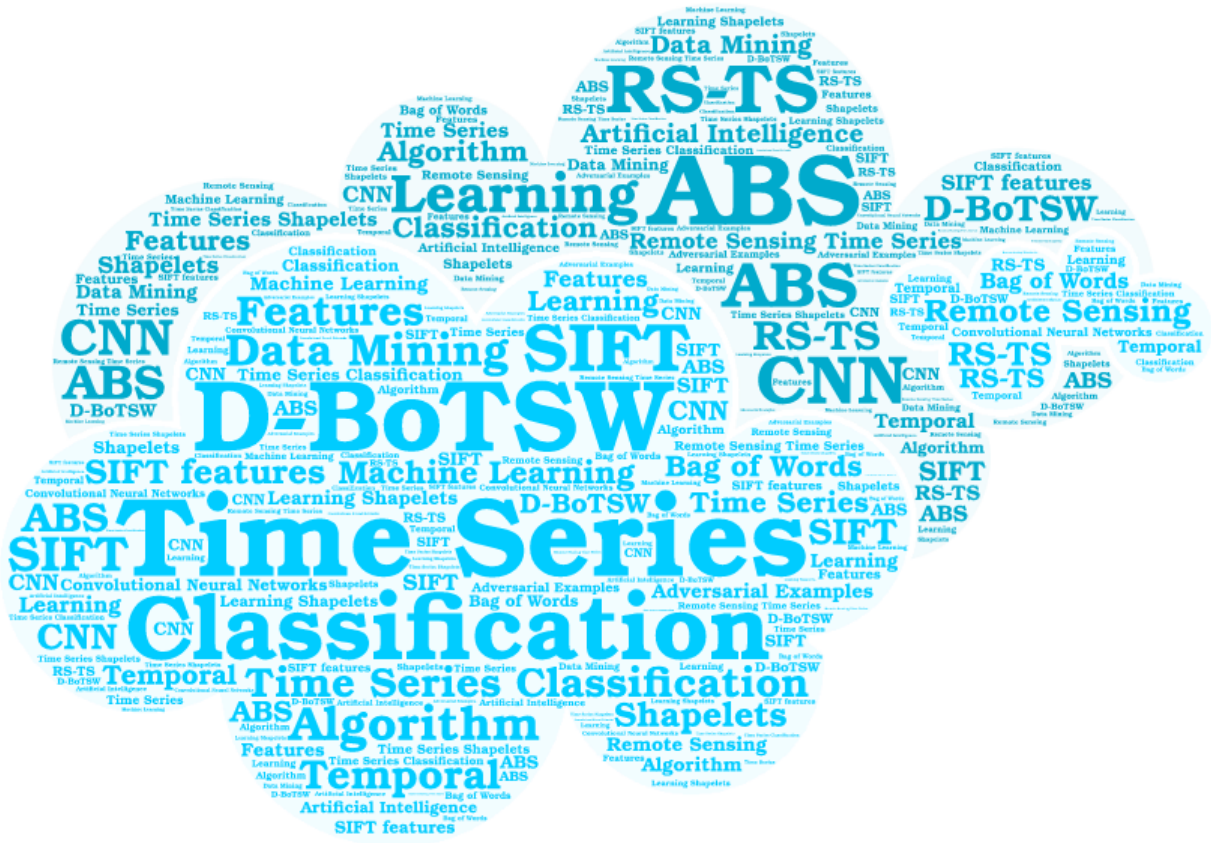
Time Series Classification Algorithms with Applications in Remote Sensing

Adeline Bailly



UNIVERSITE
BRETAGNE
LOIRE

THESE / UNIVERSITE RENNES 2
sous le sceau de l'Université Bretagne Loire
pour obtenir le titre de
DOCTEUR DE L'UNIVERSITE BRETAGNE LOIRE
Mention : Informatique
Ecole doctorale MathSTIC

Classification de Séries Temporelles avec Applications en Télédétection

présentée par
Adeline Bailly

préparée à l'Unité Mixte de Recherche 6554 CNRS
Université Rennes 2
Littoral Environnement Télédétection et Géomatique

Thèse soutenue le 25/05/2018
devant le jury composé de :

Pierre GANCARKI
Professeur à l'Université de Strasbourg / rapporteur
Emmanuel TROUVE
Professeur à l'Université Savoie Mont Blanc / rapporteur
Anthony BAGNALL
Senior Lecturer à l'Université d'East Anglia / examinateur
Maria RIFQI
Professeur à l'Université Panthéon-Assas Paris 2 / examinateur
Sébastien Lefèvre
Professeur à l'Université Bretagne Sud / directeur de thèse
Romain TAVENARD
Maître de conférences à l'Université Rennes 2 / co-directeur de thèse
Laetitia CHAPEL
Maître de conférences à l'Université Bretagne Sud / co-directrice de thèse

Time Series Classification Algorithms with Applications in Remote Sensing

Adeline BAILLY

Supervisors : Romain TAVENARD & Laetitia CHAPEL

Bailly, Adeline. Classification de séries temporelles avec applications en télédétection - 2018

RÉSUMÉ EN FRANÇAIS

Contents

La classification de séries temporelles a suscité beaucoup d'intérêt ces dernières années en raison de nombreuses applications telles que la surveillance des électrocardiogrammes, la reconnaissance vocale ou la modélisation environnementale. Dans cette thèse, notre but est de proposer de nouveaux algorithmes pour la classification de séries temporelles qui sont adaptés aux applications de télédétection mais également qui obtiennent de très bonne performance par rapport aux algorithmes de classification de séries temporelles standards. La classification des séries temporelles présente en effet un intérêt particulier pour la télédétection : celle-ci utilise des séries temporelles d'images satellitaires afin de générer automatiquement des cartes d'occupation des sols et de couverture terrestre [Clark et al., 2010; Hüttich et al., 2009; Wardlow and Egbert, 2008].

Ce résumé introduit les différentes problématiques abordées dans cette thèse. Dans un premier temps, nous développerons un court argumentaire afin de montrer l'intérêt de la classification de séries temporelles ainsi que des données issues de la télédétection. Puis nous développerons les travaux réalisés dans le cadre du doctorat. Ces travaux sont divisés en trois parties, chaque partie correspondant à un futur chapitre.

La classification de séries temporelles

L'étiquetage manuel de données est un processus long et coûteux, or le volume de données ne cesse de grandir. Afin de pouvoir utiliser ces données il est souvent nécessaire de les classer, cependant le volume est tel qu'il est impossible de le faire manuellement. L'utilisation d'algorithmes d'apprentissage est un moyen de résoudre ce problème : des données peuvent servir d'entrées à un algorithme afin d'être classées. Ces données peuvent être : des documents (du texte), des images, des nuages de points ou encore des séries temporelles.

Nous nous intéressons ici à la classification de données, pour laquelle on définit l'apprentissage comme l'étape de construction du modèle, et la prédiction comme l'étape de classification de nouvelles données. Dans le cadre de l'apprentissage supervisé, l'apprentissage sur des données d'entraînement (dont on connaît la classe d'appartenance) permet d'apprendre un modèle qui sera ensuite utilisé pour classer de nouvelles données. Le but des algorithmes de classification est d'apprendre un modèle qui minimisera le nombre d'erreur sur ces nouvelles données, ce modèle doit donc être capable de généraliser sur les données d'apprentissage. De plus, pour être exploitables, les classifieurs doivent être rapides dans l'étape de construction du modèle et/ou dans la prédiction d'une classe par le modèle. Il peut être intéressant de tenir compte du temps requis pour classer une nouvelle donnée. En effet certains algorithmes ont besoin de plus de ressources pour s'en-

traîner mais effectueront une classification rapide voire instantanée, c'est le cas pour les réseaux de neurones. Dans le cadre d'application où l'on s'entraîne une seule fois et où l'on classifie toujours avec le même modèle, choisir un algorithme qui classifie très rapidement peut être un bon choix, même si son temps d'apprentissage est plus long que d'autres algorithmes. Finalement, comme les données proviennent de nombreux domaines, il est peu probable qu'un algorithme soit le meilleur sur l'ensemble des jeux de données existants et à venir [Wolpert and Macready, 1997]. D'où la nécessité de choisir un algorithme adapté aux données utilisées.

La classification de séries temporelles On se place maintenant dans le cadre où nos données sont des séries temporelles pouvant être vues comme une suite de valeurs réelles et ordonnées. Deux séries temporelles enregistrées par un même capteur peuvent être différentes soit à cause d'un décalage temporel (en cas d'actions asynchrones) soit à cause d'une dilatation si les deux actions n'ont pas été réalisées à la même vitesse. Parmi les caractéristiques que doivent avoir les classifieurs de séries temporelles, on peut noter : la robutesse aux distorsions temporelles (telles que les décalages dans le temps et la dilatation) mais également des caractéristiques liées à la classification de données en général telles que la robutesse au bruit et aux valeurs aberrantes.

La classification de données issues de la télédétection La classification automatique de la couverture terrestre à partir de données satellitaires est un sujet important qui permet de produire des cartes représentant les différents types de sols (tels que eau, forêts ou villes). Cependant les changements rapides dans l'aménagement du paysage et l'utilisation des terres rendent nécessaire une mise-à-jour régulière des cartes, qu'il est impossible de faire de manière manuelle à l'échelle de la planète. On a donc besoin d'algorithmes de classification afin de pouvoir générer de manière régulière et automatique des cartes de couverture terrestre.

Le but de cette thèse est d'apporter de nouvelles approches pour la classification de séries temporelles, nous allons donc nous intéresser à un cas d'application précis : les séries temporelles d'images satellitaires. Chaque série temporelle représente l'évolution des valeurs d'un indice (*par exemple* mesurant le taux de végétation) au cours du temps pour une zone précise. La longueur de la série dépend de la durée d'acquisition et du temps de revisite du satellite, *par exemple* si l'on considère un satellite dont la résolution temporelle est de huit jours et une période d'acquisition d'un an, on aura une série temporelle environ de longueur 46. Grâce aux séries temporelles d'images satellitaires, on peut observer les profils des différents types de sols et notamment les profils de végétaux, qui diffèrent selon le type de plantation.

Il est donc nécessaire pour les communautées scientifiques de télédétection et d'apprentissage automatique de travailler ensemble à l'élaboration de méthodes adaptées aux données satellitaires. Dans cette thèse, nous avons choisi de nous concentrer sur des méthodes d'apprentissage permettant de prendre en compte les particularités des séries temporelles, avec des méthodes robustes aux décalages temporels, prenant en compte l'évolution des valeurs au cours du temps et qui permettent une interprétation des résultats obtenus.

Travaux sur les algorithmes de classification de séries temporelles

Dans le but d'améliorer les performances des algorithmes de classification de séries temporelles, il a été montré qu'un moyen simple était de transformer les séries en se basant sur des éléments discriminatoires tels que des caractéristiques locales [Lines et al., 2012]. C'est pourquoi, nous proposons deux algorithmes de classification de séries temporelles basées sur l'utilisation de caractéristiques locales. Ces caractéristiques correspondent aux différentes variations d'une série temporelle telles que les pics ou les vallées.

La méthode Dense Bag-of-Temporal-SIFT-Words : D-BoTSW

Dans notre première approche, correspondant au chapitre 2, nous représentons des séries temporelles sous la forme d'un ensemble de caractéristiques locales. Pour cela, on transforme les séries en histogramme comptant la fréquence d'apparition de ces caractéristiques locales, puis on donne cette nouvelle représentation à un classifieur afin de prédire sa classe d'appartenance.

De manière plus détaillée, notre algorithme se déroule de la façon suivante. Nous utilisons des descripteurs locaux basés sur la méthode SIFT, adaptés pour les données en une dimension et extraits à intervalles réguliers (on parlera d'extraction dense). Ces descripteurs sont basés sur le voisinage du point considéré et prennent en compte les différentes tendances de la série autour de ce point. Ils permettent donc d'identifier : les pics, les vallées, les tendances à la hausse, *c'est-à-dire* les différentes variations de chaque série. Afin de mieux représenter les séries temporelles, nous utilisons ensuite la représentation par sac de mots. L'idée est de générer un dictionnaire de *mots*, où chaque *mot* rassemble un ensemble de descripteurs qui sont proches : on aura donc des *mots* différents pour les pics et pour les vallées. Chaque série temporelle est donc transformée en un histogramme comptant la fréquence d'apparition de chaque mot. La taille de

l'histogramme et la taille du dictionnaire seront donc identiques. Les histogrammes servent d'entrées à un SVM linéaire pour l'étape de prédiction. Un fois le modèle appris (*c'est-à-dire* le dictionnaire et les poids du SVM), on peut prédire les classes de nouvelles séries temporelles en utilisant le dictionnaire afin de générer l'histogramme correspondant puis le classifieur appris.

L'algorithme D-BoTSW n'est pas adapté pour tous les problèmes de classification de séries temporelles. En effet, la représentation par sac de mots ignore l'ordre d'apparition des caractéristiques, D-BoTSW n'est donc probablement pas l'algorithme le plus approprié pour la classification de séries temporelles où cela importe. Au contraire, les décalages temporels dans les séries temporelles sont facilement gérés par D-BoTSW, de plus l'algorithme est robuste au bruit.

Le temps d'apprentissage dépend principalement de l'étape de génération du dictionnaire et d'apprentissage des paramètres du SVM. En effet, les descripteurs SIFT reposent sur des opérations simples et peu coûteuses ; et l'étape de transformation d'une série en histogramme est très rapide une fois le modèle généré. L'algorithme D-BoTSW peut donc être une bonne option pour les applications où l'on souhaite classer de nouvelles données en utilisant toujours le même modèle.

Des expériences approfondies, menées sur 85 jeux de données, montrent que notre méthode Dense Bag-of-Temporal-SIFT-Words (**D-BoTSW**) surpasse de façon significative quasiment tous les classificateurs de référence comparés. En effet, sur dix algorithmes de classification considérés comme références dans le domaine, des tests statistiques ont montré que D-BoTSW est significativement meilleur que six d'entre eux, équivalent à trois autres et uniquement battu par une seule méthode.

L'algorithme Adversarially-Built Shapelets : ABS

La deuxième approche, correspondant au chapitre 3, est basée sur l'apprentissage de sous-séries caractéristiques (*shapelets*) permettant de différencier les différentes classes de séries temporelles. Ye and Keogh [2009] ont proposé un premier algorithme basé sur les *shapelets*. L'un des freins majeurs à l'utilisation de cet algorithme est le temps d'apprentissage, en effet la recherche de shapelets est basée sur un algorithme force brute très gourmand en temps. Plusieurs algorithmes ont été proposés afin de réduire le temps de recherche des shapelets [Rakthanmanon and Keogh, 2013; Ye and Keogh, 2011]. Contrairement aux autres algorithmes basés sur ce principe, l'algorithme *Learning Time Series Shapelets* (LTS) apprend ces sous-séries au lieu de les rechercher. La méthode LTS est une des deux méthodes basées sur les shapelets obtenant les meilleurs performances, il est donc intéressant de proposer un moyen d'améliorer cette méthode.

Afin de comprendre la construction de notre méthode, il est nécessaire de connaître les étapes clés de l'algorithme LTS [Grabocka et al., 2014]. L'algorithme LTS est une méthode basée sur des caractéristiques locales qui cherche à apprendre des sous-séries discriminantes (*c'est-à-dire* les *shapelets*) à partir d'un ensemble de séries temporelles. La distance d'une série avec chaque shapelet est calculée, puis un classifieur linéaire est utilisé afin de prédire la classe d'appartenance de cette série. Il est donc nécessaire d'apprendre à la fois les shapelets mais également les paramètres du classifieur. Les auteurs de la méthode LTS proposent de le faire de manière itérative en actualisant en même temps les shapelets et les paramètres du classifieur.

Pour contruire l'algorithme ABS, nous montrons dans un premier temps qu'il est possible d'écrire l'algorithme LTS en tant que cas particulier d'un réseau neuronal convolutif (CNN). Pour cela, nous présentons les différentes couches d'un réseau de neurones convolutifs (*convolution*, *pooling* et *fully connected*) afin de faire le lien entre les deux approches. Il a été prouvé [Szegedy et al., 2014] que les CNNs sont vulnérables aux exemples adversaires. Les exemples adversaires sont des données générées dans le but de tromper un CNN : pour cela, on modifie de manière imperceptible mais déterministe une image bien classée par un CNN. Nous démontrons que comme les CNNs, l'algorithme LTS est suceptible d'être vulnérable aux exemples adversaires (des séries temporelles adversaires dans notre cas). Il a été établi que l'introduction d'exemples adversaires dans le processus d'apprentissage permettait de générer des réseaux plus robustes tout en améliorant légèrement les performances [Goodfellow et al., 2015]. Nous proposons donc une méthode dont l'apprentissage est basé à la fois sur des séries temporelles non-modifiées et sur des séries adversaires. À chaque itération, nous générons de nouvelles séries temporelles adversaires. Celles-ci permettent de générer des shapelets plus robustes et d'améliorer les performances globales de l'algorithme.

L'algorithme ABS est particulièrement intéressant pour les problèmes de séries temporelles où une ou plusieurs sous-séries permettent de bien différencier les différentes classes. Les algorithmes LTS et ABS étant des cas particuliers de réseaux de neurones convolutifs, il est donc possible d'implémenter des algorithmes ayant un temps d'apprentissage faible. Concernant le temps de prédiction, il est comme pour tous les réseaux de neurones très faible.

Des expériences montrent une amélioration significative de la performance entre l'algorithme LTS et notre proposition Adversarially-Built Shapelets (**ABS**). Comparés aux autres méthodes de classification de séries temporelles, ABS obtient de bonnes performances puisque sur onze algorithmes de classification (les dix précédentes ainsi que D-BoTSW), des tests statistiques ont montré que ABS est significativement meilleur que six d'entre eux, équivalent à deux autres et battu par trois (dont D-BoTSW).

Applications sur des jeux de données de séries temporelles issues de la télédétection

Le dernier chapitre se concentre sur les applications en télédétection. En effet, nous avons précédemment décrit l'intérêt de la classification automatique de séries temporelles pour les applications en télédétection. Nous avons donc choisi d'expérimenter sur deux jeux de données disponibles en ligne : *TiSeLac* et *Brazilian Amazon*.

TiSeLac Ce jeu de données a été proposé dans le cadre d'un challenge ayant pour but de rapprocher les communautés d'apprentissage automatique et de télédétection. Pour cela, des données, enregistrées par le satellite Landsat au niveau de l'île de la Réunion, ont été préparées afin de former des séries temporelles multi-variées (10 indices différents) associées à leur coordonnées. Nous avons réduit ce jeu de données à un seul indice (NDVI) afin de pouvoir l'exploiter avec des algorithmes travaillant sur des séries temporelles univariées en entrée. Le NDVI est un indice de végétation dont la valeur varie en fonction du type de sol, par exemple une végétation dense aura une valeur élevée de NDVI alors qu'un sol enneigé aura une valeur très basse.

Le jeu de données *TiSeLaC* contient à la fois des classes de végétation (telles que des forêts) ainsi que des classes non végétales (telles que des zones urbaines). De plus, les séries temporelles ont une grande variabilité intra-classe aussi bien temporellement parlant qu'en amplitude. La variabilité en amplitude est due à la topologie particulière de l'île de la Réunion aux reliefs très escarpés ainsi qu'aux différents climats présents sur l'île. La variabilité temporelle des séries issues de la télédétection est un phénomène répandu principalement dû à des dates de semis différentes et aux conditions climatiques.

Brazilian Amazon Ce jeu de données contient des séries temporelles provenant du satellite MODIS et extraites dans l'état du Mato Grosso au Brésil. C'est un exemple applicatif particulièrement intéressant puisque les données représentent différentes parcelles agriculturales issues des régions tropicales. Le jeu de donnés *Brazilian Amazon* est composé uniquement de classes végétales. L'indice végétal utilisé pour la représentation des séries temporelles est l'EVI, qui comparé au NDVI a l'avantage de ne pas saturer en présence de végétation dense.

Le jeu de données regroupe 5 classes différentes : (a) soja, (b) coton, (c) soja + millet, (d) soja + maïs, et (e) soja + coton. On peut différencier les cultures dites uniques (*c'est-à-dire* une culture par an), des cultures dites intensives ou doubles cultures (*c'est-à-dire* deux cultures / récoltes par an).

Chacun des jeux de données a été évalué sur plusieurs algorithmes de classification de séries temporelles. Bien que les méthodes basées sur des caractéristiques, telles que BOSS et D-BoTSW, ont montré des résultats prometteurs, ce sont les méthodes basées sur les shapelets (*c'est-à-dire* ABS et LTS) qui ont montré les meilleurs performances sur les deux jeux de données. Avec l'arrivée de nouveaux satellites ayant des résolutions spatiales et temporelles plus précises, il est très probable que la performance de ces algorithmes s'améliore dans le futur. En effet, une résolution temporelle plus fine permettra une représentation plus précise des differents cycles phénologiques, donc probablement de générer des caractéristiques plus discriminantes. Une résolution spatiale plus fine va permettre quant-à-elle de travailler avec des séries temporelles plus pures (*c'est-à-dire* qu'il y aura moins de mélange de classes dans chaque pixel).

Des modèles simplifiés des deux méthodes proposées dans ce manuscript (D-BoTSW et ABS) ont servi à illustrer les résultats obtenus sur *Brazilian Amazon*. Pour D-BoTSW, nous avons généré un dictionnaire réduit afin de représenter les différentes caractéristiques ainsi que les histogrammes associés. Pour ABS, nous avons généré un ensemble de shapelets ainsi que les poids du modèle linéaire. A partir des modèles, nous avons pu vérifier différentes hypothèses émises sur les données.

CONTENTS

LIST OF PUBLICATIONS

1. Adeline Bailly, Simon Malinowski, Romain Tavenard, Thomas Guyet, and Laetitia Chapel. **Bag-of-Temporal-SIFT-Words for Time Series Classification**. In *ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data*, Porto, Portugal, 2015.
2. Adeline Bailly, Simon Malinowski, Romain Tavenard, Laetitia Chapel, and Thomas Guyet. **Dense Bag-of-Temporal-SIFT-Words for Time Series Classification**. In *Advanced Analysis and Learning on Temporal Data (AALTD'15), Lecture Notes in Computer Science*, volume 9785, pages 17–30. Springer, 2016.
3. Adeline Bailly, Damien Arvor, Laetitia Chapel, and Romain Tavenard. **Classification of MODIS Time Series with Dense Bag-of-Temporal-SIFT-Words: Application to Cropland Mapping in the Brazilian Amazon**. In *IEEE International conference on Geoscience and Remote Sensing Symposium (IGARSS)*, pages 2300–2303, Beijing, China, 2016.
4. Adeline Bailly, Laetitia Chapel, Romain Tavenard, and Gustau Camps-Valls. **Nonlinear Time-Series Adaptation for Land Cover Classification**. *IEEE Geoscience and Remote Sensing Letters*, 14(6):896–900, 2017.
5. Zheng Zhang, Romain Tavenard, Adeline Bailly, Xiaotong Tang, Ping Tang, and Thomas Corpetti. **Dynamic Time Warping Under Limited Warping Path Length**. *Information Sciences*, 393:91–107, 2017.
6. Romain Tavenard, Simon Malinowski, Laetitia Chapel, Adeline Bailly, Heider Sanchez, and Benjamin Bustos. **Efficient Temporal Kernels between Feature Sets for Time Series Classification**. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery*, Skopje, Macedonia, 2017.
7. Adeline Bailly. **Remote Sensing Time Series datasets**, 2018. Available at `github.com/a-bailly/time_series_data`.

List of Symbols and Acronyms

1D	One Dimensional.
2D	Two Dimensional.
ABS	Adversarially-Built Shapelets.
BoP	Bag-of-Patterns.
BOSS	Bag-of-SFA-Symbols.
BoTSW	Bag-of-Temporal-SIFT-Words.
BoW	Bag-of-Words.
CNN	Convolutional Neural Networks.
COTE	Collective Of Transformation-based Ensembles.
D-BoTSW	Dense Bag-of-Temporal-SIFT-Words.
DDTW	Derivative Dynamic Time Warping.
DM	Data Mining.
DoG	Difference of Gaussians.
DTW	Dynamic Time Warping.
DTW-NN	1-Nearest Neighbor associated with Dynamic Time Warping.
ED	Euclidean Distance.
EDR	Edit Distance on Real Sequences.
ERP	Edit Distance on Real Penalty.
EVI	Enhanced Vegetation Index.
IDF	Inverse Document Frequency.
LANDSAT	Land Satellite.
LCSS	Longest Common Subsequences.
LTS	Learning Time series Shapelets.
MODIS	MODerate-resolution Imaging Spectroradiometer.
MSM	Move-Split-Merge.
NDVI	Normalized Difference Vegetation Index.

PROP	Proportional Elastic Ensemble.
RS	Remote Sensing.
SAX	Symbolic Aggregate approXimation.
SAX-VSM	Symbolic Aggregate approXimation - Vector Space Model.
SFA	Symbolic-Fourier Approximation.
SIFT	Scale-Invariant Feature Transform.
SSR	Signed Square Root.
TF-IDF	Term Frequency - Inverse Document Frequency.
TSBF	Time Series Bag-of-Features.
TWED	Time Warp Edit Distance.
VSM	Vector Space Model.
WDDTW	Weighted Derivative Dynamic Time Warping.
WDTW	Weighted Dynamic Time Warping.

INTRODUCTION

Motivation

This thesis aims at bringing new methodologies to classify time series data. We consider that a time series corresponds to an ordered sequence of real-valued datapoints representing the evolution of a quantity over time. They arise in many domains and have a wide range of applications. Classification aims at identifying amongst a list of categories to which category a new data belongs. Time series classification is of particular interest for the remote sensing community that uses satellite images time series in order to generate land use and land cover maps. Satellite images time series data have been widely used in order to classify vegetation [Clark et al., 2010; Hüttich et al., 2009; Wardlow and Egbert, 2008]. Indeed, time series data are able to represent the phenological cycle that differs according to the land cover type.

In the following, we present time series classification algorithms based on local features, such as peask or valleys, since results interpretation and analysis is of paramount interest for remote sensing applications. Moreover, feature-based approaches often lead to high performance. Our motivation is thus to bring machine learning and remote sensing communities closer by providing algorithms that are adapted for remote sensing applications and that also achieve high accuracy *w.r.t.* time series classification baselines.

A Need for Automatic Classification.

In the last decades, the amount of data generated by sensors, such as satellites or ECGs, is constantly and exponentially growing. In order to exploit this data, it needs to be categorized and labeled, which quickly became impossible to do manually. Hence, scalable automatic processes classifying data had to be developed.

The creation of classification algorithms is a way to solve this problem of new and numerous data to categorize. The first step is the training phase, during which the algorithm builds a model from already labeled data. During the prediction step, this model is applied to new, unlabeled data. Note that many data types can be used as inputs of such algorithms: documents (*e.g.* text), images, videos, point clouds or time series. The data may come from many different fields, such as biology, economy, or geology, and varies greatly in size and forms.

Classification algorithms aim at minimizing the prediction error on new data, to do so it is necessary to build models that generalize the data. Moreover, in order to be exploitable, classifiers need to be fast (learning and / or prediction steps) as well as to be robust to noise and outliers. Concerning the prediction step, the time required to classify a new data has to be taken into acocount. Indeed, some algorithms need more resources to train the model but will perform a fast or instantaneous classification, as it is the case for neural networks. If we plan to learn the model once and perform multiple classification using the same model, it can be interesting to choose an algorithm that predicts fastly, even if its learning time requires more resources. In order to take into account the intra-class variability (*i.e.* the differences between the time series of the same class), the classifiers must also be able to generalize well the information.

Also, the classification is more or less complex depending on the nature of the data. It is thus impossible to create an algorithm that obtains the best performance on all available data, as stated by the no free lunch theorem [Wolpert and Macready, 1997]. However, the theorem also states that the only way to find an algorithm more efficient than another is to adapt the algorithm to the problem.

Time Series Classification

Amongst all classification problems, we are interested in time series classification, for which dedicated algorithms range from the simple k-nearest-neighbor algorithm associated with Euclidean distance to ensemble methods dedicated to time series.

The temporal ordering inside time series makes them particularly interesting to study. The correlation between close points will be stronger than the one between distant points. Finally, temporal distortions such as time shifts and dilations can be present in time series, it is thus necessary to propose time series classification algorithms robust to these distortions.

Remote Sensing Applications.

For the last decades, Earth observation has been used to better understand our planet and to follow its evolution. The characterization of continental surface transformations such as deforestation, evolution of agricultural practices or urbanization, is essential to evaluate the impact of global warming, public policies, or conflicts. Remote sensing satellites acquire images over the entire surface of Earth, offering the possibility to produce an up-to-date land use / land cover map. Land use mapping is a crucial tool for remote sensing applications, such as global changes tracking or large-scale

crop monitoring.

Remote sensing data has been widely used in different forms such as images and time series. It has also been used to solve a large variety of problems such as monitoring agricultural parcels, mapping or environmental modeling. Automatic classification of remote sensing data is of high interest as the labeling step is limited (a) by the high cost, both in human resources and in time through field campaigns or labeling by experts, and (b) by the huge amount of data to label. Moreover, fast changes in landscape make necessary the frequent update of maps, which can not be done manually at a global scale.

The purpose of this thesis is to provide new approaches for the classification of time series, and we focus our applications on satellite images time series. Each remote sensing time series represents the evolution of an index value over time for a specific area. The length of the series depends on the acquisition period and the revisit time of the satellite: if we consider a satellite whose temporal resolution is of eight days and a one year acquisition period, we will have a time series of approximately 46 datapoints, which corresponds to the 46 images acquired by the satellite. Thanks to satellite images time series, it is possible to observe the profiles of different soil types and in particular vegetation profiles, which differ according to the flora.

It is therefore necessary for the scientific communities of remote sensing and machine learning to work together to develop methods adapted to satellite data. This thesis focuses on interpretable learning methods for time series, robust to time shifts and taking into account the different features (*e.g.* trends, peaks) of this type of data.

Contributions

In order to improve the performance of time series classification algorithms, transforming time series into a new representation based on discriminative features has shown to be effective [Lines and Bagnall, 2014; Lines et al., 2012]. Since characteristics such as peaks and valleys are interesting properties of time series, we aim to transform time series into a new representation, based on these characteristics, that can be used with *general* learning algorithms. Our approaches thus build on feature-based representations of time series.

Dense Bag-of-Temporal-SIFT-Words

Our first contribution is the Dense Bag-of-Temporal-SIFT-Words algorithm (**D-BoTSW**). It is based on two well-known and powerful methods:

SIFT features and Bag-of-Words representation. Basically, the algorithm combines the extraction of descriptive features at regular time steps in the time series with a histogram representation. The raw time series are transformed into a new representation that gathers the different descriptive characteristics into similarity groups, the new representation is then fed into a classifier.

The D-BoTSW algorithm can be divided into the following steps. First, we extract dense keypoints inside time series, *i.e.* we extracted a list of points at regular time step that serve as bases for our future descriptors. Then, we describe these keypoints using 1D-SIFT features, these feature use neighboring points in order to describe the neighborhood of each keypoints and to detect characteristics such as peaks or valleys inside time series. Once we have our descriptors, we generate a codebook, *i.e.* we gather similar features together in order to simplify the representation. To do so, we transform each time series into a new representation corresponding to a normalized histogram of words occurences, which allow us to easily compare two time series. Finally, we perform the classification on the new representation.

Since the Bag-of-Words representation ignores temporal order, temporal shifts in time series are easily handled by D-BoTSW. However, when ordering matters, D-BoTSW is probably not the most suitable algorithm. The proposed algorithm is robust to both noise and temporal shifts. Moreover, it exhibits high performance compared to state-of-the-art time series classification algorithms.

A detailed explanation of this algorithm as well as experimentations on it can be found in Chapter 2.

Adversarially-Built Shapelets

Our second contribution is an improved version of the Learning Time series Shapelets (LTS) algorithm. Shapelets are discriminative sub-series that are used to predict the class label. Many algorithms were proposed for shapelets discovery. However the LTS algorithm was the first to propose to learn them instead of searching them.

In order to understand the construction of our method, it is necessary to know a few key steps of the LTS algorithm [Grabocka et al., 2014]. The LTS algorithm seeks to learn a set of discriminative sub-series. To do so, the first step is to transform the time series into a set of features, where a feature corresponds to the distance between a series and a shapelet. The next step is to use these features (*i.e.* distances) in a linear classifier to predict the class label of this series. The authors of the LTS method propose to do it iteratively by updating at the same time the shapelets and the weights of the classifier using training series.

To build the ABS algorithm, the first part of our work was to demon-

strate that the LTS algorithm could be seen as an instance of a Convolutional Neural Network (CNN). Then, considering that CNNs can be fooled by adversarial examples (*i.e.* inputs generated to fool a classifier, close to the original), we prove that LTS can also be misled by adversarial examples. Finally, we propose a simple way to improve the LTS algorithm using adversarial training. Adversarial training consists in introducing adversarial examples during the training process in order to regularize the model and it has been shown that it can slightly improve the performance of a network.

This work is described is Chapter 3, with experimental proofs that ABS performs better than LTS.

Remote Sensing Applications

Many remote sensing applications use time series data such as land cover mapping or agricultural monitoring. Moreover the amount of remotely sensed data is constantly growing because of the new and upcoming satellite platforms available. It is thus important to bring machine learning and remote sensing communities closer. We present two remote sensing time series datasets we experiment on using time series baselines.

The first dataset, named *TiSeLaC*, was proposed as part of a challenge aiming to bring together the communities of machine learning and remote sensing. The time series were recorded by the Landsat satellite above Reunion Island and represent an interesting challenge to differentiate a large number and variety of classes.

The second dataset, called *Brazilian-Amazon*, contains EVI time series from the MODIS satellite extracted in the state of Mato Grosso in Brazil. This is a particularly interesting application example since the data represent different agricultural parcels and come from tropical regions. On the one hand, one can differentiate the so-called unique cultures (*i.e.* one crop per year) and double cultures (*i.e.* two crops / crops per year). On the other hand, the dataset groups 5 different classes: 2 unique culture classes and 3 double culture classes.

Simplified models of our two proposed algorithms (D-BoTSW and ABS) were used in order to illustrate the results obtained on the *Brazilian Amazon* dataset. For D-BoTSW, we have generated a reduced dictionary that represents the different characteristics as well as per-class histograms. For ABS, we generated a set of shapelets as well as the weights of the linear model. Using the information extracted from the models, we were able to verify different hypotheses on the data.

The characteristics of this data are developed both in term of machine learning (noise, high intra class variability) and remote sensing (vegetation index, climate) points of view. Results interpretation and analysis is provided for both of our algorithms.

Organisation

The purpose of this thesis is to bring new solutions for the classification of time series data, with some applications in remote sensing. In order to fulfill these goals, this manuscript will be organized in four chapters.

The first chapter gives an overview of time series classification algorithms in the literature, from basic concepts to advanced algorithms. These algorithms are gathered according to their specificities then compared to other baselines.

The second chapter introduces a first time series classifier based on feature extraction. This algorithm named *Dense Bag-of-Temporal-SIFT-Words* relies on two well-known and powerful methods: SIFT features (adapted to our data) and the Bag-of-Words representation.

The third chapter presents a second time series algorithm, which is based on an already existing classifier: *Learning Time series Shapelets* (LTS), which is also a feature-based algorithm. Our proposed method is called *Adversarially-Built Shapelets* and rely on the introduction of adversarial time series.

Finally, the fourth and last chapter focus on remote sensing applications with different datasets. The first one gathers agricultural time series from Brazil and the second one a variety of time series extracted from Reunion island.

1

TIME SERIES CLASSIFICATION STATE-OF-THE-ART

Contents

This chapter gives an overview of the state of the art *Time Series Classification* (TSC) algorithms, from basic concepts to advanced algorithms. First, we introduce notations and definitions related to TSC. Many algorithms use raw time series whereas others change the time series representation before the classification step. We thus categorize the different time series classification models that exist and review some of them with their most relevant characteristics. We start with the distance-based time series classification models, then we review the feature-based, finally we introduce the ensemble time series classifiers. In this chapter, we only aim at categorizing the different types of time series classification algorithms as well as detailing the most famous and most competitive ones ; and refer the curious reader to the numerous existing papers on this problem [Bagnall et al., 2016; Fu, 2011]. We then present metrics for model evaluation. We conclude this chapter by briefly reviewing other challenging time series problems.

1.1 *Basic Definitions and Notations*

In the following, we provide definitions as well as useful notations in order to define the time series classification problem.

Definition 1.1 (Time Series)**.** *A time series $\boldsymbol{x}_i$ of length n is an ordered sequence of valued data points, measured at regular time intervals, noted*

$$\boldsymbol{x}_i = x_{i,1}, x_{i,2}, \ldots, x_{i,n} \tag{1.1}$$

where $\boldsymbol{x}_{i,j}$ denotes the j-th element of the time series $\boldsymbol{x}_i$.

Time series can be univariate or multivariate. A time series containing records of a single (*respectively* of multiple) observation(s) is referred to as univariate (*respectively* multivariate). Since we focus on univariate time series in this document, the term *time series* should thus be considered as univariate time series. A note will be added for algorithms that can be used with multivariate time series.

Let $\mathcal{X}$ be a set of time series $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$. $\mathcal{X}$ can be referred to as *dataset*. For simplicity, we assume that all time series in the same dataset have the same length n. If a method can be applied on time series with different lengths, it will be specified.

The association of a time series with a label $(\boldsymbol{x}_i, y_i)$ corresponds to a *labeled time series*, whereas a time series without a label $(\boldsymbol{x}_i)$ is called an *unlabeled time series*.

Definition 1.2 (Labeled Dataset)**.** *A labeled dataset $\mathcal{D}$ corresponds to a set of time series associated with a set of labels (one label for each time series). Label y_i represents the class of the i-th time series $\boldsymbol{x}_i$.*

$$\mathcal{D} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_N, y_N)\} \tag{1.2}$$

In the following, we denote *labeled dataset* as a *dataset*, and a set of datasets is referred to as *database*. A time series in a dataset $\mathcal{D}$ can also be called an instance of $\mathcal{D}$.

A classification process aims at putting together individual objects or data into classes, categories or groups based on their charasteristics. Classification can be categorized into supervised, unsupervised and semi-supervised classifications. In this thesis, we focus on supervised classification, whose purpose is to correctly assign a class to a new, unseen, unlabeled data. Supervised classification first step corresponds to the training step. During training, the classification algorithm learns a model based on labeled data. For parametric methods, the best set of parameters θ is also learned at this step. This second step is called classification or prediction. In order to correctly classify new, unseen data, the classifiers must generalize well from the training data to new cases and situations.

Definition 1.3 (Classifier)**.** *A classifier is a function f that maps the time series from their original space to the space of possible class labels, noted*

$$f : \mathcal{X} \longrightarrow \{1, \ldots, m\}^N \quad or \quad f : \boldsymbol{x}_i \longmapsto \{1, \ldots, m\} \tag{1.3}$$

$f(\boldsymbol{x}_i)$ is called the prediction of the class label for the time series $\boldsymbol{x}_i$. If $f(\boldsymbol{x}_i) = y_i$ then the classifier makes the right prediction.

To evaluate the performance of a classifier, a common method is to split a dataset $\mathcal{D}$ into two datasets $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$. The classifier will learn on $\mathcal{D}_{\text{train}}$, whereas $\mathcal{D}_{\text{test}}$ will stay unseen during training and will be used to evaluate the classifier.

Classification Challenges The purpose of any classifier is to find the function f that maximizes the accuracy given $\mathcal{D}_{\text{train}}$. Considering that $\mathcal{D}_{\text{test}}$ is not known during f's training, as well as it differs from $\mathcal{D}_{\text{train}}$ due to many reasons depending on the data, this task is far from trivial. We assume in the following that $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$ share both the same data distribution and the same class label sets. Being able to generalize well on unseen and unlabeled data is a challenging problem that many researchers are working on in order to bring new – more accurate – solutions. If we also consider time series data which arise in a wide range of domains and applications, such as medicine (ElectroCardioGram (ECG)), environmental applications (Land Cover Maps), object recognition or economics with different characteristics (*e.g.* length), we can notice that there is no best single method. This intuition is confirmed by the *no free lunch* theorem [Wolpert and Macready, 1997]. This theorem states that there is no method, applicable to all problems, that can always optimizes a cost function ; and that the only way to find an algorithm more efficient than another is to adapt the algorithm to the problem.

Moreover, in order to be exploitable, classifiers need to be

- fast,
- robust to noise and outliers,

Indeed, there are more and more data collected that require fast classifiers to process them. On one hand, parametric methods, while more adaptative since they can better fit the characteristics of the data, might be a bit slower during training considering that they need to select the best parameter(s). On the other hand, one should also consider the time required to classify a new data, indeed some algorithms which need more time to train will perform fast classification, *e.g.* Time Series Shapelets [Ye and Keogh, 2011]. Classifiers also need to be robust to noise in order to take into account the intra-class variability. Outliers can appear in the data (*e.g.* due to acquisition problems) making some classifiers failing at classifying them into the right category. It is thus important to build classifiers that are robust to noise and outliers. In the following, we consider a set of (non-exhaustive) characteristics in order to compare the different similarity measures and classifiers, such as

- the ability to deal with temporal distortions (such as time shifts and dilation),
- the robustness to noise and outliers,
- the computational cost,
- the number of parameters.

TSC versus Vector Classification Vector classification problems can sometimes be seen as similar to TSC problems, however it is important to notice important differences between these two classification tasks. In both cases, we aim at classifying an object, where this object can be considered as a vector of values. An important feature to consider is the relation of temporal order between points of the time series. A time series corresponds to values measured at regular time step where the ordering matters, it is thus not possible to change the order of elements in time series. On the other side, elements of a vector correspond to values of different features such as dimension, it is thus possible to change their order without any impact on the classification. Finally, while time series length may vary, the vector length must stay the same into a dataset.

In TSC problems, we will see that the temporal structure of the data often plays an important role in order to discover / extract features that enable one to discriminate the classes.

1.2 *Distance-based Time Series Classifiers*

The distance-based methods compute point-to-point distances between time series. In this section, we review the most relevant (dis)similarity mea-

sures as well as distance based classifiers for time series classification. In order to perform the classification step, these distances then need to be associated with a classifier such as k-Nearest Neighbors (kNN) or Support Vector Machines (SVM).

1.2.1 *Dissimilarity Measures*

The notion of similarity is a fundamental concept in scientific communities. Its purpose is to compare two instances by computing a single value quantifying their similarity. We consider in the following that instances are either vectors or time series. A similarity (*respectively* dissimilarity) measure is a real-valued function that measures how close (*respectively* dissimilar) two instances are to each other. The closer the instances are, the larger the similarity is. There exists many (dis)similarity measures and definitions of it, amongst them we denote a particular case: distance metrics.

Definition 1.4 (Distance Metric)**.** *A distance metric d between two instances $\boldsymbol{x}_i$ and $\boldsymbol{x}_\ell$ must follow the following rules:*

1. *Non-negativity:* $d(\boldsymbol{x}_i, \boldsymbol{x}_\ell) \geq 0$
2. *Identity:* $d(\boldsymbol{x}_i, \boldsymbol{x}_\ell) = 0$ *if and only if* $\boldsymbol{x}_i = \boldsymbol{x}_\ell$
3. *Symmetry:* $d(\boldsymbol{x}_i, \boldsymbol{x}_\ell) = d(\boldsymbol{x}_\ell, \boldsymbol{x}_i)$
4. *Triangle inequality:* $\forall \boldsymbol{x}_z : d(\boldsymbol{x}_i, \boldsymbol{x}_z) + d(\boldsymbol{x}_z, \boldsymbol{x}_\ell) \geq d(\boldsymbol{x}_i, \boldsymbol{x}_\ell)$

The distance metric d is a function

$$d : \mathcal{X} \times \mathcal{X} \longrightarrow [0, +\infty) \tag{1.4}$$

Many techniques, that can be used for classification such as metric trees (*e.g.* M-Tree, GNAT [Ratanamahatana et al., 2005]), can only be used with distance metrics. Using distance metrics can thus be an advantage compared to using non-distance metrics.

1.2.1.1 *Based on $\mathcal{L}_p$ Distances*

$\mathcal{L}_p$ distances correspond to a set of functions where p is a strictly positive real number ($p \in \mathbb{R}_+^*$). The $\mathcal{L}_p$ distance, or Minkowski distance, can be considered as a generalization of Manhattan ($p = 1$), Euclidean ($p = 2$) and Chebyshev ($p = \infty$) distances. The $\mathcal{L}_p$ norm is defined as

$$\|\boldsymbol{z}\|_p = \left(\sum_{i=1}^{n} |z_i|^p \right)^{1/p} \tag{1.5}$$

The $\mathcal{L}_p$ distance (or *Minkowski distance*) between two sequences $\boldsymbol{z}$ and $\boldsymbol{v}$ is defined as

$$d_p(\boldsymbol{z}, \boldsymbol{v}) = \|\boldsymbol{z} - \boldsymbol{v}\|_p = \left(\sum_{i=1}^{n} |z_i - v_i|^p \right)^{1/p} \tag{1.6}$$

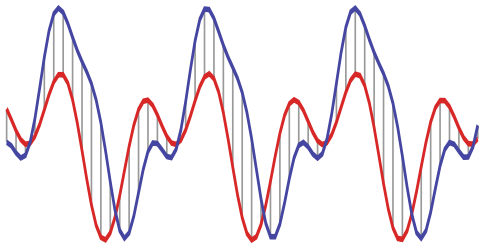


Figure 1.1 – Representation of Euclidean Distance between two Time Series

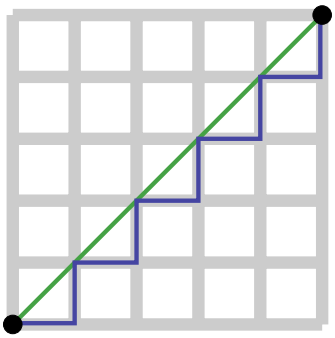


Figure 1.2 – Comparison of Manhattan Distance (blue) and Euclidean Distance (green)

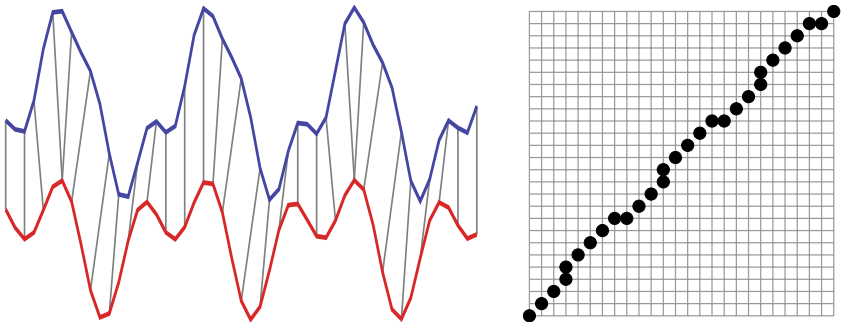
Euclidean Distance The most famous and most used distance measure is the Euclidean Distance (ED).

$$d_{\text{ED}}(\boldsymbol{z}, \boldsymbol{v}) = \|\boldsymbol{z} - \boldsymbol{v}\|_2 = \sqrt{\sum_{i=1}^{n} (z_i - v_i)^2} \tag{1.7}$$

The Figure 1.1 represents the ED between the red and the blue time series. The ED corresponds to the sum of points to points distance, as represented by the vertical lines in the figure, ED is thus visually easily interpretable. If we compare ED to Manhattan ($p = 1$, also called city-block) distance, ED can be viewed as crow flies distance (see Figure 1.2). ED is a simple, fast and parameter free distance metric, however it is sensitive to temporal shifts, outliers and noise.

1.2.1.2 *Dynamic Time Warping Measure*

$\mathcal{L}_p$ distances are easy to compute, however they are only appropriate when there is no time distortions between time series. It is indeed not possible for $\mathcal{L}_p$ distances to capture the similarity between time series when patterns inside them are shifted (*i.e.* do not match point to point). In order



(a) DTW distance (b) Warping path

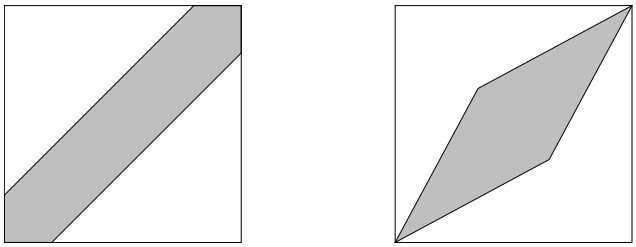
Figure 1.3 – Representation of DTW distance between two time series and its optimal warping path

to tackle shifting issues inside time series, Sakoe and Chiba [1970, 1978] introduced a dynamic programming approach called Dynamic Time Warping (DTW). DTW is another popular way to measure the similarity for time series, it has elastic properties *i.e.* can deal with distortions and shifts in the time axis. A representation of alignment between two time series using DTW algorithm can be found on Figure 1.3a (where time series are shifted in the y axis for better visualization). It can be seen that peaks from blue time series match with peaks from red time series, and valleys are aligned together. The optimal warping path, which corresponds to the alignment minimizing the overall cost (*i.e.* the DTW distance), is represented on Figure 1.3b. When time series are aligned, DTW measure is equivalent to ED.

DTW is a sequence alignment method that can handle temporal shift and compare time series with different lengths. However, it does not respect the triangle inequality thus is not a distance metric, is sensitive to outliers, noise, and has a quadratic computational time. DTW can be speeded up, *e.g.* using global constraints on warping path.

DTW algorithm Given two time series $\boldsymbol{x}_i$ and $\boldsymbol{x}_\ell$ (of length n), we form a n-by-n grid where each grid-point (j, k) corresponds to the alignment between $x_{i,j}$ and $x_{\ell,k}$. In order to compute the overall cost, we first have to compute the point-to-point cost. The warping path $\boldsymbol{w}$ corresponds to a sequence $\boldsymbol{w} = w_1, w_2, \ldots, w_L$, where w_m correspond to a pair (j_m, k_m) (indicating a match between x_{i,j_m} and x_{ℓ,k_m}). The space of possible warping paths $\mathcal{W}$ should satisfy the following three conditions:

1. **Monotonicity**, *i.e.* the points must be ordered in time:
   $j_{m-1} \leq j_m$ and $k_{m-1} \leq k_m$,
2. **Continuity**, *i.e.* the steps in the grid should be consecutive:
   $j_m - j_{m-1} \leq 1$ and $k_m - k_{m-1} \leq 1$,



(a) Sakoe-Chiba band (b) Itakura parallelogram

Figure 1.4 – Representation of global constraints for Dynamic Time Warping

3. **Boundary condtions**, *i.e.* the warping path should start at the beginning and finish at the end of time series: $w_1 = (1, 1)$ and $w_L = (n, n)$

The overall cost c of a warping path $\boldsymbol{w}$ between $\boldsymbol{x}_i$ and $\boldsymbol{x}_\ell$ is defined as

$$c_{\boldsymbol{w}}(\boldsymbol{x}_i, \boldsymbol{x}_\ell) = \sum_{m=1}^{L} d(x_{i,j_m}, x_{\ell,k_m}) \tag{1.8}$$

The optimal warping path between $\boldsymbol{x}_i$ and $\boldsymbol{x}_\ell$ is the warping path having the minimal cost amongst all possible warping paths $\mathcal{W}$. The DTW distance corresponds to the cost of the optimal warping path.

$$\text{DTW}(\boldsymbol{x}_i, \boldsymbol{x}_\ell) = \min_{\boldsymbol{w} \in \mathcal{W}} c_{\boldsymbol{w}}(\boldsymbol{x}_i, \boldsymbol{x}_\ell) \tag{1.9}$$

Global constraints on time warping Finding the optimal DTW path can be time consuming, to solve this problem many speed-up techniques have been proposed. Some of them are based on restricting the space of possible warping paths. Amongst them, the Sakoe-Chiba band [Sakoe and Chiba, 1978] and the Itakura parallelogram [Itakura, 1975] are the most popular. The *Sakoe-Chiba band* narrows down the set of possible warping paths by limiting them thanks to the following condition: $|j_m - k_m| \leq r$, where r corresponds to the window length ($r \in \mathbb{R}^+$). Basically, the Sakoe-Chiba band allows a maximum time shift of length r. As it can be observed on Figure 1.4a, the Sakoe-Chiba band forms a *band* around the diagonal and has a fixed width (horizontal and vertical one). Admissible warping paths are included into the grey section of the Figure 1.4a. The *Itakura parallelogram* is an alternative to the Sakoe-Chiba band, which also limits the number of possible warping paths. The Itakura parallelogram allows small shifts at the beginning and at the end of the time series, and a larger distortion around the middle of the time series (corresponding to the smaller diagonal of the parallelogram). See Figure 1.4b, for a representation of possible warping paths (in grey).

Derivative Dynamic Time Warping An extension of DTW called Derivative DTW (DDTW) was proposed by Keogh and Pazzani [2001]. This extension is based on the observation that two time series may also have differences in the y-axis. DTW can sometimes produce *singularities* in order to explain these differences. To solve this issue, Keogh and Pazzani [2001] thus proposed DDTW which considers the higher level of feature shape by estimating derivatives of time series data. DDTW can be seen as DTW where each time series $\boldsymbol{x}_i$ is replaced by its derivative in a preprocessing step. DDTW uses the distance between the derivatives instead of distance between time series, which reduces the number of singularities. One can easily apply global constraints on DDTW as well as proposed speed-up techniques for DTW due to their similar process.

Weighted Dynamic Time Warping Jeong et al. [2011] proposed another extension of DTW called Weighted Dynamic Time Warping (WDTW). WDTW can be viewed as a penalty-based DTW, which takes the time difference between two points into account when computing their distances. The closer the points are, the smaller weight is imposed (*i.e.* the less penalty is imposed). Large shifts, which are associated with larger weights, are more penalized than smaller shifts. WDTW algorithm is identical to DTW algorithm, the only difference is the distance measure used to compare two data points. One advantage of WDTW is than it can easily be extended to variants of DTW: *e.g.* the association of DDTW with WDTW (Weighted Derivative Dynamic Time Warping: WDDTW) has been proposed in [Jeong et al., 2011]. One can also easily apply global constraints as well as proposed speed-up techniques for DTW on WDTW.

Summary on Dynamic Time Warping DTW can be used to measure the similarity between time series with temporal shifts and distortions. Many speed-up techniques have been proposed to reduce the quadratic complexity of DTW. Amongst them global constraints enables one to reduce the complexity of DTW (*e.g.* for Sakoe-Chiba band from $\mathcal{O}(n^2)$ to $\mathcal{O}(n \cdot r)$), however it does not make DTW less sensitive to noise, outliers, nor make it a metric. To reduce the sensitivity to noise and outliers, some enhancement of DTW were proposed such as DDTW and WDTW (that can also both handle time series with different lengths).

1.2.1.3 *Edit Distance*

Edit distance is a method that quantifies how dissimilar two strings are. Strings can be considered as a sequence of discrete values, where each value is a letter or symbol of the alphabet. The distance between two strings can be seen as the smallest number of added, deleted, substituted (or transposed) symbols as stated in [Jurafsky and Martin, 2000, Part I, Chapter 5].

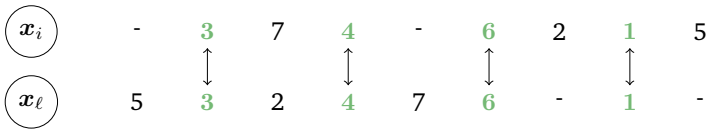


Figure 1.5 – Example of LCSS distance between two sequences

Deleting a symbol is equivalent to adding a symbol in the other time series. In the following, an added symbol will be referred to as a *gap* element.

Edit distance can be used in order to compare time series. For instance, a time series can be transformed into a sequence of discrete values, where each discrete value correspond to a range of continuous values, *i.e.* into a *string*. Another possibility is that, instead of checking whether two symbols from strings are identical, point-to-point in time series distance can also be compared to a predefined threshold.

The following edit distances are able to compare time series with different lengths as well as can be used for multivariate time series. Moreover, Edit-based distances can take into account outliers, contrary to DTW.

Distance Based on Longest Common SubSequences The basic idea of the Longest Common SubSequences (LCSS) problem is to find the longest common subsequence between two sequences, where we allow some elements to be unmatched or left out. LCSS distance allows only insertion and deletion of elements, not substitution. In order to be used for time series (which are continuous variables), we need to relax the constraint on the equality in the LCSS algorithm. Indeed, two continuous values are unlikely to be exactly equal, however they can be closed to each other ($|x_{i,j} - x_{\ell,k}| < \delta$).

Definition 1.5 (LCSS Distance)**.** *Given two time series $\boldsymbol{x}_i$ and $\boldsymbol{x}_\ell$, the LCSS distance is defined as*

$$d_\delta(\boldsymbol{x}_i, \boldsymbol{x}_\ell) = \begin{cases} 0 & \text{if } \boldsymbol{x}_i \text{ or } \boldsymbol{x}_\ell \text{ is empty} \\ 1 + d_\delta\big(h(\boldsymbol{x}_i), h(\boldsymbol{x}_\ell)\big) & \text{if } |x_{i,-1} - x_{\ell,-1}| < \delta \\ \max\big(d_\delta(h(\boldsymbol{x}_i), \boldsymbol{x}_\ell), d_\delta(\boldsymbol{x}_i, h(\boldsymbol{x}_\ell))\big) & \text{otherwise} \end{cases} \tag{1.10}$$

Where δ is the tolerance threshold and $h(\boldsymbol{x}_i)$ corresponds to the head function, i.e. *to the time series $\boldsymbol{x}_i$ without its last point $x_{i,-1}$.*

Given two sequences $\boldsymbol{x}_i = \{3, 7, 4, 6, 2, 1, 5\}$ and $\boldsymbol{x}_\ell = \{5, 3, 2, 4, 7, 6, 1\}$, their LCSS is $\{3, 4, 6, 1\}$, as shown in Figure 1.5. In the time series literature, LCSS distance has been applied on univariate time series [Das et al., 1997], as well as on multivariate time series [Vlachos et al., 2002]. LCSS is an alignment method that can handle outliers and temporal shifts, however it is not a metric and has a quadratic complexity.

Edit Distance on Real Sequences Chen et al. [2005] proposed the Edit Distance on Real Sequences (EDR). In order to take into account the continuous nature of the variables, the constraint on the equality is also relaxed.

Definition 1.6 (Edit Distance on Real Sequences)**.** *Given two time series $\boldsymbol{x}_i$ and $\boldsymbol{x}_\ell$, the EDR between two data points is defined as*

$$d_{EDR*}(x_{i,j}, x_{\ell,k}) = \begin{cases} 0 & \textit{if } |x_{i,j} - x_{\ell,k}| \leq \delta \\ 1 & \textit{if } x_{i,j} \textit{ or } x_{\ell,k} \textit{ is a gap} \\ 1 & \textit{otherwise} \end{cases} \tag{1.11}$$

where δ is the tolerance threshold.

In order to compute the distance $d_{\text{EDR}}(\boldsymbol{x}_i, \boldsymbol{x}_\ell)$ between two time series *respectively* of length n and m, we can use a dynamic programming style:

$$d_{\text{EDR}}(\boldsymbol{x}_i, \boldsymbol{x}_\ell) = \begin{cases} n & \text{if } m = 0 \\ m & \text{if } n = 0 \\ \min \left\{ \begin{array}{c} d_{\text{EDR}}(r(\boldsymbol{x}_i), r(\boldsymbol{x}_\ell)) + d_{\text{EDR}*}(x_{i,1}, x_{\ell,1}), \\ d_{\text{EDR}}(\boldsymbol{x}_i, r(\boldsymbol{x}_\ell)) + 1, \\ d_{\text{EDR}}(r(\boldsymbol{x}_i), \boldsymbol{x}_\ell) + 1 \end{array} \right\} & \text{otherwise} \end{cases} \tag{1.12}$$

Where $r(\boldsymbol{x}_i)$ corresponds to the time series $\boldsymbol{x}_i$ without its first point $x_{i,1}$.

Chen et al. [2005] showed that EDR supports local time shifts, is is more robust to noise than DTW and LCSS, however its complexity is quadratic and it is not a distance metric.

Edit Distance on Real Penalty The Edit Distance on Real Penalty (ERP) has been proposed by Chen and Ng [2004]. ERP is a distance metric, which is associated with a penalization. This penalization is a constant value g that is applied when gaps are created (to create an optimal alignment).

Definition 1.7 (Edit Distance on Real Penalty)**.** *Given two time series $\boldsymbol{x}_i$ and $\boldsymbol{x}_\ell$, the ERP between two data points is defined as*

$$d_{ERP*}(x_{i,j}, x_{\ell,k}) = \begin{cases} |x_{i,j} - x_{\ell,k}| & \textit{if neither } x_{i,j} \textit{ nor } x_{\ell,k} \textit{ are gaps} \\ |x_{i,j} - g| & \textit{if } x_{\ell,k} \textit{ is a gap} \\ |x_{\ell,k} - g| & \textit{if } x_{i,j} \textit{ is a gap} \end{cases} \tag{1.13}$$

where g a constant value.

In order to compute the distance $d_{\text{ERP}}(\boldsymbol{x}_i, \boldsymbol{x}_\ell)$ between two time series,

we can again use a dynamic programming style:

$$d_{\text{ERP}}(\boldsymbol{x}_i, \boldsymbol{x}_\ell) = \begin{cases} \sum_{k=1}^{n} |x_{\ell,k} - g| & \text{if } n = 0 \\ \sum_{j=1}^{n} |x_{i,j} - g| & \text{if } n = 0 \\ \min \begin{cases} d_{\text{ERP}}(r(\boldsymbol{x}_i), r(\boldsymbol{x}_\ell)) + d_{\text{ERP}*}(x_{i,1}, x_{\ell,1}), \\ d_{\text{ERP}}(\boldsymbol{x}_i, r(\boldsymbol{x}_\ell)) + d_{\text{ERP}*}(x_{\ell,1}, g), \\ d_{\text{ERP}}(r(\boldsymbol{x}_i), \boldsymbol{x}_\ell) + d_{\text{ERP}*}(x_{i,1}, g) \end{cases} & \text{otherwise} \end{cases} \tag{1.14}$$

Where g the penalization term (*i.e.* the *gap* – set to 0 by authors).

ERP is based on EDR but uses actual distance between points instead of fixed values. ERP is able to handle outliers and temporal shift. Plus ERP can be indexed since it is a distance metric, thus it can be easily fasten despite a quadratic complexity.

1.2.1.4 *Time Warp Edit Distance*

Marteau [2009] proposed the Time Warp Edit Distance (TWED) another elastic similarity metric (*i.e.* can handle shifts in time series). TWED elasticity is controlled by a parameter called *stiffness* γ using time stamp differences as part of the local matching cost. An infinite stiffness makes the distance equivalent to the ED, whereas a null stiffness is similar to DTW without warping window. A second parameter is involved in TWED, it corresponds to penalty λ for insertion or deletion operations, similarly to the previously seen edit distance. TWED has three operations: *delete_*x_i, *delete_*x_ℓ and *match*.

Definition 1.8 (Time Warp Edit Distance)**.** *Given two time series $\boldsymbol{x}_i$ and $\boldsymbol{x}_\ell$, the TWED is defined as*

$$d_{\lambda,\gamma}(\boldsymbol{x}_i, \boldsymbol{x}_\ell) = \\ \min \begin{cases} d_{\lambda,\gamma}(h(\boldsymbol{x}_i), \boldsymbol{x}_\ell) & + d(x_{i,-1}, x_{i,-2}) + \lambda + \gamma \cdot d(t_{i,-1}, t_{i,-2})) \\ d_{\lambda,\gamma}(\boldsymbol{x}_i, h(\boldsymbol{x}_\ell)) & + d(x_{k,-1}, x_{\ell,-2}) + \lambda + \gamma \cdot d(t_{\ell,-1}, t_{\ell,-2})) \\ d_{\lambda,\gamma}(h(\boldsymbol{x}_i), h(\boldsymbol{x}_\ell)) & + d(x_{i,-1}, x_{\ell,-1}) + d(x_{i,-2}, x_{\ell,-2}) \\ & + \gamma \cdot (d(t_{i,-1}, t_{\ell,-1}) + d(t_{i,-2}, t_{\ell,-2})) \end{cases} \tag{1.15}$$

where λ is the constant penalty added in case of a deletion ($\lambda \geq 0$), γ the stiffness parameter ($\gamma \geq 0$), $h(\boldsymbol{x}_i)$ corresponds to the time series $\boldsymbol{x}_i$ without its last point $x_{i,-1}$ (which time instant is denoted $t_{i,-1}$) and d a $\mathcal{L}_p$ metric.

The TWED metric can handle temporal distortions and is robust to noise, yet it is not a parameter free distance. Nonetheless, thanks to the stiffness parameter, TWED can control the admissible shift in time series, which is very useful on data where the temporal location matters. Note that TWED can both handle multivariate time series as well as different lengths ones.

1.2.1.5 *Move-Split-Merge*

Stefan et al. [2013] introduced the Move-Split-Merge (MSM), which is based on a set of operations: *move*, *split* and *merge*. *move* is equivalent to a substitution (*i.e.* changes the value of an element), *split* will double an element (*i.e.* add the same element right after itself) and *merge* merges two consecutive elements into one. Each operations has an associated cost c, except the *move* operation which cost corresponds to the absolue value of the difference between the original element and the new one. The distance between two time series corresponds to the cost of the cheapest sequence of operations that transforms the first time series into the second one. MSM is a distance metric which is robust to temporal shifts and which is able to deal with time series of different lengths and multivariate time series.

1.2.1.6 *Global Alignment Kernel*

Cuturi et al. [2007] proposed a kernel to handle the time series alignment problem. This DTW-inspired kernel aligns time series using the soft-minimum of all alignment costs in order to define a positive definite kernel. It has a quadratic complexity of $\mathcal{O}(n^2)$, which is similar to the DTW measure.

Definition 1.9 (Global Alignment Kernel)**.** *The Global Alignment Kernel (GAK) k_{GAK} between two time series $\boldsymbol{x}_i$ and $\boldsymbol{x}_\ell$ is defined as*

$$k_{GAK}(\boldsymbol{x}_i, \boldsymbol{x}_\ell) = \sum_{w \in \mathcal{W}} e^{-D_{\boldsymbol{x}_i, \boldsymbol{x}_\ell}(w)} \tag{1.16}$$

where we consider $\mathcal{W}$ as the set of all possible alignments, w as an alignment (of length L), and

$$D_{\boldsymbol{x}_i, \boldsymbol{x}_\ell}(w) = \sum_{j=1}^{L} \phi(x_{i,\pi_1(j)}, x_{\ell,\pi_2(j)}) \tag{1.17}$$

where ϕ is a positive-definite kernel.

As DTW, it is possible to use dynamic programming to compute the kernel value. GAK can handle both different lengths and multivariate time series. Cuturi [2011] later proposed a fast version of this kernel, that considers a smaller subset of admissible alignments. This fast version can be seen as adding global constraints, similar to Sakoe-Chiba band for DTW. Then Cuturi and Blondel [2017] build on GAK to propose soft-DTW for time series clustering and prediction, where they compute the soft-minimum of all alignment costs.

1.2.2 *Summary on (Dis)Similarity Measures*

Numerous similarity measures for time series exist, we detailed in this manuscript a small selection of them. Each one of them can be associated with a classifier. A comparison of their characteristics can be found in Table 1.1.

To evaluate the performance (last column of Table 1.1), we compute the one-sided Wilcoxon Signed Rank Test on the University of East Anglia & University of California Riverside (UEA / UCR) database (85 datasets). We use accuracies provided by Bagnall et al. [2017], on available similarity measures associated with the 1-NN classifier, as shown on Table 1.2. The one-sided Wilcoxon Signed Rank Test is a non-parametric statistical test that can be used to determine if a method is statistically better than another (*i.e.* p-value $< 5\%$). The number of stars for the performance is assigned based on the number of times the similarity measure is better than the other ones (associated with 1-NN). The number of stars ranges from 0 (the minimum), for measures that are never significantly better than the others (*i.e.* ED, DDTW), to 3 (the maximum), which corresponds to the best similarity measure (*i.e.* MSM).

1.3 *Feature-based Time Series Classifiers*

Feature-based methods extract features from time series before the classification step where a classifier can be used on the extracted features. Many different works have been proposed with different kind of features. In this section, we first start by introducing time series representations that are used in the literature. Then, we review the most relevant feature-based time series classifiers. Finally, we present some shapelet-based algorithms.

1.3.1 *Time Series Representations*

Time series classification can be performed on both raw and transformed time series. The most natural representation of time series is the *raw* representation, *i.e.* the ordered list of valued data points. However, there are many tasks for which a representation that highlights some specificities of time series is the most suitable one, *e.g.* frequency for speech recognition. Due to the wide range of possible applications for time series data, numerous representations have been proposed by the time series community from Fourier and Wavelet Transformations to SIFT-based Representation, through Piecewise Approximations. In the following, we focus on the most well-known time series representations.

Measure	Temporal Shift	Robust to Noise	Robust to Outliers	Metric	Computational Cost	# Hyper-parameters	Performance
$\mathcal{L}_p$ distances				✓	$\mathcal{O}(n)$	-	-
ED				✓	$\mathcal{O}(n)$	-	☆☆☆
DTW	✓				$\mathcal{O}(n^2)$	-	★☆☆
DDTW	✓				$\mathcal{O}(n^2)$	-	☆☆☆
EDR	✓	✓	✓		$\mathcal{O}(n^2)$	1	-
ERP	✓		✓	✓	$\mathcal{O}(n^2)$	1	★★☆
GAK	✓				$\mathcal{O}(n^2)$	1	-
LCSS	✓	✓	✓		$\mathcal{O}(n^2)$	1	★★☆
MSM	✓			✓	$\mathcal{O}(n^2)$	1	★★★
TWED	✓		✓	✓	$\mathcal{O}(n^2)$	2	★★☆
WDDTW	✓				$\mathcal{O}(n^2)$	2	★☆☆
WDTW	✓				$\mathcal{O}(n^2)$	2	★★☆

Table 1.1 – Comparison of various similarity measures

	ED	DTW	DDTW	ERP	LCSS	MSM	TWED	WDDTW	WDTW	# sign. better
ED	-	0.999	0.846	1.000	1.000	1.000	1.000	1.000	1.000	0
DTW	**0.001**	-	**0.046**	1.000	0.998	1.000	1.000	0.917	1.000	2
DDTW	0.153	0.954	-	1.000	0.999	1.000	1.000	1.000	1.000	0
ERP	**0.000**	**0.000**	**0.000**	-	0.976	1.000	0.841	0.259	0.849	3
LCSS	**0.000**	**0.002**	**0.001**	**0.023**	-	1.000	0.533	0.114	0.239	4
MSM	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	-	**0.000**	**0.000**	**0.000**	8
TWED	**0.000**	**0.000**	**0.000**	0.158	0.465	1.000	-	0.057	0.605	3
WDDTW	**0.000**	0.082	**0.000**	0.739	0.885	1.000	0.943	-	0.854	2
WDTW	**0.000**	**0.000**	**0.000**	0.150	0.760	1.000	0.393	0.145	-	3

Table 1.2 – Comparison of similarity measures (associated with 1-NN) using one-sided Wilcoxon Signed Rank Test p-values. The bold values indicate that the difference is significant, *e.g.* DTW-NN is significantly better than ED-NN since the p-value is equal to 0.001 < 5%. The last column provides the number of times a similarity measure is significantly better than another one, *e.g.* DTW is significantly better than 2 other similarity measures: ED and DDTW.

1.3.1.1 *Fourier and Wavelet Transformations*

Fourier transformation represents a time series as a sum of sinusoidal functions, whereas wavelet transformation approximates a time series by a set of orthonormal representations. Both Discrete Fourier Transform (DFT) and Discrete Wavelet Transform (DWT) reduce the dimensionality of time series.

Discrete Fourier Transform The basic idea of DFT is to represent the time series $\boldsymbol{x}_i$ as a linear combination of cosines and sines (with amplitudes a, b and phase w), where only the first few coefficients are kept.

$$x_{ij} = \sum_{k=1}^{n} (a_k \cos(2\pi w_k j) + b_k \cos(2\pi w_k j)) \tag{1.18}$$

$$DFT(\boldsymbol{x}_i) = < (a_0, b_0), (a_1, b_1), \ldots, (a_n, b_n) > \tag{1.19}$$

In order to find the best match of a time series, Agrawal et al. [1993] proposed the following process:

1. Obtain the DFT coefficients for every time series of $\mathcal{D}$,
2. Build the multidimensional index using the first few Fourier coefficients of every time series of $\mathcal{D}$,
3. Perform the similarity search.

The DFT most popular implementation is the Fast Fourier Transform (FFT) proposed by Cooley and Tukey [1965], and illustrated in Figure 1.6.

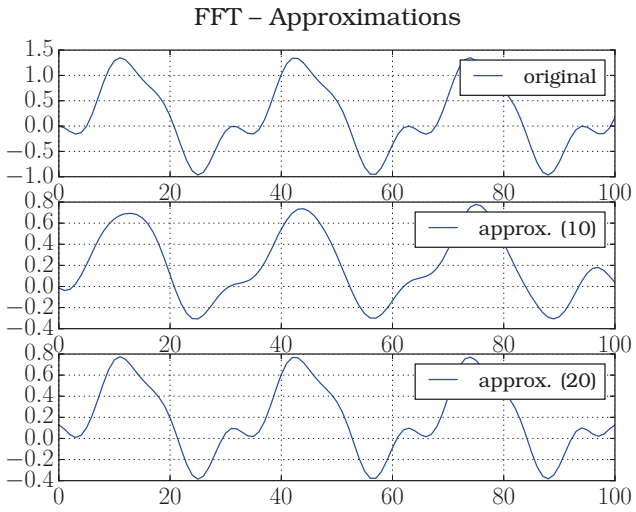


Figure 1.6 – Fast Fourier Transform. A Time Series and its Fourier Approximations using the first 10 & 20 coefficients (*respectively* 2^{nd} and 3^{rd} line).

Discrete Wavelet Transform Wavelet bases are able to represent both frequency and location informations inside a time series [Hastie et al., 2009, Chapter 5]. This is a key advantage compared to Fourier Transform which can only capture frequency information. Wavelets are generated by translations and dilations of a single scaling function $\psi(\boldsymbol{x})$ (*mother wavelet*), where translations can be defined by $\psi(z) \rightarrow \psi(z+1)$ and dilations by $\psi(z) \rightarrow \psi(2 \cdot z)$.

The first proposed DWT is the Haar wavelet, which mother wavelet is

$$\psi(z) = \begin{cases} 1 & \text{for } 0 \leq z < 1/2 \\ -1 & \text{for } 1/2 \leq z < 1 \\ 0 & \text{otherwise} \end{cases} \tag{1.20}$$

Let j be the dilation index, k the translation one, the general form is written as

$$\psi_k^j(z) = 2^{j/2}\, \psi(2^j z - k) \tag{1.21}$$

Haar wavelet corresponds to a sequence of continuous square-shaped functions that approximates the time series. Wavelet Transform has been widely used by the time series community. Chan and Fu [1999] proposed an efficient time series indexing method. Wu et al. [2000] go further and explore DFT and DWT-based similarity seach in time series datasets, where they show that both methods yield comparable results, even if DWT-based methods have lower complexity than DFT ones. Kahveci and Singh [2001]

use wavelets in order to find similar patterns in time series, thanks to the wavelet-based compression of the time series they were able to significantly reduce the computation time without performance loss.

1.3.1.2 *Piecewise Aproximations*

Numerous Piecewise Aproximations methods have been proposed in order to reduce the dimensionality of time series, we detail here the most relevant ones.

Piecewise Aggregate Aproximation Keogh and Pazzani [2000]; Yi and Faloutsos [2000] independantly introduced Piecewise Aggregate Aproximation (PAA), a dimensionality reduction technique. The idea is to divide a time series of length n into n' segments, then to compute the average of these segments. The main advantages of PAA are the interpretability and the simplicity of the method, PAA is a compression-based technique.

Adaptive Piecewise Constant Aproximation Chakrabarti et al. [2002] introduced an extension of PAA called Adaptive Piecewise Constant Aproximation (APCA). It is possible to observe that in some datasets, the time series information is concentrated on specific locations whereas the rest of the time series provides few information (*e.g.* regarding the separability of classes). In order to take this observation into account, APCA transforms the time series into a set of variable length segments instead of same length segments as PAA does. APCA has two parameters for each segment: its value (the mean of the segment) and its length. APCA is thus more flexible than PAA leading to a better approximation of time series.

Piecewise Linear Aproximation The basic idea of Piecewise Linear Aproximation (PLA, Cameron [1966]; Keogh and Pazzani [1998]) is to represent the time series as a sequence of linear segments. Given a time series $\boldsymbol{x}_i$, we seek for an approximate function: $\text{PLA}(\boldsymbol{x}_i) = \boldsymbol{a} \cdot \boldsymbol{z} + \boldsymbol{b}$, with $\boldsymbol{z} = [x_{i,z_1}, x_{i,z_2}, \dots x_{i,z_{k-1}}, x_{i,z_k}]^\top$ and followings the properties: (a) $z_1 = 1$ (b) $z_k = n$ and (c) $z_i < z_{i+1}$.

1.3.1.3 *Symbolic Aggregate approXimation*

Lin et al. [2003] introduced the Symbolic Aggregate approXimation (SAX) method that converts a time series into a symbolic form representation. The first step of the SAX algorithm is to normalize the data to have zero mean and standard deviation of one. Then to reduce the dimensionality of the time series, the PAA dimensionality reduction technique is applied. The next step is the discretization step: the symbols (corresponding to mean values

included into a specific interval) are selected such that they are equiprobable. The number of symbols is to be determined by the user.

1.3.1.4 *Symbolic-Fourier Approximation*

The Symbolic-Fourier Approximation (SFA) is a symbolic representation proposed by Schäfer and Högqvist [2012]. Its first step consists in approximating all the time series using DFT, then to determine multiple discretisations from all these DFT approximations using the Multiple Coefficient Binning (MCB) discretization technique. The MCB is a data adaptative quantization technique that computes intervals from training samples. In the second step, each DFT approximation is discretized (one by one) using the MCB discretization resulting in a SFA representation of the time series.

1.3.1.5 *SIFT-based Representation*

The Scale-Invariant Feature Transform was first introduced in the computer vision community on problems such as recognition tasks, trajectory tracking [Lowe, 2004]. The SIFT algorithm detects and describes local features in images. Local features should correspond to regions of interest and be scale and location invariant. Key location / key points are defined as extrema (minima and maxima) of the Difference of Gaussians (DoG) function, this step can be referred to as Scale-Invariant feature detection. More details on the full algorithm are available in [Lowe, 2004]. In the following, we present time series representation based on SIFT which are adapted for one-dimensional (1D) data.

Scale-Invariant Classifier with "R" metric (SIC-R) Xie and Beigi [2009] proposed an algorithm that detects keypoints based on DoG detector [Lowe, 2004], where each keypoints is represented by a scale-invariant feature descriptor. The general overwiew of the framework can be described by the following steps:

1. Keypoints detection
2. Construction of descriptors
3. Keypoints matching
4. Metric computation
5. Classification.

DTW Pruning Constraints based on Salient Feature Alignments Candan et al. [2012] introduced a new constraint to speed up DTW based on salient features: sDTW. These salient features are identified by a SIFT-like algorithm and then used to restrict the search space of possible warping

paths in DTW. The proposed approach can identify not only the center positions but also the sizes of the features, which is a key advantage in order to find a more precise alignment of time series. Experiments have shown that sDTW helps improving DTW accuracy.

1.3.2 *Codebook-based Representations*

We previously detailed some of the most relevant time series features representation. In the following we provide some keys on how to simply use these features in order to compare time series.

Codebook-based representations are highly related with the Bag-of-Words (BoW) model. We first start by introducing the BoW model then we detailed some time series classification algorithms using the BoW model (or very similar technique such as Vector Space Model (VSM)). Some previously introduced methods can be associated with a codebook-based representation, *e.g.* BoP and SAX-VSM that use SAX representation, later another example will be provided in Chapter 2 where we combine 1D SIFT descriptors with a codebook representation.

Bag-of-Words The BoW model consists in representing an instance using a histogram of word occurrences. Despite the fact that the temporal order of words is ignored, the BoW representation is able to capture high-level structural information using both local and global characteristics. Indeed, the overall representation is built by extracting the sub-sequences from the time series, local (*i.e.* peaks, valleys) and global (*i.e.* general shape) characteristics are thus taken into consideration. The BoW ability to quantize the words might lead to information loss, nonetheless it provides some robustness to noise. Many codebook-based representations use the k-means method in order to select k codewords. The selected codewords correspond to cluster centers, and each local feature is assigned to the nearest cluster center. The Bag-of-Words model has been widely used since it can lead to powerful and accurate methods.

1.3.2.1 *Bag-of-Patterns*

Lin et al. [2012] proposed the Bag-of-Patterns (BoP) approach which combines the Symbolic Aggregate approXimation (SAX) representation of the time series with the Bag-of-Words technique. In order to extract words from the transformed series, Lin et al. [2012] use a sliding window. Then, they build the 1-NN classifier using frequencies of words within the time series as features.

1.3.2.2 *SAX-VSM*

Senin and Malinchik [2013] proposed the SAX-VSM algorithm, which is based on two well-known techniques: SAX and VSM. SAX transforms raw time series into a symbolic representation, then VSM constructs a BoW representation from the symbols. SAX-VSM and BoP have many common characteristics, except that the Term Frequency - Inverse Document Frequency (TF-IDF) is used as a weighting scheme on the SAX-VSM representation (BoP uses no weighting scheme).

More precisely, the first step is to apply the SAX algorithm in order to transform raw time series to a list of SAX words. Time series are then treated as bag-of-SAX-words on which a TF-IDF normalization method is applied. The final step is to create class-characteristic weight vectors that are used for classification of new time series. To do so Senin and Malinchik [2013] look for characteristic subsequences that are representatives of a class (similarly as time series shapelets – introduced later in this chapter) that they use to build a single representation for each class (instead of one per training time series).

1.3.2.3 *Time Series Bag-of-Features*

Baydogan et al. [2013] introduced the Time Series Bag-of-Features (TSBF) algorithm. TSBF extracts time series segments of various lengths and positions and generates a codebook of those patterns that will be used to feed the classifier. TSBF can be outlined as follow:

1. Extract local features from each segment of the time-series,
2. Create a supervised codebook of the features using random forest,
3. Transform the codebook features collection into a global bag-of-features representation for each time series (using frequencies),
4. Classify the global bag-of-features representation using a random forest classifier.

TSBF leads to high performance compared to competitive baseline methods.

1.3.2.4 *Bag-of-Words based on Discrete Wavelet Coefficients*

Wang et al. [2013a] proposed an algorithm that segments time series into local patterns and measures their frequencies as classification features. Segments of pre-determined length are extracted along the time series using sliding window. The set of segments is then transformed to feature vectors using DWT. Finally the BoW representation of words frequencies is used to feed a Nearest-Neighbors classifier.

1.3.2.5 *Bag-of-SFA-Symbols*

The Symbolic-Fourier Approximation (SFA) is a symbolic representation used in Bag-of-SFA-Symbols (BOSS) which was introduced by Schäfer [2015b]. BOSS uses the sliding window technique to create SFA words representing the time series then generates a codebook representation from them. Sliding windows split the time series (of length n) in consecutive fixed-length windows (*i.e.* $n - w + 1$ windows of length w). For consistency, each window $\boldsymbol{s}$ is z-normalized ($\frac{\boldsymbol{s} - \text{mean}(\boldsymbol{s})}{\text{std}(\boldsymbol{s})}$) BOSS uses the 1-NN classification on SFA representation. This transformation aims at reducing the noise and at representing the time series as strings. BOSS is fast, robust to noise and obtains high accuracies score on several datasets.

Building on BOSS, Schäfer [2015a] proposed Bag-of-SFA-Symbols in Vector Space (BOSS VS) that can be seen as an enhancement of the BOSS method: BOSS-VS has significantly lower computation complexity. However, BOSS obtains higher performance than BOSS-VS, even if the difference is not significant.

1.3.3 *Shapelet-based Algorithms*

Time series shapelets were first introduced by Ye and Keogh [2009]. A shapelet is a subsequence extracted in a time series that enables one to differentiate the different classes. Ye and Keogh [2009] proposed a brute force algorithm to build a decision tree classifier where shapelets are chosen amongst all the possible candidates (*i.e.* the set of all continuous subsequence of the considered length). The chosen shapelets correspond to the candidates that are the most discriminative for the classes.

Definition 1.10 (Dissimilarity measure between a time series and a shapelet)**.** *The dissimilarity measure between a time series $\boldsymbol{x}_i$ (of length n) and a shapelet $\boldsymbol{s}_k$ (of length L (with $L < n$)) is defined as*

$$d(\boldsymbol{x}_i, \boldsymbol{s}_k) = \min_{j=1,\dots,J} \frac{1}{L} \sum_{\ell=1}^{L} (x_{i,j+\ell-1} - s_{k,\ell})^2 \tag{1.22}$$

with $J = n - L + 1$.

Advantages of shapelets include their interpretability as well as fast classification time. However the shapelet discovery step has a high complexity (*i.e.* is a time consuming process). The first improvement of the brute force algorithm was proposed by the same authors in [Ye and Keogh, 2011], where they introduced an early abandon pruning method. Since then, many works have been published on time series shapelets, we review the most relevant ones.

Shapelets-based methods are particularly useful when only a subpart of the time series is discriminative, as well as when some subparts of the time series are missing (*e.g.* objects outline where the object is partially broken). Shapelets-based methods also provide interpretability of the results since it is possible to extract the most discriminative shapelet(s). Methods that extract several shapelets use various discriminative subparts of time series thus generally lead to better performance.

1.3.3.1 *Fast Shapelets*

Rakthanmanon and Keogh [2013] proposed a fast version of the algorithm in order to reduce the complexity of the original algorithm from Ye and Keogh [2009]. The main idea for a fast shapelets discovery is to transform high-dimensional time series to a low-dimensional discrete representation. Indeed, it is more efficient to extract information from a smaller representation.

Time series are transformed into a SAX representation. For a given time series, multiple SAX words are generated using the sliding window technique, where each word corresponds to a symbolic representation of a subsequence. In order to compare two time series, Rakthanmanon and Keogh [2013] exploit the random projection idea by applying a random mask on the SAX representation. If two time series are similar, then there is a high probability of collisions between their SAX representations (a collision happens when two SAX representations are identical). After r random projections, it is possible to look for the best random projections candidates that maximize the separability of the different categories.

The shapelets candidates are then limited to the raw subsequences of time series corresponding to the best random projections in the SAX representation domain. Then, we apply the same process than the original algorithm on a reduced set of shapelet candidates. Fast shapelets algorithm is faster than the original shapelet algorithm, however its overall accuracy does not differ from the original algorithm.

1.3.3.2 *Shapelet Transform*

Lines et al. [2012] proposed to use shapelets as feature extractors in order to classify time series. The outline of the Shapelet Transform algorithm is the following:

1. Extract the K best shapelets
   (a) Measure quality for all possible shapelets candidates
   (b) Sort shapelets candidates by quality
   (c) Return top K shapelets candidates

2. Transform the data by computing the K dissimilarity measures between the time series and the set of selected shapelets. Each time series will be transformed to a set of K features (corresponding to the previously computed distance values)

$$\text{Transform: } N \times n \longrightarrow N \times K \tag{1.23}$$

3. Apply decision tree on shapelet transformed time series

The parameters of the model, which are the minimal and the maximal shapelet length as well as the number K of shapelets to be selected, are selected by a simple algorithm provided by Lines et al. [2012].

1.3.3.3 *Learning Shapelets*

Grabocka et al. [2014] propose to learn the shapelets instead of searching the best one(s) in a set of shapelet candidates, indeed one of the main drawback of the shapelet method from Ye and Keogh [2009] was the extensive search to find a discriminative shapelet. Grabocka's method offers a significant performance improvement compared to extensive search, moreover it is able to learn the best set of K shapelets. Learning Time series Shapelets (LTS) algorithm is based on a gradient descent method. In order to initialize the shapelets, subseries of length L are extracted from the training time series, then a K-means is performed on them. The shapelets are initialized to fit the K-means cluster centers, then the shapelets are learned iteratively during the learning step. The shapelets are updated by minimizing a classification loss function through the iterations of a gradient descent optimization process. The idea is to jointly learn both the logistic regression weights $\boldsymbol{w}$ and the shapelets $\mathcal{S}$ that minimize the classification objective function $\mathcal{F}$:

$$\arg\min_{\mathcal{S},\boldsymbol{w}} \mathcal{F}(\mathcal{S}, \boldsymbol{w}) = \arg\min_{\mathcal{S},\boldsymbol{w}} \sum_{i=1}^{N} \mathcal{L}(y_i, \boldsymbol{w}\boldsymbol{M}_i + w_0) \tag{1.24}$$

where $\mathcal{L}$ is the classification loss function and $\boldsymbol{M}$ is the matrix representing the distances between a dataset $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^{N}$ and a set of K shapelets $\mathcal{S} = \{\boldsymbol{s}_k\}_{k=1}^{K}$. The dissimilarity measure between the i-th time series $\boldsymbol{x}_i$ and the k-th shapelet $\boldsymbol{s}_k$ is written as

$$M_{i,k} = d(\boldsymbol{x}_i, \boldsymbol{s}_k) = \min_{j=1,\dots,J} \frac{1}{L} \sum_{\ell=1}^{L} (x_{i,j+\ell-1} - s_{k,\ell})^2 \tag{1.25}$$

Amongst all shapelet-based algorithms, LTS is one of the two most accurate and advanced one (with the Shapelet Transform algorithm). In Chapter 3, we build on Grabocka et al. [2014]'s work to provide an enhancement of the LTS method.

1.4 *Ensemble Classifiers for Time Series*

Ensemble classifiers main idea is to generate more precise and accurate classification systems using individual decisions combined together. Ensemble classifiers can be divided into several categories:

1. Using a single learning method with different subsets of training data (called *Bagging*, *e.g.* Random Forest)
2. Using a single learning method with different training parameters / initializations (*e.g.* initial weights of a Neural Network)
3. Using a set of different learning methods (called *Stacking*, *e.g.* PROP, COTE (defined below))

Ensemble classifiers lead to more accurate and more precise results than single classifier(s) alone. The combination of predictions using various classifiers will be able to get the advantages of the subset of classifiers that are the most well-suited for the considered dataset. This is really useful when no prior information are known about the dataset.

Amongst drawbacks, ensemble classifiers have higher computational time and space complexities as well as low interpretability. The time and space complexities grow with the number of classifiers. The total amount of space required by the ensemble corresponds to the sum of space required by each classifier of the ensemble (plus some extra space for the Decision Making Process (DMP)). However, time complexity can be reduced using parallel computing system to the time needed to process the longest time method (plus some extra time for the DMP). Finally, the more classifiers involve in the DMP, the less comprehensible the decision is. Thus, the interpretability decreases with the number of classifiers of the ensemble.

In the time series classification domain, only the stacking approach has been investigated, we thus detail two *stacking* ensemble classifiers for time series data that combine several classifiers.

1.4.1 *Proportional Elastic Ensemble*

Proportional Elastic Ensemble (PROP), which was introduced by Lines and Bagnall [2014], is a combination of nearest neighbour classifiers that use elastic baselines distance measures. PROP merges eleven elastic similarity measures: ED, DTW, DDTW, DTW-CV, DDTW-CV, WDTW, WDDTW, LCSS, ERP, TWED, MSM, where DTW-CV means DTW with Sakoe-Chiba band set through Cross-Validation.

Each classifier (a function associated with the 1-NN classifier) runs independently, then a weighted voting scheme is used for the prediction. Lines and Bagnall [2014] studied four voting schemes, we develop only the one corresponding to the PROP algorithm since it leads to the best performance.

The PROPortional voting scheme assigns the weight proportionally to the Cross-Validation accuracy on the training data. PROP significantly outperforms all standalone classifiers compared to in the paper.

1.4.2 *Collective Of Transformation-based Ensembles*

Bagnall et al. [2015] proposed the Collective Of Transformation-based Ensembles (COTE) for time series classification. Both PROP and COTE combine several standalone classifiers. Numerous time series classification algorithms transform the time series into new representations, which can lead to improvements in accuracy. COTE is partially based on transformations of time series into new domains (features representation), whereas PROP uses a set of elastic measures.

COTE is an ensemble classifier that contains transformations in time, frequency, change and shapelet domains from 35 standalone classifiers. Each classifier is associated to a weight during the classification step. The higher the accuracy on the training set, the larger the weight.

COTE combines:

- Four transformations, in frequency domain with a transformation of data using Power Spectrum and in change domain: three autocorrelation-based approaches. These transformations are associated with eight classifiers
  - k-NN
  - SVM with linear kernel
  - SVM with quadratic kernel
  - Naive Bayes
  - Bayesian network
  - C4.5 decision tree
  - Random Forest
  - Rotation Forest
- PROP Elastic Ensemble for the time domain (11 classifiers)

In COTE, weights are assigned proportionally to the cross-validation accuracy on the training data, as in PROP. COTE is significantly more accurate than competing baselines algorithms and is on average the most accurate algorithm (on UEA / UCR database).

1.5 *Summary on Time Series Classification Algorithms*

Many algorithms have been described, Table 1.3 summarizes the main characteristics of these time series classification algorithms. Performance was evaluated by computing the one-sided Wilcoxon Signed Rank Test on the UEA / UCR database (on 85 datasets) for each classifier, from classification rates provided by Bagnall et al. [2017]. The computational cost was

	Temporal Distorions	Robust to Noise	Robust to Outliers	# Params	Performance
BoP	✓	✓	✓	3	★☆☆
BOSS	✓	✓	✓	4	★★☆
COTE	✓	✓	✓	-	★★★
DTW-NN	✓			-	★☆☆
ED-NN	✓			-	☆☆☆
Fast Shapelets	✓			2	☆☆☆
GAK	✓	✓		2	-
LTS	✓	✓		3	★☆☆
PROP	✓	✓	✓	-	★★☆
SAX-VSM	✓	✓		3	☆☆☆
sDTW	✓	✓		-	-
Shapelets	✓	✓		2	★☆☆
ST	✓	✓		3	★★☆
TSBF	✓	✓	✓	2	★☆☆

Table 1.3 – Comparison of Time Series Classifiers

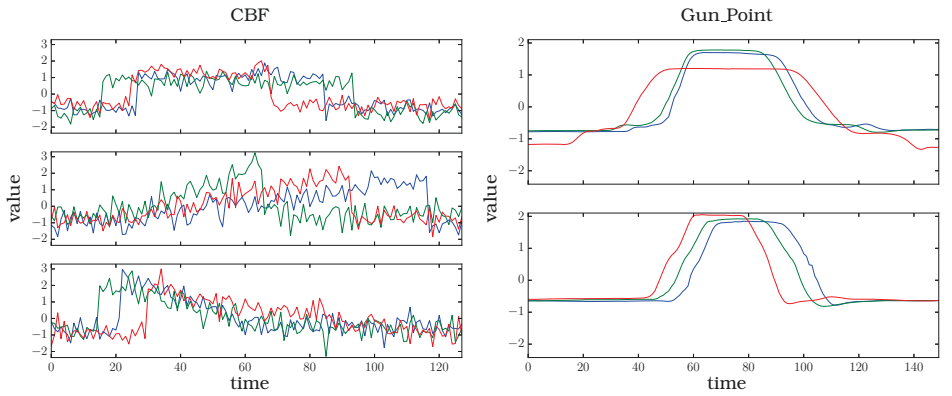
not evaluated since it is highly correlated with the implementation as well as the number of parameters.

COTE and PROP are considered to be the more powerful and accurate time series classification algorithms to this day. However since they correspond to sets of classifiers, they have a high computational cost. When no prior information is known, it might be a good choice to use COTE and PROP since they are able to adapt well on different characteristics (*e.g.* presence or absence of temporal distortions).

Standalone classifiers often offer good performance and might be considered as providing sufficient accuracy for some applications. An algorithm should be choosen carefully depending on:

1. Complexity of the algorithm
2. Constraint on computational space
3. Constraint on time
4. Dimensionality of the data (time series length and dataset size)
5. Easiness of the classification task
6. Expected level of accuracy
7. If interpretability of results is needed
8. Separability of time series using shapelets
9. Temporal distortion in the dataset

In the following, we will use UEA / UCR datasets to illustrate some characteristics and which time series classifier(s) is best suited for it. For very simple problems such as *Coffee*, a simple ED-NN will be more than satisfactory since it returns a classification score of 100%. For datasets where



(a) CBF (b) Gun_Point

Figure 1.7 – Illustration of temporal distortion: Three time series per class (one class per line) for *CBF* & *Gun_Point* datasets

temporal distortions occur *e.g. CBF* & *Gun_Point* (as shown on Figure 1.7), ED-NN will perform poorly compared to alignment method such as DTW (nearly 15% error rate for ED-NN, whereas DTW-NN has a 0.3% error rate). When orderig matters *e.g. 50words* where words were transformed in time series & *Two_Patterns* (as shown on Figure 1.8) where the different classes differs from the combinations of two patterns, the time series classifiers using a BoW representation will lead to less acurate results: BOSS and BoP *respectively* have 30% and 46% error rate whereas both PROP and COTE have less than 20% of errors. For datasets where only a small part of the time series matters such as *Gun_Point*, the shapelet based algorithms will lead to the highest classification rates (LTS has 100% good classification whereas ED-NN and DTW-NN have around 91% of accuracy).

There is no time series classifiers that outperforms all others, however a classifier choosen *w.r.t.* the dataset characteristics will often lead to higher accuracy than a randomly selected classifier.

1.6 *Model Selection and Evaluation*

1.6.1 *Metrics for Evaluating Time Series Classifiers Performance*

A wide range of metrics have been proposed, we only provide formulas of the evaluation metrics used in this dissertation.

Definition 1.11 (Accuracy)**.** *The accuracy, which measures the performance of a classifier, corresponds to the percentage of well classified data in the testing*

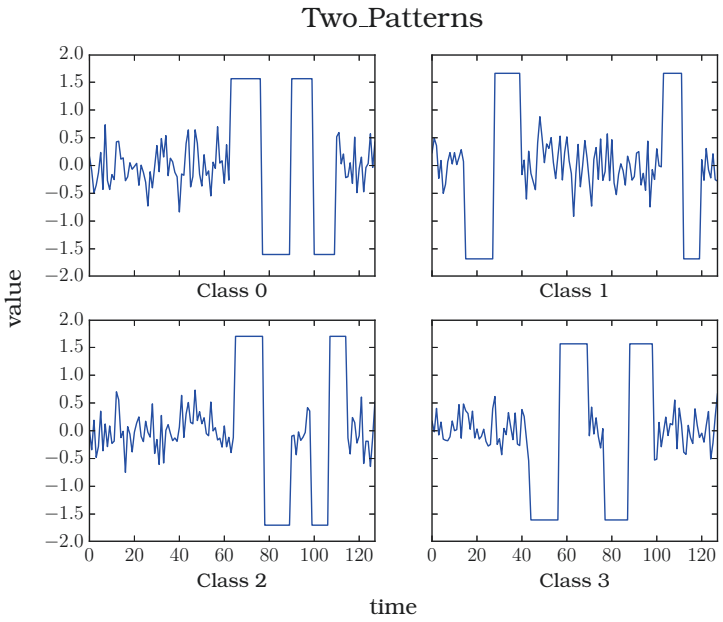


Figure 1.8 – Illustration of temporal order importance: *Two_Patterns* dataset (Only one time series per class for readibility)

dataset and is also called classification rate. For $\mathcal{D}_{test} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$*, where instances from the test set are only used to compute the accuracy and not the prediction:*

$$Accuracy(f) = \frac{|\{i|(f(\boldsymbol{x}_i) = y_i)\}_{i=1}^{N}|}{N} \tag{1.26}$$

Definition 1.12 (Error Rate)**.** *The error rate, which corresponds to the percentage of misclassified instances, can be expressed as a function of the accuracy:*

$$ER(f) = 1 - Accuracy(f) \tag{1.27}$$

1.6.2 *UEA / UCR Database*

The University of East Anglia & University of California Riverside (UEA / UCR) time series database [Bagnall et al., 2017; Chen et al., 2015] was built in order to improve the quality of papers using time series data. There are currently 85 datasets available in the UEA / UCR database, compared to 20 at the very beginning. Using all or a subset of them enables one to easily evaluate the performance of an algorithm compared to other methods. The UEA / UCR database does not aim at representing all real-life problems. It provides however a large panel of problems, from very small datasets to larger ones, see Table 1.4 for more details. UEA / UCR datasets represent

	From	Up to
# of Classes	2	60
Time Series Length	20	More than 2500
Training Set Size	Less than 20	8000
Test Set Size	20	8000

Table 1.4 – UEA / UCR Database in Numbers

a variety of problems from image outline classification to motion classification via sensor reading classification, some datasets are also simulated ones (*i.e.* they do not come from real data). Each dataset is divided into a train set and a test set. Splitting into train / test sets and classifying on the same set of data enables one to compare algorithm performances.

In the time series classification literature, the performance of algorithms is often measured on a fixed set of problems that uses separate train and test datasets, such as UEA / UCR database architecture. This database has been used in a large amount of research papers on time series analysis, many time series classification algorithms thus provide classification rates on UEA / UCR datasets. One may consider that a fixed database may not represents all real-life tasks one may encountered thus does not enable one to check if a classifier that works well on a specific database will generalize well on new tasks. That is one the reason why an open call for new time series datasets to fulfill the UCR / UEA database is still open.

The UEA / UCR database provides a large set of datasets with different characteristics. Some of these characteristics are detailed in the paper where the datasets were first introduced, however there is no paper that details characteristics for all the datasets. Datasets may be subject to temporal shifts, temporal distortions, noise, outliers, or sensitive to the order of different events, to abrupt changes, the amplitude taken by the time series values can be important, only a subpart of the time series can be relevant for the classification, etc. There is no classifier than can outperform all others on a large variety of problems with such different characteristics. Each classifier has advantages and drawbacks thus works better on dataset with specific characteristics, a detailed listing of time series dataset characteristics would enables one to better choose a classifier depending on the characteristics of the considered dataset.

Another drawback of the UEA / UCR database is the lack of very large datasets (in terms of number of time series or length). In real-life problems, it is often possible to acquired a large number of time series leading to large dataset (*e.g.* more than 10000 training time series). It has been shown than the performance depends on the train set size thanks to experiments on synthetic datasets: the larger the training set, the smaller the error rate (*e.g.* in Wang et al. [2013b]). Providing very large real-life datasets might

be difficult due to the lack of labels, however it is necessary as it can affect the performance of some methods. Moreover some methods might require more data for an optimal training and thus might be under-estimated on the UEA / UCR datasets.

In Chapter 4, we use two remote sensing time series datasets that contain many time series (500 time series per class) with few datapoints: the length of the time series is 24. These time series have a high intra-class variability. Moreover, time shifts and dilations occur in both datasets. These two remote sensing datasets can be used to enlarge the variety of problems contained in the UEA / UCR database.

1.7 *Other challenging problems related to Time Series Classification*

Time series classification brings up a lots of challenges due to the specificities of time series data, however there are related tasks using time series that are as much as challenging as TSC. In this section, we will briefly review time series challenges that will not be studied in this thesis.

Anomaly Detection In order to detect anomaly in time series, we have to find outlier points or abnormal subsequences relative to *standard* data, *i.e.* to find patterns that do not match with the *normal* behavior of time series. Anomaly detection in time series has many applications such as real-time monitoring of sensors (*e.g.* to check patients' health in hospital using ECG) or resource usage (*e.g.* to track unplanned electricity consumption in winter). Anomaly detection is thus an important topic that can prevent critical situations. Anomaly detection methods can either be supervised, semi-supervised or unsupervised. Supervised techniques can be seen as a special case of binary classification where the labeled data are either *normal* (0) or *abnormal* (1). In order to detect anomaly, a typical approach consists to generate a model *normal* behavior of time series then to look for the subseries / points that do not correspond to the model expectation. An overview of different anomaly detection methods can be found in Chandola et al. [2009]; Cheboli [2010].

Clustering Data clustering [Aghabozorgi et al., 2015; Everitt et al., 2009], which can be referred to as data partitioning, corresponds to unsupervised classification, it aims at dividing an unlabeled dataset into different homogeneous groups that share common charasteristics. Elements in the same cluster are more similar (*e.g.* according to a distance measure) to each other than those that belongs to other clusters. The definition of similarity and the number of clusters have a high influence on clustering results, that is

the reason why it should be chosen carefully. Cluster analysis can not be considered as an automatic task, since results of clustering are required to be interpreted, but more as a process extracting knowledge from data. This process can be based on a multi-objective optimization problem in order to obtain relevant and well-separated clusters, in that case the clustering function should both minimize the intra-class variability and maximize the inter-class variability. There are multiple ways and algorithms to separate data into clusters, the most famous one is a centroid-based method called k-means. Time series clustering can be easily applied using k-means on codebook representation for example.

Domain Adaptation Domain adaptation problems arise when datasets have different probability distributions. Instead of train and test sets, a source and a target domains are used ($\mathcal{D}_{\text{source}}$ and $\mathcal{D}_{\text{target}}$). The main purpose here is to transfer the knowledge learned from $\mathcal{D}_{\text{source}}$ to $\mathcal{D}_{\text{target}}$ the unknown domain. In classic supervised learning problems, it is assumed that instances of $\mathcal{D}_{\text{train}}$ and instances of $\mathcal{D}_{\text{test}}$ follow the same probability distribution $\mathcal{P}$. The goal is thus to learn a classification function f such that it maximizes the accuracy on new unseen data following $\mathcal{P}$. However for domain adaptation problems, instances of $\mathcal{D}_{\text{source}}$ (*respectively* instances of $\mathcal{D}_{\text{target}}$) follow the probability distribution $\mathcal{P}_S$ (*respectively* $\mathcal{P}_T$), where $\mathcal{P}_S$ and $\mathcal{P}_T$ are not equal. The purpose is thus to learn f such that it maximizes the accuracy on new unseen data following $\mathcal{P}_T$. f can be trained with instances following $\mathcal{P}_S$ only ; or from $\mathcal{P}_S$ and $\mathcal{P}_T$ (using labeled or unlabeled instances). Domain Adaptation methods can either be supervised, semi-supervised or unsupervised depending on the information available on the instances of each domain. Time series domain adaptation problems have been studied in Petitjean et al. [2011] where domain adaptation problem stems from the fact that the time series are acquired at two different years.

Summary Time series data comes from many domains and have many applications. Many tasks can be solved using time series data such as anomaly detection, clustering tasks, domain adaptation problems, early classification, forecasting as well as classification. Time series data thus provide challenging problems to solve.

2

TIME SERIES CLASSIFICATION BASED ON LOCAL FEATURES REPRESENTATION

Contents

As seen in Chapter 1, numerous approaches for time series classification are based on feature extraction associated with a Bag-of-Words (BoW) representation of these features. This can be easily explained by the fact that there exists many possibilities for feature extraction in time series, where some are based on symbolic representation (*e.g.* SAX) while others rely on the neighborhood of keypoints (*e.g.* Scale-Invariant Feature Transform (SIFT)). SIFT feature extraction framework has led to widely used descriptors thanks to its efficiency and effectiveness in image classification. Moreover, SIFT features are easily adaptable from 2D (for image) to 1D features (for time series), making SIFT a reasonable choice for feature representation of time series. Finally, the BoW representation has proved its robustness to noise, its easiness of use as well as its capability to help building highly accurate methods thanks to its ability to capture both local and global information.

In this chapter, we propose a new time series classification algorithm, that builds on two well-known and powerful methods in computer vision: local features which are extracted from time series using a SIFT-based approach, and a global representation of time series using these features which is produced using the BoW technique.

2.1 *Related Work*

This section first introduces the SIFT feature extraction framework proposed by [Lowe, 1999, 2004], then introduces key notions related to the BoW technique.

2.1.1 *Scale-Invariant Feature Transform*

SIFT was first introduced by [Lowe, 1999, 2004] in the computer vision community for image matching task. In order to be effective, the feature transform should follow some useful characteristics such as: (a) Scale Invariance, (b) Rotation Invariance, (c) Illumination Invariance, (d) Viewpoint Invariance. SIFT identify a large number of interest points in both locations and scales *e.g.* Figure 2.1. These keypoints are both distinctive and invariant and can be found even if the point of view of the object changes. The adaptation of SIFT framework from 2D to 1D requires some simplification in particular there is no need for illumination and rotation invariances for time series data. 1D SIFT only has to be scale and location invariant.

The overall 2D SIFT feature extraction framework can be divided into several steps:

1. Scale space computation
2. Location of keypoints

Figure 2.1 – SIFT – Keypoints Detection. Top image: 1128 detected keypoints on a 512×512 pixels image. Bottom image: 519 detected keypoints on a 780×350 pixels image (Generated using OpenSIFT Library).

Figure 2.2 – Match between two images, 161 matches (OpenSIFT Library)

3. Orientation assignment (not relevant for time series data)
4. Construction of the keypoint descriptors

Scale Space Computation The first step is to compute the scale space. To efficiently find the key locations in images, Lowe [1999] proposes to use the Difference-of-Gaussians (DoG) function to detect scale-space extrema in the scale space. This can be easily done by computing a pyramid of smoothed, convoluted images.

Let $L(x, y, \sigma)$ be the convolution ($*$) of an image $I(x, y)$ with a Gaussian function $G(x, y, \sigma)$ of width σ

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{2.1}$$

where $G(x, y, \sigma)$ is equal to

$$G(x, y, \sigma) = \frac{1}{2\pi\ \sigma^2}\ e^{-(x^2+y^2)/2\sigma^2}. \tag{2.2}$$

The DoG function is defined as

$$D(x, y, \sigma) = (G(x, y, k_{\text{sc}}\sigma) - G(x, y, \sigma)) * I(x, y) \tag{2.3}$$
$$= L(x, y, k_{\text{sc}}\sigma) - L(x, y, \sigma), \tag{2.4}$$

where k_{sc} is a parameter of the method that controls the scale ratio between two consecutive scales. The DoG function is particularly efficient to compute: an image substraction is applied on convoluted images. Moreover, the detected extrema have stable location accross images.

Location of Keypoints Keypoints correspond to DoG extrema, selection of keypoints based on the extrema of a scale-space representation is a reliable method to detect events that persist over large changes in scale [Mikolajczyk, 2002]. In order to locate them on the scale space, each pixel is compared with its neighbors. For images, each pixel has 8 neighbors on the same level and 9 on the above and below levels. If a pixel is a maximum or a minimum amongst all its neighbors then it is selected as a key location. Since most of the pixels are eliminated within a few comparisons, the detection of keypoints has a small cost. Keypoints detection is illustrated in Figure 2.1, where each arrow represents a keypoint and its main orientation.

Construction of the Keypoint Descriptors Keypoint descriptors correspond to local gradients. They are measured at the selected scale around each keypoint. Their construction ensures stable location, scale and orientation for each keypoints, as well as robustness against small local shifts. Figure 2.3 shows a 2×2 descriptors computed from a 8×8 sample array where the circle represents a Gaussian filter and where the length of

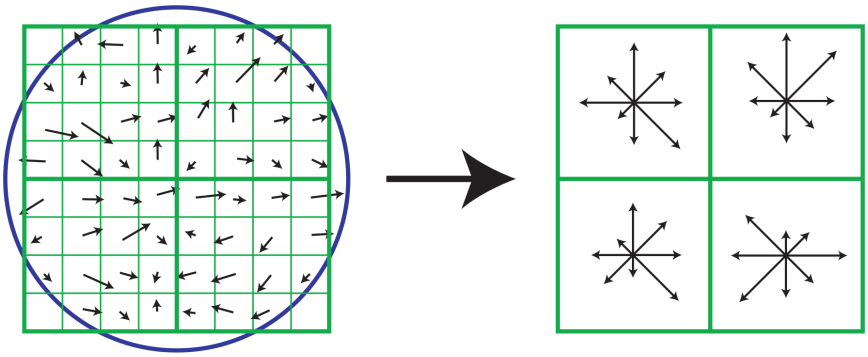


Figure 2.3 – SIFT – Keypoint Description [Lowe, 2004]

each arrow corresponds to the sum of the gradient magnitudes in the region around. For efficiency, Lowe [1999] proposes to use precomputed gradients from the pyramid that were used for key locations detection. Neighbors are separated into different regions then each region are represented by a histogram of n_b bins of size a. Keypoint descriptors are then used to generate codewords that will form a codebook. The codewords can be seen as approximations of the keypoint descriptors.

Dense Extraction Dense SIFT has been proposed as an extension to the SIFT framework. The idea is to create a codebook which is not based on key location but based on dense sampling. Dense sampling extracts *keypoints* at regular positions and scales in both location and scale instead of scale space extrema. Thus dense sampling can be seen as applying a regular grid on an image and in our case repeating this process over multiple scale. Jurie and Triggs [2005]; Wang et al. [2009] showed that dense representation consistently outperforms equivalent keypoint representation. Liu et al. [2008] were the first to propose a Dense SIFT version for image alignment and registration, later Vedaldi and Fulkerson [2010] proposed a fast algorithm for the computation of a dense set of SIFT descriptors.

Dense sampling outperforms equivalent keypoint representation, however it also produces a very large number of features, which slows down the overall computation since features need to be computed and compared.

2.1.2 *Bag-of-Words*

The Bag-of-Words model consists in representing documents using a histogram of word occurrences. It is a very common technique in text mining, information retrieval and content-based image retrieval because of its simplicity and performance. The latter can be explained by the fact that the BoW representation is able to capture both local and global structure sim-

ilarity information, despite the fact that the location of words is ignored. Another advantage of the BoW representation is its robustness to noise. However since it quantizes words, it might lead to information loss. For large images, it is often more appropriate to consider the similarity between data based on higher-level structures such as histogram or feature-based representations instead of raw data. For images, classifiers using BoW representation are based on the following operations: first converting images into words, then computing a histogram of words occurrences and finally building a classifier upon this histogram representation. Standard techniques such as Random Forests, Support Vector Machines (SVM) or k-Nearest-Neighbors (k-NN) can be used for the classification step.

BoW normalization Dense sampling on multiple Gaussian-filtered data provides considerable information to process. It also tends to generate words with little informative power, as stop words do in text mining applications. In order to reduce the impact of those words, there exists several normalization schemes for BoW, such as Signed Square Root normalization (SSR) and Term Frequency - Inverse Document Frequency normalization (TF-IDF). In order to obtain the SSR-normalized representation, the sign square root function is applied to each element z of an unnormalized BoW representation ($\mathrm{sign}(z) \cdot \sqrt{|z|}$). TF-IDF is a weighting method that measures how important is a term comparatively to the full term collection. TF-IDF normalization can be divided into two steps: first increase the weight term proportionally to the frequency of the term in the document, then weight down the frequent terms and scale up the rare ones from the full term collection. Both SSR and TF-IDF normalizations are commonly used in image retrieval and classification based on histograms such as in [Jégou and Chum, 2012; Jégou et al., 2010; Perronnin et al., 2010; Sivic and Zisserman, 2003].

Jégou et al. [2010] and Perronnin et al. [2010] showed that reducing the influence of frequent codewords before $\mathcal{L}_2$ normalization could be profitable. They apply a power $\alpha \in [0, 1]$ on their global representation. SSR normalization corresponds to the case where $\alpha = 0.5$. TF-IDF normalization also tends to lower the influence of frequent codewords. To do so, document frequency of words is computed as the number of training time series in which the word occurs. BoW are then updated by diving each component by its associated document frequency. In the following, SSR and TF-IDF are both applied before $\mathcal{L}_2$ normalization.

2.2 (Dense) Bag-of-Temporal-SIFT-Words Algorithm

In the following, Dense Bag-of-Temporal-SIFT-Words (D-BoTSW) corresponds to the dense keypoints extraction, whereas BoTSW corresponds to the scale space extrema detection. D-BoTSW can be seen as the extended and improved version of BoTSW.

Both methods are based on three main steps: (a) extraction of keypoints in time series, (b) description of these keypoints through gradient magnitude at a specific scale, (c) BoW representation of time series, where words correspond to quantized version of the description of keypoints. These steps are depicted in Figure 2.6 (dense version) and detailed below.

The Bag-of-Patterns [Lin et al., 2012] and the BOSS [Schäfer, 2015b] algorithms are two works close to D-BoTSW. Indeed, they both extract features that are used in a histogram representation. The main difference with D-BoTSW is that the feature are discrete for both BoP and BOSS, whereas descriptive features in D-BoTSW are real-valued.

2.2.1 Keypoints extraction in time series

The first step of our method consists in extracting keypoints in time series. Two approaches are described here: the first one is based on scale-space extrema detection (Bailly et al. [2015]) and the second one proposes a dense extraction scheme (Bailly et al. [2016a,b]).

2.2.1.1 Scale-space extrema detection

The 2D SIFT framework is transformed to fit 1D time series data. Keypoints in time series can be detected as local extrema in terms of both scale and (temporal) location. These keypoints are important sub-part of time series, since they correspond to extrema at a specific scale. They are often detected near time series extrema, as shown on Figure 2.4.

The scale-space extrema are identified using a DoG function, and form a list of scale-invariant keypoints. Let $L(t, \sigma)$ be the convolution ($*$) of a time series $S(t)$ with a Gaussian function $G(t, \sigma)$ of width σ:

$$L(t, \sigma) = S(t) * G(t, \sigma) \tag{2.5}$$

where $G(t, \sigma)$ is defined as

$$G(t, \sigma) = \frac{1}{\sqrt{2\pi}\ \sigma}\ e^{-t^2/2\sigma^2}. \tag{2.6}$$

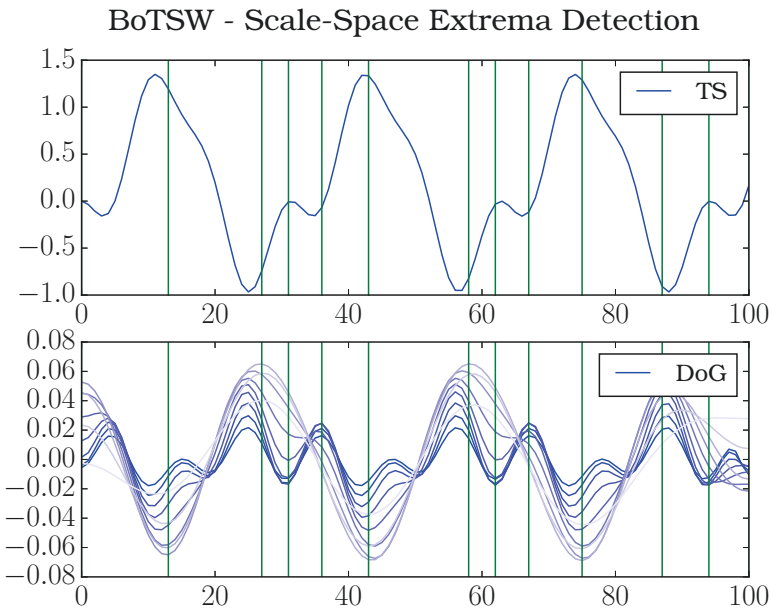


Figure 2.4 – BoTSW – Scale-Space Extrema Detection: A Time Series (top line) and its DoG representation (bottom line). The vertical lines represent the detected keypoints.

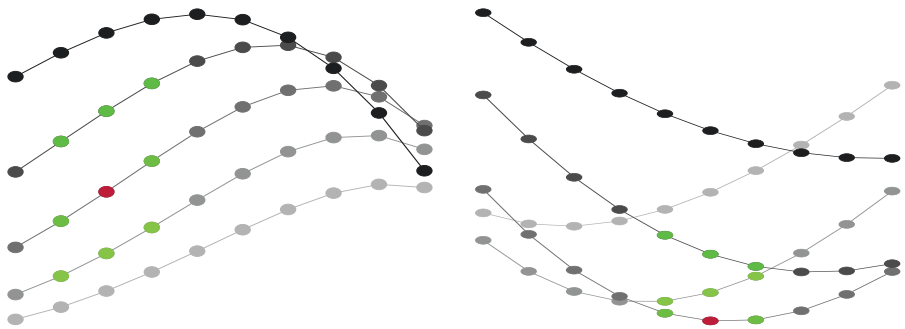


Figure 2.5 – Detection of extrema using Difference-of-Gaussians function. On the left, the considered point (in red) is not an extrema compared to its neighbors (in green). On the right, the considered point is the minima amongst its neighborhood, it is thus a selected keypoint. The different scales are represented by the grey gradient.

Lowe [1999] proposes the Difference-of-Gaussians (DoG) function to detect scale-space extrema in images. Adapted to time series, a DoG function is obtained by subtracting two time series filtered at consecutive scales:

$$D(t, \sigma) = L(t, k_{\text{sc}}\sigma) - L(t, \sigma), \tag{2.7}$$

where k_{sc} is a parameter of the method that controls the scale ratio between two consecutive scales.

Keypoints are then detected at time index t in scale j if they correspond to extrema of $D(t, {k_{\text{sc}}}^j\sigma_0)$ in both time and scale, where σ_0 is the bandwidth of the Gaussian corresponding to the reference scale. At a given scale, each point has two neighbors: one at the previous and one at the following time instant. Points also have six more neighbors: three one scale up and three others one scale down (at the previous, same and next time instants), leading to a total of eight neighbors, as shown in Figure 2.5. If a point is higher (or lower) than all of its neighbors, it is considered as an extremum in the scale-space domain and hence a keypoint. Figure 2.4 shows an example of the scale-space extrema detection, where keypoints are detected near local extrema.

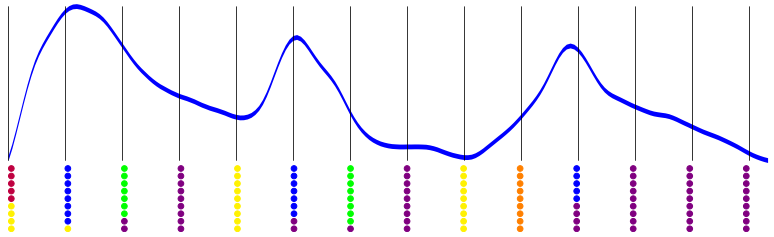
2.2.1.2 *Dense extraction*

Previous works have shown that better classification could be achieved by using densely extracted local features [Jurie and Triggs, 2005; Wang et al., 2009] at regular step. In this section, we present the adaptation of this setup to our BoTSW scheme. Keypoints selected with dense extraction no longer correspond to extrema but are rather systematically extracted at all scales every τ_{step} time steps on Gaussian-filtered time series $L(\cdot, {k_{\text{sc}}}^j\sigma_0)$.

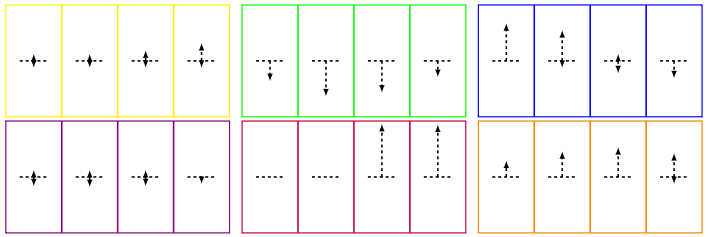
Unlike scale-space extrema detection, regular sampling guarantees a minimal amount of keypoints per time series. This is especially crucial for smooth time series from which very few keypoints are detected when using scale-space extrema detection. In all cases, description of these keypoints covers the description of scale-space extrema if τ_{step} is small enough, which leads to a more robust representation. A dense extraction scheme is represented in Figure 2.6.

2.2.2 *Description of the extracted keypoints*

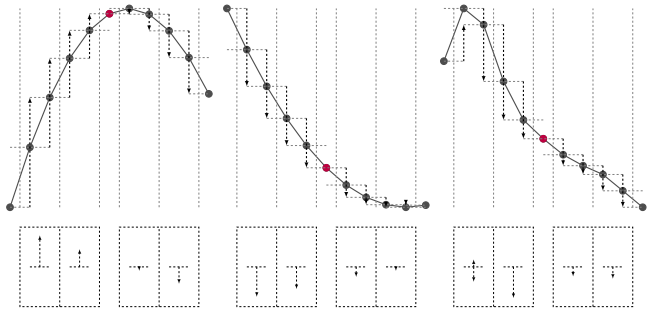
Next step in our process is the description of keypoints. A keypoint at time index t and scale j is described by gradient magnitudes of $L(\cdot, {k_{\text{sc}}}^j\sigma_0)$ around t. To do so, n_b blocks of size a are selected around the keypoint. Gradients are computed at each point of each block and weighted using a Gaussian window of standard deviation $\frac{a \times n_b}{2}$ so that points that are farther in time from the detected keypoint have lower influence. Then, each block



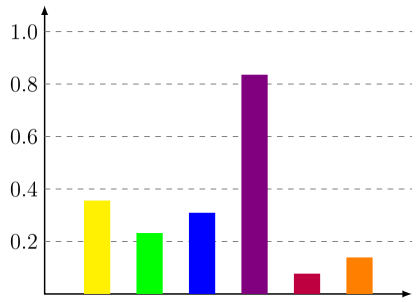
(a) Dense extraction ($\tau_{\text{step}} = 15, 9$ scales)



(b) k-means generated codebook ($k = 6$)



(c) Keypoint description ($n_b = 4, a = 2$)



(d) Resulting k-dimensional histogram

Figure 2.6 – (D-)BoTSW Approach Overview: (a) A time series and its dense-extracted keypoints. (b) Codewords obtained *via* k-means, the color is associated with the dots under each keypoint in (a). (c) Keypoint description is based on the time series filtered at the scale at which the keypoint is extracted. Descriptors are quantized into words. (d) Histograms of word occurrences are given to a classifier (linear SVM) that learns boundaries between classes.

is described by two values: the sum of positive gradients and the sum of negative gradients. Resulting feature vector is hence of dimension $2 \times n_b$.

2.2.3 *(Dense) Bag-of-Temporal-SIFT-Words for Time Series Classification*

Training features are used to learn a codebook of k words, where we apply a k-means clustering on all the features extracted from the whole set of training time series. Words represent different local behaviors in time series. Then, for a given time series, each feature vector is assigned the closest word in the codebook. The number of occurrences of each word in a time series is computed. (D-)BoTSW representation of a time series is the normalized histogram of word occurrences.

2.2.3.1 *Bag-of-Words normalization*

It has been shown that Bag-of-Words normalizations on dense sampling lead to higher accuracy of the model. Indeed the normalizations reduce the impact of words with little informative power, leading to more stable representation. We show in the experimental part that using BoW normalizations helps improving the accuracy of D-BoTSW. The normalized histograms finally feed a classifier (Linear SVM) that learns how to discriminate classes from this BoW representation.

2.2.3.2 *Complexity study*

The process of computing (D-)BoTSW representation for a time series has linear time complexity in the number of features extracted. When using scale space extrema detection (BoTSW), this number mainly depends on the data. For some datasets we might detect very few features, whereas for other datasets we extract much more features. It is thus complicated to evaluate the complexity of BoTSW. When using dense extraction (D-BoTSW), this number depends on the length of the time series. For a time series of length n, features will be computed at $\lfloor n/\tau_{\text{step}} \rfloor$ different time instants. At each time instant, features will be computed at all scales and the number of scales is $\left\lfloor \frac{\log(n/8\sigma_0)}{\log k_{\text{sc}}} \right\rfloor$. Finally, time complexity of computing D-BoTSW for a time series of length n is in $O(n \log n)$.

2.2.3.3 *Pros and cons of (D-)BoTSW*

(D-)BoTSW is based on two powerful techniques which are the SIFT features extraction framework and the Bag-of-Words representation, it thus

inherits advantages and drawbacks from both SIFT features and BoW representation.

SIFT framework is effective, efficient and provides features that are robust and invariant. Yet for smoothed data with the scale space extrema detection, few key locations will be detected leading to poorly represented data. Dense sampling solves this problem by extracting keypoints at regular locations, which guarantees a minimal amount of keypoints. However since the number of keypoints grows with the time series length and the time step, dense sampling can lead to a very large number of features for large datasets, which requires larger storage space, and which slows down the time computation for the codewords generation step.

The Bag-of-Words representation has many advantages such as its robustness to noise and its ability to handle temporal shifts. Nevertheless, the BoW representation causes information loss during the quantization step. In most cases, this information loss will have no impact, however in datasets that contain very similar classes the information loss might lead to similar BoW representation for distinct classes (thus lead to poor performance). Another facet to take into account is that BoW representation ignores the temporal order of words. For datasets where ordering of events matter, it is adviced not to use a time series classification algorithm using a BoW representation such as (D-)BoTSW. In other to tackle this *ordering* issue, we [Tavenard et al., 2017] use temporal kernels between feature sets ; this paper can be seen as an extension of D-BoTSW but will not be detailed in this manuscript.

D-BoTSW longest step corresponds to the k-means learning process and the fitting of the linear SVM classifier, since dense extraction, features description and transformation to BoW representation can be done efficiently ; D-BoTSW is thus able to fastly classify new data.

Finally, it is possible to visualized the model generated by D-BoTSW. This model consists of the descriptive features and the histogram representation. They can be used together in order to see the most important features for each class and thus offer some interpretability of the results. An illustration of this *interpretability* is provided in Chapter 4.

2.3 *Experiments on UCR/UEA datasets*

In this section, we investigate the impact of both dense extraction of keypoints and normalizations of the Bag-of-Words on classification performance. We then compare our results to the ones obtained by 9 relevant baselines.

Experiments are conducted on the 85 currently available datasets from the UEA / UCR repository [Bagnall et al., 2017].

For the sake of reproducibility, C++ source code used for (D-)BoTSW in these experiments is made available for download [1], and all raw numbers are available in Appendix A. To provide illustrative timings for our method (D-BoTSW), we ran it on a personal computer (laptop with 2.70GHz CPU and 16 GB RAM), for a given set of parameters, using dataset *Cricket_X* [Bagnall et al., 2017] that is made of 390 training time series and 390 test ones. Each time series in the dataset is of length 300. Extraction and description of dense keypoints take around 2 seconds for all time series in the dataset. Then, 75 seconds are necessary to learn a k-means and fit a linear SVM classifier using training data only. Finally, classification of all D-BoTSW corresponding to test time series takes less than 1 second. Note that the time computation could be reduced by sub-sampling the amount of features to learn the codewords in the k-means.

2.3.1 *Experimental setup*

Parameters a, n_b, k and C_{SVM} of (D-)BoTSW are learned by Cross-Validation (CV), whereas we set $\sigma_0 = 1.6$ and $k_{\text{sc}} = 2^{1/3}$, as these values have shown to produce stable results [Lowe, 2004]. Parameters a, n_b, k and C_{SVM} vary inside the following sets: $\{4, 8\}$, $\{4, 8, 12, 16, 20\}$, $\left\{2^i, \forall i \in \{5..10\}\right\}$ and $\{1, 10, 100\}$ respectively. A linear SVM is used to classify time series represented as (D-)BoTSW. For our approach, the best sets (in terms of accuracy) of $(a, n_b, k, C_{\text{SVM}})$ parameters are selected by performing cross-validation on the training set. Due to the heterogeneity of the datasets, leave-one-out cross-validation is performed on datasets where the training set contains less than 300 time series, and 10-fold cross-validation is used otherwise. These best sets of parameters are then used to build the classifier on the training set and evaluate it on the test set. For datasets with little training data, it is likely that several sets of parameters yield best performance during the cross-validation process. For example, when using *DiatomSizeReduction* dataset, BoTSW has 150 out of 180 parameter sets yielding best performance, while there are 42 such sets for D-BoTSW with SSR normalization. In both cases, the number of *best* parameter sets is too high to allow a fair parameter selection. When this happens, we keep all parameter sets with best performance at training and perform a majority voting between their outputs at test time.

Parameters a and n_b both influence the descriptions of the keypoints; different datasets will required different settings of these parameters thus their optimal values should be chosen carefully for each dataset in order the description of keypoints to fit the shape of the data. If the data contains sharp peaks, the size of the neighborhood on which features are computed (equal to $a \times n_b$) should be small. On the contrary, if it contains smooth

1. http://github.com/a-bailly/dbotsw

peaks, descriptions should take more points into account. The number k of codewords needs to be large enough to represent precisely the different features. However, it needs to be small enough in order to avoid overfitting (especially for datasets with only few training time series). We consequently allow a large range of values for k to better take into account the heterogeneity of the UEA / UCR database.

For all experiments with dense extraction, we set $\tau_{\text{step}} = 1$, and we extract keypoints at all scales. Using such a value for τ_{step} enables one to have a sufficient number of keypoints even for small time series, and guarantees that keypoint neighborhoods overlap so that all subparts of the time series are described.

2.3.2 *Comparison of Dense Extraction and Scale-Space Extrema Detection*

Figure 2.7 shows a pairwise comparison of error rates between BoTSW and its dense counterpart D-BoTSW for all datasets in the UEA / UCR repository. A point on the diagonal means that methods obtain equal error rates (●). A point above the diagonal illustrates a dataset for which D-BoTSW outperforms BoTSW (▲), whereas a point below the diagonal illustrates a dataset for which D-BoTSW is outperformed by BoTSW (▼). The further a point is from the diagonal, the larger the difference is. Win/Tie/Lose scores is given in the bottom-right corner of the figure. D-BoTSW reaches better performance than BoTSW on 66 datasets, equivalent performance on 6 datasets and worse on 13 datasets. One-sided Wilcoxon signed rank test is also used to compare the methods, if the p-value is less than the 5% significance level, one of the method is considered as significantly better than the other one, *e.g.* Figure 2.7 shows that D-BOTSW ($\mathcal{L}_2$) is significantly better than BoTSW ($\mathcal{L}_2$) ($p < 5\%$). The dense sampling version, D-BoTSW, improves classification rate on a large majority of datasets compared to the scale space extrema detection version. In the following, we show how to further improve these results thanks to BoW normalization.

Experiments have shown that scale space extrema detection did not always provide keypoints on time series. It happens in *Adiac* dataset where BoTSW has an error rate of 51.9% and it drops down to a 25.1% error rate using the dense version D-BoTSW thanks to additional keypoints.

2.3.3 *Impact on Bag-of-Words normalization*

The BoW representation was $\mathcal{L}_2$-normalized for BoTSW, we keep this normalization for D-BoTSW. Since Jégou et al. [2010] and Perronnin et al. [2010] showed that reducing the influence of frequent codewords before $\mathcal{L}_2$

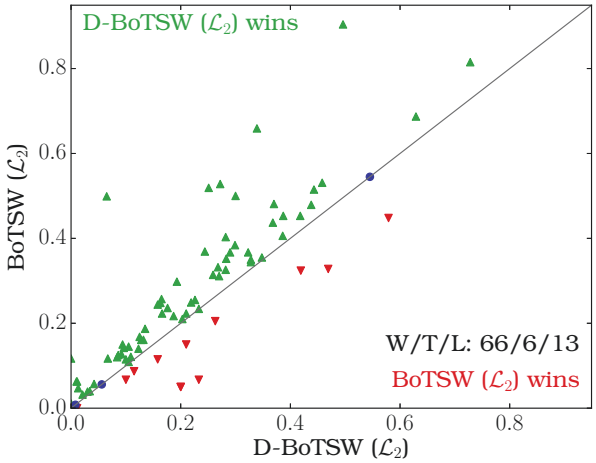


Figure 2.7 – Error rates of BoTSW compared to D-BoTSW.

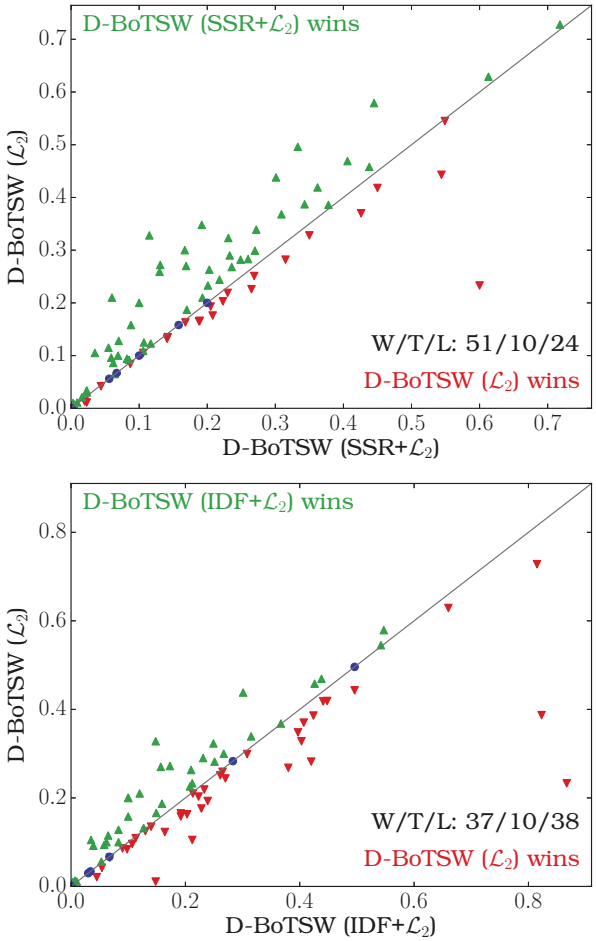
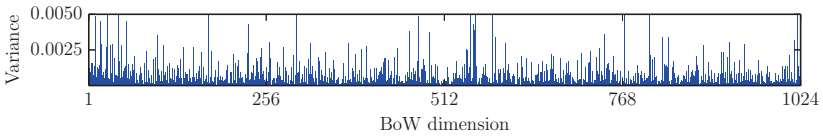
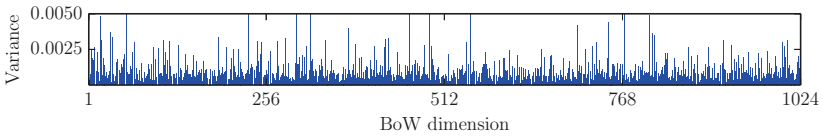


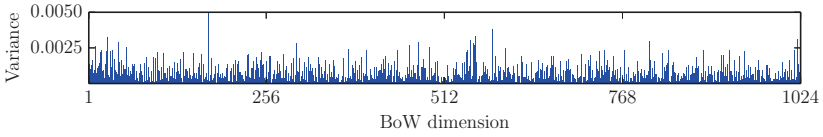
Figure 2.8 – Error rates of D-BoTSW with and without normalization.



(a) $\mathcal{L}_2$ normalized D-BoTSW



(b) IDF+$\mathcal{L}_2$ normalized D-BoTSW



(c) SSR+$\mathcal{L}_2$ normalized D-BoTSW

Figure 2.9 – Per-dimension energy of D-BoTSW vectors extracted from dataset *ShapesAll*. The same codebook is used for all normalization schemes so that dimensions are comparable across all three sub-figures.

normalization could be profitable, we study the impact of both the Signed Square Root (SSR) and the Inverse Document Frequency (IDF) normalizations prior the $\mathcal{L}_2$ normalization for our codebook.

As can be seen in Figure 2.8, the SSR normalization improves classification performance, the IDF however does not improve it. Lowering the influence of largely-represented codewords using SSR normalization hence leads to more accurate classification with D-BoTSW. SSR normalization significantly improves classification accuracy. This is backed by Figure 2.9, which shows that SSR normalization tends to spread energy across BoW dimensions, leading to a more balanced representation than other two normalization schemes.

2.3.4 *Empirical comparison with State-of-the-Art techniques*

In the following, we will consider a SSR-normalized D-BoTSW, since this setup is the one providing the best classification performance. We now compare our D-BoTSW to the most popular state-of-the-art methods for time series classification. All time series classification algorithms used here are detailed in Chapter 1. The UEA / UCR repository provides error rates for the 85 datasets with Euclidean distance 1-NN (ED-NN) and Dynamic Time Warping 1-NN (DTW-NN) [Ratanamahatana and Keogh, 2004]. Numerical performance values on UEA / UCR database are available in Appendix A. In order to check if a method is significantly better than another, we provide

D-BoTSW	is significantly better than	BoP	($p = 0.000$)
D-BoTSW	is significantly better than	DTW-NN	($p = 0.000$)
D-BoTSW	is significantly better than	ED-NN	($p = 0.000$)
D-BoTSW	is significantly better than	LTS	($p = 0.000$)
D-BoTSW	is significantly better than	SAXVSM	($p = 0.000$)
D-BoTSW	is significantly better than	TSBF	($p = 0.005$)
D-BoTSW	is not significantly better than	PROP	($p = 0.218$)
D-BoTSW	is not significantly better than	SMTS	($p = 0.173$)
BOSS	is not significantly better than	D-BoTSW	($p = 0.139$)
COTE	is significantly better than	D-BoTSW	($p = 0.000$)

Table 2.1 – D-BoTSW – One sided Wilcoxon Test p-values. If the p-value is less than the 5% significance level, the method is considered significantly better than the one it is compared to.

the p-values of the one-sided Wilcoxon signed rank test in Table 2.1.

We use published error rates on 85 datasets from [Bagnall et al., 2017] for the following time series classification algorithms: BoP [Lin et al., 2012], BOSS [Schäfer, 2015b], COTE [Bagnall et al., 2015], LTS [Grabocka et al., 2014], PROP [Lines and Bagnall, 2014], SAX-VSM [Senin and Malinchik, 2013] and TSBF [Baydogan et al., 2013] ; as well as published error rates for the SMTS method (45 datasets) [Baydogan and Runger, 2015].

2.3.4.1 *On Standalone Classifiers*

Figures 2.10 & 2.11 show that D-BoTSW performs better than 1NN combined with ED (ED-NN) or DTW (DTW-NN), BoP, LTS SAX-VSM and TSBF ; and that this difference is statistically significant. Indeed the one-sided Wilcoxon signed rank test has a p-value of $p < 5\%$ for all the methods mentioned above. D-BoTSW also performs better than SMTS, however the difference is not statistically significance ($p = 0.173$). We can also notice from Figure 2.11 that BOSS and D-BoTSW have comparable Win/Tie/Lose performance. Note that, if D-BoTSW is not significantly better than BOSS ($p = 0.862$), the reverse is also true ($p = 0.139$).

It is striking to realize that D-BoTSW not only improves classification accuracy, but might improve it considerably. Error rate on *Shapelet Sim* dataset drops from 0.495 (ED-NN) and 0.348 (DTW-NN) to 0 (D-BoTSW), for example. Pairwise comparisons of methods show that D-BoTSW is significantly better than almost all state-of-the-art standalone classifiers (*i.e.* better than 7 out of 8 tested baselines).

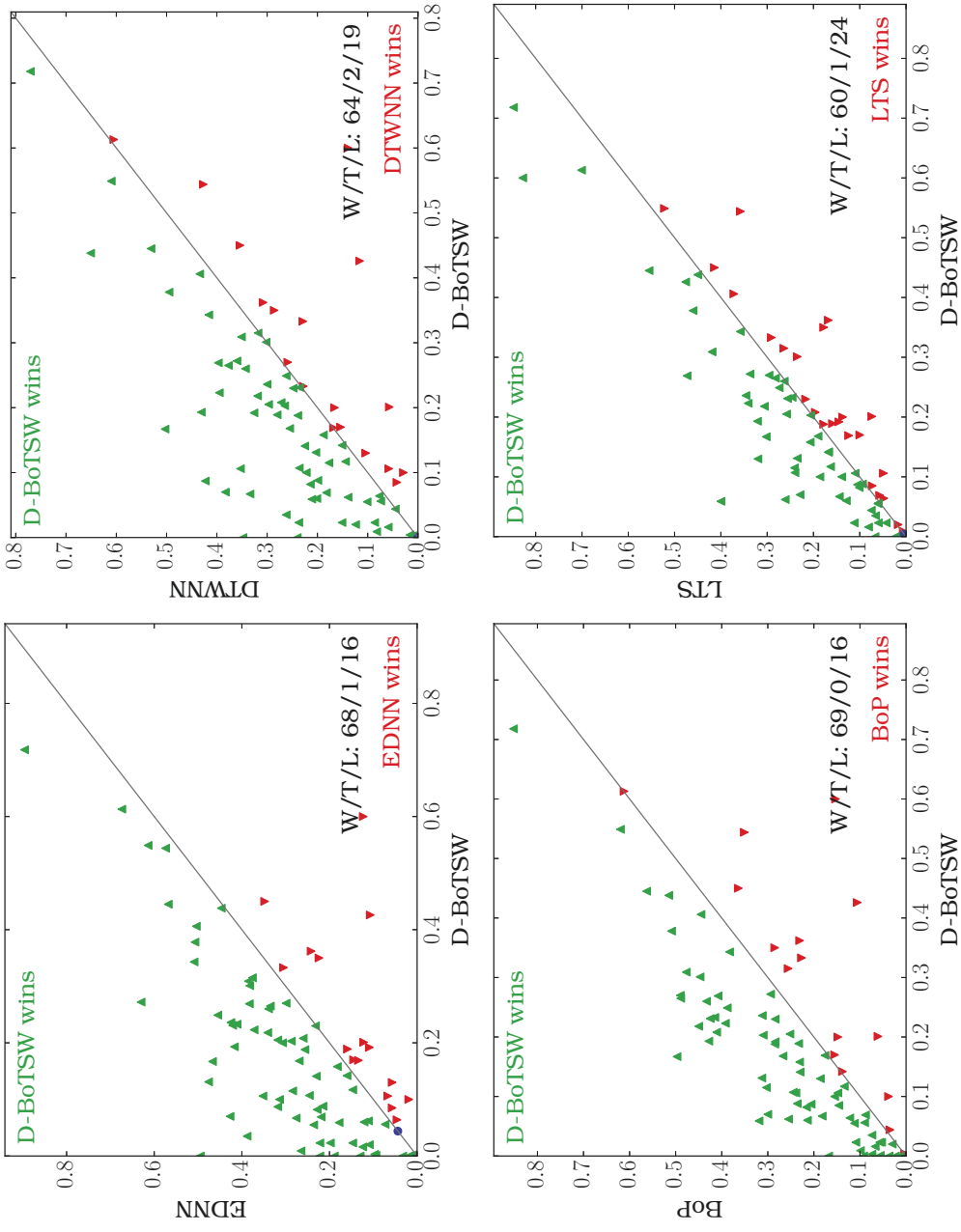


Figure 2.10 – Error rates for D-BoTSW (SSR+$\mathcal{L}_2$) versus standalone baseline classifiers (ED-NN, DTW-NN, BoP and LTS).

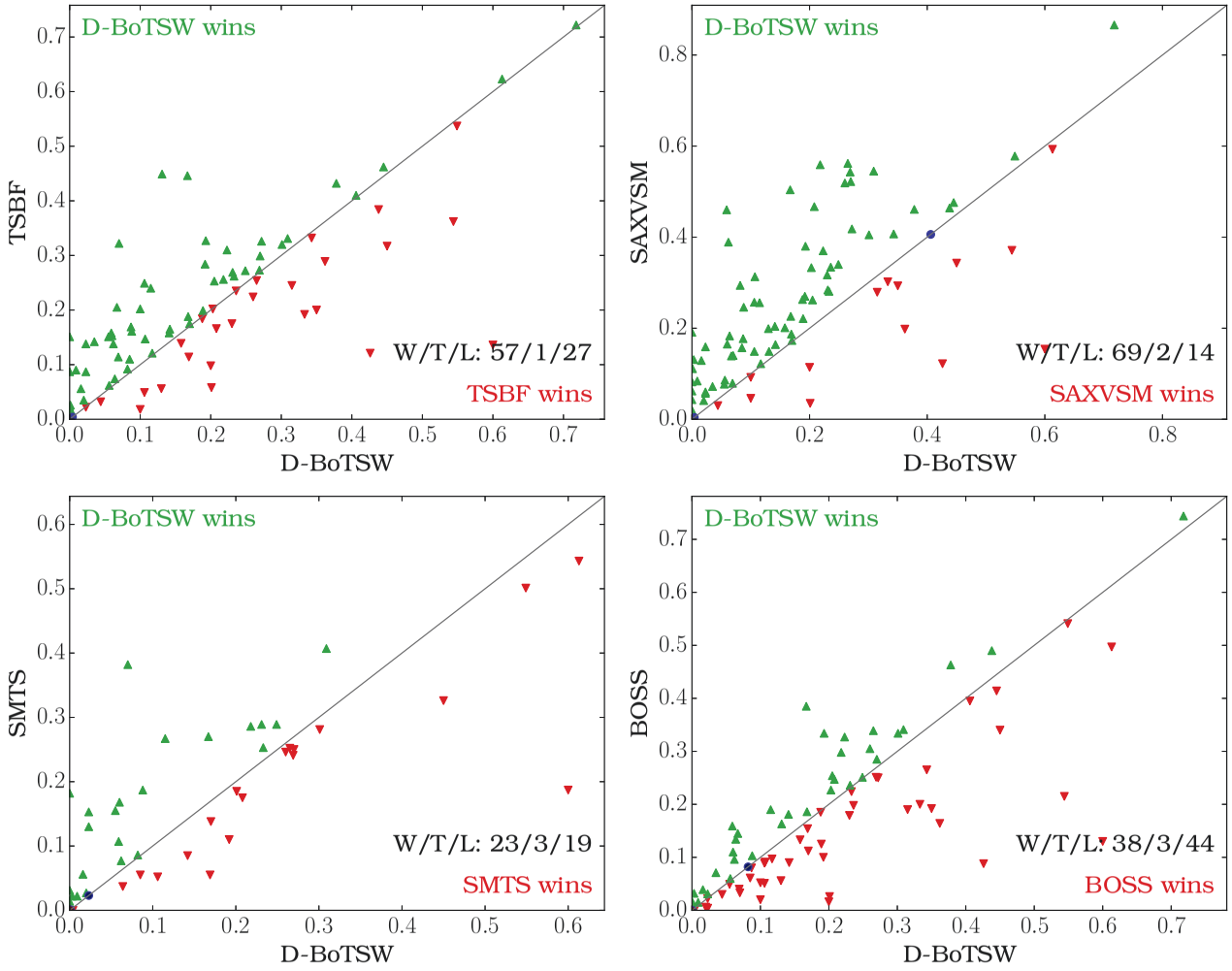


Figure 2.11 – Error rates for D-BoTSW (SSR+$\mathcal{L}_2$) versus standalone baseline classifiers (TSBF, SAX-VSM, SMTS and BOSS).

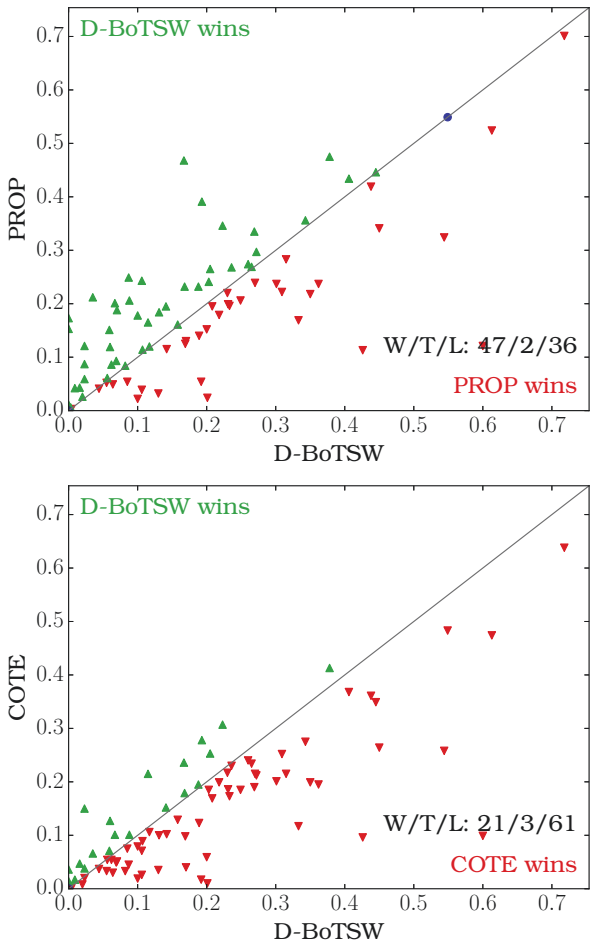


Figure 2.12 – Error rates for D-BoTSW (SSR+$\mathcal{L}_2$) versus baseline ensemble classifiers (PROP and COTE).

2.3.4.2 *With Ensemble Classifiers*

In Figure 2.12, we compare our standalone classifier D-BoTSW to two ensemble classifiers: PROP and COTE. PROP and COTE are considered to be the best time series classifiers (in terms of performance). Wilcoxon tests show that COTE outperforms D-BoTSW ($p = 0.0 < 5\%$). However neither D-BoTSW nor PROP is significantly better than the other. It is more than satisfactory to obtain a higher classification rates on more datasets than PROP, with a Win/Tie/Lose of 48/2/36 in favor of D-BoTSW.

Further work based on a more precise representation than the BoW one could lead to a more competitive algorithm, and might even be significantly better than PROP. Moreover, ensemble classifiers can benefit from the design of new standalone classifiers such as D-BoTSW.

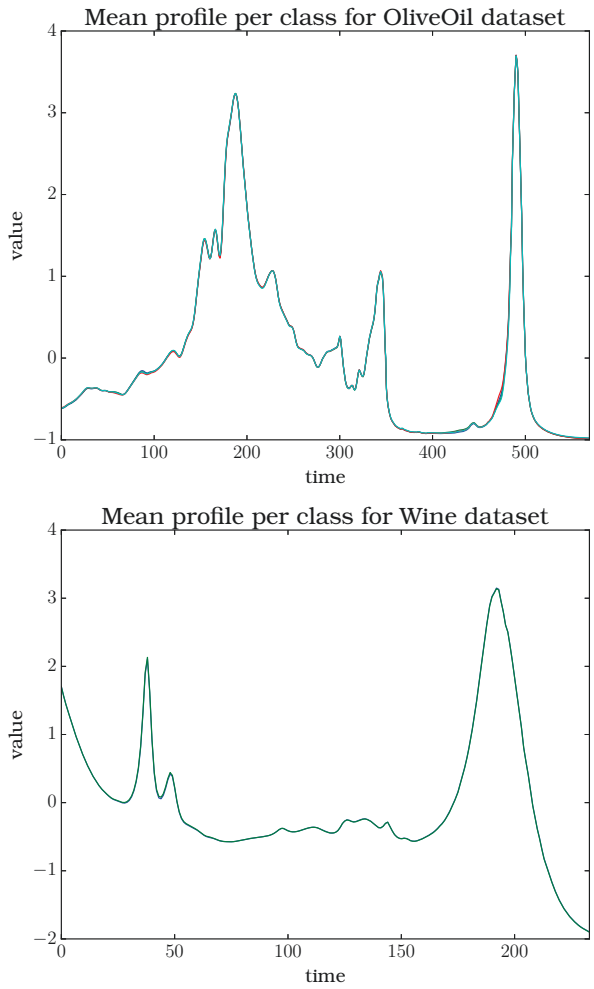


Figure 2.13 – Mean profiles for *OliveOil* and *Wine* datasets per class

2.3.5 *Discussion*

We noticed that D-BoTSW performs especially well in the presence of sufficiently long time series. On the contrary, when faced with short training time series *e.g. ItalyPowerDemand* (of length 24), it is more likely (though still a minority) that non-parametric methods such as DTW-NN or ED-NN are competitive against D-BoTSW. This can be easily explained by the fact that D-BoTSW rely on the notion of neighborhood and on the number of keypoints per time series, which are related with a time series length.

On one hand, D-BoTSW has proven its robustness to noise *e.g.* on simulated UEA / UCR datasets such as *CBF*, *synthetic_control* and *Two_Patterns*. On the other hand, D-BoTSW limitations can be seen on datasets where it is difficult to distinguish the different classes such as *Wine* (2 classes) and

OliveOil (4 classes) as shown on Figure 2.13. Concerning the dense sampling of D-BoTSW, some experiments with very large train set were not able to run on a personal computer due to the amount of space required by the generated features, using a computing cluster with more memory fixed this problem.

D-BoTSW exhibits high accuracy and high performance on a large set of datasets. These experiments, conducted on a wide variety of time series datasets, show that D-BoTSW significantly outperforms most considered standalone classifiers.

3

IMPROVING TIME SERIES SHAPELETS BASED ON ADVERSARIAL EXAMPLES

Contents

Shapelet-based methods, which were first introduced by Ye and Keogh [2009], are a family of time series classification algorithms relying on local shape-based similarity. Amongst them, the Shapelets Transform [Lines et al., 2012] and the Learning Time Series Shapelets [Grabocka et al., 2014] methods are the most competitive *w.r.t.* the state-of-the-art time series classification baselines.

Convolutional Neural Networks (CNNs) are a family of neural networks that exploits the local spatial correlation present in natural images. Szegedy et al. [2014] showed that neural networks can be fooled by adversarial examples which correspond to close-to-original synthetic data for which the network fails to predict the right class label. Szegedy et al. [2014] and Goodfellow et al. [2015] both proposed a methodology that uses adversarial examples in order to generate more robust classifiers. Indeed, one property of adversarial examples is that they can provide an additional regularization benefit.

In this chapter, we propose an enhancement of the Learning Time Series Shapelets algorithm [Grabocka et al., 2014] based on the insertion of adversarial time series in the training procedure. We first introduce the different key concepts we built on, then show how they can be related to each other. Finally, we present our proposed method first theoretically then experimentally.

3.1 *Related Work*

Our approach for time series classification , which is named Adversarially-Built Shapelets (ABS), builds on three techniques from machine learning and deep learning communities. The classifier *Learning Time series Shapelets* (LTS) proposed by [Grabocka et al., 2014] learns time series shapelets that will be used to predict the class labels of new data. In order to improve this method, we (a) make the link between the LTS algorithm and CNNs [Krizhevsky et al., 2012; LeCun et al., 1995], (b) use a regularization method specific to CNNs (*i.e.* adversarial training [Goodfellow et al., 2015; Szegedy et al., 2014]), (c) show that this regularization also works for the LTS model.

3.1.1 *On Learning Time Series Shapelets*

Details about time series shapelets baselines are available in Chapter 1. We focus here on the LTS algorithm [Grabocka et al., 2014], which learns the shapelets instead of looking for them.

LTS is based on logistic regression where both the weights of the logistic regression and the shapelet coefficients are learned and updated through the iterations of a gradient descent process. The process can be seen as the

following: at each iteration, the time series is represented by a feature vector corresponding to the similarity between the time series and the different shapelets. Then, a logistic regression is applied in order to get the prediction for the time series. The LTS method learns the K best shapelets.

More precisely, it starts by initializing the shapelets to K-means cluster centers of all the sub-series of length L extracted from the training time series. Then, the shapelets are updated (learned) by minimizing a classification loss function through the iterations of a gradient descent optimization process. The idea is to jointly learn both the weights of the logistic regression $\boldsymbol{w}$ and the shapelets $\mathcal{S}$ that minimize the following classification objective function $\mathcal{F}$:

$$\arg\min_{\mathcal{S},\boldsymbol{w}} \mathcal{F}(\mathcal{S},\boldsymbol{w}) = \arg\min_{\mathcal{S},\boldsymbol{w}} \sum_{i=1}^{N} \mathcal{L}(y_i, \boldsymbol{w}\boldsymbol{M}_i + b) \tag{3.1}$$

where

- $\mathcal{L}$ is the classification loss function, defined as

$$\mathcal{L}(y_i, \boldsymbol{w}\boldsymbol{M}_i + b) = -y_i \cdot \ln\big(\sigma(\boldsymbol{w}\boldsymbol{M}_i + b)\big) + (1 - y_i) \cdot \ln\big(1 - \sigma(\boldsymbol{w}\boldsymbol{M}_i + b)\big) \tag{3.2}$$

- $\boldsymbol{M}$ is the $N \times K$ matrix representing the distances between a dataset $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^{N}$ and a set of K shapelets $\mathcal{S} = \{\boldsymbol{s}_k\}_{k=1}^{K}$. The dissimilarity measure between the i-th time series $\boldsymbol{x}_i$ and the k-th shapelet $\boldsymbol{s}_k$ is computed as

$$M_{i,k} = d(\boldsymbol{x}_i, \boldsymbol{s}_k) = \min_{j=1,\dots,J} \frac{1}{L} \sum_{\ell=1}^{L} (x_{i,j+\ell-1} - s_{k,\ell})^2 \tag{3.3}$$

  where $d(\boldsymbol{x}_i, \boldsymbol{s}_k)$ corresponds to the minimal distance between a time series $\boldsymbol{x}_i$ (of length n) and a shapelet $\boldsymbol{s}_k$ (of length L) and $J = n - L + 1$.

For a specific time series $\boldsymbol{x}_i$, the objective function is defined as

$$\mathcal{F}_i = \mathcal{L}(\boldsymbol{\theta}|\boldsymbol{x}_i, y_i) + \frac{\lambda_W}{N} \sum_{k=1}^{K} w_k^2 \tag{3.4}$$

with $\boldsymbol{\theta} = (\boldsymbol{w}, b)$.

This approach offers a significant improvement in term of accuracy compared to approaches searching shapelets. It is also the first shapelet-based method which actually learns the shapelets instead of searching them.

3.1.2 *On Convolutional Neural Networks*

3.1.2.1 *Neural Networks Classifiers*

Also called *artificial neural networks*, they are systems based on highly connected nodes (neurons) often organized in layers and are inspired by

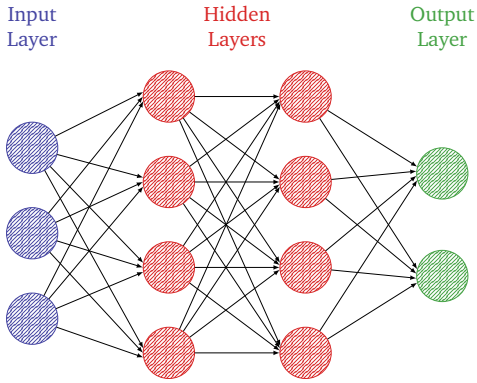


Figure 3.1 – Schematization of a Multi-Layer Perceptron with two hidden layers

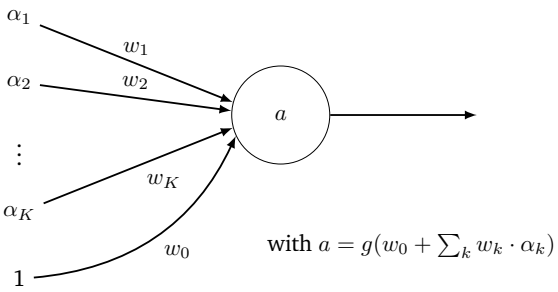


Figure 3.2 – Artificial Neural Network – Zoom on a neuron

biological neural networks (see Figure 3.1 for an example of a possible artificial neural network's architecture: Multi-Layer Perceptron). There exists several types of architecture of neural networks, we only detail the one used in this manuscript: the CNN model. A neuron is the elementary unit of a artificial neural network, it is connected to input sources (such as other neurons) and returns a new information as output. Each neuron of a layer performs a transformation on its inputs $(\alpha_1, \ldots, \alpha_K)$. Each input is associated with a weight coefficient and the neuron considers the weighted sum of its inputs: $b + \sum_{k=1}^{K} w_k \alpha_k$ (with b the bias coefficient) as the input of an activation function g. The output of a neuron will thus be (see Figure 3.2)

$$a = g(b + \sum_{k=1}^{K} w_k \alpha_k) \tag{3.5}$$

The activation function g is usually a non-linear function, such as the following sigmoid function

$$g(z) = \sigma(z) = \frac{1}{1 - e^{-z}}, \tag{3.6}$$

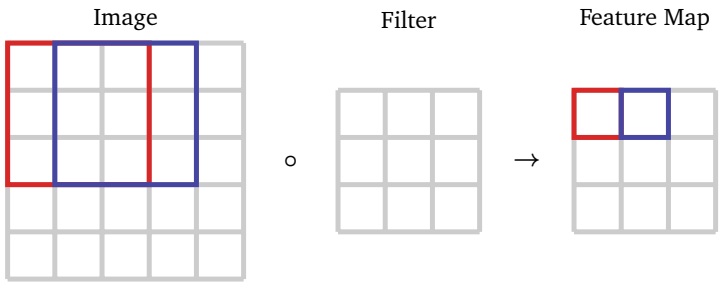


Figure 3.3 – Convolutional Neural Network – Image Convolution

or the Rectified Linear Unit (ReLU) function

$$g(z) = \max(0, z). \tag{3.7}$$

The activation functions have different characteristics and properties, *e.g.* the sigmoid function is differentiable which can be useful. Both ReLu and sigmoid function gives intermediate values, it is not always the case (*e.g.* the binary step activation function that returns 0 for $z < 0$ and 1 otherwise).

Once the model is learned, the network is able to perform fast classification. Neural networks perform well on a wide variety of problems (*e.g.* images, speech, and time series classification [LeCun et al., 1995]) and can thus be found in many domains. However, due to their complexity (*e.g.* number of parameters, layers) they are often seen and used as *black boxes*; *e.g.* AlexNet [Krizhevsky et al., 2012] has 60 millions of parameters, it is thus nearly impossible to determine the most discriminative features by studying the weights of multiple layers.

Neural networks exist in many form such as Convolutional Neural Networks (CNNs), which have been successfully applied for image recognition and classification. These networks are invariant to small transformations, distortions as well as translations. Each CNN is built on a set of three types of layers: (a) Convolution, (b) Pooling, (c) Fully Connected.

3.1.2.2 *Convolution Layer*

The primary purpose of convolution is to extract features from the input image using different filters. The number of filters as well as their size are defined by the user.

Let us consider a grayscale image represented by a 2D matrix of pixels. In order to obtain the features of an image, we apply a convolution between the image and a smaller matrix: the filter. The result obtained by the convolution of the image with the filter is called *feature map*. Since the image and the filter have different sizes, we extract sub-regions of the image using the sliding window technique, as shown on Figure 3.3. Sub-regions of the image are considered, CNNs thus exploit the *spatial* relationship among

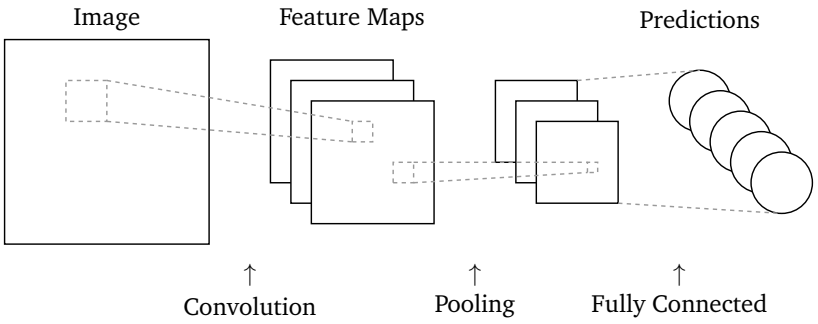


Figure 3.4 – Convolutional Neural Network overview, minimal example with three filters, five classes, and one layer of each type (convolution, pooling, fully connected).

neighboring pixels of an image. Each convolution provides one feature that corresponds to the dot product between the filter and the sub-region (on which we apply an activation function g). The convolution thus preserves the spatial relationship between neighboring pixels by learning features using sub-regions of the image.

Different filters will have different effects since they detect different characteristics from an image. Some filters can blur the visual aspect of an image (*e.g.* gaussian filter), whereas different filters will sharpen it or detect its edges. Different filters will thus result in different feature maps. The filters will be learned by the CNN during the training process.

3.1.2.3 *Pooling Layer*

Pooling is also referred to as subsampling since it reduces the dimensionality of each feature map in order to retain the most important information. There exists different types of pooling: max, average, min, etc, that *respectively* takes the larger, the average, the smaller feature value amongst the considered sub-region of the feature map. The pooling operation has several benefits, such as reducing the number of parameters of the network and making the model more robust (*e.g.* invariant to small deformations).

3.1.2.4 *Fully Connected Layer*

Fully connected layers are used to predict the class label of an input image, which convolution and pooling layers can not do. There is thus at least one fully connected layer at the end of a CNN to perform the classification. Fully connected layers join each neurons from its layer to every neurons from the previous layer, in order to use features generated by convolution and pooling layers to classify the input data. The predictions correspond to probabilities of belonging to a specific class, thus the sum of all proba-

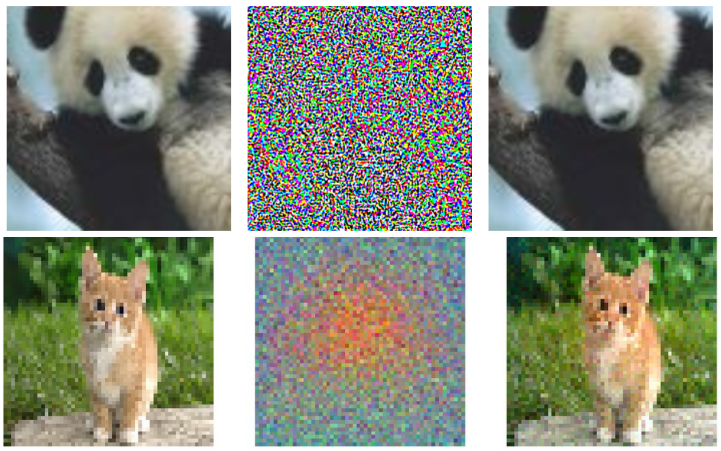


Figure 3.5 – Examples of Adversarial Image. From left to right: an original image I, a perturbation $\boldsymbol{\eta}$ and its adversarial image $\tilde{I} = I + \boldsymbol{\eta}$. On the first line, the image originally classified as a panda (with 60% level confidence) is slightly perturbed then labeled by the same network as some monkey (with 99% level confidence) [Goodfellow et al., 2015]. The second line shows an image originally classified as a kitten and with an adversarial peer labeled as a goldfish by the same network [1].

bilities in the output layer will be one. Finally, the network assigns to the input image the class that has the highest probability among all possible categories.

On Figure 3.4, we can see an input image processed by a CNN which has one convolutional layer with three filters, followed by a pooling layer, and where the network has to choose between five classes.

3.1.3 *On Adversarial Examples*

Szegedy et al. [2014] showed that applying a visually imperceptible perturbation on an image could cause a neural network (and several other machine learning models) to misclassify an image, which is correctly classified without the perturbation, as illustrated on Figure 3.5. This perturbed image is referred to as an *adversarial example*.

Deep neural networks have retained a lot of attention due to their efficiency and high performance on classification tasks. It could be expected from such networks to be robust to small changes of their input. However, a perturbation can be applied on an image in order to fool the network. This perturbation can be generated using a deterministic process that maximizes

1. From Andrej Karpathy's blog: http://karpathy.github.io/2015/03/30/breaking-convnets/

the prediction error of the network.

Let f_N be a neural network such that it takes a $n \times \ell$ image I as input and outputs a class label amongst m possible categories,

$$f_N : \mathbb{R}^{n \times \ell} \longrightarrow \{1, \dots, m\}.$$

To find an adversarial image for the image I using a $n \times \ell$ perturbation $\boldsymbol{\eta}$, the idea is to minimize $\|\boldsymbol{\eta}\|_2$ such that the adversarial image is misclassified: $f_N(I + \boldsymbol{\eta}) \neq f_N(I)$. As long as $\|\eta\|_\infty < \epsilon$, with ϵ sufficiently small for the perturbation to be imperceptible, we expect the classifier to predict the same class label for both I and $\tilde{I}$.

The existence of adversarial examples is due to the fact that deep neural networks are not smooth classifiers *i.e.* classifiers with smooth decision boundaries. As adversarial examples are often shared by different networks, Goodfellow et al. [2015] show that they correspond to low-probability pockets in the decision space and are caused by linear behavior in high dimensional spaces. Informally, adversarial examples are improbable data. However, they expose some important flaws in the model and more particularly in how the model makes its predictions. These low-probability pockets are of particular interest for our purpose as they can help regularize the model, *i.e.* build more robust model, thanks to the generation of adversarial time series.

Adversarial training consists in adding continuously generated adversarial examples into the original training set at each iteration during the training step. At the beginning of each iteration, adversarial examples are generated using the current model, then the weights of the model are updated using both *original* training data and the corresponding adversarial data. At each iteration, new adversarial examples are thus computed. Szegedy et al. [2014] and Goodfellow et al. [2015] proved that adversarial training provides an additional regularization benefit to the classifier, resulting in a reduction of the vulnerability of the model to adversarial examples and in a regularization of the model. Goodfellow et al. [2015] proposed a fast way to generate adversarial examples, making adversarial training feasible: the *fast gradient sign* method.

Let us consider an image I, its label y_I and its adversarial peer $\tilde{I} = I + \boldsymbol{\eta}$. Let θ be the set of parameters of the model and $J(\theta, I, y_I)$ the adversarial objective function, *i.e.* the cost used to train the neural network. The model will compute the dot product between this adversarial image $\tilde{I}$ and a weight vector $\boldsymbol{w}$:

$$\boldsymbol{w}^\top \tilde{I} = \boldsymbol{w}^\top I + \boldsymbol{w}^\top \boldsymbol{\eta} \tag{3.8}$$

The adversarial perturbation will cause a growing of $\boldsymbol{w}^\top \boldsymbol{\eta}$ of the activation. In order to maximize the impact of this increase under some constraints Goodfellow et al. [2015] propose to set $\boldsymbol{\eta}$ to

$$\boldsymbol{\eta} = \epsilon \cdot \text{sign}(\nabla_I \, J(\theta, I, y_I)) \tag{3.9}$$

Indeed, it will enable one to obtain an optimal max-norm constraint perturbation. This way of generating adversarial examples is called the *fast gradient sign descent*. Goodfellow et al. [2015] observe that adversarial examples occur in contiguous regions and not in fine pockets. We can thus consider a direction of perturbation rather than specific points in decision space, since it is the direction that matters the most.

3.2 *Link between Convolutional Neural Networks and Learning Time Series Shapelets*

In this section, we first start by explaining that the LTS algorithm is a special case of convolutional neural network (for 1D data). Then, we will explain how this fact can be used to build a more robust algorithm that learns time series shapelets.

3.2.1 *LTS representation as a CNN*

In LTS, the primary idea is to jointly learn both the weights of the logistic regression $\boldsymbol{w}$ and the shapelets $\mathcal{S}$ that minimize the classification objective function $\mathcal{F}$ (eq. 3.1):

$$\arg\min_{\mathcal{S},\boldsymbol{w}} \mathcal{F}(\mathcal{S},\boldsymbol{w}) = \arg\min_{\mathcal{S},\boldsymbol{w}} \sum_{i=1}^{N} \mathcal{L}(y_i, \boldsymbol{w}\boldsymbol{M}_i + w_0)$$

with (eq. 3.3):

$$M_{i,k} = \min_{j=1,\dots,J} \frac{1}{L} \sum_{\ell=1}^{L} (x_{i,j+\ell-1} - s_{k,\ell})^2$$

In the following, we denote $\boldsymbol{\tau}_i^j$ as the subsequence of the i-th time series $\boldsymbol{x}_i$ starting at the j-th index and of length L: $\boldsymbol{\tau}_i^j = [x_{i,j}, x_{i,j+1}, \dots, x_{i,j+L-1}]$.

In order to compute $\boldsymbol{M}_i$, we need to compute the minimal distances between the time series $\boldsymbol{x}_i$ and the set of shapelets $\mathcal{S}$ (of length L). It can be divided in the following steps:

1. First, we compute the distances between each shapelet and all the possible subseries of $\boldsymbol{x}_i$. The sub-series of $\boldsymbol{x}_i$ can be extracted using a sliding window technique. Shapelets from LTS are thus equivalent to filters from the convolution step of a CNN, computing the distance between shapelets and each subsequence of a time series thus corresponds to a convolution layer for time series. For LTS, the convolution exploits the temporal relationship between neighboring points, which is equivalent to a convolution applied on an image which exploits the

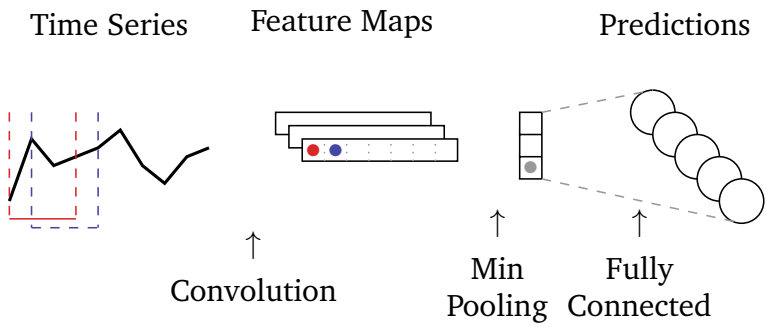


Figure 3.6 – Learning time series shapelets algorithm as a convolution neural network, with three shapelets (filters) and five classes.

spatial relationship between neighboring pixels. Indeed, the different subseries correspond to sub-parts of the time series extracted at regular step.

2. Then, we select the minimal distances between $\boldsymbol{x}_i$ and $\boldsymbol{s}_k$ for $i \in \{1, 2, \ldots, N\}$ and $k \in \{1, 2, \ldots, K\}$. This step corresponds to a min-pooling layer.
3. Finally, we use the weights of the logistic regression $\boldsymbol{w}$ in order to predict the class label of the time series, which is equivalent to the fully connected layer of a CNN.

All these steps are pictured in Figure 3.6, where:

- The red and blue bullets *respectively* correspond to the distance between the first shapelet $\boldsymbol{s}_1$ and two subsequences of the time series: $\boldsymbol{\tau}_i^1$ & $\boldsymbol{\tau}_i^2$.
- The grey bullet corresponds to the minimal distance between the first shapelet $\boldsymbol{s}_1$ and the time series $\boldsymbol{x}_i$,
- In order to get the predictions, we use the logistic regression weights

In the literature, this relationship between LTS and CNN has been mentionned – to our knowledge – only in Cui et al. [2016] and Lods et al. [2017].

3.2.2 *What does it means that LTS is a specific case of a CNN? Which impact does it have?*

We showed that it is possible to write the learning time series shapelets algorithm as a instance of a convolutional neural network. Szegedy et al. [2014] and Goodfellow et al. [2015] both show that CNNs are vulnerable to adversarial examples and that adversarial training can be used to build more robust models for image classification. A natural way to improve the LTS

algorithm is to combine adversarial examples (*i.e.* adversarial time series) with the original training set during the model training.

In the following, we provide an improvement of the LTS method, which is based on adversarial examples in order to generate more robust time series shapelets. At each epoch during the learning process, we learn new adversarial time series that we use – in association with original time series – to update the weights and the shapelets. Adversarial time series provide an additional regularization benefit for the shapelets and experiments show an improvement of the performance between the LTS baseline and our framework.

3.3 *The proposed method*

We first start by studying the general case formulation as introduced in [Goodfellow et al., 2015]. Then, we consider the shapelet formulation for our specific case. In each case, the impact of adversarial training is investigated.

3.3.1 *General problem formulation*

We first start by considering a linear model. We remind that the logistic sigmoid function is defined as

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{3.10}$$

Given a set of examples of N input time series: $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ and their associated class labels $\boldsymbol{y} = \{y_1, y_2, \ldots, y_N\}$, we have for the binary classification case ($y \in \{-1, 1\}$) and for linear models

$$P(y_i = 1|\boldsymbol{\theta}, \boldsymbol{x}_i) = \sigma(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \tag{3.11}$$

Where y_i corresponds to the target (*i.e.* class label) associated with the i-th time series $\boldsymbol{x}_i$ and where the parameters of the model are represented by $\boldsymbol{\theta} = (\boldsymbol{w}, b)$. Thus,

$$P(y_i = -1|\boldsymbol{\theta}, \boldsymbol{x}_i) = 1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \tag{3.12}$$

$$= \sigma(-(\boldsymbol{w}^\top \boldsymbol{x}_i + b)) \tag{3.13}$$

Consequently, for $y_i \in \{-1, 1\}$, we can write

$$P(y_i|\boldsymbol{\theta}, \boldsymbol{x}_i) = \sigma(y_i(w^\top \boldsymbol{x}_i + b)) \tag{3.14}$$

In order to define the cost function, we first have to consider the definition of the likelihood function where the joint probability is the product of

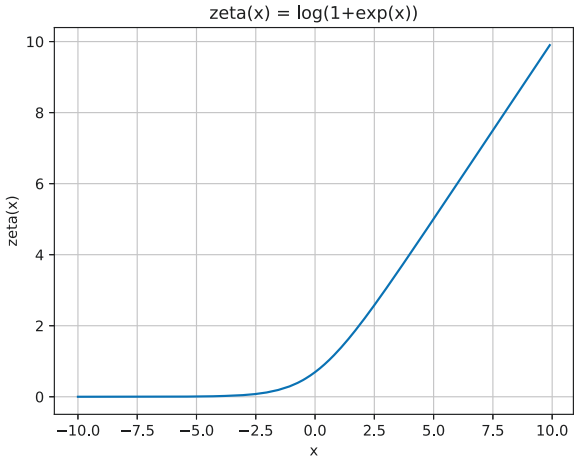


Figure 3.7 – Function $\zeta(z) = \log(1 + \exp(z))$

all individual probabilities.

$$\mathcal{L}(\boldsymbol{\theta}|\mathcal{X}, \boldsymbol{y}) = P(\boldsymbol{y}|\boldsymbol{\theta}, \boldsymbol{x}) = \prod_{i=1}^{N} P(y = y_i|\boldsymbol{\theta}, \boldsymbol{x}_i) \tag{3.15}$$

For convenience, we choose to use the log-likelihood function, defined as

$$\log \mathcal{L}(\boldsymbol{\theta}|\mathcal{X}, \boldsymbol{y}) = \sum_{i=1}^{N} \log P(y = y_i|\boldsymbol{\theta}, \boldsymbol{x}_i) \tag{3.16}$$

The training cost corresponds to the negative log-likelihood of probabilities, so

$$J(\boldsymbol{\theta}, \mathcal{X}, \boldsymbol{y}) = -\log \mathcal{L}(\boldsymbol{\theta}|\mathcal{X}, \boldsymbol{y}) \tag{3.17}$$

$$= \sum_{i=1}^{N} \log \left(1 + e^{-y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b)}\right) \tag{3.18}$$

$$= \sum_{i=1}^{N} \zeta(-y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b)) \tag{3.19}$$

with $\zeta(z) = \log(1 + \exp(z))$, a representation of this function is provided in Figure 3.7.

Finally for the general problem formulation, the training cost can be written as

$$J(\boldsymbol{\theta}, \mathcal{X}, \boldsymbol{y}) = \zeta(-\boldsymbol{y}(\boldsymbol{w}^\top \mathcal{X} + b)) \tag{3.20}$$

Impact of Adversarial Training If we consider a particular time series $\boldsymbol{x}_i$ and its adversarial equivalent $\tilde{\boldsymbol{x}}_i$, then we have

$$J(\boldsymbol{\theta}, \boldsymbol{x}_i, y_i) = \zeta(-y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b)) \tag{3.21}$$

and

$$J(\boldsymbol{\theta}, \tilde{\boldsymbol{x}}_i, y_i) = \zeta(-y_i(\boldsymbol{w}^\top \tilde{\boldsymbol{x}}_i + b)) \tag{3.22}$$

$$= \zeta(-y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + \boldsymbol{w}^\top \boldsymbol{\eta}_i + b)) \tag{3.23}$$

We know that $\boldsymbol{\eta}_i = \epsilon \cdot \text{sign}(\nabla_{\boldsymbol{x}_i} J(\theta, \boldsymbol{x}_i, y_i)) = -\epsilon\, y_i \cdot \text{sign}(\boldsymbol{w})$, thus

$$J(\boldsymbol{\theta}, \tilde{\boldsymbol{x}}_i, y_i) = \zeta(-y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) + \epsilon\|\boldsymbol{w}\|_1) \tag{3.24}$$

If we compare the training cost of *normal* and adversarial time series, we see that the perturbation results in the addition of a regularization term $\epsilon\|\boldsymbol{w}\|_1$ into the ζ function. The term $\epsilon\|\boldsymbol{w}\|_1$ would suggest a $\mathcal{L}_1$ regularization however the fact that this term is inside the ζ function implies some important differences.

- First case: The model makes an accurate prediction.
  We know that for $-y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) << 0$, *i.e.* is sufficiently lower than 0, we have $J_i(\boldsymbol{\theta}, \boldsymbol{x}_i, y_i) \simeq 0$; and y_i and $(\boldsymbol{w}^\top \boldsymbol{x}_i + b)$ have the same sign consequently the model prediction is both confident and accurate for the label prediction of x_i.
  If we add $\epsilon\|\boldsymbol{w}\|_1$ before feeding to the ζ function, we will also have $\tilde{J}_i(\boldsymbol{\theta}, \boldsymbol{x}_i, y_i) \simeq 0$ if the prediction is confident enough. Under this condition, $\tilde{J}_i(\boldsymbol{\theta}, \boldsymbol{x}_i, y_i)$ and $J_i(\boldsymbol{\theta}, \boldsymbol{x}_i, y_i)$ will be equivalent and the model will act as if no regularization has be done. Consequently, the penalty which is added during training disappears once the model has learned to make confident enough predictions.
- Second case: The model makes an inaccurate prediction.
  We also know that for $-y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) > 0$, we have $(\boldsymbol{w}^\top \boldsymbol{x}_i + b)$ and y_i of opposite signs *i.e.* it corresponds to the case where the model prediction is inaccurate.
  Here, $J_i(\boldsymbol{\theta}, \boldsymbol{x}_i, y_i) > 0$ and adding the perturbation will result in an increasing cost: $\tilde{J}_i(\boldsymbol{\theta}, \boldsymbol{x}_i, y_i) > J_i(\boldsymbol{\theta}, \boldsymbol{x}_i, y_i)$ (since $\zeta(z + c) > \zeta(z)$ for $c > 0$). A closer look at the perturbation term enables one to see that the $\mathcal{L}_1$ norm is applied on $\boldsymbol{w}$, which will tend to shrink the coefficients, but only when the model is inaccurate. Whereas $\mathcal{L}_1$ regularization continues to shrink the coefficients though the iterations process even when the model makes confident predictions.

With adversarial training, the regularization is taken into account when the model does not fit the data, whereas when the model fits the data the regularization does not have any effect. This is a very interesting property since the model will be able to converge to the true $(\boldsymbol{w}, b)$, unlike $\mathcal{L}_1$ which

fails to deactivate in case of good margin. The perturbation on inputs will thus only shrink the coefficients $\boldsymbol{w}$ to prevent overfitting when needed (*i.e.* during training when the model does not make confident predictions).

3.3.2 *Shapelet formulation*

If we consider the classification process proposed by [Grabocka et al., 2014] for the Learning Time series Shapelets algorithm, the problem formulation becomes

$$P(\boldsymbol{y}|\boldsymbol{\theta}, \mathcal{X}) = \sigma(\boldsymbol{w}^\top \boldsymbol{M} + b) \tag{3.25}$$

and the objective function is

$$\mathcal{F}_i = \mathcal{L}(\boldsymbol{\theta}|\boldsymbol{x}_i, y_i) + \frac{\lambda_W}{N} \sum_{k=1}^{K} w_k^2 \tag{3.26}$$

where λ_W is a regularization parameter and $\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{x}_i, y_i)$ corresponds to

$$\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{x}_i, y_i) = -\log P(y_i|\boldsymbol{\theta}, \boldsymbol{x}_i) = \log(1 + e^{-y_i(\boldsymbol{w}^\top \boldsymbol{M}_i + b)}) \tag{3.27}$$

We remind that the perturbation will not be applied directly on $\boldsymbol{M}$ but on the time series $\mathcal{X}$. The perturbation is thus limited by $[\mathcal{X}-\boldsymbol{\eta}; \mathcal{X}+\boldsymbol{\eta}]$, we don't need to add a regularization term for this perturbation (since by definition, we have $\|\boldsymbol{\eta}\|_\infty < \epsilon$). On Figure 3.8, we can observe, for a particular time series, the impact of the limitation $\|\boldsymbol{\eta}\|_\infty < \epsilon$ on its admissible adversarial equivalents.

The purpose here is to define the training cost of the adversarial formulation. To do so, we first compute the gradient of the objective function, then we compute the training cost of the adversarial formulation.

Gradient of the objective function In the following, we denote the gradient of the objective function for the i-th time series with respect to the j-th point of the i-th time series as

$$\nabla_{x_{i,j}} \mathcal{F}_i = \frac{\partial \mathcal{F}_i}{\partial x_{i,j}} \tag{3.28}$$

$$= \frac{\partial \mathcal{L}(\boldsymbol{\theta}|\boldsymbol{x}_i, y_i)}{\partial(\boldsymbol{w}^\top \boldsymbol{M}_i + b)} \sum_{k=1}^{K} \frac{\partial(\boldsymbol{w}^\top \boldsymbol{M}_i + b)}{\partial M_{i,k}} \frac{\partial M_{i,k}}{\partial x_{i,j}} \tag{3.29}$$

where,

- $\boldsymbol{M}_i$ corresponds to the minimal distances between the i-th time series $\boldsymbol{x}_i$ and the set of shapelets $\mathcal{S}$. $\boldsymbol{M}_i = [M_{i,1}, M_{i,2}, \ldots, M_{i,K}]$,
- $\dfrac{\partial \mathcal{L}(\boldsymbol{\theta}|\boldsymbol{x}_i, y_i)}{\partial(\boldsymbol{w}^\top \boldsymbol{M}_i + b)} = \dfrac{\partial \log(1 + e^{-y_i(\boldsymbol{w}^\top \boldsymbol{M}_i + b)})}{\partial(\boldsymbol{w}^\top \boldsymbol{M}_i + b)} = \dfrac{-y_i}{1 + e^{y_i(\boldsymbol{w}^\top \boldsymbol{M}_i + b)}}$,

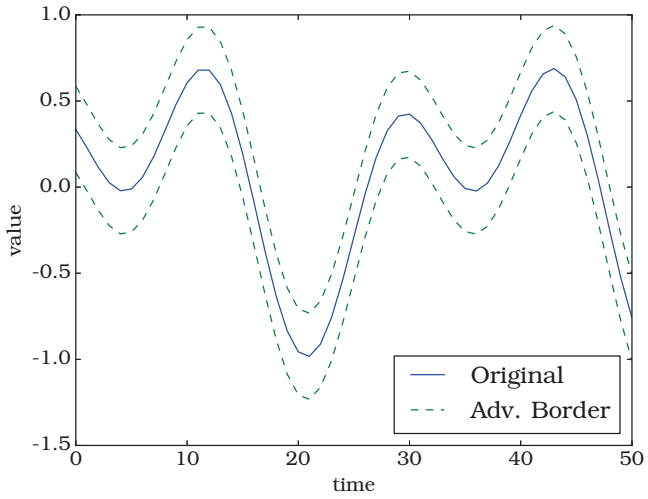


Figure 3.8 – Representation of possible adversarial time series. If $\|\boldsymbol{\eta}\|_\infty < \epsilon$ (here $\epsilon = 0.25$ for readibility), the set of all possible adversarial time series for the blue time series (continuous line) correspond to time series that are included between the two dashed green lines.

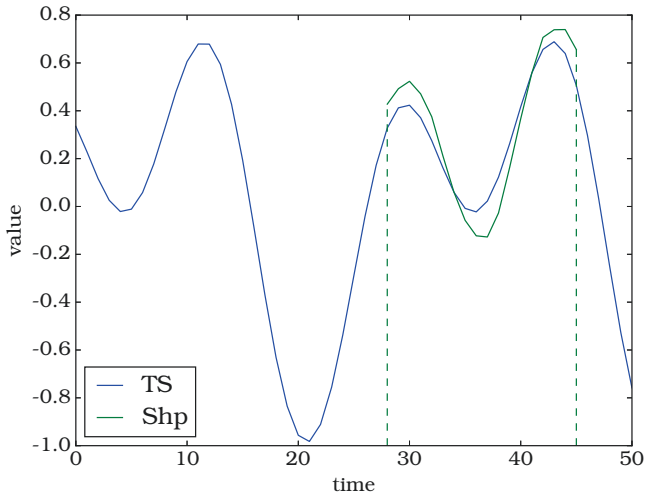


Figure 3.9 – Match between a time series $\boldsymbol{x}_i$ and a shapelet $\boldsymbol{s}_k$ (of length $L = 18$). Here $j_k^* = 28$, *i.e.* the minimal distance between $\boldsymbol{x}_i$ and $\boldsymbol{s}_k$ is obtained for $\tau_i^{j_k^*} = \tau_i^{28}$. Thus $\dfrac{\partial M_{i,k}}{\partial x_{i,j}} \neq 0$ if $j \in [j_k^*, j_k^* + L - 1] = [28, 45]$, *i.e.* if j is one of the indices where $\boldsymbol{x}_i$ matches with $\boldsymbol{s}_k$.

- $\frac{\partial(\boldsymbol{w}^\top \boldsymbol{M}_i + b)}{\partial M_{i,k}} = w_k$,
- $\frac{\partial M_{i,k}}{\partial x_{i,j}} = \frac{\partial \min_j \frac{1}{L}\sum_{l=1}^{L}(x_{i,j+l-1} - s_{k,l})^2}{\partial x_{i,j}}$

  If we consider that the minimum for $\sum_{l=1}^{L}(x_{i,j+l-1} - s_{k,l})^2$ is obtained when j_k^* is the index of $\boldsymbol{x}_i$ where $\boldsymbol{x}_i$ starts to match with $\boldsymbol{s}_k$, then the derivative $\frac{\partial M_{i,k}}{\partial x_{i,j}}$ will be non zero if $j \in [j_k^*, j_k^* + L - 1]$ (Figure 3.9).

  - If $j \in [j_k^*, j_k^* + L - 1]$, then $\frac{\partial M_{i,k}}{\partial x_{i,j}} = \frac{2}{L}(x_{i,j_k^* + l_{j_k^*} - 1} - s_{k,l_{j_k^*}})$ with $l_{j_k^*}$ the position of the match between $\boldsymbol{x}_i$ and $\boldsymbol{s}_k$. Here, we could rewrite j as $j = j_k^* + l_{j_k^*} - 1$ ($1 \leq l_{j_k^*} \leq L$).
  - Else if $j \notin [j_k^*, j_k^* + L - 1]$, then $\frac{\partial M_{i,k}}{\partial x_{i,j}} = 0$.

Since j_k^* and l_k^* depend on k, we have to consider the shapelets $\boldsymbol{s}_k$ only for a limited number of cases, j must be part of the matching indices of $\boldsymbol{x_i}$ with $\boldsymbol{s}_k$ *i.e.* $j \in [j_k^*, j_k^* + L - 1]$. If we define the set of indices $\{k_\ell\}$ (such as $\{k_\ell\} \subseteq \{1, 2, \ldots, K\}$) as the indices of shapelets where $x_{i,j}$ matches with $s_{k_\ell,l}$, we now have

$$\nabla_{x_{i,j}} \mathcal{F}_i = \frac{\partial \mathcal{L}(\boldsymbol{\theta}|\boldsymbol{x}_i, y_i)}{\partial(\boldsymbol{w}^\top \boldsymbol{M}_i + b)} \sum_{k_\ell} \frac{\partial(\boldsymbol{w}^\top \boldsymbol{M}_i + b)}{\partial M_{i,k_\ell}} \frac{\partial M_{i,k_\ell}}{\partial x_{i,j}} \tag{3.30}$$

Training cost of the adversarial formulation If we perturb the time series $\boldsymbol{x}_i$ to obtain $\tilde{\boldsymbol{x}}_i = \boldsymbol{x}_i + \boldsymbol{\eta}_i$ and if we set $\boldsymbol{\eta}_i$ to $\epsilon \cdot \text{sign}(\nabla_{\boldsymbol{x}_i} J(\theta, \boldsymbol{x}_i, y_i))$ (as in eq. 3.9) ; then $M_{i,k}$ will become

$$\tilde{M}_{i,k} = \min_j \frac{1}{L} \sum_{l=1}^{L} \Bigg(\underbrace{x_{i,j+l-1} + \epsilon \,\text{sign}(\nabla_{x_{i,j+l-1}} \mathcal{F}_i)}_{\tilde{x}_{i,j+l-1}} - s_{k,l} \Bigg)^2 \tag{3.31}$$

If we compute the cost function for the i-th time series, we will have

$$J_{i,\mathcal{S}}(\boldsymbol{\theta}, \boldsymbol{x}_i, y_i) = \zeta\left(-y_i(\boldsymbol{w}^\top \boldsymbol{M}_i + b)\right) \tag{3.32}$$

and the adversarial version

$$\tilde{J}_{i,\mathcal{S}}(\boldsymbol{\theta}, \boldsymbol{x}_i, y_i) = \zeta\left(-y_i(\boldsymbol{w}^\top \tilde{\boldsymbol{M}}_i + b)\right) \tag{3.33}$$

$$= \zeta\left(-\frac{y_i}{L}\left(\sum_k w_k \min_j \sum_l \left(x_{i,j+l-1} + \epsilon \,\text{sign}(\nabla_{x_{i,j+l-1}} \mathcal{F}_i) - s_{k,l}\right)^2\right) - y_i b\right) \tag{3.34}$$

Considering that $\min_j$ is obtained for j_k^*, *i.e.* that the minimal distance between $\tilde{\boldsymbol{x}}_i$ and $\boldsymbol{S}_k$ starts at the j_k^*-th index of $\tilde{\boldsymbol{x}}_i$.We deduce that

$$\tilde{J}_{i,\boldsymbol{S}}(\boldsymbol{\theta}, \boldsymbol{x}_i, y_i) = \zeta\Bigg(- \frac{y_i}{L} \sum_k w_k \sum_l \left(\left(x_{i,j_k^*+l-1} - s_{k,l} \right)^2 \right) - y_i \, \epsilon^2 \sum_k w_k - \frac{2y_i\epsilon}{L} \sum_k w_k \sum_l \operatorname{sign}\left(\nabla_{x_{i,j_k^*+l-1}} \mathcal{F}_i \right) \left(x_{i,j_k^*+l-1} - s_{k,l} \right) - y_i b \Bigg) \tag{3.35}$$

$$= \zeta\Bigg(\underbrace{-y_i(\boldsymbol{w}^\top \boldsymbol{M}_i + b)}_{\text{No perturbation}} \underbrace{-y_i \, \epsilon^2 \sum_k w_k - \left(\frac{2y_i\epsilon}{L} \sum_k w_k \sum_l \operatorname{sign}\left(\nabla_{x_{i,j_k^*+l-1}} \mathcal{F}_i \right) \left(x_{i,j_k^*+l-1} - s_{k,l} \right) \right)}_{A} \Bigg) \tag{3.36}$$

where A corresponds to the effect of the adversarial training (*i.e.* the perturbation part). A depends on $\operatorname{sign}\left(\nabla_{x_{i,j_k^*+l-1}} \mathcal{F}_i \right)$, we might use it to obtain a better expression of A.

We know thanks to equation 3.30, that

$$\operatorname{sign}\left(\nabla_{x_{i,j}} \mathcal{F}_i \right) = \operatorname{sign}\left(\frac{\partial \mathcal{L}(\boldsymbol{\theta}|\boldsymbol{x}_i, y_i)}{\partial(\boldsymbol{w}^\top \boldsymbol{M}_i + b)} \sum_{k_\ell} \frac{\partial(\boldsymbol{w}^\top \boldsymbol{M}_i + b)}{\partial M_{i,k_\ell}} \frac{\partial M_{i,k_\ell}}{\partial x_{i,j}} \right) \tag{3.37}$$

$$= -y_i \cdot \operatorname{sign}\left(\sum_{k_\ell} w_{k_\ell} \left(x_{i,j_{k_\ell}^* + l_{j_k^*} - 1} - s_{k_\ell, l_{j_k^*}} \right) \right) \tag{3.38}$$

In equation 3.38, j_k^* corresponds to the first index where $\boldsymbol{x}_i$ and $\boldsymbol{s}_k$ match ; whereas in equation 3.36, j_k^* corresponds to the first index where $\tilde{\boldsymbol{x}}_i$ and $\boldsymbol{s}_k$ match. So in order both j_k^* to be equivalent, we make the assumption that the match with the shapelets does not change between perturbed and the original time series.

By doing so, we are able to keep equations as *simple* as possible (since we can make some simplifications). Moreover, it is a realistic assumption since as long as we use a sufficiently small ϵ the probability that the matches between $\boldsymbol{x}_i$ and $\tilde{\boldsymbol{x}}_i$ with $\boldsymbol{s}_k$ start at the same index j_k^* is high.

According to the notation we have $j = j_k^* + l - 1$ and we previously stated that j should look like $j_k^* + l_{j_k^*} - 1$ in order $\frac{\partial M_{i,k}}{\partial x_{i,j}}$ not to be zero, thus by identification we have $l = l_{j_k^*}$. Consequently,

$$\operatorname{sign}\left(\nabla_{x_{i,j_k^*+l-1}} \mathcal{F}_i \right) = -y_i \operatorname{sign}\left(\sum_{k_\ell} w_{k_\ell} (x_{i,j_{k_\ell}^* + l - 1} - s_{k_\ell, l}) \right) \tag{3.39}$$

Then A becomes,

$$A = -y_i\ \epsilon^2 \sum_k w_k - \frac{2y_i\epsilon}{L} \sum_k w_k \sum_l \text{sign}\left(\nabla_{x_{i,j_k^*+l-1}} \mathcal{F}_i\right)\left(x_{i,j_k^*+l-1} - s_{k,l}\right) \tag{3.40}$$

$$= \underbrace{-y_i\ \epsilon^2 \sum_k w_k}_{A_1} + \underbrace{\frac{2\epsilon}{L} \sum_k w_k \sum_l \text{sign}\left(\sum_{k_\ell} w_{k_\ell}(x_{i,j_{k_\ell}^*+l-1} - s_{k_\ell,l})\right)(x_{i,j_k^*+l-1} - s_{k,l})}_{A_2} \tag{3.41}$$

Now, let us consider the specific case where k_ℓ takes a single value (*i.e.* $x_{i,j}$ matches with a single shapelet) and that this value is k. Depending on the number of shapelets this assumption might not be realistic, however in order to better understand what is happening this simplification is necessary. Considering it, we can simplify A_2 the following way:

$$A_2 = \frac{2\epsilon}{L} \sum_k w_k \sum_l \text{sign}\left(w_k(x_{i,j_k^*+l-1} - s_{k,l})\right)\left(x_{i,j_k^*+l-1} - s_{k,l}\right) \tag{3.42}$$

$$= \frac{2\epsilon}{L} \sum_k |w_k| \sum_l \left|x_{i,j_k^*+l-1} - s_{k,l}\right| \tag{3.43}$$

$$= \frac{2\epsilon}{L} \sum_k |w_k| \underbrace{\left\|\boldsymbol{x}_i^S - \boldsymbol{s}_k\right\|_1}_{\nu_k} \tag{3.44}$$

$$= \frac{2\epsilon}{L} \|\boldsymbol{w}\boldsymbol{\nu}\|_1 \tag{3.45}$$

where $\boldsymbol{x}_i^S = [x_{i,j_k^*}, \ldots, x_{i,j_k^*+L-1}]$.

Thus, under the previous hypothese, A_2 will always be positive. However $\text{sign}(A_1)$ depends on the $\text{sign}(y_i)$, which can be positive or negative. We have two cases:

- When $A_1 < A_2$, the adversarial training will add a positive element into the ζ function. Regularization will only be performed when the model doesn't fit the data; whereas when the model makes confident and accurate predictions, it will act as if no regularization was done.
- However when $A_1 > A_2$, a negative element will be including in the ζ function resulting into a smaller cost ($\tilde{J}_{i,\boldsymbol{S}}(\boldsymbol{\theta}, \boldsymbol{x}_i, y_i) < J_{i,\boldsymbol{S}}(\boldsymbol{\theta}, \boldsymbol{x}_i, y_i)$). When the model fits the data, it will have no impact (since the cost value will still be $\simeq 0$). However if the model predictions are inaccurate, the cost will be underestimated and consequences will be: we won't be abe to measure precisely how close our prediction matches the true one. Consequently, it might perturb the convergence of the model.

If we take a closer look at the expression of A_1 and A_2, we can say that A_1 is most likely to be smaller than A_2. Indeed, in A_1 we add the w_ks without paying attention to their signs, so the sum (of elements that can be either positive or negative) is likely to be smaller than the sum of absolute values of w_k (such as in A_2) even weighted by a positive element (ν_k). Another fact tends to show that A_1 is likely to be smaller than A_2: the sum in A_1 is multiply by a ϵ^2 factor, whereas the sum of absolute values in A_2 is only mutiply by a ϵ factor. Consequently, for small enough epsilon values, the probability to be in the case where $A_1 < A_2$ is high, *i.e.* most of the time, we will apply a regularization when the model does not fit the data and the penalty will disappear when the model makes confident and accurate prediction.

We obtain similar observations that in the general formulation, it thus confirms our intuition that adversarial examples can provide a regularization benefit to LTS.

3.3.3 *ABS algorithm*

There is no differences neither for the initialization of the shapelets nor for the prediction step between LTS and ABS algorithms. We thus only provide a description of the learning step for both LTS and ABS (*respectively* in Algorithms 1 & 2) to easily compare them. The differences are written in green in order to be better visualized. Moreover, we slightly simplify the algorithm (*e.g.* we consider a binary class problem and only one scale for the length of the shapelets) for readibility. Note that the function $generate\_adversarial\_time\_series(\boldsymbol{x}, \epsilon)$ returns the adversarial time series: $\boldsymbol{x}_{adv} = \boldsymbol{x} + \epsilon \ \text{sign}(\nabla_{\boldsymbol{x}} J(\theta, \boldsymbol{x}, \boldsymbol{y}))$.

Interpretability of ABS ABS is built on LTS, the main difference between the two algorithms append during the learning step. Indeed, ABS has to learn adversarial time series at the beginning of each iterations. The model generated by ABS is the same than the one generated by LTS (*i.e.* the shapelets associated with the parameters of the classifier). ABS thus benefits from the same interpretabitily than LTS. The shapelets of the model offer interpretable features; and associated with their corresponding weights, it is even possible to detect the most influential shapelets for each class. An illustration of the interpretability of the model generated by ABS is available in Chapter 4.

Require: Training dataset $\boldsymbol{x} \in \mathbb{R}^{N \times n}$, K shapelets $\mathcal{S}$ (of length L), η the learning rate, the number of iterations `maxIter`.
Ensure: The shapelets $\mathcal{S} \in \mathbb{R}^{K \times L}$, and the parameters $\boldsymbol{\theta} = (\boldsymbol{w}, b)$ (*i.e.* the classification weights $\boldsymbol{w} \in \mathbb{R}^K$ and the bias $b \in \mathbb{R}$).

for iter **from** 1 **to** `maxIter` **do**
  for i **from** 1 **to** N **do**
    for k **from** 1 **to** K **do**
      $w_k \leftarrow w_k - \eta \dfrac{\partial \mathcal{F}_i}{\partial w_k}$
      for l **from** 1 **to** L **do**
        $s_{k,l} \leftarrow s_{k,l} - \eta \dfrac{\partial \mathcal{F}_i}{\partial s_{k,l}}$
      end for
    end for
    $b \leftarrow b - \eta \dfrac{\partial \mathcal{F}_i}{\partial b}$
  end for
end for
return $\mathcal{S}, \boldsymbol{\theta}$

Algorithm 1: Learning Shapelets Algorithm [Grabocka et al., 2014]

Require: Training dataset $\boldsymbol{x} \in \mathbb{R}^{N \times n}$, K shapelets $\mathcal{S}$ (of length L), η the learning rate, the number of iterations `maxIter`, the adversarial perturbation ϵ.
Ensure: The shapelets $\mathcal{S} \in \mathbb{R}^{K \times L}$, and the parameters $\boldsymbol{\theta} = (\boldsymbol{w}, b)$ (*i.e.* the classification weights $\boldsymbol{w} \in \mathbb{R}^K$ and the bias $b \in \mathbb{R}$).

$\boldsymbol{x}_{temp} \leftarrow \boldsymbol{x}$
for iter **from** 1 **to** `maxIter` **do**
  $\boldsymbol{x}_{adv} \leftarrow generate\_adversarial\_time\_series(\boldsymbol{x}_{temp}, \epsilon)$
  $\boldsymbol{x} \leftarrow \boldsymbol{x}_{temp} + \boldsymbol{x}_{adv}$
  for i **from** 1 **to** $2 \times N$ **do**
    for k **from** 1 **to** K **do**
      $w_k \leftarrow w_k - \eta \dfrac{\partial \mathcal{F}_i}{\partial w_k}$
      for l **from** 1 **to** L **do**
        $s_{k,l} \leftarrow s_{k,l} - \eta \dfrac{\partial \mathcal{F}_i}{\partial s_{k,l}}$
      end for
    end for
    $b \leftarrow b - \eta \dfrac{\partial \mathcal{F}_i}{\partial b}$
  end for
end for
return $\mathcal{S}, \boldsymbol{\theta}$

Algorithm 2: Adversarially Built Shapelets Algorithm

3.4 Experiments on UEA / UCR datasets

In this section, we compare Learning Time series Shapelets (LTS) algorithm with our method, called Adversarially-Built Shapelets (ABS) algorithm. We then compare our results to the ones obtained by 10 other relevant baselines (*i.e.* the 9 previous and D-BoTSW).

Experiments are conducted on 74 of the 85 currently available datasets from the UEA / UCR repository [Bagnall et al., 2017], due to memory and time limitations of our computation system. Raw numbers are available in Appendix A.

3.4.1 Source code

We choose to base our ABS code on the time series classification code provided by Bagnall et al. [2016] [2] that includes many classification algorithms. Amongst them, a Java version of the LTS algorithm is implemented.

Considering the number of datasets as well as some computation resource limitation, we decided to set the defaults parameters (as recommended by the author):

- the regularization parameter $\lambda_w = 0.01$,
- the number of scales $R = 3$,
- the percentage of series length to 0.2,
- the number of iterations to 600,

and to slightly modify the `initializeShapeletsKMeans` function. In order to initialize the shapelets we take a random number of 10 000 sub-series (if the number of sub-series is greater than 10 000), or all the sub-series (*i.e.* the algorithm is unchanged).

For the sake of reproducibility, the source code used for LTS and ABS in these experiments are made available for download [3].

3.4.2 Impact of adding adversarial time series

Figure 3.10 shows a pairwise comparison of error rates between LTS and its adversarial counterpart ABS for tested datasets of the UEA / UCR database. ABS (with ϵ set by cross-validation in $\{0.001, 0.01, 0.1\}$) reaches better performance than LTS on 36 datasets, equivalent performance on 20 datasets and worse on 18 datasets. The error rate differences between LTS and ABS might seem small, however this can be explained by the fact that ABS is based on LTS. We used the previously introduced one-sided Wilcoxon

2. https://bitbucket.org/TonyBagnall/time-series-classification
3. https://github.com/a-bailly/adversarially_built_shapelets

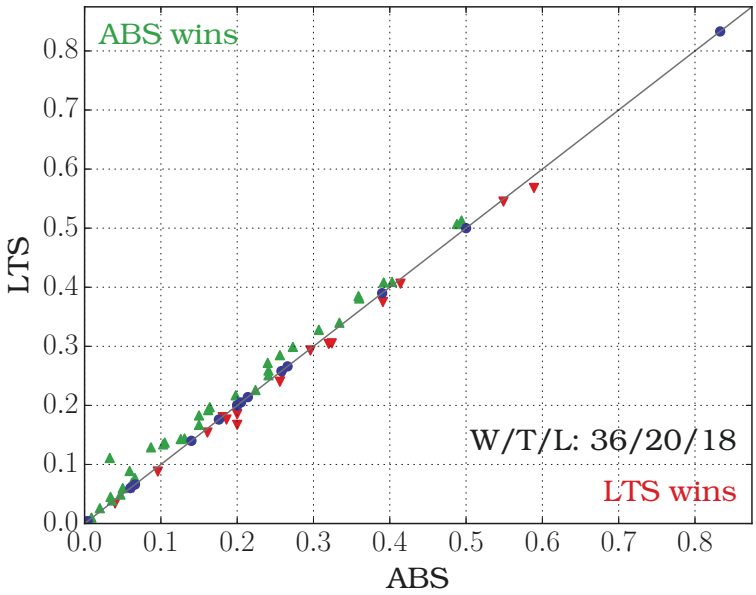


Figure 3.10 – Comparison of error rates obtained by LTS and ABS (with a perturbation ϵ in $\{0.001, 0.01, 0.1\}$)

signed rank test in order to check if the difference is significant. We obtained a p-value of 0.002 which is below the 5% significance level, our experiments thus show that ABS is significantly better than LTS.

3.4.3 *Empirical Comparison of ABS with State-of-the-Art Techniques*

We use published error rates on 85 datasets from [Bagnall et al., 2017] for the following time series classification algorithms: BoP [Lin et al., 2012], BOSS [Schäfer, 2015b], COTE [Bagnall et al., 2015], ED-NN, DTW-NN, PROP [Lines and Bagnall, 2014], SAX-VSM [Senin and Malinchik, 2013] and TSBF [Baydogan et al., 2013]. As well as published error rates for the SMTS method (45 datasets) [Baydogan and Runger, 2015].

3.4.3.1 *On Standalone Classifiers*

From Table 3.1 and Figures 3.11 & 3.12, we can observe that

- The Win/Tie/Lose score indicates that ABS obtains a better classification accuracy more datasets than 1-NN combined with ED (ED-NN) or DTW (DTW-NN), BoP, SAX-VSM and SMTS. ABS performs significantly better than ED-NN, DTW-NN, BoP as well as SAX-VSM (with

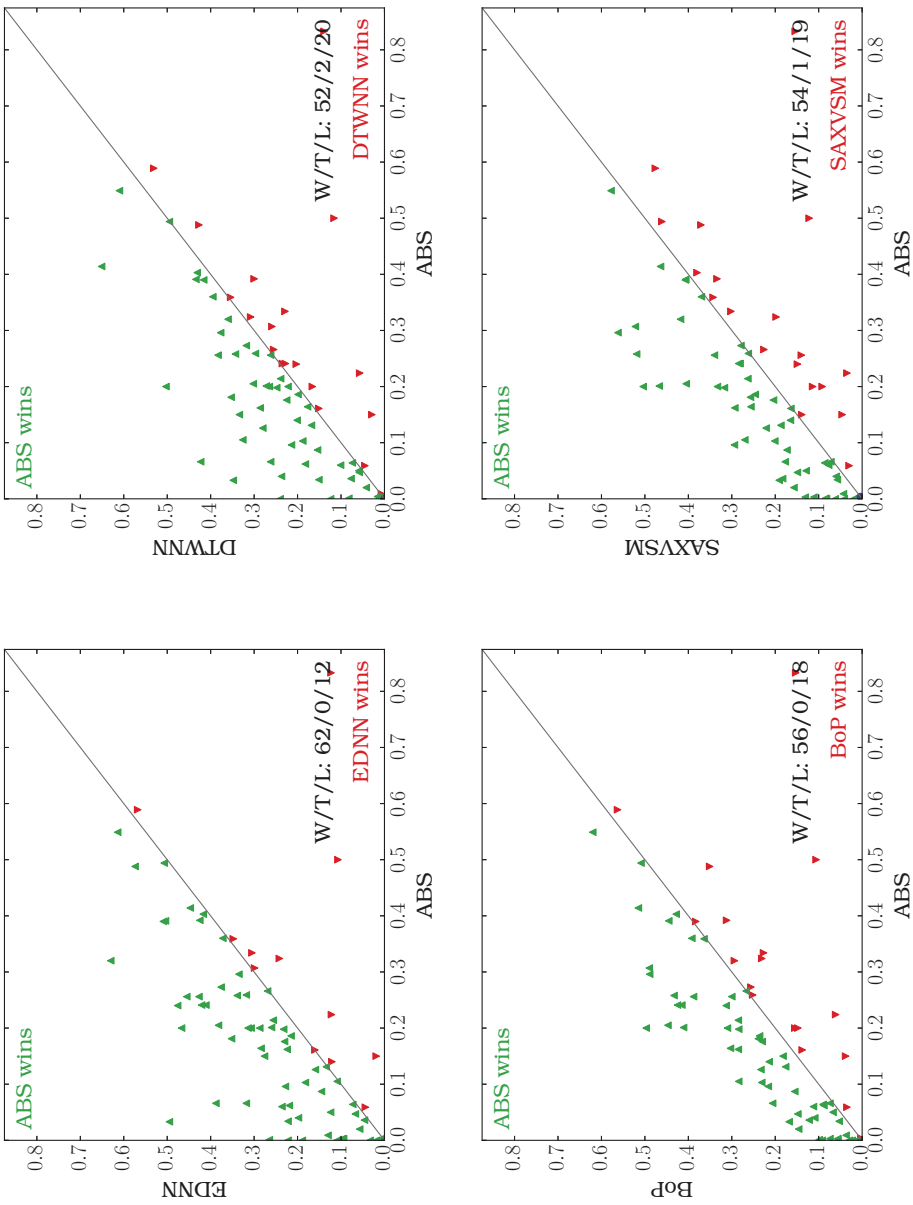


Figure 3.11 – Error rates for ABS versus standalone baseline classifiers (ED-NN, DTW-NN, BoP and LTS-Bagnall).

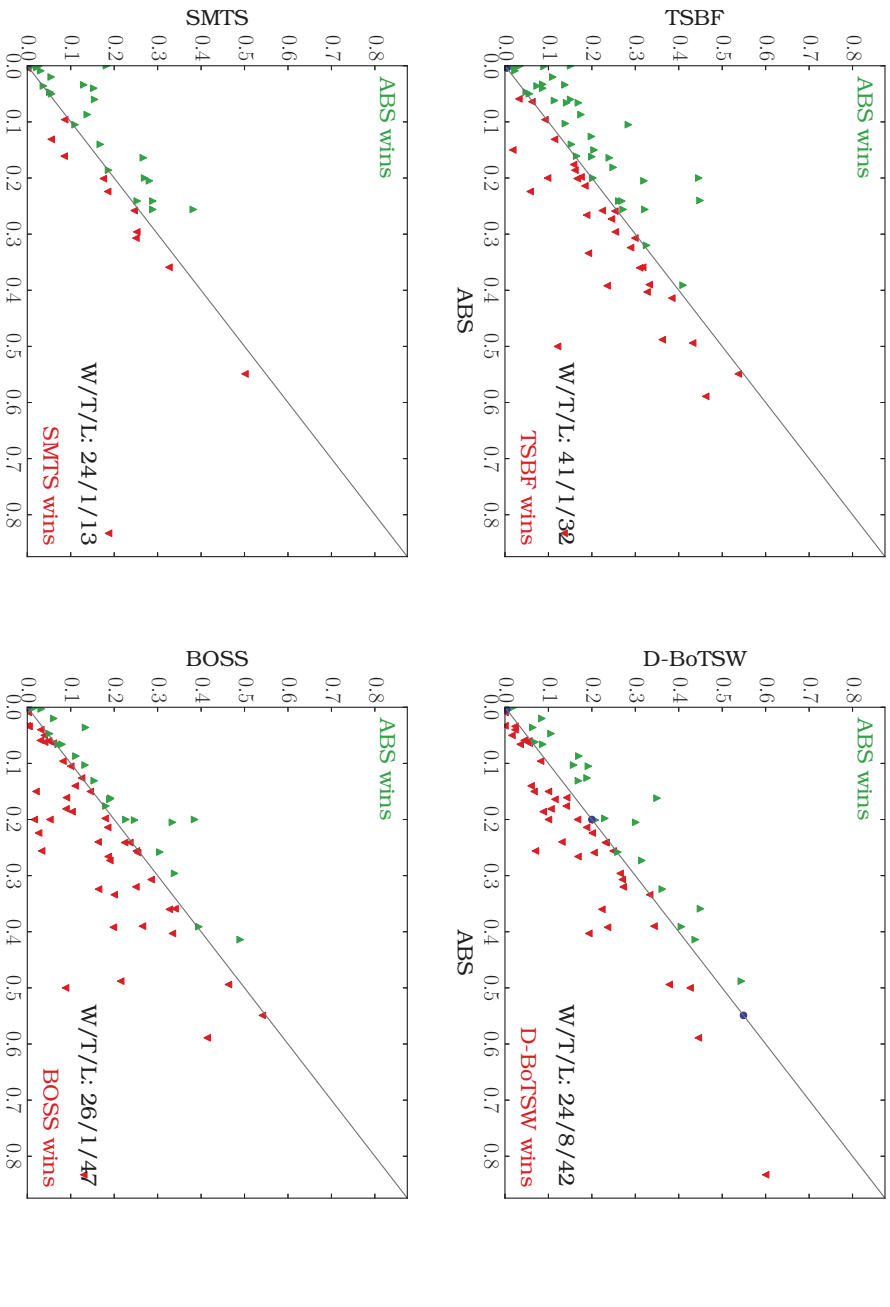


Figure 3.12 – Error rates for ABS versus standalone baseline classifiers (TSBF, SAX-VSM, SMTS and BOSS).

ABS	is significantly better than	BoP	($p = 0.000$)
ABS	is significantly better than	DTW-NN	($p = 0.000$)
ABS	is significantly better than	ED-NN	($p = 0.000$)
ABS	is significantly better than	LTS	($p = 0.002$)
ABS	is significantly better than	SAXVSM	($p = 0.000$)

ABS	is better than	SMTS	($p = 0.053$), however it is not significant
ABS	is better than	TSBF	($p = 0.270$), however it is not significant
PROP	is better than	ABS	($p = 0.243$), however it is not significant

BOSS	is significantly better than	ABS	($p = 0.002$)
COTE	is significantly better than	ABS	($p = 0.000$)
D-BoTSW	is significantly better than	ABS	($p = 0.013$)

Table 3.1 – ABS – One sided Wilcoxon Test p-values. If the p-value is less than the 5% significance level, the method is considered significantly better than the one it is compared to.

5% significance level). If we consider a 10% significance level then ABS is also significantly better than SMTS.

- The Win/Tie/Lose score indicates that ABS obtains a better classification accuracy on a few more datasets than TSBF, however Wilcoxon p-values shows that the difference is not significant.
- Finally both BOSS and D-BoTSW are significantly better than ABS, on the UEA / UCR database.

Note that the UEA / UCR contains a large variety of problems for which all algorithms are not adapted. Thus, in order to choose an algorithm, one should consider the characteristics of its data and not only the performance obtained on the UEA / UCR database.

3.4.3.2 *With Ensemble Classifiers*

In Figure 3.13, we compare our standalone classifier ABS to ensemble classifiers. Wilcoxon tests show that ABS is not statistically better than neither PROP nor COTE (*respectively* $p = 0.758$ and $p = 1.0$). Testing the reverse hypothesis that ABS is outperformed by these methods gives significance for COTE ($p < 5\%$) but not for PROP ($p = 0.243$).

3.5 *Discussion*

Since the first paper on shapelets from Ye and Keogh [2009], many approaches using time series shapelets were proposed. Our contributions for

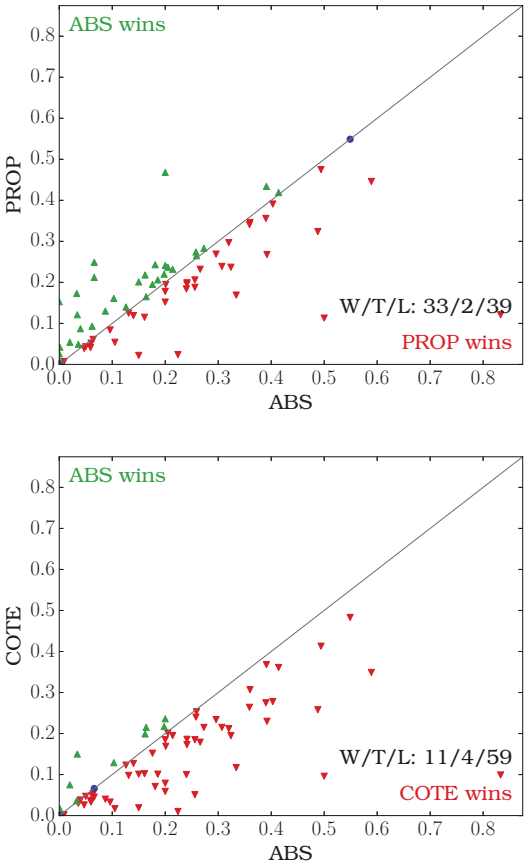


Figure 3.13 – Error rates for ABS versus baseline ensemble classifiers (PROP and COTE).

time series shapelets includes:

1. A formal proof that LTS is a special case of CNN,
2. A theoretical explanation and experiments showing that LTS can be regularized using adversarial training,
3. A regularization of the LTS model using adversarial training (with an easy and fast way to generate adversarial time series).

Considering the growing interest for deep learning model (and in particular for convolutional neural networks), this work could serve as a basis for future work on time series shapelets.

Future work

A fast implementation of ABS using a convolutional neural networks framework could be beneficial in term of computation performance (*e.g.* with CNNs libraries of Python such as `keras`).

Moreover, since we show that LTS is an instance of a CNN, it might be interesting to evaluate the impact of the CNN setup, such as the choice of the activation function or the impact of the number of layers. The shape and size of the filters often play an important role in the learning process of a CNN, further investigation on the shapelet sizes might also be necessary. A more adapted CNN setup can lead to further improvment of the method.

Several methods have been proposed to avoid overfitting for CNNs (such as dropout and early stopping), an evaluation of their impact on the LTS algorithm is relevant and might help building a better algorithm.

4

TIME SERIES CLASSIFICATION: REMOTE SENSING APPLICATIONS

Contents

In this chapter, we investigate *Remote Sensing Time Series Classification* (RS-TSC) problems. Automatic land cover classification from satellite image time series is of paramount relevance to assess vegetation and crop status, with important implications in agriculture, biofuels and food. Moreover it is also very useful in order to monitor urban development as well as to follow land cover changes.

We first present specificities of remote sensing data. Then we introduce two remote sensing time series datasets: *TiSeLac* (proposed by Ienco [2017]) and *Brazilian Amazon* (introduced in Bailly et al. [2016a]). Finally we experiment on these datasets on both time series classification baselines and previously introduced algorithms: Dense Bag-of-Temporal-SIFT-Words (D-BoTSW, Chapter 2) & Adversarially-Built Shapelets (ABS, Chapter 3).

4.1 *Remote Sensing (Time Series) Data*

4.1.1 *Remote Sensing Data*

Remote sensing data have been widely used in many forms such as images [Tuia et al., 2009; Xie et al., 2008], hyperspectral images [Melgani and Bruzzone, 2004] or time series [Arvor et al., 2011b; Bailly et al., 2016a; Gómez et al., 2016; Gond et al., 2013a]. Remote sensing data have also been used in a wide variety of problems such as agriculture monitoring [Duveiller and Defourny, 2010], vegetation mapping [Xie et al., 2008] and environmental modeling [Dusseux et al., 2013].

4.1.1.1 *Satellite Characteristics and Evolution*

Each satellite is characterized by:

- Its spatial resolution *i.e.* the pixel size of taken images that represents the area at the surface of the earth,
- Its temporal resolution or revisit time *i.e.* the time elapsed between two observations of a given location,
- Its spectral resolution which is the ability of the sensor to discriminate signals of different wavelengths. It depends on the number of spectral bands but also the width of these bands.

While the first sensors were mainly dedicated to meteorology, such as the Advanced Very High Resolution Radiometer (AVHRR) in the 70's, which has a kilometric resolution associated with a wide field of view; sensors evolve to high spatial resolution leading to a larger number of applications. In the 2000's, the sensors spatial resolution with a high temporal resolution reaches 250 to 300 meters, *e.g.* Moderate Resolution Imaging Spectrora-

diometer (MODIS) sensors abroad Terra and Aqua satellites. Projects dedicated to earth observation through satellites with high spatial resolution have thus been developed such as the Landsat program launched by the National Aeronautics and Space Administration (NASA) in 1972. Eight satellites were launched by this program, two of them are still in orbit: Landsat-7 and Landsat-8; they acquire data all over continental surfaces. Landsat-8 has two sensors that have spatial resolutions of 30 meters (visible, Near Infra-Red (NIR), Short-Wave Infra-Red (SWIR)), 100 meters (thermal) and 15 meters (panchromatic) as well as 16 days temporal resolution [1].

4.1.1.2 *Data Limitation*

Producing an accurate and up-to-date land cover map is an important topic for remote sensing applications. Moreover the amount of remotely sensed data is constantly growing because of the new and upcoming satellite platforms available. However, due to the high cost in terms of money and human resources for labelling data through field campaigns or through expert knowledge, only a very limited number of labeled data are available. Additionally, fast changes in landscape and land use would require a never-ending labeling.

4.1.2 *Remote Sensing Time Series Data*

A remote sensing time series is obtained when images of the same given location are acquired at different dates. It is characterized by its temporal resolution, *i.e.* the number of images and the gap between the different acquisitions. Note that vegetation indices (*e.g.* EVI, NDVI) are often prefered to raw values.

Remote sensing time series are often challenging to classify since they combine several characteristics that can perturb the classification. The vegetation and agricultural landscapes significantly vary across Earth resulting in temporal distortions and shifts in time series. Agricultural practices can also cause shifts in amplitude. Due to atmosphere conditions (*e.g.* clouds), remote sensing time series often contain outliers and are often noisy.

4.1.3 *Remote Sensing Time Series Classification*

Many methods have been proposed for the classification of satellite data, for applications such as land cover mapping. However, few focus on the classification of satellite images time series. This can be explained by the lack of reference data (*i.e.* labeled), as well as by the recent availability of

1. https://landsat.usgs.gov/landsat-8

time series with high spatial resolutions.

Since the early 2000s, the high temporal resolution of the Moderate Resolution Imaging Spectroradiometer (MODIS) sensor aboard the TERRA and AQUA satellites allows the monitoring of phenological cycles in order to classify vegetation types. MODIS time series have been used for many applications such as

- **Crop mapping** in [Arvor et al., 2011a; Chang et al., 2007]. Arvor et al. [2011a] use crop mapping in order to evaluate the agricultural intensification in the state of Mato Grosso, in Brazil and to understand its impact. They focus on the different agricultural practices involving three commercial crops (soybean, maize and cotton) planted in single or double cropping systems. Chang et al. [2007] investigate the use of MODIS time series in order to map and monitor United States corn and soybean fields.
- **Forest mapping** in [Gond et al., 2013b], where authors aim at generating a more precise map of forests in the Congo Basin in the context of global changes.
- **Land cover mapping** in [Clark et al., 2010; Hüttich et al., 2009]. The characterization of land cover mapping in all type of ecosystem is indeed of interest for a suitable and sustainable land management purpose.
- **Multi-year agricultural mapping** in [Brown et al., 2013], where authors aim at showing the agricultural landscape of Mato Grosso (Brazil) including the type of practices (single or double cropping).

The first Landsat satellite was launched in 1972, since then the Landsat spatial and spectral resolutions have been improved leading to *high resolution* satellite images time series. The Landsat time series have been used in many applications such as detecting forest cover changes [Hansen et al., 2013], as well as mapping the surface water [Pekel et al., 2016] in the context of global changes.

Gómez et al. [2016] provide a general overview for land cover classification using satellite images time series. Bégué et al. [2018] focus on methodologies that map cropping practices using remote sensing time series, in order to improve the monitoring of cropping practices in the context of food security.

Numerous papers have been published using remote sensing time series data, however data are often unavailable or have been preprocessed by authors, making it impossible to both reproduce the experiments and use the data for further comparison. In the following, we use two datasets of remote sensing time series, available online. These datasets includes time series collected from the Reunion island as well as the Brazilian amazon.

4.2 *TiSeLac Dataset*

In order to bring machine learning and remote sensing communities closer, a classification challenge was proposed by Ienco [2017]. The data used for this challenge corresponds to satellite image time series.

4.2.1 *The challenge*

The purpose of this challenge was to classify a set of satellite image time series collected above the Reunion Island (with a 30m resolution) acquired by LANDSAT-8 sensor in 2014. There are a fixed train and test sets coming from the same satellite images. Therefore, they cover the same time period, and training and testing data have the same distribution.

The dataset uses a set of 23 images acquired in 2014 and provides a total of ten features: seven surface reflectances as well as three complementary spectral indices. The 7 surface reflectances are Ultra Blue, Blue, Green, Red, Near-InfraRed, Short-Wavelength InfraRed (SWIR) 1 & 2. And the 3 computed indices corresponds to Normalized Difference Vegetation Index (NDVI), Normalized Difference Water Index (NDWI) and Brightness Index (BI).

Satellite image time series were labeled using two publicly available datasets: the *Corine Land Cover* (CLC) map from 2012, and the *Registre Parcellaire Graphique* (RPG) from 2014.

- The **Registre Parcellaire Graphique** is a geographical information system allowing the identification of agricultural parcels. It was set up by France in 2002 and it is the only geographic database of this size to be annually updated. Indeed as part of european union common agricultural policy, farmers have to declare which crops they cultivate for each one of their fields as well as the type of cultures (*e.g.* if it is irrigated or not). Agricultural classes information thus comes from the RPG database.
- The **Corine Land Cover**[2] consists of computer aided photo-interpretation of satellite images. The programm was launch in 1985 by the european union. CLC classes that have been considered are the non-agricultural ones (*e.g. forests* or *urban areas*).

The challenge training set consists of a set of 81 714 multi-dimensional satellite image time series with their associated labels and coordinates. The challenge testing set is composed of 17 973 satellite image time series with labels and coordinates as well.

Figure 4.1 shows Reunion island localization on Earth, whereas Figure 4.2 provides an overview of the Reunion Island including the land cover

2. https://www.eea.europa.eu/publications/COR0-landcover

ground truth.

4.2.2 *Dataset*

In order to run all our baseline classification algorithms, we reduce this multivariate dataset to a univariate dataset. Amongst all surface reflectances and radiometric indices, we choose only one index: the Normalized Difference Vegetation Index (NDVI). Indeed the NDVI is a widely used vegetation index *e.g.* for vegetation mapping.

The second step is to define the set of land cover classes to consider. We choose to keep the 9 classes present in the original dataset in order to ensure land cover diversity. The dataset thus gathers the following classes:

1. Urban Areas,
2. Other built-up surfaces,
3. Forests,
4. Sparse Vegetation,
5. Rocks and bare soil,
6. Grassland,
7. Sugarcane crops,
8. Other crops,
9. Water.

We reduce the amount of training and test time series in order to be able to run experiments on more time series classification algorithms. Thus we keep 500 time series per class in the training set (*i.e.* 4 500 training time series), as done in the *Brazilian Amazon* dataset. For testing, we select 5 000 time series from the test set.

The last step consists in removing the coordinates information from the dataset. Figure 4.3 shows per-class average NDVI time series for the reduced version of TiSeLac dataset. Figure 4.4 shows a random selection of five time series for each class.

The dataset can be reconstructed using available data from Ienco [2017] using the first 500 NDVI time series for each class for the training set and the first 5 000 NDVI time series for the testing set. This can be seen as a random selection since the data were already shuffled, as indicated by the coordinates. Moreover, classification rates obtained for the challenge are not expected to be reached since we reduce the original dataset to one index (on 10) and remove the coordinates.

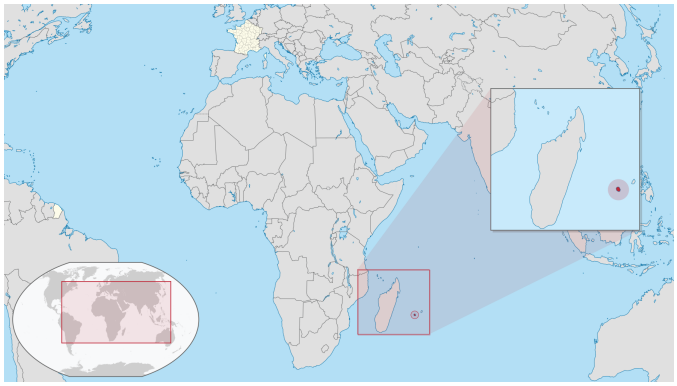
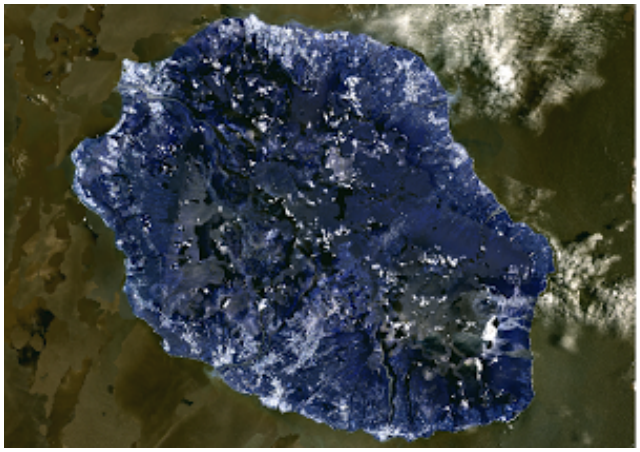


Figure 4.1 – Reunion island localization (from Wikipedia)



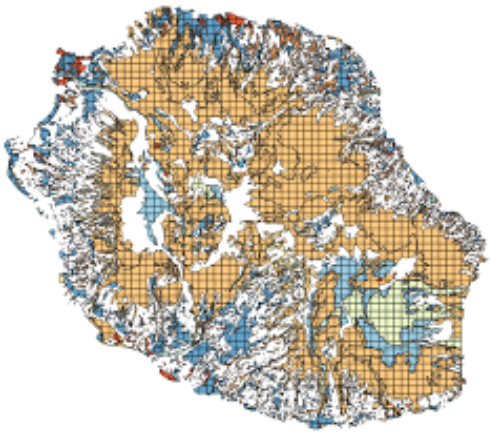


Figure 4.2 – Reunion island and the corresponding land cover classes (from Ienco [2017])

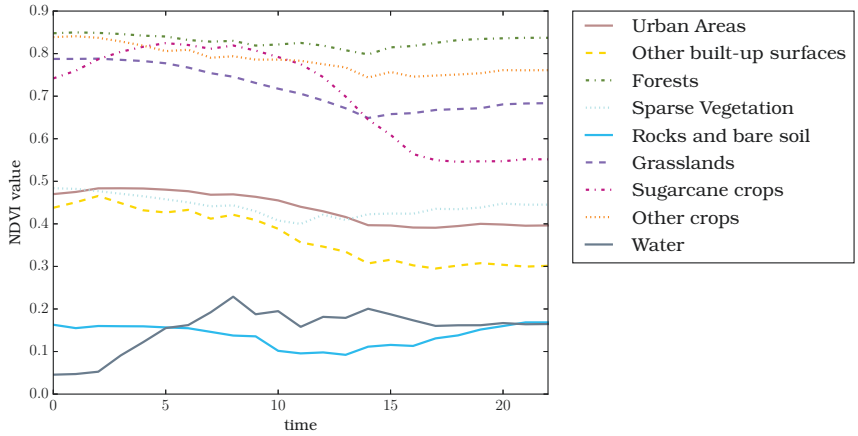


Figure 4.3 – *TiSeLaC* mean profiles per class (NDVI)

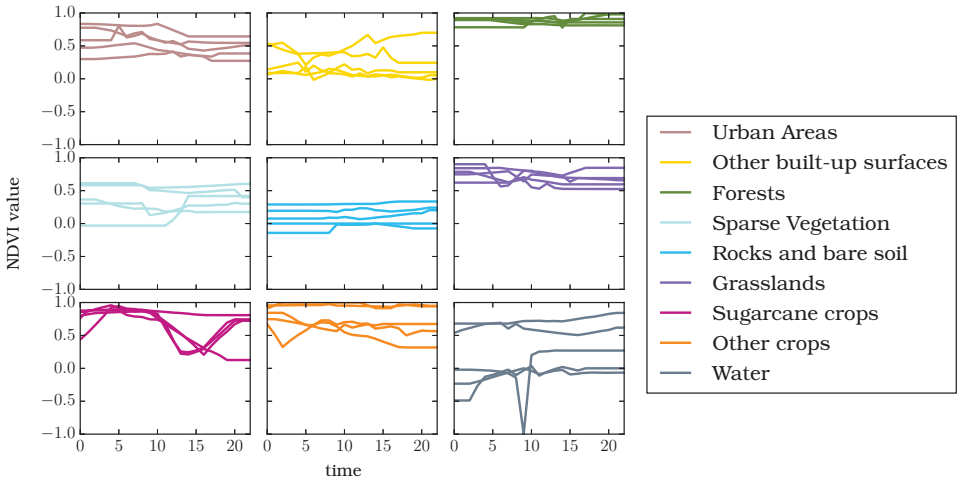


Figure 4.4 – Representation of five randomly selected time series per class from *TiSeLaC* dataset (NDVI)

4.2.3 *Specificities*

4.2.3.1 *Reunion Island*

The Reunion island, which is located in the southern hemiphere, has a tropical climate with two distinct seasons. The *wet* season starts in December and ends in February, whereas the *dry* season runs from May to September. It is a volcanic island with several volcanos such as *Piton des Neiges* and *Piton de la Fournaise*. The island relief is thus very uneven with numerous height variations causing the island to have a large number of microclimates. The Reunion island climate is thus distinguished by a great variability especially due to its particular relief resulting in large disparities. On one hand, disparities come from precipitation differences between the windward coast (east) and the leeward coast (west). On the other hand, disparities are explained by temperature differences between the warmer areas (coastal zones) and colder areas (mountainous zones).

4.2.3.2 *Information on Normalized Difference Vegetation Index*

NDVI values belong to range -1.0 to $+1.0$. Low NDVI values [3] (*e.g.* 0.1 or less) can correspond to rocks, sand or snow. Medium NDVI values represent sparse vegetation such as shrubs, grasslands or crops (low season). Finally, High NDVI values (*e.g.* > 0.6) correspond to dense vegetation such as forests and crops before harvest.

4.2.3.3 *TiSeLaC classes*

The *TiSeLaC* dataset contains both vegetation classes (*e.g. forests*) as well as non-vegetation classes (*e.g. urban areas*). On Figure 4.4, we observe that there is an high intra-class variability both temporally (*e.g. sugarcane crops*) and in amplitude (*e.g. urban areas*). Temporal variability can be explained by different sowing dates, whereas amplitude shifts might be due to altitude or climate differences.

Visually the most distinguishable class is the *sugarcane crops*. Indeed *sugarcane* time series can be easily identify thanks to their distinctive NDVI profiles. On the contrary *forests*, *other crops* and *grasslands* have very similar profiles (same observation for *urban areas*, *sparse vegetation* and *other built-up surfaces*).

The specificities of the different classes are listed below:

1. **Urban Areas**. Reunion island cities are not as dense as european cities [Rivière et al., 2013]. Moreover there exists many green spaces in

3. Information collected from Remote Sensing Phenology website: https://phenology.cr.usgs.gov/ndvi_foundation.php

the cities such as trees rows to provide shade and cooler areas. The greenness density of a pixel can influence its associated NDVI values.

2. **Other built-up surfaces** could refer to greenhouses, however this is an assumption. Indeed greenhouses are often established in green areas, thus it could explain the larger intra-class variability compared to *urban areas*.
3. **Forests**. Due to the tropical climate, forests NDVI values do not fluctuate. Time series representing forests on Reunion island thus have high but stable NDVI values.
4. The **Sparse Vegetation** class regroups pixels that do not belong to any of the other classes. This class thus regroups a variety of profiles with intermediate NDVI values.
5. **Rocks and bare soil** will have by definition low NDVI values (closed to or below 0). For a particular pixel the NDVI value is expected to have little variation.
6. The **Grassland** class gathers together grazed and mowed grasslands. Depending of the irrigation, the regrowth speed may vary resulting in different time series profiles.
7. **Sugarcane crops** time series have the most distinguishable profiles since NDVI values drop after harvests. Sugarcane crops represent $\sim$60% of the cultivated area on Reunion island, the harvesting period of sugarcane crops spreads over around a month. [Denize, 2015]
8. The **Other crops** class is formed by many crops (*e.g.* pineapple and bananas crops) as well as orchards (*e.g.* lychees and mangos). This diversity of crops results in a high intra-class variability for this class.
9. The **Water** has a high intra-class variability that can be explained by the turbidity of the water. The turbidity of the water is a measure of the water clarity which can perturb NDVI values.

In order to distinguish the different *TiSeLaC* classes, we can not rely on distinctive features. Indeed different classes have similar profiles but can be discriminated thanks to their mean NDVI values, *e.g. forests* and *grasslands*.

4.3 *Brazilian Amazon Dataset*

Mapping croplands is of primary interest to estimate cultivated areas and agricultural production or to qualify the agricultural practices and their potential impacts on the environment. This assessment is especially true for tropical areas. Indeed, these regions which concentrate major global natural resources are expected to supply a large proportion of increasing production demand at global scale.

Since the early 2000s, the temporal resolution of the MODIS sensor onboard the Terra and Aqua satellites allows the monitoring of phenological cycles in order to classify vegetation types. Such approach appears to be especially relevant in tropical areas where vegetation index time series enable one to limit the constraints related to high cloud cover rates. In the Brazilian amazon, many studies based on MODIS vegetation index time series were carried out to map croplands and crop types [Chang et al., 2007; Clark et al., 2010; Hüttich et al., 2009; Wardlow and Egbert, 2008]. However, in a context of climate change and crop expansion to other Brazilian states, agricultural calendars are expected to evolve so that it is necessary to implement new methodologies to process remote sensing time series.

In the following, we propose a dataset which time series were extracted in the Brazilian amazon and represent MODIS vegetation index time series corresponding to diverse cropping practices.

4.3.1 *Study Area*

The study area is located in the state of Mato Grosso, in the southern Brazilian amazon, as shown on Figure 4.5. This state has suffered dramatic land use changes since the 1970s due to the rapid progress of an agricultural frontier. This frontier was the result of efficient public policies to encourage people to settle the large and plane areas of native savannas and forests. It led to the establishment of a powerful agricultural sector dedicated to the production of commodities such as soybean, maize or cotton in large-scale farms. Indeed, Mato Grosso is nowadays one of the world's biggest producer of these crops. As a consequence until the mid-2000s the agricultural sector was especially criticized for its severe impacts on forests since Mato Grosso has long been known for its high deforestation rates.

Nonetheless, deforestation decreased for ten years and the historical extensive agriculture model is being replaced by intensive agriculture practices [Macedo et al., 2012]. For example, double cropping agriculture systems have been generalized in Mato Grosso (from 35 to 62% of the net cropped area [Arvor et al., 2012]) and especially concern the cultivation of soybean followed by maize, millet or cotton. In this case, soybean is sown first between late September and late November and cultivated until early January and late March. When soybean is harvested, maize and cotton are sown to be harvested between late May (for maize) and mid-July (for cotton).

Although the *average* agricultural calendar is well known in this region, slight differences may occur at regional scale due to rainfall variability [Arvor et al., 2014] or logistics issues. Indeed, for largest farms (up to 35,000 ha), the sowing or harvesting period can last up to three months. This issue is important to emphasize since it implies a large temporal variability in vegetation index time series for pixels of a same crop class thus affecting

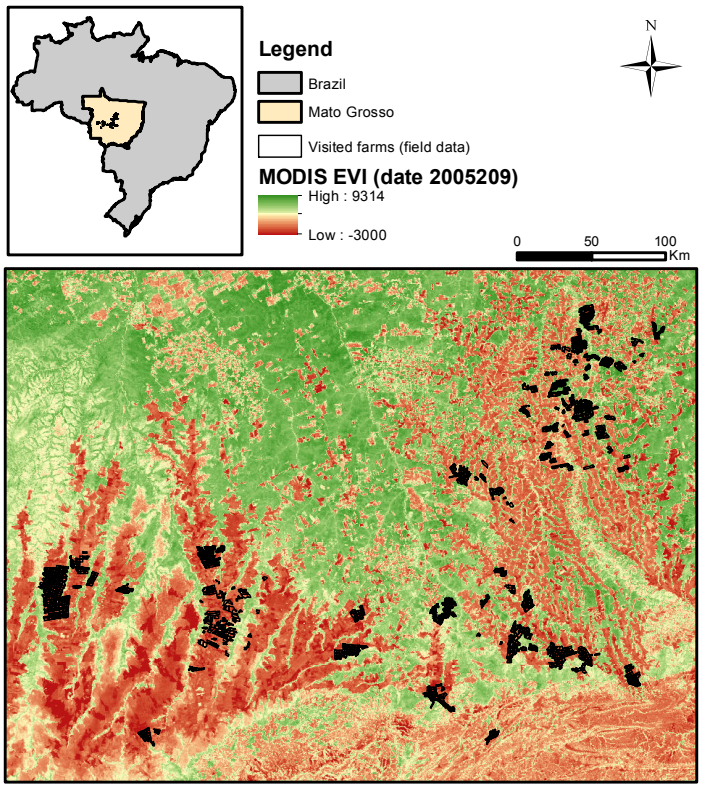


Figure 4.5 – Map of the study area with field data location.

any attempt to classify these classes.

4.3.2 *Dataset*

We extract two years of MODIS vegetation index time series corresponding to the cropping periods 2005-2006 and 2006-2007. The dataset is thus made of 46 MODIS images from July 2005 to July 2007, *i.e.* 23 MODIS images per year. This study period was chosen to match with field data in order to enable statistical validation of the method. The vegetation index time series refer to Enhanced Vegetation Index (EVI) of the MOD13Q1 product (tile h12v10), which are made freely accessible by the *NASA Land Processes Distributed Active Archive Center*[4]. The EVI was created in order to improve the quality of Normalized Difference Vegetation Index (NDVI) products. The computation of EVI is similar to the NDVI one, however EVI improves on NDVI's spatial resolution and has a higher sensibility which enables it to better differentiate heavily vegetated area (where NDVI saturates)

4. https://lpdaac.usgs.gov/

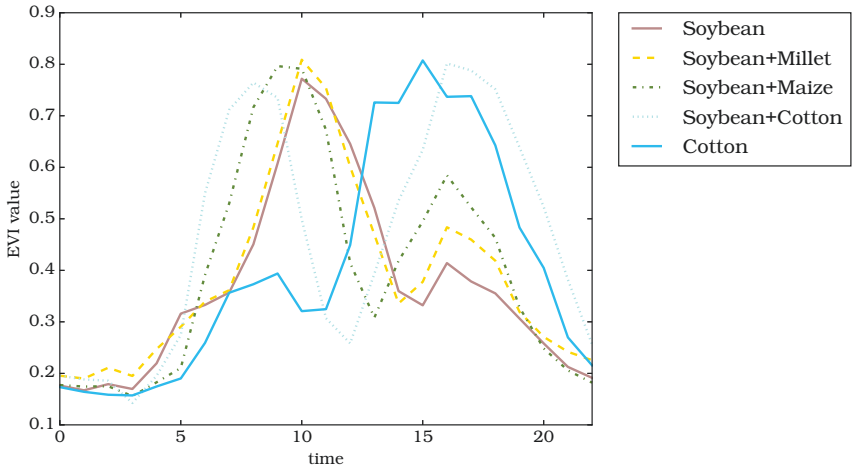


Figure 4.6 – *Brazilian Amazon* – Mean profiles per class (EVI)

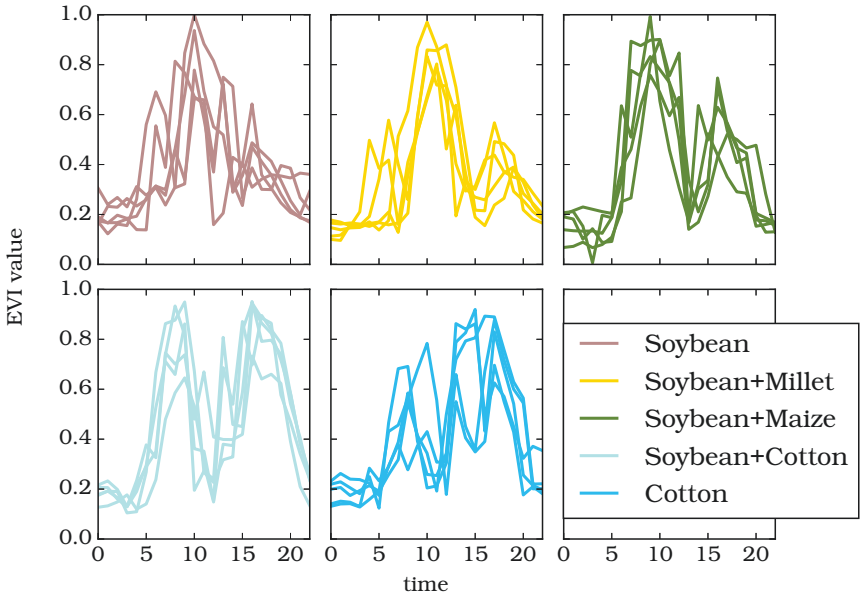


Figure 4.7 – *Brazilian Amazon* – Five randomly selected time series per class

[Huete et al., 2002]. When missing data occurs (*e.g.* due to meteorological conditions), we performed linear extrapolation using closest available data points before and after the missing data.

The field data used for validation is the same as in [Arvor et al., 2011a]. It is based on an extensive field campaign carried out in 2007 to collect validation data about crop type, crop yield and sowing and harvesting dates for two harvests (2006 and 2007). Seventy-six farms were visited and mapped in thirteen municipalities from the two main agricultural regions in Mato Grosso, *i.e.* along the BR163 road and the Chapada dos Parecis. Five crop classes were identified:

1. Soybean,
2. Soybean + Millet,
3. Soybean + Maize,
4. Soybean + Cotton,
5. Cotton.

Classes 1 and 5 refer to single cropping practices whereas classes 2, 3 and 4 refer to double cropping practices. For each class, 500 pixels were randomly selected each year. Thus, we built two datasets (each of them composed of 2500 pixels described by time series of length 23) on two different years in order to assess the method's robustness by training and validating on different years. The training set corresponds to the period 2005-2006 whereas the testing set to the period 2006-2007.

Figure 4.6 shows per-class average EVI time series for cropping period 2005-2006. Figure 4.7 shows a random selection of five time series for each class in order to illustrate high intra-class temporal variability as well as amplitude variability. The dataset is available online [5].

4.3.3 *Specificities*

Due to the climatic conditions in the state of Mato Grosso in Brazil different agricultural practices can be observed: single cropping and double cropping. These practices – as well as with large farms and logistics issues – generate a specific agricultural calendar.

Indeed, to ensure two harvests (Soybean and {Cotton, Maize or Millet}) in the same year, it is necessary for farmers to be organized. Cotton, maize and millet do not have the same phenological cycle: cotton has the longest one followed by maize ; *i.e.* cotton crops require more time from sowing to harvesting than both maize and millet crops. It is thus necessary to start sowing the soybean that will be followed by cotton earlier than the one to be

5. https://github.com/a-bailly/time_series_data/tree/master/brazilian-amazon

followed by maize, since the soybean followed by cotton need to be harvest first. The same reasonning apply to soybean with maize and soybean plus millet. Moreover since millet is usually not planted for commercial use but rather for soil cover, it is logical to be sown last (for double cropping) as it will not be harveted.

Concerning cropping practices, single soybean can be sown later than double cropping soybean since there will be only one harvest. After single cropping soybean is harvested, it is possible to observe that the vegetation grows again on Figure 4.6. This can be explained by the fact that the rainy period is not over. Cotton cultivated in single cropping is sown later than soybean for a practical reason: cotton need to be harvested during the dry season *i.e.* after the end of the rainy period.

Knowledge on remote sensing data can help building more specific approaches. For the *Brazilian Amazon*, we show that there exists a specific agricultural calendar that can bring more information to help properly classify properly the time series.

4.4 *TiSeLaC Dataset versus Brazilian Amazon Dataset*

The *TiSeLaC* dataset is more challenging than the *Brazilian Amazon* one for the following reasons:

1. It contains more classes.
2. The profiles proximity of several classes makes the classification more challenging for *TiSeLaC* than for *Brazilian Amazon*.
3. The lack of distinctive features in the data makes it more difficult to extract knowledge from *TiSeLaC*, making it harder to properly classify the time series, especially for feature-based algorithms.

We can thus expect better classification performance on *Brazilian Amazon* than on *TiSeLaC*.

4.5 *Experiments on Remote Sensing Time Series datasets*

In order to follow the UEA / UCR dataset format, we split each dataset into a training and a testing sets, as detailed in Table 4.1 (500 training time series per class for both). The experiments are carried out using only training data during the training step and error rates are reported on the test set in Table 4.2.

	Brazilian-Amazon_EVI	*TiSeLac_NDVI*
# Classes	5	9
Length	23	23
# Training Time Series	2500	4500
# Testing Time Series	2500	5000

Table 4.1 – Remote sensing time series datasets in numbers

4.5.1 *Algorithms*

We run the experiments on the following algorithms:

- The available implementations of **Bag-of-Patterns**, **BOSS**, **DTW-NN**, **LTS**, **SAX-VSM**, **SMTS** from [Bagnall et al., 2016] [6].
  Note that we slightly modifiy the LTS algorithm since we take a maximum of 10 000 random sub-series to initialize shapelets instead of all the sub-series of the training set. This modification has low impact on performance, however it requires less computational space and less time for large datasets.
- Our implementation of **ABS** which is available on GitHub [7]. As describe in Chapter 3, we initialize the shapelets using *either* the entire set of sub-series (if $< 10\,000$) *or* a maximum random set of 10 000 sub-series. We also use recommended LTS parameters:
  - The regularization parameter $\lambda_w = 0.01$,
  - The number of scales $R = 3$,
  - The percentage of series length to 0.2,
  - The number of iterations to 600.

  Finally, we use cross-validation to learn the parameter for the adversarial perturbation inside the set $\{0.001, 0.01, 0.1\}$.
- Our implementation of **D-BoTSW** which is available on GitHub [8]. Note that in order to reduce the computation time, we set the parameters k (k-means) and C (SVM) *respectively* to 1024 and 1.
- For **ED-NN**, we use the python *KNeighborsClassifier* function from package `sklearn.neighbors` associated with *euclidean* distance (from package `scipy.spatial.distance`).

6. www.timeseriesclassification.com – Corresponding to `229d523` bitbucket commit
7. https://github.com/a-bailly/adversarially_built_shapelets
8. https://github.com/a-bailly/dbotsw

	Brazilian-Amazon (EVI)	*TiSeLaC* (NDVI)
ABS	**0.285**	**0.415**
BoP	0.501	0.610
BOSS	0.467	0.474
D-BoTSW	0.295	0.583
DTW-NN	0.457	0.457
ED-NN	0.354	0.438
LTS	**0.285**	0.428
SAX-VSM	0.570	0.717

Table 4.2 – Average error rates on remote sensing time series datasets (on 6 runs)

4.5.2 *Results Interpretation*

From Table 4.2, we see that for both datasets, shapelet-based methods (LTS & ABS) perform nicely. We can also notice that in the top 4 methods there is also a feature-based method (BOSS for *TiSeLaC* and D-BoTSW for *Brazilian-Amazon*) as well as the simple ED-NN.

Learning Time series Shapelets and Adversarially-Built Shapelets are both able to identify a set a sub-series (shapelets) that characterize well the different classes thus that lead to a small amount of errors for these two remote sensing time series datasets.

ED-NN performs well on these two datasets, it might come from the fact that the time series length is relatively small (only 23 datapoints per time series). On the contrary, we observe that DTW-NN obtained a larger number of errors than ED-NN on both datasets. We show that the *Brazilian Amazon* data respect an agricultural calendar, where time series of the same class can have a small temporal shift due to practical reason. Aligning time series might seem like a good idea since it can enable one to take in account this temporal variability. However, it is necessary to consider the maximum temporal shift of the time series of the same class, which DTW-NN does not.

Finally, the feature-based methods BOSS and D-BoTSW often achieve good performance. However, the length of the time series of these two remote sensing datasets is relatively small and can affect their performances. Nonetheless, we have to keep in mind that new satellites such as Sentinel 1 & 2 with a smaller revisit time provide longer remote sensing time series, which enable these method to build their model on more characteristics, *i.e.* to have a more precise representation. All four BOSS, D-BoTSW, LTS and ABS can thus expect to achieve better performance on other remote sensing time series datasets with longer time series.

4.5.3 *Study of Robutness Methods for Remote Sensing Data Specificities: Growing Amount of Data*

We propose to further experiment in order to evalutate the impact of the number of time series per class on the performance of the different algorithms. We use a subset of the training set in order to create increasingly large training sets (*respectively* 20, 50, 100, 200 & 500 training time series per class) that we run on ABS, BoP, BOSS, D-BoTSW, ED-NN, LTS and SAX-VSM algorithms. Figures 4.9 and 4.8 summarize the results *respectively* for datasets *TiSeLaC* and *Brazilian-Amazon*. For each size, we learn on 6 different training sets and compute the error rates on the same testing set (*i.e.* the previous testing set with 5 000 time series). The average error rates are given in Table 4.2. On Figures 4.9 and 4.8, we can observe both the average error rate (continuous line) as well as minimum and maximum error rates obtained on the different training sets. Note that for the *Brazilian Amazon* we only have 500 training time series which results in a small / zero variance on Figures 4.9 on a *large* number of time series (200 and 500 time series per class) ; for the *TiSeLaC* dataset we randomly select the training time series in the 81 714 original training time series.

We observed in Table 4.2 that BoP and SAX-VSM have the highest error rates in both cases. A possible explanation comes from the fact that both BoP and SAX-VSM reduce the data dimensionality. However, for our applications, it might be a problem considering that the time series have a relatively small length (only 24 datapoints). These two approaches are thus unlikely to perform well on remote sensing datasets with time series having few datapoints.

We experiment on two other feature-based methods: BOSS and D-BoTSW. Their features are more robust and lead to better performance than BoP and SAX-VSM. However they also suffer from the low dimensionality of the data, *i.e.* from the small time series length. We expect to obtain better performance on remote sensing time series with higher temporal resolution for both BOSS and D-BoTSW, since they rely on the notion of neighborhood. Note that the difference between minimal and maximal error rate is larger for BOSS, which suggest that D-BoTSW is more robust to variations.

4.5.4 *Descriptive Features*

In order to illustrate both our proposed algorithms: ABS & D-BoTSW, we have generated some shapelets and codewords for the *Brazilian Amazon* dataset which has more distinguishable features than the *TiSeLaC* one.

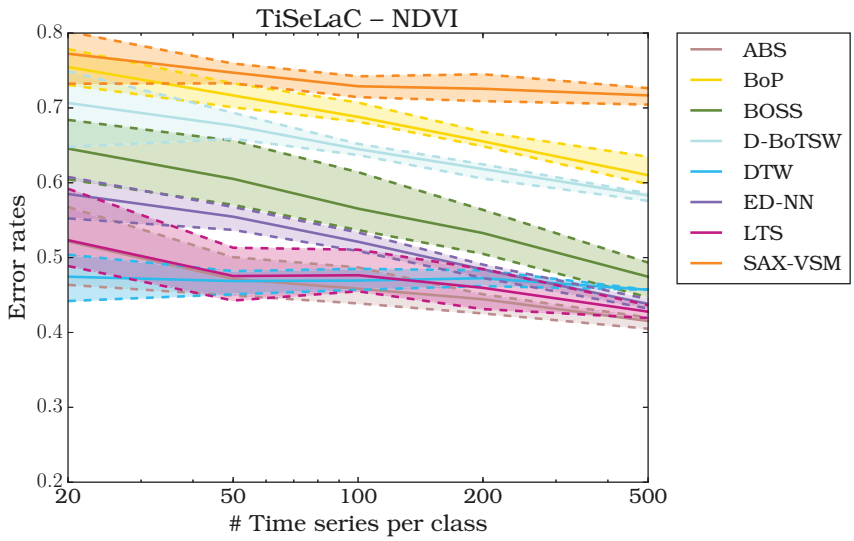


Figure 4.8 – TiSeLaC – Evolution of error rates for an increasing number of training time series

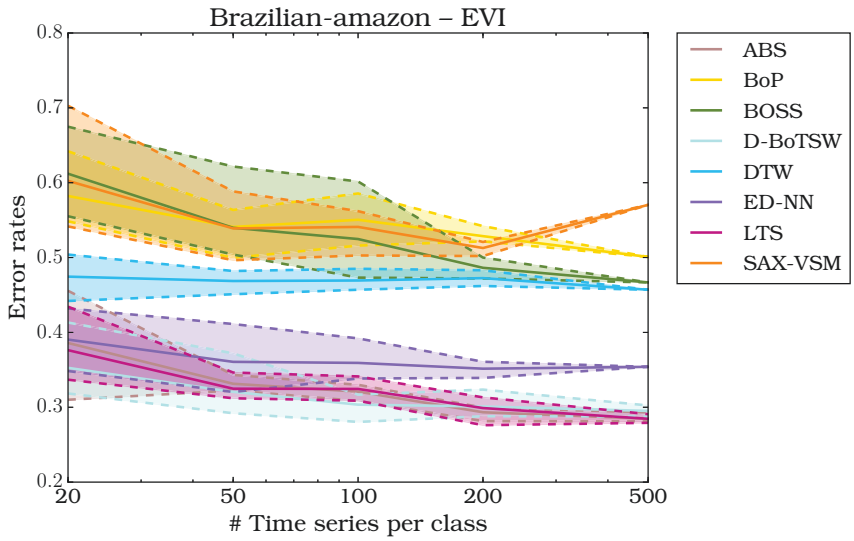


Figure 4.9 – Brazilian Amazon – Evolution of error rates for an increasing number of training time series

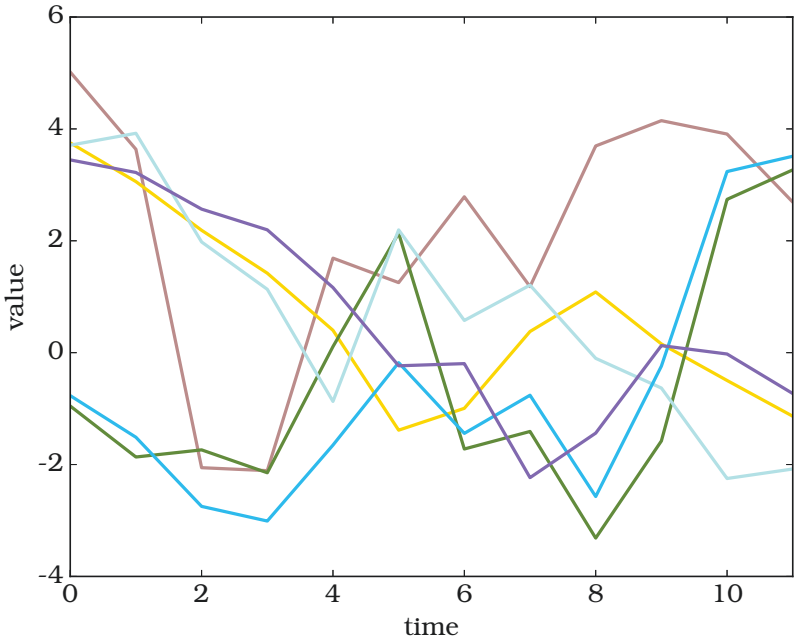


Figure 4.10 – Brazilian Amazon – The 6 shapelets generated by ABS (of length 12)

4.5.4.1 *Adversarially-Built Shapelets*

In order to illustrate ABS algorithm, we have generated a set of shapelets for the *Brazilian Amazon* dataset, as shown on Figure 4.10, we also provide associated weights and bias in Table 4.3. For readibility,

1. We fixed the number of shapelets to 6 instead of computing it using Grabocka's formula.
2. We run this experiment on default parameters (# of scales: $R = 3$ and percentage of time series length $= 0.2$), however we only plot the sufficiently long shapelets (corresponding the third scale, with 12 datapoints) for readibility.

The number of classification errors is slightly higher than the one obtained with more shapelets since we obtained an error rate of 0.306 instead of 0.285 previously.

It is possible to associate a shapelet with a specific class, such as

- The ■ shapelet with the *Soybean+Cotton* class. Indeed, there are two consecutive large peaks which corresponds to the *Soybean+Cotton* profile.
- The ■, ■ and ■ shapelets represent a large peak followed by three smaller ones.
    - The ■ shapelet has the largest second peak between the three

Shapelets \ Classes	Soybean	Soybean +Millet	Soybean +Maize	Soybean +Cotton	Cotton
■	0.458	-0.145	3.063	**-4.620**	-0.598
■	-2.967	**-2.412**	-1.246	1.602	4.106
■	-0.592	-0.443	2.529	3.937	**-2.903**
■	-0.447	-2.391	**-3.856**	2.956	2.215
■	-1.045	-1.105	1.62	3.828	**-3.855**
■	**-4.545**	-1.817	-0.421	0.115	4.290
Bias	0.643	0.211	-0.545	-0.505	-1.525

Table 4.3 – Weights associated with the shapelets represented on Figure 4.10 for each class. We indicate the most influential weight in bold *i.e.* the weight which is the more related to a class. The weights with the largest absolute values correspond to most informative features. Note that due to the construction of the model, the most influential weights are not the largest positive ones but the largest negative ones.

considered shapelets ; thus we assume that this shapelet is related to the *Soybean+Maize* class.

- The second peak of the ■ shapelet is a bit smaller than the ■ one ; we can make the following hypothesis: the ■ shapelet is associated with the *Soybean+Millet* class.
- Finally, the ■ has the smallest second peaks, our assumption is that it is related with the *Soybean* class.

- Both ■ and ■ shapelets correspond to the end of a small / medium peak followed by a large one, which can correspond to *Cotton* class. In Figure 4.7, we observe that the different classes have a high intra-class variability, these two different shapelets representing the same classes can be seen as a good illustration of this variability.

Thanks to the weights associated with each class and each shapelet (see Table 4.3), we are able to check our hypotheses. For each class we hightlight the most influential shapelet(s) which confirmed our previous assumptions *e.g.* the class *Soybean* is associated with the ■ shapelet.

4.5.4.2 *Dense Bag-of-Temporal-SIFT-Words*

In order to illustrate D-BoTSW algorithm, we have generated a set of codewords for the *Brazilian Amazon* dataset (see Figure 4.11) as well as corresponding histogram per class (see Figure 4.12). For a better visualization we set the parameters to:

- The number of points per block a to 2,
- The number of blockss n_b to 6,
- The number of codewords k to 16,

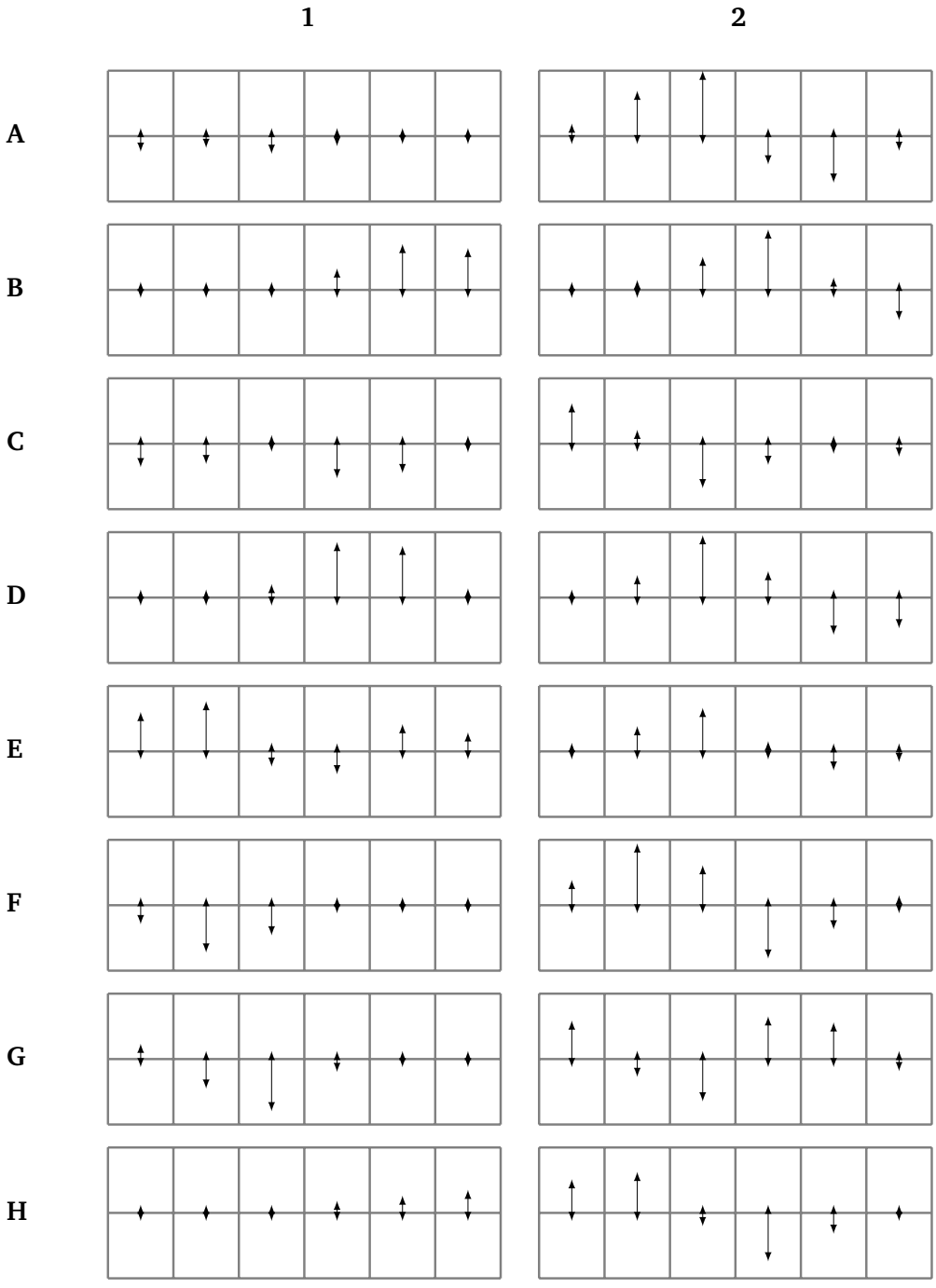


Figure 4.11 – Brazilian Amazon – The 16 generated features from D-BoTSW algorithm

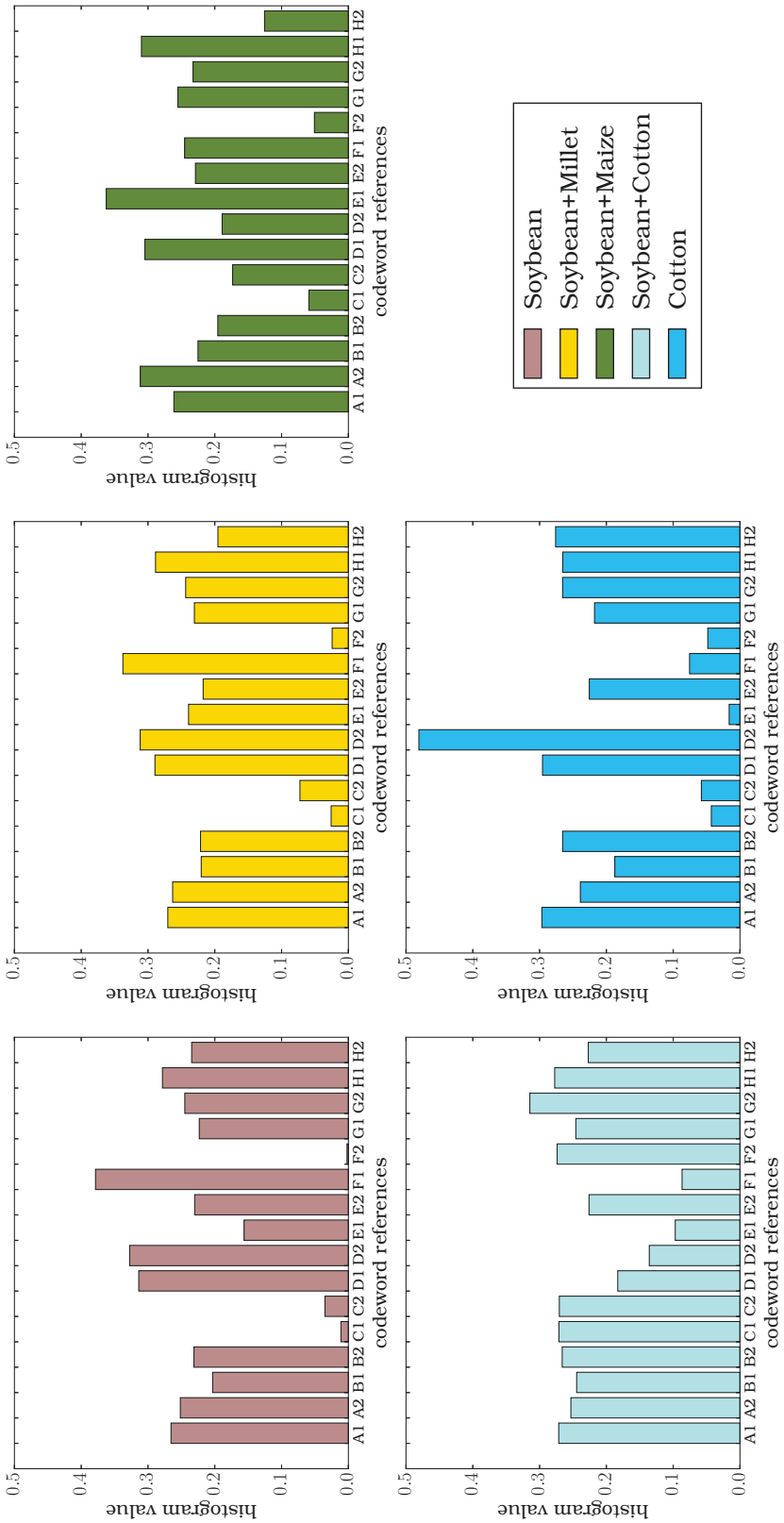


Figure 4.12 – Brazilian Amazon – Average histogram per class (D-BoTSW)

- And the SVM parameter C_{SVM} to 1.

The error rate obtained with this setup (0.316) is approximatly the same than the one obtained with the cross validation (0.295). In the following, we will refer to a specific codeword by its coordinates, *i.e.* one letter to specify the line and a number for the column, as indicated on Figure 4.11. For example, the $D1$ codewords has two long arrows pointing up for blocks four and five otherwise small arrows. Each codeword represents a total amount of 12 datapoints, *i.e.* around half of the total time series. Note that examples of time series and their corresponding codewords can be found in Figure 2.6(c).

Several codewords show the presence of one *peak* such as $A2$, $B2$ or $D2$, others suggest even two *peaks e.g.* $E1$ and $G2$. It is possible to observe that the second peak of $E1$ has much smaller values that the second peak of $G2$. The histogram value associated with $E1$ should thus be larger for class *Soybean+Maize* than for the other classes ; and those with $G2$ larger for class *Soybean+Cotton*. In order to check this hypothesis, we generated the average histogram per class for the *Brazilian Amazon* training set, on Figure 4.12. The larger the values, the more important the codeword is detected in the class. We can observe that

- Class *Soybean+Maize* has the largest value for the codeword $E1$, *i.e.* this codeword is representative of this class. On the contrary, classes *Soybean+Cotton* and *Cotton* has low values for this codeword.
- The class that has a highest value for the codeword $G2$ is the *Soybean+Cotton* class.

Consequently, our hypotheses on both codewords correspond to the reality. It is possible to find the codewords that are the most associated with specific classes logically. D-BoTSW can thus be used for results interpretation or to check data hypotheses.

4.6 *Conclusion*

In this chapter, we provide two time series datasets available online that we recommend to use in association with the well-known UCR / UEA database to evaluate the performance of any algorithm. Indeed, the variety of problems represented by the UCR / UEA can be enlarged, thanks to these datasets that contains remote sensing time series. We also provide an analysis of the data in order to better understand the nature and the specificities of these datasets.

We perform a set of experiments on these datasets using time series classification algorithms. Then we show the impact of a growing amount of time series and algorithm stability using different training sets. Finally, we depict both ABS and D-BoTSW through shapelet and feature visualizations

for a better understanding of these two algorithms.

Moreover, for some applications, we have identify some additional information that can help classifying data. The creation of algorithms to take into account these informations (*e.g.* by detecting the beginning of cropping period) can help improve current performances.

Time revisit and/or spatial resolution are increasing for new satellites. We can differentiate satellites with high spatial resolution but low temporal resolution (*e.g.* SPOT-6, SPOT-7) and satellites with lower spatial resolution but high temporal resolution (*e.g.* Sentinel-1, Meteosat). Sentinel-2 sensors combine both high spectral and spatial resolutions (10, 20 or 60m) with a high temporal resolution, since they cover the entire Earth surface every five days [Malenovskỳ et al., 2012]. Higher spatial resolution leads to purer pixel and higher temporal resolution leads to longer time series. Both purer pixels and longer time series mean that feature-based method performances will be improved.

CONCLUSION AND PERSPECTIVES

Results Summary

Time series data can be found in many domains resulting in an explosion of interest in mining time series data in the last two decades. A wide number of algorithms have already been proposed to classify time series and for many applications.

In this thesis, our main purpose was to propose new methodologies for time series classification. Indeed, there is no classifier that is the best on all problems; it is therefore important to propose new algorithms, where a list of dataset properties, for which these algorithms are adapted, has been identified. We choose to focus on methodologies, whose learned models can be interpreted, and introduce two time series classification algorithms that outperform most state-of-the-art baselines. One of the main reason to focus on interpretable methods is the importance of results interpretation and analysis for remote sensing applications in order to better understand the dynamics related to the transformations of our planet. Source code as well as data are available online for reproducibility.

We first proposed an algorithm that builds on two well-known and powerful methods: local features which are densely extracted from time series using a SIFT-based approach and a global representation of time series using these features which is produced using the Bag-of-Words technique. We identify both strengths and weaknesses of our Dense Bag-of-Temporal-SIFT-Words algorithm (D-BoTSW). Amongst weaknesses, we have identified that it is important to have enough points for each time series. Due to the Bag-of-Words representation that causes information loss during the quantization step, D-BoTSW might not perform well on datasets where it is difficult to distinguish the different classes. Amongst strengths, D-BoTSW is robust to noise and exhibits high accuracy and high performance on many datasets. Experiments, conducted on a wide variety of time series datasets, show that D-BoTSW significantly outperforms nearly all considered standalone classifiers. Finally, a visualization of the model, that includes the codewords and the histogram representations of time series, is possible in order to check hypotheses made on the data.

We then propose an enhancement of an already existing algorithm: Learning Time series Shapelets (LTS). In order to build our improved algorithm, we first prove that LTS can be seen as an instance of a Convolutional Neural Network (CNN). Then, based on the fact that adversarial training can

be used to build more robust models for image classification, we demonstrate that this assumption is also valid for LTS. Finally, we implemented the Adversarially-Built Shapelets algorithm (ABS) in order to also prove experimentally that ABS performs better than LTS. Note that with adversarial training, the performance is only slightly improved (thanks to more robust shapelets), however the improvment is statistically significant. Both LTS and ABS are interpretable algorithms. Indeed, characteristics sub-series (*i.e.* shapelets) are used in order to discriminate the classes. These shapelets can be used in association with the corresponding logistic regression weights in order to determine the most important feature(s) for each class.

Amongst drawbacks, the current learning step of ABS implementation might be seen as not fast enough, however a fast implementation based on CNN libraries is planned. ABS has the same advantages than LTS that can be found in [Grabocka et al., 2014], such as a fast prediction step. Both ABS and LTS focus on discriminative sub-series in order to predict the class labels. This is a strength point considering that time-series data often exhibit inter-class differences in terms of sub-series. Shapelets also offer interpretable features. Finally, ABS exhibit competitive performance *w.r.t.* time series classification baselines.

We tested our algorithms on datasets from many domains (using UEA / UCR database) in order to compare them to existing time series algorithms. However one of our purpose was to design classifiers adapted to satellite images time series. Our last contribution was to study how time series algorithms can perform on this data as well as to show how our algorithms can be used in this context. Namely, we benchmarked many time series classification algorithms on two remote sensing datasets, for which we detailed the specificities. These experiments show that shapelet-based methods are well-suited for this kind of data. Feature-based methods also show promising performance, which we expect to be improved in the future thanks to new and upcoming satellites with higher temporal and spatial resolutions. Finally, we provide a detailed explanation of the interpretability for both D-BoTSW and ABS. We believe that interpretability will be an incentive for remote sensing scientists to use these methods.

Perspectives

Our work can be seen as a step towards bringing closer remote sensing and machine learning communities. The introduction of methodologies able to use additional knowledge (*e.g.* pixel coordinates, calendar information) present in remote sensing data is a direction that is worth being explored. In order to propose new algorithms supporting massive amounts of data (*e.g.* Sentinel-2 satellite gathers over 10 trillions time series), it is necessary

for both communities to work together. Moreover, with the increasing number of spectral bands acquired by sensors, multivariate time series will be in the future an important topic for the time series classification community.

For satellite data it makes perfect sense to consider multivariate data since satellite sensors acquire several spectral bands, hyperspectral sensors even acquire hundreds of them. Each spectral band can bring different information that can help for the classification. Both D-BoTSW and ABS could be easily adapted to multivariate time series. It is possible for D-BoTSW to generate different codebooks (one per dimension), the new time series representation will thus correspond to the histograms concatenation. Experiments using only a sub-part of the spectral band could be easily done (by removing the corresponding histograms) in order to detect the most discriminative bands for a specific application. Concerning ABS, it is possible *e.g.* to generate shapelets for each dimension instead of different length shapelets. As for D-BoTSW, this approach allows an analysis of the discriminative power of spectral bands. Considering that a hyperspectral sensor deals with a large number of bands over a continuous spectral range, it is crucial for the time series classification community to develop more multivariate solutions for remote sensing applications. These solutions should allow results interpretability such as providing information on the most discriminative spectral bands

SIFT features for time series used in Dense Bag-of-Temporal-SIFT-Words already serve as a basis for an algorithm that considers the temporal order of the features [Tavenard et al., 2017]. Future work on D-BoTSW can focus on considering more information such as pixel coordinates for satellite image time series in order to take into account the neighboring time series. Coordinates information can help for the prediction between similar classes. Indeed, neighboring pixels are often part of the same parcel or field, thus it can help detect misclassify time series. For instance, if a pixel is labeled as maize but all its neighbors belong to the coton class, one may assume that this pixel is misclassify and should be also labeled as coton. Finally, D-BoTSW can also be included into easily-extendable frameworks such as ensemble classifiers dedicated to time series classification (*e.g.* COTE).

Our Adversarially-Built Shapelets algorithm shows very promising classification results. We show that LTS is an instance of a CNN, it might thus be interesting to evaluate the impact of the CNN setup. The activation function, the impact of the number of layers, the shape and size of the filters (*i.e.* of the shapelets) all play an important role in the learning process of a CNN. Further investigation is required in order to better understand their influence on the model, which can lead to further improvment of the method. Moreover, several methods have been proposed to avoid overfitting for CNNs (such as dropout and early stopping), an evaluation of their impact on the LTS & ABS algorithms is relevant and might help building better algorithms. Moreover, a fast implementation of both LTS & ABS using CNN

libraries must be implemented in the future to promote these methods. Finally, many scientific papers are currently based on CNNs and adversarial examples, which can lead to further improvment of LTS & ABS or even new shapelet-based approaches.

BIBLIOGRAPHY

Aghabozorgi, S., Seyed Shirkhorshidi, A., and Ying Wah, T. (2015). Time-series clustering - a decade review. *Inf. Syst.*, 53(C):16–38.

Agrawal, R., Faloutsos, C., and Swami, A. N. (1993). Efficient similarity search in sequence databases. In *Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms*, FODO '93, pages 69–84. Springer-Verlag.

Arvor, D., Dubreuil, V., Ronchail, J., Simões, M., and Funatsu, B. M. (2014). Spatial patterns of rainfall regimes related to levels of double cropping agriculture systems in Mato Grosso (Brazil): Spatial patterns of rainfall regimes in Mato Grosso. *International Journal of Climatology*, 34(8):2622–2633.

Arvor, D., Jonathan, M., Meirelles, M. S. P., Dubreuil, V., and Durieux, L. (2011a). Classification of MODIS EVI time series for crop mapping in the state of Mato Grosso, Brazil. *International Journal of Remote Sensing*, 32(22):7847–7871.

Arvor, D., Jonathan, M., Simoes, M., Dubreuil, V., and Durieux, L. (2011b). Classification of modis evi time series for crop mapping in the state of mato grosso, brazil. 32:7847–7871.

Arvor, D., Meirelles, M., Dubreuil, V., Bégué, A., and Shimabukuro, Y. E. (2012). Analyzing the agricultural transition in Mato Grosso, Brazil, using satellite-derived indices. *Applied Geography*, 32(2):702 – 713.

Bagnall, A., Lines, J., Bostrom, A., Large, J., and Keogh, E. (2016). The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, Online First.

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.

Bagnall, A., Lines, J., Vickers, W., and Keogh, E. (2017). The UEA & UCR Time Series Classification Repository. www.timeseriesclassification.com.

Bailly, A., Arvor, D., Chapel, L., and Tavenard, R. (2016a). Classification of MODIS Time Series with Dense Bag-of-Temporal-SIFT-Words: Application to Cropland Mapping in the Brazilian Amazon. In *IEEE International conference on Geoscience and Remote Sensing Symposium (IGARSS)*, pages 2300–2303, Beijing, China.

Bailly, A., Malinowski, S., Tavenard, R., Chapel, L., and Guyet, T. (2016b). Dense Bag-of-Temporal-SIFT-Words for Time Series Classification. In *Advanced Analysis and Learning on Temporal Data (AALTD'15), Lecture Notes in Computer Science*, volume 9785, pages 17–30. Springer.

Bailly, A., Malinowski, S., Tavenard, R., Guyet, T., and Chapel, L. (2015). Bag-of-Temporal-SIFT-Words for Time Series Classification. In *ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data*, Porto, Portugal.

Baydogan, M. G. and Runger, G. (2015). Learning a symbolic representation for multivariate time series classification. *Data Mining and Knowledge Discovery*, 29(2):400–422.

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.

Bégué, A., Arvor, D., Bellon, B., Betbeder, J., de Abelleyra, D., PD Ferraz, R., Lebourgeois, V., Lelong, C., Simões, M., and R Verón, S. (2018). Remote sensing and cropping practices: A review. *Remote Sensing*, 10(1):99.

Brown, J. C., Kastens, J. H., Coutinho, A. C., Victoria, D. d. C., and Bishop, C. R. (2013). Classifying multiyear agricultural land use data from Mato Grosso using time-series MODIS vegetation index data. *Remote Sensing of Environment*, 130:39–50.

Cameron, S. H. (1966). Piece-wise linear approximations. Technical report, IIT RESEARCH INST CHICAGO IL COMPUTER SCIENCES DIV.

Candan, K. S., Rossini, R., Wang, X., and Sapino, M. L. (2012). sDTW: Computing DTW Distances Using Locally Relevant Constraints Based on Salient Feature Alignments. *Proceedings of the Very Large Data Base Endowment*, 5(11):1519–1530.

Chakrabarti, K., Keogh, E., Mehrotra, S., and Pazzani, M. (2002). Locally adaptive dimensionality reduction for indexing large time series databases. *ACM Trans. Database Syst.*, 27(2):188–228.

Chan, K.-P. and Fu, A. (1999). Efficient time series matching by wavelets. In *Proceedings of the 15th International Conference on Data Engineering*, ICDE '99, pages 126–133. IEEE Computer Society.

Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15.

Chang, J., Hansen, M. C., Pittman, K., Carroll, M., and DiMiceli, C. (2007). Corn and soybean mapping in the united states using modis time-series data sets. *Agronomy Journal*, 99(6):1654–1664.

Cheboli, D. (2010). *Anomaly detection of time series*. PhD thesis, University of Minnesota Digital Conservancy.

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*, VLDB '04, pages 792–803. VLDB Endowment.

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*, SIGMOD '05, pages 491–502. ACM.

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

Clark, M. L., Aide, T. M., Grau, H. R., and Riner, G. (2010). A scalable approach to mapping annual land cover at 250 m using modis time series data: A case study in the dry chaco ecoregion of south america. *Remote Sensing of Environment*, 114(11):2816–2832.

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.

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

Cuturi, M. (2011). Fast global alignment kernels. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 929–936.

Cuturi, M. and Blondel, M. (2017). Soft-DTW: a Differentiable Loss Function for Time-Series. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 894–903.

Cuturi, M., Vert, J., Birkenes, Ø., and Matsui, T. (2007). A kernel for time series based on global alignments. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages II–413 – II–416. IEEE.

Das, G., Gunopulos, D., and Mannila, H. (1997). Finding similar time series. In *Proceedings of the First European Symposium on Principles of Data Mining and Knowledge Discovery*, PKDD '97, pages 88–100. Springer-Verlag.

Denize, J. (2015). Développement d'un modèle de classification d'images SAR pour la cartographie des cultures tropicales : Exemple de l'île de la Réunion. Master's thesis, Université Rennes 2. In French.

Dusseux, P., Corpetti, T., and Hubert-Moy, L. (2013). Temporal kernels for the identification of grassland management using time series of high spatial resolution satellite images. In *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium*, pages 3258–3260.

Duveiller, G. and Defourny, P. (2010). A conceptual framework to define the spatial resolution requirements for agricultural monitoring using remote sensing. *Remote Sensing of Environment*, 114(11):2637–2650.

Everitt, B. S., Landau, S., and Leese, M. (2009). *Cluster Analysis*. Wiley Publishing, 4th edition.

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

Gómez, C., White, J. C., and Wulder, M. A. (2016). Optical remotely sensed time series data for land cover classification: A review. *ISPRS Journal of Photogrammetry and Remote Sensing*, 116:55–72.

Gond, V., Fayolle, A., Pennec, A., Cornu, G., Mayaux, P., Camberlin, P., Doumenge, C., Fauvet, N., and Gourlet-Fleury, S. (2013a). Vegetation structure and greenness in central africa from modis multi-temporal data. 368:20120309.

Gond, V., Fayolle, A., Pennec, A., Cornu, G., Mayaux, P., Camberlin, P., Doumenge, C., Fauvet, N., and Gourlet-Fleury, S. (2013b). Vegetation structure and greenness in central africa from modis multi-temporal data. *Phil. Trans. R. Soc. B*, 368(1625):20120309.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2015). Explaining and harnessing adversarial examples. *International Conference on Learning Representations (ICLR)*.

Grabocka, J., Schilling, N., Wistuba, M., and Schmidt-Thieme, L. (2014). Learning time-series shapelets. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 392–401.

Hansen, M. C., Potapov, P. V., Moore, R., Hancher, M., Turubanova, S., Tyukavina, A., Thau, D., Stehman, S., Goetz, S., Loveland, T., et al.

(2013). High-resolution global maps of 21st-century forest cover change. *science*, 342(6160):850–853.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 edition.

Huete, A., Didan, K., Miura, T., Rodriguez, E. P., Gao, X., and Ferreira, L. G. (2002). Overview of the radiometric and biophysical performance of the MODIS vegetation indices. *Remote Sensing of Environment*, 83(1-2):195–213.

Hüttich, C., Gessner, U., Herold, M., Strohbach, B. J., Schmidt, M., Keil, M., and Dech, S. (2009). On the suitability of modis time series metrics to map vegetation types in dry savanna ecosystems: A case study in the kalahari of ne namibia. *Remote sensing*, 1(4):620–643.

Ienco, D. (2017). TiSeLaC : Time Series Land Cover Classification Challenge. https://sites.google.com/site/dinoienco/tiselc.

Itakura, F. (1975). Minimum prediction residual principle applied to speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-23(1):67–72.

Jégou, H. and Chum, O. (2012). Negative evidences and co-occurences in image retrieval: The benefit of pca and whitening. *Computer Vision–ECCV 2012*, pages 774–787.

Jégou, H., Douze, M., Schmid, C., and Pérez, P. (2010). Aggregating local descriptors into a compact image representation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3304–3311. IEEE.

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.

Jurafsky, D. and Martin, J. H. (2000). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall PTR, 1st edition.

Jurie, F. and Triggs, B. (2005). Creating efficient codebooks for visual recognition. In *Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05)*, volume 1 of *ICCV '05*, pages 604–610. IEEE Computer Society.

Kahveci, T. and Singh, A. K. (2001). Variable length queries for time series data. In *Proceedings of the 17th International Conference on Data Engineering*, pages 273–282. IEEE Computer Society.

Keogh, E. J. and Pazzani, M. J. (1998). An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, KDD'98, pages 239–243. AAAI Press.

Keogh, E. J. and Pazzani, M. J. (2000). Scaling up dynamic time warping for datamining applications. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '00, pages 285–289.

Keogh, E. J. and Pazzani, M. J. (2001). Derivative dynamic time warping. In *In First SIAM International Conference on Data Mining (SDM'2001*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

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.

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 ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 2–11.

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

Lines, J. and Bagnall, A. (2014). Time series classification with ensembles of elastic distance measures. *Data Mining and Knowledge Discovery*, 29(3):565–592.

Lines, J., Davis, L. M., Hills, J., and Bagnall, A. (2012). A shapelet transform for time series classification. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 289–297.

Liu, C., Yuen, J., Torralba, A., Sivic, J., and Freeman, W. T. (2008). Sift flow: Dense correspondence across different scenes. In *European Conference on Computer Vision*, pages 28–42. Springer.

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.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision (ICCV)*, volume 2 of *ICCV '99*, pages 1150–. IEEE Computer Society.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.

Macedo, M. N., DeFries, R. S., Morton, D. C., Stickler, C. M., Galford, G. L., and Shimabukuro, Y. E. (2012). Decoupling of deforestation and soy production in the southern Amazon during the late 2000s. *Proceedings of the National Academy of Sciences*, 109(4):1341–1346.

Malenovskỳ, Z., Rott, H., Cihlar, J., Schaepman, M. E., García-Santos, G., Fernandes, R., and Berger, M. (2012). Sentinels for science: Potential of sentinel-1,-2, and-3 missions for scientific observations of ocean, cryosphere, and land. *Remote Sensing of Environment*, 120:91–101.

Marteau, P.-F. (2009). Time warp edit distance with stiffness adjustment for time series matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(2):306–318.

Melgani, F. and Bruzzone, L. (2004). Classification of hyperspectral remote sensing images with support vector machines. *IEEE Transactions on geoscience and remote sensing*, 42(8):1778–1790.

Mikolajczyk, K. (2002). *Detection of local features invariant to affines transformations*. PhD thesis, Institut National Polytechnique de Grenoble-INPG.

Pekel, J.-F., Cottam, A., Gorelick, N., and Belward, A. S. (2016). High-resolution mapping of global surface water and its long-term changes. *Nature*, 540(7633):418.

Perronnin, F., Liu, Y., Sánchez, J., and Poirier, H. (2010). Large-scale image retrieval with compressed fisher vectors. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3384–3391. IEEE.

Petitjean, F., Inglada, J., and Gançarski, P. (2011). Temporal domain adaptation under time warping. In *International Geoscience and Remote Sensing Symposium (IGARSS 2011)*, pages 3578–3581.

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.

Ratanamahatana, C. A. and Keogh, E. (2004). Everything you know about dynamic time warping is wrong. In *Proceedings of ACM SIGKDD Workshop on Mining Temporal and Sequential Data*, pages 22–25.

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.

Rivière, P., Verges, M., Dimou, M., and Garde, F. (2013). Sustainable cities in tropical climates: Presentation of the "beauséjour" case study in reunion island. *WIT Transactions on Ecology and the Environment*, 179:641–650.

Sakoe, H. and Chiba, S. (1970). A similarity evaluation of speech patterns by dynamic programming. In *Nat. Meeting of Institute of Electronic Communications Engineers of Japan*, page 136.

Sakoe, H. and Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26.

Schäfer, P. (2015a). Bag-Of-SFA-Symbols in Vector Space (BOSS VS). Technical Report 15–30, ZIB.

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

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. ACM.

Senin, P. and Malinchik, S. (2013). SAX-VSM: Interpretable Time Series Classification Using SAX and Vector Space Model. *Proceedings of the IEEE International Conference on Data Mining*, pages 1175–1180.

Sivic, J. and Zisserman, A. (2003). Video google: A text retrieval approach to object matching in videos. In *null*, page 1470. IEEE.

Stefan, A., Athitsos, V., and Das, G. (2013). The move-split-merge metric for time series. *IEEE Trans. on Knowl. and Data Eng.*, 25(6):1425–1438.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., and Fergus, R. (2014). Intriguing properties of neural networks. *International Conference on Learning Representations (ICLR)*.

Tavenard, R., Malinowski, S., Chapel, L., Bailly, A., Sanchez, H., and Bustos, B. (2017). Efficient Temporal Kernels between Feature Sets for Time Series Classification. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery*, Skopje, Macedonia.

Tuia, D., Ratle, F., Pacifici, F., Kanevski, M., and Emery, W. (2009). Active learning methods for remote sensing image classification. 48:2218 – 2232.

Vedaldi, A. and Fulkerson, B. (2010). Vlfeat: An open and portable library of computer vision algorithms. In *Proceedings of the 18th ACM International Conference on Multimedia*, pages 1469–1472. ACM.

Vlachos, M., Gunopoulos, D., and Kollios, G. (2002). Discovering similar multidimensional trajectories. In *Proceedings of the 18th International Conference on Data Engineering*, ICDE '02, pages 673–. IEEE Computer Society.

Wang, H., Ullah, M. M., Klaser, A., Laptev, I., and Schmid, C. (2009). Evaluation of local spatio-temporal features for action recognition. In *Proceedings of the British Machine Vision Conference*, pages 124.1–124.11.

Wang, J., Liu, P., F.H. She, M., Nahavandi, S., and Kouzani, A. (2013a). Bag-of-Words Representation for Biomedical Time Series Classification. *Biomedical Signal Processing and Control*, 8(6):634–644.

Wang, X., Mueen, A., Ding, H., Trajcevski, G., Scheuermann, P., and Keogh, E. (2013b). Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, pages 1–35.

Wardlow, B. D. and Egbert, S. L. (2008). Large-area crop mapping using time-series modis 250 m ndvi data: An assessment for the us central great plains. *Remote sensing of environment*, 112(3):1096–1116.

Wolpert, D. H. and Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82.

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*, CIKM'00, pages 488–495. ACM.

Xie, J. and Beigi, M. (2009). A Scale-Invariant Local Descriptor for Event Recognition in 1D Sensor Signals. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, pages 1226–1229.

Xie, Y., Sha, Z., and Yu, M. (2008). Remote sensing imagery in vegetation mapping: a review. *Journal of plant ecology*, 1(1):9–23.

Ye, L. and Keogh, E. (2009). Time series shapelets: a new primitive for data mining. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 947–956.

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(1-2):149–182.

Yi, B.-K. and Faloutsos, C. (2000). Fast time sequence indexing for arbitrary lp norms. In *Proceedings of the 26th International Conference on Very Large Data Bases*, VLDB '00, pages 385–394. Morgan Kaufmann Publishers Inc.

Appendices

A

Error Rates Numerical Comparison

For readibility, the numerical error rate values for the UEA / UCR database are separated into two Tables: A.1 and A.2. We provide error rates for our two proposed algorithms: D-BoTSW and ABS ; as well as for LTS since the algorithm provided by Bagnall et al. [2017] has been updated and we run it on UEA / UCR database for fair comparison with ABS.

In this manuscript, we also used error rates (on 85 datasets) provided by Bagnall et al. [2017] for BoP, BOSS, COTE, DTW-NN, ED-NN, LTS, PROP, SAX-VSM and TSBF algorithms. Error rates for SMTS methods (on 45 datasets) can be found in [Baydogan and Runger, 2015].

name	D-BoTSW	ABS	LTS
50words	0.218	-1	-1
Adiac	0.269	-1	0.332
ArrowHead	0.189	0.126	0.143
Beef	0.167	0.2	0.167
BeetleFly	0.1	0.2	0.2
BirdChicken	0.2	0.2	0.2
Car	0.067	0.15	0.183
CBF	0	0.009	0.01
ChlorineConcentration	0.45	0.359	0.385
CinC_ECG_torso	0.192	0.105	0.137
Coffee	0	0	0
Computers	0.236	0.392	0.408
Cricket_X	0.231	0.241	0.251
Cricket_Y	0.249	0.256	0.285
Cricket_Z	0.233	0.241	0.259
DiatomSizeReduction	0.085	0.02	0.026
DistalPhalanxOutlineAgeGroup	0.168	0.266	0.266
DistalPhalanxOutlineCorrect	0.188	0.214	0.214
DistalPhalanxTW	0.223	0.36	0.381
Earthquakes	0.205	0.259	0.252
ECG200	0.06	0.14	0.14
ECG5000	0.056	0.064	0.061
ECGFiveDays	0	0	0
ElectricDevices	0.333	0.334	0.34
FaceAll	0.201	0.224	0.226
FaceFour	0.023	0.034	0.045
FacesUCR	0.055	0.06	0.06
FISH	0.023	0.04	0.034
FordA	0.087	0.066	0.077
FordB	0.106	0.181	0.18
Gun_Point	0.02	0	0
Ham	0.362	0.324	0.305
HandOutlines	0.117	-1	0.089
Haptics	0.549	0.549	0.545
Herring	0.406	0.391	0.375
InlineSkate	0.613	-1	0.629
InsectWingbeatSound	0.438	0.414	0.406
ItalyPowerDemand	0.064	0.036	0.039
LargeKitchenAppliances	0.131	0.24	0.272
Lightning2	0.115	0.164	0.197
Lightning7	0.301	0.205	0.205
MALLAT	0.106	0.047	0.049

Table A.1 – Error rates from various time series classification methods from *50words* to *MALLAT*

name	D-BoTSW	ABS	LTS
Meat	0.1	0.15	0.167
MedicalImages	0.27	0.307	0.328
MiddlePhalanxOutlineAgeGroup	0.193	0.403	0.409
MiddlePhalanxOutlineCorrect	0.35	0.162	0.192
MiddlePhalanxTW	0.378	0.494	0.513
MoteStrain	0.169	0.131	0.144
NonInvasiveFatalECG_Thorax1	0.059	-1	-1
NonInvasiveFatalECG_Thorax2	0.062	-1	-1
OliveOil	0.6	0.833	0.833
OSULeaf	0.07	0.256	0.24
PhalangesOutlinesCorrect	0.23	0.198	0.217
Phoneme	0.718	-1	-1
Plane	0	0	0
ProximalPhalanxOutlineAgeGroup	0.141	0.176	0.176
ProximalPhalanxOutlineCorrect	0.158	0.103	0.134
ProximalPhalanxTW	0.203	0.2	0.185
RefrigerationDevices	0.544	0.488	0.507
ScreenType	0.445	0.589	0.568
ShapeletSim	0	0.033	0.111
ShapesAll	0.107	-1	-1
SmallKitchenAppliances	0.272	0.32	0.304
SonyAIBORobotSurface	0.088	0.186	0.176
SonyAIBORobotSurfaceII	0.17	0.087	0.129
StarLightCurves	0.023	-1	0.044
Strawberry	0.044	0.059	0.089
SwedishLeaf	0.082	0.096	0.088
Symbols	0.016	0.05	0.06
synthetic_control	0.003	0.003	0.003
ToeSegmentation1	0.035	0.066	0.066
ToeSegmentation2	0.069	0.062	0.069
Trace	0	0	0
Two_Patterns	0.001	0.003	0.004
TwoLeadECG	0.009	0.001	0.003
uWaveGestureLibrary_X	0.208	0.201	0.198
uWaveGestureLibrary_Y	0.265	0.296	0.293
uWaveGestureLibrary_Z	0.26	0.258	0.258
UWaveGestureLibraryAll	0.13	-1	-1
wafer	0.004	0.004	0.004
Wine	0.426	0.5	0.5
WordsSynonyms	0.309	0.319	0.317
Worms	0.343	0.39	0.39
WormsTwoClass	0.315	0.273	0.299
yoga	0.142	0.161	0.154

Table A.2 – Error rates from various time series classification methods from *Meat* to *yoga*

List of Figures

List of Tables

CONTENTS

Classification de Séries Temporelles avec Applications en Télédétection

Résumé La classification de séries temporelles a suscité beaucoup d'intérêt au cours des dernières années en raison de ces nombreuses applications. Nous commençons par proposer la méthode Dense Bag-of-Temporal-SIFT-Words (**D-BoTSW**) qui utilise des descripteurs locaux basés sur la méthode SIFT, adaptés pour les données en une dimension et extraits à intervalles réguliers. Des expériences approfondies montrent que notre méthode D-BoTSW surpassent de façon significative presque tous les classificateurs de référence comparés. Ensuite, nous proposons un nouvel algorithme basé sur l'algorithme *Learning Time Series Shapelets* (LTS) que nous appelons Adversarially-Built Shapelets (**ABS**). Cette méthode est basée sur l'introduction d'exemples adversaires dans le processus d'apprentissage de LTS et elle permet de générer des shapelets plus robustes. Des expériences montrent une amélioration significative de la performance entre l'algorithme de base et notre proposition. En raison du manque de jeux de données labelisés, formatés et disponibles en ligne, nous utilisons deux jeux de données appelés *TiSeLaC* et *Brazilian-Amazon*.

Mots Clés Classification des séries temporelles, Télédétection, Sac-de-mots, SIFT, Shapelets de séries temporelles, Réseaux de neurones convolutionnels, Exemples adversaires

Time Series Classification Algorithms with Applications in Remote Sensing

Abstract Time Series Classification (TSC) has received an important amount of interest over the past years due to many real-life applications. In this PhD, we create new algorithms for TSC, with a particular emphasis on Remote Sensing (RS) time series data. We first propose the Dense Bag-of-Temporal-SIFT-Words (**D-BoTSW**) method that uses dense local features based on SIFT features for 1D data. Extensive experiments exhibit that D-BoTSW significantly outperforms nearly all compared standalone baseline classifiers. Then, we propose an enhancement of the Learning Time Series Shapelets (LTS) algorithm called Adversarially-Built Shapelets (**ABS**) based on the introduction of adversarial time series during the learning process. Adversarial time series provide an additional regularization benefit for the shapelets and experiments show a performance improvement between the baseline and our proposed framework. Due to the lack of available RS time series datasets, we also present and experiment on two remote sensing time series datasets called *TiSeLaC* and *Brazilian-Amazon*.

Keywords Time Series Classification, Remote Sensing Time Series, Bag-of-Words, SIFT features, Time Series Shapelets, Convolutional Neural Networks, Adversarial Examples

Laboratoire LETG-Rennes UMR 6554 CNRS
Université Rennes 2
Place du recteur Henri Le Moal, CS 24037,
35043 Rennes cédex, France