الجمهورية الجزائرية الديمقراطية الشعبية
People's Democratic Republic of Algeria
وزارة التعليم العالي والبحث العلمي
Ministry of Higher Education and Scientific Research

University of Skikda - 20 Août 1955
Faculty of Sciences
Department of Computer Science

جامعة 20 أوت 1955 سكيكدة
كلية العلوم
قسم الإعلام الآلي

THESIS

Submitted in fulfillment of the requirements for the degree of

3rd cycle doctorate (LMD system)

Major: Computer Science

Minor: Computation and Cognition of Informatics Systems

Advanced Methods for Establishing Similarity between Time Series: Study and Applications

---

Publicly defended by:

Abdelmadjid Lahreche

On July 13, 2022

Examination Committee:

Smaine Mazouzi	Professor	University of Skikda - 20 Août 1955	President
Mehdi Boulaiche	MCA	University of Skikda - 20 Août 1955	Examiner
Said Brahimi	MCA	University of Guelma - 05 Mai 1945	Examiner
Bachir Boucheham	Professor	University of Skikda - 20 Août 1955	Supervisor

Acknowledgements

I would like to express my deepest gratitude to my Ph.D supervisor, Pr. Bachir Boucheham, for his patience, suggestions and guidance during the preparation of this thesis.

A special note of thanks is also extended to the Examination Committee member: Pr. Smaine Mazouzi, University of Skikda, Dr. Said Brahimi, University of Guelma, and Dr. Mehdi Boulaiche, University of Skikda, for their time dedicated to evaluating, commenting and making suggestions regarding the content of this thesis manuscript.

Dedication

To my family...

Contents

List of Figures

List of Tables

List of Abbreviations

1-NN	1-Nearest Neighbor
ANN	Artificial Neural Network
APCA	Adaptive Piecewise Constant Approximation
ARM	Auto-Regressive Model
ASEAL	Approximate Shape Exchange Algorithm
BOP	Bag of Patterns
BOSS	Bag-of-SFA-Symbols
CDTW	Constrained DTW
CID	Complexity Invariant Distance
CNN	Convolutional Neural Network
COTE	Collective of Transformation-Based Ensembles
DD-DTW	Derivative Distance DTW
DDTW	Derivative DTW
DFT	Discrete Fourier Transform
DL	Deep Learning
DT	Decision Tree
DTW	Dynamic Time Warping
DWT	Discrete Wavelet Transform
ECG	Electrocardiogram
ED	Euclidean Distance
EDR	Edit Distance on Real sequence
EE	Ensembles of Elastic distance measures

ERP	Edit Distance with Real Penalty
FANSEA	FAN-Shape Exchange Algorithm
HIVE-COTE	Hierarchical Vote of Transformation-Based Ensembles
HMM	Hidden Markov Model
***k*-NN**	k-Nearest Neighbors
LCSS	Longest Common Subsequence
LDTW	Limited warping path length DTW
LE-DTW	Local Extrema DTW
LMDS-SEA	Local Matching and Distance Selection SEA
LSDTW	Locally Slope DTW
LWDTW	Locally Weighted DTW
MIT-BIH	Massachusetts Institute of Technology-Beth Israel Hospital
MSM	Move-Split-Merge
NBM	Naïve Bays Model
PAA	Piecewise Aggregate Approximation
PLA	Piecewise Linear Approximation
QPDTW	Quasi-Periodic DTW
QPTS	Quasi-Periodic Time Series
RNN	Recurrent Neural Network
SAX	Symbolic Aggregate Approximation
SAX-VSM	Symbolic Aggregate Approximation-Vector Space Model
SC-DTW	Shape Context DTW
SE	Shapelet Ensemble
SEA	Shape Exchange Algorithm
ST	Shapelet Transform
SVD	Singular Value Decomposition
SVM	Support Vector Machine
TSBF	Time Series Bag of Features
TSC	Time Series Classification

TS-CHIEF	Time Series Combination of Heterogeneous and Integrated Embedding Forest
TSF	Time Series Forest
TWED	Time Warp Edit Distance
UCR	University of California-Riverside
WDTW	Weighted DTW

ملخص

السلاسل الزمنية (TS) هي سلاسل من البيانات ذات القيم الحقيقية التي يتم تسجيلها بانتظام بمرور الوقت. يمكن العثور عليها في معظم تطبيقات العالم الحقيقي تقريباًا مثل الطب، المالية، الاقتصاد، الصناعة والعديد من التطبيقات الأخرى. في هذه الأطروحة نتناول إشكالية قياس التشابه بين السلاسل الزمنية. يعتبر مقياس تشابه السلاسل الزمنية مكوناًا رئيسياًا للعديد من مهام التنقيب في السلاسل الزمنية بما في ذلك على سبيل المثال لا الحصر التصنيف، البحث عن طريق المحتوى، التكتل واكتشاف الحالات الشاذة في بيانات السلاسل الزمنية.

نظراً لأهميتها الهائلة، حظيت مهمة قياس التشابه بين السلاسل الزمنية باهتمام كبير وأصبحت مجال بحث نشط في ميدان التنقيب في السلاسل الزمنية. بشكل عام، تتمثل إشكالية قياس تشابه السلاسل الزمنية في تحديد طريقة يمكنها مقارنة\محاذاة سلسلتين زمنيتين معينتين، من خلال تحديد مقدار الاتفاق-الخلاف بين هذه TS. ومع ذلك، لطالما كان قياس التشابه بين السلاسل الزمنية إشكالية صعبة. تكمن صعوبة هذه الإشكالية في حقيقة أن السلاسل الزمنية هي بطبيعتها ذات أبعاد عالية وتتوفر بكميات كبيرة من البيانات. علاوة على ذلك، فإن التشوهات المختلفة، التي غالباًا ما تصادف في السلاسل الزمنية، تزيد من تعقيد قياس تشابه TS (على سبيل المثال، الضوضاء، القيم المتطرفة، وتحول-ازاحة الوقت-السعة). لذلك، يجب تحديد مقياس التشابه بعناية من أجل التقاط التشابه بين سلسلتين زمنيتين بشكل صحيح.

من ناحية أخرى، يعتبر تصنيف السلاسل الزمنية (TSC) مهمة متزايدة الأهمية في مجال تعدين السلاسل الزمنية. TSC هي باختصار المهمة المخصصة للتنبؤ بفئة سلسلة زمنية للاستعلام من مجموعة بيانات تدريب مفهرسة. في العقدين الماضيين، كان هناك اهتمام

متزايد بـتطبيقات TSC، وبالتالي، تم نشر العديد من الأساليب في الأدبيات. من المثير للدهشة أن الدراسات المكثفة في هذا المجال من البحث أفادت أن النهج القائم على المسافة حيث يكون المصنف (1-NN) مقترناً بمقياس المسافة-التشابه المناسب أكثر دقة من معظم الأساليب الحالية. لكل الأدلة، في هذا السياق، يعتبر مقياس التشابه مكوناً مهماً ويلعب دوراً حيوياً في دقة المصنف 1-NN. وبالتالي، ركزت الغالبية العظمى من أبحاث TSC على تطوير مقاييس المسافة\التشابه جديدة.

الهدف الرئيسي من هذه الأطروحة يتعلق بمسألة قياس التشابه بين السلاسل الزمنية مع التركيز بشكل خاص على التصنيف كمجال تطبيق محفز للغاية. لذلك، فإننا نهدف إلى دراسة وتطوير تدابير التشابه في سياق TSC التي تكون قادرة على المنافسة فيما يتعلق بالطرق الحالية في الأدبيات (قوية، سريعة، وفعالة). ولهذه الغاية، تم اقتراح مساهمات مختلفة أثناء تطوير هذه الاطروحة.

في المساهمة الأولى لهذه الأطروحة، أجرينا مقارنة تجريبية مكثفة بين مقياس التشابه الشهير (DTW) ومتغيراته الأكثر شيوعاً في إطار مجال TSC. في هذه الدراسة، قمنا بتقييم الخوارزميات تجريبياً من حيث دقة التصنيف باستخدام 85 مجموعة بيانات من أرشيف جامعة كاليفورنيا ريفر ساید (UCR). تظهر النتائج التجريبية أن جميع الخوارزميات تقريباً متكافئة إحصائياً. في المساهمة الثانية، تناولنا مشكلة تصنيف السلاسل الزمنية الطويلة من خلال اقتراح مقياس تشابه جديد يسمى (LE-DTW). يقوم LE-DTW أولاً بتحويل السلاسل الزمنية الأصلية إلى فضاء ذو أبعاد منخفضة عن طريق استخراج الميزات القصوى المحلية. بعد ذلك، يقارن السلاسل الزمنية المحولة التي تم الحصول عليها باستخدام نسخة معدلة من DTW. لتقييم أداء LE-DTW، أجرينا تجارب مكثفة على مجموعة كبيرة ومتنوعة من مجموعات البيانات من أرشيف UCR. أظهرت النتائج كفاءة وفعالية LE-DTW المقترحة مقارنة بأحدث الأساليب، خاصة في السلاسل الزمنية الطويلة.

في المساهمة الثالثة، تم اقتراح نسخة أكثر فعالية من خوارزمية SEA لمحاذاة السلاسل الزمنية شبه الدورية. تعتمد FastSEA على خوارزمية فرز أكثر فعالية، لكنها بسيطة وهي (Counting Sort Algorithm). الغرض من FastSEA هو تسريع عملية محاذاة طريقة SEA دون التأثير على جودتها. المساهمة الأخيرة مكرسة لتوسيع جودة SEA إلى TSC العام. مع هذا الهدف، نقترح مقياس (LMDS-SEA)، والذي يعتمد بشكل أساسي على نموذج اختيار

المسافة الجديد. أظهرت النتائج التجريبية أن طريقة LMDS-SEA المقترحة تؤدي أداءً أفضل من SEA وتنافس DTW من حيث دقة التصنيف.

الكلمات المفتاحية: السلاسل الزمنية، مقاييس التشابه، قياس المسافات، تمثيل السلاسل الزمنية، تنقيب في السلاسل الزمنية، التصنيف، DTW, SEA, اختيار المسافة.

Abstract

Time series (TS) are sequences of real-valued data recorded regularly over time. They can be found in virtually all real-world applications, such as medicine, finance, economics, industry, and many others. In this thesis, we address the problematic of establishing the similarity between time series. Time series similarity measure is a key component of several time series mining tasks, including, but not limited to, classification, similarity search, clustering, and detection of anomalies in time series data.

Due to its tremendous importance, the time series similarity measure task has received considerable attention and has become an active research area in time series mining. Generally speaking, the problem of establishing similarity of time series consists in defining a method that can compare/align two given time series by determining the amount of agreement/discord between these TS. However, measuring the similarity between time series has always been a challenging problem. The difficulty of this problem lies in the fact that time series are inherently high dimensional and bear great amounts of data. Moreover, various distortions often encountered in time series further complicate the formalization of TS similarity measures (e.g., noise, outliers, and time/amplitude scaling/shift). Therefore, the similarity measure needs to be carefully established in order to properly capture the true similarity between two time series.

On the other hand, time series classification (TSC) is an increasingly important task in time series mining. TSC is briefly the task dedicated to predicting the class label of a query time series from a labeled training dataset. In the last two decades, there has been a growing interest in TSC and hence, many approaches have been published in the literature. Surprisingly, extensive

studies in this area of research have reported that the distance-based approach where the 1-Nearest Neighbor (1-NN) classifier combined with an appropriate distance/similarity measure is more accurate than most existing approaches. To all evidence, in this context, the similarity measure is a crucial ingredient and plays a vital role in the accuracy of the 1-NN classifier. Consequently, the vast majority of TSC research has focused on developing new distance/similarity measures.

The main objective of this thesis concerns the issue of the similarity measure between time series with a particular focus on the classification as a very motivating field of application. We, therefore, aim to study and develop similarity measures within the context of TSC that are competitive (i.e., robust, efficient, and effective) with respect to existing methods in the literature. To that end, various contributions have been proposed during the development of this work.

In the first contribution of this thesis, we carried out a large experimental comparison between the famous TS similarity measure "Dynamic Time Warping" (DTW) and its most popular variants for TSC. In this study, we empirically evaluated the methods in terms of classification accuracy using 85 datasets from the University of California-Riverside (UCR) archive. The experimental results show that practically all the methods are statistically equivalent. In the second contribution, we addressed the problem of long time series classification by introducing a new similarity measure called "Local Extrema Dynamic Time Warping" (LE-DTW). LE-DTW first transforms the original time series into low dimensional space by extracting local-extrema features. Next, it compares the so obtained transformed time series using an adapted version of DTW. To evaluate the performance of LE-DTW, we performed extensive experiments on a large variety of datasets from the UCR archive. The results showed the efficiency and the effectiveness of our newly proposed LE-DTW, compared to some state-of-the-art methods, especially on long time series.

In the third contribution, an accelerated version of the Shape Exchange Algorithm (SEA) method (Boucheham, 2008) is proposed for Quasi-Periodic Time Series (QPTS) alignment. FastSEA is based on a more efficient, yet simple sorting algorithm that is the "Counting Sort

Algorithm". The purpose of FastSEA is to speed up the alignment process of the SEA method without affecting its quality. The last contribution is devoted to extending the effectiveness of SEA to general TSC. With this goal, we propose the "Local Matching and Distance Selection SEA" (LMDS-SEA) measure, which is mainly based on a new distance selection paradigm proposed in (Kotsifakos et al., 2016; Mori et al., 2016; Mosbah and Boucheham, 2017). Experimental results demonstrated that the proposed LMDS-SEA method performs better than SEA and rivals with DTW in terms of classification accuracy.

Keywords: Time series, Similarity measures, Distances, Time series representation, Time series mining, Classification, DTW, SEA, Distance selection.

Résumé

Les séries temporelles (ST) sont des séquences de données à valeur réelle enregistrées régulièrement dans le temps. Elles peuvent être trouvées dans pratiquement toutes les applications du monde réel, telles que la médecine, la finance, l'économie, l'industrie et bien d'autres. Dans cette thèse, nous abordons la problématique de l'établissement de similarité entre séries temporelles. La mesure de similarité des séries temporelles est un élément clé de plusieurs tâches d'exploration/fouille de ces séries y compris, mais sans s'y limiter, la classification, la recherche par similarité, le clustering et la détection d'anomalies dans ce type de données.

En raison de son importance considérable, la tâche de mesure de similarité des séries temporelles a reçu une attention grandissante et est devenue un domaine de recherche très actif dans la fouille de séries temporelles. D'une manière générale, la problématique d'établissement de similarité des séries temporelles consiste à définir une méthode permettant de comparer/aligner deux séries temporelles données, en déterminant le degré d'accord/discordance entre ces ST. Cependant, mesurer la similarité entre séries temporelles a toujours été une problématique difficile. La difficulté de cette problématique réside dans le fait que les séries temporelles sont intrinsèquement de grande dimension et portent de grandes quantités de données. De plus, diverses distorsions, souvent rencontrées dans les séries temporelles, compliquent davantage l'établissement de similarité (par exemple, le bruit, les valeurs aberrantes, le changement d'échelle du temps/amplitude et le décalage dans le temps/amplitude). Par conséquent, la mesure de similitude doit être soigneusement établie afin de saisir correctement la vraie similitude entre deux séries temporelles.

D'autre part, la classification des séries temporelles (CST) est une tâche de plus en plus importante dans le cadre de la fouille de séries temporelles. La CST est, brièvement, la tâche dédiée à la prédiction de l'étiquette (classe) d'une série temporelle requête à partir d'un ensemble de données d'apprentissage étiqueté. Au cours des deux dernières décennies, il y a eu un intérêt croissant pour la CST et, par conséquent, de nombreuses approches ont été publiées dans la littérature. Étonnamment, des études approfondies, dans ce domaine de recherche, ont rapporté que l'approche basée sur distance où le classificateur 1-Nearest Neighbor (1-NN) combiné avec une mesure de distance/similarité appropriée est plus précise que la plupart des approches existantes. De toute évidence, dans ce contexte, la mesure de similarité est un ingrédient crucial et joue un rôle essentiel dans la précision du classificateur 1-NN. Par conséquent, la grande majorité de recherches dans le cadre de la CST s'est concentrée sur le développement de nouvelles mesures de distance/similarité.

L'objectif principal de cette thèse concerne, alors, la problématique de mesure de similarité entre séries temporelles avec un focus particulier sur la classification comme domaine d'application très motivant. Nous visons donc à étudier et à développer des mesures de similarité dans le contexte de la CST qui soient compétitives (robustes, précises et efficaces) par rapport aux méthodes existantes dans la littérature. À cette fin, diverses contributions ont été proposées au cours de l'élaboration de ce travail de thèse.

Dans la première contribution, nous avons effectué une large comparaison expérimentale entre la célèbre mesure de similarité des ST "Dynamic Time Warping" (DTW) et ses variantes les plus populaires dans le cadre de la CST. Dans cette étude, nous avons évalué empiriquement les méthodes en termes de précision de classification en utilisant 85 ensembles de données de l'archive publique des ST 'Université de Californie-Riverside' (UCR). Les résultats expérimentaux montrent que pratiquement toutes les variantes sont statistiquement équivalentes. Dans la deuxième contribution, nous avons abordé le problème de la classification des longues séries temporelles en introduisant une nouvelle mesure de similarité appelée "Local Extrema Dynamic Time Warping" (LE-DTW). LE-DTW transforme d'abord les séries temporelles originelles en un espace de faible dimension en extrayant des caractéristiques locales extrêmes.

Ensuite, elle compare les séries temporelles transformées ainsi obtenues en utilisant une version adaptée de DTW. Afin d'évaluer la performance de LE-DTW, nous avons effectué des expérimentations approfondies sur une grande variété d'ensembles de données provenant de l'archive UCR. Les résultats ont montré la précision et l'efficacité de notre nouvelle proposition LE-DTW par rapport à certaines méthodes de l'état de l'art, en particulier sur les longues séries temporelles.

Dans la troisième contribution, une version accélérée de la méthode Shape Exchange Algorithm (Boucheham, 2008) est proposée pour l'alignement des Séries Temporelles Quasi-Périodiques (STQP). FastSEA est basé sur un algorithme de tri plus efficace mais simple qui est 'Counting Sort Algorithm'. Le but de FastSEA est d'accélérer le processus d'alignement de la méthode SEA sans affecter sa qualité. La dernière contribution est consacrée à l'extension de la pertinence de la méthode SEA à la classification des séries temporelles générales. Dans ce contexte, nous avons proposé la mesure "Local Matching and Distance Selection SEA" (LMDS-SEA), qui est principalement basée sur un nouveau paradigme de sélection de distance proposé dans (Kotsifakos et al., 2016; Mori et al., 2016; Mosbah and Boucheham, 2017). Les résultats expérimentaux ont montré que la méthode LMDS-SEA proposée est plus performante que SEA et rivalise avec DTW en termes de précision de classification.

Mots-clés: Séries temporelles, Mesures de similarité de ST, Distances de ST, Représentation de ST, Fouille de ST, Classification des ST, DTW, SEA, Sélection de distance.

Part I

Thesis Overview

Chapter 1

General introduction

1.1 Context, problematic and objectives

Time series (TS) are a prominent type of data represented by a collection of real-valued variables recorded usually at uniform time intervals (Esling and Agon, 2012). In fact, any data sequence ordered with respect to another aspect (e.g., logic, space, etc.) can be also considered as a time series (Abanda et al., 2019; Fawaz, 2020). Practically, time series are being created over a wide variety of application domains. In the medical domain alone, various kinds of time series are frequently generated, such as electrocardiograms (ECG), electroencephalograms (EEG), and capnograms (Ratanamahatana et al., 2005). Likewise, the same remark can be found in other fields, including, but not limited to, finance (e.g., currency trading), economics (e.g., stock market values), meteorology (e.g., daily recording of temperature and humidity), and technology (e.g., network traffic). Furthermore, it has been shown that it is possible to transform and represent other non-temporal data such as images, videos, text, and handwriting into time series (Ratanamahatana et al., 2005). In brief, time series data exist in virtually every field of human endeavor (Lin et al., 2012b).

On the other hand, the volume of time series datasets is also becoming increasingly massive, especially with the advent of the Internet of Things (IoT) and sensors where a huge amount

of such data are being produced daily (Lin et al., 2012b; Sharabiani, 2018; Tan, 2019). Given the ubiquity of time series and the large size of datasets (terabytes or even more), considerable attention has been paid to analyzing and mining such data from both researchers and practitioners over the past two decades (Abanda et al., 2019; Lin et al., 2012b; Ratanamahatana et al., 2005). Recently, it has been reported that almost half (48%) of data scientists have worked on time series data during their careers (Fawaz, 2020; Neamtu et al., 2018).

It can be easily checked through Google-Scholar the existence of various domains / applications dealing with TS, such as: time series analysis, TS classification/prediction, TS anomaly detection, TS characterization, time series similarity, etc. However, TS mining is a unifying umbrella for all these applications. In that matter, the end of December 2021, Google Scholar gives the following results for the term "Time Series Mining": since 2021: 72600, since 2020: 119000, since 2019: 148000... and since 2010: 1490000.

In this thesis, we specifically address the issue of establishing the similarity between time series. The problem of time series similarity or dissimilarity measures involves defining methods for which the similitude or difference between two given time series is correctly determined. The similarity measure is one of the most important aspects of time series analysis (Esling and Agon, 2012; Fu, 2011; Wang et al., 2013). It is the core procedure for almost every time series mining task, such as classification, clustering, similarity search, indexing, and anomalies detection in time series data (Esling and Agon, 2012; Fu, 2011; Lin et al., 2012b; Paparrizos et al., 2020).

Time series similarity measure has been gaining great importance over the years and has become a very active research area in time series mining (Lin et al., 2012b; Paparrizos et al., 2020; Wang et al., 2013). However, formalizing similarity measures is a challenging task due to the inherent nature of time series, includes high dimensionality and a vast amount of datasets (Abanda et al., 2019). In addition, various distortions often encountered in time series data come to further complicate the measurement of similarity between time series (e.g., noise, outliers, amplitude scaling, amplitude shift, time scaling, and time shift). For example, two similar time series could be scaled or shifted to each other on the time and/or amplitude axis. Thus,

there is a strong need for robust similarity measures capable of handling the intrinsic characteristics of time series and properly capturing the underlying similarity between time series.

As already mentioned, the establishment of time series similarity is of fundamental importance for several time series mining tasks. Particularly, time series classification (TSC) is one of the attractive and active topics in this area. TSC applications touch a wide range of fields ranging from cyber-security (e.g., malware detection), medicine (e.g., anomaly detection), and biometrics (e.g., person identification), to cite just a few. Generally, TSC is the process of assigning a given time series to a correct class from a labeled training set using a specific classification model.

In the past two decades, great effort has been devoted to the study of the TSC problem and various methods have been proposed. However, extensive experiments have shown that the distance-based approach, where the 1-Nearest Neighbor (1-NN) classifier coupled with a proper distance or similarity measure, achieves good performance and outperforms most existing classifiers such as Support Vector Machines (SVMs), Decision Trees (DTs), and Artificial Neural Networks (ANNs) (Abanda et al., 2019; Xi et al., 2006). The outperformance of the 1-NN classifier is mainly due to the choice of distance measures and not to the 1-NN classifier itself (Abanda et al., 2019). In other words, similarity measures have a significant impact on the performance of the 1-NN classifier (Xing et al., 2010). As a result, a vast majority of works in the TSC topic has focused on developing different types of similarity measures for the 1-NN classifier over the last two decades (Bagnall et al., 2017).

In this thesis, we focus on the problem of measuring similarity between time series with a particular interest in classification as a very motivating field of application. Specifically, we aim to study and develop new and competitive similarity measures within TSC. This means that these techniques shall be: 1) as efficient, 2) as effective, and 3) as robust as possible, with respect to existing methods in the literature:

1. **Efficiency:** On one hand, time series are naturally high dimensional and increasingly large in data size; and on the other hand, there is a growing demand to analyze time

series on applications or devices with limited resources in terms of computing and storage capacity, such as sensors, embedded systems, mobile phones, and IoT (Lang et al., 2009; Sharabiani et al., 2017). Subsequently, similarity measures need to be faster and more efficient to deal with these challenges.

2. **Effectiveness:** With the increasing computing capacity of computers (GPU, TPU, and parallel architectures), effectiveness is the main goal of designing similarity measures in numerous applications (Schäfer, 2016). Effective similarity measures reflect a very precise prediction of many time series mining tasks.

3. **Robustness:** The similarity measure should have a high level of abstraction in which it can capture the similarity between two time series even if one is distorted with respect to the other (Esling and Agon, 2012).

1.2 Contributions

The main contributions of this thesis are listed as follows:

1. **An Empirical Comparison Study of Dynamic Time Warping's Variants for Time Series Classification** (Lahreche and Boucheham, 2021a): One of the most famous and effective methods in the matter of TS similarity measure is certainly Dynamic Time Warping (DTW) (Silva et al., 2018). This fact encouraged us, as a first contribution, to perform an empirical comparative study supported by statistical analysis between DTW as a common time series similarity measure and its most popular variants in the TSC literature. To that end, we combine each of these methods with the 1-NN classifier and evaluate them in terms of classification accuracy using 85 time series datasets from the UCR archive (Chen et al., 2015a). The experimental results show that practically all the methods are statistically equivalent.

2. **A Fast and Accurate Similarity Measure for Long Time Series Classification Based on Local Extrema and Dynamic Time Warping** (Lahreche and Boucheham, 2021b):

As a second contribution, we propose a new similarity measure for TSC called 'Local Extrema DTW' (LE-DTW). First, we transform original sequences into a low-dimensional space using local extrema points. Then, for each transformed time series, we physically separate between local minima and local maxima. Next, we compare the local minima points of the first transformed time series with the local minima points of the second series using the DTW method. Similarly, we apply DTW on local maxima points. At last, we combine the results using a sum function to obtain the final similarity. In order to verify the validity of our method, we carry out several experiments in the context of TSC using a wide variety of datasets from the UCR archive. The results thus obtained demonstrate that LE-DTW is both fast and accurate compared to some existing popular distances, especially for long time series.

3. **FastSEA: A Very Fast and Very Effective Matching Technique for Very Complex Time Series** (Lahreche and Boucheham, 2017): As a third contribution, an accelerated version of SEA is introduced for Quasi-Periodic Time Series (QPTS) alignment. The newly proposed method is called FastSEA, and it based on a simple and more efficient sorting algorithm named "Counting Sort Algorithm". We compare the efficiency and alignment quality of FastSEA with those of the original SEA on ECG signals from the MIT-BIH (Massachusetts Institute of Technology-Beth Israel Hospital) public database (Goldberger et al., 2000; Moody and Mark, 2001). Both quantitative and qualitative results show that FastSEA is several times faster than SEA and provides similar alignment quality.

4. **LMDS-SEA: Upgrading the Shape Exchange Algorithm (SEA) to Handle General Time Series Classification by Local Matching and Distance Selection** (Lahreche and Boucheham, 2018): As a fourth contribution, we propose 'Local Matching and Distance Selection Shape Exchange Algorithm' (LMDS-SEA), an improved version of the SEA method (Boucheham, 2008). Based on the observation that the SEA method was designed in the context of Quasi-Periodic Time Series (QPTS) Matching (Boucheham, 2008), in LMDS-SEA, we aim to upgrade the effectiveness of SEA to handle general

time series, especially in the context of TSC. For that, we use the distance selection paradigm (Kotsifakos et al., 2016; Mori et al., 2016; Mosbah and Boucheham, 2017) in which we select the most appropriate distance from the pool: Euclidean Distance (ED), Local Matching SEA (LM-SEA), and SEA for a particular TSC problem. We evaluate the proposed method on 85 UCR time series datasets. Experimental results indicate that LMDS-SEA performs better than the SEA method and rivals with the DTW method in terms of classification accuracy.

1.3 Thesis outline

The structure of this manuscript is organized as follows. In Chapter 2, we first introduce background related to time series; then, we present state-of-the-art regarding time series similarity measures. Following that, we review the most common techniques for time series representation and dimensionality reduction. Chapter 3 concerns the TSC topic in which we begin by discussing general classification algorithms; next, we review classification algorithms specific to time series. In Chapter 4, we carry out an experimental comparative study reinforced by statistical analysis between the famous DTW similarity measure and its most popular variants for TSC. Chapter 5 is devoted to our major contribution in this thesis. It introduces a new fast and accurate similarity measure called LE-DTW for long time series classification. In Chapter 6, we focus on improving the performance of the SEA method. Indeed, this chapter is composed of two parts. In the first part, we aim to speed up the SEA method without any loss of its precision for quasi-periodic time series alignment. In the second part, on the other hand, we are interested in upgrading the accuracy of SEA in the context of general TSC by introducing a new distance selection paradigm. At last, we conclude this thesis with a general conclusion and directions for future work.

Part II

State of the Art of Time Series

Chapter 2

Time series similarity measures and representations

2.1 Introduction

In this chapter, we begin by highlighting the necessary definitions and notations for time series. Next, we provide the background and related works on time-series similarity measures. Specifically, we formulate the concept of time series similarity measures and discuss the main obstacles associated with solving this problem. Then, we present the most common time series distortions. After that, we give an overview of existing time series similarity measures. Finally, we review the most commonly used time series representation techniques.

2.2 Definitions and notation

In this section, we provide basic definitions related to time series that are used throughout this manuscript.

2.2.1 Definition 1 (Time series data)

A time series is a chronologically ordered sequence of real-valued variables typically recorded at regular time intervals (Esling and Agon, 2012). Time series can be found in any domain of human endeavors that involves measurements over time including finance, econometrics, medicine, meteorology, industry, astronomy, communication engineering, and many others (Fu, 2011). However, any data sequence ordered with respect to some other well-defined aspect, (i.e., space, logic), can also be considered as time series, such as videos, DNA, image, and handwritten text (Mueen et al., 2009; Yankov et al., 2007).

In the literature, two types of time series are reported: *univariate* and *multivariate*.

2.2.2 Definition 2 (Univariate time series)

A univariate, or equivalently unidimensional, time series is a sequence of pairs (x_i, t_i) in which only one value x_i is recorded at each time stamp t_i. For example, daily temperature sampling using a single sensor and hourly measurement of electrical energy consumption in a given region produce univariate time series.

A univariate time series can be formally expressed as follows:

$$T = \{(x_1,t_1),(x_2,t_2),...,(x_i,t_i),...,(x_n,t_n)\} \tag{2.1}$$

Where, $x_i \in \mathbb{R}, \forall i, j$ such that $i < j, t_i < t_j$, and $|T| = n$ is the length of T.

As noted earlier, most commonly, the elements of time series are equally spaced in time (Esling and Agon, 2012). Formally, given a time series T, $\forall i, (i+1) \rightarrow (t_{i-1} - t_i) = c$, such that c is a constant value. With this in mind, it is not necessary to explicitly define the time index t_i in formula (2.1) (Silva et al., 2018), and hence, time series T can simply be abbreviated as:

$$T = \{(x_1),(x_2),...,(x_i),...,(x_n)\} \tag{2.2}$$

The same applies to multivariate time series.

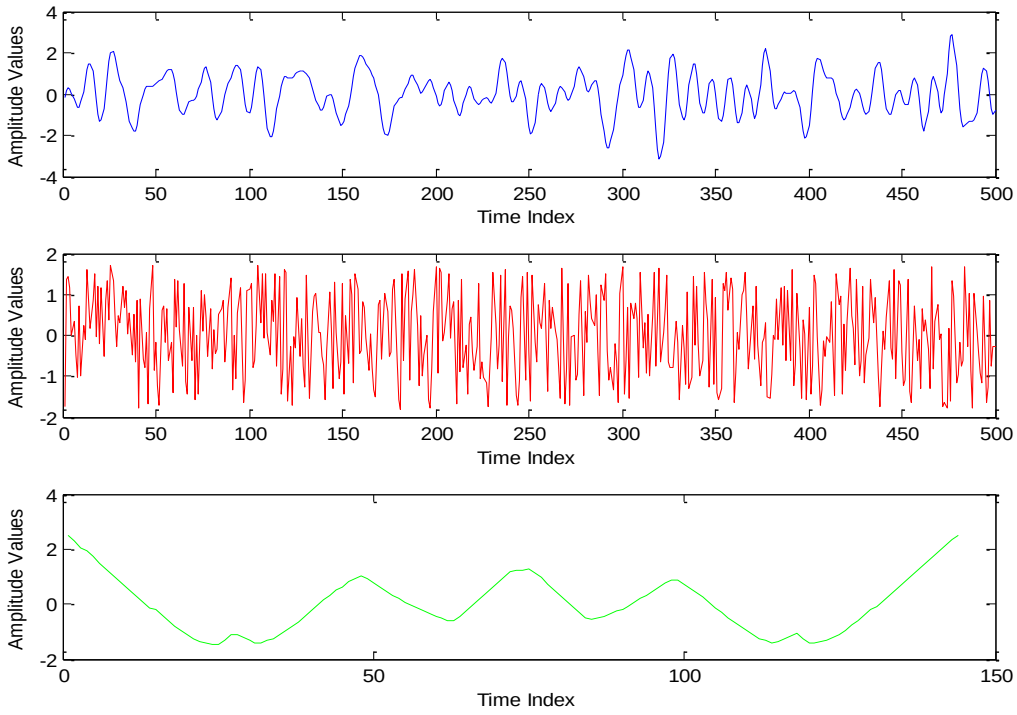

Figure 2.1: An example of three different univariate time series taken from the UCR archive (Chen et al., 2015a): Top) FordA, Middle) ShapeletSim, and Bottom) Plane.

2.2.3 Definition 3 (Multivariate time series)

A multivariate or multidimensional, time series $\mathbb{T} = \{T^1, ..., T^i, ..., T^N\}$ is a set of N different univariate time series of length n, $T^i \in \mathbb{R}^n$. In other words, at any time stamp t_i, several values (a vector of N-dimensional measurements) are recorded simultaneously. For example, the simultaneous use of several sensors to measure temperature, pressure, etc., generates a multivariate time series.

Time series are usually plotted via a graph where the X-axis denotes the time index t_i and the Y-axis denotes the amplitude values x_i. Figure 2.1 shows an example of three univariate time series taken from the UCR archive.

In this thesis, we particularly focus on univariate time series. Thus, the term *time series* will be used to refer to *univariate time series*.

2.2.4 Definition 4 (Time series dimensionality)

Time series dimensionality usually refers to the number of data points, or alternatively the length of time series (Fu, 2011; Schäfer, 2015b). A time series of length n can be considered as an object in n-dimensional space, where each data point represents a dimension (Wang et al., 2013; Echihabi et al., 2020).

Note that, for many applications, there is a need to deal with local subsequences of time series rather than the whole time series (Mueen et al., 2009).

2.2.5 Definition 5 (Time series subsequence)

Given a time series T of length n, a subsequence S of T is a subset of l consecutive values from T that begin from time stamp t_i such that: $S = \{s_i, s_{(i+1)}, s_{(i+l-1)}\}$, where $s_i \in T, i \geq 1$ and $|S| = l \leq n$.

Time series are often generated in large volumes (Lin et al., 2012b), such that they need to be efficiently stored and organized in datasets, in order to facilitate their analysis.

2.2.6 Definition 6 (Time series dataset)

A time series dataset $D = \{T_1, T_2, ..., T_M\}$ is an unordered collection of M univariate or multivariate time series of probably different lengths. D could either be a labeled or unlabeled time series dataset.

2.2.7 Definition 7 (Labeled time series dataset)

A labeled time series dataset $D = \{(T_1, C_1), (T_2, C_2), ..., (T_M, C_M)\}$ consists of a set of pairs (T_i, C_i) where each time series T_i (either univariate or multivariate) is associated with its own class label, $C_i \in [1, K]$ such that K is the number of classes in D. On the other hand, a time series dataset without classes is called an *unlabeled time series dataset*.

For simplicity and without any loss of generality, throughout the rest of this manuscript, the term *labeled time series dataset* will be referred to as a *time series dataset* and the term

database will be used to denote a *set of time series datasets*.

2.3 Time series similarity measures

In this section, we first give preliminaries about time series similarity measure concept. Then, we discuss the major challenges that researchers confront while attempting to establish the similarity between time series. Next, we provide a review of the basic distortions often encountered within time series. After that, we present an overview of common time series similarity measures, available in the literature.

2.3.1 Preliminaries

The term *similarity* usually represents a numerical description of how alike two objects are (Geler, 2015). This notion has always been a major concept of interest in many fields of science and technology (Deza and Deza, 2009). It is particularly used in mathematics, physics, psychology, economics, biology, medicine, and computer science, to cite just a few examples. Needless to mention that, computer science field include diverse and important domains, such as pattern recognition, speech recognition, signal processing, image processing, machine learning, data mining, time series analysis and mining, etc.

Particularly, in time series analysis and mining, similarity measure is one of the most important aspects (Esling and Agon, 2012; Paparrizos et al., 2020). It is the core procedure for almost every time series mining task including classification, clustering, similarity search, indexing, and anomaly/novelty/motif detection (Fu, 2011; Lin et al., 2012b). Because of its fundamental importance, the past two decades have seen an explosion of interest in establishing the similarity between time series from both academia and industrial sectors.

The task of time series similarity measure consists in establishing a method capable of comparing two given time series and determining their degree of resemblance. The similarity score is usually expressed by a real value in [0, 1] interval (Geler, 2015). The closer the score value is to one "1", the more similar the time series are, and vice versa.

More formally, let Q and C be two time series, the purpose is to define a function f, $f(Q,C) \rightarrow [0,1]$ that takes Q and C as inputs and returns a non-negative real value $0 \leq v \leq 1$ reflecting how similar they are to each other.

It should be noted that there is a distinction between the terms *similarity*, *distance*, *metric*, and *alignment* of time series as follows:

Similarity measure: A function, $s : Q * C \rightarrow \mathbb{R}^+$, is called a time series similarity measure, if it satisfies the following properties (Deza and Deza, 2009):

- Non-negativity (equivalently, positivity): $s(Q,C) \geq 0$.
- Symmetry: $s(Q,C) = s(C,Q)$.
- $s(Q,C) \leq s(Q,Q)$, if $Q \neq C$.
- $s(Q,Q) = 1$.

Distance measure: A function, $d : Q * C \rightarrow \mathbb{R}^+$, is called a time series distance, or dissimilarity, measure, if it satisfies the following properties (Deza and Deza, 2009) :

- Non-negativity: $d(Q,C) \geq 0$.
- Symmetry: $d(Q,C) = d(C,Q)$.
- Reflexivity: $d(Q,Q) = 0$.

The distance measure in some sense is the opposite of the similarity measure (Schäfer, 2015b; Spiegel, 2015). It returns a small value for similar time series and a large value for dissimilar time series. Technically, it is possible to convert a similarity measure s to a distance measure d using the formulas below (Deza and Deza, 2009):

$$d(Q,C) = 1 - s(Q,C) \tag{2.3}$$

$$d(Q,C) = \frac{(1 - s(Q,C))}{s(Q,C)} \tag{2.4}$$

$$d(Q,C) = \sqrt{(1 - s(Q,C))} \tag{2.5}$$

For simplicity, the terms *distance* and *similarity measures* will be used interchangeably.

Distance metric: A time series distance metric, $D : Q * C \rightarrow \mathbb{R}^+$, is a function that satisfies the following properties (Deza and Deza, 2009):

- Non-negativity: $D(Q,C) \geq 0$.
- Symmetry: $D(Q,C) = D(C,Q)$.
- Identity: $d(Q,C) = 0$, if and only if $Q = C$.
- Triangle inequality: $D(Q,C) \leq D(Q,T) + D(T,C)$.

The distance metric is a special distance measure that obeys the additional axiom of triangular inequality. Notably, time series distance metric is of particular importance for indexing in time series datasets (Keogh and Ratanamahatana, 2005; Lin et al., 2012b; Schäfer, 2015b).

Time series alignment: As distance/similarity measures, time series alignment methods may also be used to quantify the degree of resemblance between two time series. In addition and more importantly, they allow creating a mapping between the points of a time series and those of another time series.

Note that, in Section 2.3.4 we will present a detailed categorization of time series similarity/distance measures.

2.3.2 Why is establishing time-series similarity not trivial?

Establishing similarity between time series is not straightforward (Fu, 2011; Renard, 2017). Although humans can easily derive an abstract shape from time series and determine similarities between them due to their natural intuition, it remains a hard problem for computers and requires a lot of effort and serious attention (Esling and Agon, 2012; Schäfer, 2015b). The difficulties in dealing with the time series similarity problem relate to the intrinsic nature of time series, including:

- **High dimensionality:** Time series are typically high-dimensional data (Fu, 2011). For example, a sensor that records only one value per minute will generate a time series containing more than 0.5 million data points per year (Schäfer, 2016).

- **Large data size:** The volume of time series datasets has become increasingly huge (terabytes scale or even more) (Lin et al., 2012b), especially with the advent of new technologies (e.g., wearable sensors, satellites, smart mobile phones, Internet of Things (IoT), etc.). For instance, recording eight hours of EEG signal can occupy more than a gigabyte of memory space (Ratanamahatana et al., 2005), and the public satellite image time series dataset contains one million instances (Tan, 2017; Tan et al., 2017).

- **Distortions:** Time series are generally obtained from real-world scenario (Renard, 2017; Chen et al., 2005), so data quality can be affected by surrounding factors through producing distorted time series. A concrete example, time series may exhibit noise or outliers due to sensor disturbance during the recording process. In the literature, there is a variety of time series distortions and several studies have been documented to explain this major challenge. Considering all of this importance, we will dedicate the following subsection to highlighting time series distortions.

- **Quasi-Periodicity:** Quasi-Periodicity is another big challenge that researchers face when establishing the similarity between time series. To the best of my knowledge, Boucheham (2008) is the first researcher having shed light on this new breach of TS matching, and according to him, quasi-periodicity is the most difficult situation that may arise in time series matching. Quasi-periodicity occurs in many real-world time series including, ECG, EEG, and capnogram. Quasi-periodic time series can be defined as a collection of many quasi-similar sub-sequences of eventually different lengths and may be distorted on the time/amplitude axis.

- Time series may also have different lengths and even contain missing values.

These inherent properties of time series pose challenges that make similarity measurement of time series a complex problem. There is thus a crucial need for robust similarity measures in

which the underlying similarity between time series is correctly identified. According to Esling and Agon (2012), a time series similarity measure is of interest if it provides the following criteria:

- *"It should provide recognition of perceptually similar objects, even though they are not mathematically identical.*
- *It should be consistent with human intuition.*
- *It should emphasize the most salient features on both local and global scales.*
- *A similarity measure should be universal in the sense that it allows to identify or distinguish arbitrary objects, that is, no restrictions on time series are assumed.*
- *It should abstract from distortions and be invariant to a set of transformations".*

2.3.3 Time series distortions

Since time series data are usually the result of real-world measurements, they are subject to many external events that can alter their natural behavior (Chen et al., 2005; Renard, 2017). Besides surrounding events (environment), many phenomena generate natural variabilities in time series, such as biological signals (e.g., ECG, motion capture, and handwriting) (Batista et al., 2014). For instance, the ECG of a person at rest differs from that of the same person when running in terms of local variations. In this section, we review the basic distortions frequently encountered within time series.

- **Time shift:** Two similar time series present a time shift distortion (Fig. 2.2-A) means that their patterns appear at different positions on the time axis (X-axis). Otherwise said, they are globally shifted, relative to each other, on the time axis. This form of variance might arise when, for example, transforming a shape into time series but starting at different positions.
- **Time scaling:** A distortion is called time scaling (Fig. 2.2-B) when there exist two simi-

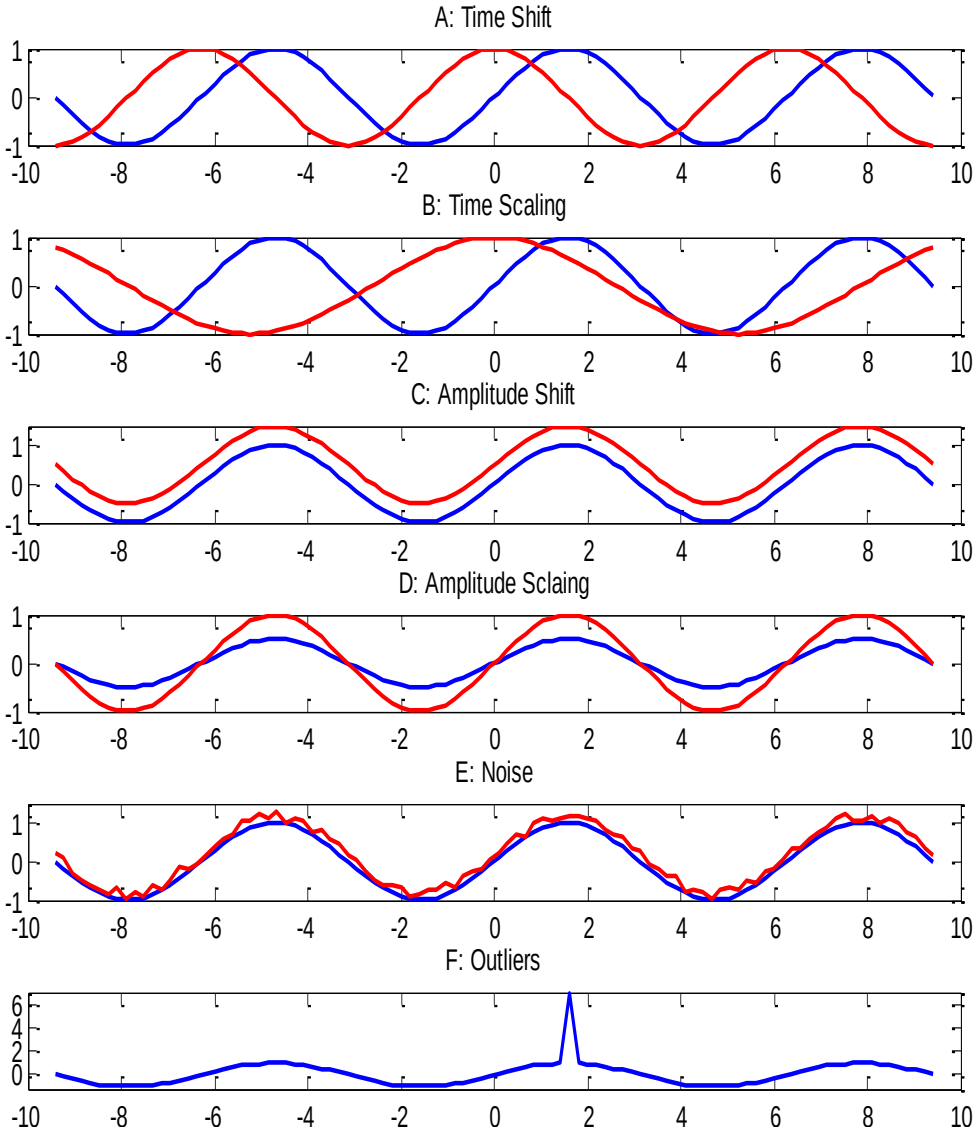

Figure 2.2: Illustration of basic typical time series distortions: A) Time shift, B) Time scaling, C) Amplitude shift, D) Amplitude scaling, E) Noise, and F) Outliers.

lar time series, but they are not locally aligned on the time axis, each, with respect to the other. Said another way, one TS sampling rate is locally accelerated or decelerated with regard to the other TS sampling rate. Time scaling is probably the most frequently encountered distortion in time series (Batista et al., 2014). For example, tracking a moving object at various speeds might cause local shift in trajectories (Chen et al., 2005).

- **Amplitude shift:** Amplitude shift (Fig. 2.2-C) is a circumstance in which two similar time series appear at significantly different levels on the amplitude axis (Y-axis) from

each other.

- **Amplitude scaling:** Amplitude scaling (Fig. 2.2-D) is defined where there are two similar time series but one behaves, to a significant extent, at local level, differently on the amplitudes axis with respect to the other. In other words, they are locally offset from each other on the amplitudes axis.

- **Noise:** A noisy time series contains burrs in which its shape appears a bit rough (Fig. 2.2-E). This is due, for example, to disturbance signals.

- **Outliers:** An outlier can simply be defined as a time series value that deviates significantly from other values. In Fig. 2.2-F, the value in the time stamp $t_i = 59$ is considered as an outlier. For instance, sensor failures and errors in detection techniques can produce outliers (Chen et al., 2005).

One of the main issues when dealing with time series data is how to handle the distortions mentioned above (Batista et al., 2014; Paparrizos and Gravano, 2017). One could suggest using a preprocessing step for this purpose. Unfortunately, data preprocessing is not enough and cannot eliminate the most complicated distortions (e.g., time shift and scaling) (Chen et al., 2005). Hence, more sophisticated techniques are still needed to overcome the underlying difficulties. In this regard, two complementary concepts: time series representation and similarity measures have shown to be a good solution (Renard, 2017; Schäfer, 2015b). In the current section, we examine time series similarity measures, and in the next, we go on to time series representations.

2.3.4 Existing time series similarity measures

Research on time series similarity measures has become very popular in the last decade, and the literature shows a multitude as well as a wide variety of approaches (Wang et al., 2013). Regardless of the progress in the area of interest, the major problem is that it is difficult to assimilate all this large number of measures that exist. Many researchers have suggested organizing them into groups in order to facilitate their understanding and to give a comprehensive

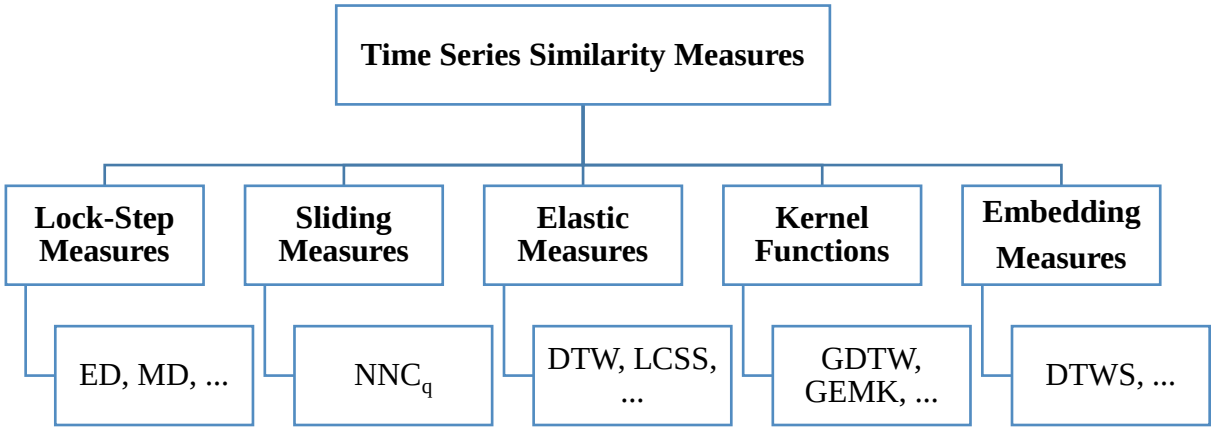

Figure 2.3: A taxonomy of time series similarity measures (Paparrizos et al., 2020).

overview to the scientific community (Boucheham, 2013; Esling and Agon, 2012; Montero and Vilar, 2015; Paparrizos et al., 2020; Wang et al., 2013).

One of the first categorizations of time series similarity measures is presented in Esling and Agon (2012), which includes four groups: shape-based measures, edit-based measures, features-based measures, and structure-based measures. In (Wang et al., 2013), another four groups are also suggested: lock-step measures, elastic measures, threshold-based measures, and pattern-based measures. Montero and Vilar (2015) also found that time series similarity measures can be organized into four different groups: model-free measures, model-based measures, complexity-based measures, and prediction-based measures. In (Boucheham, 2013), time series comparison methods are classified into three groups based on the type of the data they can handle: simplest (TS with no periodicity, no phase shift), medium (TS with no periodicity, but with phase shift), and highest (TS with periodicity and phase shift). In a recent paper by Paparrizos et al. (2020), a new taxonomy with five groups is described: lock-step measures, sliding measures, elastic measures, kernel functions, and embedding measures.

Although differences of categorization exist, in my opinion, the last categorization is the most comprehensive since it covers both old and recent developments in the field of interest. There-

fore, we chose to pursue and discuss this categorization in the following. Figure 2.3 depicts this categorization with five groups of time series similarity measures.

2.3.4.1 Lock-step measures

Similarity measures belonging to this category work by creating a linear mapping between the data points of the two time series to be compared. They compare the i^{th} data point of one time series to the i^{th} data point of the other time series. The main advantages of lock-step measures are that they are simples and have linear time complexity $O(n)$ with the length of time series. However, they are only limited to time series of equal lengths and are sensitive to noise and diverse distortions (Wang et al., 2013).

There are dozens of lock-step measures in the literature. Nevertheless, Euclidean Distance (ED) (Eq. 2.6) (Faloutsos et al., 1994) is the most appreciated measure within this category. Here, we list some lock-step measures and refer interested readers to (Cha, 2007) for more examples. Let Q and C be two time series of length n.

- **Euclidean Distance:** ED is defined as:

$$ED(Q,C) = \sqrt{\sum_{i=1}^{n}(q_i - c_i)^2} \tag{2.6}$$

- **Manhattan Distance:** MD is defined as:

$$MD(Q,C) = \sum_{i=1}^{n}|q_i - c_i| \tag{2.7}$$

- **Canberra Distance:** CD is defined as:

$$CD(Q,C) = \sum_{i=1}^{n}\frac{|q_i - c_i|}{q_i + c_i} \tag{2.8}$$

- **Ruzicka Distance:** RD is defined as:

$$RD(Q,C) = \frac{\sum_{i=1}^{n} min(q_i, c_i)}{\sum_{i=1}^{n} max(q_i, c_i)} \tag{2.9}$$

- **Inner Product Distance:** ID is defined as:

$$ID(Q,C) = \sum_{i=1}^{n} (q_i.c_i) \tag{2.10}$$

2.3.4.2 Sliding measures

The basic principle of sliding, also known as cross-correlation, measures is to linearly compare a time series to all shifted versions of another time series (Paparrizos et al., 2020). More specifically, they take a time series and slide it along the time axis of the other time series. For each shift, they evaluate the similarity between the corresponding sequences using the inner product. Sliding measures look for the combination that maximizes the correlation or equivalently minimizes the inner product between the time series to be compared (Paparrizos et al., 2020).

Formally, given Q and C two time series of length n. Sliding measures first slide Q along the time axis by $|s|$ positions to the right $s \geq 0$ or left $s < 0$, such that (Paparrizos and Gravano, 2017):

$$Q_{(s)} = \begin{cases} ((0_1, ..., 0_s), q_1, q_2, ..., q_n), s \geq 0 \\ (q_{1-s}, q_{n-1}, q_n, (0_1, ..., 0_s)), s < 0 \end{cases} \tag{2.11}$$

When all possible shifts $Q_{(s)}$ are taken into account, so that $s \in [-n, n]$, the cross-correlation $CC_w(Q,C), w \in \{1, 2, ..., 2n-1\}$ is then calculated between C and all shifts of $Q_{(s)}$ using the inner product.

With the sliding strategy, the cross-correlation measurements are invariant to time shift distortion, but their computational cost is very high, which is of quadratic order $O(n^2)$. Fortunately, by using the Fast Fourier Transform (FFT) (Cooley and Tukey, 1965) and its inverse version

(IFFT), the time complexity can be reduced to $O(n.log(n))$:

$$CC_w(Q,C) = F^{-1}(F(Q),F(C)) \tag{2.12}$$

Cross-correlation has applications in different scientific fields, in particular signal processing and, recently, deep learning (LeCun et al., 1995, 2015). In time series clustering, the state-of-the-art method is based on a cross-correlation technique (Paparrizos and Gravano, 2015, 2017).

The most popular cross-correlation variants are the following (Paparrizos and Gravano, 2017; Paparrizos et al., 2020):

$$NCC_q(Q,C) = \begin{cases} max(\frac{CC_w(Q,C)}{m}), q = "b" (NCC_b) \\ max(\frac{CC_w(Q,C)}{m-|w-m|}), q = "u" (NCC_u) \\ max(\frac{CC_w(Q,C)}{||Q||.||C||}), q = "c" (NCC_c) \\ max(CC_w(Q,C)), q = "." (NCC) \end{cases} \tag{2.13}$$

Where, NNC_b is the biased estimator, NNC_u is the unbiased estimator, and NNC_c is the coefficient normalization or SBD (Paparrizos and Gravano, 2015, 2017).

2.3.4.3 Elastic measures

Unlike lock-step measures, elastic measures aim to perform a more flexible alignment by creating a non-linear mapping between the time series elements. Each time series data point can be aligned with one, none, or many data points from another time series. This particular property allows elastic measures to compare time series under time shift/scaling distortions and of different lengths. However, elastic measures suffer from high computational complexity because they often rely on the dynamic programming paradigm. Figure 2.4 illustrates the difference between the lock-step alignment using ED and the elastic alignment using DTW.

The past two decades have seen a growing trend towards elastic measures as most research in the context of time series comparison has focused on such measurements (Tan et al., 2020). Consequently, many methods have been proposed.

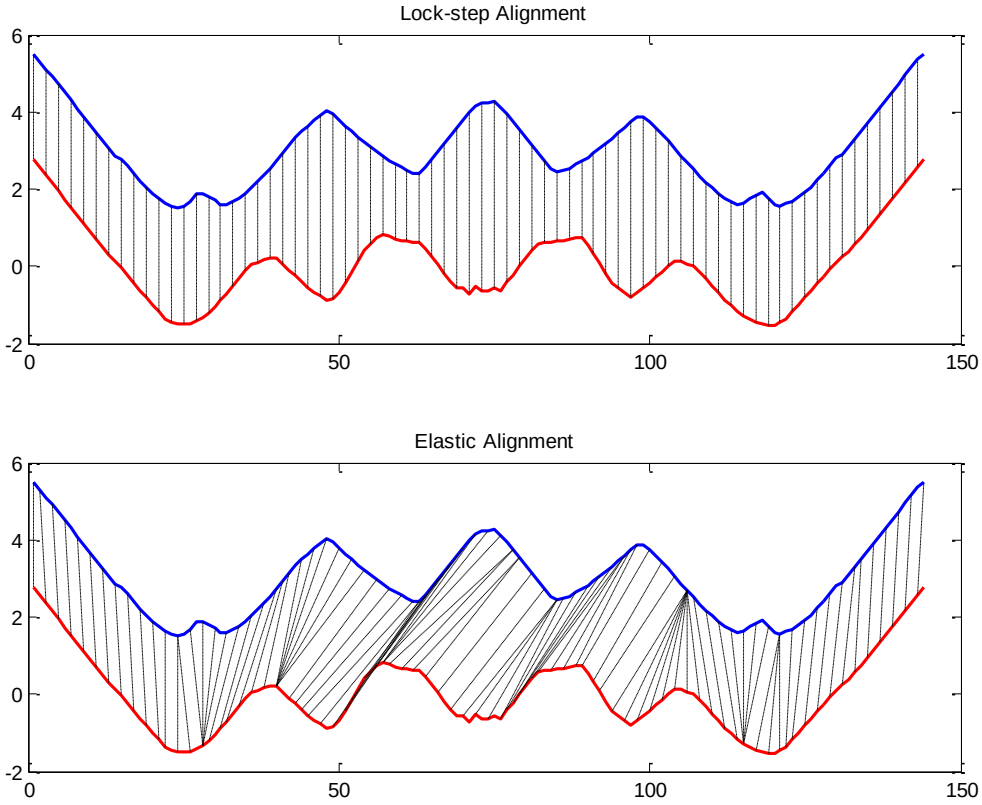

Figure 2.4: Illustration of the difference between the lock-step alignment (using ED) and the elastic alignment (using DTW).

Dynamic Time Warping (DTW): Dynamic Time Warping (DTW) is the most popular and effective similarity measure for time series (Silva et al., 2018). It was proposed for the first time in the context of speech recognition (Sakoe and Chiba, 1978). After that, ,Berndt and Clifford (1994) introduced it in the time series mining area. Since then, DTW has been effectively applied to resolve several problems from diverse domains, including robotics, biometrics, and meteorology (Mueen and Keogh, 2016).

The DTW between Q and C, two time series of length n and m, respectively is computed as follows. First, a matrix $D(n,m)$ is constructed, where each cell $D(i,j)$ contains the local distance $d(x_i,y_j)$ between the i^{th} and j^{th} data points of Q and C, respectively. Euclidean distance is typically the most used to calculate this distance. Second, DTW looks for the warping path W that verify the constraints below:

- $W = w_1, w_2, ..., w_k, ..., w_K$, where $w_k = (i, j)_k$ and $min(n,m) \leq K \leq n+m-1$.
- Boundary condition: this means that the warping path must begin and finish at the first and last cells of the matrix, respectively: $w_1 = (1,1), w_K = (n,m)$.
- Continuity: this restricts the next element $w_{(k+1)} = (i', j')$ of the warping path to be adjacent with the current element $w_k = (i, j)$. Otherwise: $i' - i \leq 1$ and $j' - j \leq 1$.
- Monotonicity: this forces the warping path to not decrease. In other words, for any $w_k = (i, j)$ and $w_{(k+1)} = (i', j') \rightarrow i' \geq i$ and $j' \geq j$.

In fact, several warping paths satisfy the above restrictions. However, DTW is only interested in the one that optimizes the alignment between Q and C. Specifically, it only keeps the optimal warping path which minimizes the accumulated local distances.

$$DTW(Q,C) = min\left\{\frac{1}{K}\sqrt{\sum_{k=1}^{K} w_k}\right\} \tag{2.14}$$

In practice, searching for the optimal warping path is computationally expensive. To speed up the process, a dynamic programming approach is adopted as shown in the bellow recursive function:

$$D(i,j) = d(q_i - c_i) + min\{D(i-1, j-1), D(i-1, j), D(i, j-1)\} \tag{2.15}$$

The final distance between Q and C is given by:

$$D(Q,C) = \sqrt{D(n,m)} \tag{2.16}$$

Figure 2.5 presents a visualization of how the DTW algorithm works.

Longest Common Subsequence (LCSS): The Longest Common Subsequence (LCSS) is an edit-based distance that was originally proposed in the context of string sequences (Hirschberg, 1975). Later, LCSS was extended to time series data by adopting a matching threshold value ε (Vlachos et al., 2003). The intuition is that two data points from two time series are considered a match if their distance is less than or equal to ε. LCSS aims to find the longest subsequence

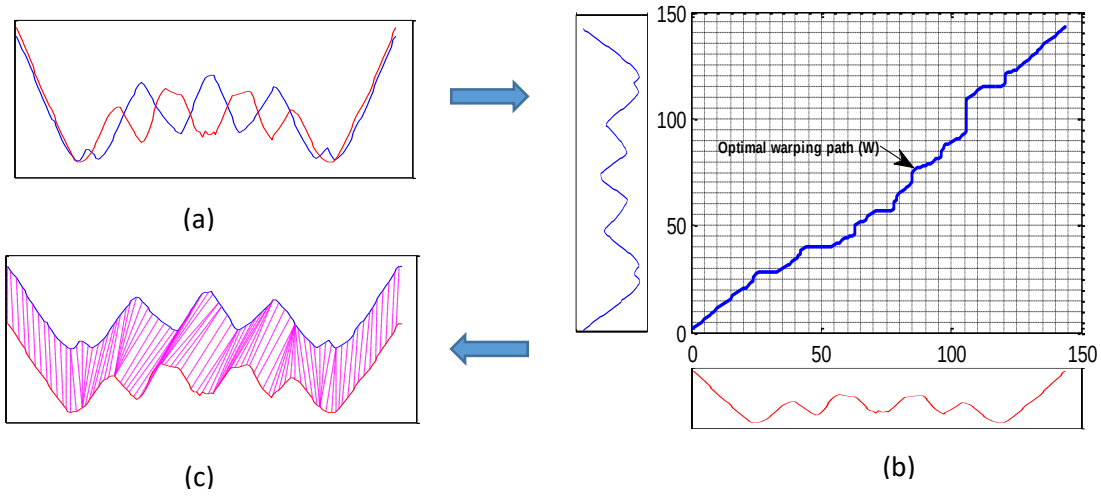

Figure 2.5: A visual example of how the DTW algorithm works. (a) Two given time series, (b) the optimal warping path, and (c) the alignment between these series.

that is common to two given time series. The length of this subsequence is, thereafter, used to express the similarity between series. The LCSS between again Q and C can be computed using a dynamic programming approach based on the recursive formula:

$$LCSS(i,j) = \begin{cases} 0, \text{if } i = 0 \; or \; j = 0 \\ 1 + LCSS(i-1, j-1), \text{if } |q_i - c_i| \leq \varepsilon \\ max(LCSS(i, j-1), LCSS(i-1, j)), \text{otherwise} \end{cases} \tag{2.17}$$

The $LCSS(n,m)$ represents the length of the longest common subsequence between Q and C, and the LCSS distance between them is (Lines and Bagnall, 2015):

$$LCSS(Q,C) = 1 - \frac{LCSS(n,m)}{min(n,m)} \tag{2.18}$$

Edit Distance on Real sequence (EDR): The Edit Distance on Real sequence (EDR) (Chen et al., 2005) is another edit-based distance for time series matching. Like LCSS, EDR also defines a threshold distance in the matching process. In contrast, EDR takes into account various gaps between two similar subsequences by adding a constant penalty for non-matching

data points. The EDR between Q and C is defined using the recursive equation:

$$EDR(i,j) = \begin{cases} j, \text{if } i = 0 \\ i, \text{if } j = 0 \\ min \begin{cases} EDR(i-1,j-1) + subcost(i,j) \\ EDR(i-1,j) + 1 \\ EDR(i,j-1) + 1), \text{otherwise} \end{cases} \end{cases} \tag{2.19}$$

Where,

$$subcost(i,j) = \begin{cases} 0, \text{if } |q_i - c_i| \leq \varepsilon \\ 1, \text{if } |q_i - c_i| > \varepsilon \end{cases} \tag{2.20}$$

Edit Distance with Real Penalty (ERP): The same author as EDR presented another elastic measure called Edit Distance with Real Penalty (ERP) (Chen and Ng, 2004). ERP is a combination of $L_p - norms$ and Edit distance. The major difference with EDR is that ERP is a distance metric. In addition, ERP assigns a real penalty expressed as the $L_1 - norm$ (Manhattan distance) between the two data points if no gap is inserted, and a constant parameter g otherwise. The ERP between Q and C is defined using the recursive equation:

$$ERP(i,j) = min \begin{cases} ERP(i-1,j-1) + |q_i - c_i| \\ ERP(i-1,j) + |q_i - g| \\ ERP(i,j-1) + |c_i - g| \end{cases} \tag{2.21}$$

Time Warp Edit Distance (TWED): (Marteau, 2008) introduced an elastic distance metric referred to as Time Warp Edit Distance (TWED). An important feature of TWED is that it takes the timestamp differences between the elements of both time series into consideration. This feature helps in comparing time series with different speeds and different sampling rates. In TWED, a *stiffness* parameter v is used to control the time warping (elasticity). TWED introduces another parameter γ that is added where there is no match between data points.

Algorithm TWED(Q, C)

Parameters: stiffness parameter ν, penalty value γ
Let D be an n+1*n+1 matrix initialized to zero.
$D(1, 1) \leftarrow 0$
$D(2, 1) \leftarrow {q_1}^2$
$D(1, 2) \leftarrow {c_1}^2$
For $i \leftarrow 2$ to n+1 do
  $D(i, 1) \leftarrow D(i\text{-}1, 1) + |q_{i-2} - q_{i-1}|$
For $j \leftarrow 2$ to n+1 do
  $D(1, i) \leftarrow D(1, j\text{-}1) + |c_{i-2} - c_{i-1}|$
For $i \leftarrow 2$ to n+1 do
  For $j \leftarrow 2$ to n+1 do
    if $i > 2$ and $j > 2$ then
      dist1 $\leftarrow D(i\text{-}1, j\text{-}1) + \nu * |i\text{-}j| * 2 + |q_{i-1} - c_{j-1}| + |q_{i-2} - c_{j-2}|$
    else
      dist1 $\leftarrow D(i\text{-}1, j\text{-}1) + \nu * |i\text{-}j| + |q_{i-1} - c_{j-1}|$
    if $i > 2$ then
      dist2 $\leftarrow D(i\text{-}1, j) + |q_{i-1} - q_{i-1}|$
    else
      dist2 $\leftarrow D(i\text{-}1, j) + |q_{i-1}| + \gamma$
    if $j > 2$ then
      dist3 $\leftarrow D(i, j\text{-}1) + |c_{j-1} - c_{j-2}| + \gamma + \nu$
    else
      dist3 $\leftarrow D(i, j\text{-}1) + |c_{j-1}| + \gamma$
  $D(i, j) \leftarrow$ min(dist1, dist2, dist3)
Return $D(n+1, n+1)$

Figure 2.6: Pseudo-code of TWED algorithm.

Figure 2.6 shows the pseudo-code of TWED.

Move-Split-Merge (MSM): Stefan et al. (2012) developed a novel elastic distance metric for time series named Move-Split-Merge (MSM). As the name suggests, MSM is based on three edit operations: Move, Split and Merge. These operations are used to transform one time series into another. The authors assigned a particular cost for each edit operation. The cost of the Move operation is the Manhattan distance between the two data points. The Split and Merge operations have the same cost and it is equal to a constant value c. The detailed implementation of MSM is given in Fig. 2.7. The C value in the MSM algorithm is calculated as follows.

$$C(q_i, q_{i-1}, c_j) = \begin{cases} c \text{ if } q_{i-1} \leq q_i \leq c_j \text{ or } q_{i-1} \geq q_i \geq c_j \\ c + min(|q_i - q_{i-1}|, |q_i, c_j|) \text{ otherwise} \end{cases} \tag{2.22}$$

Table 2.1 summarizes the properties of elastic measures.

Algorithm MSM(Q, C)

Parameters: penalty value c
1: Let D be an n*n matrix initialized to zero.
2: $D(1,1) \leftarrow |q_1 - c_1|$
3: For $i \leftarrow 2$ to n do
4: $\quad D(i,1) \leftarrow D(i\text{-}1,1) + C(q_i, q_{i-1}, c_1)$
5: For $i \leftarrow 2$ to n do
6: $\quad D(1,i) \leftarrow D(1,i\text{-}1) + C(c_i, q_1, c + i - 1)$
7: For $i \leftarrow 2$ to n do
8: $\quad$ For $j \leftarrow 2$ to n do
9: $\quad\quad D(i,j) \leftarrow \min\{D(i\text{-}1,j\text{-}1) + |q_i - c_j|, D(i\text{-}1,j) + C(q_i, q_{i-1}, c_j), D(i,j\text{-}1) + C(c_j, q_i, c_{j-1})\}$
10: **Return** $D(n,n)$

Figure 2.7: Pseudo-code of the MSM algorithm.

Table 2.1: Summary of elastic measures properties.

Elastic measure	Author	Metric	Noise	Time complexity	Parameter
DTW	(Sakoe and Chiba, 1978; Berndt and Clifford, 1994)	No	No	$O(n^2)$	0
LCSS	(Hirschberg, 1975; Vlachos et al., 2003)	No	Yes	$O(n^2)$	1(ε)
EDR	(Chen et al., 2005)	No	Yes	$O(n^2)$	1(ε)
ERP	(Chen and Ng, 2004)	Yes	No	$O(n^2)$	1(ε)
TWED	(Marteau, 2008)	Yes	No	$O(n^2)$	2(v, γ)
MSM	(Stefan et al., 2012)	Yes	No	$O(n^2)$	1(c)

2.3.4.4 Kernel measures

Methods within this category involve converting time series distance measures (e.g., lock-step, sliding, elastic measures) into kernel functions for time series. In kernel measures, the time series are also transformed into a high-dimensional feature space. Fortunately, the kernel measurements make comparisons without explicitly converting the time series, allowing more efficient computation in a high-dimensional space (Paparrizos et al., 2020).

More formally, a kernel k is a similarity measure:

$$k(Q,C) = (f(Q), f(C)) \tag{2.23}$$

Where Q and C are two inputs time series, f is a function that projects the original space of the input series into a new high-dimensional space, and <,> denotes the inner product. Indeed, there is no explicit transformation of time series.

Kernel measures are at the heart of kernel methods, a class of machine learning algorithms, such as SVM (Abanda et al., 2019). They use time series distance measures to obtain a kernel

function. The goal is to combine the potential of time series distance measures with the strength of kernel methods (Abanda et al., 2019).

Jalalian and Chalup (2013) proposed GDTW, a new kernel measure for time series. It was obtained by using DTW as a distance measure in the Gaussian Kernel Function (GKD). GDTW was then employed in combination with P-SVMs, a special type of SVM, for variable length time series classification. In (Chen et al., 2015b), the authors presented a kernel sparse representation-based method for TSC. The proposed kernel function belongs to the family of Gaussian Elastic Matching Kernels (GEMK). The authors were particularly interested in transforming elastic measures including DTW, ERP, and TWED into kernel measures.

Similarly, Zhang et al. (2010) developed another time series kernel based on elastic measures. However, the authors' attention was focused on elastic measures that are metrics. Always with elastic kernels, Marteau and Gibet (2014) introduced a novel method to construct kernel function from a given time series elastic measure. They proposed to work on regularized versions of elastic measures while preserving their natural properties.

2.3.4.5 Embedding measures

So far, the similarity measures described have been employed to compare time series, but the purpose of embedded methods is different. They exploit similarity measures to create a new representation of the time series expressed as a feature vector. The goal of embedded measures is double: on one hand, they take advantage of similarity measures, and on the other, they benefit from the power of feature vector-based algorithms (Abanda et al., 2019). Specifically, embedding measures go through two major sequential phases. First, time series are converted into feature vectors using a given time series similarity measure. Second, the acquired vectors are projected into generally a Euclidean space such that the given time series distance is approximated.

In that context, Mizuhara et al. (2006) proposed to embed time series into a vector representation by using the DTW distances. They considered three embedding methods: Euclidean space by MDS, pseudo-Euclidean space, and Euclidean space by the Laplacian Eigen-map technique,

such that they approximate the DTW distance between time series. Lei et al. (2017) proposed an efficient embedding method for time series clustering. They first introduced a new similarity measure called DTWS based on the DTW distance. They then converted the raw time series into feature vectors, preserving the DTWS in the modified space.

In (Lods et al., 2017), the authors presented another embedding measure. The Shapelet Transform (ST) algorithm serves as the foundation for the suggested technique. The authors trained shapelets such that the ED between the ST-transformed vectors preserves the DTW distance. This method was also applied in the context of time series clustering.

2.4 Time series representations

As previously mentioned, time series are by nature high dimensional, very large, and present various distortions. Direct analysis and mining of such data in its raw format can considerably affect the performance of algorithms in terms of computational cost, memory consumption, and precision (Esling and Agon, 2012; Wang et al., 2013). The key is to work on the time series representations rather than the original ones (Cassisi et al., 2012). Time series representation, also called dimensionality reduction, is the task dedicated to transforming the raw time series into another low dimensional feature space while preserving its essential characteristics (Aghabozorgi et al., 2015; Renard, 2017).

Time series representation: Given a time series $T = (x_i)_{i=1:n}$ of length n, the representation of T consists in finding a transformed time series T' of low dimensionality $|T'| << |T|$ which best approximates T.

Time series representations should include the following properties (Esling and Agon, 2012):

- **Dimensionality reduction:** Project the time series into a low dimensional space.
- **Quality:** Capture discriminate characteristics of raw time series.
- **Efficiency:** Have low computational complexity.

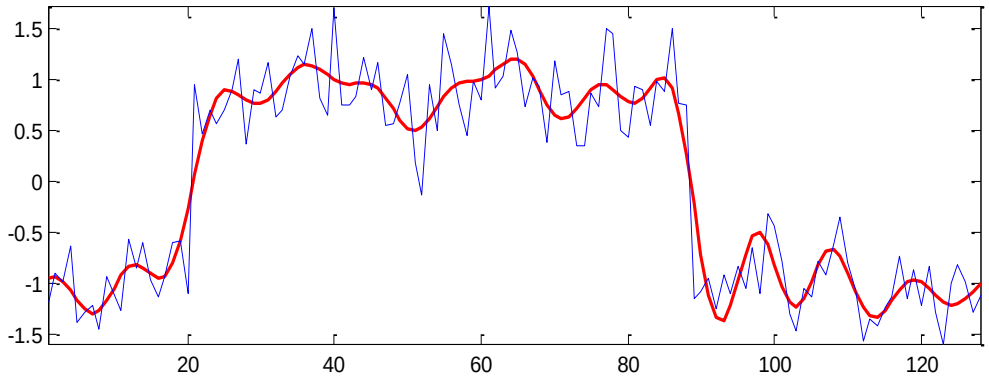

Figure 2.8: An example of a raw time series (blue curve) and its associated representation (red curve) using the DFT method with 16 coefficients.

- **Robustness:** Deal with the various time series distortions.

Time series representation has been widely researched, and several publications have appeared over the years to solve this problem. Among them, we review only the most commonly used techniques in the following.

2.4.1 Discrete Fourier Transform (DFT)

Discrete Fourier Transform (DFT) (Agrawal et al., 1993) is probably the first data representation technique proposed for time series. It transforms a time series from the time domain to the frequency domain. The idea behind DFT is that the time series can be decomposed into a sum of a finite number of sine and cosine waves. These waves are referred to as the *Fourier Series*, and each of them is represented by a complex number known as a *Fourier Coefficient*.

More formally, given a time series $T = \{x_1, x_2, ..., x_n\}$, the DFT of T is a transformed sequence $T' = \{x'_1, x'_2, ..., x'_n\}$ of n complex number, such that:

$$T'_f = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} (x_i) e^{(-2j\pi \frac{f_i}{n})}, j = \sqrt{-1}, f = 1...n \tag{2.24}$$

The DFT has a time complexity of quadratic order $O(n^2)$. But, by using an efficient version called Fast Fourier Transform (FFT) (Cooley and Tukey, 1965), the computational complexity

can be reduced to $O(nlog_n)$. An important consideration is that it is possible to work on just the first k coefficients of DFT without significantly affecting its quality and thus make DTF a more efficient representation technique although it is difficult to choose the best value of k (Fuad, 2011; Wu et al., 2000). Figure 2.8 depicts an example of a raw time series (blue curve) and its corresponding representation (red curve) using the first 16-coefficients of DFT.

2.4.2 Discrete Wavelet Transform (DWT)

Discrete Wavelet Transform (DWT) (Chan et al., 2003; Chan and Fu, 1999; Popivanov and Miller, 2002) is another well-known time series dimensionality reduction technique. The key difference between DWT and DFT is that DWT allows capturing temporal information from time series Wu et al. (2000), in addition to frequency-domain information. The principle of the algorithm consists in recursively decomposing the time series into two components: approximation and details coefficients. The approximation coefficients represent the global properties of the time series and are given by means of an averaging operation, while details coefficients capture the local properties and are obtained using a differentiating operation (Li et al., 2016). Figure 2.9 illustrates an example of HWT with 1-Level (red) of a given time series (blue).

There exist several families of DWT, with Haar Wavelet Transform (HWT) being the oldest and the simplest ones (Fuad, 2011). Concretely, the HWT of a given time series $T = \{x_1, x_2, ..., x_n\}$ is calculated as follows (Li et al., 2016):

$$A_i = \frac{(x_{2i-1} + x_{2i})}{2} \tag{2.25}$$

$$D_i = \frac{(x_{2i-1} - x_{2i})}{2} \tag{2.26}$$

Where, A and D are the approximation and details coefficients of T, respectively and $1 \geq i \geq \frac{n}{2}$.

The HWT is a very efficient method where it has a linear time complexity $O(n)$, but it is only limited to time series of length equal to an integral power of two 2^p, $p \in \mathbb{N}$ (Ratanamahatana et al., 2005).

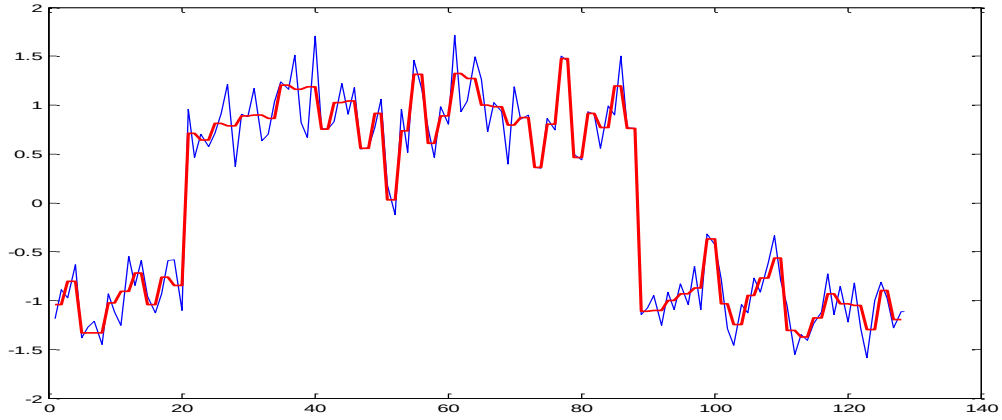

Figure 2.9: An example of a raw time series (blue curve) and its associated representation (red curve) using 1-level HWT method.

2.4.3 Singular Value Decomposition (SVD)

Singular Value Decomposition (SVD) (Korn et al., 1997) has similarities with DFT and DWT in that it uses a linear combination of base functions to represent the time series. On the other hand, SVD is a global transformation in the sense that it operates on a collection of time series (dataset) rather than a single series to perform the representation (Keogh et al., 2001a).This, in fact, can be both a disadvantage and an advantage. A disadvantage from a computational complexity point of view $O(mn^2)$, and advantage regarding the data reconstruction quality which is optimal (Keogh et al., 2001a; Ratanamahatana et al., 2005).

Formally, given a time series dataset D of m time series, each of length n, the SVD of D is a matrix $X = (m * n)$, given by:

$$X = UWV^T \tag{2.27}$$

Where, U is a $(m * n)$ orthogonal matrix, V is a $(n * n)$ orthogonal matrix, and W is a $(n * n)$ diagonal matrix that contains singular values. The time series can be only represented by the first k singular values (Fu, 2011).

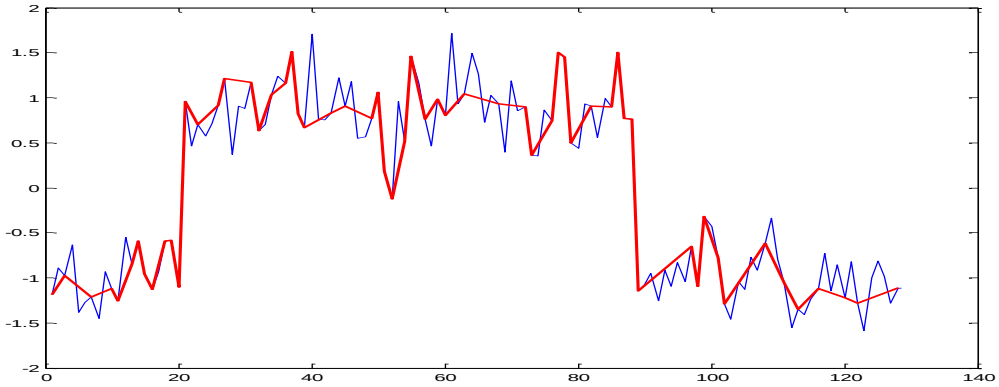

Figure 2.10: An example of a raw time series (blue curve) and its associated representation (red curve) using the PLA method.

2.4.4 Piecewise Linear Approximation (PLA)

The basic idea of Piecewise Linear Approximation (PLA) (Keogh et al., 2001c; Pavlidis and Horowitz, 1974) is to approximate the time series by a set of consecutive straight-line segments. Figure 2.10 shows an example of a raw time series (blue curve) and its corresponding PLA representation (red curve). The problem of time series approximation using PLA is also known as time series segmentation (Cassisi et al., 2012). There are generally three main approaches for time series segmentation: Sliding Windows, Top-Down, and Bottom-Up (Keogh et al., 2001c).

In the Sliding Windows approach, we start from the first data point and continue to extend the segment until a data point where the calculated error is greater than a predefined threshold value. We repeat this process until the last data point in the time series. For the Top-Down approach, the entire time series is first approximated by a single segment. Then, this segment is recursively partitioned until a stop condition is verified. Conversely, the Bottom-Up approach begins by representing the time series using $\frac{n}{2}$ segments, where n is the length of the time series. Next, it combines the two neighboring segments that are the most similar until a stop condition is true (Keogh et al., 2001c).

2.4.5 Piecewise Aggregate Approximation (PAA)

Keogh et al. (2001a) proposed Piecewise Aggregate Approximation (PAA), a simple but power-

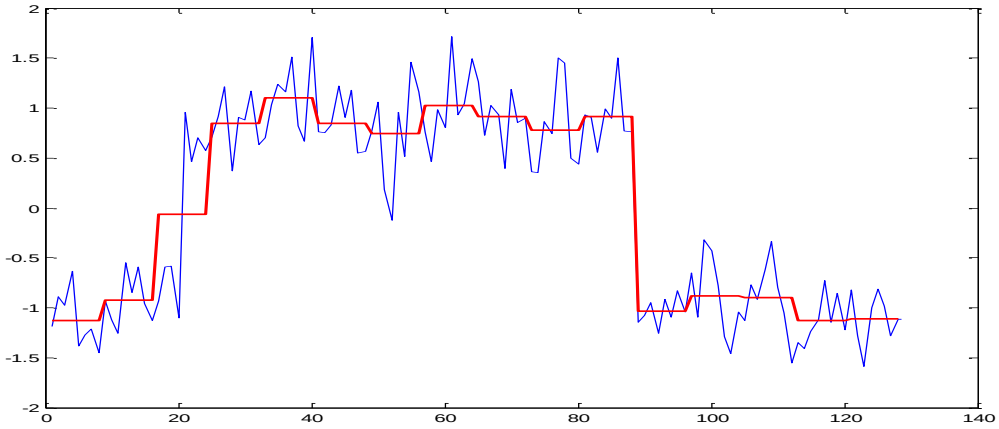

Figure 2.11: An example of a raw time series (blue curve) and its associated representation (red curve) using the PAA method.

ful dimensionality reduction technique for time series. PAA represents a time series by dividing it into equal-sized frames. Each frame is approximated by the average of its data points. An example of a raw time series (blue curve) and its associated PAA representation (red curve) is illustrated in Fig. 2.11.

Suppose we have a time series $T = \{x_1, x_2, ..., x_n\}$ of length n, the PAA of T is another time series $\bar{T} = \{\bar{x}_1, \bar{x}_2, ..., \bar{x}_N\}$ of reduced dimensionality $N << n$, where each element $\bar{x}_i$ is calculated by the following formula:

$$\bar{x}_i = \frac{n}{N} \sum_{j=\frac{n}{N}(i-1)+1}^{\frac{n}{N}i} x_j, i = 1...N \tag{2.28}$$

The computational complexity of PAA is linear with the length of the time series, but the fixed size of segments may be impractical in some situations where all time series patterns are not necessarily of equal length (Cassisi et al., 2012). Indeed, the same research team subsequently addressed this shortcoming and introduced an adaptive model named Adaptive Piecewise Constant Approximation (APCA) (Keogh et al., 2001b).

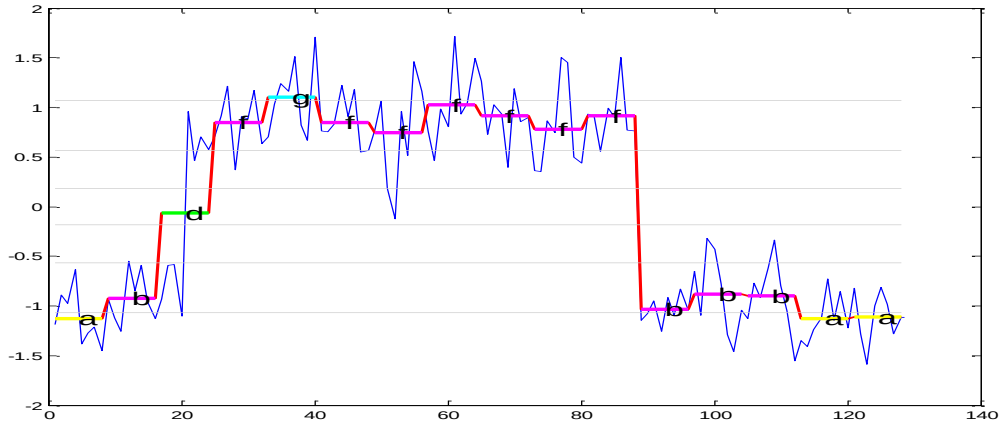

Figure 2.12: An example of a raw time series (blue curve) and its associated representation (red curve) using the SAX method (Lin et al., 2003; Lonardi and Patel, 2002).

2.4.6 Symbolic Aggregate Approximation (SAX)

Symbolic Aggregate Approximation (SAX) (Lin et al., 2003) differs from all of the above methods in that it provides a symbolic representation of the time series. It transforms the time series from its numerical space to a symbolic space of low dimensionality. To that end, two steps are necessary: dimensionality reduction and discretization.

For dimensionality reduction, SAX simply uses the PAA method. The second step consists in converting the PAA coefficients into alphabetical symbols with preferably equal probability. This latter is ensured by normalizing the time series, which is shown to have a Gaussian distribution. Given that, the breakpoints that produce α equal-sized regions under the Gaussian curve are then simply determined, and a symbol is assigned to each of them, where α represents the alphabet size. Finally, each PAA coefficient is discretized by projecting it onto its corresponding alphabet symbol. Figure 2.12 depicts an example of the SAX method where a raw time series is transformed to the word 'abdfgffffffbbbaa'.

2.5 Conclusion

In this chapter, we have provided technical background on time series data as well as the most common transformations associated with time series. We have also presented two fundamen-

tal problems in the context of time series data mining which are time series representations and similarity measures. In particular, we have introduced an in-depth review of time series similarity measures where several notions including problem formulation, challenges, and related work were discussed. As an important application for the use of time series similarity measures, we will focus on the time series classification task in the next chapter.

Chapter 3

Time Series Classification (TSC)

3.1 Introduction

This chapter is devoted to the Time Series Classification (TSC) problem. We start by introducing the fundamental notion of classification and briefly discuss the most relevant algorithms for the general classification. Following that, we focus on classification approaches, specific to time series. In that context, we define the particularity of the TSC problem, and we provide a taxonomy of available approaches.

3.2 Classification

Classification is a major area of interest within the field of data mining, pattern recognition and other fields of science and technology. It is regarded as a supervised learning method. Supervised learning means that each instance of the training dataset is associated with a class label. Classification is the process of predicting the class label of an unlabeled object using a model trained on a labeled dataset.

The classification task passes typically through two main phases, the learning phase and the classification phase (Han et al., 2011). In the first phase, a classification model is built from the training dataset. Once the model is constructed, it is then used to identify the class label of an

Algorithm *k*-Nearest Neighbors (*k*-NN)
Inputs: *D*: Training dataset, *k*: Number of NN, *d*: Distance measure, *Q*: unlabeled query **Output:** *C*: Class label of *Q* **Begin** 1: Compute $d(Q, T_i)$, the distance between *Q* and every instance in *D*. 2: Select $D_Q \in D$ the set of *k* closest training instance to *Q*. 3: Find *C*, the majority class label of the *k* class labels. 4: Return *C* **End.**

Figure 3.1: Pseudo-code of the *k*-NN algorithm.

unseen instance in the second phase.

Classification has long been a subject of much research across a wide range of scientific and industrial processes, including but not limited to Natural Language Processing (NLP) (e.g., automatically classifying news articles), cyber security (e.g., detecting credit card fraud), and image processing (e.g., tumors identification in brain scans).

3.3 Generic classification algorithms

There is a growing body of literature that focuses on the study of the classification task. In this section, we give a brief description of the main standard classifiers.

3.3.1 *k*-Nearest Neighbors (*k*-NN)

The *k*-Nearest Neighbors (*k*-NN) is one of the simplest and most intuitive classifiers (Cunningham and Delany, 2021). The *k*-NN is a lazy classifier because it does not require building explicit models. The principle of *k*-NN consists in finding the *k* closest (nearest neighbors) instances to a given unlabeled query from the training dataset. The 'closeness' is quantified by means of a distance measure, i.e., ED. The class label with the highest frequency of occurrence among the *k* retrieved classes is then assigned to the query. Despite its basic concept, *k*-NN is ranked among the top 10 data mining algorithms (Wu et al., 2008). Figure 3.1 provides a technical description of the *k*-NN classifier.

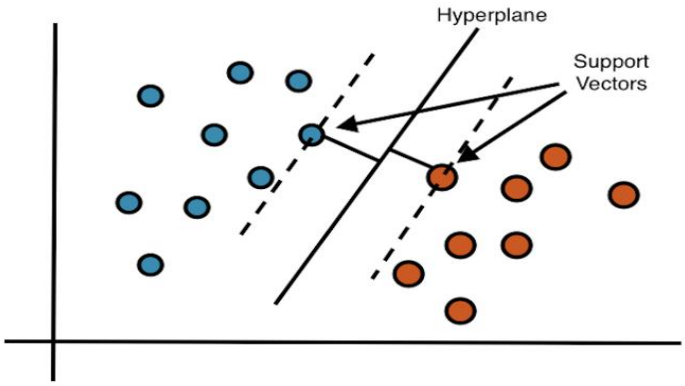

Figure 3.2: Basic idea of the SVM classification using two classes, red and blue circles.

The major advantage of the *k*-NN algorithm is that it is simple to understand and easy to implement. However, it is time demanding because it requires calculating the distances between the query and all instances in the training dataset. Another issue concerns the choice of the distance measure and the parameter *k*.

3.3.2 Support Vector Machines (SVMs)

Support Vector Machine (SVM) is another well-known classification algorithm. The basic idea of SVM is to find a hyperplane that optimally separates the different classes in the training dataset from each other; optimal in the sense that it maximizes the margin (distance) between all classes. In other words, the best hyperplane is the one in which the margin between it and the closest data point for each class is maximum. The closest data points are referred to as support vectors. Figure 3.2 illustrates a basic example of how SVM works using two classes, blue and red circles.

In real life, most data are not linearly separable (Aggarwal et al., 2015). In this case, SVM transforms the data into a higher dimension representation using kernel functions in which the data can be separable using a linear hyperplane. SVM still achieves good performance, even with high-dimensional data. It is also computationally and memory efficient, since it only works with support vectors in the classification phase (Li, 2018).

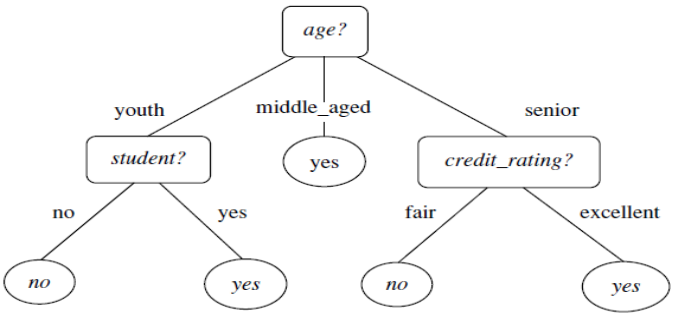

Figure 3.3: Typical Decision Tree, rectangle: internal nodes, circle: Leaf nodes and arc: branches (Han et al., 2011).

3.3.3 Decision Trees (DTs)

Decision tree (DT) is amongst the easiest and most interpretable classification algorithms (Han et al., 2011). It works by building classification models based on a set of decision rules. The decision rules are typically expressed in the form of the 'if-else-then' condition. As the name goes, the DT approach can be represented in a tree-like structure. The tree structure distinguishes three primary components: nodes, branches, and leaf nodes. For each internal node, a decision rule, known as a split criterion, is applied on an attribute in which the training dataset splits into two or more parts. These components are the tree's branches. Each leaf node, on the other hand, represents a class label. Figure 3.3 gives a graphical illustration of a typical decision tree, internal nodes, branches, and leaf nodes are represented by rectangles, arcs, and circles, respectively.

Due to the intuitive structure of DT, it is particularly easy to assimilate and interpret (Han et al., 2011). The DT approach has several representatives with CART (Classification and Regression Tree), ID3 (Iterative Dichotomiser 3), and C4.5 (Successor of ID3) being the most popular.

3.3.4 Artificial Neural Networks (ANNs)

Artificial Neural Network (ANN) is a classification approach that attempts to mimic the way the human brain and nervous system work (Han et al., 2011). The structure of ANN is conceptually similar to that of the biological neural network. Cells known as neurons are the basic unit of the ANN architecture. Each neuron has one or more inputs and produces a unique output. ANN is

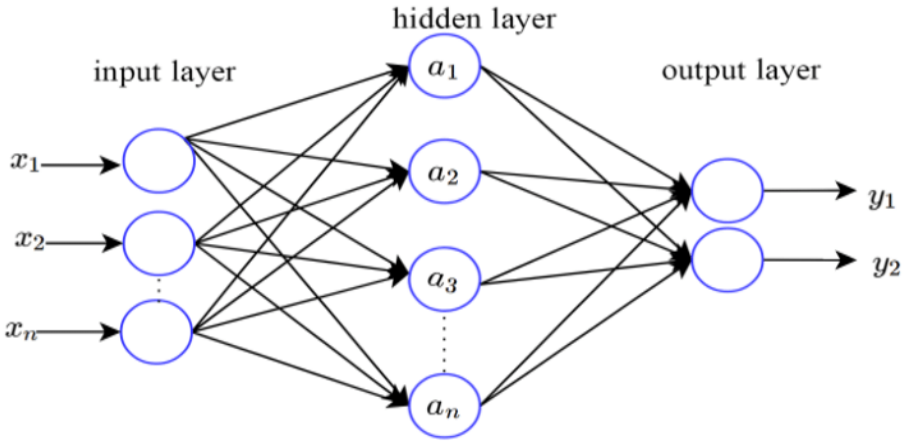

Figure 3.4: Typical ANN containing one input layer, one hidden layer, and an output layer with two classes: y1 and y2.

a collection of multiple interconnected neurons in which the output of a given neuron serves as the input to other neurons. The neurons are typically organized in one or possibly more layers. The role of the first layer (input layer) in ANN is to receive external data (e.g., image, time series), while the last layer (output layer) holds the class labels. The hidden layers are those that exist between the first and final layers. Figure 3.4 depicts a typical example of a basic ANN whit one hidden layer and two classes in the output layer.

Depending on the network topology (number of layers, number of neurons in each layer, how neurons and layers are linked to one another), there are many architectures of ANNs. However, Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) are the most common. RNNs are better suited to sequential data such as text, time series, and speech, whereas CNNs have shown good performance for image data (Bagnall et al., 2017; LeCun et al., 2015).

With the development of high-performance computers, a more complex ANN architecture composed of several layers was introduced. This model is known as Deep Neural Networks or simply Deep Learning (DL). Deep learning has achieved a lot of success and has attained state-of-the-art across a number of disciplines (Bagnall et al., 2017; LeCun et al., 2015). Nonetheless, deep learning has many drawbacks, including a high computational cost, particularly during the learning phase, low interpretability, a large number of parameters, and a requirement for a huge amount of data, which is not available in certain domain applications (Han et al., 2011).

3.3.5 Ensemble methods

Ensemble methods are a type of machine learning approaches, principally developed to increase classification accuracy (Zhang and Ma, 2012). In other words, the goal of ensemble methods is to improve classification accuracy over individual classifiers. The principle idea is to build multiple classifiers rather than simply one, and then combine their predictions to classify a new instance. Individual decisions are often aggregated via a voting mechanism. The key limitation, however, of this approach is its high temporal complexity.

Common types of ensemble methods are bootstrap aggregation (or bagging for simplicity) and boosting (Han et al., 2011). Bagging is the process of randomly dividing the training dataset into a collection of sub-sets. Each partition is utilized to construct a classification model. Bagging typically employs a single classifier that is built on many partitions. A voting mechanism is used to combine the output predictions of all models. The boosting method, on the other hand, runs several classifiers sequentially on the entire training dataset, with each classifier paying greater attention to samples misclassified by its predecessor.

3.4 Time series classification

Time series classification is a subfield of classification that is specifically concerned with time series data. TSC problems principally differ from other media classification problems in the sequence nature of time series, in which the attributes (data points) are temporally ordered (Bagnall et al., 2017). Due to the specific characteristics of time series, generic classification algorithms usually find difficulties in classifying such data (Abanda et al., 2019; Fawaz, 2020; Li, 2018).Thus, more advanced and dedicated algorithms are needed to deal with the TSC problem. The TSC issue is expressed as follows:

Time series classification: Given a labeled training dataset $D = \{(T_1, c_1), (T_2, c_2), ..., (T_M, c_M)\}$ of M time series T_i associated with their respective class labels c_i and an unlabeled time series Q, the goal of TSC is to assign Q to a correct class label $c_i \in \{1...C\}$, where C is the number of classes in D.

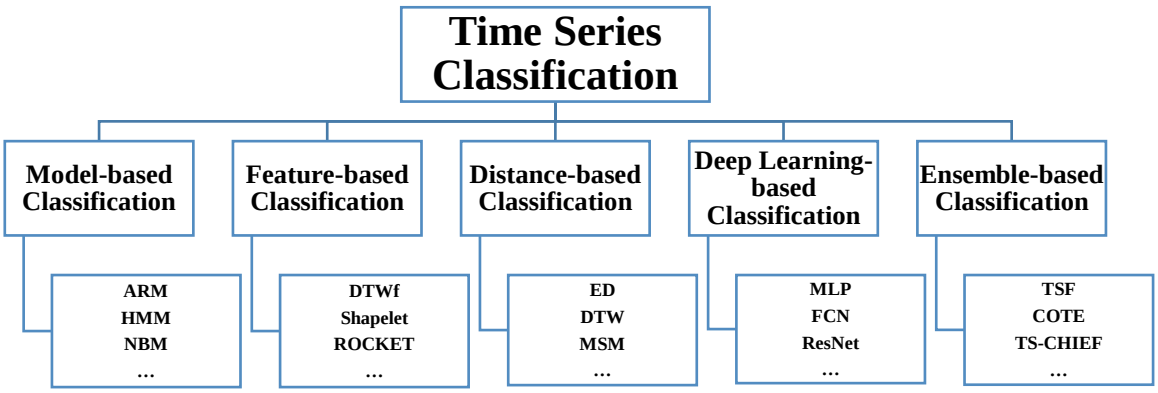

Figure 3.5: A taxonomy of TSC approaches.

TSC is becoming an increasingly important topic in the field of data mining (Esling and Agon, 2012). It has attracted a lot of interest with a wide range of real-world applications varying from medicine, cyber-security, finance, industry, among many others. In the literature, several methods have been proposed to solve the underlying problem. The TSC community has organized the TSC methods into three (03) major families named model-based classification, feature-based classification, and distance-based classification (Abanda et al., 2019; Xing et al., 2010). With the recent progress in the TSC field, we have updated this taxonomy by adding two more groups, which are ensemble-based classification and deep learning-based classification. Figure 3.5 gives a taxonomy with five groups of the TSC approaches.

3.4.1 Feature-based classification

Feature-based classifiers consist of two main steps: features extraction and classification. Features extraction involves transforming the original time series into another feature space while capturing their discriminant local and/or global properties. The extracted characteristics are usually arranged in a vector of features. After the features have been generated, they can be provided as inputs to any generic classifier (e.g., ANNs, DTs, SVMs) to perform the classification step. The goal of this type of classification is to take advantage of generic classification algorithms that typically operate on data in the form of feature vectors (vector-based classifiers) (Renard, 2017; Xing et al., 2010).

Nanopoulos et al. (2001) proposed a TSC algorithm based on statistic representations. The authors extracted the mean, standard deviation, skewness, and kurtosis, as global statistical features to represent time series. Then, they used a Multi-Layer Perceptron (MLP) neural network on the extracted features for classification. Instead of using global properties, Geurts (2001) developed a decision tree-based classification on the basis of local temporal patterns of time series. The author derives local patterns from discretized time series representations.

In (Fulcher and Jones, 2014), thousands of features of diverse nature including simple statistics, linear correlation, stationarity, information theoretic, and many more were extracted to represent time series. Based on the resulting features vector, a subset of the most suitable features is automatically and dynamically selected, using a greedy forward selection algorithm. Finally, the classification is performed using a linear discriminant classifier. In (Kate, 2016), the author proposed to exploit the strength of DTW to create new features. Specifically, each time series is represented as a function of the DTW distance between it and all of the time series in the training dataset. The author applies SVMs as the classification algorithm, but it assumes that any generic classification algorithm can be used.

Another feature-based classifier was introduced by Zhao and Itti (2015). Depending on key feature points, time series are segmented into local subsequences. Then, subsequences are represented by Histogram of Oriented Gradients (HOG-1D) and DTW-Multidimensional Scaling (DTW-MDS) descriptors. Next, each time series is encoded by a Fisher Vector (FV). At last, the classification is done using linear kernel SVM.

On the other hand, time series shapelets have gained more popularity during the last decade. Shapelets are time series subsequences that best distinguish the different classes in a training dataset (Ye and Keogh, 2011). Moreover, shapelets are discriminatory local, phase-independent subsequences that can be located anywhere in a time series. Originally, Shapalets was first proposed by Ye and Keogh (2011). Since then, shapelets have attracted considerable interest from the scientific community and hence, many shapelet-based algorithms have been proposed over the years (Grabocka et al., 2016; Hills et al., 2014; Zakaria et al., 2012).

In the original work (Ye and Keogh, 2011), Ye and Keogh (2011) used a shapelet-based decision tree classifier. All possible candidates are generated from the training dataset. At each node, the shapelet that best splits the data is determined associated with its optimal split point. This process is recursively repeated until only pure leaves remain. Finally, the trained model is employed to classify new unlabeled time series. According to (Ye and Keogh, 2011), Shapelet-based classification achieves competitive accuracy with good interpretability. However, the main drawback of this paradigm is that the shapelet discovery process requires significant computing power. It has $O(n^2m^2)$ time complexity, where m and n are the size of the training dataset and the length of the time series, respectively. Consequently, most subsequent research on shapelet-based classification has focused on speeding up the shapelet search of the original algorithm (Chang et al., 2012; Gordon et al., 2012; He et al., 2012; Hills, 2014). For instance, Logical Shapelets (Mueen et al., 2011), Fast Shapelets (Rakthanmanon and Keogh, 2013), and Learned Shapelets (Grabocka et al., 2016) are the most popular alternatives.

A recent work (Dempster et al., 2020) introduced a very fast, highly accurate, and more scalable TSC algorithm called Random Convolutional Kernel Transform (ROCKET). ROCKET uses a huge diversity of random convolutional kernels to generate features from raw time series. Random convolutional kernels in combination are capable of capturing a wide variety of features. The so-obtained features are used in conjunction with a linear classifier. A ridge regression classifier is used in the case where there is less training time series than features; otherwise, logistic regression will be used. Currently, ROCKET is reported to be amongst the most accurate TSC algorithms with high efficiency (Middlehurst et al., 2021).

3.4.2 Model-based classification

In model-based classification, it is assumed that time series within a class are generated by the same underlying model or probability distribution (Liao, 2005; Xing et al., 2010). Otherwise, similar time series usually produce the same model. Hence, a new time series will be output with the class label whose model best fits that series. A review of the TSC literature shows that model-based classifiers have received less attention from the scientific community than

other categories. The reason relies on the fact that they are not competitive for the TSC task (Bagnall et al., 2017). Examples of the most well-known approaches are Auto-Regressive Models (ARM) (Bagnall and Janacek, 2014; Corduas and Piccolo, 2008), Hidden Markov Models (HMM) (Antonucci et al., 2015; Smyth, 1996), and Naïve Bays Models (NBM) (Norvig and Intelligence, 2002).

3.4.3 Deep learning-based classification

In recent years, deep learning (LeCun et al., 2015) has become popular due to its high performance where it has reached state-of-the-art in many application domains such as computer vision, NLP, and speech recognition (Fawaz et al., 2019; Fawaz, 2020). That has motivated the time series community to develop deep learning models for TSC problems. While there are several deep learning approaches for TSC, we only survey here the most popular and we refer the interested reader to an excellent review (Fawaz et al., 2019; Fawaz, 2020) for more examples and details.

Wang et al. (2017) proposed three simple end-to-end deep neural networks approaches for TSC. The first one is a deep Multi-Layer Perceptron (MLP). MLP has four layers where three are hidden layers and one for the last output layer. Hidden layers are fully connected and each of them contains 500 neurons. In this approach, the authors use Rectified Linear Unit (ReLU) and dropout operation as the activation function and a regularization technique, respectively. A softmax classifier is used as the last layer and has the same number of neurons as classes in the training dataset.

The second approach is a Fully Convolutional Network (FCN). This approach contains three convolutional blocks, each of which comprises three successive components: a convolutional layer, a batch normalization layer, and a ReLU activation layer. The first, second, and third convolutional layers contain 128, 256, and 128 filters, respectively. Outputs of the last convolution block are averaged using a Global Averaging Pooling (GAP) layer. As in the first approach, the final layer is also a softmax classifier.

The third proposed approach is a deep Residual Network (ResNet). ResNet has the deepest architecture of deep learning approaches for TSC. It is composed of three basic residual blocks where each is in turn a convolution block. Similar to FCN, ResNet architecture has as outputs a GAP layer followed by a softmax classifier. ResNet thus can be seen as a deeper version of FCN. According to Wang et al. (2017), these three approaches offer a straightforward solution for real-world application and an excellent baseline for further research.

In (Serrà et al., 2018), the authors developed a neural network encoder approach. This encoder is inspired by FCN where is composed of three convolution blocks of 128, 256, and 512 filters, respectively. Each convolution block contains a sequence of a 1-dimensional convolution, an instance normalization layer, a Parametric Rectified Linear Unit (PReLU) activation function, and a dropout layer. The first and second convolution blocks are each followed by a 2-factor max-pooling layer. Encoder employs an attention layer rather than a GAP layer followed by a traditional softmax classifier.

Multi-scale Convolutional Neural Network (MCNN), another end-to-end deep learning approach was introduced by Cui et al. (2016). In MCNN, the authors integrate the processes of representation and classification of time series into a unified framework. For time series representation, the input time series is sampled into subsequences using a window sliding technique. Each subsequence undergoes three distinct transformations, which are identity mapping, down-sampling, and smoothing. After that, the three transformed subsequences will be fed to three parallel convolutions with a max-pooling procedure for each convolution. The output results are then concatenated and inputted sequentially into a convolutional layer, a fully connected layer, and a softmax classifier.

Motivated by the great achievement of CNN for image classification, Zhao et al. (2017) proposed a novel CNN-based approach for both univariate and multivariate TSC. The proposed Time-CNN is composed of two main modules: feature engineering and classification. The first module is used for generating deep features of time series, while the second is used for classification. The features engineering module contains two convolutional layers; each is followed by a pooling operation. The generated features are fed to an MLP for classification. In Time-CNN,

a sigmoid is employed as the activation function.

Unlike all of the previously discussed approaches, which are CNN-based architectures, Tanisaro and Heidemann (2016) proposed a Recurrent Neural Network (RNN) based approach called Time Warping Invariant Echo State Network (TWIESN). TWIESN is based on a reservoir, which is a sparsely connected random RNN. The reservoir contains several neurons with both static and random weights. Each neuron within the reservoir produces a non-linear transformation on the input time series, and the outcomes are then utilized to train a ridge classifier.

More recently, (Fawaz, 2020) introduced InceptionTime, a DL-based ensemble. InceptionTime is a collection of several homogenous CNN models. Specifically, InceptionTime combines five residual network models. All of the component models have the same architecture. However, they differ in the initial weight values, which are randomly assigned. To date, InceptionTime is reported to be the most accurate deep learning approach for TSC (Middlehurst et al., 2021).

3.4.4 Ensemble-based classification

Because of the high accuracy of ensemble methods, the focus of recent research on TSC has been oriented toward this direction (Bagnall et al., 2017).

Time Series Forest (TSF) (Deng et al., 2013): Time Series Forest (TSF) is a tree-based ensemble composed of 500 time series DT classifiers. TSF segments all the time series in the training dataset into intervals and extracts local statistical features such as mean, standard deviation, and slope from each interval. The newly obtained features dataset is then randomly divided into many sub-dataset. Each of the set of DT classifiers is trained on a single sub-dataset. The classification phase in TSF is decided by a majority vote system. The particularity of TSF is that it uses both entropy gain and distance measure (abbreviated as entrance) to assess the splitting of the training dataset.

Time Series Bag of Features (TSBF) (Baydogan et al., 2013): Researchers from the same team of TSF proposed another TSC algorithm based on an ensemble of DTs. Time Series Bag of Features (TSBF) can be considered as a continuation of TSF. TSBF is composed of four

principal steps. First, TSBF arbitrarily selects some subsequences from the raw time series and divides them into intervals. Each interval is represented using mean, standard deviation, and slope characteristics. All of the characteristic intervals of a subsequence are regrouped into a single feature vector. The class label of the original time series is attributed to the transformed subsequence. All time series in the training dataset undergo the same procedure, and hence a new dataset of transformed subsequences is constructed. Second, a random forest is trained on the newly generated dataset and produces Class Probability Estimates (CBEs) for each subsequence. Third, CBEs from the same time series are used to form a CBE histogram. Finally, a random forest model is built on the basis of the CBE histograms. Later, a new TSC algorithm was developed by the same research team called Learned Pattern Similarity (LPS) (Baydogan and Runger, 2016), which is technically similar to TSF and TSBF.

Shapelet Ensemble (SE) (Bagnall et al., 2015; Bostrom and Bagnall, 2015; Hills, 2014; Hills et al., 2014): Shapelet Ensemble (SE) is a heterogeneous ensemble of Shapelet Transform (ST)-based classifiers (Hills et al., 2014). SE relies on the best k shapelets extracted from the training dataset to create another dataset of shapelet transformations. It computes the distance between each time series and all the k shapelets. The main goal of the shapelet transformation process is to allow the use of shapletes in conjunction with a wide variety of classifiers. In SE, a set of standard classifiers, 1-NN, C4.5, Naïve Bayes, Bayesian Network, Random Forest, Rotation Forest, and SVMs, are built, independently based on the shapelet-transformed dataset. For classifying a new time series, individual predictions of classifiers are combined using a weighted vote scheme.

Ensembles of Elastic Distance Measures (EE) (Lines and Bagnall, 2015): With the great success of time series elastic measures in the context of TSC, Lines and Bagnall (2015) suggested combining 11 different elastic distance measures into a single ensemble method to further improve the TSC accuracy. The distance measures considered are ED, DTW, Constrained DTW, Derivative DTW (DDTW) (Keogh and Pazzani, 2001), Constrained DDTW (Lines and Bagnall, 2015), Weighted DTW (WDTW) (Jeong et al., 2011), WDDTW (Jeong et al., 2011), LCSS (Vlachos et al., 2003), EDR (Chen et al., 2005), TWED (Marteau, 2008), and MSM

(Stefan et al., 2012). Each of these distances is coupled with the 1-NN classifier, resulting in a meta-classifier of 11 1-NN classifiers. EE attributes a weight parameter to each of its components based on their accuracies in the training step. Recently, a more efficient version of EE is proposed, called FastEE (Tan et al., 2020).

Collective of Transformation-Based Ensembles (COTE) (Bagnall et al., 2015): Bagnall et al. (2015) proposed the Collective of Transformation-Based Ensembles (COTE), another ensemble-based classifier. The core of the COTE method lies in the use of different transformation domains. Their motivation stems from the idea that transforming time series can help achieve more accurate TSC algorithms (Bagnall et al., 2012). The representations considered include time-domain (raw time series), frequency, change, and shapelet. Each representation is combined with a standard set of classifiers. As a result, 35 classifiers are constructed under COTE. The authors investigate different techniques to combine these classifiers; however, they were found that Flat-COTE is the most accurate combination. Bagnall et al. (2017) carried out extensive experimental comparisons between 18 state-of-the-art TSC algorithms on 85 UCR datasets (Chen et al., 2015a) and found that Flat-COTE is significantly more accurate.

Hierarchical Vote Collective of Transformation-Based Ensembles (HIVE-COTE) (Lines et al., 2016, 2018): Hierarchical Vote of Transformation-Based Ensembles (HIVE-COTE or HC for short) is, in a way, the successor of Flat-COTE with more improvement in classification accuracy. The authors have made significant revisions to Flat-COTE. They propose and integrate two new ensemble classifiers, Random Interval Spectral Ensemble (RISE) and Shapelet Transform with Heterogeneous Ensemble of Standard Classification Algorithms (ST-HESCA).They also include two more data representations: interval and dictionary transformations. Finally, they provide a new hierarchical structure based on a probabilistic voting method.

HIVE-COTE method is a heterogonous meta-ensemble of five ensembles: EE, ST-HESCA, BOSS Ensemble (Schäfer, 2015a), TSF, and Random Interval Spectral Ensemble. HIVE-COTE has been demonstrated to be significantly more accurate than its predecessor Flat-COTE (Lines et al., 2016). However, in terms of computational complexity, it is quite time-consuming, especially for large datasets. To address this flaw, the method has been updated to HIVE-

COTE v1.0 (HC1), which is more scalable and orders of magnitude faster, with statistically equal accuracy (Bagnall et al., 2020).

More recently, Middlehurst et al. (2021) offered HIVE-COTE v2.0 (HC2), another upgraded version of HC. HIVE-COTE v2.0 principally differs from HIVE-COTE v1.0 in that some elements have been eliminated, while others have been added. HIVE-COTE v2.0 has four component ensembles: Temporal Dictionary Ensemble (TDE) (Middlehurst et al., 2020b), Diverse Representation Canonical Interval Forest (DrCIF) (Middlehurst et al., 2020a), ROCKET-based ensemble called Arsenal, and the most recent version of Shapelet Ensemble (SE) (Bostrom and Bagnall, 2015).

HIVE-COTE v2.0 was also extended to include multivariate time series classification. To evaluate the performance of HIVE-COTE v2.0, extensive experiments were done on 112 univariate and 26 multivariate datasets from the UCR archive (Dau et al., 2018, 2019) and UEA archive (Bagnall et al., 2018), respectively. HIVE-CTE v2.0 was shown to be much more accurate than all examined TSC algorithms, and it is currently the new state-of-the-art TSC algorithm in terms of classification accuracy.

Time Series Combination of Heterogeneous and Integrated Embedding Forest (TS-CHIEF) (Shifaz et al., 2020): Lately, Shifaz et al. (2020) developed Time Series Combination of Heterogeneous and Integrated Embedding Forest (TS-CHIEF) for TSC. The ensemble approach used by TS-CHIEF combines multiple DT classifiers. TS-CHIEF utilizes a variety of TSC methods (BOSS, EE, and RISE) on each DT node at random to split data. These methods are thus, exploited as splitting criteria in TS-CHIEF. Classification of a new unlabeled time series is based on a majority vote scheme. TS-CHIEF is, in reality, a more advanced variant of the Proximity Forest (PF) algorithm (Lucas et al., 2019). In terms of classification accuracy, TS-CHIEF has been shown to compete with the state-of-the-art HIVE-COTE, but it consumes significantly less CPU time and has more scalability (Shifaz et al., 2020).

3.4.5 Distance-based classification

The vast majority of previous research on TSC has focused on distance-based classifiers (Bagnall et al., 2017). The choice of distance measures is the essence of this form of classification. In other words, similarity measurements are critical to the accuracy of these classifiers (Serrà and Arcos, 2014). SVMs and k-NN are the leading distance-based classifiers.

In particular, the 1-NN classifier has attracted much attention from researchers, and it has been widely utilized not only within this category but also throughout TSC in general due to its simplicity and, more importantly, its excellent accuracy (Abanda et al., 2019; Jiang, 2020).There is empirical evidence that the 1-NN classifier, when combined with a proper distance measure, is more accurate than most existing approaches (Bagnall et al., 2017; Bagnall and Lines, 2014; Xi et al., 2006). The high effectiveness of 1-NN lies in the fact that the time series distance measures consider the sequential nature of time series and can handle various distortions (Abanda et al., 2019).

Because of its fundamental importance, similarity measures have been the subject of intense research within TSC, and much of the work has concentrated on developing new similarity measures for the 1-NN classifier (Abanda et al., 2019; Bagnall et al., 2012; Bagnall and Lines, 2014; Lines et al., 2018; Lines and Bagnall, 2015; Wang et al., 2013).

A plethora of time-series similarity measures has been proposed in the TSC literature. The most straightforward distance to assess the similarity between time series is the ED (Agrawal et al., 1993; Faloutsos et al., 1994). ED is one of the earliest time series similarity measures. It is simple to understand, easy to implement, and very efficient since it has a linear time complexity with respect to the length of the time series. However, ED requires that the inputs time series be of equal lengths (Ratanamahatana and Keogh, 2005). It is also very sensitive to noise and cannot tolerate most distortions (Keogh and Ratanamahatana, 2005).

To overcome the ED's shortcomings, the DTW was proposed (Berndt and Clifford, 1994; Sakoe and Chiba, 1978). DTW works by stretching or shrinking time series in order to find an optimal alignment between sequences. For more than two decades, DTW has been the leading

measurement for TSC (Bagnall et al., 2017; Lines and Bagnall, 2015; Wang et al., 2013). It has been effectively applied in many different domains varying from robotics, bioinformatics, meteorology, and others (Mueen and Keogh, 2016). The key success of DTW lies in its robustness to local misalignments along the time axis between time series. Over the years, several improved versions of DTW have been developed (Al-Naymat et al., 2012; Boulnemour and Boucheham, 2018; Keogh and Ratanamahatana, 2005; Salvador and Chan, 2007; Silva and Batista, 2016). Even though the high effectiveness of DTW, it still suffers when matching time series containing noise and outliers (Chen et al., 2005).

In this context, LCSS can be a good alternative (Vlachos et al., 2003). LCSS is broadly renowned for being invariant to noise and outliers (Soleimani and Abessi, 2020). The basic idea is to exclude some data points (noise and outliers) from the matching process. LCSS searches for the longest common subsequence that optimizes the matching between two series. Similarly, Chen et al. (2005) proposed another edit-based distance for time series called Edit Distance on Real sequence (EDR). Unlike LCSS, EDR considers the gaps between two matched subsequences. According to its authors, the EDR is particularly more robust to noise than the LCSS. A common drawback of DTW, LCSS, and EDR is that they are non-metrics (i.e., they do not satisfy the triangle inequality) and therefore are not suitable for indexing in time series databases (Chen and Ng, 2004; Stefan et al., 2012).

In this regard, three distance metrics: ERP (Chen and Ng, 2004), TWED (Marteau, 2008), and MSM (Stefan et al., 2012) were proposed. ERP is based on two distance families:Lp-norms and edit distance.The idea behind this combination is to take advantage of the metric property of Lp-norm distance as well as the local time shift invariance of edit distance. Therefore, ERP is an edit-based distance that obeys the triangular inequality. TWED is another common edit-based metric distance. In contrast to traditional edit distances, TWED takes into consideration the timestamp differences between the compared time series. TWED defines a stiffness parameter to control the elasticity of the time series along the time axis. In TWED, the similarity between two time series represents the minimum number of edit operations required to convert one time series to another. For the MSM metric, three operations (Move, Split, and Merge) are

defined and used in order to transform one time series to another. MSM looks for a sequence of operations with the lowest possible cost.

All of the aforementioned distance measures operate on raw time series, which may affect their performance, especially when dealing with high-dimensional time series. One solution to this issue is to first reduce the dimensionality of the time series and then perform a comparison based on the new representations. This category of measurements is known as feature-based distances.

Lin et al. (2012a) proposed Bag of Patterns (BOP), an efficient time series similarity measure based on the bag-of-words representation, an approach widely used in text mining and information retrieval domains (Lin et al., 2012a). More specifically, BOP uses a sliding window technique to extract subsequences from the original time series. Each extracted subsequence is discretized and converted into a symbolic word via the SAX method. BOP after that creates a histogram to calculate the frequency of the words. Once the histograms of the two time series to be compared are constructed, the authors assume that any distance measure can be used to assess the similarity between series.

Symbolic Aggregate Approximation-Vector Space Model (SAX-VSM) (Senin and Malinchik, 2013) also represents time series using SAX and Bag-of-Words approaches. However, instead of forming histograms for each time series, SAX-VSM creates histograms across classes. SAX-VSM applies VSM to generate weight coefficients for each class. The cosine similarity measure is employed by the authors for comparison. Schäfer (2015a) developed another common feature-based distance measure denoted as Bag-of-SFA-Symbols (BOSS) (Schäfer, 2015a). BOSS has similarities to BOP and SAX-VSM measures. It also uses a sliding window to extract subsequences from raw time series. Nevertheless, the key difference is that BOSS employs the Symbolic Fourier Approximation (SFA) method (Schäfer and Högqvist, 2012) rather than the SAX method to convert subsequences into symbolic words. Based on the SFA words histograms, the similarity is computed by means of an adapted version of ED.

3.5 UCR time series classification archive

The University of California-Riverside (UCR) time series archive (Chen et al., 2015a; Dau et al., 2018) offers a very valuable repository, especially for the TSC community. It provides a large variety of labeled time series datasets. The archive is organized in a very simple file format, which enormously facilitates its use and manipulation. Most importantly, the archive is freely available to everyone without any considerable restrictions. Based on that, the UCR archive has been widely used as a benchmark for testing and evaluating time series data mining algorithms (Bagnall et al., 2017; Dau et al., 2019). Besides, the UCR repository has heavily helped in improving the field of TSC (Bagnall et al., 2017).

Keogh (2002) originally introduced the UCR archive in 2002, with sixteen datasets. Since then, the archive has not ceased expanding. The latest version of the UCR archive was created in 2018 (UCR18) and contains a collection of 128 datasets (Dau et al., 2018, 2019). It is hence the largest well-known repository for TSC. The datasets in the UCR archive include real-life, synthetic, and generic time series coming from a large variety of application domains, ranging from medicine, meteorology, electricity, computer vision, etc. Each dataset has two standard partitions, training and a test set. Datasets differ from each other in terms of length of time series, size of the train/test set (varies from 16 to 16800 samples), and the number of classes (between 2 and 60).Table 3.1 shows characteristics including name, type, size of training and test set, number of classes, and length of 128 datasets from the UCR time series classification/clustering archive.

Table 3.1: Characteristics of 128 datasets from the UCR time series classification/clustering archive (Dau et al., 2018, 2019).

ID	Type	Name	Train	Test	Class	Length
1	Image	Adiac	390	391	37	176
2	Image	ArrowHead	36	175	3	251
3	Spectro	Beef	30	30	5	470
4	Image	BeetleFly	20	20	2	512
5	Image	BirdChicken	20	20	2	512
6	Sensor	Car	60	60	4	577
7	Simulated	CBF	30	900	3	128
8	Sensor	ChlorineConcentration	467	3840	3	166
9	Sensor	CinCECGTorso	40	1380	4	1639
10	Spectro	Coffee	28	28	2	286
11	Device	Computers	250	250	2	720
12	Motion	CricketX	390	390	12	300
13	Motion	CricketY	390	390	12	300

Continued on next page

ID	Type	Name	Train	Test	Class	Length
14	Motion	CricketZ	390	390	12	300
15	Image	DiatomSizeReduction	16	306	4	345
16	Image	DistalPhalanxOutlineAgeGroup	400	139	3	80
17	Image	DistalPhalanxOutlineCorrect	600	276	2	80
18	Image	DistalPhalanxTW	400	139	6	80
19	Sensor	Earthquakes	322	139	2	512
20	ECG	ECG200	100	100	2	96
21	ECG	ECG5000	500	4500	5	140
22	ECG	ECGFiveDays	23	861	2	136
23	Device	ElectricDevices	8926	7711	7	96
24	Image	FaceAll	560	1690	14	131
25	Image	FaceFour	24	88	4	350
26	Image	FacesUCR	200	2050	14	131
27	Image	FiftyWords	450	455	50	270
28	Image	Fish	175	175	7	463
29	Sensor	FordA	3601	1320	2	500
30	Sensor	FordB	3636	810	2	500
31	Motion	GunPoint	50	150	2	150
32	Spectro	Ham	109	105	2	431
33	Image	HandOutlines	1000	370	2	2709
34	Motion	Haptics	155	308	5	1092
35	Image	Herring	64	64	2	512
36	Motion	InlineSkate	100	550	7	1882
37	Sensor	InsectWingbeatSound	220	1980	11	256
38	Sensor	ItalyPowerDemand	67	1029	2	24
39	Device	LargeKitchenAppliances	375	375	3	720
40	Sensor	Lightning2	60	61	2	637
41	Sensor	Lightning7	70	73	7	319
42	Simulated	Mallat	55	2345	8	1024
43	Spectro	Meat	60	60	3	448
44	Image	MedicalImages	381	760	10	99
45	Image	MiddlePhalanxOutlineAgeGroup	400	154	3	80
46	Image	MiddlePhalanxOutlineCorrect	600	291	2	80
47	Image	MiddlePhalanxTW	399	154	6	80
48	Sensor	MoteStrain	20	1252	2	84
49	ECG	NonInvasiveFetalECGThorax1	1800	1965	42	750
50	ECG	NonInvasiveFetalECGThorax2	1800	1965	42	750
51	Spectro	OliveOil	30	30	4	570
52	Image	OSULeaf	200	242	6	427
53	Image	PhalangesOutlinesCorrect	1800	858	2	80
54	Sensor	Phoneme	214	1896	39	1024
55	Sensor	Plane	105	105	7	144
56	Image	ProximalPhalanxOutlineAgeGroup	400	205	3	80
57	Image	ProximalPhalanxOutlineCorrect	600	291	2	80
58	Image	ProximalPhalanxTW	400	205	6	80
59	Device	RefrigerationDevices	375	375	3	720
60	Device	ScreenType	375	375	3	720
61	Simulated	ShapeletSim	20	180	2	500
62	Image	ShapesAll	600	600	60	512
63	Device	SmallKitchenAppliances	375	375	3	720
64	Sensor	SonyAIBORobotSurface1	20	601	2	70
65	Sensor	SonyAIBORobotSurface2	27	953	2	65
66	Sensor	StarLightCurves	1000	8236	3	1024
67	Spectro	Strawberry	613	370	2	235
68	Image	SwedishLeaf	500	625	15	128
69	Image	Symbols	25	995	6	398
70	Simulated	SyntheticControl	300	300	6	60
71	Motion	ToeSegmentation1	40	228	2	277
72	Motion	ToeSegmentation2	36	130	2	343
73	Sensor	Trace	100	100	4	275
74	ECG	TwoLeadECG	23	1139	2	82
75	Simulated	TwoPatterns	1000	4000	4	128
76	Motion	UWaveGestureLibraryAll	896	3582	8	945
77	Motion	UWaveGestureLibraryX	896	3582	8	315
78	Motion	UWaveGestureLibraryY	896	3582	8	315
79	Motion	UWaveGestureLibraryZ	896	3582	8	315
80	Sensor	Wafer	1000	6164	2	152
81	Spectro	Wine	57	54	2	234

Continued on next page

ID	Type	Name	Train	Test	Class	Length
82	Image	WordSynonyms	267	638	25	270
83	Motion	Worms	181	77	5	900
84	Motion	WormsTwoClass	181	77	2	900
85	Image	Yoga	300	3000	2	426
86	Device	ACSF1	100	100	10	1460
87	Sensor	AllGestureWiimoteX	300	700	10	Vary
88	Sensor	AllGestureWiimoteY	300	700	10	Vary
89	Sensor	AllGestureWiimoteZ	300	700	10	Vary
90	Simulated	BME	30	150	3	128
91	Traffic	Chinatown	20	343	2	24
92	Image	Crop	7200	16800	24	46
93	Sensor	DodgerLoopDay	78	80	7	288
94	Sensor	DodgerLoopGame	20	138	2	288
95	Sensor	DodgerLoopWeekend	20	138	2	288
96	EOG	EOGHorizontalSignal	362	362	12	1250
97	EOG	EOGVerticalSignal	362	362	12	1250
98	Spectro	EthanolLevel	504	500	4	1751
99	Sensor	FreezerRegularTrain	150	2850	2	301
100	Sensor	FreezerSmallTrain	28	2850	2	301
101	HRM	Fungi	18	186	18	201
102	Trajectory	GestureMidAirD1	208	130	26	Vary
103	Trajectory	GestureMidAirD2	208	130	26	Vary
104	Trajectory	GestureMidAirD3	208	130	26	Vary
105	Sensor	GesturePebbleZ1	132	172	6	Vary
106	Sensor	GesturePebbleZ2	146	158	6	Vary
107	Motion	GunPointAgeSpan	135	316	2	150
108	Motion	GunPointMaleVersusFemale	135	316	2	150
109	Motion	GunPointOldVersusYoung	136	315	2	150
110	Device	HouseTwenty	40	119	2	2000
111	EPG	InsectEPGRegularTrain	62	249	3	601
112	EPG	InsectEPGSmallTrain	17	249	3	601
113	Traffic	MelbournePedestrian	1194	2439	10	24
114	Image	MixedShapesRegularTrain	500	2425	5	1024
115	Image	MixedShapesSmallTrain	100	2425	5	1024
116	Sensor	PickupGestureWiimoteZ	50	50	10	Vary
117	Hemodynamics	PigAirwayPressure	104	208	52	2000
118	Hemodynamics	PigArtPressure	104	208	52	2000
119	Hemodynamics	PigCVP	104	208	52	2000
120	Device	PLAID	537	537	11	Vary
121	Power	PowerCons	180	180	2	144
122	Spectrum	Rock	20	50	4	2844
123	Spectrum	SemgHandGenderCh2	300	600	2	1500
124	Spectrum	SemgHandMovementCh2	450	450	6	1500
125	Spectrum	SemgHandSubjectCh2	450	450	5	1500
126	Sensor	ShakeGestureWiimoteZ	50	50	10	Vary
127	Simulated	SmoothSubspace	150	150	3	15
128	Simulated	UMD	36	144	3	150

3.6 Conclusion

In this chapter, we presented an overview on classification, an important task in data mining and other domains and fields of science and technology. We provided general background and surveyed state-of-the-art generic classification algorithms. We concentrated on classification in the context of time series by addressing various aspects related to TSC. A taxonomy of the TSC algorithms that exist in the literature has been given, which includes five major categories named model-based, feature-based, distance-based, ensemble-based, and deep learning-based

classifiers. For each, we reviewed the most well-known algorithms. After highlighting the literature related to this thesis, we will present the personal contributions in the following chapters.

Part III

Personal Contributions

Chapter 4

An empirical comparison study of DTW's variants for time series classification (Lahreche and Boucheham, 2021a)

4.1 Introduction

In time-series data mining, it is largely admitted that DTW is the best similarity measure for a large variety of tasks (Silva et al., 2018). In particular, it has been found, through extensive experimentation, that DTW, when coupled with the 1-NN classifier, achieves very high classification accuracy and is virtually unbeatable (Abanda et al., 2019; Bagnall et al., 2017; Bagnall and Lines, 2014; Mueen and Keogh, 2016; Wang et al., 2013; Xi et al., 2006). Nevertheless, DTW still has some interesting and relevant problems to be addressed. For example, it suffers from a pathological alignment problem (Keogh and Pazzani, 2001). This problem is defined when a single data point from a time series aligns with multiple data points (subsequence) from another time series, resulting in an unintuitive alignment. Figure 4.1 shows an example of the pathological alignment problem produced by the DTW measure.

To overcome the drawbacks of DTW, researchers have investigated a large variety of ap-

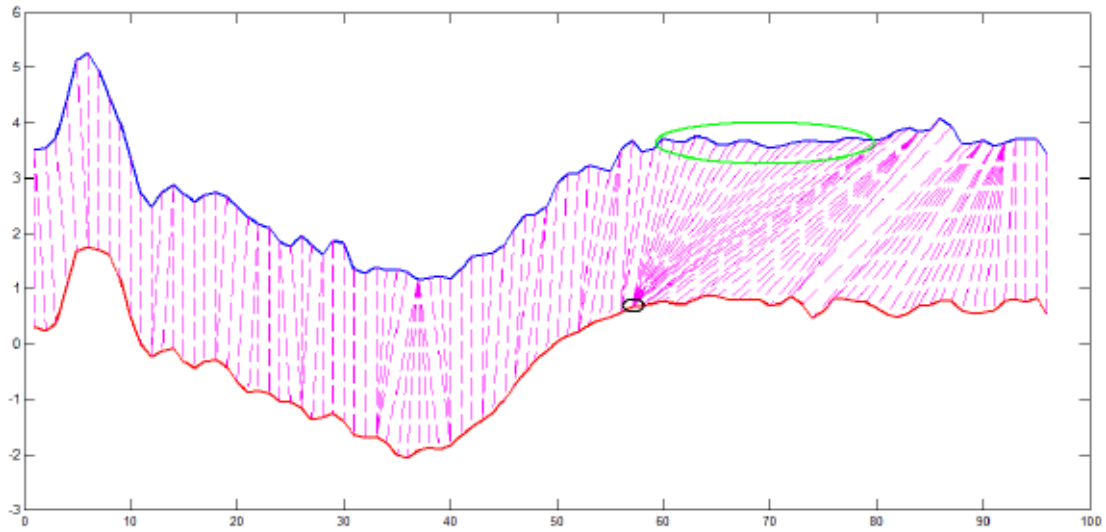

Figure 4.1: An example of the pathological alignment problem produced by the original DTW. We show that the point encircled by a black ellipse from the red time series is aligned to a large subsequence (green ellipse) from the blue time series (Lahreche and Boucheham, 2021a).

proaches, and many variants have been proposed for TSC. A critical observation is that most of the new variants did not take the earlier variants into account from an empirical comparison point of view. In other words, each method was evaluated independently regardless of other existing methods. In our opinion, we assume that DTW's variants need to be evaluated under a unified framework to develop a deeper understanding of the relationships between them. We also believe that such a study is of fundamental importance and could drive meaningful conclusions for the scientific community. However, to the best of my knowledge, no work in the literature addresses such a study within TSC.

In this chapter, we aim to provide a comprehensive empirical comparison in which we show which variant is the most appropriate for a particular TSC problem. With this goal in mind, we conduct a set of empirical evaluations reinforced by a deep statistical analysis of the original DTW and its most popular variants for TSC. In this work, we are only interested in evaluating methods in terms of classification accuracy.

4.2 Variants of DTW

In this section, we give a brief overview of the most popular variants specifically proposed for TSC. Since DTW has been thoroughly reviewed in previous chapters, we will not discuss it again to avoid repetition.

4.2.1 Constrained DTW (CDTW)

In this variant, the warping path is restricted by using a warping window technique. The warping window helps reduce the number of mappings between the two time series to be aligned. Although the simplicity of this method, it can mitigate the impact of the pathological alignment problem. There are different types of constraints on DTW. However, the most commonly used are the Sakoe-Chiba band (Sakoe and Chiba, 1978) and the Itakura parallelogram (Itakura, 1975). In this study, we are only interested in the Sakoe-Chiba band with the best value of the warping path (BWW) is set through cross-validation.

4.2.2 Derivative DTW (DDTW)

Keogh and Pazzani are maybe the first researchers to be interested in improving the alignment quality of DTW. Derivative DTW (DDTW) (Keogh and Pazzani, 2001) is one of their best contributions in this context. In contrast to the classic DTW algorithm, DDTW uses features-based rather than values-based to find the optimal alignment between sequences. As a first step, DDTW converts the original time series into a high-level representation using the first-order derivative. As a second step, the classic DTW algorithm is applied to align the generated derivatives time series. In (Keogh and Pazzani, 2001), the derivative D' of a data point x_i in a time-series T is defined as follows:

$$D'(x_i) = \frac{(x_i - x_{i-1}) + (\frac{x_{i+1} + x_{i-1}}{2})}{2}, 1 < i < n \tag{4.1}$$

4.2.3 Weighted DTW (WDTW)

Jeong et al. (2011) propose a penalty-based DTW called weighted DTW (WDTW). The key idea of this variant is that the data points of time series are not all treated in the same manner. To make the distinction, different weights are assigned to the data points according to the phase difference between a test data point and a reference data point. In the WDTW algorithm, the local distance between the i^{th} and j^{th} data points of Q and C, respectively is calculated as below:

$$d(q_i, c_i) = w_{|i-j|}(q_i - c_j)^2 \tag{4.2}$$

Where, $w_{|i-j|}$ represents a weight penalty value between the two data points q_i and c_j. The authors also proposed a modified logistic weight function in order to systematically determine the weight value w_i of the data points.

4.2.4 Derivative Distance DTW (DD-DTW)

Another variant of DTW is developed in (Górecki and Łuczak, 2013). The method is called Derivative Distance DTW (DD-DTW). It is a parameterized combination of a value-based DTW and a derivative-based DTW. DTW on raw time series and DTW on derivative time series distances are calculated in parallel. A weight value is attributed to each distance, and the sum of the weighted distances is given.

$$DD-DTW(Q,C) = \alpha * DTW(Q,C) + \beta * DTW(Q',C') \tag{4.3}$$

Where, *(Q, C)* are the original time-series, *(Q', C')* are their derivatives, respectively, and $0 \leq (\alpha, \beta) \leq 1$.

4.2.5 Complexity Invariant Distance (CID)

(Batista et al., 2014) present a simple and yet effective complexity factor for time series. The goal is to mitigate the problem of differences in the complexity between time series. The

complexity correction factor *CF* between two given time series is calculated as follows:

$$CF(Q,C) = \frac{max(CE(Q),CE(C))}{min(CE(Q),CE(C))} \tag{4.4}$$

Where *CE* is the complexity estimate of a time-series, such as:

$$CE(Q) = \sqrt{\sum_{i=1}^{n-1}(q_i - q_{i+1})^2} \tag{4.5}$$

The complexity factor could be combined with any distance *d* as follows:

$$CID(Q,C) = d(Q,C) * CF(Q,C) \tag{4.6}$$

The authors investigate their technique within ED and DTW distances. For DTW, we obtain the CIDDTW distance.

4.2.6 Shape Context DTW (SC-DTW)

In (Zhang et al., 2015), the authors propose the Shape Context DTW (SC-DTW), another alternative to DTW. SC-DTW uses feature-to-feature alignment rather than the alignment based on raw values considered by the classic DTW. The local shape of the time series is taken into account when computing the local distances between series. For that, each data point is represented using a shape context descriptor. The latter allows describing the environment around the data point by calculating the distribution of neighborhood data points.

4.2.7 Locally Weighted DTW (LWDTW)

In (Yuan et al., 2019a), the authors introduce the Locally Weighted DTW (LWDTW) method to improve the accuracy of the *k*-NN classifier under DTW. The aim is to pull the *k*-NN of the same class and to push the *k*-NN of different classes. The principle is to assign local weights to the time series elements based on discriminative features. The discriminative features are found using a learning scheme built on the neighborhood time series.

4.2.8 Limited warping path length DTW (LDTW)

Another variant of DTW is developed in (Zhang et al., 2017). As the name goes, Limited warping path length DTW (LDTW) makes constraints on the length of the warping path. LDTW consists of restricting the total number of alignments between two given time series. In other words, during the optimization process, LDTW automatically decides how many data points from a series each data point from another can be aligned to and where. It then allows to softly limit the length of the warping path.

4.2.9 Shape DTW (shapeDTW)

(Zhao and Itti, 2018) introduce the shapeDTW method, which mainly focuses on improving the alignment quality of the classic DTW. The shapeDTW does not treat the data points as isolated elements from each other, but it also examines the neighborhood data points to generate rich information about the local shape. Different descriptors are employed to extract local structure information including DWT, derivative, and Histogram of Gradient (HOG1D), etc. Therefore, the original time series are transformed into new sequences of local descriptors. The authors apply the classic DTW based on these so-obtained sequences to find the optimal alignment.

4.2.10 Locally Slope DTW (LSDTW)

Recently, a novel variant of DTW is proposed in (Yuan et al., 2019b). The method is called Locally Slope DTW (LSDTW), and it is based on local slope features. LSDTW attempts to align local similar shapes by considering the neighborhood information around each data point. LSDTW consists of three main steps. First, a filtering technique is used to remove noise. Second, for each data point, local slope features are extracted from adjacent data points. Next, local slope characteristics are transformed into symbols. Finally, DTW is applied on the generated representations to find the best warping path.

4.3 Experimental results and discussion

In this section, we conduct an extensive experimental comparison between the aforementioned variants (10 methods) and their original DTW method. We evaluate the methods in the context of TSC using the 1-NN classifier. The selection of 1-NN as a classifier is based on the fact that 1-NN is a parameter-free approach and its performance directly reflects the quality of the similarity measure used (Abanda et al., 2019). For the data, we have used 85 UCR time series datasets (Chen et al., 2015a).

Given that classification accuracy is the most important metric for assessing the performance of TSC algorithms (Schäfer, 2016), here, we are only interested in evaluating the methods according to this metric. Besides, since the classification accuracy is independent of the machine characteristics (CPU, RAM, etc.), we directly took the results of classifiers from these references (Bagnall et al., 2017; Chen et al., 2015a; Yuan et al., 2019a,b) without reproducing the experiments.

For clarity and simplicity, we conduct the assessment process in the form of a series of studies. In the first study, we perform an overall comparison. In the second study, we make a comparison according to the type of the dataset. In the last one, we evaluate the classifiers based on the nature of the time series dataset.

4.3.1 General comparison

In this study, we first compare the classification error rates of the methods together and then perform a pairwise comparison. Table 4.1 shows the classification error rates of the classic DTW and eight (08) of its variants on 85 datasets from the UCR benchmark (Chen et al., 2015a). The best results are marked in bold. We notice that we have excluded SC-DTW and LDTW variants from this comparison because we do not have their results on the full UCR benchmark. In other words, the original authors evaluated their methods on a subset of the UCR repository.

In general, we observe from the results that no method outperforms the others on all datasets.

The best error rates are distributed over all algorithms with varying proportions. However, we show that the DTW's variants perform better than the original DTW. Specifically, the results indicate that the LSDTW variant achieves superior accuracy against the others, where it is the best on 34 datasets. The shapeDTW method also performs good results, where it is the best on 20 datasets. By contrast, the rest accomplishes nearly similar performances. DTW(BWW) and DDTW are less accurate than DTW in terms of the number of wins. Note that a 'win' is credited to method 'A' over the rest methods on a particular dataset if the classification error rate of method 'A' is lower than those of all the other methods on that dataset.

Table 4.1: Classification error rates comparison between the classic DTW and 8 of its variants on 85 datasets from the UCR time series repository (Lahreche and Boucheham, 2021a).

Dataset	DTW	DTW(BWW)	DDTW	WDTW	DD-DTW	CIDDTW	LWDTW	shapeDTW	LSDTW
Adiac	0,396	0,391	0,417	0,383	0,332	0,373	0,386	**0,269**	0,345
ArrowHead	0,297	0,2	**0,132**	0,189	0,162	0,171	0,171	0,177	0,269
Beef	0,367	0,333	0,467	0,476	0,443	0,469	0,333	0,267	**0,167**
BeetleFly	0,3	0,3	**0,189**	0,196	**0,189**	0,194	0,200	0,2	0,2
BirdChicken	0,25	0,3	0,123	0,169	0,16	0,152	0,250	0,05	**0**
Car	0,267	0,233	0,265	0,281	0,269	0,286	0,233	**0,133**	**0,133**
CBF	0,003	0,004	0,436	0,007	0,007	0,016	**0,002**	0,08	0,003
ChlorineConcentration	0,352	0,35	0,319	0,352	0,3	0,351	0,356	0,355	**0,279**
CinCECGtorso	0,349	0,07	0,283	0,092	0,269	**0,046**	0,065	0,349	0,08
Coffee	**0**	**0**	0,042	0,014	0,014	0,011	**0**	0,036	0
Computers	0,3	0,38	0,3	0,313	**0,275**	0,293	0,312	0,356	0,392
CricketX	0,246	0,228	0,397	0,221	0,239	0,23	0,226	0,208	**0,182**
CricketY	0,256	0,238	0,455	0,25	0,258	0,275	0,241	0,226	**0,177**
CricketZ	0,246	0,254	0,418	0,216	0,227	0,224	0,251	0,208	**0,177**
DiatomSizeReduction	0,033	0,065	0,087	0,042	0,042	0,056	**0,023**	0,069	0,033
DistalPhalanxOutlineCorrect	**0,208**	0,228	0,241	0,246	0,243	0,247	0,261	0,233	0,279
DistalPhalanxOutlineAgeGroup	0,232	0,232	0,287	0,266	0,267	0,273	0,230	**0,228**	0,324
DistalPhalanxTW	0,29	**0,272**	0,388	0,381	0,396	0,377	0,381	0,29	0,338
Earthquakes	**0,258**	**0,258**	**0,285**	0,305	0,292	0,306	0,331	**0,258**	0,281
ECG200	0,23	0,12	0,188	0,136	0,183	0,13	**0,060**	0,1	0,15
ECG5000	0,076	0,075	0,082	0,073	0,074	0,073	0,075	0,071	**0,07**
ECGFiveDays	0,232	0,203	0,333	0,176	0,249	0,177	0,165	**0,057**	0,1
ElectricDevices	0,399	0,376	0,245	**0,209**	0,224	0,224	0,398	0,4	0,383
FaceAll	0,192	0,192	0,083	**0,04**	0,055	0,044	0,164	0,238	0,217
FaceFour	0,17	0,114	0,281	0,14	0,154	0,168	0,114	**0,091**	**0,091**
FacesUCR	0,095	0,088	0,155	0,077	0,102	0,091	0,092	0,081	**0,065**
FiftyWords	0,31	0,242	0,306	0,235	0,252	0,226	0,231	0,242	**0,178**
Fish	0,177	0,154	0,109	0,183	0,079	0,187	0,154	**0,051**	0,069
FordA	0,438	0,341	0,302	0,323	0,297	0,319	0,319	0,279	**0,264**
FordB	0,406	0,414	0,287	0,337	0,279	0,286	0,379	**0,261**	0,341
GunPoint	0,093	0,087	0,018	0,044	0,045	0,049	0,067	0,007	**0**
Ham	0,533	0,4	0,343	**0,253**	0,325	0,285	0,391	0,457	0,343
HandOutlines	0,202	0,197	0,215	0,145	0,132	0,145	0,135	0,206	**0,119**
Haptics	0,623	0,588	0,692	0,594	0,623	0,585	0,591	0,623	**0,542**
Herring	0,469	0,469	0,463	0,45	0,473	0,456	**0,422**	0,5	0,531
InlineSkate	0,616	0,613	0,53	0,596	**0,447**	0,572	0,602	0,616	0,576
InsectWingbeatSound	0,645	**0,422**	0,756	0,447	0,657	0,446	0,430	0,584	0,491
ItalyPowerDemand	0,05	0,045	0,114	0,066	0,08	0,05	**0,044**	0,103	0,05
LargeKitchenAppliances	0,205	0,205	0,222	0,205	0,207	0,217	0,200	**0,16**	0,197

Continued on next page

Dataset	DTW	DTW(BWW)	DDTW	WDTW	DD-DTW	CIDDTW	LWDTW	shapeDTW	LSDTW
Lightning2	0,131	0,131	0,338	0,163	0,185	0,174	**0,098**	0,115	0,131
Lightning7	0,274	0,288	0,45	0,246	0,306	0,281	0,247	**0,233**	0,274
Mallat	0,066	0,086	0,082	0,055	0,05	**0,046**	0,079	0,062	0,1
Meat	0,067	0,067	0,241	0,029	0,032	**0,02**	0,067	0,1	0,2
MedicalImages	0,263	0,253	0,336	0,249	0,262	0,257	0,259	0,264	**0,239**
MiddlePhalanxOutlineCorrect	0,25	0,253	0,276	0,247	0,27	**0,223**	0,268	0,26	0,435
MiddlePhalanxOutlineAgeGroup	0,352	0,318	0,425	0,434	0,424	0,428	0,481	**0,25**	0,268
MiddlePhalanxTW	**0,416**	0,419	0,495	0,491	0,489	0,501	0,474	0,429	0,5
MoteStrain	0,165	0,134	0,296	0,152	0,179	0,213	0,130	0,11	**0,051**
NonInvasiveFatalECGThorax1	0,209	0,185	0,302	**0,183**	0,186	0,203	0,195	0,219	0,211
NonInvasiveFatalECGThorax2	0,135	0,129	0,174	**0,112**	0,12	0,134	0,127	0,14	0,139
OliveOil	0,167	0,133	0,217	0,132	0,15	0,124	0,133	**0,1**	0,167
OSULeaf	0,409	0,388	0,131	0,357	0,115	0,34	0,388	0,132	**0,099**
PhalangesOutlinesCorrect	0,272	0,239	0,248	0,237	0,245	0,235	0,248	0,261	**0,228**
Phoneme	0,772	0,773	0,747	0,777	0,74	0,779	0,785	0,736	**0,708**
Plane	**0**	**0**	0,001	**0**	**0**	0,005	**0**	**0**	**0**
ProximalPhalanxOutlineCorrect	0,195	0,215	0,181	0,186	0,185	0,183	0,213	0,21	**0,18**
ProximalPhalanxOutlineAgeGroup	0,216	0,21	0,21	0,227	0,229	0,234	0,185	0,206	**0,124**
ProximalPhalanxTW	0,263	0,263	0,278	0,269	0,263	0,271	**0,234**	0,275	0,239
RefrigerationDevices	0,536	0,56	0,433	0,43	**0,397**	0,413	0,517	0,507	0,517
ScreenType	0,603	0,589	0,454	0,535	**0,438**	0,494	0,573	0,525	0,579
ShapeletSim	0,35	0,3	0,438	0,318	0,367	0,246	0,294	**0,028**	0,128
ShapesAll	0,232	0,198	0,151	0,189	0,138	0,182	0,193	**0,112**	0,118
SmallKitchenAppliances	0,357	0,328	0,364	0,321	0,36	0,32	0,317	0,301	**0,219**
SonyAIBORobotSurface1	0,275	0,305	0,231	0,189	0,195	**0,085**	0,241	0,193	0,218
SonyAIBORobotSurface2	0,169	0,141	0,138	0,147	0,137	**0,107**	0,129	0,174	0,163
StarlightCurves	0,093	0,095	**0,035**	0,087	**0,035**	0,084	0,100	0,1	0,041
Strawberry	0,06	0,062	**0,04**	0,046	0,044	0,045	0,054	0,051	0,051
SwedishLeaf	0,208	0,154	0,106	0,142	0,104	0,13	0,152	**0,085**	0,141
Symbols	0,05	0,062	0,034	0,058	0,044	0,07	0,059	0,039	**0,017**
SyntheticControl	0,007	0,017	0,433	0,011	0,009	0,021	0,017	0,153	**0**
ToeSegmentation1	0,228	0,25	0,261	0,272	0,258	0,282	0,224	**0,101**	0,158
ToeSegmentation2	0,162	**0,092**	0,171	0,138	0,172	0,156	0,100	0,138	0,108
Trace	**0**	0,01	**0**	**0**	**0**	0,005	0,010	**0**	**0**
TwoLeadECG	0,096	0,132	0,029	0,09	0,054	0,115	0,115	**0,006**	0,068
TwoPatterns	**0**	0,002	0,002	**0**	**0**	0,001	0,001	0,001	**0**
UWaveGestureLibraryX	0,273	0,227	0,324	0,225	0,226	0,213	0,222	0,263	**0,179**
UWaveGestureLibraryY	0,366	0,301	0,419	0,313	0,29	0,282	0,298	0,358	**0,237**
UWaveGestureLibraryZ	0,342	0,322	0,411	0,316	0,305	0,288	0,321	0,338	**0,235**
UWaveGestureLibraryAll	0,108	0,034	0,154	0,039	0,066	0,096	0,037	0,058	**0,024**
Wafer	0,02	0,005	0,025	**0,004**	0,017	0,006	**0,004**	0,01	0,005
Wine	0,426	0,389	0,152	0,115	0,119	**0,109**	0,389	0,537	0,426
WordSynonyms	0,351	0,252	0,338	0,269	0,29	0,262	0,251	0,26	**0,224**
Worms	0,536	0,586	**0,362**	0,421	0,383	0,367	0,429	0,475	0,403
WormsTwoClass	0,337	0,414	0,291	0,323	0,291	**0,264**	0,286	0,287	0,312
Yoga	0,164	0,155	0,169	0,142	0,132	0,142	0,153	**0,117**	0,136
Average	0,256	0,237	0,266	0,221	0,219	0,218	0,228	0,217	**0,205**
Wins	7	5	6	9	9	8	11	20	**34**

Figure 4.2 presents the box plot of ranks of measures on 85 UCR datasets. The red line inside the boxes represents the median rank; the less median rank, the better measure. As shown, LSDTW performs better than all the other measures with a median rank equal to three (03). On the other hand, DDTW is the worst one. The shapeDTW also provides good results, while

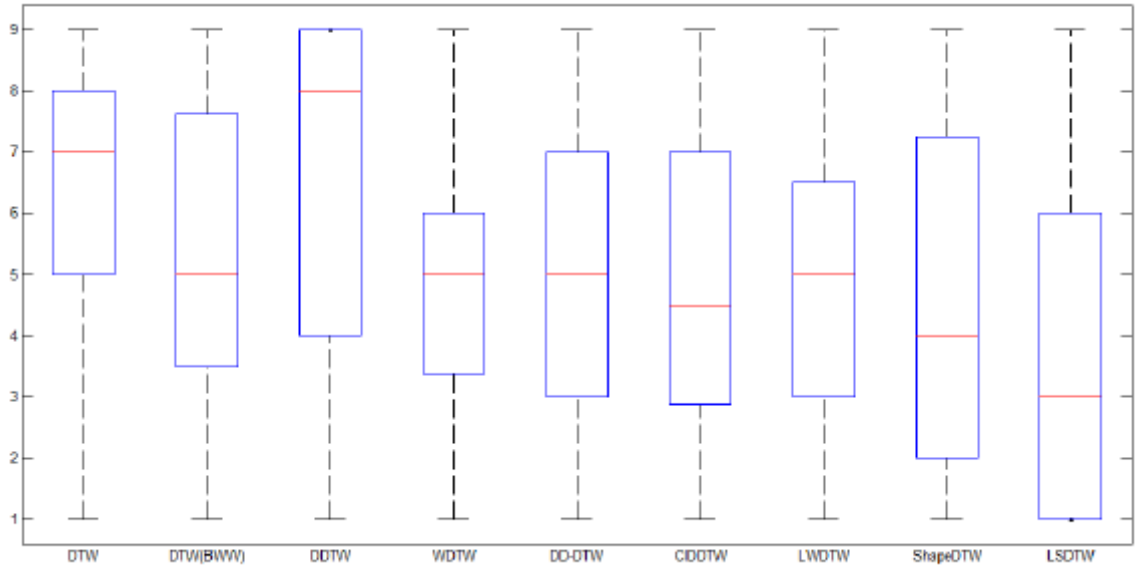

Figure 4.2: Box plot of ranks of each measure across 85 UCR time series archive (Lahreche and Boucheham, 2021a).

DTW(BWW), DD-DTW, WDTW, and LWDTW methods achieve the same performance.

To deeply compare the accuracy of methods and to show whether there is a significant difference between them, we carry out a statistical test comparison. In this context, we use the *Friedman test* (Friedman, 1937, 1940), which is recommended in (Demšar, 2006). We start by computing the ranks of methods for each dataset based on the classification error rates. Then, we compute the average ranks across all datasets (85 datasets in our study). In (Demšar, 2006), if we want to compare nine (09) classifiers $(k = 9)$ over 85 datasets $(N = 85)$, at a confidence level of $(\alpha = 0,05)$, the critical value is $(q_{0.05} = 3.102)$, (Eq. 4.7). Accordingly, a classifier is significantly better than another if the absolute value of the difference between their average ranks is superior to the critical difference *CD* value, which is calculated as follows:

$$CD = q_{0,05}\sqrt{\frac{k*(k+1)}{6N}} = 3,102\sqrt{\frac{9*10}{6*85}} = 1,303 \tag{4.7}$$

The results of this study are illustrated in Fig. 4.3 (Critical difference diagram: CD). The quantified line represents the average rank; the less average rank, the better method. Methods connected by a black bar are not statistically significantly better with each other, and vice versa.

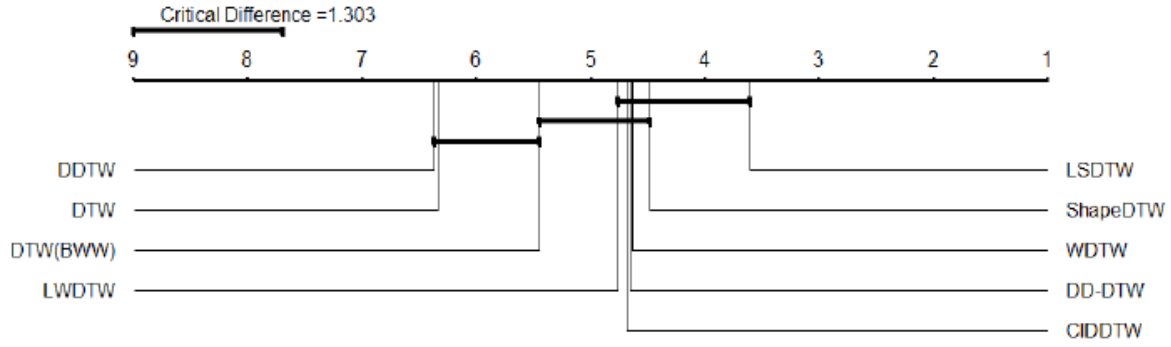

Figure 4.3: Comparison of 9 algorithms against each other using Nemenyi test (Nemenyi, 1963). Groups of classifiers that are not statistically significantly different at $(p = 0,05)$ are connected by a black bar (Lahreche and Boucheham, 2021a).

The critical difference diagram shows that 6 out of 8 variants are significantly more accurate than the classic DTW. On the other hand, DTW(BWW) and DDTW are not significantly better than DTW. Moreover, there is no significant difference between virtually all DTW's variants.

Table 4.2 shows a pairwise comparison of DTW and 10 of its variants (including this time SC-DTW and LDTW methods) in terms of classification error rate. Values in the table present the number of wins-ties-losses, respectively. We give an example to show how to read the results in the table: e.g., the DTW method (column 1, line 2) wins on 25, ties on 11, and losses on 49 datasets compared to DTW(BWW) method (column 2, line 1). If we look at the number of wins, we can get Fig. 4.4, which shows the number of methods each is better than. We observe that LSDTW is better than all the others. SC-DTW, LDTW, and shapeDTW methods perform superior to the majority of methods. For CIDDTW, DD-DTW, and WDTW methods, each beats half of the methods (5/10). LWDTW, DTW(BWW), and DTW are better than 3, 2, and 1 methods, respectively. By contrast, DDTW does not beat any method.

4.3.2 Comparison against the problem type

In this study, we take the domain (problem) of datasets into account. In the UCR time series archive, there are seven (07) problem families: *image outline*, *sensor readings*, *motion capture*, *spectrographs*, *electric devices*, *ECG measurements*, and *simulated*. The number of datasets in each domain is shown in Table 4.3 (Counts column). This study aims to show which method

Table 4.2: Pairwise comparison of classifiers in terms of the classification accuracy on the UCR time series repository (Lahreche and Boucheham, 2021a).

Methods	DTW(BWW)	DDTW	WDTW	DD-DTW	CIDDTW	LWDTW	shapeDTW	LSDTW	SC-DTW	LDTW
DTW	25-11-49	47-2-36	21-5-59	28-5-52	27-1-57	18-4-63	24-7-54	13-11-61	5-3-26	1-6-15
DTW(BWW)		51-2-32	31-2-52	39-2-44	36-0-49	19-13-53	35-3-47	22-4-59	9-1-24	1-5-16
DDTW			30-1-54	15-4-66	29-0-56	31-1-53	21-1-63	23-2-60	9-2-23	4-1-17
WDTW				41-6-38	39-3-43	45-3-37	41-3-41	27-3-55	9-2-23	3-3-16
DD-DTW					45-1-39	45-1-39	36-3-46	31-3-51	12-2-20	6-3-14
CIDDTW						47-4-34	41-1-43	29-1-55	9-1-24	4-0-18
LWDTW							37-4-44	28-4-53	12-1-21	3-5-14
shapeDTW								31-6-48	14-2-18	10-3-9
LSDTW									18-4-12	8-6-8
SC-DTW										7-2-11

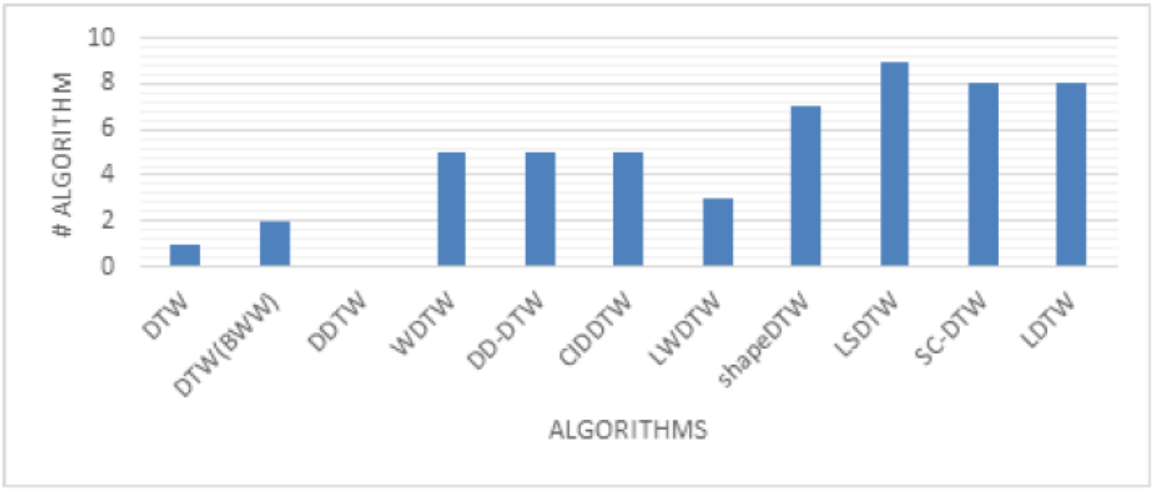

Figure 4.4: Comparison of algorithms in terms of classification accuracy (Lahreche and Boucheham, 2021a).

is the most adequate for a specific problem. Table 4.3 presents the performance (in percentage %) of methods for each problem type. Best results are marked in bold. For example, for the image outline problem, DTW is better on 6% of datasets.

From the results, we remark that LSDTW achieves superior performance on almost all problems. DD-DTW is the most appropriate for the electric device problem. On the other hand, WDTW and shapeDTW methods outperform the others for the ECG measurements problem. We also observe that shapeDTW and LWDTW accomplish good performance, particularly for image outline and sensor readings problems. Indeed, these results are not conclusive due to the small number of datasets in each problem.

Table 4.3: Performance comparison (in %) of classifiers according to the problem type (Lahreche and Boucheham, 2021a).

Problem	DTW	DTW(BWW)	DDTW	WDTW	DD-DTW	CIDDTW	LWDTW	shapeDTW	LSDTW	Counts
Image Outline	6,897	3,448	6,897	3,448	3,448	3,448	10,345	27,586	41,379	29
Sensor Readings	16,667	16,667	11,111	16,667	16,667	16,667	22,222	33,333	38,889	18
Motion Capture	0,000	0,000	7,143	0,000	7,143	7,143	7,143	7,143	64,286	14
Spectrographs	14,286	14,286	14,286	14,286	0,000	2,000	14,286	14,286	28,571	7
Electric Devices	0,000	0,000	0,000	16,667	50,000	0,000	0,000	16,667	16,667	6
ECG Measurements	0,000	0,000	0,000	33,333	0,000	0,000	16,667	33,333	16,667	6
Simulated	20,000	0,000	0,000	20,000	20,000	20,000	20,000	20,000	40,000	5

4.3.3 Comparison against time series nature

Now, we try to compare the effectiveness of methods according to the nature of the time-series datasets. In this study, we mean by nature: Long or Short Time Series datasets (LTS/STS). Here, a time series of length equal or superior to 500 time points is considered as long, otherwise, it is short. Accordingly, in the UCR time series repository, there are 28 LTS and 57 STS.

Figure 4.5 displays the performance of algorithms on both LTS and STS. For long time-series datasets, we see that LSDTW, shapeDTW, and DD-DTW methods are better than the others, but LSDTW is the best. For the rest, they provide nearly equivalent performances. On the other hand, for short time-series datasets, it is clear that LSDTW is largely better than all the others. Surprisingly, we remark that the classic DTW performs superior to four (4) of its variants which are DTW(BWW), DDTW, DD-DTW, and CIDDTW methods for short time-series datasets.

To summarize, from all these extensive studies, we can draw the following conclusions:

- No variant outperforms all the others on all datasets of the UCR time series archive.
- Practically all DTW's variants are statistically significantly better than the classic DTW.
- There is no significant difference between virtually all variations of DTW.
- Overall, the LSDTW method performs superior to all the others.
- DDTW is the only variant that achieves poor results than DTW.

Finally, we believe that this study will be useful to researchers interested in improving DTW.

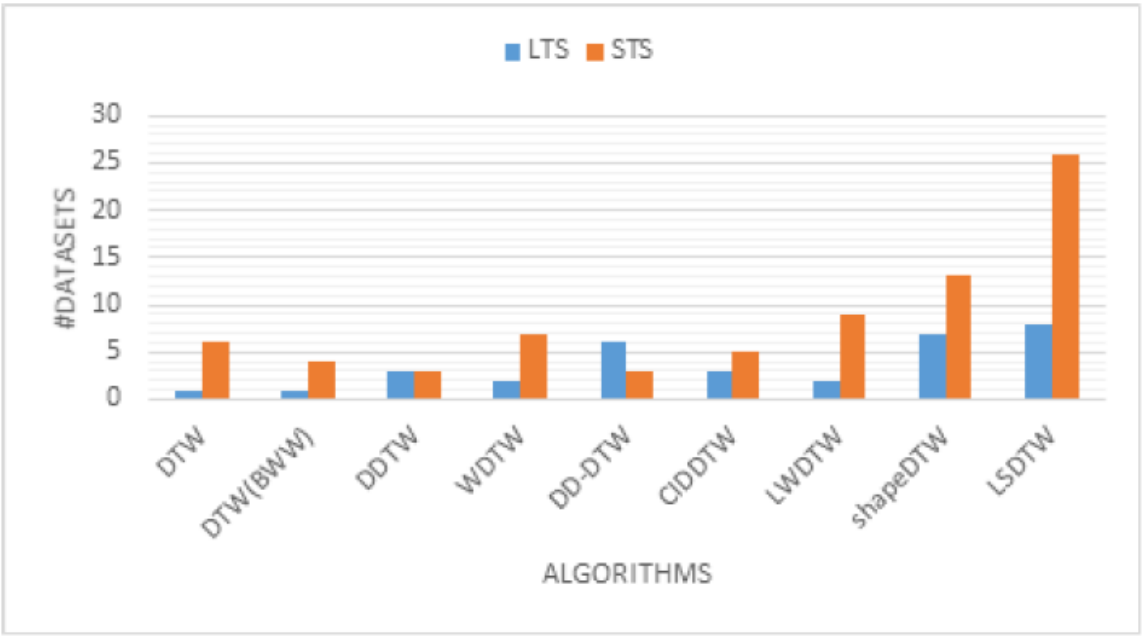

Figure 4.5: Performance comparison of algorithms according to time series dataset nature. LTS: Long Time Series dataset, STS: Short Time Series dataset (Lahreche and Boucheham, 2021a).

We also believe that it will help practitioners and engineers to choose the appropriate variant for their specific purpose. Moreover, we highly recommend researchers consider these variants when proposing a new improved version of DTW. We claim that a novel variant is only of interest if it is significantly accurate than at least one of the current variants and ideally significantly superior to the LSDTW variant.

4.4 Conclusion

In this chapter, we performed a set of empirical comparisons between the original DTW and its most popular variants. The comparison was conducted in the context of the TSC task using the 1-NN classifier on the public UCR time series classification archive. In this work, we restricted our evaluation to the classification accuracy metric. Our objective was to provide a comprehensive comparison in which we show which variant is the most suitable for a particular TSC problem. From the results, we found that no variant outperforms all the others on the full UCR time series repository. However, LSDTW is generally the best one in terms of classification

accuracy. Results also showed that practically all variations are significantly better than the classic DTW. Moreover, we found that there is no significant difference between virtually all variants.

Chapter 5

A fast and accurate similarity measure for long time series classification based on local extrema and dynamic time warping (Lahreche and Boucheham, 2021b)

5.1 Introduction

With the ubiquitous of long time series and the increasing demand for classifying them on limited resource devices, there is a crucial need for efficient and accurate measures to deal with such data (Lang et al., 2009; Sharabiani et al., 2017). Indeed, there is a plethora of good time-series similarity measures in the literature. However, most existing methods achieve good performance for short time series, but their effectiveness decreases quickly as time series are longer (Lang et al., 2009; Lin et al., 2012a; Schäfer, 2016).

In this chapter, we develop a new parameter-free measure for the specific purpose of efficiently and accurately assessing the similarity between two given long time series. The proposed "Local Extrema Dynamic Time Warping" (LE-DTW) (Lahreche and Boucheham, 2021b) consists

of two steps. The first is a time series representation technique that starts by reducing the dimensionality of a given time series using its local extrema. Next, it physically separates the minima and maxima points for more intuitiveness and consistency of the so-obtained time series representation. The second step consists in adapting the DTW measure so as to evaluate the score of similarity between the generated representations.

We test the performance of LE-DTW on a wide range of real-world problems from the UCR time series archive for TSC. Experimental results indicate that for short time series, the proposed method achieves reasonable classification accuracy as compared to DTW. However, for long time series, LE-DTW performs much better. Specifically, it outperforms DTW while providing competitive results against popular distance-based classifiers. Moreover, in terms of efficiency, LE-DTW is orders of magnitude faster than DTW.

5.2 Background and related work

In this section, we first discuss the concept of long time series. Next, we review related works on long time series classification. At last, we give an overview of time series representations based on local extrema.

5.2.1 Long time series

The amount of time series did not cease to grow remarkably over the years. Long time series constitute a large part of this data (Lang et al., 2009; Sharabiani et al., 2017). Long time series are time series with the particularity to have very high dimensionality. They exist in a wide range of fields including, but not limited to, medicine, where, for only five minutes of ECG recording at a sampling rate of 360 Hz, the resulting signal contains more than 100,000 samples; astronomy, where, for example, tracking specific particles can last many years. In the literature, there is a conflicting of opinions regarding the length of long time series. For example, Sharabiani et al. (2017) regard time series equal to or longer than 500 samples to be long. However, in (Lang et al., 2009), it is from 1000 samples that a time series is considered long.

5.2.2 Related work

To the best of my knowledge, very few publications are available in the literature that exclusively address the problem of classifying long time series (Lang et al., 2009; Grabocka et al., 2012; Sharabiani et al., 2017). Lang et al. (2009) investigate a compression technique to calculate the similarity of both short and long time series for classification. They proposed a method called Dictionary Compression Score (DCS). It is based on two techniques: Kolmogorov complexity and the Lempel-Ziv compression framework. The authors also proposed a heuristics algorithm in order to automatically find an appropriate parameter for tuning their method.

A few years later, the problem of classifying long time series was raised again by Grabocka et al. (2012). They introduce the FMF-SVM method, in which they first use matrix factorization to transform the original time series to a low space dimensionality. Then, they employ an SVM with polynomial kernels on the reduced space to classify data.

Recently, Sharabiani et al. (2017) develop 3-D DTW for efficiently classifying long time series. 3-D DTW begins by reducing the length of the raw time series using the control chart approximation (CCA) technique. This technique consists in transforming the time series into a set of segments, where each contains aggregated values and durations. Afterward, an adaptation of DTW was proposed to be applied to the approximate representation for the 1-NN classifier.

All the above-mentioned methods require the setting of some parameters, whereas, parameter-laden algorithms have several disadvantages. In particular, the miss-configuration of parameters may influence the quality of results. It is also largely admitted that it is generally difficult to reproduce the results of algorithms with many parameters (Keogh et al., 2004). Therefore, it is desirable to design algorithms with as few parameters as possible, ideally none (Keogh et al., 2004). In this contribution, we propose a non-parametric similarity measure, which we denote by LE-DTW.

5.2.3 Extrema features-based representation

The use of extrema points to represent time series was shown to be a promising approach for time series analysis (Fink and Pratt, 2004; Fu et al., 2008; Ma and Zhang, 2017; Pratt and Fink, 2002; Perng et al., 2000; Vemulapalli et al., 2012; Yan et al., 2013). Perng et al. (2000) introduce a representation method for efficiently querying in time series databases. The method is based on the landmarks model, which consists of local minima and maxima. Pratt and Fink (2002) and Fink and Pratt (2004) propose a compression method for indexing time series method, which is exclusively based on major minima and maxima.

Fu et al. (2008) also explore the concept of important points in their work on financial time series. They represent financial time series according to their minima and maxima. Vemulapalli et al. (2012) studied the problem of extracting proper extrema features for time series analysis. The authors suggested an optimal filter based on training time series for robustly identifying extrema features. Recently, Ma and Zhang (2017) propose a time-series representation technique based on the local minimum and maximum to extract trend features of time series.

Surprisingly, although the importance of extrema features within time series analysis, to the best of my knowledge, extrema-based methods have never been used in the context of long time series classification. Motivated by the fundamental importance of extrema feature in diverse time series tasks (similarity search, compression, indexation, etc.), our objective in this contribution is to investigate its utility for classifying long time series. In this regard, we propose a new local extrema-based representation for long time series classification. Our method is intuitive and parameter-free. Compared to existing extrema-based representation techniques, ours provides a new intuitive idea in which it physically separates minima and maxima, which give us two independent sub-sequences.

5.3 Proposed LE-DTW method

In this section, we deeply describe our proposed LE-DTW method. First, we present the local extrema extraction technique, and then we describe the similarity establishment strategy. Figure

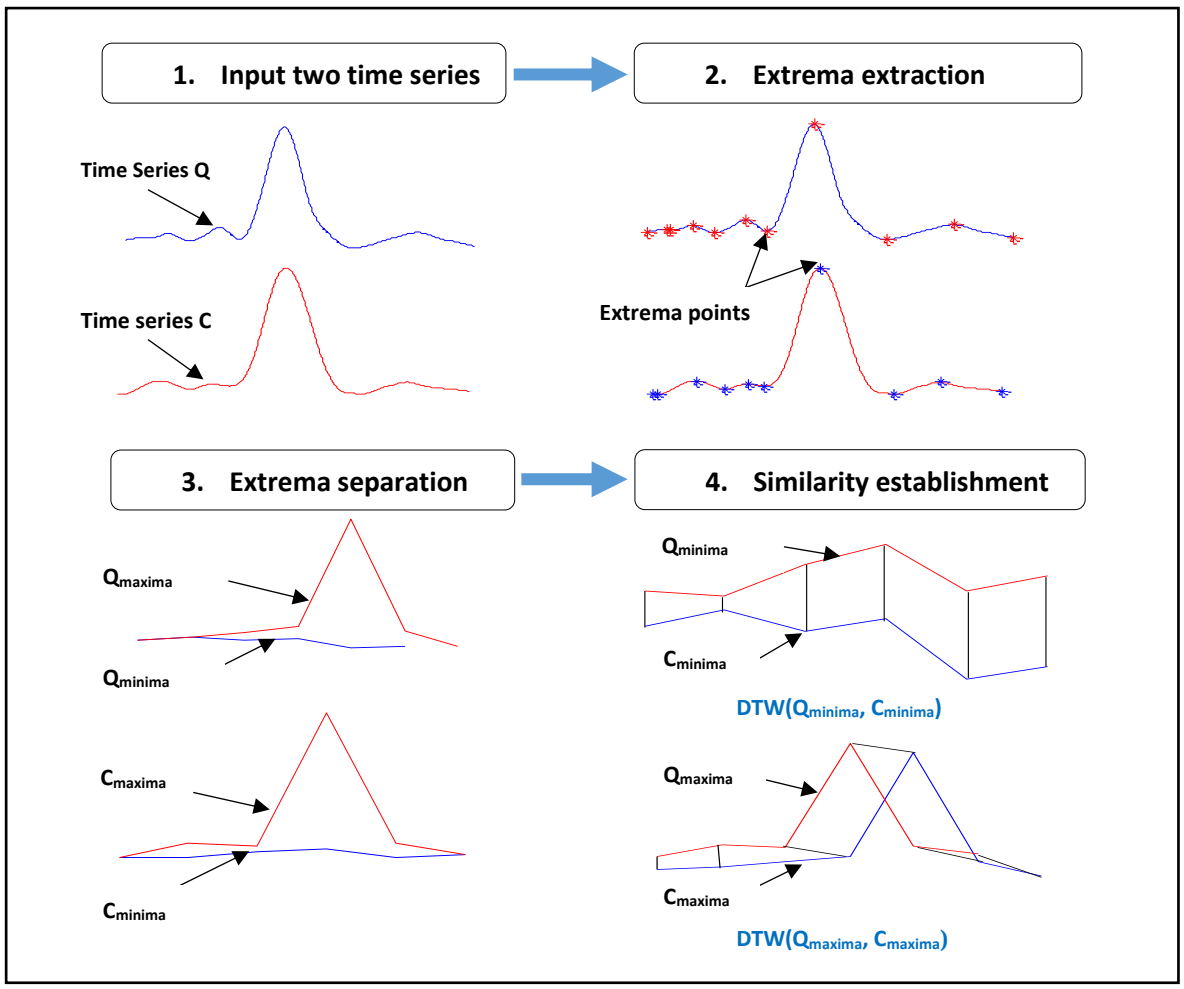

Figure 5.1: Pipeline of LE-DTW. LE-DTW consists of three major steps: Local-extrema extraction, Extrema separation, and Similarity establishment (Lahreche and Boucheham, 2021b).

5.1 represents an overview of the proposed method.

5.3.1 Local-extrema extraction

In this step, we propose a representation technique that reduces the dimensionality of the original time series while preserving its main discriminant characteristics. The technique consists in approximately representing a time series using its local extrema feature. In other words, from the entire time series, we extract only the local extrema points and discarding the others. In this paper, we distinguish four types of local extrema in time series called: *first point*, *last point*, *local minima*, and *local maxima*. Figure 5.2 visualizes these four extrema types.

The major motivations behind using local extrema-based features to represent time series are:

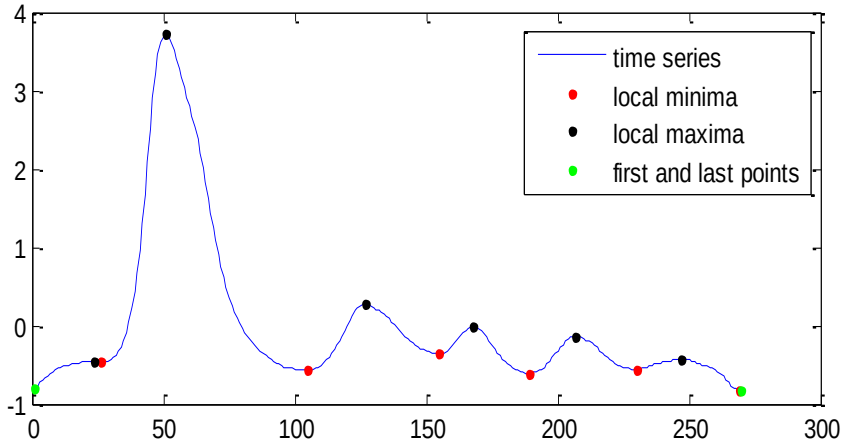

Figure 5.2: Illustration of local extrema points of a time series: first/last points (green pints), local minima (red points), and local maxima (black points) (Lahreche and Boucheham, 2021b).

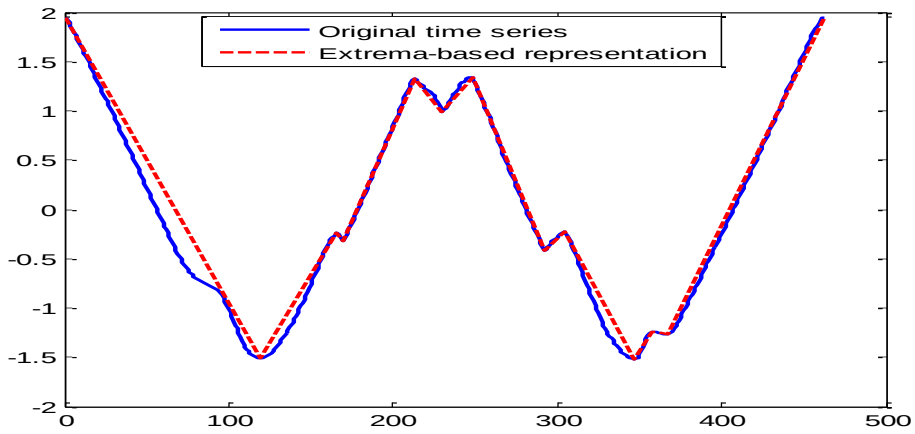

Figure 5.3: Example of time series (blue line) and its approximate representation using extrema points (red dotted line) (Lahreche and Boucheham, 2021b).

- Local extrema features can capture the overall shape of a time series since it usually depends on the fluctuant of its local minima/maxima points. Indeed, the shape of the time series is of fundamental importance in time series analysis and mining, even humans typically consider two time series similar if they have the same overall shape (Esling and Agon, 2012). Figure 5.3 shows the ability of local extrema features in capturing the overall shape of the time series.

- Local extrema can keep the variation in the time series trends (Ma and Zhang, 2017).

- They could also be robust under some common distortions in time series such as scaling and shifting (Perng et al., 2000; Vemulapalli et al., 2012).

- As can be easily checked, the procedure for detecting local extrema is of linear order and constant in memory consumption, which explains its high efficiency.

We notice here that in most of the related works, the corresponding authors use a parameter (threshold) to select some interesting local extrema points. In our representation, we use no parameter and extract all the local extrema points existing in the original time series. The aim is for our method to be parameter-free.

Formally, given a time series $Q = \{q_1, ..., q_i, ..., q_n\}$ of length n and q_i is its data point. We identify q_i as local extremum of Q if and only if, it satisfies one of the following conditions:

- q_i is the first point: $i = 1$.

- q_i is the last point: $i = n$.

- q_i is a local minimum (q^{lmin}): $q_i < q_{i-1}$ and $q_i \leq q_{i+1}$ or $q_i \leq q_{i-1}$ and $q_i < q_{i+1}$.

- q_i is a local maximum (q^{lmax}): $q_i > q_{i-1}$ and $q_i \geq q_{i+1}$ or $q_i \geq q_{i-1}$ and $q_i > q_{i+1}$.

Therefore, the approximate representation Q' of Q is formulated as follows:

$$Q' = \{q_1^{extrema}, ..., q_i^{extrema}, ..., q_l^{extrema}\} \tag{5.1}$$

Where l is its length with ($l << n$), $q_1^{extrema} = q_1$, $q_l^{extrema} = q_n$, and $q_i^{extrema} = q^{lmin}$ or $q_i^{extrema} = q^{lmax}$.

5.3.2 Similarity establishment

After transforming Q and C two time series to compare to their approximate representations Q' and C', respectively in the previous step as follows:

$$Q' = \{q_1^{extrema}, ..., q_i^{extrema}, ..., q_l^{extrema}\} \tag{5.2}$$

$$C' = \{c_1^{lextrema}, ..., c_j^{lextrema}, ..., c_h^{lextrema}\} \tag{5.3}$$

We can at this stage directly apply an already existing similarity measure in the literature on these approximate representations to evaluate their score of resemblance. However, we indicate here that *Q'* and *C'* could be of different lengths. Therefore, the selected similarity measure must be able to match this kind of time series. In this context, elastic similarity measures such as DTW and LCSS can deal with time series even they have different lengths.

In this contribution, we particularly choose to use DTW as a similarity measure due to its popularity, notable effectiveness, and especially it is being parameter-free. Nevertheless, DTW could generate an unintuitive alignment, where a minimum (maximum) point from one sequence could be aligned with a maximum (minimum) point of the other sequence, if they are close form a point of view distance. We claim that the reason why DTW does not differentiate between minimum and maximum points is that it does not take into account the neighbor points. To overcome this issue, we suggest a novel strategy. This strategy consists in physically separating the local minima from the local maxima. Otherwise, from a single approximate representation sequence, we obtain two sub-sequences, one contains only the local minima (minima sub-sequence) and the other contains only the local maxima (maxima sub-sequence) as follows:

$$Q' = \{Q'^{lmin}, Q'^{lmax}\} \tag{5.4}$$

$$C' = \{C'^{lmin}, C'^{lmax}\} \tag{5.5}$$

Where,

$$\begin{cases} Q'^{lmin} = (q'_{mini}{}^{lmin})_{mini=1:minl} \\ Q'^{lmax} = (q'_{maxi}{}^{lmax})_{maxi=1:maxl} \end{cases} \tag{5.6}$$

$$\begin{cases} C'^{lmin} = (c'_{mini}{}^{lmin})_{mini=1:minh} \\ C'^{lmax} = (c'_{maxi}{}^{lmax})_{maxi=1:maxh} \end{cases} \tag{5.7}$$

Figure 5.4 illustrates the local-extrema separation technique. As shown, the local-extrema points of the original time series on top (a) have been identified, local minima in red and local

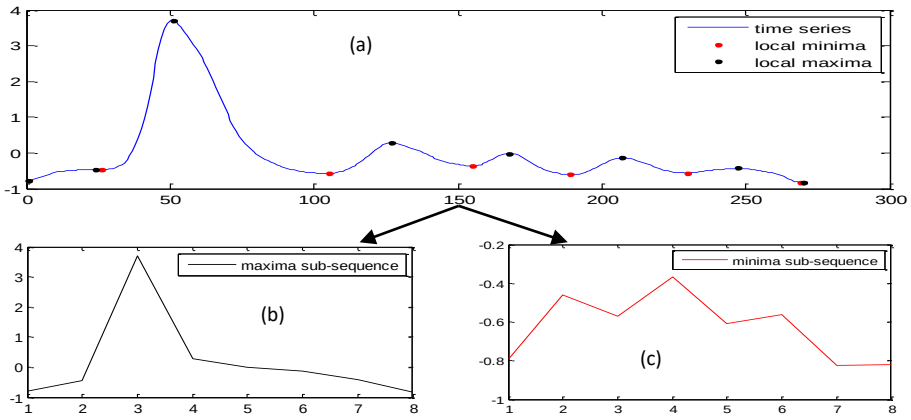

Figure 5.4: Example showing the extrema-features separation technique proposed in this paper, (a) time series and its local minima (red points) and local maxima (black points), (b) subsequence only formed of local maxima, (c) subsequence only formed from of minima (Lahreche and Boucheham, 2021b).

maxima in black. Next, local minima and maxima points have beenphysically separated: local minima subsequence in (b), and local maxima subsequence in (c).

As stated above, local extrema also include endpoints (first/last), we, therefore, put these points in both minima sub-sequences and maxima sub-sequences as also endpoints as follows:

$$\begin{cases} Q'^{lmin} = (q'_1, q'^{lmin}, q'_n) \\ Q'^{lmax} = (q'_1, q'^{lmax}, q'_n) \end{cases} \tag{5.8}$$

$$\begin{cases} C'^{lmin} = (c'_1, c'^{lmin}, c'_m) \\ C'^{lmax} = (c'_1, c'^{lmax}, c'_m) \end{cases} \tag{5.9}$$

The key advantage of the local-extrema separation technique is that it, potentially, gives more harmony and consistency to our representation method, where similar features are grouped together. Consequently, it thereafter allows a more intuitive similarity measure to be calculated. At present, we can use DTW to compute the similarity between generated representations. However, we have four (04) sub-sequences, two minima sub-sequences, and two maxima sub-sequences. Accordingly, we apply DTW two times, one on minima sub-sequences to calculate the first local distance ld^{min}, and the other on maxima sub-sequences to compute the second

local distance ld^{min}:

$$ld^{min}(Q'^{lmin}, C'^{lmin}) = DTW(Q'^{lmin}, C'^{lmin}) \tag{5.10}$$

$$ld^{max}(Q'^{lmax}, C'^{lmax}) = DTW(Q'^{lmax}, C'^{lmax}) \tag{5.11}$$

At this stage, we did not yet calculate the final similarity *Sim(Q,C)* between Q and C. To do that, we use a function $f(ld^{min}, ld^{max})$ that takes local distances $f(ld^{min}, ld^{max})$ as inputs and returns the similarity *Sim(Q,C)*. This function can be a simple arithmetic operation (addition+, multiplication*, etc.), as it can be a complex function such as weighed function. We have experimentally tested some functions and we have found that the addition function gives better accuracy. Therefore, the final similarity is the sum of the local distances:

$$Sim(Q,C) = f(ld^{min}, ld^{max}) = ld^{min} + ld^{max} \tag{5.12}$$

5.3.3 Algorithm overview

In this section, we give the pseudo-code of our proposed LE-DTW algorithm in order to correctly reproduce it. From Algorithm in Fig. 5.5 shown below, we can divide it into two parts. The first part (from line 1 to line 22) is the extraction and separation procedures of minima/maxima points from the two time series Q and C to compare. The second part (from line 23 to line 26) is the similarity establishment technique. In lines 23 and 24, we use the method DTW to calculate the local distances between minima sub-sequences and maxima sub-sequences, respectively.

5.3.4 Computational complexity

As shown in Algorithm in Fig. 5.5, our proposal is composed of three principal steps: local-extrema extraction, local-extrema separation, and similarity establishment. The complexity analysis of the LE-DTW algorithm shows that its theoretical time complexity $TC(LE-DTW)$ is equal to the sum of the time complexity of each step $C(S_i)$.

- **Local-extrema extraction:** this procedure traverses the original time series from the first

Algorithm **LE-DTW**

Inputs: Q,C: Time series;
Output: Sim: Similarity between Q and C;

Begin
$Q_{min}(1) \leftarrow Q(1)$;
$Q_{max}(1) \leftarrow Q(1)$;
for i ← 2 to length(Q)-1
  if (Q(i) < Q(i-1) and Q(i) <= Q(i+1)) or (Q(i) <= Q(i-1) and Q(i) < Q(i+1)) **then**
    $Q_{min} \leftarrow$ cat(Q_{min},Q(i));% Concatenation function
  elseif ((Q(i) > Q(i-1) and Q(i) >= Q(i+1)) or (Q(i) >= Q(i-1) and Q(i) > Q(i+1)))
    $Q_{max} \leftarrow$ cat(Q_{max},Q(i));
  end if
end for
$Q_{min} \leftarrow$ cat(Q_{min},Q(end));
$Q_{max} \leftarrow$ cat(Q_{max},Q(end));
$C_{min}(1) \leftarrow C(1)$;
$C_{max}(1) \leftarrow C(1)$;
for i ← 2 to length(C)-1
  if (C(i) < C(i-1) and C(i) <= C(i+1)) or (C(i) <= C(i-1) and C(i) < C(i+1)) **then**
    $C_{min} \leftarrow$ cat(Q_{min},C(i));
  elseif ((C(i) > C(i-1) and C(i) >= C(i+1)) or (C(i) >= C(i-1) and C(i) > C(i+1)))
    $C_{max} \leftarrow$ cat(C_{max},C(i));
  end if
end for
$C_{min} \leftarrow$ cat(C_{min},C(end));
$C_{max} \leftarrow$ cat(C_{max},C(end));
$ld^{min} \leftarrow$ DTW(Q_{min},C_{min});
$ld^{max} \leftarrow$ DTW(Q_{max},C_{max});
Sim ← ld^{min} + ld^{max};
Return Sim;
End.

Figure 5.5: Pseudo-code of the proposed LE-DTW algorithm.

to the last point and detects the local-extrema. It takes linear time with the length of the original time series, $C(S_1) = O(n)$.

- **Extrema separation:** this procedure is also done in linear time, but here in the length of the generated approximate representation. In fact, local extrema extraction and extrema separation procedures can be done at the same time in the single procedure as shown in Algorithm in Fig. 5.5. So, $C(S_1, S_2) = O(n)$.

- **Similarity establishment:** this step is composed of two functions: local distance, and global distance. The first is certainly the step that consumes much CPU time in all the proposed LE-DTW algorithm due to the dynamic programming approach used in DTW. In the first function, we apply DTW two times, therefore, its complexity is $C(S_3) = O(l^2) + O(h^2)$, where l and h are the lengths of the generated representations of Q' and C', respectively (for simplicity and without any loss of generality, we assume that minima

sub-sequences have the same length l and maxima sub-sequences are also of equal length l). It is very important to indicate here that the two local distances are independent of each other. Therefore, we can simultaneously execute them on a parallel architecture to yet accelerate the LE-DTW computation. The second function is a simple addition operation which takes constant time $C(S_4) = O(1)$.

Consequently, $TC(LE-DTW) = C(S_1,S_2)+C(S_3)+C(S_4) = n+l^2+h^2+1 \approx l^2+h^2$. Suppose that $l = h$, so, $TC(LE-DTW) = O(2*l^2)$, where $l << n$. In conclusion, the computational complexity of the LE-DTW algorithm is of quadratic order in the length of the approximate representations. Therefore, it is better than DTW complexity which is $O(n^2)$.

5.4 Experimental evaluation

5.4.1 Methodology

We evaluate the performance of LE-DTW on the classification problem using the 1-NN classifier. As earlier mentioned, the accuracy of 1-NN highly depends on the choice of the similarity measure, which makes it a very good way to verify the effectiveness of similarity measures. In our experiments, we use the public UCR time series repository (Dau et al., 2018) for evaluation.

We differentiate two particular experiments. In the first one, we compare 1-NN LE-DTW against 1-NN DTW as a reference method and also against popular distance-based methods, on long time series. The second experiment consists in comparing our proposed with 1-NN DTW on short time series. 1-NN DTW often serves as a benchmark to verify the effectiveness of time series classifiers (Bagnall et al., 2017).

For reference, the classification error rate (or equivalently, accuracy) is limited to the interval [0, 1] and it is obtained as follows:

$$\text{Error rate} = \frac{\text{Size of the testing test} - \text{Number of correctly classified TS}}{\text{Size of the testing test}} \tag{5.13}$$

All experiments LE-DTW and DTW have been done on a machine with the following charac-

teristics: Intel(R) Core(TM) i3 CPU M370 @ 2.40 GHz 2.40 GHz, RAM 4GO, Windows 7 64 bits, Matlab R2014a 64 bits. However, the other classification accuracies have been obtained from, (Dau et al., 2018) for ED and DTW (with and without a warping window constraint); (Sharabiani et al., 2017) for 3D-DTW and BOSS VS; and (Yuan et al., 2019b) for the remaining algorithms examined in these experiments.

5.4.2 Long time series

In this section, we test our method against different similarity measures for long TSC. First, we compare LE-DTW against DTW in terms of efficiency and effectiveness. Second, we compare LE-DTW against relevant distance-based classifiers in terms of accuracy.

5.4.2.1 LE-DTW vs. DTW

In this work, we regard time series equal or longer than 1000 temporal points to be "long" (Lang et al., 2009). In this experiment, we use all long time series datasets exiting in the UCR archive, which are 21 datasets. The description of these datasets is provided in Table 5.1.

Table 5.1 presents the classification error rates of both LE-DTW and DTW methods on 21 long time series datasets from the UCR benchmark. Results show that our proposed measure outperforms the DTW method, which provides better effectiveness on most datasets. Exactly, LE-DTW is more accurate than DTW on 13 out of 21 datasets, and worse on 8 datasets. Results also indicate that the accuracy is improved by more than 10% on 6 datasets. Especially, for *PigArtPressure*, *PigCVP*, and *SemgHandMovementCh2* datasets, the accuracy improvement is very large (34%, 25%, and 18%, respectively). In contrast, in the case where our LE-DTW method is worse, the rate of loss is practically negligible (less than 3% on the majority of datasets). On average, our proposed is over 5% more accurate than the DTW method. It is useful to note here that our LE-DTW method is slightly sensitive to the noise. Accordingly, the poor performance of our method on some datasets is probably justified by the presence of noise in these datasets.

To verify whether there is a statistical difference between the classification accuracies of our

Table 5.1: Classification error rates of both LE-DTW and DTW methods on 21 long time series datasets from the UCR benchmark (Lahreche and Boucheham, 2021b).

Dataset	**Length**	**LE-DTW**	**DTW**
ACSF1	1460	**0,2000**	0,3600
CinCECGTorso	1639	**0,3152**	0,3493
EOGHorizontalSignal	1250	0,5111	**0,4972**
EOGVerticalSignal	1250	0,5580	**0,5525**
EthanolLevel	1751	0,7300	**0,7240**
HandOutlines	2709	0,3324	**0,1189**
Haptics	1092	**0,5520**	0,6234
HouseTwenty	2000	0,1429	**0,0756**
InlineSkate	1882	**0,5909**	0,6164
Mallat	1024	0,0810	**0,0661**
MixedShapesRegularTrain	1024	**0,0920**	0,1584
MixedShapesSmallTrain	1024	**0,1414**	0,2202
Phoneme	1024	0,7996	**0,7716**
PigAirwayPressure	2000	**0,8029**	0,8942
PigArtPressure	2000	**0,4135**	0,7548
PigCVP	2000	**0,5962**	0,8462
Rock	2844	0,5800	**0,4000**
SemgHandGenderCh2	1500	**0,0900**	0,1983
SemgHandMovementCh2	1500	**0,2356**	0,4156
SemgHandSubjectCh2	1500	**0,1644**	0,2733
StarLightCurves	1024	**0,0553**	0,0934
Average		**0,3802**	0,4290

proposed LE-DTW and DTW methods, we perform the *Wilcoxon signed rankstest* (Wilcoxon, 1992) which is recommended in (Demšar, 2006; Garcia and Herrera, 2008). With a confidence level of $(\alpha = 0,05)$ and $N = 21$ (Number of datasets), and from the table of exact critical values for the Wilcoxon signed ranks test, the null hypothesis (both methods perform equally) is rejected. Therefore, LE-DTW and DTW methods are not significantly different. However, the major advantage of our method is its efficiency, which is several times faster than DTW as we will show in the next sub-section.

On the other hand, the execution time is also an important metric for assessing algorithms for TSC (Bagnall et al., 2017). In this sub-section, we are interested in empirically testing the efficiency of our proposed against DTW on long time series datasets. The execution time $ET(.)$ is used to verify the efficiency of both algorithms. It is calculated as follows: for each long time series dataset, we randomly select 10 long time series from the testing set. Next, for

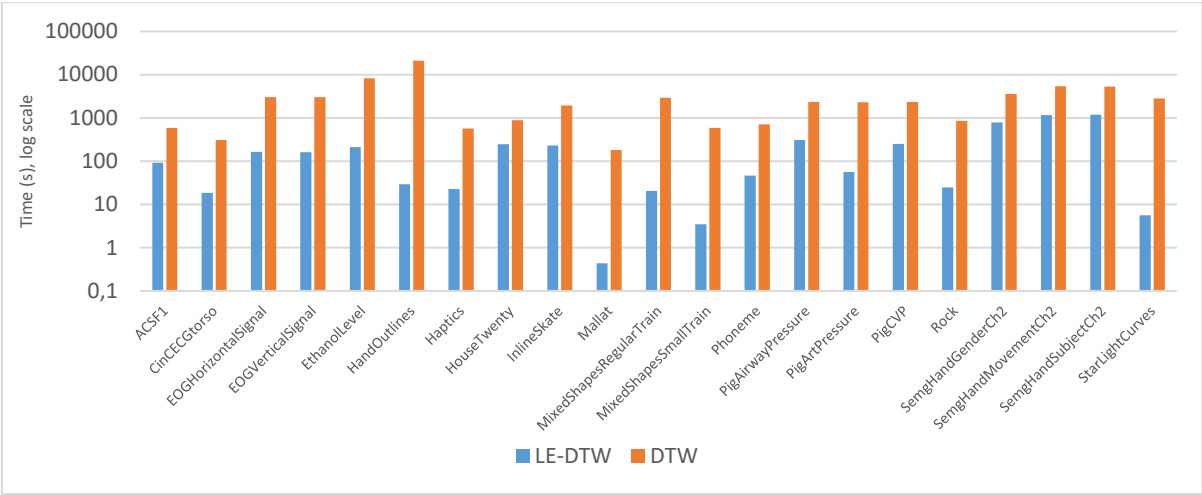

Figure 5.6: Classification execution time in seconds of both LE-DTW and DTW methods on 21 long time series datasets from UCR archive (Lahreche and Boucheham, 2021b).

each long time series, we record the CPU time consumption for classifying it, $TC_i(.)$. Finally, the averaged time consumption is taken:

$$ET(.) = \frac{1}{10}\sum_{i=1}^{10} TC_i \tag{5.14}$$

Figure 5.6 illustrates the classification execution time in seconds (s) of both LE-DTW and DTW methods on 21 long time series datasets. From results, we can easily see two things: 1) First, our proposed has the lowest execution time on all long time series datasets. 2) Second, the difference in execution time between the two methods is very large. Hence, our LE-DTW method is more efficient than DTW.

We can distinguish three levels of superiority in terms of efficiency of our proposal compared to the DTW method as follows: *hyper-fast*, *super-fast*, and *very fast*. In the first level, LE-DTW is more than 100 times faster than DTW. Therefore, LE-DTW is hyper-fast on 5 out of 21 datasets. As an example, for the *HandOutlines* dataset, LE-DTW takes almost 30s to classify a single time series, while DTW takes more than 5 hours (about 720 times slower). In the second level, we consider our method super-fast, if it is 10 up to 100 times faster than DTW. LE-DTW is super-fast on 8 out of 21 datasets. As an example, for the *PigArtPressure* dataset, the execution time of LE-DTW and DTW are approximately 56s and 38 minutes (m), respectively (40 times faster). For the third level, LE-DTW is less than 10 times faster than DTW. Therefore, from 21

datasets, LE-DTW is very fast on 8. On average, the proposed LE-DTW method is up to 105 times faster than the DTW method.

In summary, we experimentally showed trough this study that for long time series, our LE-DTW method provides better performance to that of the famous DTW on both accuracy and efficiency. Specifically, our method is more accurate but not significantly and several times faster than DTW. Consequently, LE-DTW is a very promising measure and could be an excellent alternative to the popular DTW method for long time series.

5.4.2.2 LE-DTW vs. Distance-based methods

3D-DTW is the most similar similarity measure to our proposed in the literature. It was recently proposed for fast classification of long time series. The authors have evaluated their 3D-DTW method on 28 long time series datasets using the earlier version (Chen et al., 2015a) of the UCR archive. In addition, they regarded time series with a minimum 500 point temporal as long. Here, we compare the effectiveness of our proposed to that of the 3D-DTW method on long time series. For that end, and to perform comparison under the same methodology, we also use the same long time series datasets as in 3D-DTW (28 datasets).

As LE-DTW is a distance-based method, we also compare it against relevant distance-based methods. The distance-based methods examined in this work are the following: LCSS, MSM, TWED, DTW, DTW(BWW) (Ratanamahatana and Keogh, 2004), WDTW, DDTW, BOSS VS (Schäfer, 2016), and ED.

Classification error rates comparison between LE-DTW and popular distance-based classifiers are given in Table 5.2. The value in bold indicates the best classification error rate. As seen in Table 5.2, the proposed LE-DTW and MSM methods are the most accurate, which have the highest number of best error rate (10 wins out of 28 datasets). In the second position, we find the DDTW method with 5 wins, followed by TWED with 4 wins. Then, 3D-DTW with 3 wins. Next, they come DTW (BWW), WDTW, and ED with the same number of wins (2 times). In the last places, the LCSS and BOSS VS methods with 1 and 0 win, respectively.

Table 5.2: Classification error rates comparison between LE-DTW and popular distance-based classifiers (Lahreche and Boucheham, 2021b).

Dataset	LE-DTW	3D-DTW	DTW(WW)	DDTW	LCSS	MSM	TWED	WDTW	BOSSVS	ED
BeetleFly	**0,100**	0,150	0,300	0,250	0,200	0,400	0,300	0,300	0,300	0,250
BirdChicken	**0,150**	**0,150**	0,300	**0,150**	0,200	**0,150**	**0,150**	0,250	0,250	0,450
Car	0,350	0,350	0,233	0,267	0,167	0,100	**0,083**	0,217	0,267	0,267
CinCECGtorso	0,314	0,332	0,070	0,289	0,054	**0,038**	0,154	0,141	0,349	0,103
Computers	**0,264**	0,416	0,380	0,332	0,356	0,424	0,368	0,300	0,300	0,424
Earthquakes	**0,217**	0,320	0,258	0,324	0,273	0,309	0,324	0,274	0,258	0,326
FordA	0,394	0,398	0,341	0,283	0,302	**0,281**	0,382	0,347	0,438	0,341
FordB	0,381	0,365	0,414	**0,335**	0,352	0,358	0,391	0,401	0,406	0,442
HandOutlines	0,295	0,196	0,197	0,262	0,151	**0,124**	0,130	0,130	0,202	0,199
Haptics	**0,558**	0,591	0,588	0,714	0,610	**0,558**	0,584	0,630	0,623	0,630
Herring	0,422	**0,391**	0,469	0,500	0,438	0,438	0,484	0,469	0,469	0,484
InlineSkate	0,593	0,738	0,613	**0,547**	0,573	**0,547**	0,578	0,596	0,616	0,658
LargeKitchenAppliances	**0,197**	0,235	0,205	0,232	0,416	0,331	0,248	0,203	0,205	0,507
Lighting2	**0,098**	0,246	0,131	0,328	0,230	0,180	0,164	**0,098**	0,131	0,246
MALLAT	0,082	0,100	0,086	0,103	0,089	0,068	0,088	**0,062**	0,066	0,086
NonInvasiveFatalECGThorax1	0,317	0,509	0,185	0,303	0,188	0,184	0,180	0,184	0,210	**0,171**
NonInvasiveFatalECGThorax2	0,219	0,416	0,129	0,174	0,117	0,117	**0,112**	0,116	0,135	0,120
OliveOil	**0,133**	0,233	**0,133**	**0,133**	0,600	0,167	**0,133**	0,167	0,167	**0,133**
Phoneme	0,800	0,748	0,773	0,744	0,744	**0,708**	0,730	0,839	0,772	0,891
RefrigerationDevices	**0,475**	0,517	0,560	0,592	0,509	0,517	0,488	0,573	0,536	0,605
ScreenType	**0,517**	0,565	0,589	0,573	0,563	0,541	0,573	0,589	0,603	0,640
ShapeletSim	0,450	**0,078**	0,300	0,461	0,178	0,133	0,422	0,244	0,350	0,461
ShapesAll	0,270	0,313	0,198	0,165	0,155	**0,128**	0,140	0,188	0,232	0,248
SmallKitchenAppliances	0,347	0,355	0,328	0,347	0,528	**0,301**	0,355	0,325	0,357	0,659
StarLightCurves	0,055	0,095	0,095	**0,038**	0,121	0,132	0,117	0,105	0,093	0,151
UWaveGestureLibraryAll	0,125	0,091	**0,034**	0,423	0,311	0,300	0,316	0,324	0,108	0,052
Worms	0,541	0,508	0,586	0,416	**0,403**	0,429	0,429	0,468	0,536	0,635
WormsTwoClass	0,343	0,387	0,414	0,364	0,351	**0,325**	0,390	0,429	0,338	0,414
Wins	**10**	3	2	5	1	**10**	4	2	0	2

For deeper evaluation, we separately compare our method against each of the distance-based classifiers. Figure 5.7 (scatter plot) illustrates the pair-wise classification accuracies comparison between LE-DTW and Fig. 5.7(a) 3D-DTW, Fig. 5.7(b) DTW (BWW), Fig. 5.7(c) LCSS, Fig. 5.7(d) MSM, Fig. 5.7(e) TWED, Fig. 5.7(f) WDTW, Fig. 5.7(g) DDTW, Fig. 5.7(h) BOSS VS and Fig. 5.7(i) ED. Each point in the scatter plot represents a dataset. The points locate in the withe region indicate that LE-DTW performs better on these datasets.

From Fig. 5.7, it is remarkable that LE-DTW outperforms 3D-DTW and ED methods. Compared with DTW(BWW), which has been considered as the benchmark for comparison in nearly all TSC works (Lines and Bagnall, 2015), our method achieves better accuracy on 16 datasets, ties on one, and worse on 11. LE-DTW is slightly superior to LCSS, DDTW, and TWED. For WDTW, MSM, and BOSS VS methods, which have been found that are significantly better than DTW (Bagnall et al., 2017; Schäfer, 2016), LE-DTW is slightly better than WDTW, better than BOSS VS and slightly worse than MSM method.

For statistical pairwise comparison, we again use the Wilcoxon signed ranks test. With a confidence level of $(\alpha = 0,05)$ and $N = 28$ (Number of datasets) for all methods, and from to

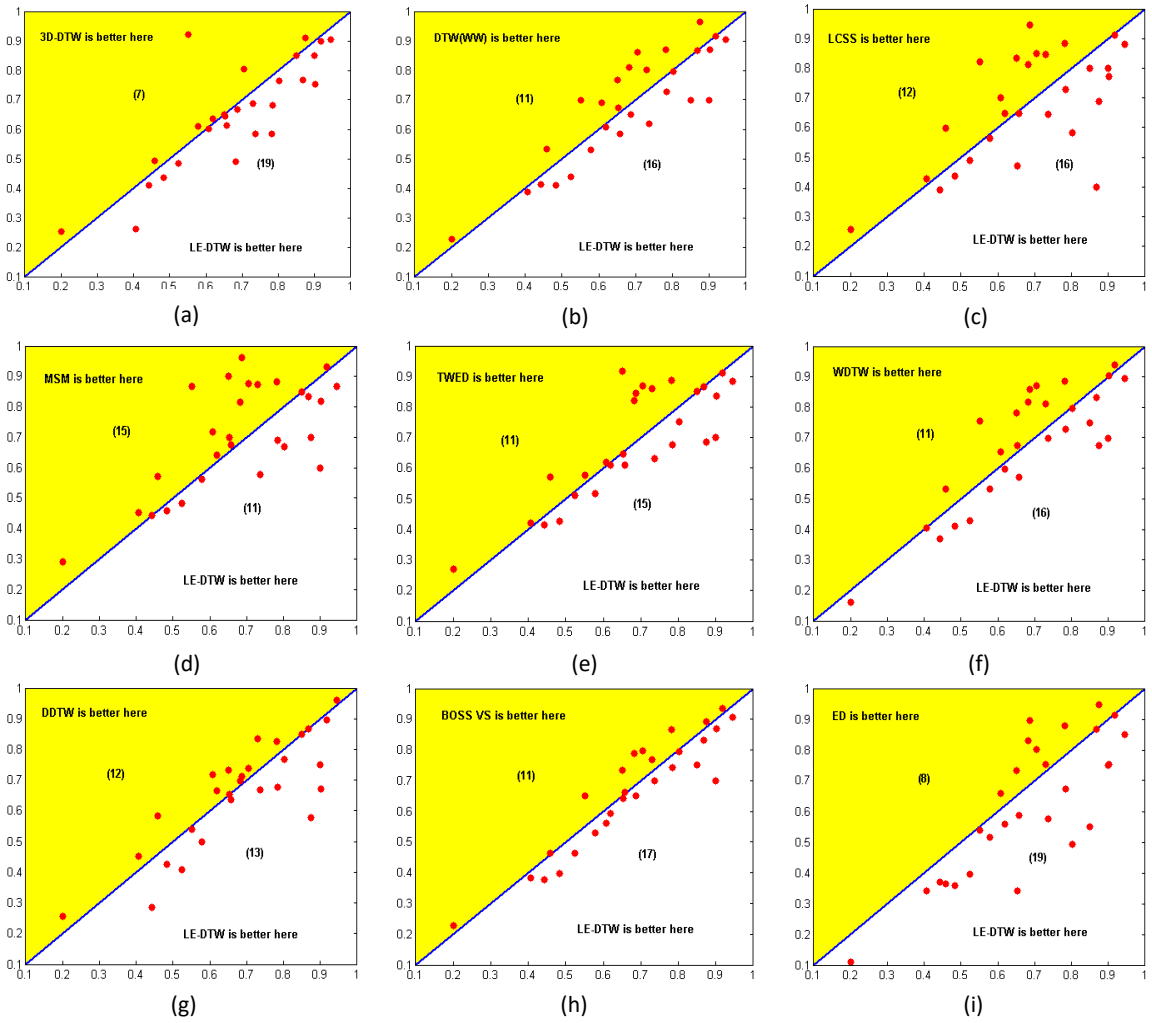

Figure 5.7: Pairwise classification accuracies comparison between LE-DTW and relevant distance-based classifiers (Lahreche and Boucheham, 2021b).

the table of exact critical values for the Wilcoxon signed ranks test, we found that the null hypothesis is rejected for ED and 3D-DTW and not for the others methods. Therefore, LE-DTW is significantly more accurate than ED and 3D-DTW, and there is no significant difference between our proposed and the others.

To summarize, our method is significantly better than 3D-DTW and ED, better but not significantly than TWED, WDTW, DDTW, DTW(BWW), LCSS, and BOSS VS. Furthermore, LE-DTW is slightly worse than MSM.

All aforementioned distance-based methods are based on the dynamic programming (DP) approach, except ED and BOSS VS methods. Moreover, they work directly on the raw time series

Table 5.3: Description of 69 short time series datasets taken from UCR archive (Lahreche and Boucheham, 2021b).

DATASETS	LENGTH	DATASETS	LENGTH	DATASETS	LENGTH
Adiac	176	FISH	463	ProximalPhalanxOutlineAgeGroup	80
ArrowHead	251	FreezerRegularTrain	301	ProximalPhalanxOutlineCorrect	80
Beef	470	FreezerSmallTrain	301	ProximalPhalanxTW	80
BME	128	Fungi	201	SmoothSubspace	15
CBF	128	GunPoint	150	SonyAIBORobotSurface1	70
Chinatown	24	GunPointAgeSpan	150	SonyAIBORobotSurface2	65
ChlorineConcentration	166	GunPointMaleVersusFemale	150	Strawberry	235
Coffee	286	GunPointOldVersusYoung	150	SwedishLeaf	128
CricketX	300	Ham	431	Symbols	398
CricketY	300	InsectWingbeatSound	256	syntheticControl	60
CricketZ	300	ItalyPowerDemand	24	ToeSegmentation1	277
DiatomSizeReduction	345	Lighting7	319	ToeSegmentation2	343
DistalPhalanxOutlineAgeGroup	80	Meat	448	Trace	275
DistalPhalanxOutlineCorrect	80	MedicalImages	99	TwoLeadECG	128
DistalPhalanxTW	80	MelbournePedestrian	24	TwoPatterns	82
ECG200	96	MiddlePhalanxOutlineAgeGroup	80	UMD	150
ECG5000	140	MiddlePhalanxOutlineCorrect	80	uWaveGestureLibraryX	315
ECGFiveDays	136	MiddlePhalanxTW	80	uWaveGestureLibraryY	315
ElectricDevices	96	MoteStrain	84	uWaveGestureLibraryZ	315
FaceAll	131	OSULeaf	427	wafer	152
FaceFour	350	PhalangesOutlinesCorrect	80	Wine	234
FacesUCR	131	Plane	144	WordsSynonyms	270
Fiftywords	270	PowerCons	144	yoga	426

without performing any dimensionality reduction technique as a preliminary step, excluding 3D-DTW and BOSS VS. In fact, even our method is based on DP, but it considerably reduces the dimensionality of the input time series. In this regard, it is evident that our proposal is faster than LCSS, MSM, TWED, DDTW, and WDTW methods. For 3D-DTW, DTW(BWW), and BOSS VS methods, they could exceed our LE-DTW method in some cases in terms of efficiency. Nevertheless, they require many parameters to be involved in the training phase, in contrast to our technique, which is parameter-free. Regarding the ED method, it is certainly faster than our method due to its linear time complexity.

5.4.3 Short time series

As previously discussed, our method is particularly designed for long time series. However, we aim in this section to briefly see how it behaves on short time series. For that, we compare the efficiency and effectiveness of LE-DTW to those of the DTW method. Here, we consider time series shorter than 500 time points to be "short". In these experiments, we use 69 short time series datasets coming from the UCR archive for evaluation. The description of these short time series datasets is presented in Table 5.3.

The classification accuracies of LE-DTW against DTW over the 69 short time series datasets

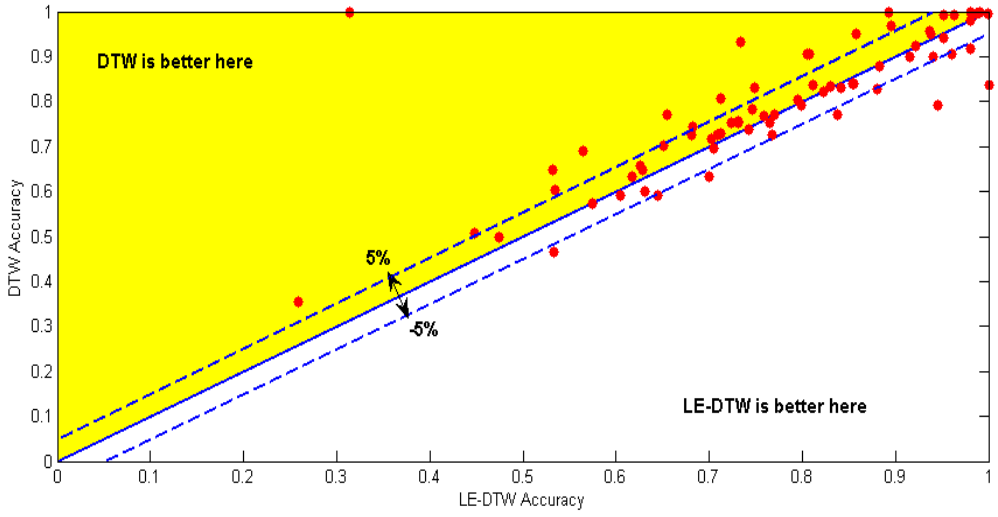

Figure 5.8: Classification accuracies of LE-DTW against DTW over 69 short time series datasets from UCR archive (Lahreche and Boucheham, 2021b).

are illustrated in Fig. 5.8. From the scatter plot, it is clear that most of the points localize inside the DTW's region, which means that our proposed method has the lowest classification accuracy on the majority of datasets. Statistically, the LE-DTW wins on just 25 out of 69 datasets, ties on 2, and loses on 42. However, we observe that the accuracy of the two methods is very near to each other on most the datasets. Concretely, the accuracy difference ranges from -5% to 5% on 43 out of 69 datasets. Besides, the proposed LE-DTW is specially designed for long time series.

In addition to comparing the classification accuracy of both LE-DTW and DTW methods, we have also compared them regarding the execution time aspect. We have followed the same methodology as in (Section 5.4.2) to verify their efficiency. The empirical results of this experiment are illustrated in Fig. 5.9. It is easy to discover that the execution time of our method is less than that of DTW on all datasets. We can also observe that the execution time of LE-DTW is lower to 1s on most of the datasets (40/69), while there are only 7 datasets on which the DTW's execution time is lower to 1s. The min and max execution times of LE-DTW are around 0,0093s and 29s for *Chinatown* and *ElectricDevices* datasets respectively. In contrast, for DTW are around 0,0351s and 461s for *Chinatown* and *UWaveGestureLibraryZ*, respectively. On average, the proposed LE-DTW method is up to 41 times faster than the DTW

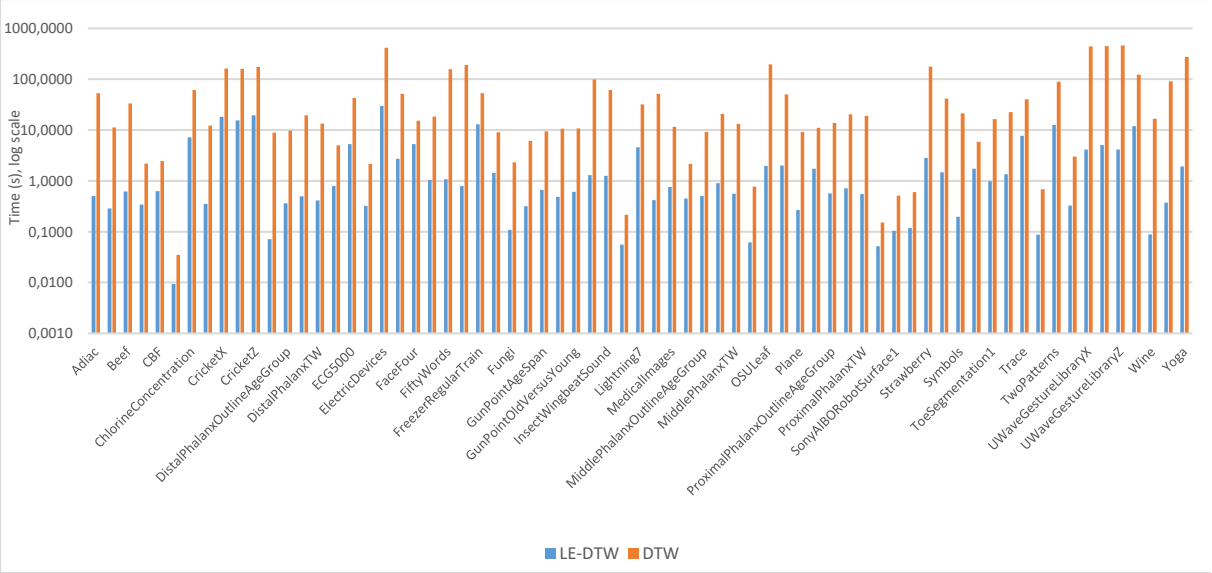

Figure 5.9: Classification execution time in seconds of both LE-DTW and DTW methods on 69 short time series datasets from UCR archive (Lahreche and Boucheham, 2021b).

method.

To summarize, although our proposed LE-DTW is worse against DTW in terms of accuracy, it always gains in terms of efficiency which is several times faster than DTW. Moreover, it is interesting to note that in most practical applications, faster methods with acceptable accuracy are preferred than slow methods with a slight difference in accuracy (Sharabiani et al., 2017).

5.5 Conclusion

In this chapter, a new similarity measure, LE-DTW, is proposed for fast and accurate classification of long time series. The newly proposed LE-DTW is intuitive and parameter-free. LE-DTW is mainly based on two strategies: time series representation and similarity establishment. In the first step, the original time series is represented according to its local extrema points. Next, minima and maxima are physically separated to consistently represent time series. Second, we adapt the DTW measure so as to evaluate the score of similarity between the two generated approximate representations. Extensive experiments have been conducted to verify the performance of the proposed method on a broad range of real-world datasets from the UCR repository for classification. Results show that LE-DTW achieves improved classification ac-

curacy, especially on long time series where it is better than DTW, and it is competitive with popular distance-based methods. Moreover, LE-DTW is orders of magnitude faster than DTW.

Chapter 6

Upgrading the SEA method to efficiently align quasi-periodic time series and accurately classify general time series (Lahreche and Boucheham, 2017, 2018)

6.1 Introduction

The Shape Exchange Algorithm (SEA) is proposed by Boucheham (2008) to handle the problem of Quasi-Periodic Time Series (QPTS) matching. In this context, the method has proved its worth, and was able to be a competitive rival against the famous DTW method. However, the SEA method, on the other hand, has two key drawbacks. First, the temporal complexity of the SEA method is quadratic $O(n^2)$, which makes it impractical in aligning very long time series. Second, the usefulness of SEA is more marked in the case of QPTS. Specifically, for general series, the method provides poor accuracy within the TSC task. Figure 6.1 shows the underperformance of SEA against the simple ED in terms of classification accuracy over 85 UCR time series datasets.

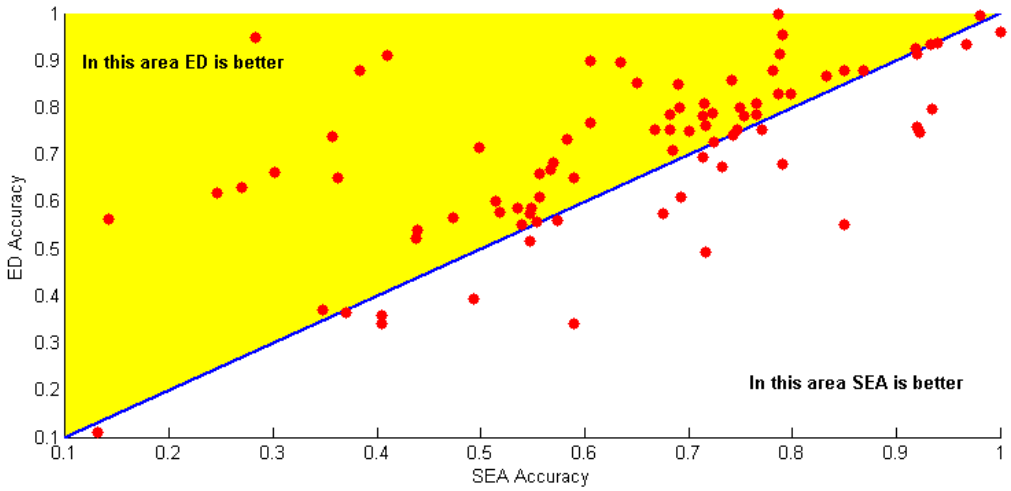

Figure 6.1: Classification accuracy comparison between 1-NN SEA and 1-NN ED over 85 UCR datasets.

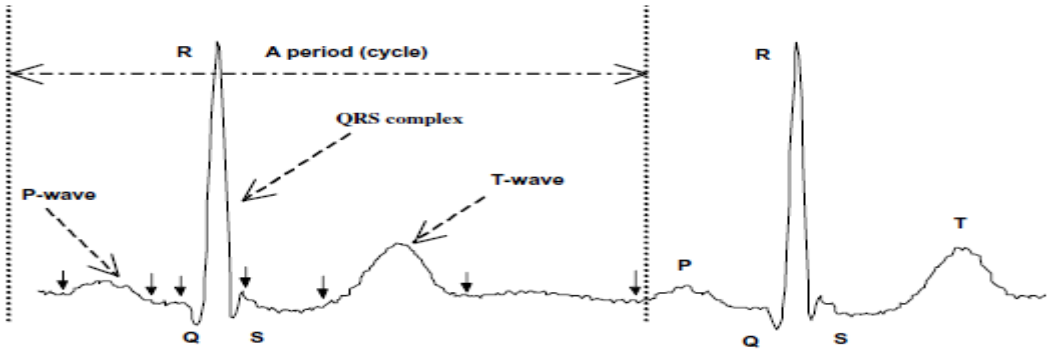

Figure 6.2: Typical quasi-periodic ECG time series with two periods (Boucheham, 2008).

In this chapter, we focus on upgrading the performance of the SEA method by specifically addressing its aforementioned weaknesses. To this end, we have proposed two contributions. In the first contribution, we are interested in accelerating the SEA so that it can match very long QPTS in very short timescales. The new method is named FastSEA and has linear computational complexity. Whereas, in the second contribution, we aim to extend the effectiveness of SEA to general time series in the context of TSC.

6.2 Shape Exchange Algorithm (SEA)

The SEA method is developed as a rival method of DTW for the problem of QPTS matching. This particular type of time series is composed of repeated quasi-similar patterns. In practice,

it is more difficult to match this kind of time series because they are eventually composed of a different number of periods, of different lengths, and that might be shifted/scaled on the time/amplitude axis (Boucheham, 2008). Many real-world time series are quasi-periodic by nature, e.g., ECG, EEG, capnogram signal, etc. Figure 6.2 illustrates a typical quasi-periodic ECG time series with two periods.

The SEA is a parameter-free method and relies particularly on a signatures exchange technique. These signatures are obtained by sorting the two time series to match on their magnitude indexes, respectively. The signature allows defining a global descriptor of the time series and is robust to most distortions that complicate the alignment process (Boucheham, 2008). Indeed, similar time series with significant distortions will still have similar signatures upon application of the SEA method.

Once both signatures are constructed from the two time series, the SEA performs an exchange between them. Then, the two time series are reconstructed by sorting them again on their natural time indices. Finally, the author proposes to compare the original time series with its reconstructed version using the normalized Percent Root Difference (PRD) (Eq. 6.1) and the Correlation (Corr) (Eq. 6.2) measures.

$$PRD(Q,C) = \sqrt{\frac{\sum_{i=1}^{n}(q_i - c_i)^2}{max(\sum_{i=1}^{n}(q_i - \bar{q}_i)^2, \sum_{i=1}^{n}(c_i - \bar{c}_i)^2)}} * 100\% \tag{6.1}$$

$$Corr(Q,C) = \frac{cov(Q,C)^2}{var(Q), var(C)} \tag{6.2}$$

Note that, if the two time series have different lengths, a preliminary step is needed which consists in forcing the longer time series to shrink to the length of the shorter time series and the shorter to expand to the length of the longer time series in a linear fashion. Figure 6.3 summarizes the different steps of the SEA method.

The SEA has proved its superiority over the more popular DTW in terms of precision, consumed time, and memory in the context of QPTS matching (Boucheham, 2008). Despite this, SEA has been improved over the years in terms of both efficiency and effectiveness aspects.

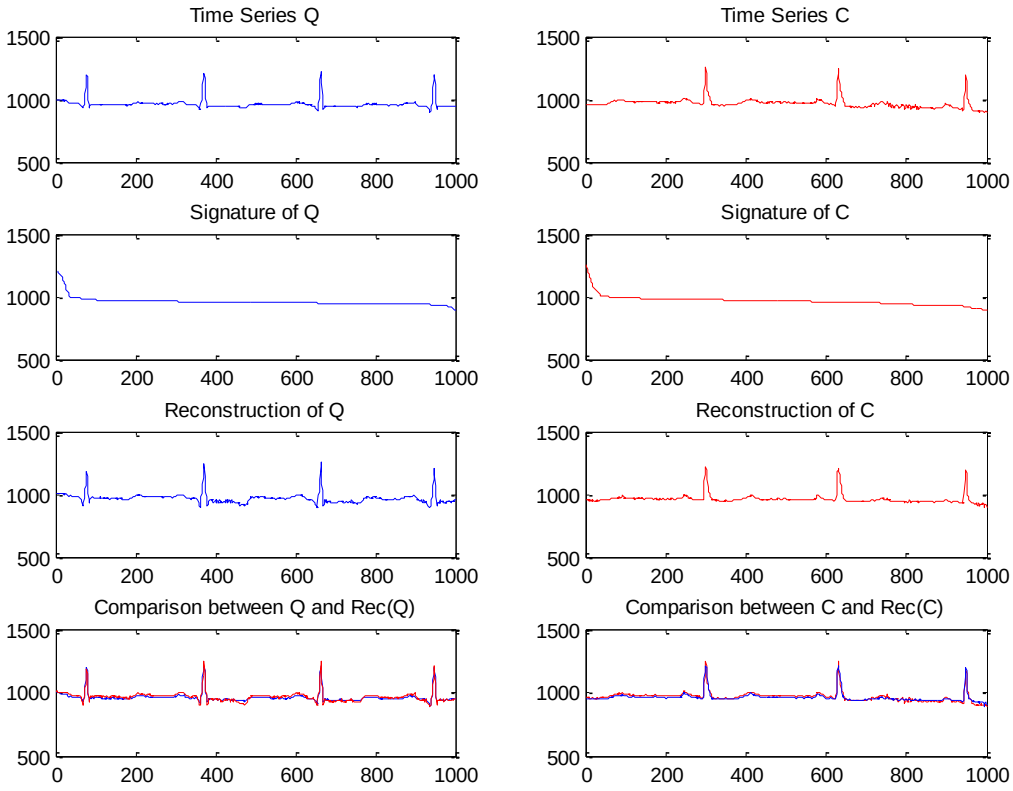

Figure 6.3: Illustration of the different steps of SEA. From top to down, two time series Q (left) and C (right), their signatures respectively, reconstruction process, and comparison.

For the efficiency aspect, the same author of SEA relied on the advantage of the data reduction technique and presented two much faster methods named ASEAL (Boucheham, 2010) and FANSEA (Boucheham, 2013). Nevertheless, these methods fail to keep the exact alignment quality as in the original method. In our first proposal, we aim to speed up the SEA without any loss of precision. On the other hand, Betancourt et al. (2013) and Boulnemour and Boucheham (2015) propose Fuzzy-SEA and I-SEA, respectively in order to enhance the effectiveness of the SEA method. However, their works are exclusively designed for the specific type of QPTS. In the second proposition, we explore the possibility of making SEA applicable in all types of time series within TSC.

6.3 FastSEA (Lahreche and Boucheham, 2017)

In this section, we present our first contribution for efficiently matching QPTS. The proposed method is referred to by the acronym FastSEA. FastSEA is based on an accelerated version of the SEA method. The motivation behind speeding up SEA is, on one hand, its high effectiveness and, on the other hand, the quite limited power of SEA in aligning long time series in acceptable times due to its high computational complexity. Therefore, the main objectives of this work are: (a) significantly accelerate the SEA method (b) without any loss in precision.

The complexity analysis of the SEA method shows that its main quadratic nature is due to the used Selection Sort Algorithm in one of its steps. Therefore, in this study, we have accelerated SEA by considering a more efficient sorting algorithm. The literature on sorting algorithms shows that the Count Sort Algorithm is distinguished by its exceptional linear order complexity. However, this algorithm is applicable only for data comprised within a known [0..k] range. Fortunately, this is the case of most time series and signals due to quantization and coding requirements. From another perspective, the selected Counting Sort Algorithm, besides it being theoretically faster than existing comparison based sorting algorithms, is simpler than the Selection Sort Algorithm used in the original SEA technique.

The direction towards sorting algorithms for speeding up SEA is further illustrated by the results shown in Table 6.1 and Fig. 6.4. The experiment has been performed on different lengths of ECG signal (rec 100) taken from the MIT-BIH Arrhythmia Database (Goldberger et al., 2000; Moody and Mark, 2001), by determining the execution time in seconds (s) of the main procedures of SEA: Sorting process (sorting on the magnitude and sorting on time), Shape Exchange procedure and PRD and Correlation factor. The results show clearly that more than 94% of the SEA time is consumed by the Sorting process. The results of Table 6.1 have been transformed to Fig. 6.4 for better exhibition.

Table 6.1: The execution time of the main procedures of the SEA method (Lahreche and Boucheham, 2017)

TS Length	Sorting Process (s)	Shape Exchange (s)	PRD and Corr (s)	Total Excution Time (s)	Sorting Process (%)
500	0.298	0.003	0.162	0.463	64.363
1000	1.291	0.006	0.335	1.632	79.105
2000	4.657	0.005	0.498	5.16	90.252
3000	9.258	0.002	0.613	9.873	93.771
5000	25.519	0.003	1.398	26.92	94.796

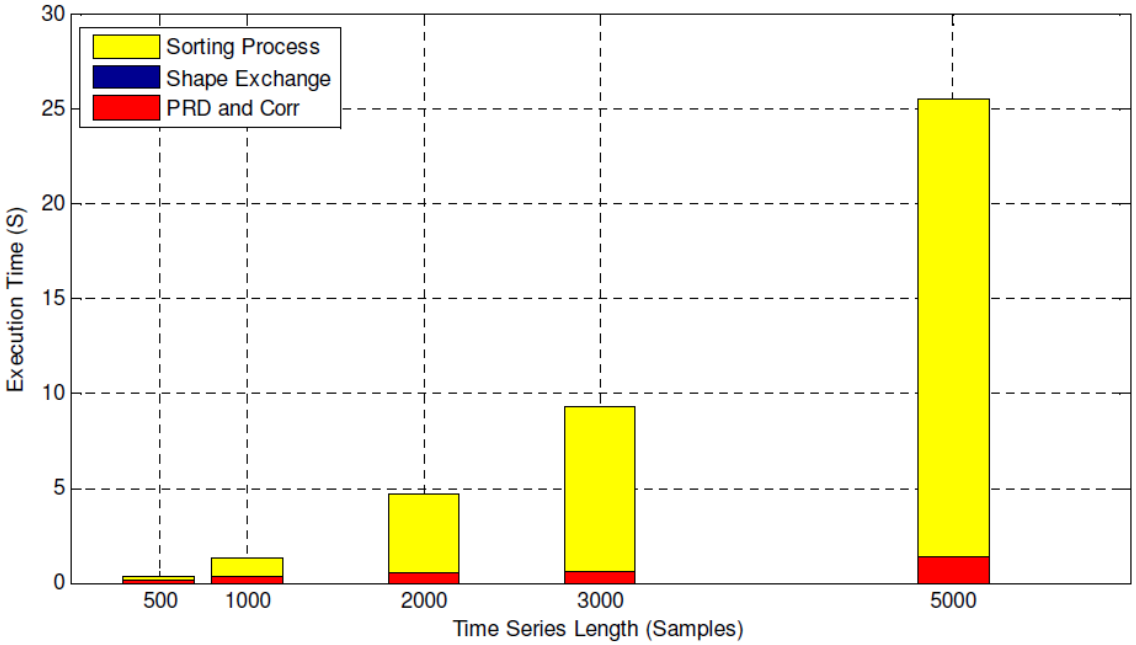

Figure 6.4: Execution time (second) of the main procedures of the SEA method (Lahreche and Boucheham, 2017)

6.3.1 Counting Sort Algorithm

The idea behind the Counting Sort Algorithm is that it is suitable for situations in which several elements have the same value. Luckily, this is the case with QPTS. Indeed, to all evidence, in QPTS, the longer the series, the more we have repetitive equal values for samples within different periods. The basic idea behind this sorting algorithm is to find out the number of elements "smaller than" for each input element. Now, to sort elements using counting sort, we need three arrays: Input array A[1..N], Output array B[1..M], and C[1..K] for temporary working storage. Figure 6.5 shows the pseudo-code of the Counting Sort Algorithm.

Algorithm Counting Sort

Inputs: A: Array of length N;
C: Array of length K;
Output: B: Array of length M;
Begin
1. For i ← 0 to K
2. C(i) ← 0;
3. End for
4. For j ← 1 to N
5. C(A(j)) ← C(A(j)) + 1;
6. End for
7. For i ← 1 to K
8. C(i) ← C(i) + C(i-1);
9. End for
10. For j ← N to 1
11. B(C(A(j))) ← A(j);
12. C(A(j)) ← C(A(j)) – 1;
13. End for
End.

Figure 6.5: Pseudo-code of the Counting Sort Algorithm.

6.3.2 FastSEA method

Like the original SEA, the proposed FastSEA for QPTS alignment is composed of three main steps: Signature establishment, Shape exchange, and Matching. This is due to the fact that only the sorting technique has been upgrared in FastSEA with regard to that in SEA.

- **Step 1 (Signature establishment):** Let Q and C two time-series to match. In this step, the Counting Sort Algorithm is applied to sort ascending or descending both time series on the magnitude values. The outputs of this step are referred to as the signature of Q and the signature of C.

- **Step 2 (Shape exchange):** In this step, an exchange of the shape between the two time series Q and C is performed. The time series Q receives the signature of the time series C while conserving its proper temporal values, and vice versa, the time series C will receive the signature of the time series Q while conserving its proper temporal values.

- **Step 3 (Reconstruction and Matching):** The reconstruction operation consists in re-arranging the two time series Q and C in their natural order by sorting them on the temporal index. Finally, the comparison between Q and C is performed through PRD and Correlation measures.

6.3.3 Theoretical time and space complexity

The main procedures of FastSEA are sort-on-magnitude, sort-on-temporal-index, and comparison where PRD and Corr measures are used. In FastSEA, the used Counting Sort Algorithm has a linear $(O(n))$ temporal complexity because it is not a comparison-based sort. The PRD and Corr procedures have also a linear temporal complexity. Therefore, the overall FastSEA theoretical temporal complexity is of linear order $(O(n))$. The space complexity is, likewise, of linear complexity $(O(n))$, since the main data are proportional to the sizes of the time series n. This is a huge complexity reduction of the SEA method from quadratic $(O(n^2))$ to linear $(O(n))$.

6.3.4 Experimental results and discussion

To evaluate the performance of our newly developed FastSEA method, we have applied it to solve the problem of QPTS alignment. In this study, we particularly focus on ECG signals selected from the public MIT-BIH Arrhythmia Database. The ECG reflects the electrical activity of the heart. Besides its quasi-periodicity, the motivation behind choosing the ECG is based on the fact that the ECG is widely used to diagnose many diseases and has many important applications, such as abnormality detection, heartbeat classification, and person identification. Note that the MIT-BIH Arrhythmia Database includes 48 half-hour ECG recording segments at a sampling rate of 360 Hz. Fig. 6.6 shows three different ECG records (rec 100, rec 102, and rec 103) selected from that database.

FastSEA has been evaluated in comparison with SEA in terms of efficiency and effectiveness and this visually and numerically using PRD and Corr factors. All applications in this study have been carried out on a computer with the following characteristics: PC Intel (R) Core (TM) i3 CPU, 2.40 GHz, 4Go of main memory, OS Windows 7 (64 bits), and MATLAB 8.1.0.604 development environment.

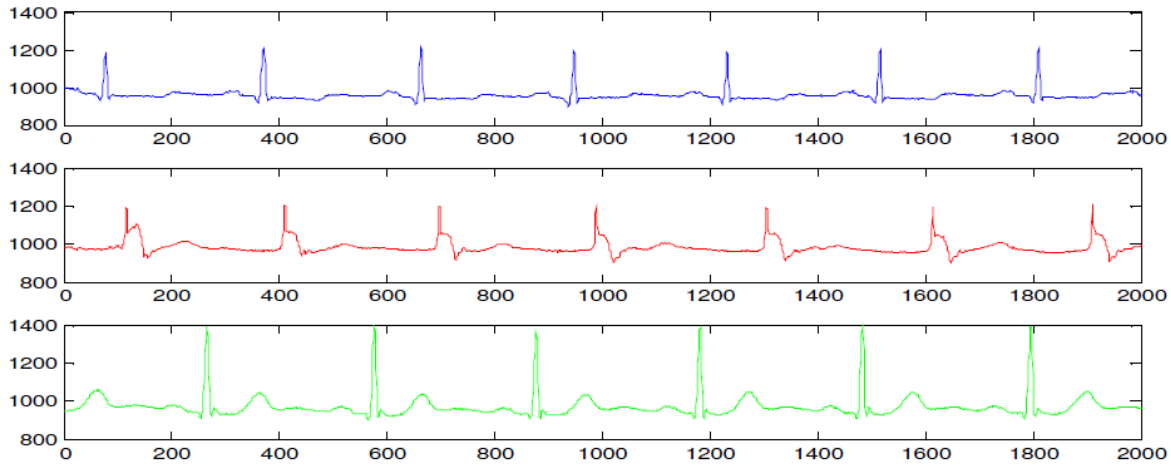

Figure 6.6: An Example of Three ECG records (Top: rec 100, Middle: rec 102 and Bottom: rec 103) selected from MIT-BIH public database.

6.3.4.1 Application 1

In this experiment, we show the effect of time series length on the execution time for both methods FastSEA and SEA. The record 100 of the MIT-BIH database has been selected for this purpose. The execution time and reduction time have been determined by applying a range of lengths between 1000 and 100000 samples for the two methods. The results of this application are shown in Fig. 6.7.

Figure 6.7 illustrates the execution time in seconds of FastSEA versus that of SEA. The results show that when the size of the time series increases significantly, the difference in execution time between the two methods becomes more remarkable. For instance, when the length of the time series is 1000 samples, the execution time is 3,060587 and 0,953463 seconds for SEA and FastSEA, respectively. However, for 100000 samples the execution time is 14542,21608 seconds (equivalently, 4 hours) for SEA and 44,978761 seconds for FastSEA. Hence, our proposed method FastSEA is up to many hundred times faster than SEA.

Figure 6.8 presents the percentage of time reduction recorded by FastSEA with respect to SEA. The results show that the reduction time increases quickly with the length of the time series from 68% for short time series to 99% for very long time series. These results confirm the efficiency of FastSEA over SEA and the ability of the proposed method to match very long

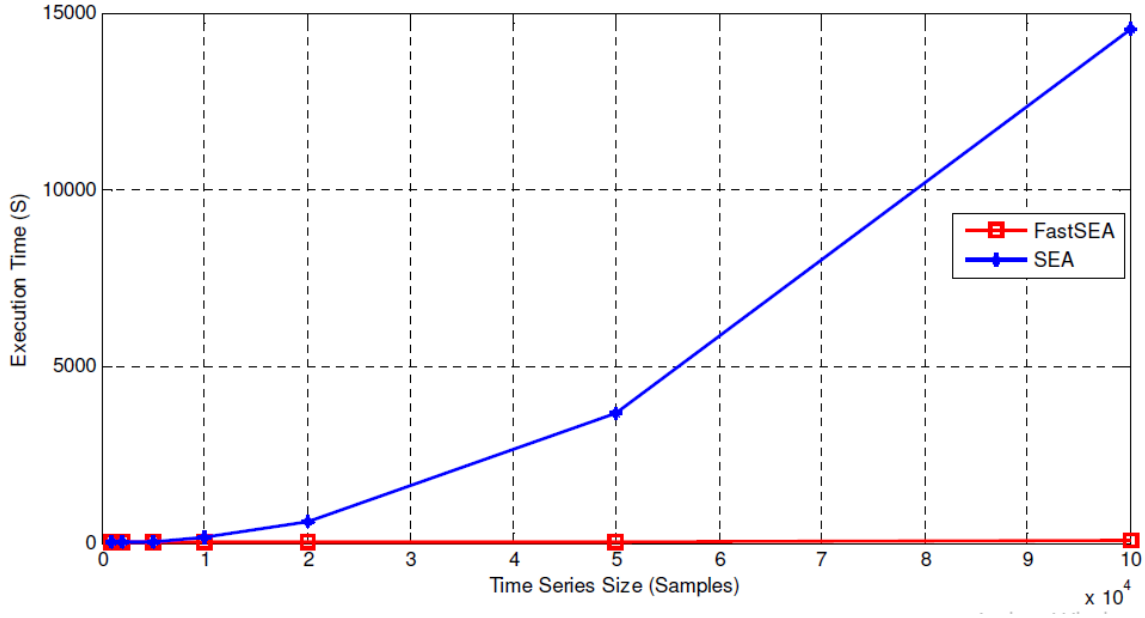

Figure 6.7: Execution time (seconds) as a function of time series length (samples) for FastSEA (squares) and SEA (stars) (Lahreche and Boucheham, 2017).

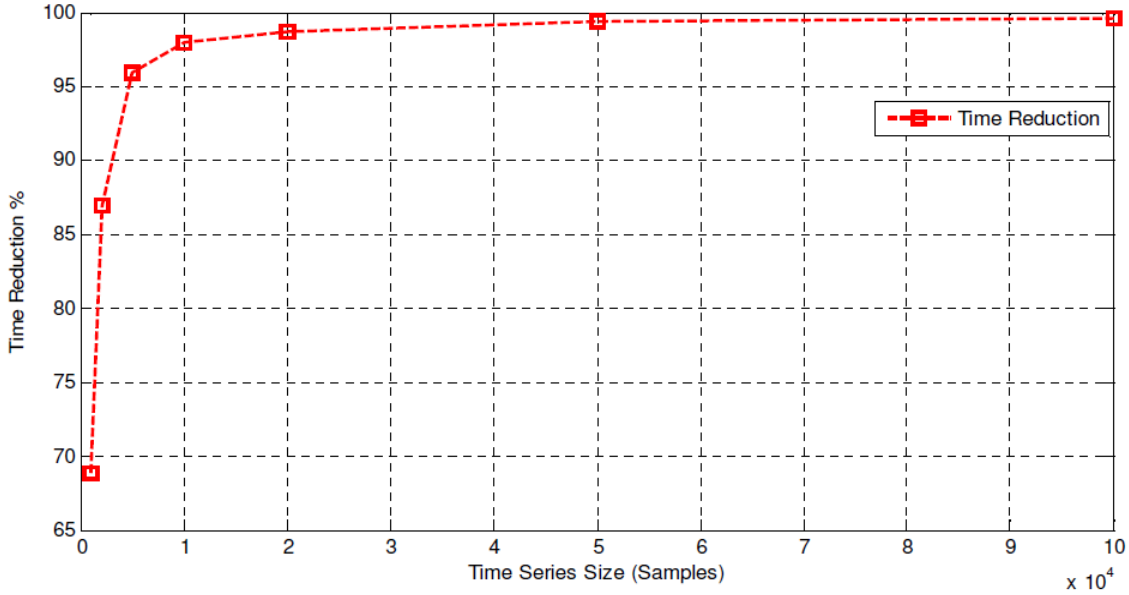

Figure 6.8: Time reduction percentage as a function of time series length (samples) recorded by FastSEA with respect to SEA (Lahreche and Boucheham, 2017).

time series in a very short time.

6.3.4.2 Application 2

In this application, the proposed FastSEA is compared to the SEA method in terms of effectiveness. For this purpose, we use two records 101 and 103 of 5000 samples each one taken from the MIT-BIH database. The results are reported in Fig. 6.9, Fig. 6.10, and Table 6.2.

Figure 6.9 and Fig. 6.10 show the perfect similar matching between FastSEA and SEA. The numerical results (PRD and Corr factors) reported in Table 6.2 demonstrate the results obtained in Fig. 6.9 and Fig. 6.10 where we have the same PRD and Corr values for both FastSEA and SEA methods.

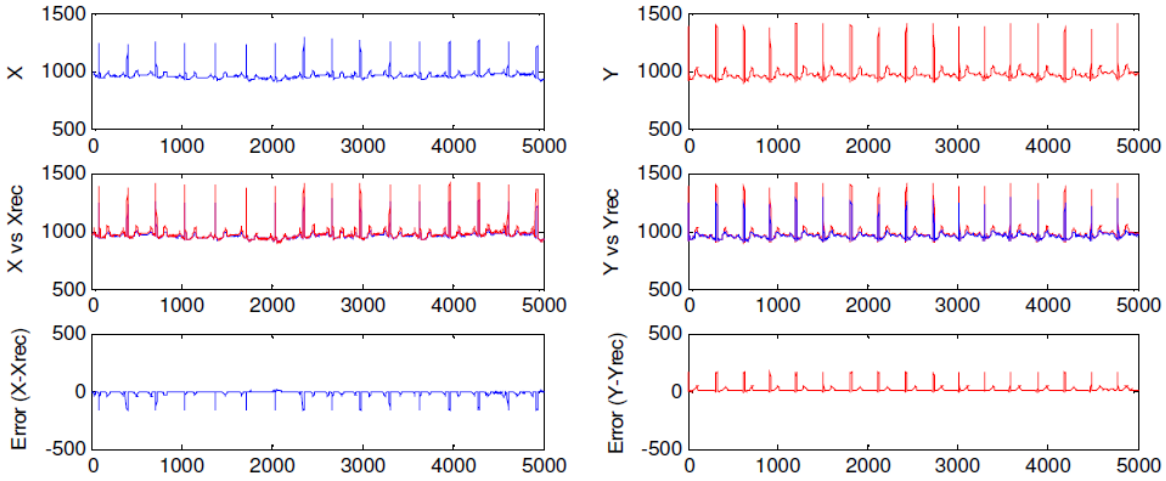

Figure 6.9: Two ECG signals (rec 101 and rec 103) of 5000 samples taken from MIT-BIH database matching by FastSEA method (Lahreche and Boucheham, 2017).

Table 6.2: PRD and Corr measures of FastDEA and SEA recording by matching two ECG signals (rec 101 and rec 103) of 5000 samples selected from MIT-BIH database (Lahreche and Boucheham, 2017).

Mean	**FastSEA**	**SEA**
PRD (%)	4.522	4.522
Corr (0..1)	0.989	0.989

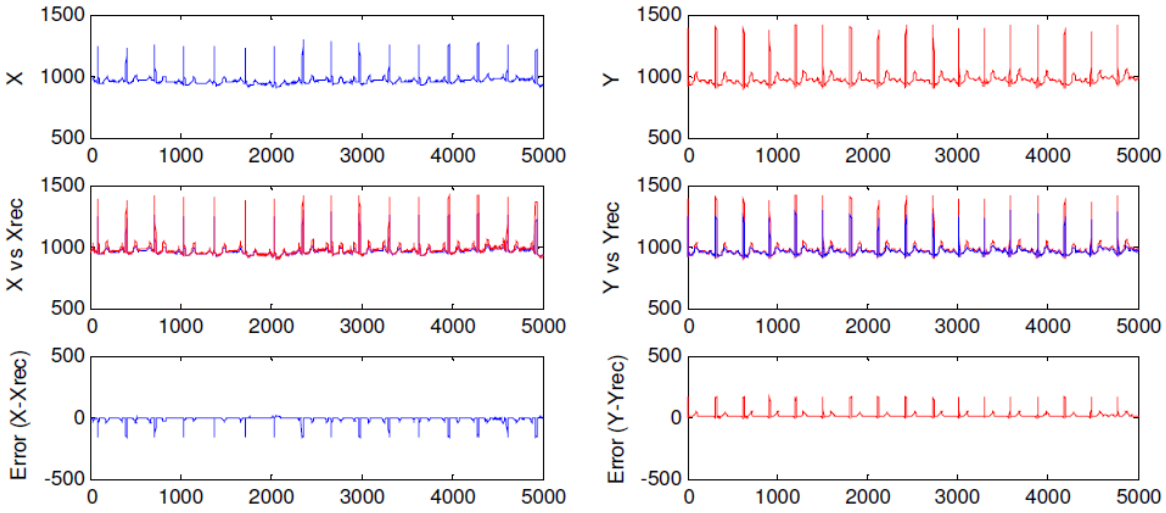

Figure 6.10: Two ECG signals (rec 101 and rec 103) of 5000 samples taken from MIT-BIH database matching by SEA method (Lahreche and Boucheham, 2017).

6.4 LMDS-SEA (Lahreche and Boucheham, 2018)

In this section, we first present the distance selection paradigm and then we provide our second contribution for general TSC.

6.4.1 Distance selection paradigm

Before going into the distance/similarity selection paradigm, we first present the concept of selection. The selection paradigm is the process of automatically choosing the most appropriate solution for a given task based on some specified performance criteria. By 'solution', we mean any aspect including feature (Benloucif and Boucheham, 2014), distance (Mosbah and Boucheham, 2017), algorithm (Hamreras and Boucheham, 2018; Hamreras et al., 2021), parameter, etc.

The mathematical foundation of the selection paradigm is the No-Free-Lunch Theorem (NFLT) (Ho and Pepyne, 2002). In general, the NFLT states that no algorithm is universally better than all the others. In other words, if Algorithm A is superior to Algorithm B on a specific problem, then there is necessarily another problem where Algorithm B is better than A. From an empirical perspective, different experiments have validated the NFLT. For instance, it has

been shown that none of the available time series distance measures outperforms all the others on all problems (Wang et al., 2013).

The selection paradigm can be used as a data reduction technique to achieve efficiency, and it is especially useful for building accurate algorithms (Jović et al., 2015). Feature selection is probably the most widely explored aspect in that area. It has been extensively investigated in applications as diverse as text mining (Forman et al., 2003), bioinformatics (Abusamra, 2013), and industry (Liu et al., 2014). On the other hand, distance selection has surprisingly received less attention. To the best of my knowledge, Kotsifakos et al. (2016), Mori et al. (2016), and Mosbah and Boucheham (2017) are the first researchers to introduce the distance selection paradigm into the literature. In (Mosbah and Boucheham, 2017), they propose for the first time a flexible distance selection model based on the SFS metaheuristic in the context of image retrieval. Mori et al. (2016) suggest a distance selection-based method for the time series clustering task, whereas Kotsifakos et al. (2016) propose a distance selection method in the context of TSC.

For time series distance selection, the problem can be stated as follows. Given a pool of time series distance measures $dist = \{dist_1, ..., dist_d\}$, a specific task P, and a performance function f. The goal is to find the most suitable distance $dist_i$ for which the performance f of task P is optimal. To the best of my knowledge, in addition to (Mori et al., 2016) for TS clustering, only one other work (Kotsifakos et al., 2016) has explicitly come up with this issue. Note that, (Kotsifakos et al., 2016) use the distance selection paradigm for solving the TSC problem.

6.4.2 LMDS-SEA method

This contribution concerns an enhanced version of the SEA method for general time series classification. Indeed, the SEA method is very effective for matching QPTS. However, for general time series classification, it has been found (see Fig. 6.1) that this method needs some kind of enhancement to meet state-of-the-art TSC algorithms. The reason relies on the fact that the SEA method uses global information through a kind of signatures extracted from the two time series to match, and no local information is considered in that method. As a result, the

SEA method shows low capabilities in dealing with the TSC problem.

The proposed Local Matching with Distance Selection Shape Exchange Algorithm (LMDS-SEA) method is based on two ideas: a) Distance Selection as an adaptation paradigm for classification method and b) Local Matching. The distance selection contribution consists in dynamically selecting the best distance to apply as a function of the data to classify from a pool of ED, SEA, and Local Matching SEA (LM-SEA). The LM-SEA proposition consists in matching equal and corresponding sub-sequences from the two time series to match using the SEA method locally. These ideas are explained in detail below.

6.4.2.1 Distance selection

First, we segment the two time series to match Q and C into a set of equal subsequences (S). The purpose is to extract local information from both series. One important question that arises at this step is: what would be the best length of the subsequences? In order to satisfy such demand, we perform a learning step on the training dataset to automatically derive the appropriate length. Simply stated, the appropriate length is that yields the best results in terms of classification accuracy in the training step. Consequently, the length of sub-sequences may differ from one dataset to another.

Technically, the length of the sub-sequences S is between 1 and n (length of the whole time series). In this regard, we distinguish three (03) cases:

- **The length l of the sub-sequences S is equal to 1 (l=1):** This means that there is a maximum segmentation of the time series. In this case, we compare the two time series to match using ED.

- **The length l of the sub-sequences S is equal to n (l=n):** This means that there is only one sub-sequence per time series, which is in fact the series itself. In other words, there is no segmentation in this case. Thus, we use the SEA method for comparison.

- **The length l of the sub-sequences S is between 1 et n (1<l<n):** In this case, a segmentation process is required before the comparison step. This involves the application of

the LM-SEA method.

As can be observed, there is an automatic selection of the best matching similarity measure from the pool: ED, SEA, and LM-SEA.

6.4.2.2 Local matching

This step concerns only the third case where the length of sub-sequences is between 1 and n. In this situation, we also propose another method designated by LM-SEA. LM-SEA is a new version of SEA that integrates local information within SEA. We segment time series and apply SEA locally between the corresponding sub-sequences from the two time series to match.

6.4.3 Experimental results and discussion

In this section, we evaluate the accuracy of our proposed LMDS-SEA within the TSC task. We then combine LMDS-SEA with the 1-NN classifier and compare its classification accuracy with those of 1-NN SEA and 1-NN DTW using the UCR archive (Chen et al., 2015a). Tables 6.3 and 6.4 report the classification error rate of 1-NN LMDS-SEA against 1-NN SEA and 1-NN DTW, respectively on 85 UCR time series datasets. To better visualize these results, we present a graphic illustration displayed in Fig. 6.11 for 1-NN LMDS-SEA vs. 1-NN SEA and Fig. 6.12 for 1-NN LMDS-SEA vs. 1-NN DTW.

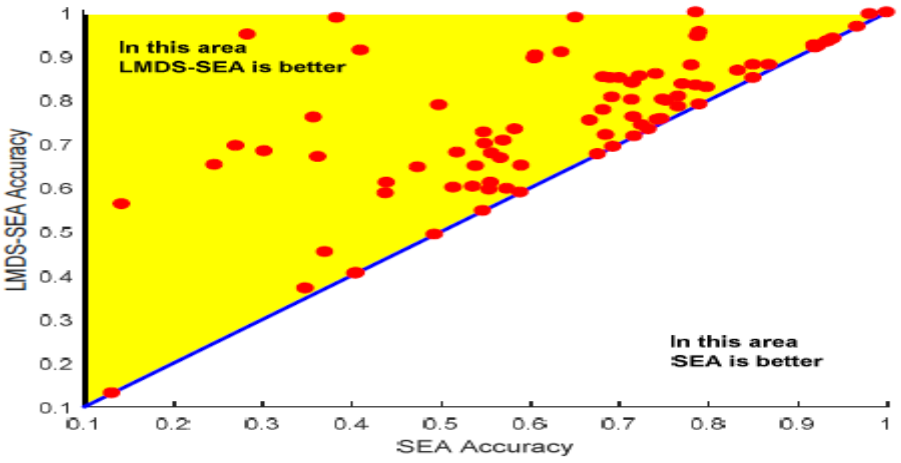

Figure 6.11: 1-NN classification accuracy comparison between LMDS-SEA and SEA on 85 UCR datasets (each point represents one dataset) (Lahreche and Boucheham, 2018).

Table 6.3: Classification error rates of 1-NN LMDS-SEA and 1-NN SEA on 85 UCR datasets (Lahreche and Boucheham, 2018).

Dataset	SEA	LMDS-SEA	S	Dataset	SEA	LMDS-SEA	S
50words	0,73	0,305	39	MedicalImages	0,43	0,293	4
Adiac	0,307	0,307	n	MiddlePhalanxOutlineAgeGroup	0,257	0,245	30
ArrowHead	0,308	0,194	10	MiddlePhalanxOutlineCorrect	0,333	0,247	1
Beef	0,433	0,3	100	MiddlePhalanxT	0,426	0,403	30
BeetleFly	0,3	0,15	25	MoteStrain	0,219	0,121	1
BirdChicken	0,15	0,15	n	NonInvasiveFatalECGThorax1	0,202	0,153	39
Car	0,417	0,267	1	NonInvasiveFatalECGThorax2	0,132	0,12	1
CBF	0,349	0,012	50	OliveOil	0,167	0,133	1
ChlorineConcentration	0,41	0,35	1	OSULeaf	0,562	0,413	45
CinCECGtorso	0,365	0,048	100	PhalangesOutlinesCorrect	0,284	0,239	1
Coffee	0,214	0	1	Phoneme	0,869	0,869	n
Computers	0,324	0,324	n	Plane	0	0	n
CricketX	0,482	0,303	20	ProximalPhalanxOutlineAgeGroup	0,234	0,215	5
CricketY	0,526	0,354	20	ProximalPhalanxOutlineCorrect	0,234	0,192	1
CricketZ	0,451	0,3	12	ProximalPhalanxTW	0,315	0,28	10
DiatomSizeReduction	0,033	0,033	n	RefrigerationDevices	0,507	0,507	n
DistalPhalanxOutlineAgeGroup	0,247	0,202	10	ScreenType	0,596	0,596	n
DistalPhalanxOutlineCorrect	0,253	0,243	5	ShapeletSim	0,561	0,389	10
DistalPhalanxTW	0,275	0,257	10	ShapesAll	0,318	0,223	100
Earthquakes	0,267	0,267	n	SmallKitchenAppliances	0,411	0,411	n
ECG200	0,15	0,12	1	SonyAIBORobotSurface	0,286	0,16	10
ECG5000	0,081	0,071	30	SonyAIBORobotSurfaceII	0,259	0,141	1
ECGFiveDays	0,066	0,066	n	StarLightCurves	0,31	0,127	200
ElectricDevices	0,461	0,351	10	Strawberry	0,06	0,06	n
FaceAll	0,502	0,212	19	SwedishLeaf	0,277	0,146	10
FaceFour	0,318	0,136	14	Symbols	0,394	0,081	41
FacesUCR	0,395	0,105	19	syntheticcontrol	0,617	0,013	6
FISH	0,286	0,2	15	ToeSegmentation1	0,21	0,21	n
FordA	0,443	0,322	10	ToeSegmentation2	0,285	0,115	45
FordB	0,446	0,376	40	Trace	0,08	0,08	n
GunPoint	0,08	0,08	n	TwoPatterns	0,59	0,087	10
Ham	0,486	0,4	1	TwoLeadECG	0,077	0,077	n
HandOutlines	0,251	0,197	1	uWaveGestureLibrary$_X$	0,643	0,24	45
Haptics	0,652	0,63	10	uWaveGestureLibrary$_Y$	0,698	0,317	45
Herring	0,453	0,453	n	uWaveGestureLibrary$_Z$	0,638	0,33	45
InlineSkate	0,596	0,596	n	UWaveGestureLibraryAll	0,717	0,042	50
InsectWingbeatSound	0,858	0,438	1	wafer	0,019	0,004	5
ItalyPowerDemand	0,21	0,045	1	Wine	0,444	0,389	1
LargeKitchenAppliances	0,283	0,283	n	WordsSynonyms	0,754	0,348	31
Lighting2	0,229	0,147	45	Worms	0,63	0,547	78
Lighting7	0,452	0,274	10	WormsTwoClass	0,464	0,398	30
MALLAT	0,212	0,055	100	yoga	0,214	0,165	100
Meat	0,067	0,067	n				
Average	**0,359**	**0,236**					
Win/Tie/Loss	**00/21/64**	**64/21/00**					

As shown in Table 6.3 and Fig. 6.11, the proposed LMDS-SEA method clearly outperforms the SEA method. Our method is best on 64 datasets and for the remaining 21 datasets, a tie is observed. On average, the difference between the two methods is huge in favor of our method, which is superior by more than 10%. Consequently, The LMDS-SEA has significantly improved the classification accuracy compared to the SEA method. Table 6.4 and Fig. 6.12 also demonstrate the superiority of our LMDS-SEA method with respect to DTW. LMDS-SEA has 50 wins while DTW has only 28 wins over 85 UCR datasets.

Table 6.4: Classification error rates of 1-NN LMDS-SEA and 1-NN DTW on 85 UCR datasets (Lahreche and Boucheham, 2018).

Dataset	DTW	LMDS-SEA	S	Dataset	DTW	LMDS-SEA	S
50words	0,31	0,305	39	MedicalImages	0,263	0,293	4
Adiac	0,396	0,307	n	MiddlePhalanxOutlineAgeGroup	0,25	0,245	30
ArrowHead	0,297	0,194	10	MiddlePhalanxOutlineCorrect	0,352	0,247	1
Beef	0,367	0,3	100	MiddlePhalanxTW	0,416	0,403	30
BeetleFly	0,3	0,15	25	MoteStrain	0,165	0,121	1
BirdChicken	0,25	0,15	n	NonInvasiveFatalECGThorax1	0,209	0,153	39
Car	0,267	0,267	1	NonInvasiveFatalECGThorax2	0,135	0,12	1
CBF	0,003	0,012	50	OliveOil	0,167	0,133	1
ChlorineConcentration	0,352	0,35	1	OSULeaf	0,409	0,413	45
CinCECGtorso	0,349	0,048	100	PhalangesOutlinesCorrect	0,272	0,239	1
Coffee	0	0	1	Phoneme	0,772	0,869	n
Computers	0,3	0,324	n	Plane	0	0	n
CricketX	0,246	0,303	20	ProximalPhalanxOutlineAgeGroup	0,195	0,215	5
CricketY	0,256	0,354	20	ProximalPhalanxOutlineCorrect	0,216	0,192	1
CricketZ	0,246	0,3	12	ProximalPhalanxTW	0,263	0,28	10
DiatomSizeReduction	0,033	0,033	n	RefrigerationDevices	0,536	0,507	n
DistalPhalanxOutlineAgeGroup	0,208	0,202	10	ScreenType	0,603	0,596	n
DistalPhalanxOutlineCorrect	0,232	0,243	5	ShapeletSim	0,35	0,389	10
DistalPhalanxTW	0,29	0,257	10	ShapesAll	0,232	0,223	100
Earthquakes	0,258	0,267	n	SmallKitchenAppliances	0,357	0,411	n
ECG200	0,23	0,12	1	SonyAIBORobotSurface	0,275	0,16	10
ECG5000	0,076	0,071	30	SonyAIBORobotSurfaceII	0,169	0,141	1
ECGFiveDays	0,232	0,066	n	StarLightCurves	0,093	0,127	200
ElectricDevice	0,399	0,351	10	Strawberry	0,06	0,06	n
FaceAll	0,192	0,212	19	SwedishLeaf	0,208	0,146	10
FaceFour	0,17	0,136	14	Symbols	0,05	0,081	41
FacesUCR	0,095	0,105	19	syntheticcontrol	0,007	0,013	6
FISH	0,177	0,2	15	ToeSegmentation1	0,228	0,21	n
FordA	0,438	0,322	10	ToeSegmentation2	0,162	0,115	45
FordB	0,406	0,376	40	Trace	0	0,08	n
GunPoint	0,093	0,08	n	TwoPatterns	0	0,087	10
Ham	0,533	0,4	1	TwoLeadECG	0,096	0,077	n
HandOutlines	0,202	0,197	1	uWaveGestureLibraryX	0,273	0,24	45
Haptics	0,623	0,63	10	uWaveGestureLibraryY	0,366	0,317	45
Herring	0,469	0,453	n	uWaveGestureLibraryZ	0,342	0,33	45
InlineSkate	0,616	0,596	n	UWaveGestureLibraryAll	0,108	0,042	50
InsectWingbeatSound	0,645	0,438	1	wafer	0,02	0,004	5
ItalyPowerDemand	0,05	0,045	1	Wine	0,426	0,389	1
LargeKitchenAppliances	0,205	0,283	n	WordsSynonyms	0,351	0,348	31
Lighting2	0,131	0,147	45	Worms	0,536	0,547	78
Lighting7	0,274	0,274	10	WormsTwoClass	0,337	0,398	30
MALLAT	0,066	0,055	100	yoga	0,164	0,165	100
Meat	0,067	0,067	n				
Average	**0,256**	**0,237**					
Win/Tie/Loss	**28/07/50**	**50/07/28**					

6.5 Conclusion

In this chapter, we have proposed two new methods that are improved versions of the SEA method. In the first proposed FastSEA method, we have concentrated on increasing the efficiency of SEA while maintaining its effectiveness for the QPTS matching problem. In the second proposal, we have expanded the effectiveness of SEA to general time series in the context of TSC. Extensive experiments have been conducted to evaluate the newly proposed methods. The obtained results indicated that both contributions are better than SEA. Specifically, FastSEA is several times faster, especially for very long QPTS, than SEA while offering

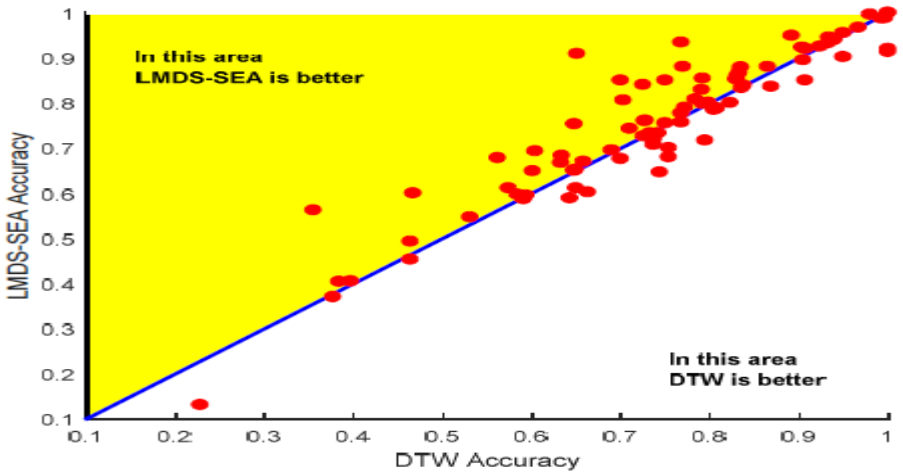

Figure 6.12: 1-NN classification accuracy comparison between LMDS-SEA and DTW on 85 UCR datasets (each point represents one dataset) (Lahreche and Boucheham, 2018).

the same alignment quality. On the other hand, LMDS-SEA is significantly more accurate than SEA for TSC.

General conclusions and perspectives

Conclusions

The objective of this thesis concerned the establishment of similarity in the context of time series. The emphasis was on evaluating the similarity between time series through similarity or distance measures. In particular, the study stressed the point that similarity measures are a key component of practically every time series mining task and other science and technology applications. In this thesis, we focused on time series classification as an increasingly important application. During this work, we investigated and explored different techniques and approaches in order to solve the problematic of interest.

In Chapter 4, we conducted a large experimental comparison reinforced by a thorough statistical analysis of DTW and its most popular variants for TSC. All methods were combined with the 1-NN classifier and then evaluated in terms of classification accuracy using 85 datasets from the UCR archive. Generally, results indicated that virtually all variants are statistically equivalent. In Chapter 5, we addressed the problem of long time series classification. In this regard, we developed a new fast and accurate similarity measure. The proposed LE-DTW consists of two steps. The first involves representing and reducing the dimensionality of raw time series using local extrema points. The second step consists in adapting the DTW method so as to measure the similarity between the generated representations. Extensive experiments were performed to evaluate the newly LE-DTW proposed on TSC using a wide variety of datasets from the UCR archive. The results showed that LE-DTW is more efficient while providing competitive accuracy against popular distance-based classifiers, especially on long time series data.

Chapter 6 was devoted to improving the performance of the SEA method. As a result, two enhanced versions have been proposed: FastSEA and LMDS-SEA. In FastSEA, we focused on accelerating the SEA method without losing its quality in the context of quasi-periodic time series alignment. While in LMDS-SEA, we aimed to extend the effectiveness of SEA to general TSC by introducing a novel distance selection paradigm. Experiments on dozens of public datasets demonstrated that both proposals are better than the SEA method.

To summarize, based on the work accomplished through this manuscript, it is possible to conclude that we achieved all the goals envisaged for this thesis. Finally, we hope that this thesis, with its theoretical and practical parts, could be a useful reference for both researchers and practitioners, especially on time series similarity measures.

Perspectives

In this section, we present some potential directions for future works.

- **Multivariate time series:** This thesis addressed only univariate time series data. However, in the past few years, multivariate time series data became increasingly omnipresent in many real-world applications. In future work, we plan to extend the applications of the proposed similarity measures to such a form of time series data.

- **Time series mining:** In this thesis, we mainly focused on a specific task of time series mining which is TSC. However, it would also be interesting to explore the impact of the newly proposed measures on the performance of other distance-based tasks such as clustering.

- **Distance selection paradigm:** While distance selection is relatively a new paradigm in the context of time series, further research of the issue would be of interest. In our future work, we intend to concentrate on two possible directions: 1) Integration of a large variety of distances and 2) Development of more sophisticated strategies to select the more appropriate distance such as meta-heuristics.

- **LE-DTW:** Despite the promising findings of the proposed LE-DTW, some drawbacks remain to be addressed in order to further improve it. For instance, future work will involve investigating more advanced techniques to well identify local extrema points, and hence minimize false alarms.

- **Comparative study:** In this context, the next stage of our research will be around two axes: 1) Include all DTW's variants available in the literature and 2) Evaluate the variants according to another criterion such as efficiency.

Appendix A

List of publications

1. Journal papers

- Lahreche, A., Boucheham, B. (2021). A fast and accurate similarity measure for long time series classification based on local extrema and dynamic time warping. Expert Systems with Applications, 168, 114374.

- Lahreche, A., Boucheham, B. A Comparison Study of Dynamic Time Warping's Variants for Time Series Classification. International Journal of Informatics and Applied Mathematics, 4(1), 56-71.

2. Conference papers

- Lahreche, A., Boucheham, B. (2017, December). FastSEA: A very fast and very effective matching technique for very complex time series. In 2017 International Conference on Mathematics and Information Technology (ICMIT) (pp. 286-293). IEEE.

- Lahreche, A., Boucheham, B. (2018, October). LMDS-SEA: Upgrading the Shape Exchange Algorithm (SEA) to handle general time series classification by local matching and distance selection. In 2018 3rd International Conference on Pattern Analysis and

Intelligent Systems (PAIS) (pp. 1-6). IEEE.

Bibliography

Abanda, A., Mori, U., and Lozano, J. A. (2019). A review on distance based time series classification. *Data Mining and Knowledge Discovery*, 33(2):378–412.

Abusamra, H. (2013). A comparative study of feature selection and classification methods for gene expression data of glioma. *Procedia Computer Science*, 23:5–14.

Aggarwal, C. C. et al. (2015). *Data mining: the textbook*, volume 1. Springer.

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

Agrawal, R., Faloutsos, C., and Swami, A. (1993). Efficient similarity search in sequence databases. In *International conference on foundations of data organization and algorithms*, pages 69–84. Springer.

Al-Naymat, G., Chawla, S., and Taheri, J. (2012). SparseDTW: A novel approach to speed up dynamic time warping. *arXiv preprint arXiv:1201.2969*.

Antonucci, A., De Rosa, R., Giusti, A., and Cuzzolin, F. (2015). Robust classification of multivariate time series by imprecise hidden markov models. *International Journal of Approximate Reasoning*, 56:249–263.

Bagnall, A., Dau, H. A., Lines, J., Flynn, M., Large, J., Bostrom, A., Southam, P., and Keogh, E. (2018). The UEA multivariate time series classification archive, 2018. *arXiv preprint arXiv:1811.00075*. http://www.timeseriesclassification.com/dataset.php.

Bagnall, A., Davis, L., Hills, J., and Lines, J. (2012). Transformation based ensembles for time series classification. In *Proceedings of the 2012 SIAM international conference on data mining*, pages 307–318. SIAM.

Bagnall, A., Flynn, M., Large, J., Lines, J., and Middlehurst, M. (2020). On the usage and performance of the hierarchical vote collective of transformation-based ensembles version 1.0 (HIVE-COTE v1. 0). In *International Workshop on Advanced Analytics and Learning on Temporal Data*, pages 3–18. Springer.

Bagnall, A. and Janacek, G. (2014). A run length transformation for discriminating between auto regressive time series. *Journal of classification*, 31(2):154–178.

Bagnall, A. and Lines, J. (2014). An experimental evaluation of nearest neighbour time series classification. *arXiv preprint arXiv:1406.4757*.

Bagnall, A., Lines, J., Bostrom, A., Large, J., and Keogh, E. (2017). The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data mining and knowledge discovery*, 31(3):606–660.

Bagnall, A., Lines, J., Hills, J., and Bostrom, A. (2015). Time-series classification with cote: the collective of transformation-based ensembles. *IEEE Transactions on Knowledge and Data Engineering*, 27(9):2522–2535.

Batista, G. E., Keogh, E. J., Tataw, O. M., and de Souza, V. (2014). CID: an efficient complexity-invariant distance for time series. *Data Mining and Knowledge Discovery*, 28(3):634–669.

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

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

Benloucif, S. and Boucheham, B. (2014). Impact of feature selection on the performance of

content-based image retrieval (CBIR). In *2014 4th International Symposium ISKO-Maghreb: Concepts and Tools for knowledge Management (ISKO-Maghreb)*, pages 1–7. IEEE.

Berndt, D. J. and Clifford, J. (1994). Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, USA:.

Betancourt, J. P., Fatichah, C., Tangel, M. L., Yan, F., Sanchez, J. A. G., Dong, F.-Y., and Hirota, K. (2013). Similarity-based fuzzy classification of ecg and capnogram signals. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 17(2):302–310.

Bostrom, A. and Bagnall, A. (2015). Binary shapelet transform for multiclass time series classification. In *International conference on big data analytics and knowledge discovery*, pages 257–269. Springer.

Boucheham, B. (2008). Matching of quasi-periodic time series patterns by exchange of block-sorting signatures. *Pattern Recognition Letters*, 29(4):501–514.

Boucheham, B. (2010). Reduced data similarity-based matching for time series patterns alignment. *Pattern Recognition Letters*, 31(7):629–638.

Boucheham, B. (2013). Efficient matching of very complex time series. *International Journal of Machine Learning and Cybernetics*, 4(5):537–550.

Boulnemour, I. and Boucheham, B. (2015). I-sea: improved shape exchange algorithm for quasi-periodic time series alignment. In *International Conference on Computer Vision and Image Analysis Applications*, pages 1–6. IEEE.

Boulnemour, I. and Boucheham, B. (2018). QP-DTW: upgrading dynamic time warping to handle quasi periodic time series alignment. *Journal of Information Processing Systems*, 14(4):851–876.

Cassisi, C., Montalto, P., Aliotta, M., Cannata, A., and Pulvirenti, A. (2012). Similarity measures and dimensionality reduction techniques for time series data mining. *Advances in data mining knowledge discovery and applications'(InTech, Rijeka, Croatia, 2012,*, pages 71–96.

Cha, S.-H. (2007). Comprehensive survey on distance/similarity measures between probability density functions. *City*, 1(2):1.

Chan, F.-P., Fu, A.-C., and Yu, C. (2003). Haar wavelets for efficient similarity search of time-series: with and without time warping. *IEEE Transactions on knowledge and data engineering*, 15(3):686–705.

Chan, K.-P. and Fu, A. W.-C. (1999). Efficient time series matching by wavelets. In *Proceedings 15th International Conference on Data Engineering (Cat. No. 99CB36337)*, pages 126–133. IEEE.

Chang, K.-W., Deka, B., Hwu, W.-M. W., and Roth, D. (2012). Efficient pattern-based time series classification on gpu. In *2012 IEEE 12th International Conference on Data Mining*, pages 131–140. IEEE.

Chen, L. and Ng, R. (2004). On the marriage of Lp-norms and edit distance. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 792–803.

Chen, L., Özsu, M. T., and Oria, V. (2005). Robust and fast similarity search for moving object trajectories. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 491–502.

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

Chen, Z., Zuo, W., Hu, Q., and Lin, L. (2015b). Kernel sparse representation for time series classification. *Information Sciences*, 292:15–26.

Cooley, J. W. and Tukey, J. W. (1965). An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301.

Corduas, M. and Piccolo, D. (2008). Time series clustering and classification by the autoregressive metric. *Computational statistics & data analysis*, 52(4):1860–1872.

Cui, Z., Chen, W., and Chen, Y. (2016). Multi-scale convolutional neural networks for time series classification. *arXiv preprint arXiv:1603.06995*.

Cunningham, P. and Delany, S. J. (2021). k-Nearest Neighbour classifiers-A tutorial. *ACM Computing Surveys (CSUR)*, 54(6):1–25.

Dau, H. A., Bagnall, A., Kamgar, K., Yeh, C.-C. M., Zhu, Y., Gharghabi, S., Ratanamahatana, C. A., and Keogh, E. (2019). The UCR time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305.

Dau, H. A., Keogh, E., Kamgar, K., Yeh, C.-C. M., Zhu, Y., Gharghabi, S., Ratanamahatana, C. A., Yanping, Hu, B., Begum, N., Bagnall, A., Mueen, A., Batista, G., and Hexagon-ML (2018). The UCR time series classification archive. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/.

Dempster, A., Petitjean, F., and Webb, G. I. (2020). Rocket: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery*, 34(5):1454–1495.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30.

Deng, H., Runger, G., Tuv, E., and Vladimir, M. (2013). A time series forest for classification and feature extraction. *Information Sciences*, 239:142–153.

Deza, M. M. and Deza, E. (2009). Encyclopedia of distances. In *Encyclopedia of distances*, pages 1–583. Springer.

Echihabi, K., Zoumpatianos, K., Palpanas, T., and Benbrahim, H. (2020). Return of the lernaean hydra: Experimental evaluation of data series approximate similarity search. *arXiv preprint arXiv:2006.11459*.

Esling, P. and Agon, C. (2012). Time-series data mining. *ACM Computing Surveys (CSUR)*, 45(1):1–34.

Faloutsos, C., Ranganathan, M., and Manolopoulos, Y. (1994). Fast subsequence matching in time-series databases. *ACM Sigmod Record*, 23(2):419–429.

Fawaz, H. I. (2020). *Deep learning for time series classification*. PhD thesis, Université de Haute-Alsace, France. https://arxiv.org/abs/2010.00567.

Fawaz, H. I., Forestier, G., Weber, J., Idoumghar, L., and Muller, P.-A. (2019). Deep learning for time series classification: a review. *Data mining and knowledge discovery*, 33(4):917–963.

Fink, E. and Pratt, K. B. (2004). Indexing of compressed time series. In *Data mining in time series databases*, pages 43–65. World Scientific.

Forman, G. et al. (2003). An extensive empirical study of feature selection metrics for text classification. *J. Mach. Learn. Res.*, 3(Mar):1289–1305.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association*, 32(200):675–701.

Friedman, M. (1940). A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11(1):86–92.

Fu, T. C. (2011). A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181.

Fu, T.-c., Chung, F.-l., Luk, R., and Ng, C.-m. (2008). Representing financial time series based on data point importance. *Engineering Applications of Artificial Intelligence*, 21(2):277–300.

Fuad, M. M. M. (2011). *Similarity Search in High-dimensional Spaces with Applications to Time Series Data Mining and Information Retrieval*. PhD thesis, Université de Bretagne Sud. https://tel.archives-ouvertes.fr/tel-00619953.

Fulcher, B. D. and Jones, N. S. (2014). Highly comparative feature-based time-series classification. *IEEE Transactions on Knowledge and Data Engineering*, 26(12):3026–3037.

Garcia, S. and Herrera, F. (2008). An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of machine learning research*, 9(12).

Geler, Z. (2015). *Role of similarity measures in time series analysis*. PhD thesis, University of Novi Sad (Serbia). https://nardus.mpn.gov.rs/handle/123456789/1703.

Geurts, P. (2001). Pattern extraction for time series classification. In *European conference on principles of data mining and knowledge discovery*, pages 115–127. Springer.

Goldberger, A. L., Amaral, L. A., Glass, L., Hausdorff, J. M., Ivanov, P. C., Mark, R. G., Mietus, J. E., Moody, G. B., Peng, C.-K., and Stanley, H. E. (2000). Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *circulation*, 101(23):e215–e220.

Gordon, D., Hendler, D., and Rokach, L. (2012). Fast randomized model generation for shapelet-based time series classification. *arXiv preprint arXiv:1209.5038*.

Górecki, T. and Łuczak, M. (2013). Using derivatives in time series classification. *Data Mining and Knowledge Discovery*, 26(2):310–331.

Grabocka, J., Bedalli, E., and Schmidt-Thieme, L. (2012). Efficient classification of long time-series. In *International Conference on ICT Innovations*, pages 47–57. Springer.

Grabocka, J., Wistuba, M., and Schmidt-Thieme, L. (2016). Fast classification of univariate and multivariate time series through shapelet discovery. *Knowledge and information systems*, 49(2):429–454.

Hamreras, S. and Boucheham, B. (2018). Adaptive content based image retrieval based on rice algorithm selection model. In *2018 International Symposium on Programming and Systems (ISPS)*, pages 1–6. IEEE.

Hamreras, S., Boucheham, B., Molina-Cabello, M. A., Benítez-Rochel, R., and López-Rubio, E. (2021). Dynamic selection of classifiers for content based image retrieval. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Han, J., Pei, J., and Kamber, M. (2011). *Data mining: concepts and techniques*. Elsevier.

He, Q., Zhuang, F., Shang, T., Shi, Z., et al. (2012). Fast time series classification based on infrequent shapelets. In *2012 11th international conference on machine learning and applications*, volume 1, pages 215–219. IEEE.

Hills, J., Lines, J., Baranauskas, E., Mapp, J., and Bagnall, A. (2014). Classification of time series by shapelet transformation. *Data mining and knowledge discovery*, 28(4):851–881.

Hills, J. F. (2014). *Mining time-series data using discriminative subsequences*. PhD thesis, University of East Anglia.

Hirschberg, D. S. (1975). A linear space algorithm for computing maximal common subsequences. *Communications of the ACM*, 18(6):341–343.

Ho, Y.-C. and Pepyne, D. L. (2002). Simple explanation of the no-free-lunch theorem and its implications. *Journal of optimization theory and applications*, 115(3):549–570.

Itakura, F. (1975). Minimum prediction residual principle applied to speech recognition. *IEEE Transactions on acoustics, speech, and signal processing*, 23(1):67–72.

Jalalian, A. and Chalup, S. K. (2013). GDTW-P-SVMs: Variable-length time series analysis using support vector machines. *Neurocomputing*, 99:270–282.

Jeong, Y.-S., Jeong, M. K., and Omitaomu, O. A. (2011). Weighted dynamic time warping for time series classification. *Pattern recognition*, 44(9):2231–2240.

Jiang, W. (2020). Time series classification: nearest neighbor versus deep learning models. *SN Applied Sciences*, 2(4):1–17.

Jović, A., Brkić, K., and Bogunović, N. (2015). A review of feature selection methods with applications. In *2015 38th international convention on information and communication technology, electronics and microelectronics (MIPRO)*, pages 1200–1205. Ieee.

Kate, R. J. (2016). Using dynamic time warping distances as features for improved time series classification. *Data Mining and Knowledge Discovery*, 30(2):283–312.

Keogh, E. (2002). The UCR time series data mining archive. https://www.cs.ucr.edu/~eamonn/time_series_data/.

Keogh, E., Chakrabarti, K., Pazzani, M., and Mehrotra, S. (2001a). Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and information Systems*, 3(3):263–286.

Keogh, E., Chakrabarti, K., Pazzani, M., and Mehrotra, S. (2001b). Locally adaptive dimensionality reduction for indexing large time series databases. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, pages 151–162.

Keogh, E., Chu, S., Hart, D., and Pazzani, M. (2001c). An online algorithm for segmenting time series. In *Proceedings 2001 IEEE international conference on data mining*, pages 289–296. IEEE.

Keogh, E., Lonardi, S., and Ratanamahatana, C. A. (2004). Towards parameter-free data mining. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 206–215.

Keogh, E. and Ratanamahatana, C. A. (2005). Exact indexing of dynamic time warping. *Knowledge and information systems*, 7(3):358–386.

Keogh, E. J. and Pazzani, M. J. (2001). Derivative dynamic time warping. In *Proceedings of the 2001 SIAM international conference on data mining*, pages 1–11. SIAM.

Korn, F., Jagadish, H. V., and Faloutsos, C. (1997). Efficiently supporting ad hoc queries in large datasets of time sequences. *Acm Sigmod Record*, 26(2):289–300.

Kotsifakos, A., Athitsos, V., and Papapetrou, P. (2016). Query-sensitive distance measure selection for time series nearest neighbor classification. *Intelligent Data Analysis*, 20(1):5–27.

Lahreche, A. and Boucheham, B. (2017). FastSEA: A very fast and very effective matching technique for very complex time series. In *2017 International Conference on Mathematics and Information Technology (ICMIT)*, pages 286–293. IEEE.

Lahreche, A. and Boucheham, B. (2018). LMDS-SEA: Upgrading the shape exchange algorithm (SEA) to handle general time series classification by local matching and distance selection. In *2018 3rd International Conference on Pattern Analysis and Intelligent Systems (PAIS)*, pages 1–6. IEEE.

Lahreche, A. and Boucheham, B. (2021a). A comparison study of dynamic time warping's variants for time series classification. *International Journal of Informatics and Applied Mathematics*, 4(1):56–71.

Lahreche, A. and Boucheham, B. (2021b). A fast and accurate similarity measure for long time series classification based on local extrema and dynamic time warping. *Expert Systems with Applications*, 168:114374.

Lang, W., Morse, M., and Patel, J. M. (2009). Dictionary-based compression for long time-series similarity. *IEEE transactions on knowledge and data engineering*, 22(11):1609–1622.

LeCun, Y., Bengio, Y., et al. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

Lei, Q., Yi, J., Vaculin, R., Wu, L., and Dhillon, I. S. (2017). Similarity preserving representation learning for time series clustering. *arXiv preprint arXiv:1702.03584*.

Li, D. (2018). *Transforming time series for efficient and accurate classification*. PhD thesis, University of Luxembourg, France.

Li, D., Bissyandé, T. F., Klein, J., and Traon, Y. L. (2016). Time series classification with discrete wavelet transformed data. *International Journal of Software Engineering and Knowledge Engineering*, 26(09n10):1361–1377.

Liao, T. W. (2005). Clustering of time series data—a survey. *Pattern recognition*, 38(11):1857–1874.

Lin, J., Keogh, E., Lonardi, S., and Chiu, B. (2003). A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 2–11.

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

Lin, J., Williamson, S., Borne, K., and DeBarr, D. (2012b). Pattern recognition in time series. *Advances in Machine Learning and Data Mining for Astronomy*, 1(617-645):3.

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

Lines, J., Taylor, S., and Bagnall, A. (2016). HIVE-COTE: The hierarchical vote collective of transformation-based ensembles for time series classification. In *2016 IEEE 16th international conference on data mining (ICDM)*, pages 1041–1046. IEEE.

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

Liu, C., Jiang, D., and Yang, W. (2014). Global geometric similarity scheme for feature selection in fault diagnosis. *Expert Systems with Applications*, 41(8):3585–3595.

Lods, A., Malinowski, S., Tavenard, R., and Amsaleg, L. (2017). Learning DTW-preserving shapelets. In *International Symposium on Intelligent Data Analysis*, pages 198–209. Springer.

Lonardi, J. and Patel, P. (2002). Finding motifs in time series. In *Proc. of the 2nd Workshop on Temporal Data Mining*, pages 53–68.

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

Ma, D.-L. and Zhang, Y.-L. (2017). Time series piecewise linear representation based on trend feature points. In *International Conference on Green Intelligent Transportation System and Safety*, pages 19–28. Springer.

Marteau, P.-F. (2008). Time warp edit distance with stiffness adjustment for time series matching. *IEEE transactions on pattern analysis and machine intelligence*, 31(2):306–318.

Marteau, P.-F. and Gibet, S. (2014). On recursive edit distance kernels with application to time series classification. *IEEE transactions on neural networks and learning systems*, 26(6):1121–1133.

Middlehurst, M., Large, J., and Bagnall, A. (2020a). The canonical interval forest (CIF) classifier for time series classification. In *2020 IEEE international conference on big data (big data)*, pages 188–195. IEEE.

Middlehurst, M., Large, J., Cawley, G., and Bagnall, A. (2020b). The temporal dictionary ensemble (TDE) classifier for time series classification. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 660–676. Springer.

Middlehurst, M., Large, J., Flynn, M., Lines, J., Bostrom, A., and Bagnall, A. (2021). Hive-cote 2.0: a new meta ensemble for time series classification. *Machine Learning*, 110(11):3211–3243.

Mizuhara, Y., Hayashi, A., and Suematsu, N. (2006). Embedding of time series data by using dynamic time warping distances. *Systems and computers in Japan*, 37(3):1–9.

Montero, P. and Vilar, J. A. (2015). TSclust: An r package for time series clustering. *Journal of Statistical Software*, 62:1–43.

Moody, G. B. and Mark, R. G. (2001). The impact of the MIT-BIH arrhythmia database. *IEEE Engineering in Medicine and Biology Magazine*, 20(3):45–50.

Mori, U., Mendiburu, A., and Lozano, J. A. (2016). Similarity measure selection for clustering time series databases. *IEEE Transactions on Knowledge and Data Engineering*, 28(1):181–195.

Mosbah, M. and Boucheham, B. (2017). Distance selection based on relevance feedback in the context of cbir using the sfs meta-heuristic with one round. *Egyptian Informatics Journal*, 18(1):1–9.

Mueen, A. and Keogh, E. (2016). Extracting optimal performance from dynamic time warping. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2129–2130.

Mueen, A., Keogh, E., and Bigdely-Shamlo, N. (2009). Finding time series motifs in disk-resident data. In *2009 Ninth IEEE International Conference on Data Mining*, pages 367–376. IEEE.

Mueen, A., Keogh, E., and Young, N. (2011). Logical-shapelets: an expressive primitive for time series classification. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1154–1162.

Nanopoulos, A., Alcock, R., and Manolopoulos, Y. (2001). Feature-based classification of time-series data. *International Journal of Computer Research*, 10(3):49–61.

Neamtu, R., Ahsan, R., Rundensteiner, E. A., Sarkozy, G., Keogh, E., Dau, H. A., Nguyen, C., and Lovering, C. (2018). Generalized dynamic time warping: Unleashing the warping power hidden in point-wise distances. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 521–532. IEEE.

Nemenyi, P. B. (1963). *Distribution-free multiple comparisons.* Princeton University.

Norvig, P. R. and Intelligence, S. A. (2002). *A modern approach*. Prentice Hall Upper Saddle River, NJ, USA:.

Paparrizos, J. and Gravano, L. (2015). k-shape: Efficient and accurate clustering of time series. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, pages 1855–1870.

Paparrizos, J. and Gravano, L. (2017). Fast and accurate time-series clustering. *ACM Transactions on Database Systems (TODS)*, 42(2):1–49.

Paparrizos, J., Liu, C., Elmore, A. J., and Franklin, M. J. (2020). Debunking four long-standing misconceptions of time-series distance measures. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 1887–1905.

Pavlidis, T. and Horowitz, S. L. (1974). Segmentation of plane curves. *IEEE transactions on Computers*, 100(8):860–870.

Perng, C.-S., Wang, H., Zhang, S. R., and Parker, D. S. (2000). Landmarks: a new model for similarity-based pattern querying in time series databases. In *Proceedings of 16th international conference on data engineering (cat. no. 00cb37073)*, pages 33–42. IEEE.

Popivanov, I. and Miller, R. J. (2002). Similarity search over time-series data using wavelets. In *Proceedings 18th international conference on data engineering*, pages 212–221. IEEE.

Pratt, K. B. and Fink, E. (2002). Search for patterns in compressed time series. *International Journal of Image and Graphics*, 2(01):89–106.

Rakthanmanon, T. and Keogh, E. (2013). Fast shapelets: A scalable algorithm for discovering time series shapelets. In *proceedings of the 2013 SIAM International Conference on Data Mining*, pages 668–676. SIAM.

Ratanamahatana, C. A. and Keogh, E. (2004). Everything you know about dynamic time warping is wrong. In *Third workshop on mining temporal and sequential data*, volume 32. Citeseer.

Ratanamahatana, C. A. and Keogh, E. (2005). Three myths about dynamic time warping data mining. In *Proceedings of the 2005 SIAM international conference on data mining*, pages 506–510. SIAM.

Ratanamahatana, C. A., Lin, J., Gunopulos, D., Keogh, E., Vlachos, M., and Das, G. (2005). Mining time series data. In *Data mining and knowledge discovery handbook*, pages 1069–1103. Springer.

Renard, X. (2017). *Time series representation for classification: a motif-based approach*. PhD thesis, Université Pierre et Marie Curie-Paris VI.

Sakoe, H. and Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49.

Salvador, S. and Chan, P. (2007). Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580.

Schäfer, P. (2015a). The BOSS is concerned with time series classification in the presence of noise. *Data Mining and Knowledge Discovery*, 29(6):1505–1530.

Schäfer, P. (2015b). *Scalable time series similarity search for data analytics*. PhD thesis, Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät. https://edoc.hu-berlin.de/handle/18452/17990.

Schäfer, P. (2016). Scalable time series classification. *Data Mining and Knowledge Discovery*, 30(5):1273–1298.

Schäfer, P. and Högqvist, M. (2012). SFA: a symbolic fourier approximation and index for similarity search in high dimensional datasets. In *Proceedings of the 15th international conference on extending database technology*, pages 516–527.

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

Serrà, J. and Arcos, J. L. (2014). An empirical evaluation of similarity measures for time series classification. *Knowledge-Based Systems*, 67:305–314.

Serrà, J., Pascual, S., and Karatzoglou, A. (2018). Towards a universal neural network encoder for time series. In *CCIA*, pages 120–129.

Sharabiani, A. (2018). *Novel Approaches Towards Fast and Accurate Time series Classification*. PhD thesis, University of Illinois at Chicago.

Sharabiani, A., Darabi, H., Rezaei, A., Harford, S., Johnson, H., and Karim, F. (2017). Efficient classification of long time series by 3-D dynamic time warping. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(10):2688–2703.

Shifaz, A., Pelletier, C., Petitjean, F., and Webb, G. I. (2020). TS-CHIEF: a scalable and accurate forest algorithm for time series classification. *Data Mining and Knowledge Discovery*, 34(3):742–775.

Silva, D. F. and Batista, G. E. (2016). Speeding up all-pairwise dynamic time warping matrix calculation. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pages 837–845. SIAM.

Silva, D. F., Giusti, R., Keogh, E., and Batista, G. E. (2018). Speeding up similarity search under dynamic time warping by pruning unpromising alignments. *Data Mining and Knowledge Discovery*, 32(4):988–1016.

Smyth, P. (1996). Clustering sequences with hidden markov models. *Advances in neural information processing systems*, 9.

Soleimani, G. and Abessi, M. (2020). DLCSS: A new similarity measure for time series data mining. *Engineering Applications of Artificial Intelligence*, 92:103664.

Spiegel, S. (2015). *Time series distance measures: segmentation, classification, and clustering of temporal data*. PhD thesis, Berlin, Technische Universität Berlin, Diss. https://depositonce.tu-berlin.de/handle/11303/4916.

Stefan, A., Athitsos, V., and Das, G. (2012). The move-split-merge metric for time series. *IEEE transactions on Knowledge and Data Engineering*, 25(6):1425–1438.

Tan, C. W. (2017). Dataset: Time series indexing (TSI). Monash University. Dataset. https://bridges.monash.edu/articles/dataset/Dataset/4557349.

Tan, C. W. (2019). *Time Series Classification at Scale*. PhD thesis, Monash University. https://bridges.monash.edu/articles/thesis/Time_Series_Classification_at_Scale/7924397.

Tan, C. W., Petitjean, F., and Webb, G. I. (2020). Fastee: Fast ensembles of elastic distances for time series classification. *Data Mining and Knowledge Discovery*, 34(1):231–272.

Tan, C. W., Webb, G. I., and Petitjean, F. (2017). Indexing and classifying gigabytes of time series under time warping. In *Proceedings of the 2017 SIAM international conference on data mining*, pages 282–290. SIAM.

Tanisaro, P. and Heidemann, G. (2016). Time series classification using time warping invariant echo state networks. In *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 831–836. IEEE.

Vemulapalli, P. K., Monga, V., and Brennan, S. N. (2012). Robust extrema features for time-series data analysis. *IEEE transactions on pattern analysis and machine intelligence*, 35(6):1464–1479.

Vlachos, M., Hadjieleftheriou, M., Gunopulos, D., and Keogh, E. (2003). Indexing multidimensional time-series with support for multiple distance measures. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 216–225.

Wang, X., Mueen, A., Ding, H., Trajcevski, G., Scheuermann, P., and Keogh, E. (2013). Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, 26(2):275–309.

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

Wilcoxon, F. (1992). Individual comparisons by ranking methods. In *Breakthroughs in statistics*, pages 196–202. Springer.

Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Yu, P. S., et al. (2008). Top 10 algorithms in data mining. *Knowledge and information systems*, 14(1):1–37.

Wu, Y.-L., Agrawal, D., and El Abbadi, A. (2000). A comparison of DFT and DWT based similarity search in time-series databases. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 488–495.

Xi, X., Keogh, E., Shelton, C., Wei, L., and Ratanamahatana, C. A. (2006). Fast time series classification using numerosity reduction. In *Proceedings of the 23rd international conference on Machine learning*, pages 1033–1040.

Xing, Z., Pei, J., and Keogh, E. (2010). A brief survey on sequence classification. *ACM Sigkdd Explorations Newsletter*, 12(1):40–48.

Yan, C., Fang, J., Wu, L., and Ma, S. (2013). An approach of time series piecewise linear representation based on local maximum minimum and extremum. *JOURNAL OF INFORMATION &COMPUTATIONAL SCIENCE*, 10(9):2747–2756.

Yankov, D., Keogh, E., Medina, J., Chiu, B., and Zordan, V. (2007). Detecting time series motifs under uniform scaling. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 844–853.

Ye, L. and Keogh, E. (2011). Time series shapelets: a novel technique that allows accurate, interpretable and fast classification. *Data mining and knowledge discovery*, 22(1):149–182.

Yuan, J., Douzal-Chouakria, A., Varasteh Yazdi, S., and Wang, Z. (2019a). A large margin time series nearest neighbour classification under locally weighted time warps. *Knowledge and Information Systems*, 59(1):117–135.

Yuan, J., Lin, Q., Zhang, W., and Wang, Z. (2019b). Locally slope-based dynamic time warping for time series classification. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 1713–1722.

Zakaria, J., Mueen, A., and Keogh, E. (2012). Clustering time series using unsupervised-shapelets. In *2012 IEEE 12th International Conference on Data Mining*, pages 785–794. IEEE.

Zhang, C. and Ma, Y. (2012). *Ensemble machine learning: methods and applications*. Springer.

Zhang, D., Zuo, W., Zhang, D., and Zhang, H. (2010). Time series classification using support vector machine with gaussian elastic metric kernel. In *2010 20th International Conference on Pattern Recognition*, pages 29–32. IEEE.

Zhang, Z., Tang, P., and Duan, R. (2015). Dynamic time warping under pointwise shape context. *Information sciences*, 315:88–101.

Zhang, Z., Tavenard, R., Bailly, A., Tang, X., Tang, P., and Corpetti, T. (2017). Dynamic time warping under limited warping path length. *Information Sciences*, 393:91–107.

Zhao, B., Lu, H., Chen, S., Liu, J., and Wu, D. (2017). Convolutional neural networks for time series classification. *Journal of Systems Engineering and Electronics*, 28(1):162–169.

Zhao, J. and Itti, L. (2015). Classifying time series using local descriptors with hybrid sampling. *IEEE Transactions on Knowledge and Data Engineering*, 28(3):623–637.

Zhao, J. and Itti, L. (2018). shapeDTW: Shape dynamic time warping. *Pattern Recognition*, 74:171–184.