eman ta zabal zazu

Universidad del País Vasco Euskal Herriko Unibertsitatea

PhD Thesis in Mathematics & Statistics

Contributions to Time Series Classification: Meta-Learning and Explainability

by

Amaia Abanda

Supervised by Usue Mori and Jose A. Lozano

Donostia - San Sebastián, November 2021

PhD Thesis in Mathematics & Statistics

Contributions to Time Series Classification: Meta-Learning and Explainability

by

Amaia Abanda

Supervised by Usue Mori and Jose A. Lozano

Dissertation submitted to the University of the Basque Country (UPV/EHU) as partial fulfillment of the requirements for the PhD degree in Mathematics and Statistics

Donostia - San Sebastián, November 2021

To my family

This research was carried out at the Basque Center for Applied Mathematics (BCAM) within the Data Science - Machine Learning group and was supported by by the Basque Government through the BERC 2018-2021 program and by Spanish Ministry of Economy and Competitiveness MINECO through BCAM Severo Ochoa excellence accreditation SEV-2017-0718, as well as through project TIN2017-82626-R funded by (AEI/FEDER, UE) and acronym GECECPAST. In addition, by the Research Group IT1244-19 programs (Basque Government), PID2019-104966GB-I00 (Spanish Ministry of Economy, Industry and Competitiveness) and Elkartek project 3KIA (KK2020/00049). A. Abanda is also supported by the Grant BES-2016-076890. The support of MINECO Grants is acknowledged for the three months visit at University of Rennes, Rennes, France.

Acknowledgments

I would like to express my appreciation to my supervisors, Usue Mori and Jose A. Lozano, for their patient and guidance during my Ph.D. My thesis would not have been done without their great support. I also would like to specially thank Simon Malinowski for giving me the opportunity to collaborate as a visiting researcher at INRIA, University of Rennes. I have learn a lot and I spent there a great time. I wish to thank all the members of the research group for their hospitality, the lunch time talks and after-work beers.

I feel also very grateful for the lab-mates with whom I have spent so much time both from the ISG group and from BCAM. Thanks for all the eternal coffee breaks, for rising me up in the hard moments and making my Ph.D. period easier. I do not want to forget the inspiring and empowering women I have met during my Ph.D. and, in particular, the R-Ladies group we founded at BCAM. The world is a bit better place thanks to women like you.

Lastly, I would like to express my deepest gratitude to my family and friends, which have always been my support and greatest treasure. I am very proud of you.

Contents

Introduction 1
1.1 Time series classification 1
1.1.1 Supervised classification with time series data 3
1.1.2 Time series classifiers 6
1.2 Meta-learning 10
1.2.1 Meta-learning for classifier recommendation 11
1.2.2 Meta-learning for time series 13
1.3 Explainability 13
1.3.1 Explanation methods for time series 15
1.4 Data and software sources 17
1.5 Overview of the dissertation 17

A Review on Distance based Time Series Classification 21
2.1 Introduction 21
2.2 A taxonomy of distance based time series classification 22
2.2.1 k-Nearest Neighbour 23
2.2.2 Distance features 24
2.2.3 Distance kernels 33
2.3 Computational cost 45
2.4 Discussion 48

Time Series Classifier Recommendation by a Meta-Learning Approach 53
3.1 Introduction 53
3.2 Time Series Classifier Recommendation 54
3.2.1 Meta-Attributes 54
3.2.2 Meta-target 58
3.3 Hierarchical inference of meta-targets 60
3.4 Experimentation 62
3.4.1 Experimental set-up 62
3.4.2 Analysis of the landmarkers 64

3.4.3 TSCR 67
3.4.4 Hierarchical inference of meta-targets 71
3.5 Conclusion and future work 72

Ad-Hoc Explanation for Time Series Classification 74
4.1 Introduction 74
4.2 Time series Transformations 75
4.2.1 Warp 75
4.2.2 Scale 77
4.2.3 Noise 78
4.2.4 Slice 79
4.3 Time Series Classification Explanation Method 81
4.3.1 High-level explanation 81
4.3.2 Low-level explanation 83
4.4 Experimentation 85
4.4.1 Set-up 85
4.4.2 Qualitative evaluation: case of study 86
4.4.3 Quantitative evaluation on UCR 88
4.5 Conclusions and Future Work 93

General Conclusions and Future Work 96
5.1 Conclusions 96
5.2 Future Works 98
5.3 Main Achievements 99

Appendix 101
A.1 Landmarkers: computation time of subsample landmarkers in the largest datasets from the UCR repository 101
A.2 Landmarkers: computation times 102
A.3 Landmarkers: Correlation between the landmarkers and the corresponding original classifiers 103
A.4 Landmarkers: Accuracies of classifiers and landmarkers by dataset 104

References 108

1 Introduction

In this dissertation, we present three contributions to the field of Time Series Classification (TSC). With the aim of providing the background needed to understand the presented works, this chapter includes an introduction to several topics. The first contribution consists of reviewing and proposing a taxonomy of distance based time series classifiers, so, an introduction to TSC is provided in Section 1.1. In the second contribution, a time series classifier recommendation system is presented, for which we follow a meta-learning approach. As a background for this, the basics of meta-learning are introduced in Section 1.2. In the third contribution, we propose a method for explaining the predictions of time series classifiers. In order to provide the fundamental notions on classifier explanations to the reader, an introduction to the field of explainability is presented in Section 1.3. In Section 1.4, we briefly present the data and software sources employed throughout the dissertation and, lastly, we present an overview that summarizes the dissertation in Section 1.5.

1.1 Time series classification

Time series data are being generated every day in a wide range of application domains, such as bioinformatics, financial fields, engineering, etc [1]. They are a particular type of data due to their *temporal nature*; a time series is an ordered sequence of observations which are usually taken through time, but may also be ordered with respect to another aspect, such as space. More formally, a time series can be defined as follows:

Definition 1. *A time series $T = (t_1, \ldots, t_l)$ of length l is a ordered sequence of pairs (i, t_i), where $i = 1, \ldots, l$ refers to the timestamp and t_i represents the value of the time series at timestamp i.*

The timestamps $i = 1, \ldots, l$ conform a sequence of positive and ascending discrete values, which, in most of the cases, are equally spaced, but may be also irregularly spaced. The time series can be univariate, when t_i is a single value, or multivariate, when $\mathbf{t_i}$ is a vector. At the same time, depending on the problem, the values t_i can be either discrete or continuous. In this dissertation, we will focus on univariate equally spaced and continuous time series.

One of the main advantages of time series is that, even if they are high dimensional data, we are able to easily visualize them. For instance, Figure 1.1 shows two time series that belong to the ECG200 dataset [2]. The time series in this dataset represent the electrical activity recorded during one heartbeat and they are divided into two classes: normal heartbeat and myocardial infarction. One time series from each class is shown in the figure and, as it can be seen, their shapes are visually different.

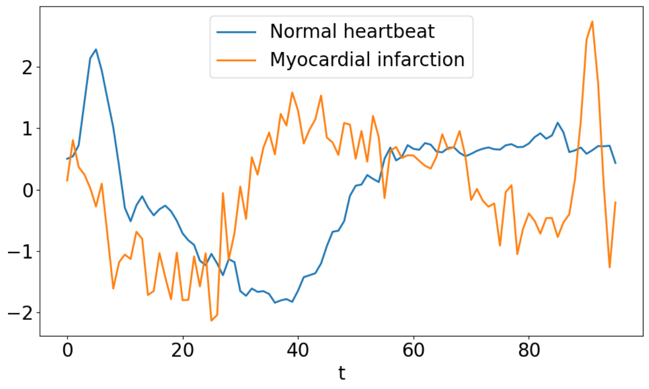


Fig. 1.1: Two time series that belong to the ECG200 dataset [2].

With the growing amount of recorded data, the interest in researching this particular type of data has also increased, giving rise to a vast amount of new methods for forecasting, representing, indexing, clustering, and classifying time series, among other tasks [3, 4]. Most of these tasks arise naturally from real world problems or challenges; in a time series that represents the evolution of the price of a product, for example, predicting future prices is a time series forecasting problem [5]. In another scenario, a problem may comprise several time series that represent, for instance, the daily temperature in different locations. In this case, the goal may be to group the different locations depending on the temperature pattern [6], which is a time series clustering task.

In another scenario, a problem may consist of a set of time series that represent the daily variation in brightness of different astronomical objects, such as stars or meteors, captured by a telescope [7]. In this context, the goal may be to predict whether a new time series corresponds to a star, a meteor or other type of astronomical object. In this case, the problem is conformed by supervised (labeled) time series and the task is called supervised classification of time series. This dissertation focuses on this learning problem, which we introduce in more detail in the following section.

1.1.1 Supervised classification with time series data

Supervised classification can be defined as the task of finding a mapping $f : \mathbf{X} \rightarrow \mathbf{y}$ between the space of input and the space of output, based on some example input-output pairs. In standard supervised classification, the instances are described by feature vectors, i.e., order-free vectors that contain the values of a set of qualitative or quantitative variables. Due to the popularity and usefulness of this task, in the past years, many variants of supervised classification have been proposed, TSC among them. Since standard classifiers do not take the order of the variables that conform the instances into account, the main challenge of TSC consists of dealing with this order [8]. With the purpose of formally defining the task of TSC, we first define a dataset of supervised time series:

Definition 2. *A dataset of supervised time series is a collection of n instance-label pairs $D = \{(T_1, y_1), \ldots, (T_n, y_n)\}$, where y_i is the label associated to the time series T_i. The labels y_i can take k possible values, where k is the number of classes.*

Supervised time series datasets may contain time series that have the same length but also time series of varying lengths. In this way, if $\mathbf{T}$ is the space of possible time series (input) and $\mathbf{y}$ is the space of possible labels (output), TSC can be defined as follows [4, 9]:

Definition 3. *TSC is the task of finding a function $f : \boldsymbol{T} \rightarrow \boldsymbol{y}$ that maps the space of time series to the the space of labels, based on the example pairs in a training set D.*

As can be seen in Figure 1.2, a classifier is learned by applying a learning algorithm in the supervised time series dataset. Then, the trained classifier is used to classify (predict the class of) new unlabeled time series.

In standard classification problems, in general, the class membership of an instance is related to (possibly complex) combinations of the features. In TSC, on the contrary, the class membership of a time series is generally related to specific characteristics of the time series, such as shapes, locations of patterns or the scale of a pattern, among others. Hereafter, we will refer to these aspects that allow to distinguish between the time series of different classes as the discriminatory characteristics of a TSC problem.

With the aim of illustrating the concept of discriminatory characteristics, some example datasets are presented in the following. As a first example, Figure 1.3 shows some synthetic time series extracted from the CBF dataset [2].

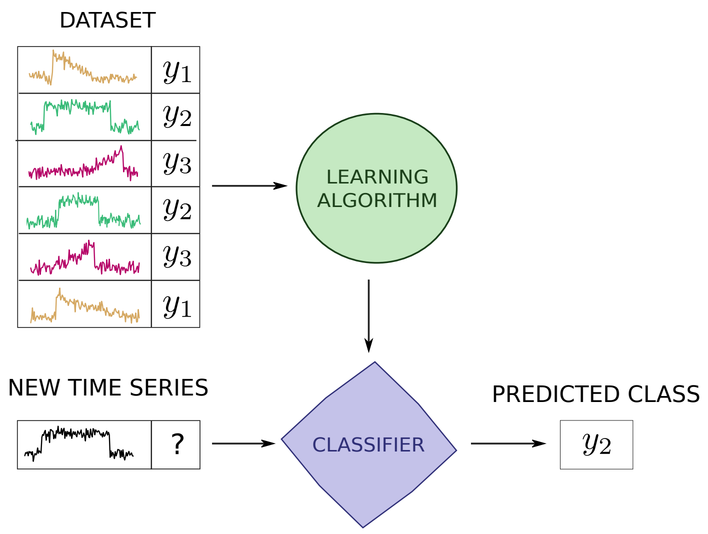


Fig. 1.2: Scheme of TSC.

This dataset contains three classes: *Cylinder*, *Bell* and *Funnel*. As it can be seen, there are three shapes or patterns and each pattern determines a class. This problem may seem simple since the classes are visually distinguishable. However, making a classifier understand our visual perception is not straightforward. For example, the time series that belong to the *Cylinder* class share the same pattern, but it can be seen that the width and location of the pattern are not the same in the two time series. More specifically, these patterns have two types of time distortions: warp, which modifies the width of a pattern, and shift, which modifies the position of a pattern [4]. In this case, the discriminatory characteristic is the shape of a time series, regardless of the width and position of this shape.

The discriminatory characteristics of a problem, however, may not be so evident in many cases. As a second example, we introduce the GunPoint dataset [2], which is a human activity recognition problem set up in a simulated controlled environment. In this dataset, the time series represents the x-axis of a movement made by a person doing a certain exercise, captured by a sensor. This movement consists of rising a gun from the hip, pointing it at a target, and bringing the gun back to the hip. Some of the people raise a real gun, while others are asked to replicate the movement without a gun, i.e., rising and pointing at a target with a finger. As such, this dataset contains two classes, *Gun* and *Point*, and the goal is to discriminate between these two types of movements. Figure 1.4 shows three example time series from each class. In this case, the main pattern of the time series is very similar in both classes, so it is not straightforward to visually discriminate between the

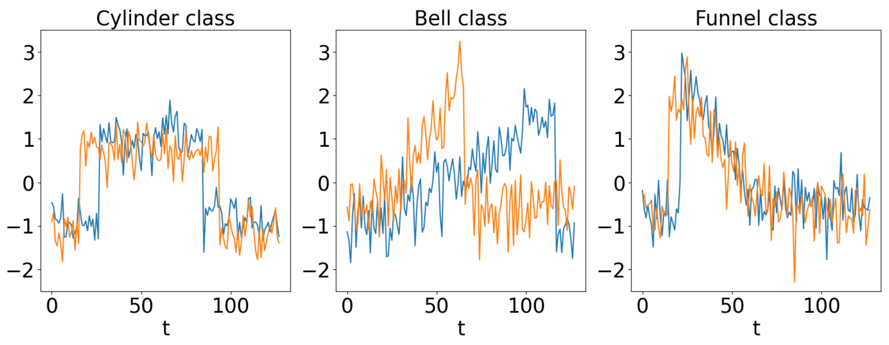


Fig. 1.3: Time series that belong to CBF dataset from the UCR repository [2].

classes. It can be seen that the height or scale of the time series could be a possible characteristic to discriminate between the classes, since, in the examples displayed in Figure 1.4, the time series from the *Gun* class are slightly lower than those in the *Point* class. At the same time, the part in which the person rises the gun (the interval from $t = 10$ to $t = 50$, approximately) seems to have a little step in the time series from the *Gun* class, which could also be used to identify the time series that belong to this class.

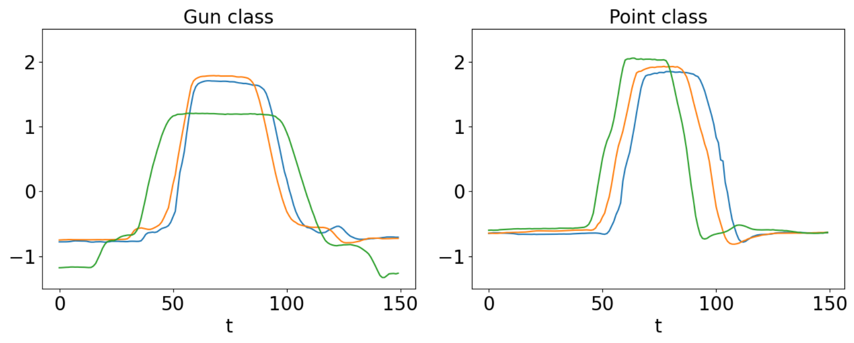


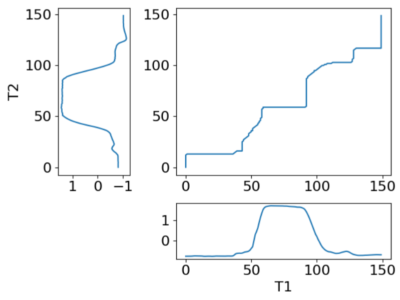
Fig. 1.4: Time series that belong to GunPoint dataset from the UCR repository [2].

In TSC, since these discriminatory characteristics enable to differentiate between the classes, time series classifiers focus on obtaining information about them from the training set [8]. In the next section, a background on time series classifiers is provided, in which two taxonomies of the different methods are presented.

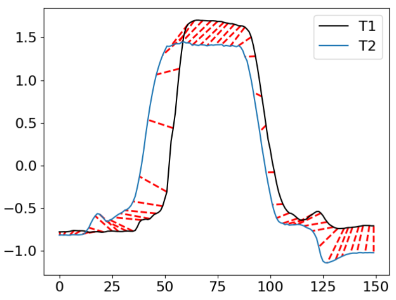
1.1.2 Time series classifiers

The first attempts to deal with TSC focused on proposing specific distance measures for time series and exploiting them within basic distance based classification methods, such as the k-Nearest Neighbor (k-NN) classifier. The Dynamic Time Warping (DTW) [10] distance is one of the first distance measures of this sort. The DTW is an elastic measure that makes non-linear alignments between series in order to deal with distortions in the time axis, such as time warping and shifting.

Given two time series $T1$ and $T2$, the underlying idea of the DTW distance is to find the optimal alignment between them by allowing flexible alignments in the time axis. For this, as shown in Figure 1.5a, the point-wise (usually Euclidean) distance matrix between $T1$ and $T2$ is computed, and the optimal alignment is defined as the path within this matrix that minimizes the cumulative distance under certain conditions [10]. The optimal path is found by using a recursive dynamic programming formulation. The optimal alignment between the two time series is shown the example in Figure 1.5b.



(a) The optimal path in the distance matrix.



(b) The optimal alignment between the two time series.

Fig. 1.5: Visualization of the DTW alignment between two time series from the GunPoint dataset. These plots have been computed employing the *dtw* package [11].

For a long time, the k-NN classifier together with the DTW was an unbeatable classifier in the TSC community [12]. Given the success of the DTW, several new distance measures were proposed afterward, which are computationally faster or handle other characteristics of the time series. Some popular examples of these distance measures include the Longest Common Subsequence (LCSS) [13], Time Warp Edit (TWE) [14] or Move-Split-Merge (MSM) [15] measures, but the list of existing distance measures nowadays is extremely long [8].

These distance measures are used within a distance based classification method, which, in the TSC community, is the most popular approach. Initially, those distance measures were employed exclusively with the k-NN method, but new distance based classification methods have also been arisen to exploit their strengths. Kate *et al.* [16], for instance, propose a highly competitive method in which the time series are represented by their DTW distances with respect to the time series in the training set. Another example consists of using the distance measures to construct kernels for time series [17], and employing them within kernel based methods, such as the well known Support Vector Machine (SVM) [18].

In addition to distance based methods, new approaches for classifying time series have also been proposed in the past few years. As such, the heterogeneity of classifiers has considerably increased, and this has led to many different ways to categorize the existing methods. In a first attempt to organize the existing methods, Xing *et al.* proposed to divide the time series classifiers into three groups, depending on how the time series are treated within the classifier:

- **Feature based classifiers**: in these methods, the time series are transformed into feature vectors and then classified by a conventional classifier such as a neural network or a decision tree. Some methods for feature extraction include spectral methods such as the Discrete Fourier Transform (DFT) [19] or the Discrete Wavelet Transform (DWT) [20].

- **Model based classifiers**: these classifiers assume that all the time series in a class are generated by the same underlying model, and thus a new series is assigned with the class of the model that fits the best. Some model based approaches are formed using auto-regressive models [21, 22] or hidden Markov models [23], among others.

- **Distance based classifiers**: distance based methods are those in which a (dis)similarity measure between series is defined, and then these distances are introduced in some manner within distance based classification methods such as the k-NN or SVMs. Some examples of these methods include the already mentioned 1-NN-DTW or 1-NN-LCSS.

Recently, Bagnall *et al.* [8] proposed a categorization that groups the existing classifiers depending on the type of discriminatory characteristics each approach focuses on. They divide the existing classifiers into five main categories: whole series classifiers, interval classifiers, shapelet based, dictionary based classifiers, and combinations. In the following, a brief description of the classifiers in each category is provided, including an overview of the discriminatory characteristics each type of classifier focuses on.

- **Whole series**: the classifiers in this group employ specific distance measures to compare whole time series. They are especially suitable for problems in which the discriminatory characteristic of the series is the shape of the time series, that appear over the whole series. For example, in the dataset shown in Figure 1.6, this type of classifier should perform especially well because the discriminatory characteristic is the shape of the time series, even if it is warped and shifted. Some popular examples of these classifiers are the already mentioned 1-NN-DTW or 1-NN-MSM [15].

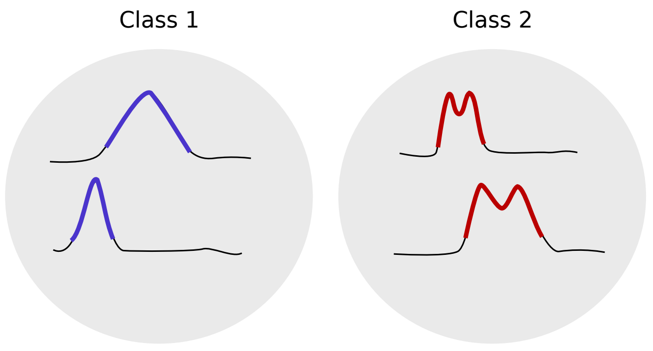


Fig. 1.6: An illustrative example of a dataset type in which the classifiers in the whole series category should perform specially well. The discriminatory characteristic is the shape of whole the time series, regardless the width and location of the pattern.

- **Interval classifier**: these methods extract features from different intervals of the series and build a tree on this representation. As such, instead of employing the whole series, the classifiers in this category focus local characteristics. In particular, they focus on some intervals in specific timestamps (location-dependent) of the series. They are appropriate for problems in which a particular subsequence of the series situated in a specific location is discriminatory. As shown in the example of Figure 1.7, this pattern can be the same in different classes, but the location of the pattern is a key aspect to distinguish between the classes. Some example classifiers in this category are the Time Series Forest (TSF) [24] or the Learned Pattern Similarity (LPS) [25].

- **Shapelet based classifiers**: the *shapelets* [26] are discriminatory subsequences that are representatives of the classes. In this type of methods, shapelets are extracted or learned from the training dataset and the prediction is made based on the minimum distances between the series and the shapelets. As shown in the example in Figure 1.8, shapelet based classifiers focus on location-independent subsequences of the series, i.e., discriminatory patterns that may appear in different locations in the time series within a given class. The most popular shapelet based classifiers are the Fast Shapelets (FS) [27] and the ST [28].

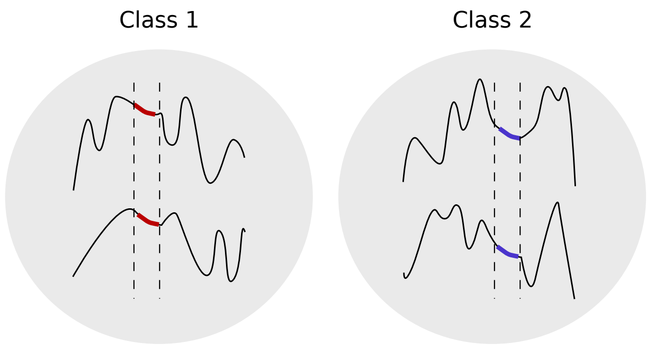


Fig. 1.7: An illustrative example of a dataset type in which interval based classifiers should perform specially well. The discriminatory characteristic is the highlighted pattern and the position in which it is located in all the series from the same class.

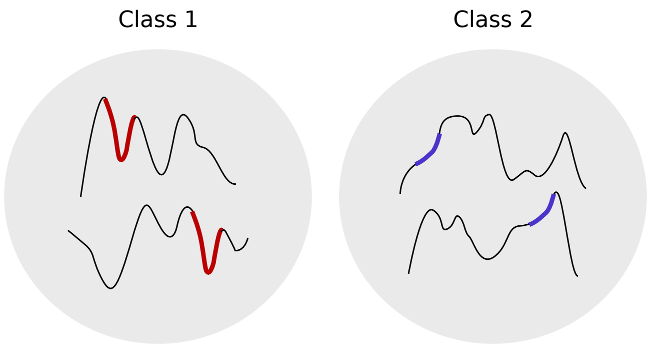


Fig. 1.8: An illustrative example of a dataset type in which shapelet based classifiers should perform specially well. The discriminatory characteristic is a local shape that is representative of the class membership, and the location of this pattern in the series from one class is not relevant.

- **Dictionary based classifiers**: These methods convert the intervals of the time series into words using a discretization technique and represent the time series by the histogram of the frequencies of these words. This type of classifier focus on global discriminatory characteristics. As such, as shown in Figure 1.9, rather than focusing only on the presence or absence of a particular pattern, the classifiers in this category focus on the frequency or repetitions in which a pattern appears within a series. Some examples of this type of classifier are the Bag of Patterns (BOP) [29] and BOSS [30].

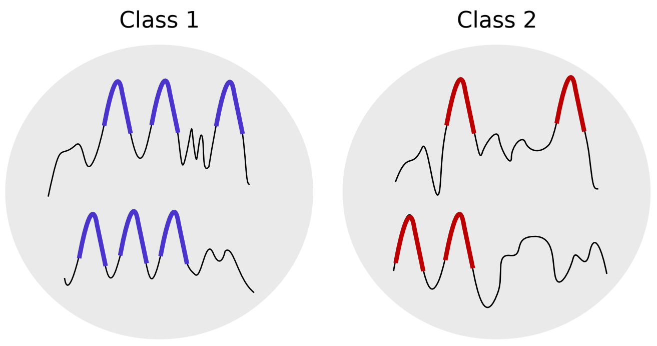


Fig. 1.9: An illustrative example of a dataset type in which dictionary based classifiers should perform specially well. The discriminatory characteristic in this dataset is the number of repetitions of a given pattern, regardless of the location. Note that, in this example the patterns are identical in both classes, which is not necessary.

- **Combinations**: Some classifiers combine two or more of the above mentioned approaches in ensemble methods. Given the hybrid nature of the ensembles, this type of methods usually perform very well in many types of problems but require extreme computational costs. Some examples of this type of classifiers are the Dynamic Time Warping Features (DTW_F) [16] or the COTE ensemble [31].

In the original taxonomy presented by Bagnall *et al.* [8], the authors included another category called model based classifiers. However, they stated that this type of classifiers are not included in their review and experimentation due to several reasons, such as the lack of available code and the low performance compared with other state-of-the-art classifiers.

Note that the distance based (in the first taxonomy) and the whole series (in the second taxonomy) categories share many classifiers but are not exactly the same. The classifiers in the whole series category are those that employ the 1-NN with different distance measures [8], while distance based methods include more classifiers that take advantage of the existing distance measures in different ways. For example, some of the classifiers that are based on shapelets [28], employ distances between the time series and the shapelets to classify new series, and, thus, can be considered as distance based approaches.

1.2 Meta-learning

Meta-learning is usually described as the task of learning to learn. The main goal of meta-learning systems is to provide automatic recommendation on model selection to a non-expert user. It is commonly used for classifier recommendation [32] or clustering method recommendation [33] , but it has also been employed for hyperparameter optimization [34]. Since this dissertation is focused on TSC, in the following section an introduction on how meta-learning is used in a classification framework is provided.

1.2.1 Meta-learning for classifier recommendation

Meta-learning systems aim at giving advice about model selection by exploiting meta-knowledge, i.e., knowledge about the learning process of a model in a given problem [35]. A meta-learning system for classifier recommendation has two main elements: a set of datasets and a set of classifiers. Additionally, the accuracy obtained by the classifiers in the considered datasets is known. With this information, the main objective is to learn a mapping between the datasets and the performances (expressed in different manners) of the classifiers. The general methodology of meta-learning systems for classifier recommendation is shown in Figure 1.10.

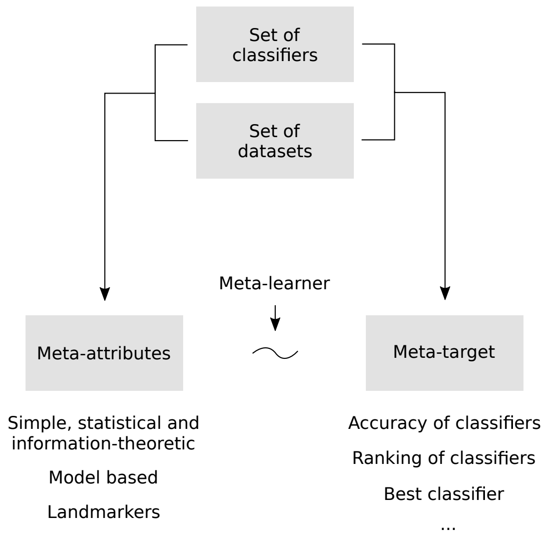


Fig. 1.10: A schema of the general methodology of meta-learning.

Given a set of datasets, meta-knowledge (also known as meta-attributes) is extracted from them and is used to characterize the datasets. Then, these meta-attributes are used as input of the recommendation system. From the accuracies of the classifiers in the datasets, a meta-target is constructed, which is the output of the recommendation system. This meta-target can take several forms, such as the accuracy of the classifiers in the given datasets or the index of the best performing classifier. If the output of the meta-learning system is the best classifier, for instance, the user is "constrained" to use this classifier and has no information about the rest of the classifiers. If the meta-target is a complete ranking of all the classifiers, contrarily, the user has much more information but, from a learning point of view, the problem becomes more complex.

Lastly, a mapping between the datasets (characterized by the meta-attributes) and the performance of the classifiers (described by the meta-target) is learned by a meta-learner. This algorithm will depend on the type of output (meta-target) that we choose: a regression model if the outputs are classifier accuracies, a ranking-based classifier if the output is a ranking of classifiers, etc.

The meta-attributes extracted from the datasets need to accomplish two requirements: on the one hand, they need to be good predictors of the performance of the classifiers. On the other, they need to be fast to compute, significantly faster than applying all the classifiers to a dataset. In the literature, there are three main ways in which meta-attributes can be extracted [35]: by employing simple, statistical and information-theoretic meta-features, model-based meta-features, and landmarkers.

- **Simple, statistical and information-theoretic** [36]: this type of meta-attributes employ descriptive statistics or information-theoretic measures extracted from the datasets, such as class entropy, to describe the datasets. As pointed out in [35], this characterization type has been widely employed in the literature, with highly competitive results. However, a major drawback is that these meta-attributes are designed for a standard classification framework in which the instances are feature vectors. Hence, do not take the temporal information contained in time series into account.

- **Model based** [37]: this type of meta-attributes are defined by parameters or characteristics found when applying a given classifier to a dataset. For instance, if a decision tree is used to classify the instances of a given dataset, some characteristics of the learned tree, such as the depth, can be used to describe the dataset. Model based meta-attributes generalize better to different types of data, since specific models (time series classifiers, in our case) can be used in each situation. However, this type of meta-attributes require to train a model and (in some cases) to tune the hyper-parameters, which can be computationally very expensive [38].

- **Landmarkers** [39, 40]: *landmarkers* are based on the idea that the accuracy obtained by fast and simple classifiers can be used to predict the performance of other computationally more expensive classifiers. As such, for each of the classifiers considered in the meta-learning problem, the corresponding landmarker is a quick estimator of the accuracy obtained by the classifier in a given dataset. These estimators are generally obtained in two manners [35]:

– Algorithm reduction [41]: this technique consists of running simplified versions of the candidate classifiers. This can be done, for example, by reducing the range of search of the hyper-parameters.

– Dataset reduction [42]: this procedure consists of estimating the performance of a candidate classifier by running the original classifier in a subsample of the data. The landmarkers obtained by this technique are also known as *subsampling landmarkers*.

The advantage of this meta-attribute type is that it can be designed to characterize any type of data and can be used for any type of classifier.

1.2.2 Meta-learning for time series

Despite its potential, the relationship between meta-learning and time series is almost non-existent in the literature. As far as we know, the few methods that exploit meta-learning for time series data are focused on forecasting method recommendation [43, 44, 45, 46]. Note that the meta-attributes introduced in the previous section are meta-attributes to characterize classification problems, i.e., problems that contain many instances. In forecasting, by contrast, a problem consists of a single time series, so the characterization can be more easily addressed. In the mentioned works, specific characteristics of time series are employed as meta-attributes, such as the trend, the seasonality or the kurtosis.

In [47], instead, a characterization for unsupervised time series datasets is proposed for automatic similarity measure selection for time series clustering. For this, the authors propose a set of specific meta-attributes for unsupervised time series datasets, such as the mean cross-correlation or the mean shift between each pair of time series.

Regarding the scope of this work, TSC, to the best of our knowledge, no recommendation system has been proposed. In particular, there is no specific characterization in the literature for supervised time series datasets.

1.3 Explainability

Over the years, the main trend in data mining research has focused on proposing novel, more accurate, and usually more complex models. The first models were transparent and interpretable (the decision making process was understandable for humans), but in the past few years, with the advent of deep learning models, most of the state-of-the-art methods are often opaque. Even if a model obtains a good performance in a given problem, being able to understand and interpret the decision making process is a key aspect to trust the model.

In order to deal with this issue, a new field of research, called Explainable Artificial Intelligence (XAI) [48], has emerged in recent years. Explainability can be defined as the task of finding a relation between the input data and the prediction of a model, in such a way that a human can understand or visualize this relation. For image data, for instance, the explanation of a model for a given instance is usually shown as a saliency map [49], where each pixel is colored depending on its relevance in the output of the model (as shown in the example in Figure 1.11). The explanations displayed in the figure show that the most relevant regions for dog breed classification are the faces. This reasoning agrees with the human perception and gives confidence to the model.

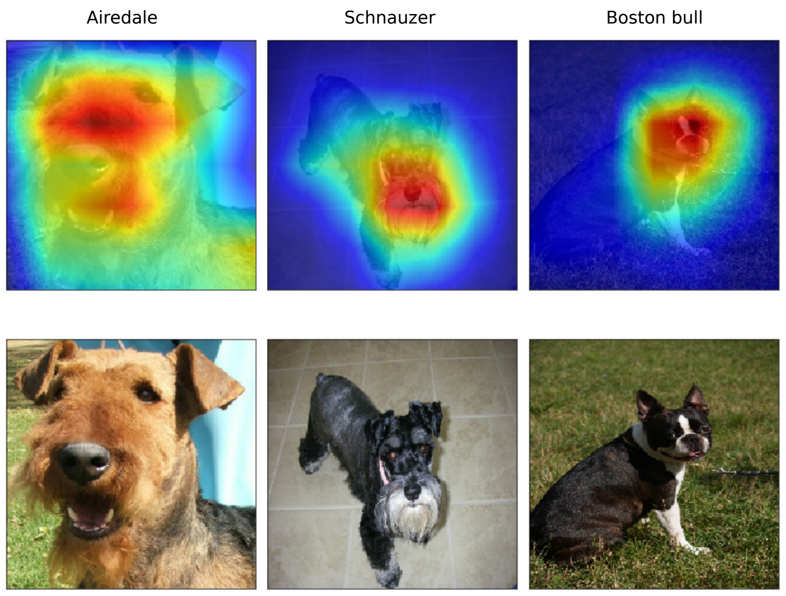


Fig. 1.11: Examples of explanations obtained for dog breed classification with the ResNet-CAM model [49][1]. Each pixel is coloured depending on the relevance in the prediction of the classifier (the red color indicates high relevance, while the blue color indicates low relevance.)

Explainability has become one of the main challenges in the data mining research community due to two main reasons: firstly, it can help experts to better understand the relationship between the input and output data of their problem. For example, in a medical scenario in which a model predicts that a patient will suffer from flu, an explanation method can reveal which are the symptoms that led to the prediction [50, 51]. Secondly, it has been shown that explanations can help the researchers or experts to detect wrong reasoning in classification problems [52].

With the aim of organizing the existing explanation methods, Du *et al.* [53] propose to categorize the approaches following two main criteria: model-dependence and scope. Regarding the model-dependence, a method can be model-dependent (also known as intrinsic) or model-independent (also called agnostic). Intrinsic explanations are obtained directly from the model and, as can be seen in several surveys [53, 54, 55, 56], most of the existing explanation methods are of this type. Some examples of intrinsic explanations include the Grad-CAM explainer for image classification with Neural Networks [57] or the ExplainD method for Support Vector Machines with tabular data [58].

Note that intrinsic methods are limited to the particular model they are designed for. As such, agnostic methods, i.e., methods that separate the explanation from the model and can be used to explain any model, have powerfully emerged in the past few years. Some examples of agnostic methods are the Anchors [59] or the LIME method [51].

Regarding the scope of the explanation methods, the methods can provide global or local explanations. Global explanations provide an understanding of the whole logic of a classifier, such as the importance of each feature for the prediction. An example of a global explanation method is presented in [60], in which a model agnostic iterative algorithm is used to find the relevance of each attribute in the prediction of a given classifier. Local explanations, instead, focus on individual instances and provide a specific explanation for each instance. All the examples of intrinsic and agnostic methods provided in the previous paragraphs [51, 57, 58, 59] are local explanation methods.

Since this dissertation is focused on TSC, a brief overview of the existing explanation methods for time series classifiers is introduced in the next section.

1.3.1 Explanation methods for time series

In the field of TSC, the growing interest in explainability has been materialized in several ways. As pointed out in the recent review [56], several intrinsic methods for explaining time series classifiers [61, 62, 63, 64, 65, 66, 67] have been proposed, some of them focused on explaining deep learning models [64, 65, 66, 67].

Concerning the intrinsic methods for explaining non-deep learning models, two main approaches can be distinguished. On the one hand, in [61, 63], the time series are represented in other symbolic spaces, for example by using a discretization based on bag of words, and a classifier that reports the relevance of each dimension is learned in those spaces. This information is then translated to the original temporal space, and an explanation that quantifies the relevance of each region of the series in the prediction is obtained. An example of the explanation provided by [63] is shown in Figure 1.12. On the other hand, in [62], shapelets are used to obtain an explanation of the Random Shapelet Forest (RSF) [68] classifier. In this method, small modifications are applied to the original time series until the classifier changes its decision. In this way, the explanation is not given in the form of the relevance of each region of the

[1] Images obtained from https://github.com/alexisbcook/ResNetCAM-keras.

series in the prediction, but in the form of a minimum transformation that changes the prediction.

In the case of intrinsic methods for explaining deep learning models, these methods generally employ a specific layer of the architecture, called the Class Activation Map (CAM), to induce the explanations. In [65], a review of this type of explanations is carried out. The explanation obtained by those methods is similar to that shown in Figure 1.12, in which the relevance of each region of the series in the prediction is visualized.

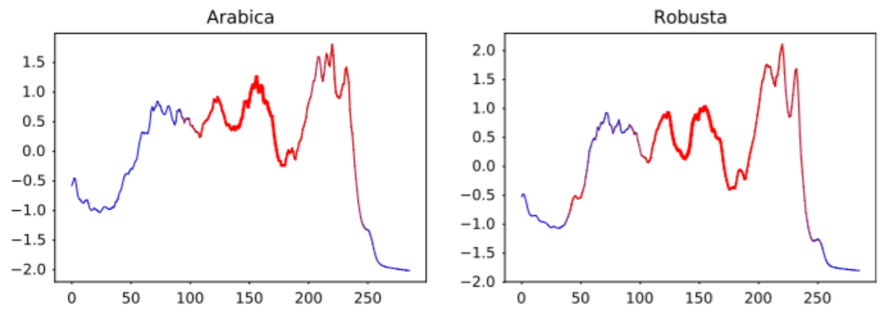


Fig. 1.12: An explanation obtained for two time series from the Coffee dataset by the method proposed in [63].

Even if most of the effort in TSC explanation has been focused on intrinsic methods, the weakness of this type of explanation is that it is limited to provide explanations of a specific classifier. As mentioned previously, agnostic explanations overcome this limitation. To the best of our knowledge, two agnostic methods have been proposed for TSC [69, 70].

In [69, 70], an agnostic local explanation method is presented by adapting one of the most popular perturbation-based approaches, Local Interpretable Model-agnostic Explanations (LIME) [51], to time series. The LIME method consists of locally approximating the classifier in the neighbourhood of the time series whose prediction will be explained. To this end, a synthetic neighbourhood is generated around the instance subject to explanation by perturbing it, i.e., by applying small modifications. The original method was designed for a standard classification framework and, hence, the perturbations are applied to the different dimensions or variables of the instance. Then, the explanation is obtained by measuring, for each feature, how the perturbation influences the prediction. In this sense, the main idea is that a feature is considered relevant for the prediction if a small modification of the value of this variable changes the prediction.

The adaptation to time series is carried out by, instead of applying perturbations to variables, applying perturbations to intervals [69, 70] of the time series. In both works, these modifications consist of replacing the given interval(s) with another subsequence, such as an interval of random noise, the linear interpolation, or other subsequences extracted from the time series of the training set.

1.4 Data and software sources

In order to carry out high quality contributions in any field of research, it is important to have a unified and publicly available framework for 1) reproducibility and 2) fair evaluation and comparison of methods. For this, in our field, the first required resource is a public data repository that allows the different proposed methods to be fairly compared under the same conditions (same data, in this case). In TSC, many efforts have been made in this regard and Bagnall *et al.* [2] created a public repository of supervised time series datasets called “The UEA & UCR Time Series Classification Repository”. This archive includes 128 univariate and 30 multivariate datasets, with a great variety of application domains, time series lengths, and number of classes.

Another aspect to be considered when performing a fair evaluation and comparison of different methods is the computational cost of a given method, which highly depends on the implementation itself and, in particular, on the programming language. In TSC, Bagnall *et al.* [8] made the first attempt in the direction of unifying implementations of time series classifiers, and implemented several benchmark methods in a Weka-compatible Java toolbox, called *tsml* [71]. Recently, a unified framework for machine learning with time series was implemented in Python, *sktime* (and the extension for deep learning *sktime-dl*) [72], that is a *scikit-learn* compatible tool that includes many time series classifiers.

In this dissertation, we will perform the corresponding evaluations employing the time series datasets from the UCR repository and the classifiers implemented in the *tsml*, *sktime* and *sktime-dl* toolboxes. In addition, with the aim of supporting open access research, all the code developed for the contributions presented in this dissertation is publicly available in https://gitlab.bcamath.org/aabanda.

1.5 Overview of the dissertation

In this dissertation, we present three contributions to the area of TSC. As introduced in Section 1.1, the main trend in TSC has been to propose novel and more accurate classifiers, and nowadays the number of available competitive classifiers is considerably large .

However, less effort has been made to study the similarities and differences of the state-of-the-art methods and it is difficult to have a global idea of the current outlook of the field. Moreover, in most cases, there is a lack of understanding of the behavior of the classifiers. With this in mind, the three contributions presented in this dissertation are focused on providing knowledge about the current state-of-the-art time series classifiers.

Firstly, note that, even if there are many ways to classify time series, distance based approaches are among the most popular methods. Together with the growing number of distance measures for time series, new manners (apart from the k-NN method) to exploit these distances within a classification framework have been proposed. The existing literature about distance based methods is unorganized and it is hard to draw a general idea of the current state-of-the-art distance based methods. As such, the first objective of this dissertation, addressed in Chapter 2, is:

Objective 1:

To provide a general taxonomy and review of distance based TSC methods.

We propose to categorize the existing approaches into three categories: those that use the distance together with k-NN, those that transform the series into feature vectors using distances, and those in which the distances are employed to obtain kernels for time series. We will see that converting the time series into feature vectors by a distance based representation can be seen as a preprocessing step, which bridges the gap between standard classifiers and time series data. Regarding the approaches that convert distances into kernels, we will see that a kernel, by definition, need to be a positive semi-definite function, and we will organize the proposed methods for TSC depending on how each approach handles this condition. We will comprehensively analyze each method, stressing its advantages, drawbacks and computational cost.

With the aim of overcoming the limitations of distance based classifiers, other approaches that focus on different characteristics of the series have been proposed (see Section 1.1.2). Bagnall *et al.* [8] stated that the methods in each of the proposed categories *should* be especially suitable to a particular type of problem or dataset. However, even if some studies have been carried out in this direction [73], this correlation has never been proved. Hence, given the heterogeneous and growing set of available classifiers, from a pragmatic user's point of view, choosing a suitable classifier for a given problem is a difficult task. Moreover, due to the computational cost of the process, applying all the classifiers to a given problem and choosing the most suitable method based on the obtained performances becomes an unrealistic approach. In this way, in Chapter 3, we will address the second objective of this dissertation:

Objective 2:

To propose a time series classifier recommendation system.

Our proposal is the first recommendation system for TSC in the literature, and we will follow a meta-learning approach for this. In our framework, 24 state-of-the-art time series classifiers will be considered, and the time series datasets will be characterized by employing landmarkers, based both on algorithm reduction and dataset reduction. In preliminary experiments, we will show how the accuracies obtained by the proposed landmarkers are highly correlated with the accuracies obtained by the original corresponding classifiers, while the landmarkers are much faster to compute. For the output of the recommendation system, five meta-targets will be considered: classifier accuracies, complete ranking, top-M ranking, best set, and best classifier. For each meta-target type, two specific meta-learners are considered, depending on the output data type. Given that this is the first time a time series classifier recommender is proposed, these meta-learners are compared with two baseline methods we propose, which our method outperforms significantly. Moreover, since some meta-targets are more fine-grained than others, we will exploit this relation.

Even if the second contribution helps us select a classifier for a given problem, many of the considered classifiers are complex in nature and difficult to explain. Indeed, understanding the relationship between the characteristics of a time series and the decision of a classifier is still an open question for many state-of-the-art classifiers. As mentioned in Section 1.3.1, some agnostic explanation methods for TSC [69, 70] have already been proposed. However, the main drawback of these methods is that the perturbations considered for creating the neighborhood of an instance are not realistic. For instance, if a time series represents the electricity consumption of a country (as in the ItalyPowerDemand dataset from the UCR repository [2]), replacing a given interval of the series by random noise does not produce a realistic time series. In particular, the generated neighbour does not have a semantic meaning, since an interval of random noise can not be interpreted from a point of view of electricity consumption. As such, the third objective of this dissertation, presented in Chapter 4, is:

Objective 3:

To propose an ad-hoc model-agnostic explanation method for time series classifiers, in which the perturbations considered are realistic for time series.

For this, we will follow the perturbation based approach employed in [69, 70], but the four perturbations (or transformations) considered are specific for time series: warp, scale, noise, and slice. Our method provides explanations at two levels: in the higher-level, the robustness of a classifier's prediction with respect to a given transformation is measured, while in the lower-level, the relevance of each region of the series in the prediction is computed.

For this, given a time series and a perturbation, we will create a synthetic neighbourhood of the time series by applying these transformations to random intervals of the series. We will define the robustness of a classifier with respect to a transformation by quantifying the percentage of neighbours for which the classifier changes the prediction. Then, we will identify the regions of the time series in which performing a perturbation will result in a change of prediction. In this way, we will consider that a region of the series is relevant for the prediction (with respect to a transformation) if a perturbation in this region changes the classifier's prediction. Our explanation method will be evaluated both qualitatively and quantitatively. For the qualitative evaluation, we will take advantage of the semantic meaning of the time series to visually validate our proposal in 2 benchmark datasets. The quantitative evaluation, instead, consists of an adaptation of the evaluation methodology proposed in [74] to our framework. The experimentation, carried out in 4 datasets and employing 3 benchmark classifiers, clearly shows that our explanation method is informative in almost all the considered combinations of dataset-classifier-transformation.

Lastly, the main conclusions drawn within this dissertation are presented in Chapter 5. We will also review the possible future directions of research, together with the main achievements of this dissertation, including journal and congress publications.

2

A Review on Distance based Time Series Classification

2.1 Introduction

In the field of TSC, as mentioned in Section 1.1, most of the research has been focused on distance based approaches. Given the popularity and competitiveness of the 1-NN-DTW method, distance based classification has been oriented to defining different types of distance measures and then exploiting them within k-NN classifiers. Due to the temporal (ordered) nature of the series, the high dimensionality, the noise, and the possible different lengths of the series in the database, the definition of a suitable distance measure is a key issue in distance based TSC.

One of the ways to categorize time series distance measures is shown in Figure 2.1; *Lock-step measures* refer to those distances that compare the ith point of one series to the ith point of another (e.g., Euclidean distance), while *elastic measures* aim to create a non-linear mapping in order to align the series and allow comparison of one-to-many points (e.g., DTW [10]). These two types of measures consider the important aspect to define the distance is the shape of the series, but there are also structure based or edit based measures, among others [4]. In this sense, different distance measures are able to capture different types of dissimilarities, and, even if in theory there is a best distance for each case [75], in practice it is hard to find it. Nevertheless, the experimentation in [4, 9, 12, 76, 77, 78, 79] has shown that, on average, the DTW distance seems to be particularly difficult to beat.

One of the simplest ways to exploit a distance measure within a classification process is by employing k-NN classifiers. One could expect that a more complex classifier would outperform the performance of the 1-NN and, as such, the bad performance of these complex classifiers may be attributed to the inability of the classifiers to deal with the temporal nature of the series using the default settings. On the other hand, it is known that the underlying distance is crucial to the performance of the 1-NN classifier [80] and, hence, the high accuracy of 1-NN classifiers may arise from the efficiency of the time series distance measures -which take into consideration the temporal

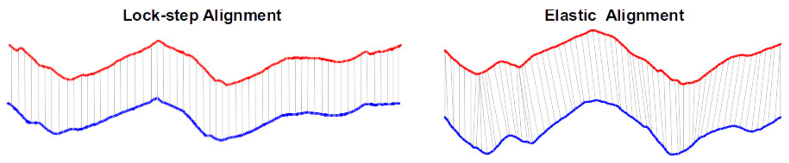


Fig. 2.1: Mapping of Euclidean distance (lock-step measure) vs. mapping of DTW distance (elastic measure) [76].

nature- for classification. In this way, methods that exploit the potential of these distances within more complex classifiers have emerged in the past few years [16, 81, 82], achieving performances that are competitive or outperform the classic 1-NN.

These new approaches aim to take advantage of the existing time series distances to exploit them within more complex classifiers. We have differentiated between two new ways of using distance measures in the literature: the first employs the distance to obtain a new feature representation of the series [16, 28, 83], i.e., a representation of the series as an order-free vector, while the second uses the distance to obtain a kernel [17, 82, 84], i.e., a similarity between the series that will then be used within a kernel method. Both approaches have achieved competitive classification results and, thus, different variants have arisen [85, 86, 87]. The purpose of this review is to present a taxonomy of all those methods which are based on time series distances for classification. At the same time, the strengths and shortcomings of each approach are discussed in order to give a general overview of the current research directions in distance based TSC.

The rest of the Chapter is organized as follows: in Section 2.2 the taxonomy of the reviewed methods is presented, as well as a brief description of the methods in each category. In Section 2.4 a discussion on the approaches in the taxonomy is presented, where we draw our conclusions and specify some future directions.

2.2 A taxonomy of distance based time series classification

As mentioned previously, the taxonomy we propose intends to include and categorize all the distance based approaches for TSC. A visual representation of the taxonomy can be seen in Figure 2.2. From the most general point of view, the methods can be divided into three main categories: in the first one, the distances are used directly in conjunction with k-NN classifiers; in the second one, the distances are used to obtain a new representation of the series by transforming them into features vectors, while in the third one, the distances are used to obtain kernels for time series.

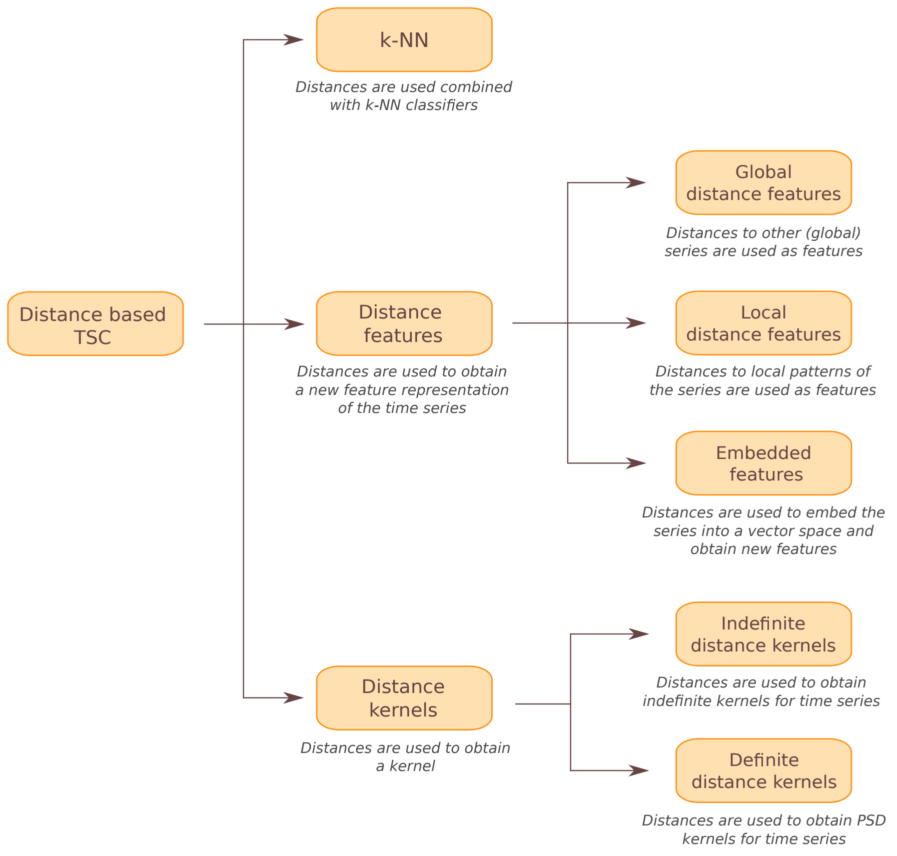

Distance based TSC
k-NN
Distances are used combined with k-NN classifiers
Distance features
Distances are used to obtain a new feature representation of the time series
Distance kernels
Distances are used to obtain a kernel
Global distance features
Distances to other (global) series are used as features
Local distance features
Distances to local patterns of the series are used as features
Embedded features
Distances are used to embed the series into a vector space and obtain new features
Indefinite distance kernels
Distances are used to obtain indefinite kernels for time series
Definite distance kernels
Distances are used to obtain PSD kernels for time series

Fig. 2.2: A visual representation of the proposed taxonomy of distance based time series classification methods.

2.2.1 k-Nearest Neighbour

This approach employs the existing time series distances within k-NN classifiers. In particular, the 1-NN classifier has mostly been used in TSC due to its simplicity and competitive performance [78, 88]. Given a distance measure and a time series, the 1-NN classifier predicts the class of this series as the class of the object closest to it from the training set. Despite the simplicity of this rule, a strength of the 1-NN is that as the size of the training set increases, the 1-NN classifier guarantees an error lower than two times the Bayes error [89]. Nevertheless, it is worth mentioning that it is very sensitive to noise in the training set, which is a common characteristic of time series datasets. This approach has been widely applied in TSC, as it achieves, in conjunction with the DTW distance, the best accuracies achieved on many benchmark datasets. As such, quite a few studies and reviews include the 1-NN in the time series literature [8, 76, 79, 90], and hence, it is not going to be further detailed in this review.

2.2.2 Distance features

In this group, we include the methods that employ a time series distance measure to obtain a new representation of the series in the form of feature vectors. In this manner, the series are transformed into feature vectors (order-free vectors in $\mathbb{R}^N$), overcoming many specific requirements that are encountered in TSC, such as dealing with ordered sequences or handling instances of different lengths. The main advantage of this approach is that it bridges the gap between TSC and conventional classification, enabling the use of general classification algorithms designed for vectors, while taking advantage of the potential time series distances. In this manner, calculating the distance features can be seen as a preprocessing step, thus, the transformation can be used in combination with any classifier. Note that even if these methods also obtain some features from the series, they are not considered within feature based TSC, but within distance based TSC. The reason is that the methods in feature based TSC obtain features that contain information about the series themselves, while distance features contain information relative to their relation with the other series. Three main approaches are distinguished within this category: those that directly employ the vector made up of the distances to other series as a feature vector, those that define the features using the distances to some local patterns, and those that use the distances after embedding the series into some vector space.

2.2.2.1 Global distance features

The main idea behind the methods in this category is to convert the time series into feature vectors by employing the vector of distances to other series as the new representation. Firstly, the distance matrix is built by calculating the distances between each pair of samples in the training set, as shown in Figure 2.3. The number of time series in the training set is n_{tr}. Then, each row of the distance matrix is used as a feature vector describing a time series, i.e., as input for the classifier. It is worth mentioning that this is a general approach (not specific for time series) but becomes specific when a time series distance measure is used. Learning with the distance features is also known as learning in the so-called *dissimilarity space* [91]. For more details on learning with global distance features in a general context, see [91, 92, 93, 94].

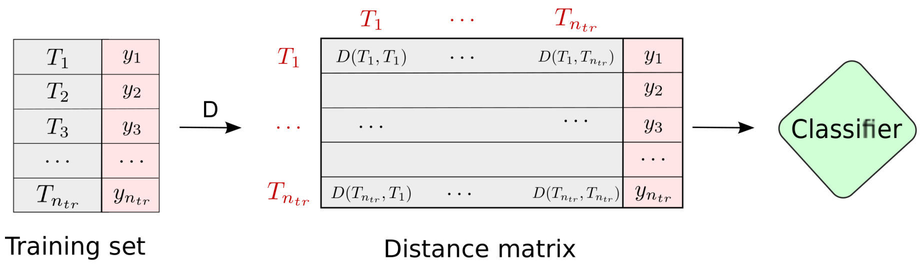


Fig. 2.3: A visual representation of the global distance features method.

Even if learning with distance features is a general solution, it is particularly advantageous for time series; the distance to each series is understood as an independent dimension and the series can be seen as vectors in a Euclidean space. This new representation enables the use of conventional classifiers that are designed for feature vectors, while it takes advantage of the existing time series distances. However, learning from the distance matrix has some important drawbacks; first, the distance matrix must be calculated, which may be costly depending on the complexity of the distance measure. Then, once the distance matrix has been calculated, learning a classifier may also incur large computational cost, due to the possible large size of the training set. Note that in the prediction stage, the consistent treatment of a new time series is straightforward -just the distances from the new series to the series in the training set have to be computed- but it can also become computationally expensive depending on the distance measure. Henceforth, given a distance measure D, we will refer to the methods employing the corresponding distance features as DF_d .

After this brief introduction of the distance based features, a summary of the methods employing them is now presented. In [84], the authors made the first attempt at investigating the feasibility of using a time series distance measure within a more complex classifier than the k-NN. In particular, they aimed at taking advantage of the potential of SVMs on the one hand, and of DTW on the other. First, they converted the DTW distance measure into two DTW based similarity measures, shown in equation (2.1). Then, they employed the distance features obtained from these similarity measures, DF_{GDTW} and DF_{NDTW}, in combination with SVMs for classification.

$$GDTW(T_i, T_j) = \exp\left(-\frac{DTW(T_i, T_j)^2}{\sigma^2}\right) \tag{2.1}$$
$$NDTW(T_i, T_j) = -DTW(T_i, T_j)$$

where $\sigma > 0$ is a free parameter and T_i, T_j are two time series. They concluded the new representation in conjunction with SVMs is competitive with the benchmark 1-NN with DTW.

In [81], the authors introduced a *Two-step DTW-SVM* classifier where the DF_{DTW} are used in order to solve a multi-class classification problem. In the prediction stage, the new time series is represented by the distance to all the series in the training set and a voting scheme is employed to classify the series using all the trained SVMs in a one-vs-all schema. They concluded that even if DF_{DTW} achieves acceptable accuracy values, the prediction of new time series is too slow for real world applications when the training set is relatively big.

Additionally, based on the potential of using distances as features for TSC, Kate *et al.* [16] carried out a comprehensive experimentation in which different distance measures are used as features within SVMs. In particular, they tested not only DF_{DTW} but also a constrained version DF_{DTW-R} (a window-size constrained version of DTW which is computationally faster [95]), features obtained from the Euclidean distance DF_{ED} and also concatenations of these distance features with other feature based representations. In their experimentation, they showed that even the DF_{ED}, when used as features with SVMs, outperforms the accuracy of 1-NN classifier based on the same Euclidean distance. An extension of [16] was presented in [96], who argued that not all relevant features can be described in the time domain (frequency domain can be more discriminative, for example) and added new representations to the set of features. Specifically, they generalized the concept of distance features to other domains and employed four different representations of the series with six different distance measures, giving rise to 24 distance features. For each representation of the series R_i, $i = 1, \ldots, 4$, they computed six different distance features $\mathrm{DF}_{d_1}^{R_i}, \ldots, \mathrm{DF}_{d_6}^{R_i}$. In their experimentation on 85 datasets from UCR, they showed that using representation diversity improves the classification accuracy. Finally, in their work about early classification of time series, [97] benefit from Euclidean distance features DF_{ED} in order to classify the series with SVMs and Gaussian Processes [98].

Recently, Wu *et al.* proposed another distance feature approach for TSC [99] which is based on *Random Features* [100] approximation. Following the methodology of the D2KE kernel [101] discussed in Section 2.2.3, the authors exploit the idea of randomly sampled time series and employ the distances from the original series to the random series as features: DF_{RF}. The random series are defined by S segments -where the length S is a user-defined parameter-, each segment associated with a random number. The idea is that these random series can be interpreted as the possible shapes of the time series. In the experiments carried out on 16 UCR datasets, they compare their representation -in combination with SVMs- against 6 state-of-the-art distance based classification methods. In particular, they propose two variants of their method: the first employs a large number of random series, while the second employs a small number. The experimentation shows that the first approach outperforms the accuracies of the baseline methods but incurs in large computational times, while the second obtains comparable accuracies in less time (reducing the time complexity from quadratic to linear).

With the aim of addressing the limitation of the high computational cost of the DTW distance, Goebel *et al.* [102] proposed the use of a fast lower bound for the DTW algorithm, called LB_Keogh [103]. Employing DF$_{LB\_Keogh}$ with SVMs, Janyalikit *et al.* showed in their experimentation on 47 UCR datasets that their method speeds the classification task up by a large margin, while maintaining the accuracies comparing with the state-of-art DF$_{DTW-R}$ proposed in [16].

As previously mentioned, another weakness of using distances as features is the high dimensionality of the distance matrix, since for n instances a $n \times n$ matrix is used as the input to the classifier. In view of this, Goebel *et al.* [104] proposed a dimensionality reduction approach using Principal Component Analysis (PCA) in order to keep only those dimensions that retain the most information. In their experimentation they compare the use of DF$_{DTW}$ with the reduced version of the same matrix, DF$_{DTW+PCA}$ in combination with SVMs. They showed PCA can have a consistent positive effect on the performance of the classifier but this effect seems to be dependent of the choice of the kernel function applied in the SVM. Note that for prediction purposes, they transform the new time series using the PCA projection learned from the training examples and, hence, the prediction process will also be significantly faster.

Another dimensionality reduction approach used in these cases is prototype selection, employed by Iwana *at el.* [83]. The idea is to select a set of p reference time series, called prototypes, and compute only the distances from the series to the p prototypes. The authors pointed out that the distance features let each feature be treated independently and, consequently, prototype selection can be seen as a feature selection process. As shown in [104], this dimensionality reduction technique not only speeds up the training phase but also the prediction of new time series. The proposed method uses the AdaBoost [105] algorithm, which is able to select discriminative prototypes and combine a set of weak learners. They experimented with DF$_{DTW+PROTO}$ and analyzed different prototype selection methods.

To conclude this section, a summary of the reviewed methods of *Global distance features* for TSC can be found in Table 2.1.

Table 2.1: Summary of global distance feature approaches

Authors	**Features**	**Classifier**	**Datasets**
Gudmundsson *et al.* [84]	DF_{GDTW}, DF_{NDTW}	SVMs	20 UCR
Jalalian *et al.* [81]	DF_{DTW}	SVMs	20 UCR
Kate *et al.* [16]	$\mathrm{DF}_{ED} - \mathrm{DF}_{DTW} - \mathrm{DF}_{DTW-R} - SAX$	SVMs	47 UCR
Giusti *et al.* [96]	$\mathrm{DF}^{R_{1,\ldots,4}}_{d_{1,\ldots,6}}$	SVMs	85 UCR
Mori *et al.* [97]	DF_{ED}	GPs, SVMs	45 UCR
Wu *et al.* [99]	DF_{RF}	SVMs	16 UCR
Goebel *et al.* [102]	DF_{LB_Keogh}	SVMs	47 UCR
Goebel *et al.* [104]	$\mathrm{DF}_{DTW+PCA}$	SVMs	42 UCR
Iwana *et al.* [83]	$\mathrm{DF}_{DTW+PROTO}$	Adaboost	1 (UNIPEN)

2.2.2.2 Local distance features

In this section, instead of using distances between entire series, distance to some local patterns of the series are used as features. Instead of assuming that the discriminatory characteristics of the series are global, the methods in this section consider that they are local. As such, the methods in this category employ the so-called *shapelets* [26], subsequences of the series that are identified as being representative of the different classes. An example of three shapelets belonging to different time series can be seen in Figure 2.4. An important advantage of working with shapelets is their interpretability, since an expert may understand the meaning of the obtained shapelets. By definition, shapelets are subsequences and as such, the methods employing shapelets are not a priori applicable to other types of data. However, it is worth mentioning that the original shapelet discovery technique [26] is carried out by enumerating all possible candidates (all possible subsequences of the series) and using a measure based on information theory that takes $O(n^2l^4)$, where n is the number of time series and l is the length of the longest series. Thereby, most of the work related to shapelets has focused on speeding up the shapelet discovery process [27, 106, 107, 108] or on proposing new shapelet learning methods [109]. However, we will not focus on that but rather on how shapelets can be used within distance based classification.

Building on the achievements of shapelets in classification, Lines *et al.* [88] introduced the concept of *Shapelet Transform* (ST). First, the m most discriminative (over the classes) shapelets are found using one of the methods referenced above. Then, the distances from each series to the shapelets are computed and the shapelet distance matrix shown in Figure 2.5 is constructed. Finally, the vectors of distances are used as input to the classifier. In [88], the distance between a shapelet of length s and a time series is defined as the minimum Euclidean distance between the shapelet and all the subsequences of the series of length s. Shapelet transformation can be used in combination with any classifier and, in their proposal, the authors experimented with seven classifiers (C4.5, 1-NN, Naïve Bayes, Bayesian Network,

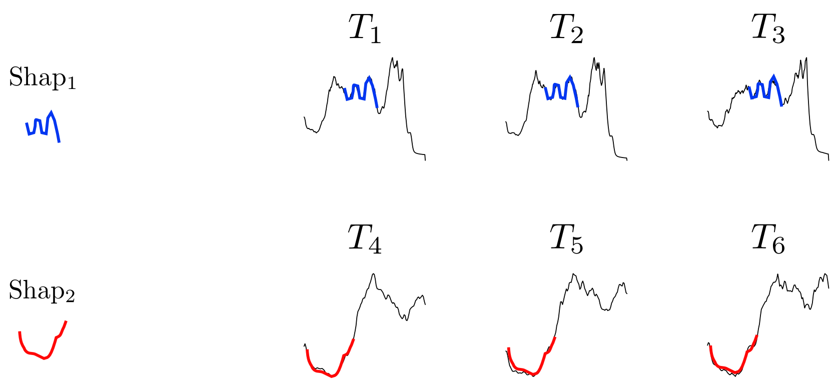


Fig. 2.4: Visual representation of two shapelets (Shap$_1$ and Shap$_2$) and six time series from the Coffee dataset (UCR). These shapelets are identified as being representative of class membership: Shap$_1$ belongs to class 1, as can be seen in the three time series (T$_1$, T$_2$ and T$_3$) which belong to class 1, while Shap$_2$ belongs to class 2, as can be seen in the three time series (T$_4$, T$_5$ and T$_6$) which belong to class 2.

Random Forest, Rotation Forest and SVMs) and 26 datasets, showing the benefits of the proposed transformation.

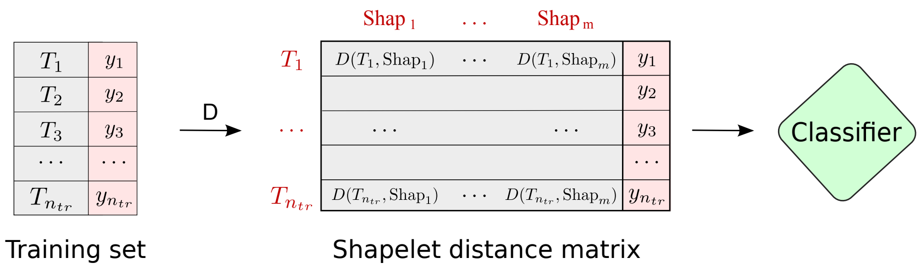

T_1 | y_1
T_2 | y_2
T_3 | y_3
$\cdots$ | $\cdots$
$T_{n_{tr}}$ | $y_{n_{tr}}$
Training set
D
Shap$_1$ $\cdots$ Shap$_m$
T_1 $D(T_1, \text{Shap}_1)$ $\cdots$ $D(T_1, \text{Shap}_m)$ y_1
y_2
$\cdots$ $\cdots$ $\cdots$ y_3
$\cdots$
$T_{n_{tr}}$ $D(T_{n_{tr}}, \text{Shap}_1)$ $\cdots$ $D(T_{n_{tr}}, \text{Shap}_m)$ $y_{n_{tr}}$
Shapelet distance matrix
Classifier

Fig. 2.5: Example of the local distance features methods using ST.

Hills *et al.* [28] provided an extension of [88] that includes a comprehensive evaluation which analyzes the performance of the seven aforementioned classifiers using the complete series and the ST as input. As such, the authors concluded that the ST gives rise to improvements in classification accuracy in several datasets. In the same line, Bostrom *et al.* [110] proposed another shapelet learning strategy (called *binary ST*) and evaluated their ST in conjunction with an ensemble classifier on 85 UCR datasets, showing that it clearly outperforms conventional approaches of TSC.

Recently, Li & Lin [2018] proposed another approach that exploits time series distances in a novel way: their method maps the series into a specific dissimilarity space in which the different classes are effectively separated. This specific dissimilarity space is defined based on what they call Separating References (SRs), which, in practice, are subsequences. These SRs are found, by means of an evolutionary process, such that the distances between the SRs and series belonging to different classes differs with a large margin. The corresponding decision boundaries that split the classes in the dissimilarity space are also found during the same process. As such, this approach does not specifically employ distances as features but, since it is very related to the methods in this category, it has been included. They experiment with 40 UCR datasets showing that their Evolving Separating References (ESR) approach is competitive with the benchmark TSC methods, being particularly suitable for datasets in which the size of learning set is small."

Lastly, [111] introduced another representative subsequence based approach that is similar to shapelet based methods but from a novel perspective. Their method first transforms the real-valued series into discrete-valued series using Symbolic Aggregate approXimation (SAX) [112] and employs a grammar induction (GI) procedure [113] to generate a pool of representative pattern candidates. Then, it selects the most representative patterns from these candidates and transforms them back into subsequences. Finally, the series are represented by a vector containing the distances from the series to these subsequences, and the classification is carried out using SVMs.A significant difference between this method, called Representative Pattern Mining (RPM), and shapelet based methods is that, while a shapelet may be representative of more than one class -exclusiveness is not required-, in RPM the representative subsequences can only belong to one class. In addition, the pattern discovery in RPM is much more efficient than the existing shapelet discovery procedures.

To sum up, a summary of the reviewed methods that employ *Local distance features* can be found in Table 2.2.

Table 2.2: Summary of Local distance feature approaches

Authors	**Features**	**Classifier**	**Datasets**
Lines *et al.* [88]	ST	7 classifiers*	18 UCR + 8 own
Hills *et al.* [28]	ST	7 classifiers*	17 UCR + 12 own
Bostrom *et al.* [114]	Binary ST	Ensemble	85 UCR
Li *et al.* [115]	SRs	ESR	40 UCR
Wang *et al.* [111]	RPM	SVMs	42 UCR + 1 own

* C4.5, 1-NN, Naïve Bayes, Bayesian Network, Random Forest, Rotation Forest and SVMs

2.2.2.3 Embedded features

The methods presented until now within the *Distance features* category employ the distances directly to create feature vectors representing the series, however, this is not the only way to use the distances. In the last approach within this section, the methods using *Embedded features* do not employ the distances directly as the new representation. Instead, they make use of them to obtain a new representation. In particular, the distances are used to isometrically embed the series into some Euclidean space while preserving the distances.

The distance embedding approach is not a specific method for time series. In many areas of research, such as empirical sciences, psychology, or biochemistry, it is common to have (dis)similarities between the input objects and not the objects per se. As such, one may learn directly in the dissimilarity space mentioned in Section 2.2.2.1, or one may try to find some vectors whose distances approximate the given (dis)similarities. If the given dissimilarities come from the Euclidean distance, it is possible to easily find some vectors that approximate the given distances. This is known in literature as *metric multidimensional scaling* [116]. On the contrary, if the distances are not Euclidean (or even not metric), the embedding approach is not straightforward and many works have addressed this issue in research [93, 94, 117, 118].

In the case of time series, this approach is particularly advantageous since a vector representation of the series is obtained such that the Euclidean distances between these vectors approximate the given time series distances. The main motivation is that many classifiers are implicitly built on Euclidean spaces [118] and this approach aims to bridge the gap between the Euclidean space and elastic distance measures. However, as it will be seen, the consistent treatment of new test instances is not straightforward and it is an issue to be considered.

As examples in TSC, the authors of [119] and [120] proposed, for the first time, a time series embedding approach in which a vector representation of the series is found such that the Euclidean distances between these vectors approximate the DTW distances between the series, as represented in Figure 2.6. They applied three embedding methods: multidimensional scaling, pseudo-Euclidean space embedding, and Euclidean space embedding by the Laplacian eigenmap technique [121]. They experimented with linear classifiers and a unique dataset (Australian Sign Language (ASL) [122]), in which their Laplacian eigenmap based embedded method achieved a better performance than the 1-NN classifier with DTW.

Another approach presented in [123] first defines a DTW based similarity measure, called DTWS, following the relation between distances and inner products [124] (see equation (2.2)). Then they search for some vectors such that the inner product between these vectors approximates the given DTWS:

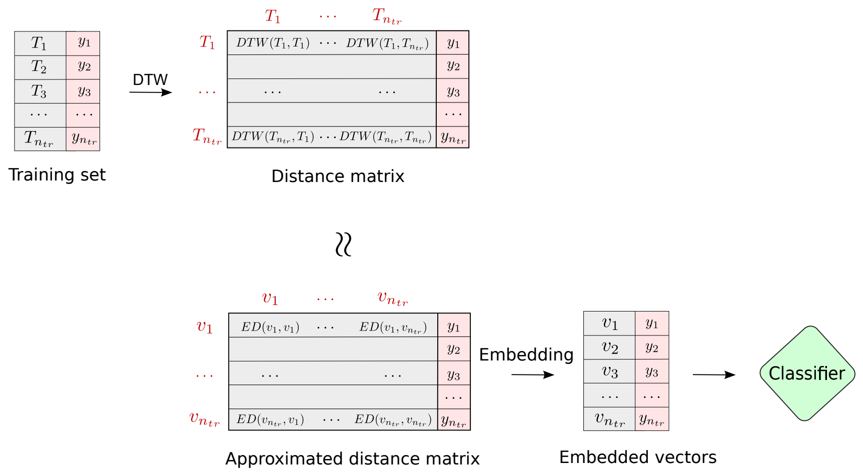


Fig. 2.6: Example of the stages of embedded distance features methods using the approach proposed in [119].

$$DTWS(T_i, T_j) = \frac{DTW(T_i, 0)^2 + DTW(T_j, 0)^2 - DTW(T_i, T_j)^2}{2} \tag{2.2}$$

where 0 denotes the time series of length one of value 0. Their method learns the optimal vector representation preserving the DTWS by a gradient descent method, but a major drawback is that it learns the transformed time series, but not the transformation itself. The authors propose an interesting solution to deal with the high computational cost of DTW, which consists of assuming that the obtained DTWS similarity matrix is a low-rank matrix. As such, by applying the theory of matrix completion, sampling only $O(n \log n)$ pairs of time series is enough to perfectly approximate a $n \times n$ low-rank matrix [125]. However, it is not possible to transform new unlabeled time series, which makes the method rather inapplicable in most contexts.

Finally, Lods *et al.* [87] presented a particular case of embedding that is based on the shapelet transform (ST) presented in the previous section. Their proposal learns a vector representation of the series (the ST), such that the Euclidean distance between the representations approximates the DTW between the series. In other words, the Euclidean distances between the row vectors representing each series in Figure 2.5 approximate the DTW distances between the corresponding time series. The main drawback of this approach is the time complexity in the training stage: first all the DTW distances are computed and then, the optimal shapelets are found by a stochastic gradient descent method. However, once the shapelets are found, the transformation of new unlabeled instances is straightforward, since it is done by computing the Euclidean distance between these series and shapelets. Note that the authors do not use their approach for classifying time series but for clustering, but since it is closely related to the methods in this review and their transformation can be directly applied to classification, it has been included in the taxonomy.

As previously mentioned, an important aspect to be considered in the methods using embedded features is the consistent treatment of unlabeled test samples, which depends on the embedding technique used. In [120], for instance, it is not clearly specified how unlabeled instances are treated. The method by Lei *et al.* [123], on the other hand, learns the transformed data and not the transformation, hence it is not applicable to real problems. Lastly, in [87], new instances are transformed by computing the distance from these new series to the learnt shapelets.

To end this section, a summary of the reviewed methods employing *Embedded distance features* for TSC can be found in Table 2.3.

Table 2.3: Summary of embedded distance feature approaches

Authors	**Features**	**Classifier**	**Datasets**
Mizuhara *et al.* [120]	DTW distance preserving vectors	Linear classifiers	ASL
Lei *et al.* [123]	DTWS similarity preserving vectors	XGBoost	6 own
Lods *et al.* [87]	DTW distance preserving ST	clustering	15 UCR

2.2.3 Distance kernels

The methods within this category do not employ the existing time series distances to obtain a new representation of the series. Instead, they use them to obtain a kernel for time series. Before going in-depth into the different approaches, a brief introduction to kernels and kernel methods is presented.

2.2.3.1 An introduction to kernels

The kernel function is the core of kernel methods, a family of pattern recognition algorithms, whose best known instance is the SVM [126]. Many machine learning algorithms require the data to be in feature vector form, while kernel methods require only a similarity function (known as kernel) expressing the similarity over pairs of input objects [127]. The main advantage of this approach is that one can handle any kind of data including vectors, matrices, or structured objects, such as sequences or graphs, by defining a suitable kernel which is able to capture the similarity between any two pairs of inputs. The idea behind a kernel is that if two inputs are similar, their output on the kernel will be similar, too.

More specifically, a kernel κ is a similarity function

$$\begin{aligned}\kappa : \mathcal{X} \times \mathcal{X} &\rightarrow \mathbb{R}\\ (x, x') &\rightarrow \kappa(x, x')\end{aligned}$$

that for all $x, x' \in \mathcal{X}$ satisfies

$$\kappa(x, x') = \langle \Phi(x), \Phi(x') \rangle \tag{2.3}$$

where Φ is the mapping from $\mathcal{X}$ into some high dimensional feature space and $\langle , \rangle$ is an inner product. As equation (2.3) shows, a kernel κ is defined by means of a inner product $\langle , \rangle$ in some high dimensional feature space. This feature space is called a Hilbert space and the power of kernel methods lies in the implicit use of these spaces [128].

In practice, the evaluation of the kernel function is one of the steps within the phases of a kernel method. Figure 2.7 shows the usage of the kernel function within a kernel method and the stages involved in the process. First, the kernel function is applied to the input objects in order to obtain a kernel matrix (also called Gram matrix), which is a similarity matrix with entries $K_{ij} = \kappa(x_i, x_j)$ for each input pair x_i, x_j. Then, this kernel matrix is used by the kernel method algorithm in order to produce a pattern function that is used to process unseen instances.

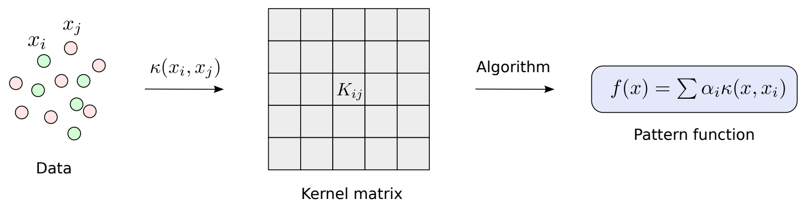


Fig. 2.7: The stages involved in the application of kernel methods [127].

An important aspect to consider is that the class of similarity functions that satisfies (2.3), and hence are kernels, coincides with the class of similarity functions that are symmetric and positive semi-definite [127].

Definition 4. ***(Positive semi-definite kernel)*** *A symmetric function* $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ *satisfying*

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j \kappa(x_i, x_j) \geq 0 \tag{2.4}$$

for any $n \in \mathbb{N}, x_1, \ldots, x_n \in \mathcal{X}, c_1, \ldots, c_n \in \mathbb{R}$ *is called a positive semi-definite kernel (PSD) [129].*

As such, any PSD similarity function satisfies (2.3) and (since it is a kernel) defines an inner product in some Hilbert space. Moreover, since any kernel guarantees the existence of the mapping implicitly, an explicit representation for Φ is not necessary. This is also known as the *kernel trick* (see [127] for more details).

Remark 1. We will also refer to a PSD kernel as a *definite kernel.*

Remark 2. We will informally denominate *indefinite kernels* to non-PSD kernels which are employed in practice as kernels, even if they do not strictly meet the definition.

Providing the analytical proof of the positive semi-definiteness of a kernel is rather cumbersome. In fact, a kernel does not need to have a closed-form analytic expression. In addition, as Figure 2.7 shows, the way of using a kernel function in practice is via the kernel matrix and, hence, the definiteness of a kernel function is usually evaluated experimentally for a specific set of inputs by analysing the positive semi-definiteness of the kernel matrix.

Definition 5. ***(Positive semi-definite matrix)*** *A square symmetric matrix* $\mathbf{K} \in \mathbb{R}^{n \times n}$ *satisfying*

$$\mathbf{v}^T \mathbf{K} \mathbf{v} \geq 0 \tag{2.5}$$

for any vector $\mathbf{v} \in \mathbb{R}^n$ *is called a positive semi-definite matrix [129].*

The following well-known result is obtained from [127]:

Proposition 1. *The inequality in equation (2.5) holds* $\Leftrightarrow$ *all eigenvalues of* $\mathbf{K}$ *are non-negative.*

Therefore, if all the eigenvalues of a kernel matrix are non-negative, this kernel function is considered PSD for the particular instance set in which it has been evaluated. In this manner, the definiteness of a kernel function is usually studied by the eigenvalue analysis of the corresponding kernel matrix. However, a severe drawback of this approach is that the analysis is only performed for a particular set of instances, and it cannot be generalized.

After introducing the basic concepts related to kernels, some examples of different types of kernels are now presented. As previously mentioned, one of the main strengths of kernels is that they can be defined for any type of data, including structured objects, for instance:

- **Kernels for vectors:** Given two vectors $\mathbf{x}, \mathbf{x}'$, the popular Gaussian Radial Basis Function (RBF) kernel [127] is defined by

  $$\kappa(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{||\mathbf{x} - \mathbf{x}'||^2}{2\sigma^2}\right) \tag{2.6}$$

  where $\sigma > 0$ is a free parameter.
- **Kernels for strings:** Given two strings, the p-spectrum kernel [130] is defined as the number of sub-strings of length p that they have in common.

- **Kernels for time series:** Give two time series, a kernel for time series returns a similarity between the series. There are plenty of ways of defining a similarity. For instance, two time series may be considered similar if they are generated by the same underlying statistical model [131]. In this review, we will focus on those kernels that employ a time series distance measure to evaluate the similarity between the series.

Therefore, in this category denominated *Distance kernels*, instead of using a distance to obtain a new representation of the series, the distances are used to obtain a kernel for time series. As such, the methods in this category aim to take advantage of the potential of time series distances and the power of kernel methods. Two main approaches are distinguished within this category: those that construct and employ an indefinite kernel, and those that construct kernels for time series that are, by definition, PSD.

2.2.3.2 Indefinite distance kernels

The main goal of the methods in this category is to convert a time series distance measure into a kernel. Most distance measures do not trivially lead to PSD kernels, so many works focus on learning with indefinite kernels. The main drawback of learning with indefinite kernels is that the mathematical foundations of the kernel methods are not guaranteed [132]. The existence of the feature space to which the data is mapped (equation (2.3)) is not guaranteed and, due to the missing geometrical interpretation, many good properties of learning in that space (such as orthogonality and projection) are no longer available [132]. In addition, some kernel methods do not allow indefinite kernels (due to the implementation or the definition of the method) and some modifications must be carried out, but for others the definiteness is not a requirement. For example, in the case of SVMs, the optimization problem that has to be solved is no longer convex, so reaching the global optimum is not guaranteed [92]. However, note that good classification results can still be obtained [133, 134, 135], and as such, some works focus on studying the theoretical background about SVMs feature space interpretation with indefinite kernels [136]. Another approach, for instance, employs heuristics on the formulation of SVMs to find a local solution [137] but, to the best of our knowledge, it has not been applied to TSC. Converting a distance into a kernel is not a specific challenge of time series and there is a considerable amount of work done in this direction in other contexts [92, 138].

For TSC, most of the work focuses on employing the distance kernels proposed in [138]. They propose to replace the Euclidean distance in traditional kernel functions, such as the Gaussian kernel in equation 2.6, by the problem specific distance measure. They called these kernels *distance substitution kernels*. In particular, we will call the following kernel *Gaussian Distance Substitution (GDS)* [138]:

$$GDS_D(x, x') = \exp\left(-\frac{d(x, x')^2}{\sigma^2}\right) \tag{2.7}$$

where x, x' are two inputs, D is a distance measure and $\sigma > 0$ is a free parameter. This kernel can be seen as a generalization of the Gaussian RBF kernel presented in the previous section, in which the Euclidean distance is replaced with the distance calculated by D. For the GDS kernel, the authors of [138] state that GDS_D is PSD if and only if D is isometric to an L-2 norm, which is generally not the case. As such, the methods which use this type of kernel for time series generally employ indefinite kernels.

Within the methods employing indefinite kernels, there are different approaches, and for TSC we have distinguished three main directions (shown in Figure 2.8). Some of them just learn with the indefinite kernels [90, 133, 135, 139, 140] using kernel methods that allow this kind of kernels and without taking into consideration that they are indefinite; others argue that the indefiniteness adversely affects the performance and present some alternatives or solutions [81, 84, 141]; finally, others focus on a better understanding of these distance kernels in order to investigate the reason for the indefiniteness [86, 142].

Indefinite distance kernels

- **Employing indefinite kernels**
  Jalalian *et al.* [81]
  Gudmundsson *et al.* [84]
  Kaya *et al.* [90]
  Bahlmann *et al.* [133]
  Shimodaira *et al.* [135]
  Pree *et al.* [139]
  Jeong *et al.* [140]
- **Dealing with the indefiniteness**
  Jalalian *et al.* [81]
- **Regularization**
  Chen *et al.* [141]
- **Analyzing the indefiniteness**
  Zhang *et al.* [86]
  Lei *et al.* [142]

Fig. 2.8: Different approaches taken with indefinite distance kernels

Employing indefinite kernels

Bahlmann *et al.* [133] made the first attempt to introduce a time series specific distance measure within a kernel. They introduced the GDTW measure presented in equation (2.1) as a kernel for character recognition with SVMs. This kernel coincides with the GDS kernel in equation (2.7), in which the distance d is replaced by the DTW distance, i.e., GDS_{DTW}. They remarked that this kernel is not PSD since simple counter-examples can be found in which the kernel matrix has negative eigenvalues. However, they obtained good classification results and argued that for the UNIPEN[1] dataset, most of the eigenvalues of the kernel matrix were measured to be non-negative, concluding that somehow, in the given dataset, the proposed kernel matrix is almost PSD. Following the same direction, Jeong *et al.* [140] proposed a variant of GDS_{DTW} which employs the Weighted DTW (WDTW) measure in order to prevent distortions by outliers, while in [90] also employed the GDS kernel with SVMs, but instead of using the distance calculated by the DTW, they explored other distances derived from different alignment methods of the series, such as Signal Alignment via Genetic Algorithm (SAGA) [143]. Pree *et al.* [139] proposed a quantitative comparison of different time series similarity measures used either to construct kernels for SVMs or directly for 1-NN classification, concluding that some of the measures benefit from being applied in an SVM, while others do not. Note that in this last work, how they construct the kernel for each distance measure is not exactly detailed.

There is another method that employs a distance based indefinite kernel but takes a completely different approach to construct the kernel: the idea of this kernel is to, rather than use an existing distance measure, incorporate the concept of alignment between series into the kernel function itself. Many elastic measures for time series deal with the notion of alignment of series. The DTW distance, for instance, finds an optimal alignment between two time series such that the Euclidean distance between the aligned series is minimized. Following the same idea, in DTAK, Shimodaira *et al.* [135] align two series so that their similarity is maximized. In other words, their method finds an alignment between the series that maximizes a given similarity (defined by the user), and this maximal similarity is used directly as a kernel. They give some good properties of the proposed kernel but they remark that it is not PSD, since negative eigenvalues can be found in the kernel matrices of DTAK [144].

On the other hand, Gudmundsson *et al.* [84] employed the DTW based similarity measures they proposed (shown in equantion (2.1)) directly as kernels. Their method achieved low classification accuracies and the authors claimed that another way of introducing a distance into a SVM is by using the distance features introduced in Section 2.2.2.1. They compared the performance of DTW based distance features and DTW based distance kernels, concluding that distance features outperform the distance kernels due to the indefiniteness of these second ones.

Dealing with the indefiniteness

There is a group of methods that attribute the poor performance of their kernel methods to the indefiniteness, and propose some alternatives or solutions to overcome these limitations. Jalalian *et al.* [81], for instance, proposed the use of a special SVM called Potential Support Vector Machine (P-SVM) [146] to overcome the shortcomings of learning with indefinite kernels. They employed the GDS$_{DTW}$ kernel within this SVM classifier which is able to handle kernel matrices that are neither positive definite nor square. They carried out an extensive experimentation including a comparison of their method with the 1-NN classifier and with the methods presented by Gudmundsson *et al.* [84]. They conclude that their DTW based P-SVM method significantly outperforms both distance features and indefinite distance kernels, as well as the benchmark methods in 20 UCR datasets.

Regularization

Another approach that tries to overcome the use of indefinite kernels consists of regularizing the indefinite kernel matrices to obtain PSD matrices. As previously mentioned, a matrix is PSD if and only if all its eigenvalues are non-negative, and a kernel matrix therefore can be regularized by clipping all the negative eigenvalues to zero, for instance. This technique has been usually applied for non-temporal data [92, 147, 148] but it is rather unexplored in the domain of indefinite time series kernels. Chen *et al.* [141] proposed a Kernel Sparse Representation based Classifier (SRC) [149] with some indefinite time series kernels and applied spectrum regularization to the kernel matrices. In particular, they employed the GDS$_{DTW}$, GDS$_{ERP}$ (Edit distance with Real Penalty (ERP) [150]) and GDS$_{TWED}$ (Time Warp Edit Distance (TWED) [14]) kernels and their method checks whether the kernel matrix obtained for a specific dataset is PSD. If it is not, the corresponding kernel matrix is regularized using the spectrum clip approach.

Regarding this approach, it is also worth mentioning that in the work by Gudmundsson *et al.* [84], the authors point out that they tried to apply some regularization to the kernel matrix subtracting the smallest eigenvalue from the diagonal but they found out that the method achieved a considerably low performance. Additionally, the authors added that matrix regularization can lead to matrices with large diagonal entries, which may result in overfitting [151].

Finally, the consistent treatment of training and new unlabeled instances is not straightforward and is also a matter to bear in mind [92]. When new unlabeled instances arrive, the kernel between them and the training set has

[1] On-line handwritten digit data set [145].

to be computed. If the kernel matrix corresponding to the training set has been regularized, the kernel matrix corresponding to the unlabeled set should also be modified in a consistent way, which is not a trivial operation. Therefore, the benefit of matrix regularization in the context of time series is an open question.

Analyzing the indefiniteness

The last group of methods do not focus on solving the problems of learning with indefinite kernels but, instead, focus on a better understanding of these distance kernels and their indefiniteness. Lei *et al.* [142] theoretically analyze the GDS$_{DTW}$ kernel, proving that it is not a PSD kernel. This is because DTW is not a metric (it violates the triangle inequality [152]) and non-metricity prevents definiteness [138]. That is, if d is not metric, GDS$_d$ is not PSD. However, the contrary is not true and, hence, the metric property of a distance measure is not a sufficient condition to guarantee a PSD kernel. In any case, Zhang *et al.* [86], hypothesized kernels based on metrics give rise to better performances than kernels based on distance measures which are not metrics. As such, they define what they called the Gaussian Elastic Metric Kernel (GEMK), a family of GDS kernels in which the distance D is replaced by an elastic measure which is also a metric. They employed GDS$_{ERP}$ and GDS$_{TWED}$ and stated that, even if the definiteness of these kernels is not guaranteed, they did not observe any violations of their definiteness in their experimentation on 20 UCR datasets. In fact, these kernels are shown to perform better than the GDS$_{DTW}$ and the Gaussian kernel in those experiments. The authors attribute this to the fact that the proposed measures are both elastic and obey metricity. In order to provide some information about the most common distance measures applied in this context, table 2.4 shows a summary of properties of the main distance measures employed in this review. In particular, we specify if a given distance measure D is a metric or not, if it is an elastic measure or not, and if the corresponding GDS$_D$ is proven to be PSD or not.

Table 2.4: Summary of distance properties used in GDS

Distance	**metric**	**elastic**	GDS_d **is PSD**
Euclidean	✓	×	✓
DTW	×	✓	×
ERP	✓	✓	×
TWED	✓	✓	×

To sum up, there are some results that suggest a relationship between the metricity of the distance and the performance of the corresponding distance kernel. However, it is hard to investigate the contribution of metricity in the accuracy since several factors take part in the classification task. The definiteness of a distance kernel seems to be related to the metricity of given distance -metric distances seem to lead to kernels that are closer to definiteness than those based on non-metric distances-, and the definiteness of a kernel may directly affect on the accuracy. In short, the relationship between metricity, definiteness and performance is not clear and is, thus, an interesting future direction of research.

To conclude, a summary of the reviewed methods of *Indefinite distance kernels* can be found in Table 2.5.

Table 2.5: Summary of indefinite kernel approaches

Authors	**Kernel**	**Classifier**	**Datasets**
Employing indefinite kernels			
Bahlmann *et al.* [133]	GDS_{DTW}	SVMs	1 (UNIPEN)
Jeong *et al.* [140]	GDS_{WDTW}	SVDD[1], SVMs	20 UCR
Kaya *et al.* [90]	GDS + alignment based distances	SVMs	40 UCR
Pree *et al.* [139]	Unspecified similarity based kernels	SVMs	20 UCR
Shimodaira *et al.* [135]	DTAK	SVMs	ATR
Gudmundsson *et al.* [84]	NDTW, GDS_{DTW}	SVMs	20 UCR
Dealing with the indefiniteness			
Jalalian *et al.* [81]	GDS_{DTW}	P-SVM	20 UCR
Regularization			
Chen *et al.* [141]	GDS_{DTW}, GDS_{ERP} , GDS_{TWED}	KSRC[2]	16 UCR
Analyzing the indefiniteness			
Lei *et al.* [142]	GDS_{DTW}	SVMs	4 UCR
Zhang *et al.* [86]	GDS_{ERP}, GDS_{TWED}	SVMs	20 UCR

2.2.3.3 Definite distance kernels

We have included in this section those methods that construct distance kernels for time series which are, by definition, PSD. First of all, we want to remark that there are other kernels for time series in the literature that are PSD but have not been included in this review. We have only incorporated those kernels based on time series distances and, in particular, those which construct the kernel functions directly on the raw series. Conversely, the Fourier kernel [131] computes the inner product of the Fourier expansion of two time series, and hence, does not compute the kernel on the raw series but on the Fourier expansion of them. Another example is the kernel by Gaidon *et al.* [154] for action recognition, in which the kernel is constructed on the auto-correlation of the series. There are also smoothing kernels that smooth the series with different techniques and then define the kernel for those smoothed representations [155, 156, 157, 158]. On the contrary, we will focus on those that define a kernel directly on the raw series. Regarding those included, all of them aim to introduce the concept of time elasticity directly within the kernel function by means of a distance, and we can distinguish two main approaches: in the

[2] Support Vector Data Descriptor [146, 153].
[2] Kernel Sparse Representation based Classifiers [149].

first, the concept of the alignment between series is exploited, while in the second, the direct construction of PSD kernels departing from a given distance measure is addressed.

Xue *et al.* [159] proposed the Altered Gaussian DTW (AGDTW) kernel, in which, first, the alignment that minimizes the Euclidean distance between the series is found, as in DTW. For each pair of time series T_i and T_j, once this alignment is found, the series are modified to this alignment resulting in ${T_i}'$ and ${T_j}'$. Then, if l is the maximum length of both series, the AGDTW kernel is defined as follows:

$$\kappa_{AGDTW}(T_i, T_j) = \sum_{s=1}^{l} \exp\left(-\frac{||{T_i}'_s - {T_j}'_s||^2}{\sigma^2}\right)$$

Since AGDTW is, indeed, a sum of Gaussian kernels, they provide the proof of the definiteness of the proposed kernel.

There is another family of methods that also exploits the concept of alignment but, instead of considering just the optimal one, considers the sum of the scores obtained by all the possible alignments between the two series. Cuturi *et al.* [17] claimed that two series can be considered similar not only if they have one single good alignment, but rather if they have several good alignments. They proposed the Global Alignment (GA) kernel that takes into consideration all the alignments between the series and provide the proof of its positive definiteness under certain mild conditions. It is worth mentioning that they obtain kernel matrices that are exceedingly diagonally dominant, that is, that the values of the diagonal in the matrix are many orders of magnitude larger than those out of the diagonal. Thus, they use the logarithm of the kernel matrix because of possible numerical problems. That transformation makes the kernel indefinite (even if it is not indefinite per se), so they apply some kernel regularization to turn all its eigenvalues positive. However, since the kernel they obtain is PSD and it is because of the logarithm transformation that it becomes indefinite, it has been included within this section. In [144], the author elaborates on the GA kernels, give some theoretical insights, and introduce an extension called Triangular Global Alignment (TGA) kernel, which is faster to compute and also PSD.

There is another kernel that takes a similar approach. In their work about periodic time series in astronomy, Wachman *et al.* [160] investigate the similarity between just shifted time series. In this way, they define a kernel that takes into consideration the contribution of all possible alignments obtained by employing just time shifting:

$$K_{shift}(T_i, T_j) = \sum_{h=1}^{l} e^{\gamma \langle T_i, T_{j+h} \rangle}$$

where $\gamma \geq 0$ is a user-defined constant. In this way, the kernel is defined by means of a sum of inner products between T_i and all the possible shifted versions of T_j with a shift of h positions. The authors provided the proof of the PSD of the proposed kernel.

On the other hand, there are methods that, instead of focusing on alignments, address the construction of PSD kernels departing from a given distance measure. These methods can be seen as refined versions of the GDS kernel in which the obtained kernel is PSD. Marteau *et al.* [161] elaborate on the indefiniteness of GDS kernels derived from elastic measures, even when such measures are metrics. As previously mentioned, metricity is not a sufficient condition to obtain PSD kernels. They postulated that elastic measures do not lead to PSD kernels due to the presence of *min* or *max* operators in their definitions, and define a kernel where they replaced the *min* or *max* operators by a sum ($\sum$). In [162], these same authors define what they called an elastic inner product, *eip*. Their goal was to embed the time series into an inner product space that somehow generalizes the notion of the Euclidean space, but retains the concept of elasticity. They provide proof of the existence of such a space and showed that this *eip* is, indeed, a PSD kernel. Since any inner product induces a distance [163], they obtained a new elastic metric distance δ_{eip} that avoids the use of *min* or *max* operators. They evaluated the obtained distance within a SVM by means of the $\text{GDS}_{\delta_{eip}}$ kernel, in order to compare the performance of δ_{eip} with the Euclidean and DTW measures. Their experimentation showed that elastic inner products can bring a significant improvement in accuracy compared to the Euclidean distance, but the GDS_{DTW} kernel outperforms the proposed $\text{GDS}_{\delta_{eip}}$ in the majority of the datasets.

They extended their work in [82] and introduced the Recursive Edit Distance Kernels (REDK), a method to construct PSD kernels departing from classical edit or time-warp distances. The main procedure to obtain PSD kernels is, as in the previous method, to replace the *min* or *max* operators by a sum. They provided the proof of the definiteness of these kernels when some simple conditions are satisfied, which are weaker than those proposed in [17] and are satisfied by any classical elastic distance defined by a recursive equation. Note that, while in [162] the authors define an elastic distance and construct PSD kernels with it, in [82] the authors present a method to construct a PSD kernel departing from any existing elastic distance measure. As such, the REDK can be seen as a refined version of the GDS kernel which leads to PSD kernels. In this manner, they proposed the REDK_{DTW}, REDK_{ERP} and REDK_{TWED} methods and compare their performance with the corresponding distance substitutions kernels GDS_{DTW}, GDS_{ERP} and GDS_{TWED}. An interesting result they reported is that REDK methods seem to improve the performance of non-metric measures in particular. That is, while the accuracies of REDK_{ERP} and REDK_{TWED} are slightly better than the accuracies of GDS_{ERP} and GDS_{TWED}, in the case of DTW the improvement is really significant. In fact, they presented some measures to quantify the deviation from definiteness of a matrix and showed that while GDS_{ERP} and GDS_{TWED} are *almost definite*, GDS_{DTW} is rather far from being definite. This makes us wonder if metricity implies proximity to definiteness, and in addition, if accuracy is directly correlated to the definiteness of the kernel.

Furthermore, they explored the possible impact of the indefiniteness of the kernels on the accuracy by defining several measures to quantify the deviation from definiteness based on eigenvalue analysis. If $\mathbf{D}_\delta$ is a distance matrix, $\text{GDS}_{\mathbf{D}_\delta}$ is PSD if and only if $\mathbf{D}_\delta$ is negative definite [164], and $\mathbf{D}_\delta$ is negative definite if it has a single positive eigenvalue. In this manner, the authors studied the deviation from definiteness of some distance matrices, and stated that when the distance matrix $\mathbf{D}_\delta$ was far from being negative definite, the REDK_δ outperforms the GDS_δ kernel in general, while when the matrix is close to negative definiteness, REDK_δ and GDS_δ perform similarly.

Recently, Wu *et al.* [101] introduced another distance substitution kernel, called D2KE, that addresses the construction of a family of PSD kernels departing from any distance measure. It is not specific for time series but in their experimentation they include a kernel for time series departing from the DTW distance measure. Their kernel employs a probability distribution over random structured objects (time series in this case) and defines a kernel that takes into account the distance from two series to the randomly sampled objects. In this manner, the authors point out that the D2KE kernel can be interpreted as a soft version of the GDS kernel, which is PSD. Their experimentation on four time series datasets showed that their D2KE_{DTW} kernel outperforms other distance based approaches such as 1-NN or GDS_{DTW} both in accuracy and computational time.

To conclude this section, a summary of the reviewed methods on *Definite distance kernels* can be found in Table 2.6.

Table 2.6: Summary of definite distance kernels

Authors	**Kernel**	**Classifier**	**Datasets**
Xue *et al.* [159]	AGDTW	KSRC, SVMs	4 UCR
Cuturi *et al.* [17]	GA	SVMs	TI46[1]
Cuturi *et al.* [144]	TGA	SVMs	5 UCI
Marteau *et al.* [162]	$\text{GDS}_{\delta_{eip}}$	SVMs	20 UCR
Marteau *et al.* [82]	REDK_{DTW}, REDK_{ERP}, REDK_{TWED}	SVMs	20 UCR
Wu *et al.* [101]	D2KE	SVMs	3 UCI + 1 own
Wachman *et al.* [160]	K_{shift}	SVMs	Astronomy

[1] TI46 speech dataset [165].

2.3 Computational cost

An important aspect that has not been addressed in depth when presenting the taxonomy is the computational cost of the methods included. The time complexity of the classification methods, in general, is dominated by the learning phase and depends on the size of the dataset from which the model is learnt; in distance based classification, in addition to the size of the dataset -understood as the number of instances-, the complexity of both the learning and prediction phases also depends on the computational cost of the employed distance measure. At the same time, the cost of the distance measure also highly depends on the lengths of the series we are working with. In this way, many time series distances, especially the most commonly employed measures (DTW, ERP, TWED...), are characterized by a quadratic complexity on the length of the series, which results in methods which are very time consuming for cases in which the length of the series is large. In this context, many of the methods that employ common time series distance measures usually turn out to be too time consuming for real world applications. Even if this is so, and even if some of the reviewed works experimentally evaluate the running times of their methods or aim at speeding up their learning processes, most of them do not even address this issue. Thereby, in this section, a brief overview of the complexity of distance based TSC methods is provided in order to review the computational specificities of the methods in each category of the taxonomy.

First of all, it is important to highlight that one of the most significant differences between distance based and non-distance based classification methods (from the point of view of the computational cost) is the time complexity of the prediction phase. In non-distance based methods, normally, the learning phase depends on the size of the training dataset but, once the model is learnt, the prediction of unlabeled instances does not depend on this dataset and is usually independent from the size of the dataset. In distance based classification, on the contrary, both the learning and the prediction stages computationally depend on the size of the dataset and on the chosen distance measure, so they must both be taken into account. Thereby, from now, we are going distinguish between the computational cost of the learning and the prediction phases of the reviewed methods. Note that we are going to provide a general computational time analysis of the methods but there are exceptions which do not exactly fit into the computational characterization that we provide for each category.

In the case of the methods based on the 1-NN classifier, there is no learning phase and the computational cost of prediction is determined by the size of the dataset and the complexity of the distance measure (which, in turn, depends on the lengths of the series). For instance, the distances DTW, ERP or TWED have a complexity of $O(l^2)$, where l is the length of the longest time series, while the cost of the Euclidean distance is $O(l)$. As such, the computational cost of predicting an unlabeled time series using the DTW distance, for instance, is $O(l^2 n_{tr})$ (where n_{tr} is the size of the training dataset), since the n_{tr} distances between the unlabeled series and the series in the dataset have to computed. The approach adopted by most researchers to accelerate this process is to speed up the computation of the employed distance measure, for example by using the fast lower bound for the DTW [103], which reduces the complexity of the distance to $O(l)$ [4].

Regarding the methods that exploit distances as features, it is important to note that the computation of the distances and the learning/prediction of the classifier are two independent steps with their corresponding computational costs. In the learning stage, first, the pairwise distances between all the series in the dataset are computed -as a preprocessing step- to obtain the distance features, which are then used as input for learning the classifier. We focus only on the complexity of the first step, which is specific for distance based methods: the computational cost of this step depends on the complexity of the distance measure, as well as on the size of the training dataset. For instance, computing the DTW distance matrix of the n series in a dataset has a complexity of $O(n^2l^2)$. For prediction, the distances from the new unlabeled series to all the series in the training dataset have to be computed also as a preprocessing step. Then, the obtained distance features are introduced into the classifier to predict the unknown label. As in the previous case, the distance computation depends on the complexity of the distance measure and the size of the dataset. As such, an important drawback is that, for cases with large datasets or high time consuming distances, the prediction can become unrealistically time consuming. In view of this, several approaches have been taken to mitigate the effect of these two factors: Goebel *et al.* [102] employed the fast lower bound to speed up the computation of the distances (from quadratic to linear), while Iwana *et al.* [83] and Goebel *et al.* [104] address the issue of reducing the dimension of the distance matrix that is used as input to learn the model. The former proposed using time series prototypes and used the distances to them instead of calculating the entire distance matrix, while the latter applied PCA in order to reduce its dimensionality.

In the shapelet based approaches, there are some preprocessing steps in order to obtain the features before the application of the classifier. In the learning phase, first, a shapelet discovery stage is carried out in which the *best* shapelets are learnt and, then, the pairwise distances between the series in the dataset and the obtained shapelets are computed. The initially proposed shapelet discovery technique takes $O(l^4{n_{tr}}^2)$, which turns out to be very time consuming for real world applications. As such, over the years, many methods have been proposed to speed up this search [27, 106, 107, 108]. Once the shapelets have been discovered, the computational cost of calculating the pairwise distances between series and shapelets depends on the complexity of the distance, the number of series and the number of shapelets. The distance between a series and a shapelet is computed using the Euclidean distance most of the times -which has a complexity of $O(l)$-, so, once the shapelets are learnt, the distance computation has a complexity of $O(ln_{tr}m)$, where m is the number of shapelets. This number is determined in the shapelet discovery process, which usually involves techniques such as candidate pruning or shapelet clustering in order to reduce the amount of shapelets [26, 28]. In the prediction phase, the shapelet based methods require a preprocessing step that involves a distance computation between the new unlabeled series and the learnt shapelets, which has $O(lm)$ complexity in the case of the commonly employed Euclidean distance.

For the embedding based methods, the pairwise distances between the series in the dataset have to be computed before they are embedded into another space. In the learning, this process has $O(l^2 {n_{tr}}^2)$ complexity (with the DTW distance, for example), while the complexity of the embedding process depends on the specific technique employed. Hayashi *et al.* [119] and Mizuhara *et al.* [120], for instance, applied multidimensional scaling, pseudo-Euclidean space embedding, and Euclidean space embedding by the Laplacian eigenmap technique, but they do not specify the computational cost of these methods so it is hard to draw conclusions. Lei *et al.* [123] and Lods *et al.* [87], employed gradient descent based techniques, and, while the formers do not specify the complexity of the method, the latter points out that the complexity of the learning phase is quite high. Then, the obtained features are introduced into a classifier. In prediction, the pairwise distances between the unlabeled series and the training dataset have to be computed, which has a complexity of $O(l^2 n_{tr})$ for cases using DTW [119, 120].

In the methods that employ distance kernels, there is no preprocessing step and the series are directly used as input to the given kernel method. However, the distance kernels are derived from time series distances, so the computational cost of the kernel methods is mainly dominated by the computation of the kernel matrix (analogous to the distance matrix). In particular, this computation depends on the complexity of the distance measure from which the kernel is derived as in [133, 140, 141] methods. As such, the distance substitution kernels derived from DTW, ERP, EDR or TWED are computationally more expensive ($O(l^2 {n_{tr}}^2)$) than the Gaussian RBF kernel ($O(l {n_{tr}}^2)$), for instance. In the prediction phase, the kernel matrix -computed in the learning phase- is extended with the pairwise values between the unlabeled series and the series in the dataset, which has the same complexity as the previous 1-NN or global distance features methods.

Apart from the distance substitution kernels, the review includes other distance kernels that are specific for time series and whose computational cost has to be analysed more in depth. The kernel proposed in [17] considers all the alignments instead of only the optimal one and, thus, has a complexity of $O(l^2 {n_{tr}}^2)$ in the learning phase and $O(l^2 n_{ts})$ in prediction phase, where n_{ts} in the number of time series in the test set. In view of this, the same authors proposed another version of the kernel [144], which, by means of adding additional constraints on the allowed alignments, is faster than the original kernel but equally accurate. In the definite kernel derived from an elastic inner product proposed in [162], the computational cost is evaluated experimentally and the authors show that the proposed elastic kernel has a complexity of $O(l)$. As such, the learning phase takes $O(l{n_{tr}}^2)$, while the prediction phase $O(ln_{ts})$. In other words, they obtained an elastic kernel for time series that is characterized by a linear complexity instead of the quadratic complexity derived from the traditional elastic distances, which is a significant improvement.

From a general point of view, it is hard to draw accurate comparative results between the methods presented due to their variants and the lack of experimental computational time results available in the published works. Wu *et al.* [99] carried out the most comprehensive evaluation of the computational cost of several distance based TSC methods until now. They first compare their DF$_{RF}$ distance features method with two embedding methods: the method proposed in [120], and the one in [87], concluding that their method outperforms the other two, both in accuracy and in computational time. In addition, two variants of their method are also evaluated on 16 UCR datasets against other baseline distance based TSC approaches (1-NN with DTW, the GA kernel [17] and DF$_{DTW}$ [16]); the first variant of their method outperforms the other approaches in accuracy but involves a high computational cost, while the second variant achieves competitive accuracies, significantly reducing the required computational time.

To summarize, distance based TSC methods have usually quadratic complexity both in the length of the series and in the size of the dataset, due to the common use of elastic measures. In this context, if the series are long enough or the size of dataset is large, the methods can become too time consuming for real world applications. As such, it is an important aspect to be considered. Some of the methods take this into account and evaluate the running time of their method but, in general, in our opinion, it has not been addressed enough. There are some attempts to speed up the distance based methods [83, 102, 104, 144, 162] but it is still a direction in which there is considerable room for improvement. In addition, we think that a comprehensive comparison of the running times of the methods would be a great contribution as future work.

2.4 Discussion

In this Chapter, we have presented a review on distance based TSC and have included a taxonomy that categorizes all the discussed methods depending on how each approach uses the given distance. We have seen that from the most general point of view, there are three main approaches: those that directly employ the distance together with the 1-NN classifier, those that use the distance to obtain a new feature representation of the series, and those which construct kernels for time series departing from distance measure. The first approach has been widely reviewed, so we refer the reader to [76, 78, 166] for more details about the discussion.

Regarding the methods that employ a distance to obtain a new feature representation of the series, these approaches have been considerably studied for time series as it bridges the gap between traditional classifiers (that expect a vector as input) and time series data, taking advantage of the existing time series distances. In addition, some methods within this category have outperformed existing time series benchmark classification methods [16]. Note that distance features can be seen as a preprocessing step, where a new representation of the series is found which is independent of the classifier. Depending on the specific problem, these representations vary and can be more discriminative and appropriate than the original raw series [28]. As such, an interesting point that has yet to be addressed is to compare the different transformations of the series in terms of how discriminative they are for classification.

Nevertheless, learning with the distance features can often become cumbersome depending on the size of the training set and a dimensionality reduction technique must be applied in many cases in order to lower the otherwise intractable computational cost. Some of the methods [83, 104] reduce the dimensionality of the distance matrix once it is computed. Another direction focuses on time series prototype selection [83], that is, selecting some representative time series in order to compute only the distances to them instead of to the whole training set. It is worth mentioning that there has been some work done in this context in other dissimilarity based learning problems [167] but it is almost unexplored in TSC. Due to the interpretability of the time series and, in particular, of their prototypes, we believe that this is a promising future direction of research.

Another feature based method consists of embedding. The embedded distance features have only been employed in combination with linear classifiers [120] or the tree based XGBoost classifier [87], which, in our opinion, do not take direct advantage of the transformation. The main idea of the embedded features is that if the Euclidean distances of the obtained features are computed, the original time series distances are approximated. In this way, we believe classifiers that compute Euclidean distances within the classification task (such as the SVM with the RBF kernel, for instance) will profit better from this representation. In addition, in the particular case of kernel methods, the use of embedded features can be seen as a kind of regularization; the RBF kernel obtained from the embedded features would be a definite kernel that approximates the GDS indefinite kernel.

As already pointed out, the third way of using a distance measure is trying to construct kernels departing from these existing distances. However, these distances do not generally lead to PSD kernels. Both distance features and distance kernel approaches are not specific for time series, and some work has been done to compare the benefits of each approach in a general context. Chen *et al.* [92] mathematically studied the influence of distances features and distances kernels within SVMs in a general framework. In TSC, Gudmundsson *et al.* [84] and Jalalian *et al.* [81] address the problem of experimentally evaluating whether it is preferable to use distance features or distance kernels. Both works assert that the indefiniteness of the distance kernels negatively affects the performance, although their proposals are restricted to the DTW distance. It would be interesting to comprehensively compare these two approaches taking into account different distances, kernels and classifiers in order to draw more general conclusions.

The problem of the definiteness of a kernel has been widely addressed within the methods in this review. Note that the definiteness of a kernel guarantees the mathematical foundations of the kernel method and, therefore, it seems natural to think that definiteness and performance are correlated, which is the assumption of almost all the methods. Some authors confirm that the performance is still good and do not care about the indefiniteness of the kernels, while, in general, the research focuses mainly on trying to somehow deal with the indefiniteness of the kernels. Isolating the contribution of the definiteness of a kernel to the performance is rather challenging due to the many other factors (optimization algorithm or the choice of the kernel function) that also affect it. However, since the relation between definiteness and accuracy is a general matter -not specific for time series, and in fact, not specific for distance kernels-, a promising future direction would be to evaluate whether there exists or not a direct correlation between them.

Within the methods that try to deal with the indefiniteness there are two main directions. The first uses kernel based classifiers that can handle indefinite kernels. This approach is almost unexplored in TSC, since only the P-SVM in [81] has been applied, achieving very competitive results. Indeed, there are some studies on learning with indefinite kernels from a general point of view [132], and considering that indefinite kernels appear often within TSC, this approach may be interesting future work.

The second approach, called kernel regularization, aims to adapt the indefinite kernel to be PSD. As in the previous direction, this is also an almost unexplored approach for time series. Only eigenvalue analysis has been applied with ambiguous results. Chen *et al.* [168] used eigenvalue regularization techniques but they do not evaluate the regularization itself, while Gudmundsson *et al.* [84] argued that the method after kernel regularization achieves lower performance than the method with the indefinite kernel. One of the main shortcomings of this specific regularization is that it is data dependent, and, in addition, the consistent treatment of new test samples is not straightforward. As previously mentioned, it is not clear whether regularization is helpful or whether the new kernel becomes so different from the initial one that the information loss is too big; this is an open question which has not been studied in detail.

As previously mentioned, another direction focuses on a better understanding of the indefiniteness of these kernels. Concerning the GDS kernels, which are distance kernels valid for any type of data, the first attempt in the time series domain was to define kernels departing from distances that are metrics. Although it has been proven that the metric property does not guarantee the definiteness of the induced GDS kernel, Zhang *et al.* [86] argued that the performance of metric distance kernels is significantly better than those defined with non-metric distances, suggesting that kernels with metric distances are closer to definiteness. In addition, Marteau *et al.* [82] conjecture that the reason of the indefiniteness is the presence of *min* or *max* operators in the recursive definition of time series distance measures. An interesting observation is that these discussions arise from time series distances but are, regardless, general issues concerning the characteristics of a distance measure and the derived GDS kernel. Even if the mentioned works address the relation between metricity and definiteness, this connection is not yet clear. It is also an interesting future research direction due to the generalizability of the problem and the possible applications.

Cuturi *et al.* [17], by contrast, focused on the specific challenge of constructing ad-hoc kernels for time series. As such, they found a direct way of constructing PSD kernels that take into account the time elasticity by defining a kernel that does not consider just the optimal alignment between two series but, instead, considers all the possible alignments. Moreover, given an elastic distance measure defined by a recursive equation, Marteau *et al.* [14] address the construction of distance based PSD kernels. Their kernel can be seen as a particular case of GDS kernel for elastic measures that become PSD by replacing the *min* or *max* operators in the recursive definition of the distance by a sum. By using this *trick*, they obtain kernels for time series that take into account time elasticity and are also PSD. Their comprehensive experimentation shows that SVM based approaches which use these kernels clearly outperform the 1-NN benchmark approaches, even for the DTW distance. Furthermore, they reported that the REDK kernel brings significant improvement in comparison with the GDS kernel, especially when the kernel matrices of the GDS kernels are far from definiteness, which in their particular case corresponds to the non-metric measures. However, they experimented with just two metric and one non-metric measures which is not enough to draw strong conclusions.

It is also worth mentioning that many methods introduced in the taxonomy are not specific for time series, but become specific when a time series distance is employed. In particular, only the methods that are based on shapelets and the methods that construct kernels for time series considering the concept of alignment between series are specific for time series. The rest of the methods are general methods of distance based classification for any type of data. An interesting observation is that questions or problems arising for time series can be extrapolated to a general framework. In the same manner, some of the presented approaches are specific for some classifiers (1-NN, kernel methods), while others can be used in combination with any classifier. Also note that many of the methods, such as those which employ *global distance features*, *embedded features* or *indefinite distance kernels*, are directly applicable in the case of multi-variate or streaming time series, provided a suitable distance for this kind of series is defined. The extension of these methods for multi-variate or streaming time series could be a possible future direction. It would be interesting also to extend other methods, such as the shapelet based methods or the ad-hoc definite kernels, to these kind of time series, since, in these cases, the adaptation of the methods by itself would be a great contribution.

To conclude, note that in contrast to the number and variety of existing kernels for other types of data, there are rather few benchmark kernels for time series in current literature [127]. Therefore, we would like to highlight the value of these kernels for time series, especially those that are able to deal with the temporal nature of the series and are PSD.

3

Time Series Classifier Recommendation by a Meta-Learning Approach

3.1 Introduction

In TSC, there is a great number of competitive and diverse classifiers (see Section 1.1). In this context, from a pragmatic user's point of view, choosing a suitable classifier for a certain problem is a difficult task, since there is a lack of orientation. As introduced in Section 1.2, meta-learning is field that has been focused on automatic model recommendation and, in the case of time series data, has been used to design automatic forecasting method recommendation [43, 44, 45, 46] or similarity measure selection for clustering [47]. In this Chapter, we will exploit meta-learning for proposing an automatic time series classifier recommendation system.

Since state-of-the-art meta-features are designed for non-ordered data, we propose a novel set of 24 specific landmarkers for TSC, which meet the requirements of being simple, fast and good predictors of the performance of state-of-the-art TSC algorithms. We validate the proposed landmarkers for classifier recommendation considering several types of meta-targets. Finally, we explore the hierarchical inference of meta-targets; some types of meta-targets are more fine-grained than others and, from the prediction of these meta-targets, less fine-grained meta-targets can be inferred. As such, we experimentally compare the two approaches for obtaining a given meta-target: by direct prediction of the meta-target and by inference from the predictions of more fine-grained meta-targets.

The rest of the Chapter is organized as follows. In Section 4.3, our method for time series classifier recommendation is described in two parts: meta-attributes and meta-targets. The hierarchical inference of meta-targets is presented in Section 3.3, while the experimentation is exposed in Section 3.4. Lastly, the main conclusions and future works of this Chapter are presented in Section 3.5.

3.2 Time Series Classifier Recommendation

The main objective addressed in this Chapter is to propose a time series classifier recommendation (TSCR) system. Algorithm recommendation, as stated in [35], aims at ”saving time by reducing the number of algorithms tried out on a given problem with minimal loss in the quality of the results obtained when compared to the best possible ones”.

From a practical point of view, the scheme of a TSCR system is shown in Figure 3.1. Given a repository of supervised time series datasets and a set of candidate classifiers, firstly, the accuracies of the candidate classifiers in those datasets are calculated by evaluating each classifier in each dataset. Then, the accuracies of the classifiers are employed to construct the meta-target. Secondly, a vector of meta-attributes is extracted from each dataset. Lastly, the meta-attributes and meta-target are used to build the TSCR system employing a meta-learner. Note that this is a supervised learning scenario and each meta-target type requires a specific meta-learner. In this way, for a new time series dataset -in which the accuracies of the candidate classifiers are unknown-, the meta-attributes are extracted and passed to the TSCR, which outputs a classifier recommendation based on the chosen meta-target type. In the following sections, the meta-attributes proposed and meta-targets considered in this work are presented.

3.2.1 Meta-Attributes

The characterization of a dataset is, probably, one of the challenges that has attracted most attention in meta-learning, since the performance of a meta-learning system highly depends on this characterization [169]. Indeed, meta-attributes need to fulfil two crucial requirements: on the one hand, they need to be fast to compute, and, on the other hand, they need to contain useful information for discriminating between the performances of the candidate classifiers. The definition of the meta-attributes highly depends on the specific task for which the meta-learning system is built.

In the case of time series data, as mentioned earlier, there are rather few works that have addressed the development of meta-attributes [43, 44, 45, 46, 47, 170]. In fact, most of these works focus on time series forecasting algorithm recommendation [43, 44, 45, 46, 170]. In contrast to TSC or clustering, where a set of time series is needed, a time series forecasting problem is composed of a single time series. This is an important distinction from the point of view of meta-learning, since, in the former case, meta-attributes correspond to single time series, while in the latter, meta-attributes correspond to a time series dataset. As far as we know, the only work in which time series dataset characterization has been addressed is [47], where a set of meta-attributes are proposed with the purpose of similarity measure selection for time series clustering. However, this characterization is defined for unsupervised time series

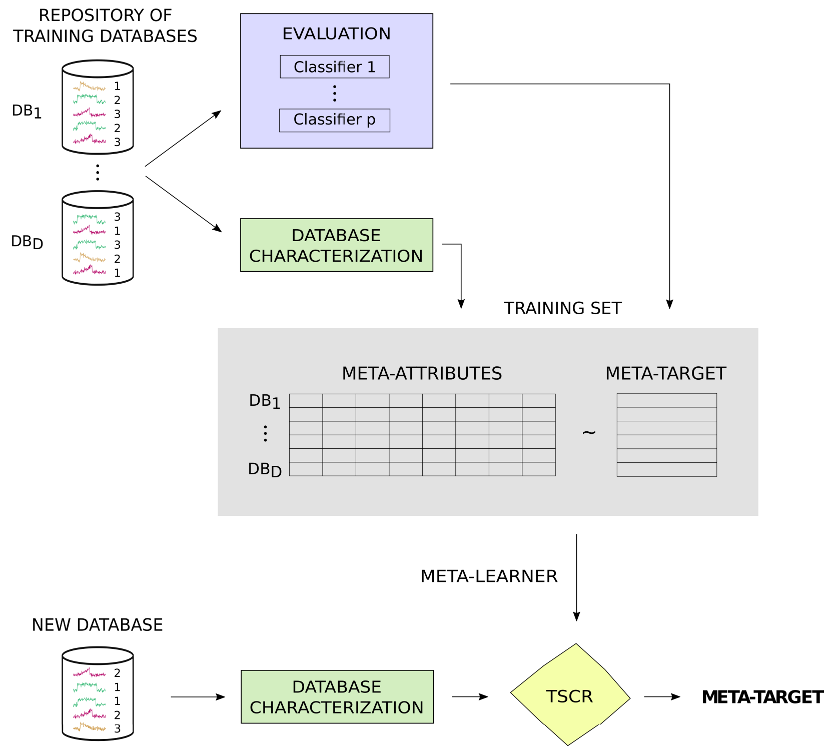


Fig. 3.1: Meta-learning scheme for TSC algorithm recommendation.

datasets and can not be applied in our case, which focuses on supervised time series datasets. This contribution focuses on the definition of meta-attributes for supervised time series datasets.

In this context, most of the state-of-the-art meta-attributes are designed for non-ordered instances and they become meaningless for describing time series datasets. The most generalizable meta-attributes are landmarkers, since they are quick estimators of the accuracies of the candidate classifiers and, hence, can be defined for any type of task and data. As such, we propose a set of landmarker based meta-attributes for supervised time series datasets.

The design of landmarkers totally depends on the candidate classifiers considered in the meta-learning problem. In our case, the candidate classifiers have been chosen from those that appear in the work by Bagnall *et al.* [8], since the results obtained in their extensive experimentation are publicly available [171] -as well as a Weka-compatible Java toolbox, *tsml* [71], with the implementation of most of the classifiers included in their work-. In this way, we choose as candidate classifiers those that are included in [8] and implemented in the *tsml* or *sktime* (and the extension for deep learning *sktime-dl*) [72] toolboxes. These classifiers are: C4.5 decision tree (C45) [172], naive Bayes (NB) [173], Bayes Network (BN) [174], SVM with linear (SVML) [18] and quadratic kernel (SVMQ) [18], Rotation Forest (RotF) [175], Random Forest (RandF) [176] , Multilayer Perceptron (MLP) [177], 1-NN with Euclidean distance (NN), 1-NN with DTW distance (DTW) [10], 1-NN with Weighted DTW distance (WDTW) [140], 1-NN with Time Warp Edit distance (TWE) [14], 1-NN with Move–Split–Merge distance (MSM) [15], 1-NN with Complexity Invariant distance (NN_CID) [178], 1-NN with Edit Distance with Weal Penalty (ERP) [150], 1-NN with Derivative DTW (DD_DTW) [179], 1-NN with Derivative Transform distance (DTD_C) [180], Time Series Forest (TSF) [24], Fast Shapelets (FS) [27], Shapelet Transform (ST) [28], Bag of Patterns (BOP) [29], and Bag of SFA Symbols (BOSS) [30]. Additionally, even if they are not included in [8], we have added two new benchmark deep learning classifiers: Resnet [181] and InceptionTime [182] in order to have a more up-to-date set of candidate classifiers. In this way, the list of candidate classifiers includes 24 heterogeneous time series classifiers. The discarded classifiers from [8] are: Longest Common Subsequence (LCSS) [13] (the *tsml* implementation does not reproduce the results), Elastic Ensemble (EE) [183] (implementation with bugs), Time Series Bag-of-Features (TSBF) [184] (not implemented in *tsml*), Learned Pattern Similarity (LPS) [185] (implementation with bugs), DTW Features (DTW_F) [16] (code not available) and Collective of Transformation-Based Ensembles (COTE) [31] (it is an ensemble of several TSC methods, so from an algorithm recommendation point of view, it does not make sense to include it). Additionally, 3 classifiers have been discarded due to their high requirement of computational resources: Logistic [186] (which requires more than 50GB of RAM memory in the PigCVP dataset from the UCR repository), and Learn Shapelets [109] and SAX Vector Space Model (SAXVSM) [187] (which take more than 5 and 25 days to classify a single train/test split in the Crop dataset from the UCR, correspondingly). Note that some of the considered classifiers (ST and BOSS, for instance) are also ensembles, but they are ensembles of classifiers of the same nature, while COTE is an ensemble of many types of different classifiers.

Given a candidate classifier C_i, its landmarker L_i is a quick estimator of the accuracy obtained by C_i. This quick estimator is generally obtained in two ways: by running simplified versions of the candidate classifier [41] (algorithm reduction), or by running the original algorithm in a subsample of the data, obtaining the so-called *subsampling landmarkers* [42] (dataset reduction). In both types of reductions, the greater the reduction, the higher the loss of relation between the landmarker and the original algorithm [188], so striking a balance between the level of reduction and the relation with the original algorithm is a key aspect. Our approach exploits both types of reductions: first, subsampling landmarkers are applied by evaluating the classifiers on a subsample of the dataset. Given a time series dataset with n instances, we propose a stratified subsample that depends on n. The proportion to which the dataset is reduced, the subsample ratio (r), is shown in Table 3.1. We are dealing with supervised datasets so, in those datasets in which the subsample ratio does not permit a minimum number of instances of each class, the r specified in Table 3.1 is increased by steps of 0.1 until $n * r/k \geq 10$ or $r = 0.8$, where k is the number of classes.

Table 3.1: The subsample ratio (r) applied to a time series dataset with n instances (before the modification to ensure a minimum number of representatives of each class).

Intervals	Subsample ratio (r)
$n < 100$	0.80
$100 \leq n < 300$	0.60
$300 \leq n < 800$	0.40
$800 \leq n < 1500$	0.20
$1500 \leq n < 5000$	0.10
$5000 \leq N$	0.05

In the second step, an algorithm reduction is carried out for those classifiers that still take too long in the subsampled datasets. In order to identify the slow classifiers, we conduct the following analysis: we sort the 112 datasets from the UCR repository employed in this work by the dimension (defined by n*l, where n is the number and l the length of the series) and run the 24 classifiers in subsampled versions of the 5 largest datasets (*StarlightCurves*, *UWaveGestureLibraryAll*, *HandOutlines*, *MixedShapesRegularTrain*, and,*NonInvasiveFetalECGThorax1*)[1]. If a classifier takes more than 30 minutes in any of the aforementioned datasets, it is considered a slow classifier. In this way, we identify 7 slow classifiers: BOSS, DD_DTW, DTD_C, MLP, ST, ResNet and InceptionTime, to which algorithm reductions are carried out. Three types of algorithm reductions are proposed (summarized in Table 3.2) :

- Reducing the parameter range: in those cases in which a parameter is set by a grid search, this grid search is deleted and, by default, the value of the parameter is set to the middle of the search range (DD_DTW and DTD_C). In the R parameter of the LS, however, some values in the search range incur larger costs than others, so we set it to the computationally least expensive value.
- Reducing the iterative process/ensemble: for the classifiers that have an iteration process (MLP, ResNet and InceptionTime), we limited the number of iterations. An analogous reduction is carried out for the ensembles (BOSS).

[1] The computation times of the 24 classifiers in the 5 subsampled datasets are included in Appendix A.1.

- Setting a time limit: in the ST classifier, the training time can be directly limited by the user, so we use this characteristic to reduce the computation time.

Table 3.2: Algorithm reductions carried out to the slow classifiers.

Classifier	**Parameter**	**Default**	**Reduced**
BOSS	MaxEnsembleSize	500	100
DD_DTW	α	grid search in $\{0,0.01, \ldots, 1\}$	0.5
DTD_C	α	grid search in $\{0, 0.01, \ldots, 1\}$	0.5
MLP	NumEpochs	500	50
ST	time limit	unlimited	5 minutes
ResNet	NumEpochs	1500	200
InceptionTime	NumEpochs	1500	200

To sum up, given a supervised time series dataset, we propose a set of 24 landmarkers to describe the dataset: 17 of them are based on dataset reductions, while the rest are based on dataset and algorithm reductions.

3.2.2 Meta-target

The meta-target, also known as the recommendation, corresponds to the output of the recommendation system. Five types of meta-targets are considered in this work (summarized in Table 3.3): classifier accuracies, complete ranking, top-M ranking, best set and best classifier. Each meta-target type gives rise to a different TSCR system, in which both the meta-target in the training set and meta-learner shown in Figure 3.1 must be specifically designed. We will refer to the meta-learner associated to a given meta-target type as the specific meta-learner (SML). The SMLs employed for each meta-target type are summarized in Table 3.4. In the following lines, a brief overview of the considered meta-target types is presented.

Classifier accuracies: this meta-target provides the accuracies obtained by each candidate classifier in a given dataset. When building the training set for learning this TSCR system, the values of the meta-targets for the training instances are directly defined by the accuracies of the classifiers. This meta-target type gives rise to a multi-output regression problem, and the SML chosen for this problem is the linear multi-output regression. It is the most fine-grained meta-target type, and the user is free to choose among the candidate classifiers with the provided information.

Complete ranking: this meta-target provides an accuracy based ranking of the candidate classifiers in a given dataset. The values of the meta-targets in

Table 3.3: Summary of the considered meta-target types.

Meta-target	**Output type**
Classifier accuracies	C_1 C_2 … C_8 … C_P 0.91 0.87 … 0.95 … 0.72
Complete ranking	C_8 C_1 C_2 … C_5 1 2 3 … P
Top-M ranking	C_8 C_1 C_2 1 2 3
Best subset	$\{C_1 , C_8\}$
Best	C_8

the training set are obtained by converting the raw accuracies of the candidate classifiers into a ranking. In case of ties, the best ranking position of the tied classifiers is assigned to all the tied classifiers. This meta-target type requires a ranking learning strategy, and the SML employed in this case is the k-NN for rankings [32] (based on the Euclidean distance), commonly used in meta-learning. From the user point of view, ranking recommendation is less fine-grained than the classifier accuracies but still leaves the user with a great choice since information regarding all the classifiers is provided.

Top-M ranking: in this case, the meta-target corresponds to the partial ranking of the M best-performing classifiers in a given dataset. When building the training set, the values of the meta-targets are obtained by converting the classifier accuracies into a ranking and selecting the classifiers at the first M positions (M is predefined by the user). The ties are handled in the same manner as for complete rankings. Analogously to the case of complete rankings, the SML employed for this meta-target type is the k-NN for rankings based on the Euclidean distance. This meta-target type is less fine-grained than the complete ranking since only the ranking of the M best-performing classifiers is provided.

Best subset: this meta-target provides the set of best-performing classifiers in a given dataset. Establishing which algorithms perform well on a given dataset is not straightforward, and it is usually defined in relative terms. A commonly employed approach to obtain the values of the meta-targets in the training set consists of defining a margin [189], such that the classifiers with accuracies within this margin are considered *applicable* classifiers in the given dataset. We propose a margin based on the proposal in [189], but enhanced to consider the range of the accuracies obtained by the candidate classifiers in the given dataset:

$$\Big[a_{\max}, a_{\max} - W(a_{\max} - a_{\min})\Big) \tag{3.1}$$

where W is a user-defined parameter, while $a_{\max}$ and $a_{\min}$ are the maximum and minimum accuracies obtained by the candidate classifiers in the given dataset. Note that, the greater W, the wider the interval and, hence, the larger the set of applicable classifiers. The best set meta-target gives rise to a multi-label classification problem, and, analogous to the ranking meta-targets, a k-NN version based on the Euclidean distance- for multi-label classification [190] is employed as SML. This meta-target type is less fine-grained than the top-M meta-target types since it does not provide a ranking, but an unordered set of the applicable classifiers.

Table 3.4: The specific meta-learner employed for each meta-target.

Meta-target	**SML**
Classifier accuracies	Linear multi-output regression
Complete ranking	k-NN for rankings
Top-M rankings	k-NN for rankings
Best set	k-NN for multi-label
Best	k-NN for multi-class

Best: this meta-target provides the best classifier among the candidate classifiers. The values of the meta-targets in the training set are obtained in a straightforward manner. The best classifier meta-target gives rise to a multi-class standard classification problem, and, for the sake of consistency, a k-NN for multi-class problems is used. This meta-target type is the least fine-grained meta-target, since it only provides the best performing classifier and gives no information about the rest of the classifiers.

3.3 Hierarchical inference of meta-targets

In this section, we explore the hierarchical relationship between the different meta-target types. Some meta-target types are more fine-grained than others and this fact gives rise to a hierarchy. By using this hierarchy, instead of building a specific TSCR system for each meta-target type, we investigate the approach of inferring meta-targets from the predictions made by more fine-grained TSCR systems.

Figure 3.2 shows the scheme of the hierarchical inference of the meta-target MT_j from the meta-target MT_i, where MT_i is a more fine-grained meta-target than MT_j. In Section 3.2.2, we showed how $\widetilde{\mathrm{MT}}_i$ and $\widetilde{\mathrm{MT}_j}(1)$ are predicted by the associated TSCR_i and TSCR_j systems, correspondingly. In the hierarchical inference, $\widetilde{\mathrm{MT}_j}(2)$ is obtained by transforming the prediction $\widetilde{\mathrm{MT}}_i$ made by TSCR_i.

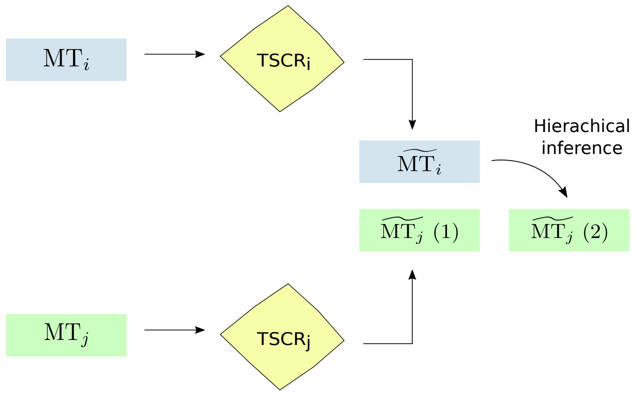

MT_i
TSCRi
Hierachical inference
$\widetilde{\mathrm{MT}}_i$
$\widetilde{\mathrm{MT}}_j$ (1)
$\widetilde{\mathrm{MT}}_j$ (2)
MT_j
TSCRj

Fig. 3.2: Scheme of the two approaches for obtaining $\widetilde{\mathrm{MT}_j}$: (1) by employing the TSCR_j associated to MT_j and (2) by hierarchical inference from the prediction of the MT_i meta-target.

More fine-grained information is learnt by TSCR_i than by TSCR_j, so $\widetilde{\mathrm{MT}_j}(2)$ could be expected to be more accurate than $\widetilde{\mathrm{MT}_j}(1)$. However, from a learning point of view, the prediction of MT_i is a harder task, and, therefore, more errors could be committed, so the comparison of these two approaches can shed some light on which learning strategy should be adopted when building a recommendation system for TSC algorithms.

As far as we know, despite its interest, this point of view is still almost unexplored in the meta-learning community. In [191], the author inferred the expected ranking and the expected best classifier from the predicted classifier accuracies, but reported that the inferred versions seem to perform slightly worse than the specific strategies. Bensusan *et al.* carried out a similar experiment in [192], where a ranking is inferred from the predicted classifier accuracies. In the same manner, the authors reported that even if the regression models obtained low errors, the results of the inferred rankings were not very promising. Moreover, not all the possible hierarchical dependencies are studied, but only a few pairs. In this work, we explore all the feasible hierarchical inferences for the considered meta-target types.

The feasible inferences for each meta-target type are shown in Table 3.5. From classifier accuracies, all the meta-targets can be inferred: complete ranking, top-M ranking, best set and best classifier. Complete ranking, top-M ranking and best classifier transformations are straightforward, while, in the best set transformation, the same procedure described in Section 3.2.2 is followed. A complete ranking can be directly transformed into a top-M ranking and best classifier. The best set, however, can not be inferred since the criterion employed for the transformation is not applicable in rankings. In the same manner, a top-M ranking can not be transformed into a best set, and from this meta-target, only the best classifier can be inferred. From the best set, the only meta-target that could be inferred (best classifier) is not feasible, due to the unordered nature of the best set. The best classifier meta-target is the least fine-grained meta-target, and hence, no meta-target can be inferred from it.

Table 3.5: Feasible hierarchical inferences.

Meta-target	**Feasible inferences**
Classifier accuracies	Complete ranking, Top-M ranking Best set, Best
Complete ranking	Top-M ranking, Best
Top-M ranking	Best

3.4 Experimentation

The experimentation is divided into four parts: in the first part, the experimental set up is introduced, as well as the evaluation procedure employed for the proposed methods. In the second part, the results obtained by the landmarkers are analysed and compared to those obtained by the original classifiers. The experimental evaluation of the proposed TSCR systems is presented in the third part. Lastly, the hierarchical inference of the meta-targets is experimentally studied[1].

3.4.1 Experimental set-up

3.4.1.1 Datasets

The experimentation has been carried out employing datasets from the UCR repository [168]. In this work, we have decided to use datasets of univariate equal-length time series; most of time series classifiers can only be applied in datasets of this type and the need for meta-learning becomes more evident in a context with a large number of candidate classifiers. As such, the 112 datasets from the UCR with these characteristics have been used.

[1] The TSCR systems, as well as the code for reproducing all the experiments presented in this Chapter are available at https://gitlab.bcamath.org/aabanda/tscr.

3.4.1.2 Classifiers

All the experiments have been carried out in python with the help of the *sklearn* library. For the classifier accuracies meta-target, the linear multi-output regression included in the *sklearn* library is used. We set the number of neighbours in the k-NN to 5 after preliminary experiments. For the top-M rankings, we explore the results for M=3, M=5 and M=10. In the best subset prediction, we need to establish the width of the margin for the best classifier, W, beforehand. Figure 3.3 shows the number of labels by instance for the different values of W we have considered. It can be seen that the number of labels increases as W grows. We employed the k-NN version of the *sklearn* for multi-label classification with K=5 as it obtains competitive results compared to other classifiers and ensures consistency with the previous approaches. Lastly, regarding the best classifier meta-target, Figure 3.4 shows the distribution of labels among the 112 datasets. It can be seen that InceptionTime is best performing classifier in 37 datasets, followed by ResNet, ST and BOSS (which are the best performing classifiers in 15, 15 and 11 datasets, respectively). In fact, only 16 of the 24 algorithms win at least once, so it is a 16-class classification problem, for which we employ the k-NN classifier for multi-class of the *sklearn* library with K=1, set by preliminary experiments.

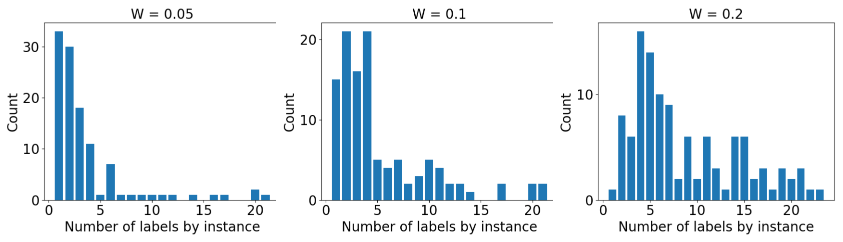


Fig. 3.3: Distribution of labels for the best set meta-target.

3.4.1.3 Evaluation

Each meta-target type requires a suitable evaluation measure for the corresponding output type. In the following, the metrics employed for each meta-target type are presented (summarized in Table 3.6). For the classifier accuracies meta-target, we employ the Mean Absolute Error (MAE), where A_c and $\widetilde{A_c}$ are the true and predicted accuracies of the candidate classifier c, and p is the number of classifiers. In order to evaluate a complete ranking meta-target, we employ a similarity measure based on the normalized Kendall's τ distance for rankings -which measures the pairwise disagreements between two rankings-. Specifically, given a Kendall's τ distance between two rankings

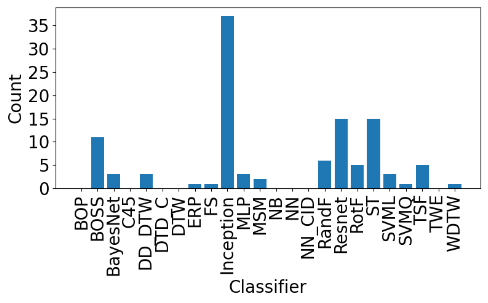


Fig. 3.4: Distribution of labels for the best classifier meta-target.

r_1 and r_2 of length n, $\tau(r_1, r_2)$, we propose the similarity measure $\tilde{\tau}_c$, which takes values from 0 (for reverse rankings) to 1 (for identical rankings). There are several ways for generalizing the KendallâĂŹs τ distance to top-M rankings [193]; a basic generalization consists of, given two complete rankings r_1 and r_2, considering all the elements at positions higher than M are tied in both rankings, and computing the Kendall's τ distance between those partial rankings. In this way, given two top-M rankings r_1' and r_2', if $\tau(r_1', r_2')$ is the Kendall's τ distance between these partial rankings, we propose the similarity measure between top-M rankings $\tilde{\tau}_p$. This similarity measure takes values analogously to $\tilde{\tau}_c$. For the best set meta-target, the evaluation measure used is the mean label based accuracy, L_a, where Y_{true} and Y_{pred} are the set of true and predicted labels, correspondingly. This metric takes values from 0 (when the intersection is zero) to 1 (when the predicted set is equal to the true set). Finally, in the case of best classifier meta-target, a weighted version of the F1 score (F1$_w$) is employed as the evaluation measure: the F1 score is calculated for each label and then a weighted average that takes into account the label frequency is computed. The F1$_w$ score takes values from 0 (worst prediction) to 1 (perfect prediction).

3.4.2 Analysis of the landmarkers

The estimations of the landmarkers have been obtained by executing the classifiers in a stratified random train/test partition with the same proportions as in [171]. Then, in order to assess their quality, their accuracies and computational times are analysed, with the objective of comparing them to those obtained by the original classifiers. Note that the accuracies of the original classifiers are publicly available in [171], but the computational time of the full process (learning, parameter optimization and test phase) is not. In order to give some insights into the computational cost of executing all the classifiers, we run the 24 original classifiers in the *StarlightCurves* and *Crop* datasets. The former is one of largest datasets in the UCR repository (9236 series of length 1024, with 3 classes), while the latter is a large dataset with many classes

Table 3.6: The meta-learner and evaluation metric employed for each meta-target type.

Meta-target	**Metric**
Classifier accuracies	$MAE = \frac{1}{p}\sum_{c=1}^{p} \lvert A_c - \widetilde{A_c} \rvert$
Complete ranking	$\tilde{\tau}_c(r_1, r_2) = 1 - \frac{\tau(r_1,r_2)}{n(n-1)/2}$
Top-M rankings	$\tilde{\tau}_p(r'_1, r'_2) = 1 - \frac{\tau(r'_1,r'_2)}{M^2}$
Best set	$L_a = \frac{\lvert Y_{\text{true}} \cap Y_{\text{pred}} \rvert}{\lvert Y_{\text{true}} \cup Y_{\text{pred}} \rvert}$
Best	$\text{F1}_w = \frac{1}{L}\sum_{l=1}^{L} w_l F1_l$

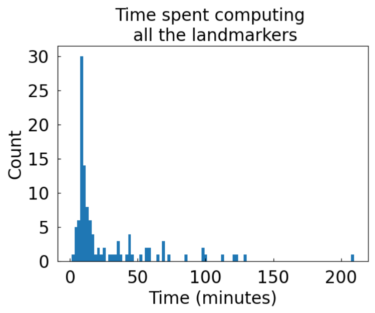
(24000 series of length 46, with 24 classes). The total time spent for executing the 24 classifiers is more than 12 days for the *StarlightCurves* dataset and almost 4 days for the *Crop* dataset in a high performance computing cluster.

Regarding the computational cost of the landmarkers, Figure (3.5a) shows the time spent (in minutes) in applying all the landmarkers to the 112 datasets. It can be seen that, in 98 of the 112 datasets, the computation of all the landmarkers takes less than an hour. Those datasets in which the computation takes more than an hour are large datasets that contain many classes, for which, in order to ensure a minimum number of representatives of each class, the subsample ratio is not as small as in other datasets with similar dimensions. In this way, the time reduction is especially significant for those dataset with few classes. This is the case of *StarlightCurves*, for example, for which the computation of all the landmarkers takes 44 minutes, compared to the 12 days spent by the original classifiers. In the case of datasets with many classes, the time reduction is not so great, even if it is still significant. In the *Crop* dataset, for example, the computation of all the landmarkers takes almost an hour (58 minutes), compared to the 4 days spent by the original classifiers. Concerning the individual computation times of the landmarkers[1], in 80 of the 112 datasets, the slowest classifier is the ST classifier (which has a time limitation), while BOSS, ResNet and RotF are slowest in 28, 3, and 1 dataset, respectively.

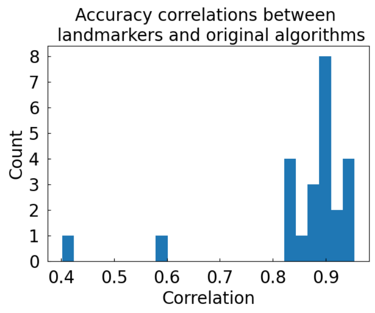
In order to explore the relation between the accuracies obtained by a landmarker and those obtained by the corresponding original classifiers, we employed the *Pearson correlation coefficient*. The correlations between the

[1] The cost of computing the 24 landmarkers in each dataset, including the slowest landmarker and its corresponding time is included in Appendix A.2.

landmarkers and the original classifiers[1] are shown in Figure 3.5b. It can be seen that most of the landmarkers have a high correlation with the corresponding original classifiers; the mean correlation is 0.86. The landmarker with the lowest correlation is the InceptionTime, which is one of the slowest classifiers, so finding a suitable trade-off between computation time reduction and relation with the original classifier is a hard task. Indeed, small changes in the reduction grade result in a considerable increase in the time needed to compute the landmarker. Taking into account that a correlation of 0.40, even if not high, still means that there is a positive correlation between the landmarker and the original classifier, we chose to prioritize the time over the correlation.



(a)



(b)

Fig. 3.5: (a) Histogram of the time spent computing the landmarkers in the 112 dataset. (b) Histogram of the correlations of the 24 landmarkers and the corresponding original algorithms.

Another way of testing the correlation of the landmarkers and the original classifiers is dataset-wise. Given a dataset, the results obtained by the landmarkers and the original classifiers in this dataset can be compared visually to see whether or not they are related. Due to the lack of space, only two examples[2] are shown in Figure 3.6. It can be seen that there is a clear relation between the accuracies obtained by the original algorithms and the corresponding landmarkers in the both datasets shown in the figure, since they share a similar pattern. The landmarkers of both datasets are good potential predictors of classifier recommendations.

[1] The correlation of each landmarker and the corresponding classifier is included in Appendix A.3.

[2] The figures of the 112 datasets are included in Appendix A.4.

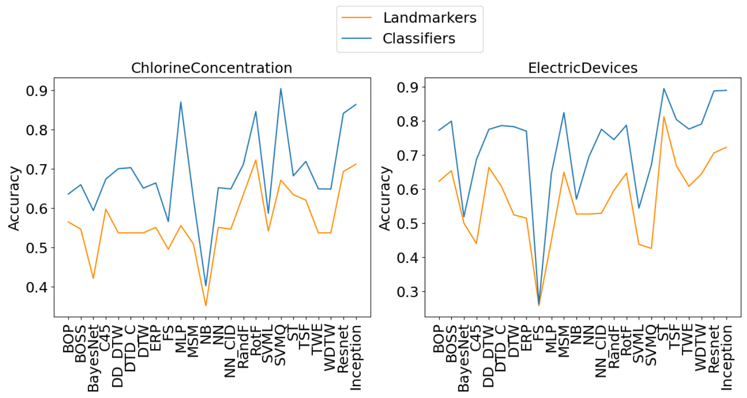


Fig. 3.6: Accuracies obtained by the original 24 classifiers and the landmarkers in *ChlorineConcentration* (left) and *ElectricDevices* (right) datasets.

3.4.3 TSCR

In this section, the proposed TSCR systems are experimentally evaluated. Apart from the SMLs presented in Section 3.2.2, we present three additional methods for each meta-target type: on the one hand, since classifier recommendation has never been addressed before in TSC, in order to determine if the results obtained by our methods are accurate or not, a baseline (BS) for each meta-target type is presented. Moreover, in order to highlight the contribution of the proposed landmarkers, the recommendation performance of the landmarkers is compared with the recommendation performance of standard meta-features for non-sequential data (MF). On the other hand, since not all landmarkers necessarily have a great prediction power, a method that includes a forward landmarker selection (SML_FLS) is considered. In the following, a brief introduction to those methods is presented.

BS: since there are no state-of-the-art proposals, we consider a set of agnostic methods that, for each meta-target type, outputs a constant prediction without any learning process. This prediction is made based on the mean values of the accuracies of the classifiers in the training set: for the classifier accuracies meta-target, the BS is a method that, for any instance in the testing set, predicts the mean accuracy of the candidate classifiers in the training set. In the same manner, for the BS of the complete ranking meta-target, the output is the ranking of the mean accuracies of the candidate classifiers in the training set, while the partial rankings of these rankings are used for the top-M ranking meta-target. The prediction of the best set is computed by applying the criterion of the margin in equation 3.1 to the mean accuracies of the candidate classifiers in the training set. Lastly, for the BS of the best classifier meta-target, the classifier with the best mean accuracy in the training set is used.

MF: In this method, instead of the landmarkers, standard meta-features for non-sequential data are used as meta-attributes for the recommendation system. Specifically, 73 standard meta-features (general, statistical and info-theoretical) are extracted from each dataset employing the *pymfe* package [36] for Python. These meta-features are used as input for the recommendation system, employing the corresponding meta-learner in each case.

SML_FLS: the followed forward landmarker selection (FLS) procedure is described in Algorithm 1. The FLS starts with a random landmarker and, at each iteration, adds the landmarker that improves the performance of the meta-learner the most to the set of selected landmarkers, until there is no improvement. The operator δ depends on the meta-target type and is the result of evaluating the predictions of the SML with the corresponding evaluation metric. For complete ranking, top-M ranking, best set and best classifier meta-targets the aim is to maximize δ (as in the pseudo-code), while for classifier accuracies meta-target is evaluated in terms of error, so the aim is to minimize δ.

Algorithm 1 Forward landmarker selection (FLS)

```
Input:
    LM = set of all the landmarkers
    $\delta$ = operator that computes the performance of a set of landmarkers
Output:
    S = set of selected landmarkers
Algorithm:
    Initialize S = one.random(LM)
    candidate = $\underset{L_i \in LM \setminus S}{\operatorname{argmax}} \ \delta\ (S \cup L_i)$
    while $\delta$ (S ∪ candidate) > $\delta$ (S) do
        S = S ∪ candidate
        if |S| = |LM| then
            break
        else
            candidate = $\underset{L_i \in LM \setminus S}{\operatorname{argmax}} \ \delta\ (S \cup L_i)$
```

From a general point of view, the evaluation procedure of the methods is the following: for the BS and SML, a 10 times repeated 5-fold nested cross validation [194] is carried out. For the forward selection, instead, a two-level nested cross validation is performed: in the first level, a 10 times repeated 5-fold nested cross validation is performed for evaluating the set of landmarkers, which have been selected in an internal 3-fold nested cross validation of training sets of the first level. The method starts with all the landmarkers once and selects the set of landmarkers that obtains the best performance in the 3-folds.

The result obtained by the BS, MF, SML and SML_FLS approaches for the considered meta-target types are shown in Table 3.7. Recall that each meta-target type is validated employing a different evaluation measure and, hence, the results of different rows can not be compared with each other.

Table 3.7: Results of the baselines (BS), specific meta-learners with meta-features as input (MF), specific meta-learners with landmarkers (SML) as input, and specific meta-learners with forward landmarker selection (SML_FLS) for the considered meta-target types.

Meta-target	**Metric**	**BS**	**MF**	**SML**	**SML_FLS**
Classifier accuracies	MAE	0.15	0.14	**0.07**	**0.07**
Complete ranking	$\tilde{\tau}_c$	0.73	0.73	**0.74**	**0.74**
Top-M ranking M=3		0.37	0.31	0.47	**0.48**
M=5	$\tilde{\tau}_p$	0.50	0.50	0.59	**0.61**
M=10		0.63	0.64	0.68	**0.69**
Best set W=0.05		**0.35**	0.28	0.28	0.30
W=0.1	L_a	0.25	0.34	0.36	**0.39**
W=0.2		0.41	0.43	0.46	**0.48**
Best	$F1_w$	0.17	**0.27**	0.18	0.18

In the case of the classifier accuracies meta-target, it can be seen that, both the BS and the MF obtain worse results than the SML, while the forward selection does not improve the performance obtained by the SML. Additionally, the best performance obtained, 0.07, is the MAE of the prediction of 24 classifier accuracies, which, taking into account the range of the accuracies, is fairly accurate.

In the complete ranking prediction, the SML is the best performing approach -closely followed by the BS and MF-, while the landmarker selection does not improve the results. The good performance of the BS deserves attention; the BS always predicts the mean ranking in the training set, so it can be deduced that most of the rankings in the testing set are close to that mean ranking. In order to explore this issue, we conduct the same experiment but, instead of computing the overall performance of all the predicted rankings, the performance is evaluated separately for instances that have different similarities ($\tilde{\tau}_c$) with respect to the mean ranking in the training set. For each fold in the cross validation, the percentiles P_{25}, P_{50} and P_{75} are extracted from the similarities between the rankings in the training set and the mean ranking in the training set, and the instances in the testing set are divided into 4 sets: S_1 is the set of test instances with similarity within the range $[P_0, P_{25})$, and S_2, S_3 and S_4 within $[P_{25}\ P_{50})$, $[P_{50}\ P_{75})$ and $[P_{75}\ P_{100}]$, respectively. Figure 3.7 shows the mean performance and standard deviation of the BS and SML for the different sets. It can be seen that, as expected, the SML outperforms the BS in those instances with a true ranking that is very dissimilar to the mean ranking in the training set. Even if the average performances are similar (0.73 and 0.74), the SML distributes the error more uniformly among the instances.

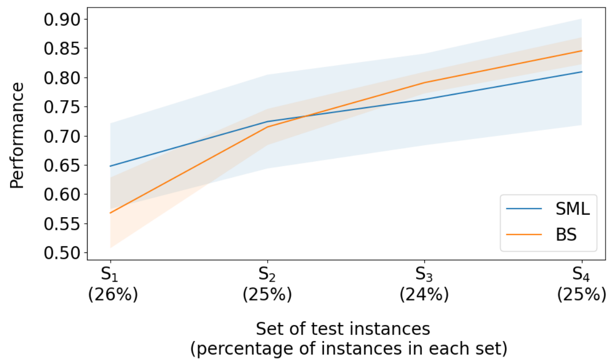


Fig. 3.7: Performance depending on the similarity of an instance respect to the mean ranking in the training set.

In the case of the top-M rankings, two main conclusions can be drawn: the SML_FLS is the best approach for all values of M, and the performances of all the meta-learners increases with M. These results are quite reasonable since it is expected that predicting a top-10 ranking is an easier task than predicting a top-3 ranking.

A similar pattern can be seen in the best set prediction: the performance of the MF, SML and SML_FLS increases with W and, hence, with the number of labels per instance, while the performance of the BS does not seem to follow this trend. The forward selection improves the results of the SML for any W. The main difference with the results obtained for the top-M rankings is that, in the best set prediction, for W=0.05 the BS outperforms the rest of the approaches. A possible explanation for this fact is that, as Figure 3.3 shows, for W=0.05, most of the instances have very few labels. As shown in Figure 3.4, InceptionTime, BOSS and ST are the best performing classifiers in most of the datasets, so many of the instances contain these labels. Since BS predicts the most frequent classifiers, it obtains good results due to the label imbalance.

Lastly, regarding the best classifier meta-target, we handle the label imbalance by the metric specified in Section 3.4.1.3. It can be seen that the MF is the best performing approach, followed by SML and BS, while the SML_FLS does not improve the results.

Summarizing, the experimentation carried out shows that the SMLs are, in 7 out of 9 of the considered scenarios, the best performing TSCR approaches. In some cases, the corresponding BS obtains results that are similar -in a single case better- to the SML, which we think is because there are some classifiers that, in general, obtain better results than others, a fact that benefits the BS. The approach that employs the standard meta-features as input outperforms the landmarker based approaches in a single case (best meta-target), and the SML_FLS improves the results of the SML in several cases, which indicates that all the landmarkers are informative in some cases, while in orders they are not.

3.4.4 Hierarchical inference of meta-targets

In this section, we experimentally compare the hierarchical inference of meta-targets with the corresponding TSCR system in order to explore whether or not a specific TSCR is needed. For each meta-target type, we chose the best performing approach between the SML and the SML_FS (Table 3.7) for the hierarchical inference. The experimental set up is the same as that specified in Section 3.4.3.

Table 3.8 displays the results obtained for the hierarchical inference of the considered meta-target types. Following the scheme in Figure 3.2, the columns indicate the MT$_i$ meta-target, while the rows refer to the inferred MT$_j$ meta-targets; the first column, for example, displays the performances obtained for the different meta-target types inferred from the predicted classifier accuracies. The results obtained by the SML of each meta-target type are shown in the diagonal, while the $\times$ symbol reflects that the inference is not feasible for this couple of meta-target types. Analogously to Table 3.7, each meta-target type and, hence, each row, is evaluated employing a different metric, so the results of different rows can not be compared with each other.

Table 3.8: Results obtained for the hierarchical inference approach of the considered meta-target types. Columns indicate the departing meta-target, while rows indicate the inferred meta-target.

	Classifier accuracies	Complete ranking	Top-M ranking M = 3, 5, 10	Best set W = 0.05, 0.1, 0.2	Best
Classifier accuracies	**0.07**				
Complete ranking	**0.76**	0.74			
Top-M ranking M=3	**0.55**	0.46	0.48		
M=5	**0.68**	0.59	0.61		
M=10	0.65	0.68	**0.69**		
Best set W=0.05	**0.30**			**0.30**	
W=0.1	0.35	$\times$	$\times$	**0.39**	
W=0.2	0.45			**0.48**	
Best	**0.26**	0.20	0.22 0.21 0.19	$\times$	0.18

It can be seen, that, in general, the inference obtains very competitive results. In fact, in most of the cases, the results of our experimentation show that there is no need to employ a specific TSCR system for each meta-target type, since almost equally or even better results can be inferred from more fine-grained meta-targets. That means that, in this problem, a single model is sufficient to obtain competitive results for many meta-target types. For the complete ranking meta-target, for instance, our experimentation suggests that a ranking learning method is not necessary since even a better performance is obtained by inferring the rankings from the predictions of the linear multi-output regression. An interesting pattern that is observed for the top-M rankings meta-target is that, the smaller the parameter, the better the results of the hierarchical inference (from the classifier accuracies) are compared to the SML. A possible explanation for this fact is that, smaller parameters give rise to more fine-grained meta-targets (shorter partial rankings), and these meta-targets seem to benefit from more fine-grained TSCR systems.

To sum up, in contrast to what Kalousis *et al.* [191] and Bensusan *et al.* [192] reported, the results of our experimentation suggest that, given a meta-target type, the hierarchical inference obtains results that are competitive with the specific TSCR. The main conclusion of this finding is that, at least in our scenario, a single linear multi-output regression is enough to infer almost all the meta-target types with results that are competitive with the corresponding SML.

3.5 Conclusion and future work

In this Chapter, time series classifier recommendation has been addressed by a meta-learning approach for the first time in the literature. The proposed method consists of three main parts: landmarker based time series supervised dataset characterization, classifier recommendation and hierarchical inference of the meta-targets.

In the first part, the temporal supervised dataset characterization is tackled by the proposal of a set of 24 TSC landmarkers, which are obtained by dataset subsampling and algorithm reductions. The experimental analysis of the landmarkers show that they are fast to compute (in the *StarlightCurves* dataset, for instance, the time is reduced from more than 12 days to 44 minutes), while the accuracies obtained by the landmarkers are highly correlated to the results of the original classifiers (mean correlation of 0.86).

Five standard meta-target types are considered: classifier accuracies, complete ranking, top-M ranking, best set and best classifier. For each meta-target type, four TSCR approaches are considered: two specific meta-learners (the SML and the SML_FLS), as well as a baseline BS and an approach with standard meta-features MF, for comparative purposes. The experimentation validates the landmarkers we proposed since, in 7 of the considered 9 recommendations, the SML outperforms the BS and MF, while the forward selection improves or equals the results in 7 of 9 scenarios. Moreover, in those cases in which the BS obtains competitive results, we prove that the proposed methods are more stable than the BS -in the sense that they distribute the error more uniformly among the datasets-.

In the last part of the Chapter, the hierarchical inference of meta-targets is addressed. This issue is an almost unexplored point of view in the meta-learning literature, and the few works that have addressed it [191, 192] did not report competitive results. In our work, by contrast, it is proved to be a promising approach. Most of the meta-target types can be inferred from a single linear multi-output regression, obtaining even better results than those obtained with the corresponding TSCR system.

Regarding future work, the proposed TSCR systems could be extended to other TSC scenarios, such as multi-variate or unequal-length TSC. In addition, it would be interesting to find out which characteristic of the time series datasets makes a classifier obtain better results than others. This is a pending aspect in the TSC community and could be addressed by the definition of meta-attributes that describe supervised time series datasets, based on the characteristics that they contain.

4

Ad-Hoc Explanation for Time Series Classification

4.1 Introduction

In the recent years, and with the rise of deep learning models, the state-of-the-art methods are ofter too complex and understanding their decision making process is rather complicated. In this context, Explainability [48] (introduced in Section 1.3) is an increasing field of research that has arisen with the aim of providing simple explanations of the predictions of models in a manner that humans can understand.

In TSC, some efforts have been done to propose methods that explain time series classifiers. Most of them have focused on intrinsic methods for explaining deep learning models [64, 65, 66, 67] or machine learning classifiers for time series [61, 62, 63]. However, as already mentioned in Chapter 1, intrinsic explanation methods are limited to the model they are designed for. Agnostic methods, instead, provide an explanation that is independent on the model, and can be used to explain the predictions of any classifier. In TSC, until now, two agnostic methods have been proposed [69, 70], both of them following on a perturbation based approach. These perturbations, however, are not specific and realistic for time series.

Our proposal aims at going a step further and provides a local model-agnostic explanation based on a more realistic set of perturbations for time series. To this end, some characteristic of the original series is perturbed and the vicinity of a time series is generated by applying transformations that are more natural for time series data. In this work, we consider 4 transformations based on [4, 195, 196]: warp, scale, noise and slice. In this way, the explanations provided by our method have an interpretation: an interval is important because, if a transformation is applied to this interval, the prediction will change. In this way, the proposed explanations offer a semantic meaning; if a time series dataset is collected from sensors that capture the movements of a person doing a certain exercise, a possible interpretation provided by our method is that an interval is important for the prediction because, if the speed of the movement in this interval changes, the time series will be categorized into another class.

Given a transformation type, the proposed method provides two explanations: the robustness for the prediction of the classifier with respect to that transformation, and the relevance of each region of the series in the prediction. This will measure how sensitive the classifier is in the time series of interest to that transformation. In the case where a prediction is sensitive to a transformation, the relevance of each region of the series in the prediction is computed.

The rest of the Chapter is organized as follows: in Section 4.2, the proposed transformations for time series are presented. The explanation method is thoroughly described in Section 4.3, while the experimentation is presented in Section 4.4. Finally, the main conclusions are drawn in Section 4.5.

4.2 Time series Transformations

In order to define a proper set of realistic and natural transformations for time series, we follow the ideas presented in [4, 195, 196]. In [4], the authors state that a similarity measure for time series should be robust to a set of transformations and they propose the scale, warp, noise, and outliers transformations. In [195], the authors claim that time series with unequal-lengths appear naturally in real-world problems due to reasons such as variations in the frequency of measurements (warp) or variations in the starting or/and ending index of the time series (slice), and propose transformations to simulate this kind of time series. Lastly, time series data augmentation is addressed in [196], where two methods are proposed: time series slice extraction and window warping. Inspired by the previous works, we decided to consider 4 transformations: warp, scale, noise, and slice. In our case, transformations that involve intervals of the time series are considered, so the outlier transformation -which applies to certain indexes of the series but not to a whole interval- has not been taken into account. In the following sections, the considered transformations are presented in detail.

4.2.1 Warp

This transformation is a deformation of the series in the time axis (x-axis), which produces a compression or expansion (depending on the warp level) of the values of a series in a given interval. Time series with an interval warped at different warp levels appear frequently in problems such as the GunPoint dataset from the UCR repository. Figure 4.1 shows two time series from this dataset, one from each class. The interval that represents the gun rising can be seen as a warped version of the interval that represents the hand rising (or vice-versa). In particular, in the time series from the Gun class, this movement is made in a quicker manner than in the time series from the Point class. In this case, this interval is considered discriminative with respect to the warp transformation, because if we compress/expand it, the classifier could change its class prediction.

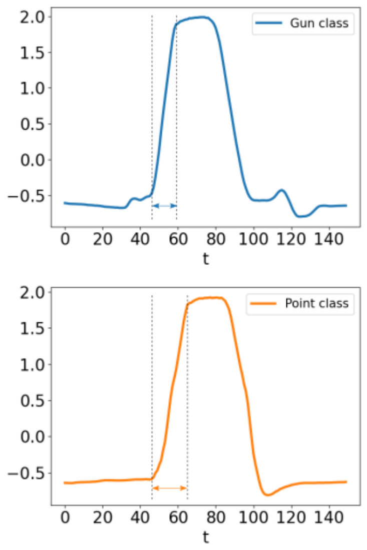


Fig. 4.1: Two time series from the GunPoint dataset. The time series from the Gun class can be seen as a warped version of the time series from the Point class (or vice-versa), where the warp has been applied in the marked interval.

In order to synthetically create warped versions of a reference time series $T = (t_1, \ldots, t_i, \ldots, t_l)$, we propose the warp transformation; given an interval $[s, e]$ in which the transformation is applied (such that $0 < s < e < l$) and a warp level k_w, the warped version of T is defined by:

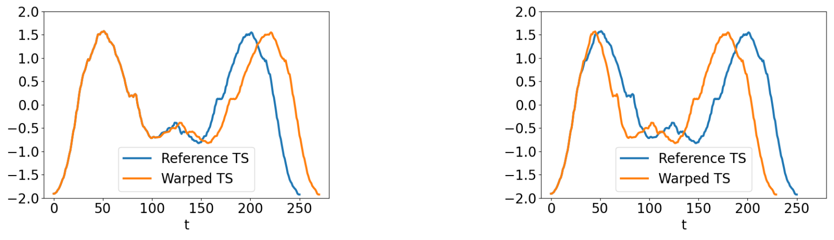
$$T' = (t'_1, \ldots, t'_i, \ldots, t'_{s+(e-s)*k_w+(l-e)}) \tag{4.1}$$

where

$$t'_i = \begin{cases} t_i, & \text{if } i \leq s \\ (w_1 t_p + w_2 t_q)/k_w, & \text{if } s < i \leq s + k_w(e-s) \\ t_{e+i-(s+k_w(e-s))}, & i > s + k_w(e-s) \end{cases} \tag{4.2}$$

where $t_p = \max_{j=1,\dots,l} [jk_w] < i$ and $t_q = \min_{j=1,\dots,l} [jk_w] \geq i$. The notation [ ] refers to the nearest integer and the weights are defined by $w_1 = k_w - (i - p)$ and $w_2 = k_w - (q - i)$. The first and last parts in (4.2) correspond to the interval that remains from the reference time series. The second part corresponds to the warped interval, which is a weighted average of the two nearest warped points of the reference time series.

Note that this transformation compresses ($k_w < 1$) or expands ($k_w > 1$) an interval of the reference series, thus the transformed series will be shorter or larger than the reference series. Two examples of synthetic warped versions of a reference time series from the ArrowHead dataset from the UCR repository are shown in Figure 4.2.



(a) $k_w = 1.2$ in $[100, 200]$. (b) $k_w = 0.7$ in $[30, 100]$.

Fig. 4.2: A reference time series from the ArrowHead dataset and two examples of synthetic warped version.

4.2.2 Scale

This transformation is a deformation of the series in the y-axis. It produces an upward or downward displacement (depending on the scale level) of the values of a series in a given interval. Time series with different scale levels can be found, for example, in the Adiac dataset from the UCR repository. In this dataset, images of 37 classes of diatoms (unicellular algae) are converted into time series, and time series from different classes differ in the scale in several intervals (see Figure 4.3). The interpretation of this fact is that these two diatoms have very similar shapes, but differ in the amplitude of the shape in some intervals. As such, the scale in those intervals is considered as relevant for the prediction.

We propose the scale transformation of a reference time series $T = (t_1, \dots, t_i, \dots, t_l)$ as: given an interval in which the transformation is applied $[s, e]$ (such that $0 < s < e < l$) and a scale level k_s, the scaled version of the series is:

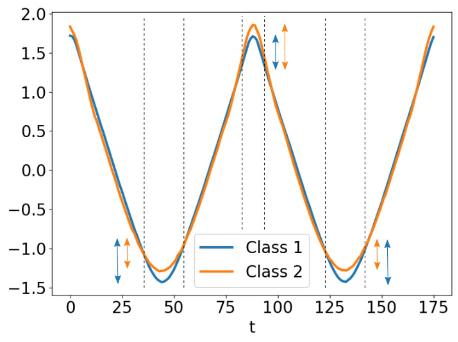


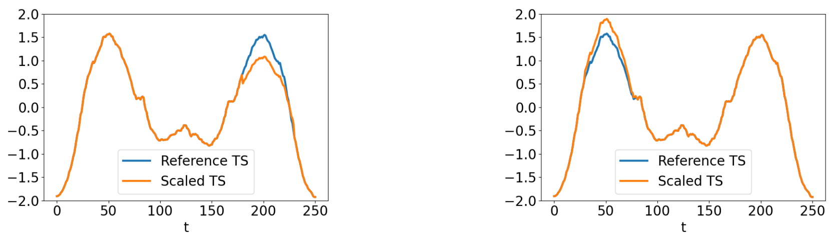
Fig. 4.3: Two time series from different classes of the Adiac dataset that differ in the scale level in the marked intervals.

$$T' = (t'_1, \ldots, t'_i, \ldots, t'_l) \tag{4.3}$$

where

$$t'_i = \begin{cases} t_i * k_s, & \text{if } s \leq i \leq e \\ t_i, & \text{otherwise} \end{cases} \tag{4.4}$$

A reference time series from the ArrowHead dataset and two examples of synthetic scaled versions are shown in Figure 4.4.



(a) $k_s = 0.7$ in $[180, 230]$. (b) $k_s = 1.2$ in $[30, 80]$.

Fig. 4.4: A reference time series from the ArrowHead dataset and two examples of synthetic scaled versions.

4.2.3 Noise

The noise transformation consists of adding noise to the series in a given interval. An example of time series with different noise levels that determine the class can be seen in the ECG200 dataset from the UCR repository, for instance. In this dataset, the electrical activity of a heartbeat is recorded in two scenarios (normal heartbeat and a myocardial infarction). Figure 4.5 shows two time series from different classes that differ in the noise level in the marked interval. In these two examples, it can be seen that the heartbeat in both scenarios is very similar, but the normal heartbeat is smoother than the heartbeat of myocardial infarction. In this case, a noise transformation in this interval may make the classifier predict the series into another class and, hence, the noise level in the given interval is considered as discriminative.

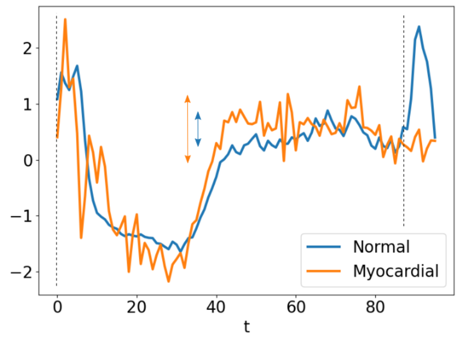


Fig. 4.5: Two time series from different classes of the ECG200 dataset that differ in the noise level in the marked interval.

Given a reference time series $T = (t_1, \dots, t_i, \dots, t_l)$, an interval in which the transformation is applied $[s, e]$ (such that $0 < s < e < l$) and a noise level k_n, the noise-transformed version of the series is defined by:

$$T' = (t'_1, \dots, t'_i, \dots, t'_l) \tag{4.5}$$

where

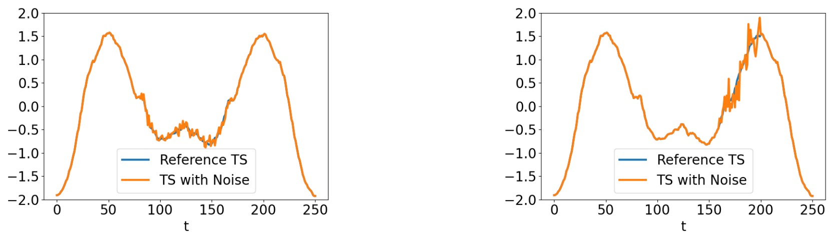
$$t'_i = \begin{cases} t_i + \mathcal{N}(0, \frac{A * k_n}{100}), & \text{if } s \leq i \leq e \\ t_i, & \text{otherwise} \end{cases} \tag{4.6}$$

with $A = |max(T) - min(T)|$ the amplitude of the series. As Equation 4.6 shows, the added noise is a Gaussian noise $\mathcal{N}(\mu, \sigma)$, with $\mu = 0$ and a σ that depends on the amplitude of the time series and the noise level k_n. In this way, for $k_n = 5$, for example, the standard deviation is set to 5% of the amplitude of the series.

A reference time series from the ArrowHead dataset and two examples of synthetic versions with noise are shown in Figure 4.6.

4.2.4 Slice

The slice transformation selects a subsequence of the time series and aligns it at the beginning of the reference time series. This transformation can be seen



(a) $k_n = 2$ in $[80, 170]$. (b) $k_n = 7$ in $[160, 200]$.

Fig. 4.6: A reference time series from the ArrowHead datasets and two examples of synthetic versions with noise.

as selecting and shifting a subsequence of the time series. As the authors of [195] pointed out, for many time series that represent audio, video, or sensor recordings, the exact moment at which the recording starts and finishes may vary. For instance, for a time series that represents the audio recording of a person saying a word, the location of subsequence that represents the word may be different depending on the exact starting and ending point of the recording.

Sliced time series can be found, for example, in the AllGestureWiimoteY dataset from the UCR repository. This dataset consists of the y-axis recording of 10 individuals performing 10 different gestures measured by the Nintendo Wiimote controller. Two series from different classes are shown in Figure 4.7; it can be seen that both time series have very similar shapes, but the time series from class 5 is a sliced version of the time series from class 7. That is, if the recording of the time series from class 7 had started a bit later and had finished before, the classifier would probably have classified it in class 5. In this way, the marked interval and, in particular, its location are considered as relevant for the prediction.

Given a reference time series $T = (t_1, \ldots, t_i, \ldots, t_l)$ and a random interval $[s, e]$, the slice transformation consists of removing the intervals $[1, s)$ and $(e, l]$ from the reference series. In this way, in contrast to the previous transformations, the slice transformations does not modify the series in the $[s, e]$ interval, but it is applied in $[1, s) \cup (e, l]$. The sliced version of T is defined by:

$$T' = (t'_1, \ldots, t'_i, \ldots, t'_{e-s}) \tag{4.7}$$

where

$$t'_i = t_{s+i}, \qquad i = 1, \ldots, e - s \tag{4.8}$$

This transformation involves the time axis of the series, and hence, the transformed time series are shorter than the reference series. A reference time series from the ArrowHead dataset and two examples of sliced versions are shown in Figure 4.8.

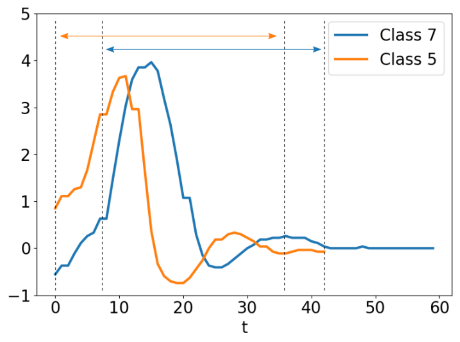
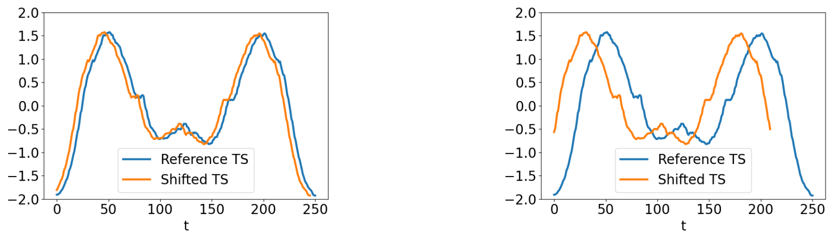


Fig. 4.7: Two time series from the AllGestureWiimoteY dataset. The time series from class 5 can be seen as a sliced version of the time series from class 7.



(a) Sliced in $[5, 250]$.

(b) Sliced in $[20, 230]$.

Fig. 4.8: A reference time series from the ArrowHead dataset and two examples of synthetic sliced versions.

4.3 Time Series Classification Explanation Method

Given a time series in a labelled dataset, a classifier, and a transformation, the method provides a two level explanation: the high-level explanation describes the robustness of the prediction with respect to the transformation, while the low-level explanation displays the relevance of each region of the series in the prediction.

4.3.1 High-level explanation

The high-level explanation can be summarized in 3 steps (see Figure 4.9):

Neighbour generation. We create neighbour time series by sampling a random interval of the reference series and applying the transformation to this interval. For this, a random interval generation is needed, such that all the time indexes of a series have the same probability to appear in an interval. We

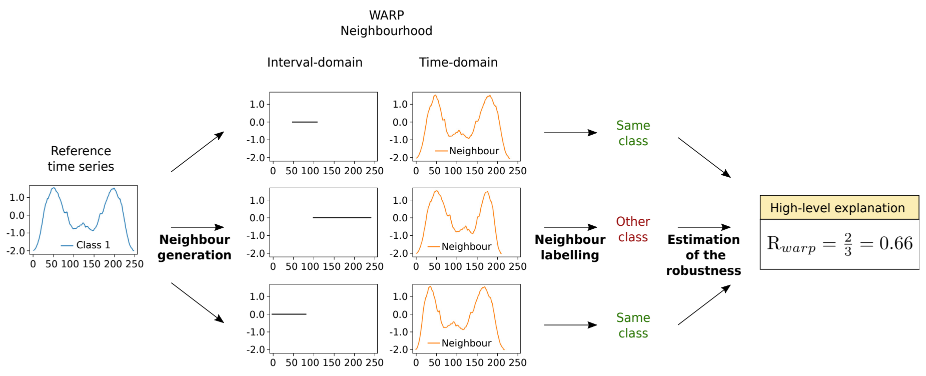


Fig. 4.9: Scheme of the high-level explanation with an example of time series from the ArrowHead dataset and warp transformation with $k_w = 0.7$.

employ an approach called *randomized unwrap the circle method* [197], which considers the intervals as wrapped around the perimeter of a circle (described in Algorithm 2).

Algorithm 2 Randomized unwrap the circle method

```
Input: Parameters of Beta prime distribution $\alpha$, $\beta$
Output: Random interval in $[0, 1]$ Sample $x$ from BetaPrime($\alpha$, $\beta$)
1: Sample $u$ uniformly on $[-x, 1]$
2: Return $[0, 1] \cap [u, u + x]$
```

Neighbour labelling. In this step, the generated neighbours are labelled by the classifier. Some of the transformations involve deformations in the time axis of the time series, giving rise to neighbours with larger or shorter lengths than the reference time series. Since many benchmark classifiers for TSC are not adapted to varying length series [195], this leaves us two options: to employ a pre-processing step to equal the lengths of all the series, or to limit our method to classifiers that can handle series of varying lengths. The first option involves adding information to the shorter series and removing information from the larger series, while the second works with all the information but restricts our method to a shorter number of classifiers. In this work, in order to keep the information as raw as possible, the second option has been adopted.

Estimation of the robustness. We consider that the prediction is robust with respect to the transformation if all the neighbours are classified into the same class of the reference time series. In the following, we call $I = I_= \cup I_{\neq}$ the set of generated random intervals, where $I_=$ refers to those intervals that result

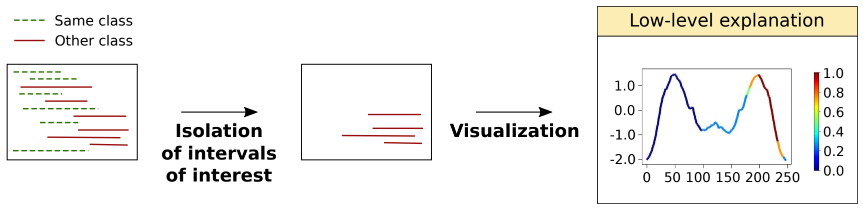


Fig. 4.10: Scheme of the low-level explanation with an example of a time series from the ArrowHead dataset.

in neighbours from the same class of the reference series, and $I_{\neq}$ to those that result in neighbours from other classes. As such, in this step, the robustness of a prediction with respect to a transformation, R_{transf}, is quantified by measuring the percentage of neighbours that are classified into the same class of the reference time series:

$$\mathrm{R}_{transf} = |I_{=}|/|I|$$

where the operator $|\ |$ refers to the cardinal. R_{transf} varies in the range $[0, 1]$, where a value of 0 means that the prediction is completely sensitive to the transformation, since all the considered neighbours are labelled into another class. A value of 1, in contrast, means that the prediction is completely robust with respect to the transformation, since all the considered neighbours are labelled into the same class of the reference series.

4.3.2 Low-level explanation

The low-level explanation consists of computing the relevance of each region of the time series in the prediction. In this step, two possible scenarios are considered: the prediction is completely robust with respect to a transformation ($\mathrm{R}_{transf} = 1$), or the prediction is somewhat sensitive to a transformation ($\mathrm{R}_{transf} < 1$). In the first case, a transformation never affects the prediction, and hence, the low-level explanation is that there are no intervals that have a special impact on the prediction, with respect to the transformation. In the second case, the low-level explanation is computed following the next steps (see Figure 4.10).

Isolation of intervals of interest. An interval is considered relevant for the prediction if a transformation in this interval changes the prediction of the classifier (i.e., those intervals in $I_{\neq}$). In some regions of the time series, however, there may be intervals both from $I_{\neq}$ and from $I_{=}$. In order to guarantee that a region of the series is relevant for the prediction, we define the intervals of interest as those intervals from $I_{\neq}$ that do not have an interval from $I_{=}$ around.

The procedure to isolate the intervals of interest can be summarized in the following steps:

(i) Identify those intervals from $I_{\neq}$ that are closer than a parameter z from any interval from $I_{=}$ employing the $\gamma(z)$ function:

$$\gamma(z) = \{i_{\neq} \in I_{\neq} | \ \exists i_{=} \in I_{=}, \ \ d_H(i_{=}, i_{\neq}) < z\}$$

where d_H is the Hausdorff distance between intervals[1].

(ii) The set of isolated intervals of interest is defined by:

$$I_{\neq} \setminus \gamma(z)$$

Note that the set of isolated intervals depends on the parameter z, so an adequate threshold z^* needs to be set for each instance. Our preliminary studies showed that the number of intervals returned by γ increases with z, but the slope is generally decreasing (see Figure 4.11 for an example). However, there are some jumps in the function, where the slope increases and then decreases again. The values of z in which there is a jump are interesting as candidates for $z*$, since they indicate that there is a significantly larger group of intervals at distance z than at distance $z-1$. In order to define an automatic z^* for each neighbourhood, we propose the following rule: if there is a jump in the γ function, z^* is fixed to the first distance z in which a jump occurs, while if there is not a jump, z^* is fixed to the distance in which half of the intervals are removed. This jump is defined in terms of the second symmetric numerical derivative (the complete rule is summarized in Algorithm 3).

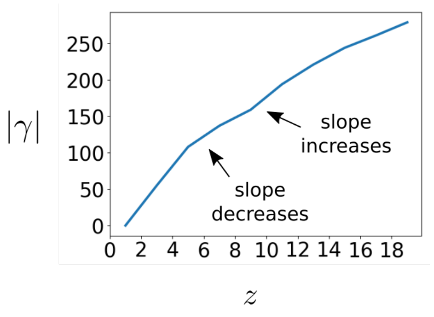


Fig. 4.11: Example of the γ function on a time series from the CBF dataset from the UCR repository and warp transformation with $k_w = 0.7$.

[1] Given two intervals $A = [a_1, a_2]$ and $B = [b_1, b_2]$ in $\mathbb{R}$, the Hausdorff distance [198] is defined by $d_H(A, B) = \max\{|a_1 - b_1|, |a_2 - b_{12}|\}$. Note that the slice transformation is applied in the union of intervals instead of in intervals (Section 4.2.4), thus the extension for unions [199] is used for this transformation.

Algorithm 3 Automatic threshold

```
Input: Curve of removed intervals $\gamma(z)$
Output: Threshold $z^*$
  if $\exists z$ s.t. $|\gamma(z+1) - 2\gamma(z) + \gamma(z-1)| > 0$ then
      $z^* = \arg\min_{z \in \mathbb{N}} \{\gamma(z+1) - 2\gamma(z) + \gamma(z-1) > 0\}$
  else
      $z^* = \arg\min_{z \in \mathbb{N}} \{\gamma(z) > \frac{|I_{\neq}|}{2}\}$
```

Visualization. This step consists of summarizing and representing the information of the interval of interest in the original time series. The idea is that, the more intervals of interest that contain a time index i, the more relevant this index is for the prediction. As such, given a time series $T = (t_1, \dots, t_i, \dots, t_l)$, we compute the number of isolated intervals that contain the index i for $i = 1 \dots l$ and this values are stored in a vector $w = (w_1, \dots, w_i, \dots, w_l)$. The explanation, thus, can be seen as a weight vector w, in which w_i represents the relevance of the time index i in the prediction. Then, the time series is coloured depending on these values. The red colour indicates that these time indexes are contained in many relevant intervals, so this region is important for the prediction, while the blue colour, indicates the opposite. Note that the colorbar is normalized to $[0, 1]$.

4.4 Experimentation

Explainability is a singular field in which the evaluation can be quite subjective and, hence, difficult to assess quantitatively. In this work, the evaluation is carried out both qualitatively and quantitatively. The experimental set-up is specified firstly, followed by a case of study. Lastly, for the quantitative evaluation, we adapt the evaluation methodology presented in [74] for quantifying the informativeness of explanation methods for TSC to the context of the proposed transformations. All the code has been developed in Python and is publicly available[1].

4.4.1 Set-up

The experimentation has been carried out considering several parameters of the transformations. The warp and scale levels considered in this experimentation, k_w and k_s, are $\{0.7, 0.8, 0.9, 1.1, 1.2, 1.3\}$, while the considered noise

[1] https://gitlab.bcamath.org/aabanda/tscexplanation.

levels, k_n are $\{1, 3, 5, 7, 9\}$. Each transformation level is independently studied. The slice transformation does not depend on any parameter but, in order to ensure a minimum length in the generated neighbours, we set the minimum interval length to $0.3 * l$, where l is the length of the series that is the object of study.

Regarding the neighbour generation process, the size of the neighbourhood is set to 500, while the parameters in 2 are set to $\alpha = 8$ and $\beta = 18$. such that the probability of a index to be covered by an interval is 0.3 [197]. In this way, each time index appears in approximately 150 of the 500 generated neighbours.

Lastly, regarding the classifiers, as mentioned before, in this experimentation we employ classifiers that handle varying length time series. As such, 3 benchmark and diverse classifiers for TSC have been chosen: from the category of elastic or distance based classifiers, the 1-NN-DTW classifier, from shapelet based classifiers, the Shapelet Transform (ST), and, from the dictionary based category, the Bag-of-SFA-Symbols (BOSS).

4.4.2 Qualitative evaluation: case of study

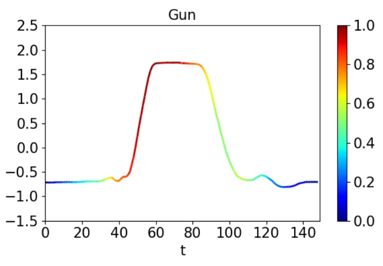
In this section, the proposed explanation method is qualitatively evaluated in some example time series extracted from datasets of the UCR repository. Given a time series and a classifier, our method provides a separate explanation for each transformation, so, due to the lack of space, a single representative example of each transformation is presented in this section. The explanations are computed for two well-known datasets from the UCR repository (GunPoint and Coffee), since these datasets have already been used before in methods concerning TSC explainability [61, 63, 65]. The GunPoint dataset has already been introduced, while the Coffee dataset is a binary dataset composed of time series that represent food spectrographs of two types of coffee: arabica and robusta. From each class in each dataset, a time series has been randomly taken for the following examples[1]. In each example, the classifier and the transformation level has been chosen such that the robustness of the combination classifier-transformation in the given time series is lower than 1 -which makes the semantic analysis of the explanation meaningful-.

Warp. Figure 4.12a shows the explanation obtained for the warp transformation ($k_w = 0.7$) and the 1-NN-DTW classifier in a time series from the Gun class of the GunPoint dataset. The R_{warp} is 0.45, which indicates that the output of the classifier in this time series is quite sensitive to the warp transformation - more than a half of the neighbours generated by warping random intervals of the series are classified into the Point class-. Our method

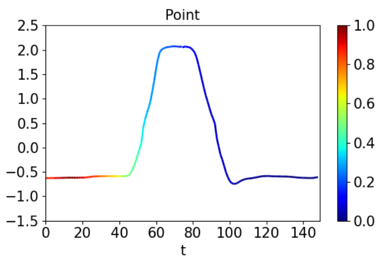
[1] For the sake of reproducibility, the time series index of each example is specified. From the GunPoint dataset, time series 52 (Fig. 4.12a) and 1 (Fig. 4.12b). From the Coffee dataset, time series 6 (Fig. 4.13a) and 21 (Fig. 4.13b).

states that the most discriminant region is the part in which the subject raises the gun from the hip; if the subject does this movement faster, the time series is likely to be classified into the other class.

Scale. The explanation for a time series from the Point class of the GunPoint dataset for the scale transformation ($k_s = 1.2$) and the 1-NN-DTW classifier is shown in Figure 4.12b. The R$_{scale}$ is 0.64, so the prediction is rather robust to the scale transformation (35% of the scale transformed neighbours are classified into the other class). The explanation indicates that the most discriminative region is the part in which the subject is pretending to hold the gun on the hip; if the vertical position of his/her hand was lower than what it is, the 1-NN-DTW would probably classify this time series into the other class.



(a) Warp ($k_w = 0.7$) with the 1-NN-DTW classifier, R$_{warp}$= 0.45.

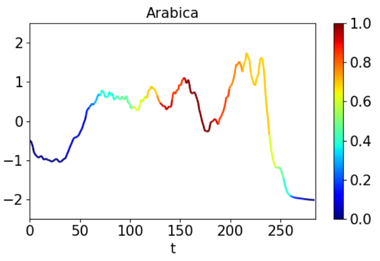


(b) Scale ($k_s = 1.2$) with the 1-NN-DTW classifier, R$_{scale}$= 0.64

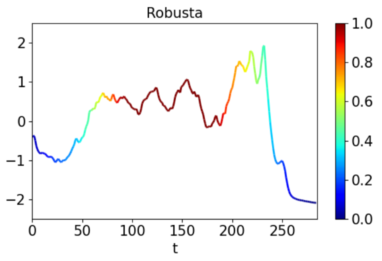
Fig. 4.12: Two explanations provided by our method in a time series from the Gun class (a) and Point class (b) from GunPoint dataset.

Noise. Figure 4.13a shows the explanation obtained for a time series from the Arabica class of the Coffee dataset with the noise transformation ($k_n = 9$) and the BOSS classifier. In this case, the output of the BOSS classifier is very robust to the noise transformation (R$_{noise}$ = 0.94); it is very difficult to mislead the classifier by adding noise to the time series. An interesting finding is that the explanation obtained by our method is very similar to that reported in [63] for a time series from the same class, even when the methods are fundamentally different. The authors point out that the highlighted regions correspond to the chlorogenic acid and caffeine contents of the coffee blends, i.e., the regions that discriminate between the Arabica and the Robusta coffee types [200].

Slice. An explanation obtained for a time series from the Robusta class of the Coffee dataset with the slice transformation and the ST classifier is shown in Figure 4.13b. The prediction of the ST classifier in this time series is quite sensitive to the slice transformation, with a R_{slice} of 0.48. Analogously, the explanation obtained by our method is very similar to that presented in [63], where the highlighted regions correspond to the discriminative parts reported in [200].



(a) Noise ($k_n = 9$) with the BOSS classifier, R_{noise}= 0.94.



(b) Slice with the ST classifier, R_{slice}= 0.48.

Fig. 4.13: Two explanations provided by our method in a time series from the Arabica class (a) and Robusta class (b) from Coffee dataset.

4.4.3 Quantitative evaluation on UCR

In this section, the quantitative evaluation of the proposed method is presented.

4.4.3.1 Experimental Design

In the field of TSC, almost all the explanation methods in the literature are evaluated in a qualitative way in a specific dataset [62, 63, 65, 66, 67, 69]. Recently, Nguyen *et al.* [74] presented a general methodology to quantitatively evaluate whether an explanation method for TSC is informative. Given a time series and an explanation, the authors propose perturbing the time series by adding noise to the important and non-important regions of the series. The idea behind their method is that perturbations on the important regions should change the class prediction more frequently than perturbations on the non-important regions.

Given the weight vector w computed in Section 4.4, the important and non-important regions are defined based on the distribution of its values: the $p\%$ most important indexes are the indexes of the $p\%$ highest values of

w, while the $p\%$ least important indexes are those corresponding to the $p\%$ lowest values of w. In their work, however, the only considered perturbation is noise, and it is added independently to each time index. The adaptation to this evaluation method that we propose, allows the considered transformations to be employed as perturbations (not only noise).

The general evaluation procedure is summarized in Figure 4.14: given a classifier and a dataset divided into train/test sets, the classifier is trained using the train set and the explanations for all the time series in the test set, $w^1, \ldots, w^{n_{ts}}$ (where n_{ts} is the number of time series in the test set), are computed. Then, perturbed versions of the test set are computed for different values of p (in this work, 3 values of p are considered: 10, 50, 90). More specifically, if X is the test set, X^{p-} is a modified version of the test set in which the least important $p\%$ of the time indexes are perturbed in all the time series of this test set. Analogously, X^{p+} is a modified version of the test set in which the most important $p\%$ of the time indexes are perturbed in all the time series of this test set. Then, the labels of all the series in the perturbed test sets are predicted by the classifier; given a perturbed test set, X^{10+} for instance, the classifier is employed to obtain the corresponding labels Y^{10+}. In order to measure the frequency in which the labels of a perturbed test set are different from the labels of the true test set, the consensus between them is defined by:

$$C^{p+} = \frac{1}{n_{ts}} \sum_{s=0}^{n_{ts}} \mathbb{1}_{y_s = y_s^{p+}}$$

where

$$\mathbb{1}_{y_s = y_s^{p+}} = \begin{cases} 1 & \text{if} \quad y_s = y_s^{p+} \\ 0 & \text{otherwise} \end{cases} \tag{4.9}$$

The consensus varies from 0 to 1, where 0 means that all the labels in Y^{p+} are different from those in Y, while 1 means that all the time series in the perturbed test set are classified in the same class as the original time series. The case for perturbations p_{-} is analogous. The consensus is computed sequentially for the different values of $p = (10, 50, 90)$, such that C^{p-} and C^{p+} form two curves (as shown at the bottom of Figure 4.14).

The area under each curve, $eLoss^1$ and $eLoss^2$ correspondingly, is computed employing the trapezoidal rule. Lastly, the area between both curves (named in the original work as $\Delta eLoss$, equation 4.10) is calculated; if it is positive, the explanation is considered informative and if it is negative, uninformative.

$$\Delta eLoss = eLoss^1 - eLoss^2 \tag{4.10}$$

In the original work, given the $p\%$ most important indexes of w, the transformation is directly applied to these indexes. In the context of our work, the

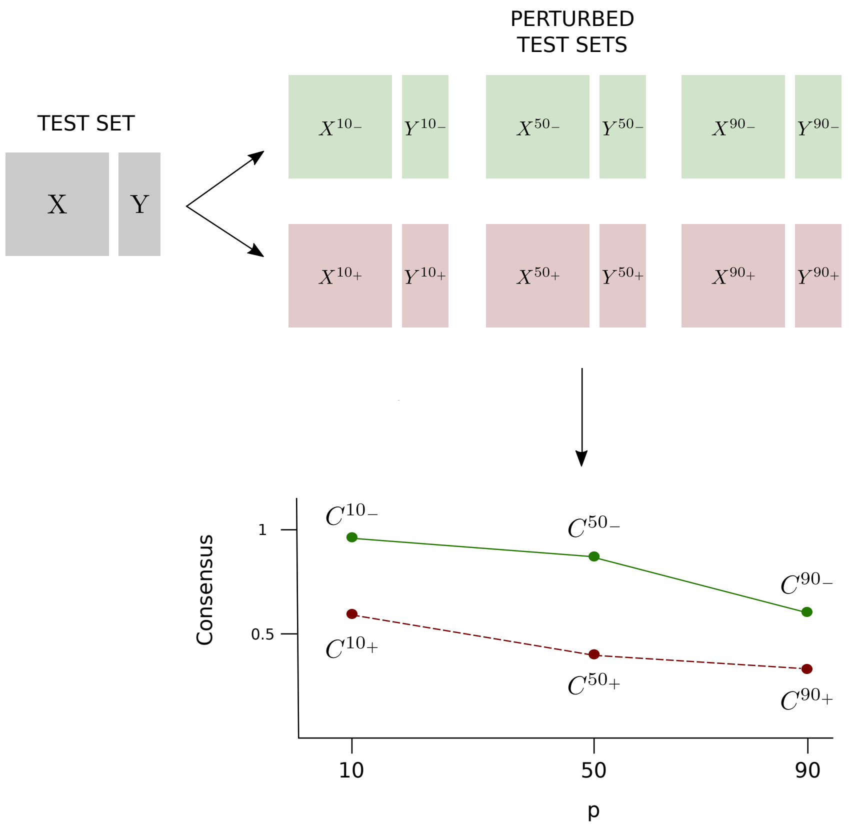


Fig. 4.14: Scheme of the evaluation methodology.

transformations are applied interval-wise instead of index-wise and, hence, an interval in the $p\%$ most important indexes is needed. As such, two scenarios are considered: 1) the $p\%$ most important time indexes form a single interval or 2) the $p\%$ most important time indexes form more than one interval. In the first scenario, a random interval is sampled in this range, and the perturbation is applied to this interval. In the second scenario, the longest interval is chosen, a random interval is sampled in this range and the perturbation is applied to this interval. The case of $p\%$ least important time indexes is analogous.

Lastly, note that the authors of the original work [74] employ different classifiers to obtain the explanations and to evaluate the explanations (i.e., to label the modified test sets). In our work, since the explanation depends on the classifier for which it has been calculated, the same classifier is employed for both purposes.

4.4.3.2 Results

The evaluation is carried out in datasets from the UCR repository that have already been used for the scope of TSC explanations [61, 63, 65]. In particular, the 4 datasets from the UCR employed in [74] are used in this experimentation: CBF, Coffee, ECG200 and GunPoint.

Recall that, in some cases, the prediction may be robust with respect to a given transformation and hence, there is no important region for this instance, classifier and transformation. As such, the evaluation is carried out using only those instances for which the R_{transf} is lower than 1. In order to provide more information to the reader, we also report the dataset-robustness with respect to a transformation, measured by the number of instances in the test set in which the prediction is robust to the transformation.

With the considered settings, $\Delta eLoss$ varies in the range $[-2, 2]$, and an explanation is considered informative if the corresponding $\Delta eLoss$ is positive. The evaluation procedure is repeated 10 times in order to remove the effect of randomness in the interval-wise perturbation process. Tables 4.1, 4.2, 4.3 and 4.4 report the $\Delta eLoss$ and the dataset-robustness between parenthesis for each combination of classifier, dataset and transformation. The accuracy obtained by each classifier in each dataset is also shown in parentheses in the header of the tables.

CBF	1-NN-DTW (0.97)	BOSS (0.99)	ST (0.98)
Warp 0.7	0.33 (0.81)	0.36 (0.99)	0.45 (0.85)
Warp 0.8	0.30 (0.86)	0.64 (0.99)	0.55 (0.91)
Warp 0.9	0.29 (0.89)	0.47 (0.99)	0.42 (0.94)
Warp 1.1	0.36 (0.92)	0.32 (0.99)	0.56 (0.91)
Warp 1.2	0.31 (0.92)	0.46 (0.99)	0.72 (0.86)
Warp 1.3	0.46 (0.93)	0.55 (0.99)	0.65 (0.82)
Scale 0.7	0.48 (0.79)	0.37 (0.95)	0.47 (0.85)
Scale 0.8	0.52 (0.92)	0.43 (0.98)	0.52 (0.91)
Scale 0.9	0.47 (0.98)	0.55 (0.99)	0.27 (0.96)
Scale 1.1	0.78 (0.99)	1.05 (0.99)	0.41 (0.95)
Scale 1.2	0.77 (0.99)	0.66 (0.99)	0.54 (0.93)
Scale 1.3	0.39 (0.99)	0.44 (0.98)	0.46 (0.91)
Noise 1	0.02 (0.99)	0.13 (0.99)	*-0.06* (0.97)
Noise 3	0.10 (0.99)	0.20 (0.99)	0.04 (0.94)
Noise 5	0.11 (0.98)	0.00 (0.99)	0.14 (0.92)
Noise 7	0.10 (0.98)	0.06 (0.99)	0.24 (0.90)
Noise 9	0.14 (0.98)	0.20 (0.99)	0.24 (0.86)
Slice	0.54 (0.03)	0.09 (0.37)	0.07 (0.03)

Table 4.1: The $\Delta eLoss$ obtained for the explanations in CBF dataset. The dataset-robustness is shown in parenthesis.

Coffee	1-NN-DTW (0.98)	BOSS (0.99)	ST (0.99)
Warp 0.7	- (1)	0.23 (0.67)	0.10 (0.25)
Warp 0.8	- (1)	0.20 (0.71)	0.08 (0.57)
Warp 0.9	- (1)	0.85 (0.96)	0.30 (0.89)
Warp 1.1	- (1)	0.35 (0.96)	0.82 (0.89)
Warp 1.2	- (1)	1.25 (0.96)	0.49 (0.78)
Warp 1.3	- (1)	0.95 (0.92)	0.51 (0.75)
Scale 0.7	0.55 (0.25)	0.25 (0.71)	0.23 (0.82)
Scale 0.8	0.61 (0.61)	0.40 (0.92)	0.23 (0.89)
Scale 0.9	1.20 (0.92)	0.70 (0.96)	0.28 (0.89)
Scale 1.1	0.85 (0.93)	0.99 (0.96)	0.05 (0.93)
Scale 1.2	0.51 (0.50)	0.75 (0.96)	0.23 (0.82)
Scale 1.3	0.76 (0.14)	0.26 (0.78)	0.27 (0.75)
Noise 1	- (1)	0.00 (0.96)	1.10 (0.96)
Noise 3	0.10 (0.89)	0.05 (0.96)	0.32 (0.82)
Noise 5	0.24 (0.82)	0.15 (0.92)	0.04 (0.46)
Noise 7	0.07 (0.50)	0.05 (0.93)	*-0.04* (0.39)
Noise 9	0.18 (0.32)	0.20 (0.93)	*-0.05* (0.39)
Slice	0.41 (0.00)	0.79 (0.86)	0.36 (0.86)

Table 4.2: The $\Delta eLoss$ obtained for the explanations in Coffee dataset. The dataset-robustness is shown in parenthesis.

Tables 4.1, 4.2, 4.3 and 4.4 show that the $\Delta eLoss$ is positive in almost all the dataset-transformation-classifier combinations, which means that the explanation is informative. In fact, there are only 7 cases (of the 216 cases) in which it is negative, which are shown in italics. The results clearly validate our explanation method, since, perturbing the important parts change the prediction of the classifier more frequently than perturbing the non-important parts, regardless of the classifier, transformation or dataset. Note that, this is done in spite of the high values of robustness reached.

The dataset-robustness is, generally, very high in all the cases, which means that it is rather hard to change the prediction of the classifiers employing the considered transformations. This can be due to the distribution of the classes within each dataset, or due to the capacity of the classifiers employed in this work to handle the transformations. However, it is worth noting that the dataset-robustness depends on the parameter of the transformation; for the warp and scale transformations, the closer the parameter is to 1, the greater the dataset-robustness, while for the noise transformation, the lower the parameter, the greater the dataset-robustness (which seems reasonable in both cases, since those are the parameters that involve the slightest modifications).

ECG200	1-NN-DTW (0.87)	BOSS (0.89)	ST (0.84)
Warp 0.7	0.58 (0.77)	0.55 (0.43)	0.38 (0.01)
Warp 0.8	0.54 (0.83)	0.40 (0.49)	0.34 (0.20)
Warp 0.9	0.33 (0.84)	0.61 (0.70)	0.54 (0.82)
Warp 1.1	0.58 (0.80)	0.35 (0.71)	0.24 (0.88)
Warp 1.2	0.45 (0.83)	0.39 (0.47)	0.46 (0.73)
Warp 1.3	0.40 (0.83)	0.39 (0.47)	0.56 (0.62)
Scale 0.7	0.66 (0.52)	0.37 (0.68)	0.64 (0.76)
Scale 0.8	0.83 (0.73)	0.33 (0.74)	0.24 (0.87)
Scale 0.9	0.71 (0.88)	0.36 (0.81)	0.26 (0.97)
Scale 1.1	0.71 (0.82)	0.46 (0.80)	0.06 (0.97)
Scale 1.2	0.80 (0.71)	0.47 (0.78)	0.52 (0.92)
Scale 1.3	0.66 (0.59)	0.52 (0.73)	0.28 (0.84)
Noise 1	0.12 (0.89)	0.10 (0.81)	0.00 (0.95)
Noise 3	0.26 (0.74)	*-0.02* (0.78)	0.00 (0.87)
Noise 5	0.21 (0.62)	0.05 (0.68)	0.00 (0.76)
Noise 7	0.25 (0.52)	0.01 (0.62)	0.00 (0.54)
Noise 9	0.27 (0.41)	0.06 (0.58)	0.00 (0.33)
Slice	0.40 (0.06)	*-0.31* (0.55)	*-0.07* (0.54)

Table 4.3: The $\Delta eLoss$ obtained for the explanations in ECG200 dataset. The dataset-robustness is shown in parenthesis.

There are some interesting findings related with the classifiers that deserve attention; on the one hand, the prediction of the 1-NN-DTW classifier is found to be robust to the warp transformation in the Coffee dataset for all the considered warp levels. In the rest of the datasets however, the robustness of the 1-NN-DTW is similar to that of the BOSS and ST classifier, so the ability of a classifier to deal with the warp transformation varies depending on the dataset. On the other hand, the BOSS classifier is known to be robust to noise [69], but, in the ECG200 dataset, for example, both the 1-NN-DTW and the ST classifiers are found to be more robust to noise.

4.5 Conclusions and Future Work

In this Chapter, an explanation method for TSC that provides realistic -and specific to time series- explanations is proposed. For this, 4 transformations for time series are presented (warp, scale, noise and slice) and a synthetic neighbourhood of a time series is created by applying these transformations to random intervals of the series. The method provides explanations at two levels: in the high-level, the robustness of a classifier's prediction with respect

GunPoint	1-NN-DTW (0.95)	BOSS (0.99)	ST (0.99)
Warp 0.7	0.58 (0.88)	0.21 (0.57)	0.23 (0.75)
Warp 0.8	0.69 (0.92)	0.48 (0.86)	0.03 (0.93)
Warp 0.9	0.54 (0.94)	0.63 (0.96)	*-0.39* (0.97)
Warp 1.1	0.83 (0.93)	0.32 (0.97)	0.89 (0.98)
Warp 1.2	0.76 (0.93)	0.54 (0.91)	0.70 (0.98)
Warp 1.3	0.41 (0.87)	0.55 (0.82)	0.78 (0.98)
Scale 0.7	0.47 (0.28)	0.32 (0.73)	0.30 (0.65)
Scale 0.8	0.51 (0.36)	0.40 (0.88)	0.36 (0.77)
Scale 0.9	0.45 (0.53)	0.18 (0.96)	0.29 (0.94)
Scale 1.1	0.64 (0.54)	0.32 (0.98)	0.60 (0.97)
Scale 1.2	0.76 (0.37)	0.32 (0.90)	0.40 (0.91)
Scale 1.3	0.83 (0.30)	0.41 (0.84)	0.37 (0.78)
Noise 1	0.46 (0.84)	0.00 (0.99)	0.14 (0.97)
Noise 3	0.43 (0.67)	0.02 (0.89)	0.09 (0.85)
Noise 5	0.44 (0.55)	0.06 (0.67)	0.14 (0.66)
Noise 7	0.39 (0.40)	0.03 (0.56)	0.24 (0.49)
Noise 9	0.43 (0.33)	0.07 (0.54)	0.60 (0.49)
Slice	0.01 (0.11)	0.74 (0.83)	0.75 (0.49)

Table 4.4: The $\Delta eLoss$ obtained for the explanations in GunPoint dataset. The dataset-robustness is shown in parenthesis.

to a given transformation is measured, while in the low-level, the relevance of each region of the series in the prediction is computed.

The experimentation is divided into a qualitative and a quantitative evaluation. In the former, the explanation provided by our method is visually validated in some time series extracted from datasets of the UCR repository. Taking advantage of the semantic meaning of the time series, our methods show, for example, that the speed of the action recorded in the GunPoint dataset is discriminant. In the latter, the proposed method is quantitatively evaluated by adapting an existing evaluation methodology [74] to the context of our transformations. The methodology consists of perturbing the time series in the important and non-important regions (according to the explanation), and checking whether the perturbations in the important regions change the prediction of the classifier more frequently than perturbations in the non-important regions. The results show that this holds in almost all the considered combinations of dataset-classifier-transformation, which confirms the informativeness of our method. Moreover, the dataset-robustness (measured by the number of time series in the test set for which the prediction is robust to a transformation), give some insights into how robust the classifiers are to the considered transformations; the BOSS classifier, for example, which is supposed to be very robust to noise, is shown to be less robust to noise than the 1-NN-DTW or the ST classifier in the ECG200 dataset from the UCR repository.

Lastly, there are many possible directions for the future work: due to the lack of space, the experimentation in this work is limited, but it could be extended to more classifiers and datasets, and, eventually, to the multi-variate TSC scenario. Moreover, in this method, the transformations have been studied independently, while it could be interesting to study combinations of transformations, for example, by successively applying different transformations to random intervals. To conclude, we think that our method is a first attempt at empirically studying the robustness in TSC, and it could be interesting to continue in this direction.

5 General Conclusions and Future Work

5.1 Conclusions

In this thesis, three contributions to the field of time series classification have been proposed. In Chapter 2, a review that organizes and comprehensively revises the existing distance based time series classification methods is presented. The methods are categorized into three groups, depending on how each approach uses the chosen distance: together with the 1-NN classifier, to obtain new distance based features or to construct kernels. The approaches that use the distance together with the 1-NN classifier have been widely reviewed and we refer the reader to [76, 78, 166] for more details.

Regarding those methods that employ the distance to obtain a new feature representation of the series, we distinguish between three approaches: global distance features, local distance features, and embedded features. Global distance features are obtained by computing the distance matrix between the time series in the training set and employing these distances to represent the series in a new vector space, in which any non-temporal classifier can be applied. In the methods that employ local distance features, instead of computing the distance between the series, distances between the series and some representative patterns (subsequences) of the series are computed. Lastly, given that many classifiers are implicitly built on Euclidean spaces, some of the methods aim at isometrically embedding the distance matrix obtained from the training set into some Euclidean space.

Regarding those methods that employ the distance to construct kernels, two main directions are followed: those that construct indefinite kernels, and those that construct definite kernels. Time series distances do not generally lead to PSD kernels and many of the methods just employ the indefinite kernels without any special treatment or consideration. Recall that the definiteness guarantees the mathematical properties of a kernel, but the relationship between the (grade of) definiteness and the accuracy obtained by employing the kernel within a classifier is still an open question. In this context, some of the methods apply different types of regularizations to convert the indefinite kernels into definite kernels, while some others study the relationship between the definiteness and the accuracy more in-depth. Lastly, there are also some works in which specific kernels for time series are proposed that are, by definition, PSD.

In Chapter 3, with the goal of providing a practical tool for a non-expert user, we develop the first time series classifier recommendation system. For this objective, a meta-learning approach is followed to define a a characterization of supervised time series datasets based on landmarkers. We choose the classifiers included in [171] and implemented in the tsml [71] or sktime [72] toolboxes as candidate classifiers. As such, our proposal includes 24 landmarkers for time series classifiers, obtained by dataset subsampling and algorithm reductions. The experimental analysis shows that the proposed landmarkers fulfill the conditions to be good landmarkers: they are fast to compute (in one of the biggest datasets from the UCR repository, the computation time is reduced from 12 days to 44 minutes), and the performances obtained by the landmarkers are highly correlated with the results obtained by the original classifiers (with a mean correlation of 0.86).

The recommendation system we proposed is based on 5 types of meta-targets: classifiers accuracies, complete ranking, partial ranking, best set, and best classifier. For each meta-target type, the performance of a specific meta-learner based on the designed landmarkers is compared with the performance of a baseline especially designed for this experimentation and the performance of the standard meta-features. The results clearly validate our recommendation system, which beats the baselines and standard meta-features in 7 of the 9 considered scenarios.

The last part of the recommendation system relies on exploiting the hierarchical relationship between the meta-targets. Some meta-targets are more fine-grained than others, which allows to infer from them other less fine-grained meta-targets: the complete ranking, for instance, can be directly obtained by the classifier accuracies. In this sense, we have experimentally studied if, given a meta-target, it is worth employing the specific meta-learner for this meta-target or, instead, similar performances can be obtained by inferring the latter from more fine-grained meta-targets. We showed that, in our framework, most of the meta-target types can be inferred from a single linear multi-output regression on the classifiers accuracies, obtaining even better results than those obtained with the corresponding specific meta-learner.

Lastly, the third contribution of this dissertation is presented in Chapter 4. In this chapter, with the aim of providing a deeper understanding of the existing classifiers, we present an ad-hoc agnostic time series classification explanation method. A perturbation based approach is followed for obtaining the explanation, together with 4 specific and ah-hoc perturbations that we propose for time series: warp, scale, noise and slice. With this perturbations, the proposed method provides a two-level explanation: on the one hand, the robustness of a prediction with respect to a transformation is computed and, on the other hand, a visualization of the series in which the relevance of each region of the series respect to the transformation is provided.

Given that the evaluation of explanation methods can be very subjective, we tackle it in two manners. Firstly, a qualitative evaluation is carried out in two well known time series datasets from the UCR repository. Secondly, a quantitative ad-hoc evaluation methodology is proposed for the framework of this work based on the evaluation procedure proposed in [74]. In this procedure, an explanation method is considered informative if perturbations on the relevant regions change the prediction more frequently than perturbations on the non-relevant regions. The results obtained by this evaluation methodology clearly validate our explanation method, which is found to be informative in almost all the considered combinations of dataset-classifier-transformation.

5.2 Future Works

The contributions presented in this thesis have led to many new potential directions of research. In the following paragraphs, we briefly present the possible future directions:

- *Extensions to other types of time series*

  The contributions presented in Chapters 3 and 4 have mainly focused on univariate time series with finite and equal lengths. However, most of the work can be easily adapted to multi-variate or unequal length time series. In the the case of the recommendation system, this could be done by designing landmarkers for the classifiers that can handle multi-variate or unequal length time series. In Chapter 4, the classifiers considered can deal with unequal length time series, so, even if we have not carried it out because of the lack of space, the framework could be directly applied to obtain explanations for this type of time series. Given that many of the time series collected in real world scenarios are multi-variate or have unequal lengths, we think that the extension of our work to this type of data is a promising future work.

- *Temporal information*

  In the TSC community, it is assumed that specific methods for time series perform better than standard classifiers. However, there are certain problems in which standard classifiers (which are, generally, faster) are more suitable [8]. Since standard classifiers do no take the order of the values of the time series (the temporal information) into account, it means that there are TSC problems in which the temporal information is not relevant for the classification. In this sense, it would be interesting to study whether, given a TSC problem, the temporal information is a relevant characteristic for the classification or not.

- *Discriminatory characteristics of datasets and performance of classifiers*

  In the contribution presented in Chapter 3, we have addressed the problem of classifier recommendation by predicting the performance of time series classifiers in certain problems. However, it would be interesting to investigate which are the characteristic of a dataset that make a classifier perform better than the others in a certain problem. In particular, given that time series datasets have specific discriminatory characteristic, understanding the relationship between the discriminatory characteristic of a dataset and the accuracy of different classifiers would be extremely valuable.

- *Robustness of classifiers*

  In TSC, it is common to state that a particular classifier is robust to noise or to time warping, but these statements have never been empirically proven. In Chapter 4, we have presented a quantitative measure to determine the robustness of a prediction (made by a certain classifier for a given time series) with respect to the proposed time series transformations. We think that the idea of the dataset-robustness proposed in our work could be used to study and compare the robustness of certain classifiers in different TSC problems.

5.3 Main Achievements

Journal Papers

- **Abanda, A.**, Mori, U., Lozano, J. A. (2019). A review on distance based time series classification. Data Mining and Knowledge Discovery, 33(2), 378-412. https://doi.org/10.1007/s10618-018-0596-4
- **Abanda, A.**, Mori, U., Lozano, J. A. (2020). Time Series Classifier Recommendation by a Meta-Learning Approach. Submitted (major Revision).
- **Abanda, A.**, Mori, U., Lozano, J. A. (2021). Ad-Hoc Explanation for Time Series Classification. Submitted.

Conference Papers

- **Abanda, A.**, Mori, U., Lozano, J. A. (2018). ¿Requiere la clasificación de series temporales métodos específicos?. TAMIDA , XVIII Conferencia de la Asociación Española para a Inteligencia Artificial (CAEPIA 2018). Granada, Spain.
- **Abanda, A.**, Mori, U., Lozano, J. A. (2018). PhD Thesis proposal: A study on the discriminatory capacity of the temporal information on supervised time series classification problems. Doctoral Consortium, XVIII Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA 2018). Granada, Spain.

Posters

- **Abanda, A.**, Mori, U., Lozano, J. A. (2018). A preliminary study about the need of specific methods in time series classification. The 3rd Bilbao Data Science Workshop, BIDAS18, BCAM, Bilbao.
- **Abanda, A.**, Mori, U., Lozano, J. A. (2019). Feature Extraction for Algorithm-Type Selection in Time Series Classification. The 7th EITIC Doctoral Student Workshop, Universitat Pompeu Fabra (UPF), Barcelona.
- **Abanda, A.**, Mori, U., Lozano, J. A. (2019). An Exploratory Analysis of the Influence of the Temporal Information in Time Series Classification. Learning from data streams and time series: convergences, specificities and common challenges, FDST2019, Telecom, Paris.

Workshops

- 5-day toolbox development and coding workshop for *Sktime* and *MLJ* projects. Participation as code contributor. University College London (UCL), London

Short Visits

- 10 September-19 December 2020: LINKMEDIA Team (IRISA, INRIA). University of Rennes, France. Supervisor: Simon Malinowski.

Awards

- Best doctoral thesis project at the XVIII Conference of the Spanish Association for Artificial Intelligence (CAEPIA), 2018.

Dissemination

- Intelligent Systems Group Seminar in UPV/EHU Donostia
- Dissemination talk to the member of Aupatuz during their visit to BCAM, 22 June, 2018, Bilbao.
- Organization of the visit to BCAM of students of La Salle Bilbao, 26 April, 2018. Organization of three talks and a workshop for building a 3D hologram, Bilbao.
- Dissemination talk within the festival Pint of Science entitled: âĂIJInteligencia Artificial: progresos y amenazasâĂİ, 22 May 2019, Bilbao.

Software

- https://gitlab.bcamath.org/aabanda/tscr: Time Series Classifier Recommendation by a Meta-Learning Approach.

- https://gitlab.bcamath.org/aabanda/tscexplanation: Ad-Hoc Explanation for Time Series Classification.

A

Appendix

A.1 Landmarkers: computation time of subsample landmarkers in the largest datasets from the UCR repository

	MixedShapesRegularTrain	*HandOutlines*	*NonInvasiveFetalECGThorax1*	*StarlightCurves*	*UWaveGestureLibraryAll*
BOP	0.01	0.01	0.01	0.01	0.01
BOSS	**131.62**	**60.79**	**32.65**	16.16	22.45
BayesNet	0.35	0.02	0.04	0.03	0.02
C45	0.36	0.05	0.05	0.01	0.02
DD_DTW	**176.52**	**142.27**	**64.95**	**41.42**	**41.84**
DTD_C	**847.53**	**344.01**	**128.27**	**65.47**	**72.08**
DTW	0.03	0.04	0.01	0.01	0.01
ERP	0.01	0.01	0.01	0.01	0.01
FS	0.01	0.01	0.01	0.01	0.01
MLP	**141.11**	**220.46**	20.08	25.86	14.37
MSM	0.23	0.10	0.04	0.03	0.04
NB	0.26	0.01	0.01	0.01	0.01
NN	0.01	0.01	0.01	0.01	0.01
NN_CID	0.01	0.01	0.01	0.01	0.01
RandF	0.32	0.05	0.11	0.05	0.05
RotF	27.42	10.55	8.56	2.23	5.71
SVML	0.46	0.02	0.13	0.02	0.04
SVMQ	0.03	0.02	0.07	0.01	0.01
ST	**614.89**	**62.17**	**69.68**	**62.17**	**62.22**
TSF	3.83	1.12	2.21	0.46	0.88
TWE	0.36	0.17	0.07	0.05	0.04
WDTW	0.10	0.05	0.01	0.01	0.01
ResNet	**87.56**	**190.30**	**168.18**	**131.06**	**127.50**
InceptionTime	**78.24**	**85.86**	**115.32**	**96.03**	25.84

Table A.1: For the 5 largest datasets in the UCR, the time (in minutes) spent in computing the subsample landmarkers (those that take more than 30 minutes are in bold).

A.2 Landmarkers: computation times

Dataset	**Total time**	**Slowest (time)**
Adiac	14.74	ST (9.87)
ArrowHead	9.28	ST (6.23)
Beef	8.42	ST (6.19)
BeetleFly	8.45	ST (5.83)
BirdChicken	8.05	ST (5.66)
Car	10.9	ST (6.02)
CBF	7.53	ST (5.91)
ChlorineConcentration	12.9	ST (9.11)
CinCECGTorso	112.32	BOSS (47.34)
Coffee	8.9	ST (5.67)
Computers	17.21	ST (6.04)
CricketX	16.59	ST (8.5)
CricketY	15.63	ST (7.91)
CricketZ	14.38	ST (8.32)
DiatomSizeReduction	9.02	ST (5.91)
DistalPhalanxOutlineCorrect	6.28	ST (3.81)
DistalPhalanxOutlineAgeGroup	8.23	ST (6.39)
DistalPhalanxTW	8.41	ST (6.68)
Earthquakes	13.28	ST (6.08)
ECG200	3.97	ST (1.97)
ECG5000	8.55	ST (6.01)
ECGFiveDays	7.88	ST (6.25)
ElectricDevices	19.43	ST (14.19)
FaceAll	11.38	ST (7.13)
FaceFour	8.13	ST (5.76)
FacesUCR	9.66	ST (7.06)
FiftyWords	56.49	ST (41.31)
Fish	10.82	ST (6.15)
FordA	34.36	BOSS (11.92)
FordB	58.16	BOSS (29.22)
GunPoint	7.71	ST (5.9)
Ham	12.27	ST (6.18)
HandOutlines	129.6	BOSS (59.91)
Haptics	35.48	ST (6.71)
Herring	10.3	ST (5.93)
InlineSkate	68.54	BOSS (25.75)
InsectWingbeatSound	12.27	ST (6.34)
ItalyPowerDemand	1.43	Resnet (0.5)
LargeKitchenAppliances	32.09	BOSS (15.6)
Lightning2	12	ST (6.01)
Lightning7	8.74	ST (5.96)
Mallat	44.04	BOSS (17.91)
Meat	9.07	ST (5.9)
MedicalImages	8.47	ST (6.72)
MiddlePhalanxOutlineCorrect	6.62	ST (4.52)
MiddlePhalanxOutlineAgeGroup	9.33	ST (7.07)
MiddlePhalanxTW	10.47	ST (7.77)
MoteStrain	10.94	ST (8.02)
NonInvasiveFetalECGThorax1	98.48	BOSS (36.56)
NonInvasiveFetalECGThorax2	98.4	BOSS (35.09)
OliveOil	8.37	ST (6.05)
OSULeaf	11.59	ST (6.13)
PhalangesOutlinesCorrect	9.19	ST (6.94)
Phoneme	209.38	RotF (89.85)
Plane	7.6	ST (5.98)
ProximalPhalanxOutlineCorrect	6.59	ST (4.66)
ProximalPhalanxOutlineAgeGroup	9.7	ST (7.3)
ProximalPhalanxTW	9.78	ST (7.07)
RefrigerationDevices	22.67	ST (7.17)
ScreenType	43.33	BOSS (25.19)
ShapeletSim	12.43	ST (6.07)
ShapesAll	56.41	BOSS (24.06)
SmallKitchenAppliances	42.22	BOSS (24.57)
SonyAIBORobotSurface1	8.96	ST (7.02)
SonyAIBORobotSurface2	5.58	ST (3.37)
StarLightCurves	44.3	BOSS (19.67)
Strawberry	9.96	ST (6.34)
SwedishLeaf	9.08	ST (7.04)
Symbols	14.41	ST (6.12)
SyntheticControl	4.4	ST (2.6)
ToeSegmentation1	11	ST (6.31)
ToeSegmentation2	9.85	ST (5.97)
Trace	9.4	ST (6.12)
TwoLeadECG	9.25	ST (7.14)
TwoPatterns	9.61	ST (6.73)
UWaveGestureLibraryX	35.03	ST (12.78)
UWaveGestureLibraryY	20.8	ST (7.07)
UWaveGestureLibraryZ	36.13	ST (17.24)
UWaveGestureLibraryAll	68.52	BOSS (31.69)
Wafer	9.2	ST (6.65)
Wine	7.4	ST (5.65)
WordSynonyms	15.25	ST (7.42)
Worms	17.17	ST (6.14)
WormsTwoClass	16.19	ST (6.05)
Yoga	20.9	ST (7.84)
ACSF1	46.76	BOSS (16.57)
BME	11.44	ST (6.81)
Chinatown	4.86	Resnet (2.84)
EOGHorizontalSignal	121.21	BOSS (66.99)
EOGVerticalSignal	100.25	BOSS (42.54)
FreezerRegularTrain	13.84	ST (6.74)
FreezerSmallTrain	11.38	ST (6.62)
GunPointAgeSpan	9.27	ST (6.23)
GunPointMaleVersusFemale	9.04	ST (6.17)
GunPointOldVersusYoung	8.55	ST (6.28)
HouseTwenty	25.07	BOSS (7.23)
InsectEPGRegularTrain	13.48	ST (5.92)
InsectEPGSmallTrain	14.52	ST (5.95)
MixedShapesSmallTrain	24.91	BOSS (7.89)
PigAirwayPressure	72.53	BOSS (28.4)
PigArtPressure	84.78	BOSS (36.01)
PigCVP	122.92	BOSS (67.03)
PowerCons	10.39	ST (6.01)
Rock	64.99	BOSS (34.39)
SemgHandGenderCh2	38.06	BOSS (17.34)
SemgHandMovementCh2	53.09	BOSS (23.1)
SemgHandSubjectCh2	68.49	BOSS (37.51)
SmoothSubspace	4.02	Resnet (3.15)
UMD	9.42	ST (6.3)
Crop	58.15	ST (42.1)
EthanolLevel	43.82	BOSS (13.48)
MixedShapesRegularTrain	28.72	BOSS (9.93)

Table A.2: For each dataset, the total time (in minutes) spent in computing all the landmarkers, the slowest classifier and the corresponding time (in minutes).

A.3 Landmarkers: Correlation between the landmarkers and the corresponding original classifiers

Classifier	**Correlation**
BOP	0.90
BOSS	0.90
BayesNet	0.93
C45	0.89
DD_DTW	0.87
DTD_C	0.83
DTW	0.84
ERP	0.84
FS	0.93
MLP	0.89
MSM	0.89
NB	0.95
NN	0.90
NN_CID	0.85
RandF	0.90
RotF	0.88
SVML	0.93
SVMQ	0.90
ST	0.87
TSF	0.92
TWE	0.82
WDTW	0.92
ResNet	0.60
InceptionTime	0.40

Table A.3: The Pearson correlation between the accuracies obtained by the landmarkers and the corresponding original classifiers in the 112 datasets from the UCR.

A.4 Landmarkers: Accuracies of classifiers and landmarkers by dataset

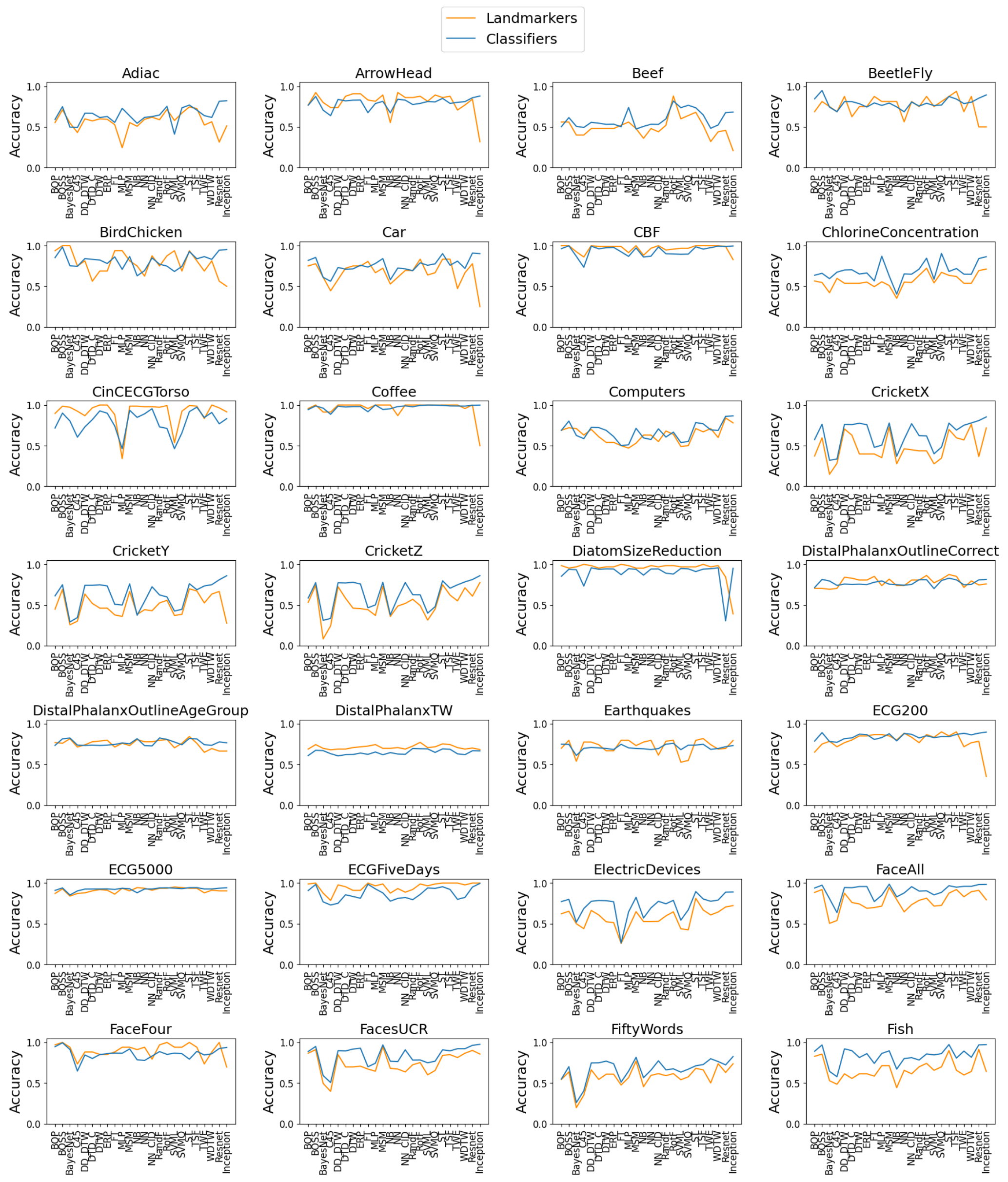


Fig. A.1: Accuracies of classifiers and landmarkers by dataset I.

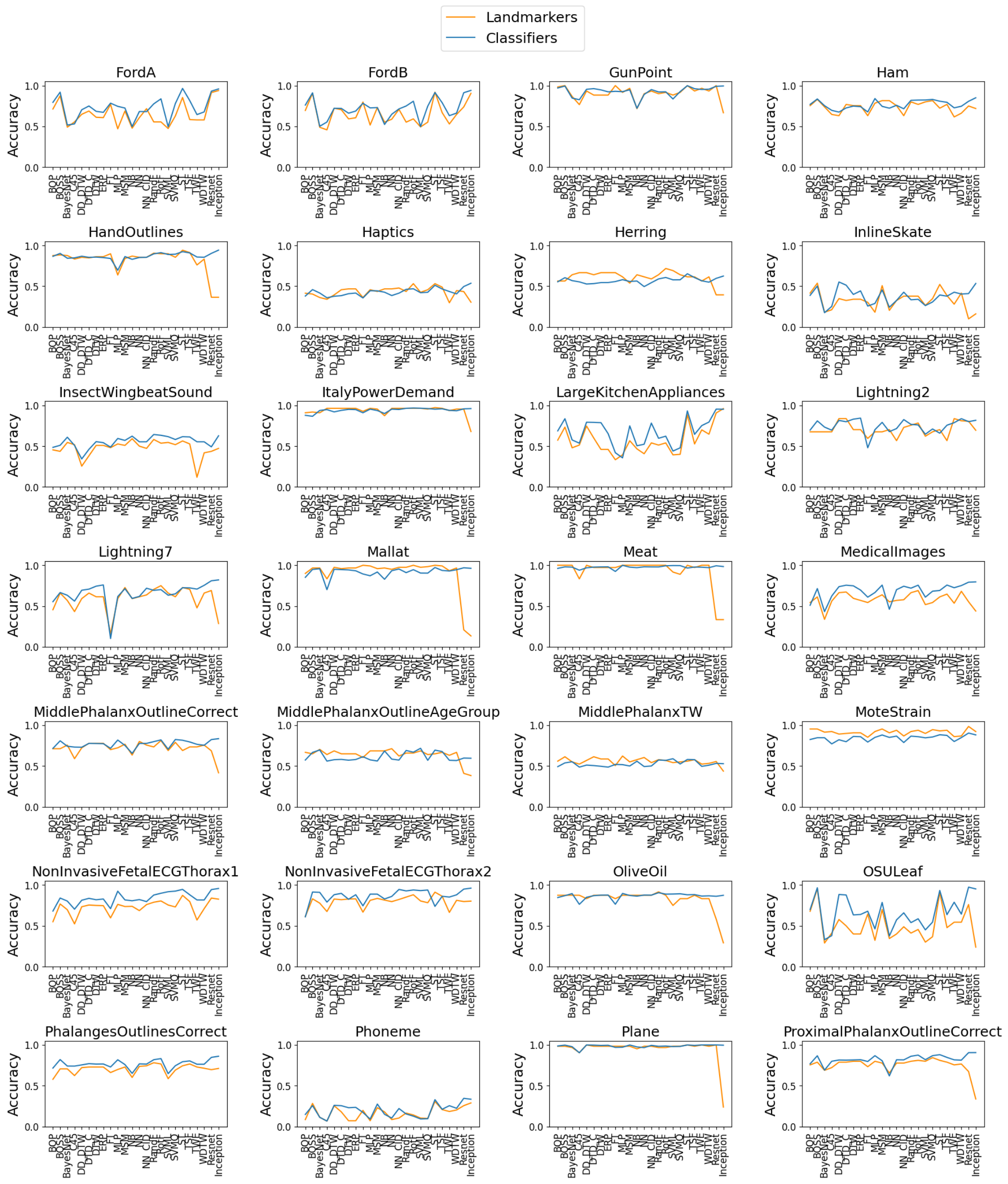


Fig. A.2: Accuracies of classifiers and landmarkers by dataset II.

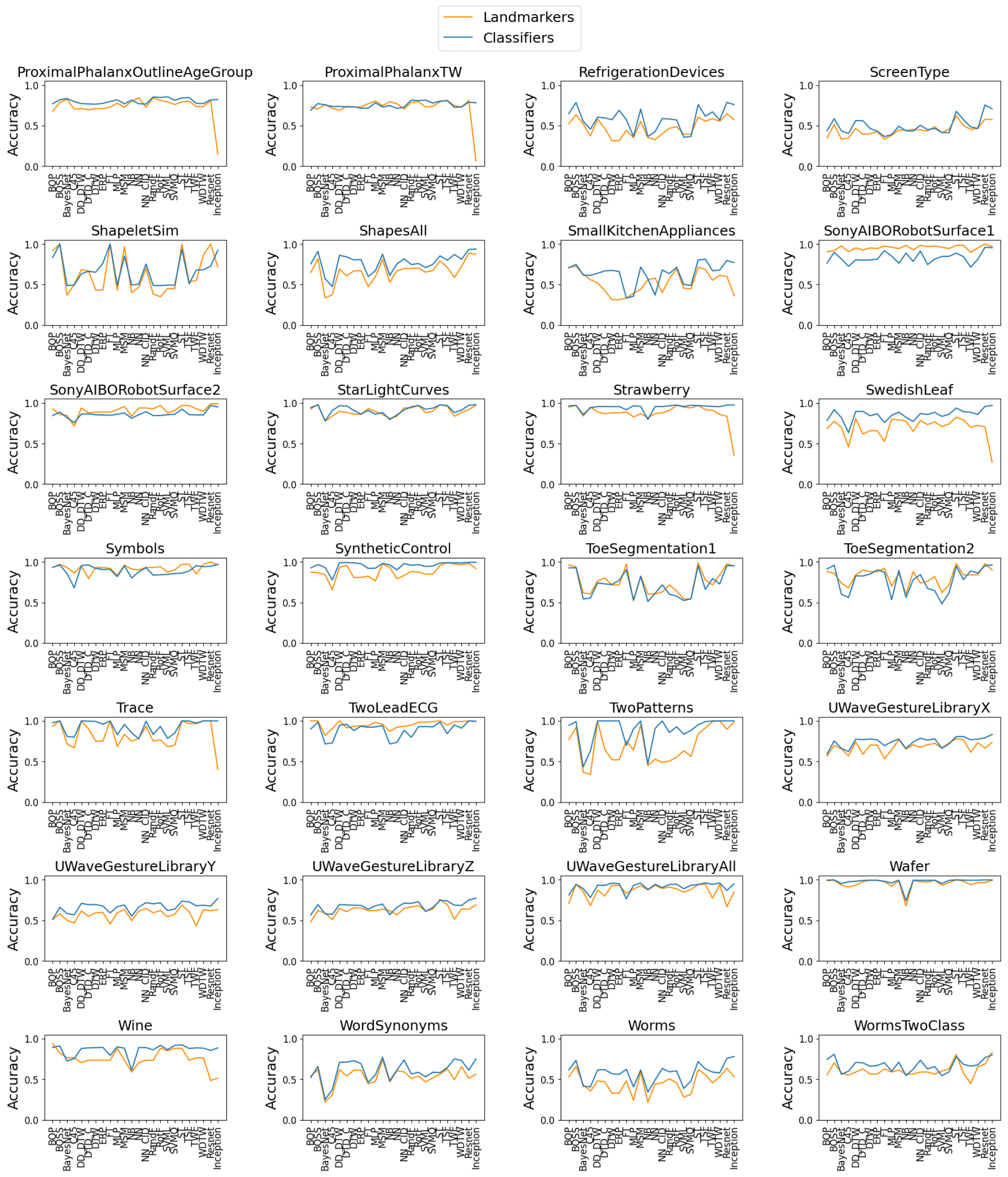


Fig. A.3: Accuracies of classifiers and landmarker by dataset III.

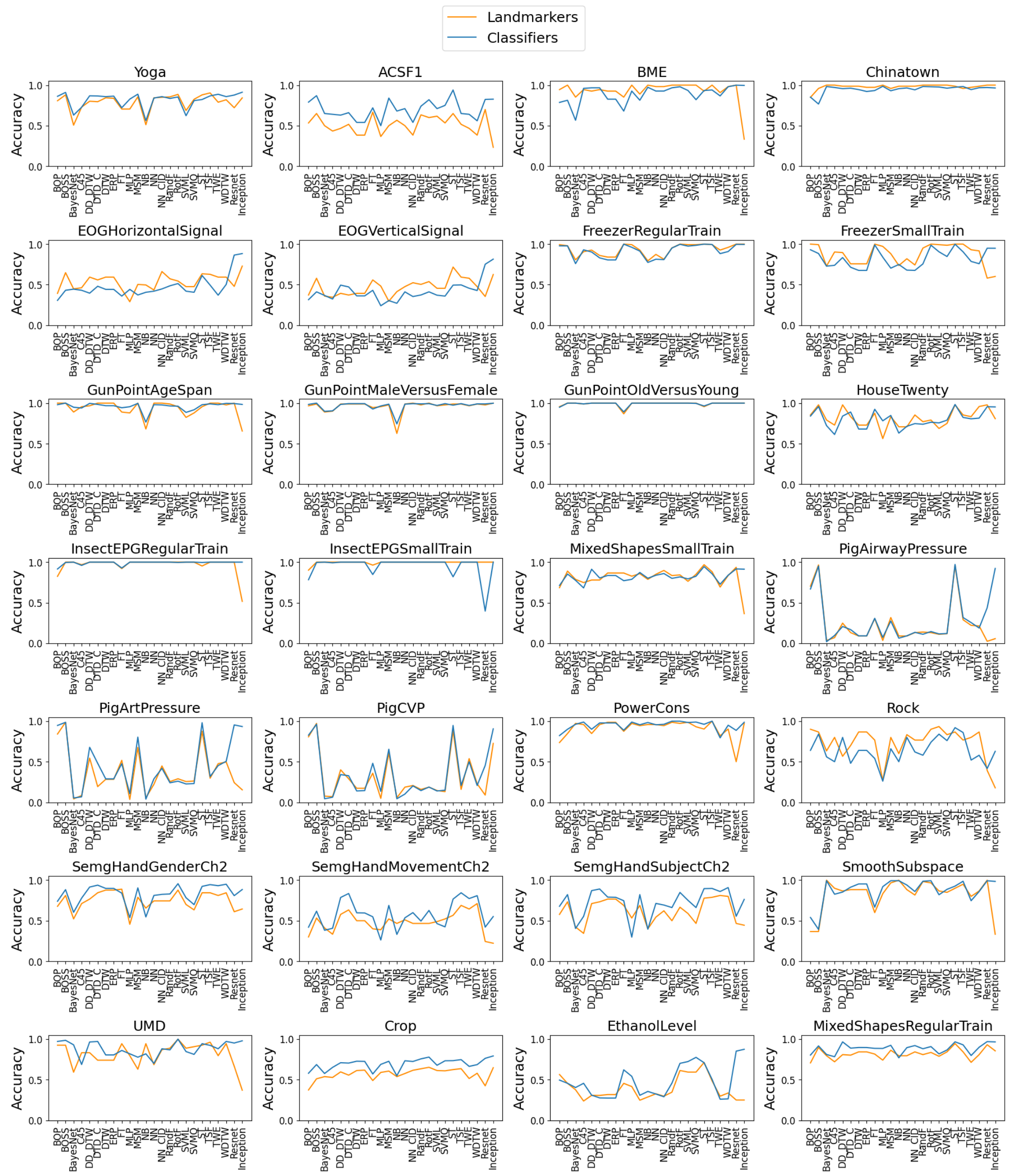


Fig. A.4: Accuracies of classifiers and landmarker by dataset IV.

References

[1] Eamonn Keogh and Shruti Kasetty. On the need for time series data mining benchmarks. *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 102, 2002. ISSN 13845810. doi: 10.1145/775047.775062. URL http://dl.acm.org/citation.cfm?id=775047.775062.

[2] Anthony Bagnall, Jason Lines, William Vickers, and Eamonn Keogh. The UEA & UCR Time Series Classification Repository. 2015. URL www.timeseriesclassification.com.

[3] Tak Chung Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181, 2011. ISSN 09521976. doi: 10.1016/j.engappai.2010.09.007. URL http://dx.doi.org/10.1016/j.engappai.2010.09.007.

[4] Philippe Esling and Carlos Agon. Time-Series Data Mining. *ACM Computing Surveys*, 45(1):1–34, 2012. ISSN 03600300. doi: 10.1145/2379776.2379788. URL http://dl.acm.org/citation.cfm?doid=2379776.2379788.

[5] F J Nogales, J Contreras, A J Conejo, and R Espinola. Forecasting Next-Day Electricity Prices by Time Series Models. *IEEE Transactions on Power Systems*, 17(2):342–348, 2002.

[6] T. Warren Liao. Clustering of time series data - A survey. *Pattern Recognition*, 38(11):1857–1874, 2005. ISSN 00313203. doi: 10.1016/j.patcog.2005.01.025.

[7] The Plasticc team, Tarek Allam Jr, Anita Bahmanyar, Rahul Biswas, and Mi Dai. The Photometric LSST Astronomical Time-series Classification Challenge (PLAsTiCC): Data set. *arXiv preprint arXiv:1810.00001*, pages 1–15, 2018.

[8] Anthony Bagnall, Jason Lines, Aaron Bostrom, James Large, and Eamonn Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 31(3):606–660, 2017. ISSN 1573756X. doi: 10.1007/s10618-016-0483-9.

[9] Zhengzheng Xing, Jian Pei, and Eamonn Keogh. A brief survey on sequence classification. *ACM SIGKDD Explorations Newsletter*, 12(1): 40, 2010. ISSN 19310145. doi: 10.1145/1882471.1882478. URL http://portal.acm.org/citation.cfm?doid=1882471.1882478.

[10] Donald Berndt and James Clifford. Using dynamic time warping to find patterns in time series. *Workshop on Knowledge Knowledge Discovery in Databases*, 398:359–370, 1994. URL http://www.aaai.org/Papers/Workshops/1994/WS-94-03/WS94-03-031.pdf.

[11] Toni Giorgino. Computing and Visualizing Dynamic Time Warping Alignments in R : The dtw Package. *Journal of Statistical Software*, 31 (7):1–24, 2009. ISSN 1548-7660. doi: 10.18637/jss.v031.i07. URL http://www.jstatsoft.org/v31/i07{%}5Cnhttp://www.jstatsoft.org/v31/i07/.

[12] Xiaopeng Xi, Eamonn Keogh, Christian Shelton, Li Wei, and Chotirat Ann Ratanamahatana. Fast time series classification using numerosity reduction. *Proceedings of the 23rd ICML International Conference on Machine learning*, pages 1033—-1040, 2006. doi: 10.1145/1143844.1143974. URL http://dl.acm.org/citation.cfm?id=1143974.

[13] D. S. Hirschberg. Algorithms for the longest common subsequence problem. *Journal of the ACM (JACM)*, pages 664–675, 1977.

[14] Pierre-François Marteau. Time Warp Edit Distance with Stiffness Adjustment for Time Series Matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):306–318, 2009.

[15] A Stefan and Das G Athitsos V. The move-split-merge metric for time series. *IEEE Transactions on Knowledge and Data Engineering*, pages 1425–1438, 2013.

[16] Rohit J. Kate. Using dynamic time warping distances as features for improved time series classification. *Data Mining and Knowledge Discovery*, 30(2):283–312, 2015. ISSN 1384-5810. doi: 10.1007/s10618-015-0418-x. URL http://link.springer.com/10.1007/s10618-015-0418-x.

[17] Marco Cuturi and Jp Vert. A kernel for time series based on global alignments. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 1:413–416, 2007. ISSN 1520-6149. doi: 10.1109/ICASSP.2007.366260. URL http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=4217433.

[18] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[19] Christos Faloutsos, M. Ranganathan, and Yannis Manolopoulos. Fast subsequence matching in time-series databases. *ACM SIGMOD International Conference on Management of Data*, pages 419–429, 1994. ISSN 01635808. doi: 10.1145/191843.191925.

[20] I. Popivanov and R. J. Miller. Similarity Search Over Time-Series Data Using Wavelets. *Proceedings 18th International Conference on Data Engineering (ICDE)*, pages 212–221, 2002. ISSN 1063-6382. doi: 10.1109/ICDE.2002.994711.

[21] Anthony Bagnall and Gareth Janacek. A run length transformation for discriminating between auto regressive time series. *Journal of Classification*, 31(October):274–295, 2014. ISSN 01764268. doi: 10.1007/s00357-.

[22] Marcella Corduas and Domenico Piccolo. Time series clustering and classification by the autoregressive metric. *Computational Statistics and Data Analysis*, 52(4):1860–1872, 2008. ISSN 01679473. doi: 10.1016/j.csda.2007.06.001.

[23] Padhraic Smyth. Clustering sequences with hidden Markov models. *Advances in Neural Information Processing Systems*, 9:648–654, 1997. ISSN 10495258. doi: 10.1017/CBO9781107415324.004. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.44.3648.

[24] Houtao Deng, George Runger, Eugene Tuv, and Martyanov Vladimir. A time series forest for classification and feature extraction. *Information Sciences*, 239:142–153, 2013. ISSN 0020-0255. doi: 10.1016/j.ins.2013.02.030. URL http://dx.doi.org/10.1016/j.ins.2013.02.030.

[25] Mustafa Gokce Baydogan, George Runger, and Eugene Tuv. A Bag-of-Features Framework to Classify Time Series. *EEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2796–2802, 2013.

[26] Lexiang Ye and Eamonn Keogh. Time series shapelets: A New Primitive for Data Mining. *Proceedings of the 15th ACM SIGKDD International conference on Knowledge Discovery and Data Mining*, page 947, 2009. doi: 10.1145/1557019.1557122. URL http://portal.acm.org/citation.cfm?doid=1557019.1557122.

[27] Thanawin Rakthanmanon and Eamonn Keogh. Fast Shapelets: A Scalable Algorithm for Discovering Time Series Shapelets. *Proceedings of the 13th ICDM International Conference on Data Mining*, pages 668–676, 2013. doi: 10.1137/1.9781611972832.74. URL http://epubs.siam.org/doi/abs/10.1137/1.9781611972832.74.

[28] Jon Hills, Jason Lines, Edgaras Baranauskas, James Mapp, and Anthony Bagnall. Classification of time series by shapelet transformation. *Data Mining and Knowledge Discovery*, 28(4):851–881, 2014. ISSN 13845810. doi: 10.1007/s10618-013-0322-1.

[29] Jessica Lin, Rohan Khade, and Yuan Li. Rotation-invariant similarity in time series using bag-of-patterns representation. *Journal of Intelligent Information Systems*, 39(2):287–315, 2012. ISSN 09259902. doi: 10.1007/s10844-012-0196-5.

[30] Patrick Schäfer. The BOSS is concerned with time series classification. *Data Mining and Knowledge Discovery*, pages 1505–1530, 2015. ISSN 1384-5810. doi: 10.1007/s10618-014-0377-7. URL http://dx.doi.org/10.1007/s10618-014-0377-7.

[31] Anthony Bagnall, Jason Lines, Jon Hills, and Aaron Bostrom. Time-series classification with COTE: The collective of transformation-based ensembles. *IEEE Transactions on Knowledge and Data Engineering*, 27(9):1548–1549, 2016. ISSN 10414347. doi: 10.1109/ICDE.2016.7498418.

[32] P. B. Brazdil, C. Soares, and J. P. Da Costa. Ranking Learning Algorithms: Using IBL and Meta-Learning on Accuracy and Time Results. *Machine Learning*, pages 251–277, 2003.

[33] Rodrigo G F Soares, Daniel S A De Araujo, Ivan G Costa, Teresa B Ludermir, and Alexander Schliep. Ranking and Selecting Clustering Algorithms Using a Meta-Learning Approach. *IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence*, pages 3729–3735, 2008.

[34] Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. Initializing Bayesian Hyperparameter Optimization via Meta-Learning. *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence 1128*, pages 1128–1135, 2014.

[35] Pavel Brazdil, Christophe Giraud-carrier, Carlos Soares, and Ricardo Vilalta. *Metalearning: Applications to Data Mining*. Springer, 2009. ISBN 9783540732624.

[36] Edesio Alcobaca, Felipe Siqueira, Adriano Rivolli, LuÃŋs P. F. Garcia, Jefferson T. Oliva, and AndrÃľ C. P. L. F. de Carvalho. Mfe: Towards reproducible meta-feature extraction. *Journal of Machine Learning Research*, 21(111):1–5, 2020. URL http://jmlr.org/papers/v21/19-348.html.

[37] Yonghong Peng, Peter A Flach, Carlos Soares, and Pavel Brazdil. Improved Dataset Characterisation for Meta-learning. In *International Conference on Discovery Science*, pages 141–152, Berlin, Heidelberg, 2002. Springer.

[38] Adriano Rivolli, Carlos Soares, Joaquin Vanschoren, and F De Carvalho. Characterizing classification datasets: a study of meta-features for meta-learning. *arXiv preprint arXiv:1808.10406*, 2019.

[39] Ashvini Balte, Nitin Pise, and Kulkarni Parag. Meta-Learning With Landmarking: A Survey. In *International Journal of Computer Applications*, volume 105, pages 47–51, 2014.

[40] Bernhard Pfahringer and Christophe Giraud-carrier. Meta-Learning by Landmarking Various Learning Algorithms. (January):743–750, 2000.

[41] Hilan Bensusan and Christophe Giraud-carrier. Discovering Task Neighbourhoods through Landmark Learning Performances A Set of Landmarkers. pages 325–330, 2000.

[42] Carlos Soares, Johann Petrak, and Pavel Brazdil. Sampling-Based Relative Landmarks: Systematically Test-Driving Algorithms Before Choosing. pages 88–95, 2001. doi: 10.1007/3-540-45329-6.

[43] Marin Matijaš, Johan A K Suykens, and Slavko Krajcar. Load forecasting using a multivariate meta-learning system. *Expert Systems with Applications*, 40:4427–4437, 2013.

[44] Christiane Lemke and Gabrys Bogdan. Meta-learning for time series forecasting and forecast combination. *Neurocomputing*, 73(10-12):2006–2016, 2016. ISSN 0925-2312. doi: 10.1016/j.neucom.2009.09.020. URL http://dx.doi.org/10.1016/j.neucom.2009.09.020.

[45] Xiaozhe Wang, Kate Smith-miles, and Rob Hyndman. Rule induction for forecasting method selection: Meta-learning the characteristics of univariate time series. *Neurocomputing*, 72:2581–2594, 2009. doi: 10.1016/j.neucom.2008.10.017.

[46] R. Prudêncio and T. Ludermir. Using Machine Learning Techniques to Combine Forecasting Methods. pages 1122–1127, 2004.

[47] Usue Mori, Alexander Mendiburu, and Jose A. Lozano. Similarity Measure Selection for Clustering Time Series Databases. *IEEE Transactions on Knowledge and Data Engineering*, 28(1):181–195, 2016. ISSN 10414347. doi: 10.1109/TKDE.2015.2462369.

[48] Alejandro Barredo, Natalia Díaz-Rodríguez, Javier Del, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable Artificial Intelligence (XAI): Concepts , Taxonomies, Opportunities and Challenges Toward Responsible AI. *Information Fusion*, pages 82–115, 2020. ISSN 1566-2535. doi: 10.1016/j.inffus.2019.12.012. URL https://doi.org/10.1016/j.inffus.2019.12.012.

[49] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Computer Vision and Pattern Recognition*, 2016.

[50] Erico Tjoa and Cuntai Guan. A Survey on Explainable Artificial Intelligence (XAI): Toward Medical XAI. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–21, 2020. doi: 10.1109/TNNLS.2020.3027314.

[51] Marco Tulio Ribeiro and Carlos Guestrin. âĂIJWhy Should I Trust You?âĂİ Explaining the Predictions of Any Classifier. *The 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144, 2016.

[52] Sebastian Lapuschkin, Stephan Wäldchen, Alexander Binder, Grégoire Montavon, Wojciech Samek, and Klaus-robert Müller. Unmasking Clever Hans predictors and assessing. *Nature Communications*, pages 1–8, 2019. ISSN 2041-1723. doi: 10.1038/s41467-019-08987-4. URL http://dx.doi.org/10.1038/s41467-019-08987-4.

[53] B Y Mengnan Du, Ninghao Liu, and Xia Hu. Techniques for Interpretable Machine Learning. *Communications of the ACM*, 2019.

[54] Riccardo Guidotti, Anna Monreale, and Salvatore Ruggieri. A Survey of Methods for Explaining Black Box Models. *ACM computing surveys*, 51(5), 2018.

[55] Amina Adadi and Mohammed Berrada. Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access*, 6: 52138–52160, 2018. doi: 10.1109/ACCESS.2018.2870052.

[56] Thomas Rojat, David Filliat, Javier Del Ser, Rodolphe Gelin, and D Natalia. Explainable Artificial Intelligence (XAI) on Time Series Data: A Survey. *arXiv:2104.00950v1*, 2021.

[57] Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. *Proceedings of the IEEE international conference on computer vision*, 2017.

[58] Brett Poulin, Roman Eisner, Duane Szafron, Paul Lu, Russ Greiner, D S Wishart, Alona Fyshe, Brandon Pearcy, Cam Macdonell, and John Anvik. Visual Explanation of Evidence in Additive Classifiers. *Proceedings Of The National Conference On Artificial Intelligence*, pages 1822–1829, 2006.

[59] Marco Tulio Ribeiro, S Singh, and Carlos Guestrin. Anchors: High-Precision Model-Agnostic Explanations. *Proceedings of the AAAI conference on artificial intelligence*, 32(1), 2018.

[60] Andreas Henelius, Kai Puolamäki, Henrik Boström, and Lars Asker. A peek into the black box: exploring classifiers by randomization. *Data Mining and Knowledge Discovery*, pages 1503–1529, 2014. doi: 10.1007/s10618-014-0368-8.

[61] Pavel Senin. SAX-VSM: Interpretable Time Series Classification Using SAX and Vector Space Model. *IEEE International Conference on Data Mining (ICDM)*, pages 1175–1180, 2013. doi: 10.1109/ICDM.2013.52.

[62] Isak Karlsson, Jonathan Rebane, and Panagiotis Papapetrou. Explainable Time Series Tweaking via Irreversible and Reversible Temporal Transformations. *IEEE International Conference on Data Mining (ICDM)*, 2018. doi: 10.1109/ICDM.2018.00036.

[63] Thach Le, Nguyen Severin, Gsponer Iulia, Ilie Martin, and Georgiana Ifrim. Interpretable Time Series Classification using Linear Models and Multi-Resolution Multi-Domain Symbolic Representations. *Data Mining and Knowledge Discovery*, 33(4):1183–1222, 2019. ISSN 1573-756X. doi: 10.1007/s10618-019-00633-3. URL https://doi.org/10.1007/s10618-019-00633-3.

[64] Yichang Wang, Elisa Fromont, and Simon Malinowski. Learning Interpretable Shapelets for Time Series Classification through Adversarial Regularization. *arXiv preprint arXiv:1906.00917*, 2019.

[65] Hassan Ismail, Fawaz Germain, Lhassane Idoumghar Pierre-Alain Muller, and Jonathan Weber. Deep Learning for Time Series Classification: a Review. *Data Mining and Knowledge Discovery*, 33(4):917–963, 2019. ISSN 1573-756X. doi: 10.1007/s10618-019-00619-1. URL https://doi.org/10.1007/s10618-019-00619-1.

[66] Zhiguang Wang, Weizhong Yan, and Tim Oates. Time Series Classification from Scratch with Deep Neural Networks: a Strong Baseline. *International Joint Conference on Neural Networks (IJCNN)*, pages 1578–1585, 2017.

[67] Roy Assaf, Ioana Giurgiu, Frank Bagehorn, and Anika Schumann. MTEX-CNN: Multivariate Time series EXplanations for Predictions with Convolutional Neural Networks. *IEEE International Conference*

on Data Mining (ICDM), pages 952–957, 2019. doi: 10.1109/ICDM.2019.00106.

[68] Isak Karlsson, Panagiotis Papapetrou, and Henrik Boström. Generalized random shapelet forests. *Data Mining and Knowledge Discovery*, 30(5): 1053–1085, 2016. ISSN 1573-756X. doi: 10.1007/s10618-016-0473-y.

[69] Laurence Roze and Alexandre Termier. Agnostic Local Explanation for Time Series Classification. *International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 432–439, 2019. doi: 10.1109/ICTAI.2019.00067.

[70] Felix Mujkanovic, Vanja Doskoč, and Martin Schirneck. TimeXplain – a Framework for Explaining the Predictions of Time Series Classifiers. *arXiv preprint arXiv:2007.07606*, pages 1–17, 2020.

[71] J. Large, J. Lines, G. Oastler, M. Middlehurst, M. Flynn, A. Bostrom, P. Schäfer, C. Wei Tan, and Anthony Bagnall. UEA Time Series Classification Weka-compatible Java toolbox. 2017.

[72] Markus Löning, Anthony Bagnall, Sajaysurya Ganesh, Viktor Kazakov, Jason Lines, and Franz J Király. sktime: A Unified Interface for Machine Learning with Time Series, 2019.

[73] Anthony Bagnall, Aaron Bostrom, James Large, and Jason Lines. Simulated Data Experiments for Time Series Classification Part 1: Accuracy Comparison with Default Settings. *arXiv preprint arXiv:1703.09480*, pages 1–, 2017. URL https://arxiv.org/pdf/1703.09480v1.pdf{%}0Ahttp://arxiv.org/abs/1703.09480.

[74] Thu Trang Nguyen, Thach Le Nguyen, and Georgiana Ifrim. A Model-Agnostic Approach to Quantifying the Informativeness of Explanation Methods for Time Series Classification. *International Workshop on Advanced Analytics and Learning on Temporal Data (AALTD20)*, 2020.

[75] Ming Li, Xin Chen, Xin Li, Bin Ma, and Paul M.B. Vitányi. The similarity metric. *IEEE Transactions on Information Theory*, 50(12): 3250–3264, 2004. ISSN 00189448. doi: 10.1109/TIT.2004.838101.

[76] Xiaoyue Wang, Abdullah Mueen, Hui Ding, Goce Trajcevski, Peter Scheuermann, and Eamonn Keogh. Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, 26(2):275–309, 2013. ISSN 13845810. doi: 10.1007/s10618-012-0250-5.

[77] Yanping Chen, Bing Hu, Eamonn Keogh, and Gustavo E.A.P.A. Batista. DTW-D: Time Series Semi-Supervised Learning from a Single Example. *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 383, 2013. ISSN 9781450321747. doi: 10.1145/2487575.2487633. URL http://dl.acm.org/citation.cfm?doid=2487575.2487633.

[78] Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proceedings of the VLDB Very Large Database Endowment*, 1(2):

1542–1552, 2008. ISSN 2150-8097. doi: 10.1145/1454159.1454226. URL http://dl.acm.org/citation.cfm?id=1454159.1454226{%}5Cnfile:///Users/bwilcox6/Dropbox/Thesis/Mekentosj/Library.papers3/Files/08/08A2DC46-9492-4520-BB79-9AED63E1370D.pdf{%}5Cnpapers3://publication/uuid/C5531CD9-BAC2-42E8-89C9-413B86FB7C35.

[79] Jason Lines and Anthony Bagnall. Time series classification with ensembles of elastic distance measures. *Data Mining and Knowledge Discovery*, 29(3):565–592, 2015. ISSN 13845810. doi: 10.1007/s10618-014-0361-2.

[80] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison-Wesley, Reading, MA, addison-we edition, 2005.

[81] Arash Jalalian and Stephan K. Chalup. GDTW-P-SVMs: Variable-length time series analysis using support vector machines. *Neurocomputing*, 99:270–282, 2013. ISSN 09252312. doi: 10.1016/j.neucom.2012.07.006. URL http://dx.doi.org/10.1016/j.neucom.2012.07.006.

[82] Pierre-François Marteau and Sylvie Gibet. On Recursive Edit Distance Kernels With Application to Time Series Classification. *IEEE Transactions on Neural Networks and Learning Systems*, 26(6):1–15, 2014. ISSN 2162237X. doi: 10.1109/TNNLS.2014.2333876. URL https://arxiv.org/pdf/1005.5141.pdf.

[83] Brian Kenji Iwana, Volkmar Frinken, Kaspar Riesen, and Seiichi Uchida. Efficient temporal pattern recognition by means of dissimilarity space embedding with discriminative prototypes. *Pattern Recognition*, 64 (September 2016):268–276, 2017. ISSN 00313203. doi: 10.1016/j.patcog.2016.11.013. URL http://dx.doi.org/10.1016/j.patcog.2016.11.013.

[84] S Gudmundsson, T P Runarsson, and S Sigurdsson. Support vector machines and dynamic time warping for time series. In *Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 2772–2776, 2008. ISBN 2161-4393 VO -. doi: 10.1109/IJCNN.2008.4634188.

[85] Young-Seon Jeong and Raja Jayaraman. Support vector-based algorithms with weighted dynamic time warping kernel function for time series classification. *Knowledge-Based Systems*, 75(June):184–191, 2015. ISSN 0950-7051. doi: http://dx.doi.org/10.1016/j.knosys.2014.12.003. URL http://www.sciencedirect.com/science/article/pii/S095070511400433X.

[86] Dongyu Zhang, Wangmeng Zuo, David Zhang, and Hongzhi Zhang. Time series classification using support vector machine with Gaussian elastic metric kernel. *Proceedings - International Conference on Pattern Recognition*, pages 29–32, 2010. ISSN 10514651. doi: 10.1109/ICPR.2010.16.

[87] Arnaud Lods, Simon Malinowski, Romain Tavenard, and Laurent Amsaleg. Learning DTW-Preserving Shapelets. In *International Symposium on Intelligent Data Analysis*, pages 198–209. Springer, Cham, 2017.

[88] Jason Lines, Luke M. Davis, Jon Hills, and Anthony Bagnall. A shapelet transform for time series classification. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 289, 2012. ISBN 9781450314626. doi: 10.1145/2339530.2339579. URL http://dl.acm.org/citation.cfm?doid=2339530.2339579.

[89] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967. ISSN 0018-9448. doi: 10.1109/TIT.1967.1053964. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1053964.

[90] Hüseyin Kaya and Åđule Gündüz-ÖÊĞüdücü. A distance based time series classification framework. *Information Systems*, 51:27–42, 2015. ISSN 03064379. doi: 10.1016/j.is.2015.02.005.

[91] Elżbieta Pękalska and Robert P.W. Duin. *The Dissimilarity Representation for Pattern Recognition: Foundations and Applications*. 2005. ISBN 9812565302.

[92] Yihua Chen, Eric Garcia, and Maya Gupta. Similarity-based classification: Concepts and algorithms. *Journal of Machine Learning Research*, 10(206):747–776, 2009. ISSN 15324435. URL http://jmlr.csail.mit.edu/papers/volume10/chen09a/chen09a.pdf{%}5Cnhttp://dl.acm.org/citation.cfm?id=1577096.

[93] Elżbieta Pękalska, Pavel Paclík, and Robert P.W. Duin. A Generalized Kernel Approach to Dissimilarity-based Classification. *Journal of Machine Learning Research*, 2:175–211, 2001. ISSN 15324435. doi: 10.1162/15324430260185592.

[94] Thore Graepel, Ralf Herbrich, Peter Bollmann-Sdorra, and Klaus Obermayer. Classification on Pairwise Proximity Data. *Advances in Neural Information Processing Systems*, 11:438–444, 1999. ISSN 10495258. URL http://books.google.com/books?hl=en{&}lr={&}id=bMuzXPzlkG0C{&}oi=fnd{&}pg=PA438{&}dq=Classification+on+Pairwise+Proximity+Data{&}ots=MvnhyCBIPi{&}sig=VlrAY0hS9e7RXEmLBFReVbH4r74.

[95] Hiroaki Sakoe and Seibi Chiba. Dynamic Programming Algorithm Optimization for Spoken Word Recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 26(1):43–49, 1978. ISSN 00963518. doi: 10.1109/TASSP.1978.1163055.

[96] Rafael Giusti, Diego F. Silva, and Gustavo E.A.P.A. Batista. Improved time series classification with representation diversity and SVM. In *International Conference on Machine Learning and Applications*, number 1, pages 1–6, 2016. ISBN 9781509061662. doi: 10.1109/ICMLA.2016.108.

[97] Usue Mori, Alexander Mendiburu, Eamonn Keogh, and Jose A. Lozano. Reliable early classification of time series based on discriminating the classes over time. *Data Mining and Knowledge Discovery*, 31(1):233–263, 2017. ISSN 1573756X. doi: 10.1007/s10618-016-0462-1.

[98] Carl Rasmussen and Cristopher Williams. *Gaussian Processes for Machine Learning*. 2006. ISBN 026218253X.

[99] Lingfei Wu, Ian En-Hsu Yen, Jinfeng Yi, Fangli Xu, Qi Lei, and Michael Witbrock. Random Warping Series: A Random Features Method for Time-Series Embedding. *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, 84:793–802, 2018. URL http://proceedings.mlr.press/v84/wu18b.html.

[100] Ali Rahimi and Benjamin Recht. Random Features for Large-Scale Kernel Machines. *Neural Information Processing Systems NIPS*, (1): 1–10, 2007.

[101] Lingfei Wu, Ian En-Hsu Yen, Fangli Xu, Pradeep Ravikuma, and Michael Witbrock. D2KE: From Distance to Kernel and Embedding. *arxiv.org/abs/1802.04956v3*, pages 1–18, 2018. URL http://arxiv.org/abs/1802.04956.

[102] Thapanan Janyalikit, Phongsakorn Sathianwiriyakhun, Haemwaan Sivaraks, and Chotirat Ann Ratanamahatana. An Enhanced Support Vector Machine for Faster Time Series Classification. In *Asian Conference on Intelligent Information and Database Systems*, pages 616–625, 2016. ISBN 9783662493809.

[103] Eamonn Keogh and Chotirat Ann Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and information systems*, (February 2003):358–386, 2005. doi: 10.1007/s10115-004-0154-9.

[104] Brijnesh Jain and Stephan Spiegel. Dimension Reduction in Dissimilarity Spaces for Time Series Classification. In *International Workshop on Advanced Analytics and Learning on Temporal Data*, pages 31–46, 2015. ISBN 9783319444116. URL https://link.springer.com/content/pdf/10.1007/978-3-319-44412-3.pdf.

[105] Yoav Freund and Robert E Schapire. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Computer and System Sciences*, 139:119–139, 1997.

[106] Qing He, Dong Zhi, Fuzhen Zhuang, Tianfeng Shang, and Zhongzhi Shi. Fast time series classification based on infrequent shapelets. *Proceedings of the 11th ICMLA International Conference on Machine Learning and Applications*, 1:215–219, 2012. doi: 10.1109/ICMLA.2012.44.

[107] Abdullah Mueen, Eamonn Keogh, and Neal Young. Logical-shapelets: an expressive primitive for time series classification. *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1154–1162, 2011. doi: 10.1145/2020408.2020587. URL http://dl.acm.org/citation.cfm?doid=2020408.2020587.

[108] Lexiang Ye and Eamonn Keogh. Time series shapelets: A novel technique that allows accurate, interpretable and fast classification. *Data Mining and Knowledge Discovery*, 22(1-2):149–182, 2011. ISSN 13845810. doi: 10.1007/s10618-010-0179-5.

[109] Josif Grabocka, Nicolas Schilling, Martin Wistuba, and Lars Schmidt-Thieme. Learning time-series shapelets. In *Proceedings of the 20th*

ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 392–401, 2014. ISBN 9781450329569. doi: 10.1145/2623330.2623613. URL http://dl.acm.org/citation.cfm?doid=2623330.2623613.

[110] Aaron Bostrom and Anthony Bagnall. Binary Shapelet Transform for Multiclass Time Series Classification. *Transactions on Large Scale Data and Knowledge Centered Systems*, 8800:24–46, 2014. ISSN 16113349. doi: 10.1007/978-3-662-45714-6.

[111] Xing Wang, Jessica Lin, Pavel Senin, Los Alamos, Tim Oates, Sunil Gandhi, Arnold P Boedihardjo, Crystal Chen, and Susan Frankenstein. RPM: Representative Pattern Mining for Efficient Time Series Classification. *Proceedings of the 19th International Conference on Extending Database Technology*, pages 185–196, 2016. ISSN 23672005. doi: 10.5441/002/edbt.2016.19.

[112] Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. Experiencing SAX: A novel symbolic representation of time series. *Data Mining and Knowledge Discovery*, 15(2):107–144, 2007. ISSN 13845810. doi: 10.1007/s10618-007-0064-z.

[113] Pavel Senin, Jessica Lin, Xing Wang, Tim Oates, Sunil Gandhi, Arnold P Boedihardjo, Crystal Chen, Susan Frankenstein, and Manfred Lerner. GrammarViz 2 . 0: A Tool for Grammar-Based Pattern Discovery in Time Series. (1218325):468–472, 2014.

[114] Aaron Bostrom, Anthony Bagnall, and Jason Lines. Evaluating Improvements to the Shapelet Transform. In *www-bcf.usc.edu*, 2016. URL http://www-bcf.usc.edu/{~}liu32/milets16/paper/MiLeTS{_}2016{_}paper{_}8.pdf.

[115] Xiaosheng Li and Jessica Lin. Evolving Separating References for Time Series Classification. pages 243–251, 2018. URL https://epubs.siam.org/doi/pdf/10.1137/1.9781611975321.28.

[116] Ingwer Borg and Patrick Groenen. Modern Multidimensional Scaling: Theory and Applications. *Springer-Verlag*, 1997.

[117] Richard C. Wilson, Edwin R. Hancock, Elżbieta Pękalska, and Robert P.W. Duin. Spherical and hyperbolic embeddings of data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11): 2255–2269, 2014. ISSN 01628828. doi: 10.1109/TPAMI.2014.2316836.

[118] David W Jacobs, Daphna Weinshall, and Yoram Gdalyahu. Classification with Nonmetric Distances: Image Retrieval and Class Representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6):583–600, 2000.

[119] Akira Hayashi, Yuko Mizuhara, and Nobuo Suematsu. Embedding time series data for classification. *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 356—-365, 2005.

[120] Yuko Mizuhara, Akira Hayashi, and Nobuo Suematsu. Embedding of time series data by using Dynamic Time Warping distances. *Systems*

and Computers in Japan, 37(3):1–9, 2006. ISSN 08821666. doi: 10.1002/scj.20486.

[121] Mikhail Belkin and Partha Niyogi. Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering. *Advances in Neural Information Processing Systems*, 14:585–591, 2002. ISSN 10495258. doi: 10.1.1.19.9400. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.19.9400{&}rep=rep1{&}type=pdf.

[122] M. Lichman. UCI Machine Learning Repository, 2013. URL http://archive.ics.uci.edu/ml.

[123] Qi Lei, Jinfeng Yi, Roman Vaculin, Lingfei Wu, and Inderjit S. Dhillon. Similarity Preserving Representation Learning for Time Series Analysis. *arXiv:1702.03584 [cs]*, 2017. URL http://arxiv.org/abs/1702.03584{%}5Cnhttp://www.arxiv.org/pdf/1702.03584.pdf{%}5Cnhttps://arxiv.org/abs/1702.03584.

[124] Colins C. Adams. *The Knot Book: An Elementary Introduction to the Mathematical Theory of Knots*. 2004.

[125] Ruoyu Sun and Zhi Quan Luo. Guaranteed Matrix Completion via Non-Convex Factorization. *IEEE Transactions on Information Theory*, 62 (11):6535–6579, 2016. ISSN 00189448. doi: 10.1109/TIT.2016.2598574.

[126] Corinna Cortes and Vladimir Vapnik. Support-Vector Networks. *Machine Learning*, 297:273–297, 1995.

[127] John Shawe-Taylor and Nello Cristianini. *Kernel methods for pattern analysis*. 2004. ISBN 9780521813976.

[128] Vladimir Vapnik. *Statistical Learning Theory*, volume 2. New York, 1998. ISBN 0-471-03003-1. doi: 10.2307/1271368. URL http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0471030031.html.

[129] Bernhard Schölkopf. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. 2001. ISBN 0262194759. doi: 10.1198/jasa.2003.s269. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.167.5140{&}rep=rep1{&}type=pdf.

[130] Christina Leslie, Eleazar Eskin, and William Stafford Noble. the Spectrum Kernel: a String Kernel for Svm Protein Classification. In *Proceedings of the Pacific Symposium on Biocomputing*, pages 564–575, 2002. URL http://www.ics.uci.edu/{~}welling/teatimetalks/kernelclub04/spectrum.pdf.

[131] Stefan Rüping. SVM Kernels for Time Series Analysis. Technical report, 2001.

[132] Cheng Soon Ong, Xavier Mary, Stéphane Canu, and Alexander J. Smola. Learning with non-positive kernels. *Proceedings of the 21th ICML International Conference on Machine Learning*, (7):81, 2004. ISSN 00200255. doi: 10.1145/1015330.1015443. URL http://eprints.pascal-network.org/archive/00000714/{%}5Cnhttp://portal.acm.org/citation.cfm?doid=1015330.1015443.

[133] Claus Bahlmann, Bernard Haasdonk, and Hans Burkhardt. Online handwriting recognition with support vector machines - A kernel ap-

proach. *Proceedings - International Workshop on Frontiers in Handwriting Recognition, IWFHR*, pages 49–54, 2002. ISSN 15505235. doi: 10.1109/IWFHR.2002.1030883.

[134] Dennis Decoste and Bernhard Schölkopf. Training Invariant Support Vector Machines using Selective Sampling. *Machine Learning*, 46,:161–190, 2002.

[135] Hiroshi Shimodaira, Ken Ichi Noma, Mitsuru Nakai, and Shigeki Sagayama. Dynamic Time-Alignment Kernel in Support Vector Machine. *Advances in Neural Information Processing Systems*, 2 (1):921–928, 2002. ISSN 1049-5258. doi: citeseer.ist.psu.edu/shimodaira01dynamic.html. URL http://hdl.handle.net/1842/1161.

[136] Bernard Haasdonk. Feature space interpretation of SVMs with indefinite kernels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4):482–492, 2005. ISSN 01628828. doi: 10.1109/TPAMI.2005.78.

[137] Pai-hsuen Chen, Rong-en Fan, and Chih-jen Lin. A Study on SMO-Type Decomposition Methods for Support Vector Machines. *IEEE Transactions on Neural Networks and Learning Systems*, 17(4):893–908, 2006.

[138] Bernard Haasdonk and Claus Bahlmann. Learning with Distance Substitution Kernels. *Pattern Recognition*, pages 220–227, 2004. ISSN 03029743. doi: 10.1007/978-3-540-28649-3_27.

[139] Helmuth Pree, Benjamin Herwig, Thiemo Gruber, Bernhard Sick, Klaus David, and Paul Lukowicz. On general purpose time series similarity measures and their use as kernel functions in support vector machines. *Information Sciences*, 281:478–495, 2014. ISSN 00200255. doi: 10.1016/j.ins.2014.05.025. URL http://dx.doi.org/10.1016/j.ins.2014.05.025.

[140] Young-seon Jeong, Myong K Jeong, and Olufemi A Omitaomu. Weighted dynamic time warping for time series classification. *Pattern Recognition*, 44(9):2231–2240, 2011. ISSN 0031-3203. doi: 10.1016/j.patcog.2010.09.022. URL http://dx.doi.org/10.1016/j.patcog.2010.09.022.

[141] Zhihua Chen, Wangmeng Zuo, Qinghua Hu, and Liang Lin. Kernel sparse representation for time series classification. *Information Sciences*, 292:15–26, 2015. ISSN 00200255. doi: 10.1016/j.ins.2014.08.066. URL http://dx.doi.org/10.1016/j.ins.2014.08.066.

[142] Hansheng Lei and Bingyu Sun. A Study on the Dynamic Time Warping in Kernel Machines. *Proceedings of the 3rd SITIS International IEEE Conference on Signal-Image Technologies and Internet-Based System*, pages 839–845, 2007. doi: 10.1109/SITIS.2007.112.

[143] Hüseyin Kaya and Åđule Gündüz-ÖÊĞüdücü. SAGA: A novel signal alignment method based on genetic algorithm. *Information Sciences*, 228:113–130, 2013. ISSN 00200255. doi: 10.1016/j.ins.2012.12.012.

[144] Marco Cuturi. Fast Global Alignment Kernels. In *Proceedings of the 28th ICML International Conference on Machine Learning*, pages 929–

936, 2011. ISBN 9781450306195. URL http://www.iip.ist.i.kyoto-u.ac.jp/member/cuturi/Papers/cuturi11fast.pdf.

[145] Isabelle Guyon, Lambert Schomaker, Rkjean Planiondon, Mark Liberman, Stan Janet, Ecole Polytechnique De Montreal, and Linguistic Data Consortium. UNIPEN project of on-line data exchange. pages 29–33, 1994.

[146] Sepp Hochreiter and Klaus Obermayer. Support Vector Machines for Dyadic Data. *Neural Computation*, 1510:1472–1510, 2006.

[147] Gang Wu, Edward Y. Chang, and Zhihua Zhang. An analysis of transformation on non-positive semidefinite similarity matrix for kernel machines. *Proceedings of the 22th ICML International Conference on Machine Learning*, 8, 2005. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.133.4077{&}rep=rep1{&}type=pdf{%}5Cnpapers2://publication/uuid/0C628F77-949C-406A-BDE8-9FBEB8FD4424.

[148] Gang Wu, Edward Y. Chang, and Zhihua Zhang. Learning with non-metric proximity matrices. *Proceedings of the 13th ACM International Conference on Multimedia*, page 411, 2005. doi: 10.1145/1101149.1101239. URL http://portal.acm.org/citation.cfm?doid=1101149.1101239.

[149] Li Zhang, Pei-chann Chang, Jing Liu, Zhe Yan, Ting Wang, and Fanzhang Li. Kernel Sparse Representation-Based Classifier. *IEEE Transactions on Signal Processing*, 60(4):1684–1695, 2012.

[150] Lei Chen and Raymond Ng. On The Marriage of Lp-norms and Edit Distance. In *International conference on Very large data bases*, pages 792–803, 2004.

[151] Jason Weston, Bernhard Schölkopf, Eleazar Eskin, Christina Leslie, and William Stafford Noble. Dealing with large diagonals in kernel matrices. In *Annals of the Institute of Statistical Mathematics*, volume 55, pages 391–408, 2003. doi: 10.1023/A:1026338322729.

[152] F Casacuberta, E Vidal, and H Rulot. On the metric properties of dynamic time warping. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35(11):1631–1633, 1987. ISSN 0096-3518. doi: 10.1109/TASSP.1987.1165065.

[153] David M.J. Tax and Robert P.W. Duin. Support Vector Data Description. *Machine Learning*, 54,:45–66, 2004.

[154] Adrien Gaidon, Zaid Harchoui, and Cordelia Schmid. A time series kernel for action recognition. In *Procedings of the British Machine Vision Conference*, pages 63.1–63.11, 2011. ISBN 1-901725-43-X. doi: 10.5244/C.25.63. URL https://hal.inria.fr/inria-00613089/.

[155] A. Troncoso, M. Arias, and J. C. Riquelme. A multi-scale smoothing kernel for measuring time-series similarity. *Neurocomputing*, 167:8–17, 2015. ISSN 18728286. doi: 10.1016/j.neucom.2014.08.099. URL http://dx.doi.org/10.1016/j.neucom.2014.08.099.

[156] Karthik Kumara, Rahul Agrawal, and Chiranjib Bhattacharyya. A large margin approach for writer independent online handwriting classification. *Pattern Recognition Letters*, 29(7):933–937, 2008. ISSN 01678655. doi: 10.1016/j.patrec.2008.01.016.

[157] K R Sivaramakrishnan and Chiranjib Bhattacharyya. Time Series Classification for Online Tamil Handwritten Character Recognition âĂŞ A Kernel Based Approach. In *International Conference on Neural Information Processing*, pages 800–805, 2004.

[158] Zhengdong Lu, K. Todd Leen, Yonghong Huang, and Deniz Erdogmus. A Reproducing Kernel Hilbert Space Framework for Pairwise Time Series Distances. In *Proceedings of the 25th ICML International Conference on Machine learning*, volume 56, pages 624–631, 2008. URL https://hal.inria.fr/inria-00613089/.

[159] Yangtao Xue, Li Zhang, Zhiwei Tao, Bangjun Wang, and Fan-zhang Li. An Altered Kernel Transformation for Time Series Classification. In *International Conference on Neural Information Processing*, pages 455–465, 2017. ISBN 9783319701394.

[160] Gabriel Wachman, Roni Khardon, Pavlos Protopapas, and Charles R. Alcock. Kernels for Periodic Time Series Arising in Astronomy. In *European Conference on Machine Learning and Knowledge Discovery in Databases*, 2009. ISBN 9783642041730.

[161] Pierre-François Marteau and Sylvie Gibet. Constructing Positive Definite Elastic Kernels with Application to Time Series Classification. *CoRR*, pages 1–18, 2010. doi: 10.1109/TNNLS.2014.2333876. URL http://arxiv.org/abs/1005.5141{%}5Cnhttp://dx.doi.org/10.1109/TNNLS.2014.2333876.

[162] Pierre-François Marteau, Nicolas Bonnel, and Gilbas Ménier. Discrete Elastic Inner Vector Spaces with Application in Time Series and Sequence Mining. *IEEE Transactions on Knowledge and Data Engineering*, 25(9):2024–2035, 2012. doi: 10.1109/TKDE.2012.131. URL http://arxiv.org/abs/1206.6196{%}0Ahttp://dx.doi.org/10.1109/TKDE.2012.131.

[163] W. H. Greub. *Linear algebra.* Springer-Verlag, 1975.

[164] Corinna Cortes, Patrick Haffner, and Mehryar Mohri. Rational Kernels: Theory and Algorithms. *Journal of Machine Learning Research*, 5:1035–1062, 2004.

[165] Mark Liberman. TI46 speech corpus. In *Linguistic Data Consortium*, 1993.

[166] Joan Serrà and Josep Ll Arcos. An empirical evaluation of similarity measures for time series classification. *Knowledge-Based Systems*, 67:305–314, 2014. ISSN 09507051. doi: 10.1016/j.knosys.2014.04.035. URL http://dx.doi.org/10.1016/j.knosys.2014.04.035.

[167] Elżbieta Pękalska, Robert P.W. Duin, and Pavel Paclík. Prototype selection for dissimilarity-based classifiers. *Pattern Recognition*, 39(2):189–208, 2006. ISSN 00313203. doi: 10.1016/j.patcog.2005.06.012.

[168] Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo E.A.P.A. Batista. The UCR Time Series Classification Archive. 2015. URL www.cs.ucr.edu/{~}eamonn/time{_}series{_}data/.

[169] Jorge Kanda, Andre De Carvalho, Eduardo Hruschka, Carlos Soares, and Pavel Brazdil. Meta-learning to select the best meta-heuristic for the Traveling Salesman Problem: A comparison of meta-features. *Neurocomputing*, 205:393–406, 2016. ISSN 0925-2312. doi: 10.1016/j.neucom.2016.04.027. URL http://dx.doi.org/10.1016/j.neucom.2016.04.027.

[170] Teresa Bernarda Ludermir. Meta-learning approaches to selecting time series models. (October), 2004. doi: 10.1016/j.neucom.2004.03.008.

[171] Anthony Bagnall, Jason Lines, W. Vickers, and Eamonn Keogh. The UEA and UCR time series classification repository. URL http://www.timeseriesclassification.com.

[172] J. R. Quinlan. C4.5: Programs for Machine Learning. *Morgan Kaufmann Publishers*, 1993.

[173] George H. John and Pat Langley. Estimating continuous distributions in bayesian classifiers. In *Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 338–345, San Mateo, 1995. Morgan Kaufmann.

[174] I BenGal. Bayesian networks. In *Bayesian networks.* Encyclopedia of Statistics in Quality and Reliability, 2008.

[175] J. J. Rodriguez, L. I. Kuncheva, and C. J. Alonso. Rotation forest: A new classifier ensemble method. *IEEE transactions on pattern analysis and machine intelligence*, pages 1619–1630, 2006.

[176] L. Breiman. Random forests. *Machine learning*, pages 5–32, 2001.

[177] H. Taud and J. F. Mas. Multilayer perceptron (mlp). *Geomatic Approaches for Modeling Land Change Scenarios*, pages 451–455, 2018.

[178] G. E. Batista, E. J. Keogh, O. M. Tataw, and V. M. De Souza. Cid: an efficient complexity-invariant distance for time series. *Data Mining and Knowledge Discovery*, pages 634–669, 2014.

[179] Tomasz Górecki and Maciej Łuczak. Using Derivatives in Time Series Classification. *Data Mining and Knowledge Discovery*, pages 310–331, 2013. doi: 10.1007/s10618-012-0251-4.

[180] Tomasz Górecki. Using derivatives in a longest common subsequence dissimilarity measure for time series classificatio. *Pattern Recognition Letters*, 45:99–105, 2014. doi: 10.1016/j.patrec.2014.03.009.

[181] Z. Wang, W. Yan, and T. Oates. Time series classification from scratch with deep neural networks: A strong baseline. *In 2017 International joint conference on neural networks (IJCNN)*, pages 1578–1585, 2017.

[182] B. Fawaz, H. I.and Lucas, G. Forestier, C. Pelletier, D. F. Schmidt, J. Weber, and F. Petitjean. Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery*, pages 1936–1962, 2020.

[183] J. Lines and A. Bagnall. Time series classification with ensembles of elastic distance measures. *Data Mining and Knowledge Discovery*, pages 29(3), 565–592, 2015.

[184] M. G. Baydogan, G. Runger, and E. Tuv. A bag-of-features framework to classify time series. *IEEE transactions on pattern analysis and machine intelligence*, pages 35(11), 2796–2802, 2013.

[185] M. G. Baydogan and G. Runger. Time series representation and similarity based on local autopatterns. *Data Mining and Knowledge Discovery*, pages 30(2), 476–509, 2016.

[186] S. le Cessie and J.C. van Houwelingen. Ridge estimators in logistic regression. *Applied Statistics*, 41(1):191–201, 1992.

[187] P. Senin and S. Malinchik. SAX-VSM: Interpretable Time Series Classification Using SAX and Vector Space Model. pages 1175–1180, 2013. doi: 10.1109/ICDM.2013.52.

[188] Rui Leite and Pavel Brazdil. Improving Progressive Sampling Via Meta-Learning on Learning Curves. pages 250–261, 2004.

[189] L. Todorovski and S. Džeroski. Experiments in Meta-level Learning with ILP. pages 98–106, 1999.

[190] Min-ling Zhang and Zhi-hua Zhou. M L-KNN: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40:2038–2048, 2007. doi: 10.1016/j.patcog.2006.12.019.

[191] A. Kalousis. *Algorithm Selection via Meta-Learning*. PhD thesis, University of Geneva, 2002.

[192] Hilan Bensusan and Alexandros Kalousis. Estimating the Predictive Accuracy of a Classifier. pages 25–36, 2001.

[193] Ronald Fagin, Ravi Kumar, and D Sivakumar. Comparing top k lists. *SIAM Journal on discrete mathematics*, 17(1):134–160, 2003.

[194] M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society, Series B (Statistical Methodology)*, pages 36(2):111–147, 1974.

[195] Chang Wei Tan, Eamonn Keogh, and Geoffrey I Webb. Time Series Classification for Varying Length Series. *arXiv:1910.04341v1*, 2019.

[196] Arthur Le Guennec, Simon Malinowski, and Romain Tavenard. Data Augmentation for Time Series Classification using Convolutional Neural Networks. *International Workshop on Advanced Analytics and Learning on Temporal Data (AALTD16)*, 2016.

[197] Luc Devroye, Peter Epstein, and Jörg-Rüdiger Sack. On Generating Random Intervals and Hyperrectangles. *Journal of Computational and Graphical Statistics*, 2(3):291–307, 1993.

[198] Felix Hausdorff. *Set Theory*. Republished by Americal Mathematical Society (AMS) – Chelsea 2005, 2nd edition, 1962.

[199] Michael Fielding Barnsley. Superfractals. *Australian National University, Canberra*, 2006.

[200] Romain Briandet, E Katherine Kemsley, and Reginald H Wilson. Discrimination of Arabica and Robusta in Instant Coffee by Fourier Trans-