travel time estimation and forecasting. *Transportation*, 36(1):77–95.

[158] Park, D., Rilett, L. R., and Han, G. (1999). Spectral basis neural networks for real-time travel time forecasting. *ASCE Journal of Transportation Engineering*, 3(December):515–523.

[159] Park, T. and Lee, S. (2004). A Bayesian Approach for Estimating Link Travel Time on Urban Arterial Road Network. In *Computational Science and Its Applications âĂŞ ICCSA 2004*, pages 1017–1025.

[160] Parrish, N., Anderson, H. S., and Hsiao, D. Y. (2013). Classifying With Confidence From Incomplete Information. *Journal of Machine Learning Research*, 14:3561–3589.

[161] Petty, K. F., Bickel, P., Jiang, J., Ostland, M., Rice, J., Ritov, Y., and Schoenberg, F. (1998). Accurate estimation of travel times from single-loop detectors. *Accurate estimation of travel times from single-loop detectors*, 32(1):1–17.

[162] Pham, D. T. and Chan, A. B. (1998). Control chart pattern recognition using a new type of self-organizing neural network. *Proceedings of The Institution of Mechanical Engineers Part I-journal of Systems and Control Engineering*, 212(2):115–127.

[163] Pree, H., Herwig, B., Gruber, T., Sick, B., David, K., and Lukowicz, P. (2014). On general purpose time series similarity measures and their use as kernel functions in support vector machines. *Information Sciences*, 281:478–495.

[164] Pu, W., Lin, J. J., and Long, L. (2009). Real-Time Estimation of Urban Street Segment Travel Time Using Buses as Speed Probes. *Transportation Research Record: Journal of the Transportation Research Board*, 2129:81–89.

[165] Quinlan, R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers.

[166] Quiroga, C. A. and Bullock, D. (1998). Determination Of Sample Sizes For Travel Time Studies. *ITE Journal*, 68(8):92–98.

[167] R Core Team (2014). R: A Language and Environment for Statistical Computing.

[168] Rakthanmanon, T. and Keogh, E. (2013). Fast shapelets: A scalable algorithm for discovering time series shapelets. *Proceedings of the thirteenth SIAM conference ....*

[169] Ramezani, M. and Geroliminis, N. (2012). On the estimation of arterial route travel time distribution with Markov chains. *Transportation Research Part B: Methodological*, 46(10):1576–1590.

[170] Rasmussen, C. E. . and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.

[171] Read, J. (2012). MEKA: A Multi-label Extension to WEKA.

[172] Read, J., Pfahringer, B., Holmes, G., and Frank, E. (2011). Classifier chains for multi-label classification. *Machine Learning*, 85:333–359.

[173] Rehfeld, K., Marwan, N., Heitzig, J., and Kurths, J. (2011). Comparison of correlation analysis techniques for irregularly sampled time series. *Nonlinear Processes in Geophysics*, 18(3):389–404.

[174] Reiss, A. and Stricker, D. (2011). Towards global aerobic activity monitoring. *Proceedings of the 4th International Conference on PErvasive Technologies Related to Assistive Environments - PETRA '11*, (July):1.

[175] Reiss, A., Weber, M., and Stricker, D. (2011). Exploring and extending the boundaries of physical activity recognition. *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, pages 46–50.

[176] Rice, J. and van Zwet, E. (2001). A Simple and Effective Method for Predicting Travel Times on Freeways. In *IEEE Intelligent Transportation Systems Conference Proceedings*, number T 6, pages 227–232.

[177] Ripley, B. and Venables, W. (2014). nnet: Feed-forward Neural Networks and Multinomial Log-Linear Models. R package version 7.3-8.

[178] Rodríguez, J. D., Pérez, A., and Lozano, J. a. (2013). A general framework for the statistical analysis of the sources of variance for classification error estimators. *Pattern Recognition*, 46(3):855–864.

[179] Rousseeuw, P., Croux, C., Todorov, V., Ruckstuhl, A., Salibian-Barrera, M., Verbeke, T., Koller, M., and Maechler, M. (2013). robustbase: Basic Robust Statistics.

[180] Ruppert, D., Wand, M. P., and Carroll, R. J. (2003). *Semiparametric Regression (Volume 12 of the Cambridge Series on Statistical and Probabilistic Mathematics)*. Cambridge University Press, New York.

[181] Ryan, J. A. (2013). quantmod: Quantitative financial modelling framework. r package version 0.4-0.

[182] Ryan, J. A. and Ulrich, J. M. (2013). xts: eXtensible Time Series.

[183] Sakoe, H. and Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. *Acoustics, Speech, and Signal Processing [see also {IEEE} Transactions on Signal Processing], {IEEE} Transactions on*, 26(1):43–49.

[184] Schmitt, E. J. and Jula, H. (2007). On the Limitations of Linear Models in Predicting Travel Times. In *Proceedings of the 2007 IEEE Intelligent Transportation Systems Conference*, pages 830–835.

[185] Shen, L. and Hadi, M. (2013). Practical approach for travel time estimation from point traffic detector data. *Journal of Advanced Transportation*, 47:526–535.

[186] Shuo Li, P., Zhu, K., van Gelder, B., Nagle, J., and Tuttle, C. (2002). Reconsideration of Sample Size Requirements for Field Traffic Data Collection Using GPS Devices. *Transportation Research Record: Journal of the Transportation Research Board*, 1804:17–22.

[187] Simroth, A. and Zähle, H. (2011). Travel Time Prediction Using Floating Car Data Applied to Logistics Planning. *IEEE Transactions on Intelligent Transportation System*, 12(1):243–253.

[188] Sisiopiku, V. P. and Rouphail, N. M. (1994). Toward The Use Of Detector Output For Arterial Link Travel Time Estimation: A Literature Review. *Transportation Research Record: Journal of the Transportation Research Board*, 1457:158–165.

[189] Soriguera, F., Abeijon, D., Rosas, D., Thorson, L., and Robuste, F. (2007). Travel Time Estimation from Multiple Data Sources. In *Proceedings of the 11th World Conference on Transport Research*, pages 1–36.

[190] Soriguera, F. and Robusté, F. (2011a). Estimation of traffic stream space mean speed from time aggregations of double loop detector data. *Transportation Research Part C: Emerging Technologies*, 19(1):115–129.

[191] Soriguera, F. and Robusté, F. (2011b). Highway travel time accurate measurement and short-term prediction using multiple data sources. *Transportmetrica A: Transport Science.*, 7(1):85–109.

[192] Soriguera, F. and Robusté, F. (2011c). Requiem for Freeway Travel Time Estimation Methods Based on Blind Speed Interpolations Between Point Measurements. *IEEE Transactions on Intelligent Transportation Systems*, 12(1):291–297.

[193] Soriguera, F., Rosas, D., and Robusté, F. (2010). Travel time measurement in closed toll highways. *Transportation Research Part B*, 44(10):1242–1267.

[194] Spodick, D. H. (2003). Acute cardiac tamponade. *New England Journal of Medicine*, 349(7):684–690.

[195] S.R.I (1998). TransGuide model deployment report. Technical report, Southwest Research Institute, San Antonio, Texas.

[196] Srinivasan, K. K. and Jovanis, P. P. (1996). Determination of number of probe vehicles required for reliable travel time measurement in urban network. *Transportation Research Record: Journal of the Transportation Research Board*, 1537:15–22.

[197] Stathopoulos, V. and Jones, K. E. (2014). Bat Call Identification with Gaussian Process Multinomial Probit Regression and a Dynamic Time Warping Kernel. In *Proceedings of the 17th International Conference on Artifcial Intelligence and Statistics*, volume 33, pages 913–921.

[198] Sun, C., Ritchie, S. G., Tsai, K., and Jayakrishnan, R. (1999). Use of vehicle signature analysis and lexicographic optimization for vehicle reidentication on freeways. *Transportation Research Part C*, 7(4):167–185.

[199] Sun, H., Liu, H. X., Xiao, H., He, R. R., and Ran, B. (2003). Short-Term Traffic Forecasting Using the Local Linear Regression Model. In

Transportation Research Board 2003 Annual Meeting, number 608, pages 1–30.

[200] Sun, L., Yang, J., and Mahmassani, H. (2008). Travel time estimation based on piecewise truncated quadratic speed trajectory. *Transportation Research Part A*, 42:173–186.

[201] Tam, M. L. and Lam, W. H. (2008). Using automatic vehicle identification data for travel time estimation in Hong-Kong. *Transportmetrica A: Transport Science.*, 4(3):179–194.

[202] Tam, M. L. and Lam, W. H. (2011). Application of automatic vehicle identification technology for real-time journey time estimation. *Information Fusion*, 12(1):11–19.

[203] Taylor, N. B. (2003). CONTRAM (CONtinuous TRaffic Assignment Model). *Networks and Spatial Economics*, 3:297–322.

[204] Therneau, T., Atkinson, B., and Ripley, B. (2014). rpart: Recursive Partitioning and Regression Trees. R package version 4.1-8.

[205] Tiesyte, D. and Jensen, C. S. (2008). Similarity-based prediction of travel times for vehicles traveling on known routes. *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems.*

[206] Torgo, L. (2010). *Data Mining with R, learning with case studies.* Chapman and Hall.

[207] Tormene, P., Giorgino, T., Quaglini, S., and Stefanelli, M. (2009). Matching incomplete time series with dynamic time warping: An algorithm and an application to post-stroke rehabilitation. *Artificial Intelligence in Medicine*, 45.

[208] Tsoumakas, G. and Vlahavas, I. (2007). Random k-Labelsets: An Ensemble Method for Multilabel Classification. In *Lecture Notes in Artificial Intelligence*, volume 4701, pages 406–417. Springer Berlin.

[209] Tu, H., van Lint, H., and van Zuylen, H. (2008). The Effects of Traffic Accidents on Travel Time Reliability. In *Proceedings of the 11th International IEEE Conference on Intelligent Transportation Systems*, pages 79–84. Ieee.

[210] Tu, H., van Lint, J. W. C., and van Zuylen, H. J. (2007). The Impact of Adverse Weather on Travel Time Variability of Freeway Corridors. In *Transportation Research Board 86th Annual Meeting*, pages 1–15.

[211] Turner, S. M., Eisele, W., Benz, R., and Holdener, D. (1998). Travel Time Data Collection Handbook. Technical Report March 1998, Federal Highway Administration, Report FHWA-PL-98-035.

[212] Turner, S. M. and Holdener, D. J. (1995). Probe Vehicle Sample Sizes for Real-Time Information: The Houston Experience. In *Proceedings of the Vehicle Navigation and Information Systems Conference*, pages 3–10.

[213] Ulanova, L., Begum, N., and Keogh, E. (2015). Scalable Clustering of Time Series with U-Shapelets. In *SIAM International Conference on Data Mining (SDM 2015).*

[214] van Aerde, M. and Rakha, H. (2010). INTEGRATION Rel. 2.30 for Windows - User's Guide, Volume I, Volume II. Technical report, M. Van Aerde & Associates, Ltd., Blacksburg, Virginia.

[215] van Arem, B., van der Vlist, M. J. M., Muste, M. R., and Smulders, S. A. (1997). Travel time estimation in the GERDIEN project. *International Journal of Forecasting*, 13:73–85.

[216] van Hinsbergen, C. P. I. J., Hegyi, A., van Lint, J. W. C., and van Zuylen, H. J. (2011). Bayesian neural networks for the prediction of stochastic travel times in urban networks. *Intelligent Transport Systems*, (November 2009):259–265.

[217] van Hinsbergen, C. P. I. J., van Lint, J. W. C., and Sanders, F. (2007). Short Term Traffic Prediction Models. In *Proceedings of the 14th World Congress on Intelligent Transport Systems (ITS)*, pages 1–18.

[218] van Hinsbergen, C. P. I. J., van Lint, J. W. C., and van Zuylen, H. J. (2009). Bayesian committee of neural networks to predict travel times with confidence intervals. *Transportation Research Part C*, 17(5):498–509.

[219] van Lint, J. W. C. (2004). *Reliable Travel Time Prediction for Freeways*. PhD thesis, Delft University of Technology.

[220] van Lint, J. W. C. (2008). Online Learning Solutions for Freeway Travel Time Prediction. *IEEE Transactions on Intelligent Transportation Systems*, 9(1):38–47.

[221] van Lint, J. W. C. and van der Zijpp, N. (2003). An Improved Travel-time Estimation Algorithm using Dual Loop Detectors. *Transportation Research Record: Journal of the Transportation Research Board*, 1855:41–48.

[222] Vanajakshi, L., Subramanian, S. C., and Sivanandan, R. (2009). Travel time prediction under heterogeneous traffic conditions using global positioning system data from buses. *IET Intelligent Transport Systems*, 3(1):1–9.

[223] Vanajakshi, L. D. (2004). *Estimation and Prediction of Travel Time From Loop Detector Data For Intelligent Transportation Systems Applications*. PhD thesis, Texas A\&M University.

[224] Vlahogianni, E. I., Golias, J. C., and Karlaftis, M. G. (2004). Short-term Traffic Forecasting : Overview of Objectives and Methods. *Transport Reviews*, 24(5):533–557.

[225] Wagner, S. and Wagner, D. (2007a). Comparing Clusterings - An Overview. Technical Report 001907.

[226] Wagner, S. and Wagner, D. (2007b). Comparing Clusterings - An Overview. Technical Report 2006-04, Universität Karlsruhe (TH).

[227] Wang, X. and Khattak, A. (2013). Role of travel information in supporting travel decision adaption: exploring spatial patterns. *Transportmetrica A: Transport Science.*, 9(4):316–334.

[228] Wang, X., Mueen, A., Ding, H., Trajcevski, G., Scheuermann, P., and Keogh, E. (2012). Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, 26(2):275–309.

[229] Wang, X., Smith, K., and Hyndman, R. (2006). Characteristic-Based Clustering for Time Series Data. *Data Mining and Knowledge Discovery*, 13(3):335–364.

[230] Wei, C.-H. and Lee, Y. (2007). Development of Freeway Travel Time Forecasting Models by Integrating Different Sources of Traffic Data. *IEEE Transactions on vehicular technology*, 56(6):3682–3694.

[231] Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics*, 1(6):80–83.

[232] Wu, C.-H., Ho, J.-M., and Lee, D. T. (2004). Travel-Time Prediction With Support Vector Regression. *IEEE Transactions on Intelligent Transportation Systems*, 5(4):276–281.

[233] Wunderlich, K. E., Kaufman, D. E., and Smith, R. L. (2000). Link Travel Time Prediction for Decentralized Route Guidance Architectures. *IEEE Transactions On Intelligent Transportation Systems*, 1(1):4 – 14.

[234] Xeno-canto Foundation (2005). xeno-canto: Compartiendo cantos de aves de todo el mundo.

[235] Xia, J., Chen, M., and Qian, Z. (2010). Predicting Freeway Travel Time Under Incident Conditions. *Transportation Research Record: Journal of the Transportation Research Board*, 2178:58–66.

[236] Xing, Z., Pei, J., and Keogh, E. (2010). A brief survey on sequence classification. *ACM SIGKDD Explorations Newsletter*, 12(1):40.

[237] Xing, Z., Pei, J., and Yu, P. S. (2011a). Early classification on time series. *Knowledge and Information Systems*, 31(1):105–127.

[238] Xing, Z., Yu, P. S., and Wang, K. (2011b). Extracting Interpretable Features for Early Classification on Time Series. In *Proceedings of the Eleventh {SIAM} International Conference on Data Mining*, pages 247–258.

[239] Yang, J.-S. (2005a). A Study of Travel Time Modeling via Time Series Analysis. In *Proceedings of the 2005 IEEE Conference on Control Applications*, pages 855–860.

[240] Yang, J.-S. (2005b). Travel time prediction using the GPS test vehicle and Kalman filtering techniques. In *Proceedings of the 2005 American Control Conference*, pages 2128–2133. Ieee.

[241] Ye, L. and Keogh, E. (2009). Time Series Shapelets : A New Primitive for Data Mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 947–956.

[242] Ye, Q., Szeto, W. Y., and Wong, S. C. (2012). Short-Term Traffic Speed Forecasting Based on Data Recorded at Irregular Intervals. *IEEE Transactions on Intelligent Transportation Systems*, 13(4):1727–1737.

[243] Yeon, J., Elefteriadou, L., and Lawphongpanich, S. (2008). Travel time estimation on a freeway using Discrete Time Markov Chains. *Transportation Research Part B*, 42:325–338.

[244] Yildirimoglu, M. and Geroliminis, N. (2013). Experienced travel time prediction for congested freeways. *Transportation Research Part B: Methodological*, 53:45–63.

[245] Yildirimoglu, M. and Ozbay, K. (2012). Comparative Evaluation of Probe-Based Travel Time Prediction Techniques Under Varying Traffic Conditions. In *Transportation Research Board 91st Annual Meeting*.

[246] Yim, Y. (2003). The State of Cellular Probes. Technical Report July, Institute of Transportation Studies, University of California., Berkeley.

[247] You, J. and ÂĎKim, T. J. (2000). Development and evaluation of a hybrid travel time forecasting model. *Transportation Research Part C*, 8(1-6):231–256.

[248] Yu, Y. J. and Cho, M.-G. (2008). A Short-Term Prediction Model for Forecasting Traffic Information Using Bayesian Network. In *Third 2008 International Conference on Convergence and Hybrid Information Technology*, pages 242–247.

[249] Zeileis, A. and Grothendieck, G. (2005). zoo: S3 Infrastructure for Regular and Irregular Time Series. *Journal of Statistical Software*, 14(6):1–27.

[250] Zhang, M.-L. and Zhou, Z.-H. (2013). A Review On Multi-Label Learning Algorithms. *IEEE Transactions on Knowledge and Data Engineering*.

[251] Zhang, W. (2006). *Freeway Travel Time Estimation Based on Spot Speed Measurements*. PhD thesis, Virginia Polytechnic Institute and State University.

[252] Zhang, X. and Rice, J. A. (2003). Short-term travel time prediction. *Transportation Research Part C*, 11(3-4):187–210.

[253] Zheng, F. (2011). *Modelling Urban Travel Times*. PhD thesis, Netherlands TRAIL Research School.

[254] Zhu, G. and Wang, X. (2009). Study on Route Travel Time Prediction Based on RBF Neural Network. In *Proceedings of the 2009 First International Workshop on Education Technology and Computer Science*, pages 1118–1122. Ieee.

[255] Zhu, T., Kong, X., and Lv, W. (2009). Large-scale Travel Time Prediction for Urban Arterial Roads Based on Kalman Filter. In *Proceedings of the IEEE International Conference on Computational Intelligence and Software Engineering*, pages 1–5.

[256] Zou, N., Wang, J., and Chang, G.-L. (2008). A Reliable Hybrid Prediction Model for Real-time Travel Time Prediction with Widely Spaced Detectors. In *Proceedings of the 11th International IEEE Conference on Intelligent Transportation Systems*, pages 91–96.