MSAX: Multivariate symbolic aggregate approximation for time series classification

Manuel Pereira da Conceição Monteiro Anacleto

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor(s): Prof. Alexandra Sofia Martins de Carvalho
Prof. Susana de Almeida Mendes Vinga Martins

Examination Committee
Chairperson: Prof. Teresa Maria Sá Ferreira Vazão Vasques
Supervisor: Prof. Alexandra Sofia Martins de Carvalho
Member of the Committee: Prof. Rui Miguel Carrasqueiro Henriques

November 2019
Declaration:

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.
Acknowledgments

I would like to thank Prof. Alexandra Carvalho and Prof. Susana Vinga for the great guidance, teaching and patience that supported me throughout this work.

This work was partially supported by national funds from Fundação para a Ciência e a Tecnologia (FCT), through projects PTDC/EMS-SIS/0642/2014, PTDC/CCI-CIF/29877/2017, PTDC/EEI-SII/1937/2014, UID/EEA/50008/2013, UID/CEC/50021/2019, and UID/EMS/50022/2019.
Resumo

A análise de séries temporais (TS) é um tópico de pesquisa central em áreas como finanças, bioinformática e previsão do tempo, em que o objetivo é extrair conhecimento através de técnicas de data mining. O Symbolic aggregate approximation (SAX) é um método actual que realiza discretização e redução de dimensionalidade para TS univariadas. Estas são abordagens chaves para representação e análise de TS. Neste trabalho, propomos o MSAX, uma extensão deste algoritmo para TS multivariadas que toma em consideração a estrutura de covariância dos dados. O método é testado em vários conjuntos de dados, incluindo os datasets Pen Digits e Character Trajectories e mais doze datasets de referência. Em alguns dos testes feitos, o MSAX exibe desempenho comparável com métodos actuais em termos de precisão de classificação. Embora não seja superior a 1-NN e DTW, possui características interessantes para alguns casos e, portanto, enriquece o conjunto de métodos para analisar TS multivariadas.

Palavras-chave: SAX, Séries Temporais, Classificação, Análise Multivariada
Abstract

Time series (TS) analysis is a central research topic in areas such as bioinformatics, finance, and weather forecasting, where the goal is to extract knowledge through data mining techniques. Symbolic aggregate approximation (SAX) is a state-of-the-art method that performs discretization and dimensionality reduction for univariate TS, which are critical steps for TS representation and analysis. Due to its good results, SAX has had a great number of applications in TS analysis tasks, nonetheless, in multivariate cases, its direct application to each variable individually does not consider the relationship between the attributes. In this work, we propose MSAX, an extension of this discretization algorithm to multivariate TS that considers all variables simultaneously and their relations through the covariance structure of the data. The method is tested in several datasets, including the Pen Digits, Character Trajectories, and other twelve benchmark files. Depending on the experiment, MSAX exhibits comparable performance with state-of-the-art methods in terms of classification accuracy. Although not superior to well-known classification methods like the first nearest neighbour with Dynamic Time Warping, it has interesting characteristics for some cases, and thus enriches the set of methods to analyze multivariate TS.

Keywords: SAX, Time Series, classification, multivariate analysis
# Contents

Acknowledgments ................................................................. v
Resumo .................................................................................. vii
Abstract ............................................................................... ix
List of Tables ......................................................................... xiii
List of Figures ......................................................................... xv
Nomenclature ......................................................................... xvii
Glossary ................................................................................ 1

1 Introduction .......................................................................... 1
  1.1 Motivation ........................................................................ 1
  1.2 Objectives and Contributions ........................................ 2
  1.3 Document Outline ............................................................ 2

2 Temporal Data Mining .......................................................... 5
  2.1 Machine Learning ............................................................. 5
     2.1.1 Classification ............................................................. 5
  2.2 Time series definition and related concepts ....................... 7
  2.3 Temporal Data ................................................................ 9

3 Time Series Classification ....................................................... 11
  3.1 TS Representations .......................................................... 12
     3.1.1 Piecewise Aggregate Composition (PAA) .................. 12
     3.1.2 Symbolic Aggregate Approximation (SAX) ............... 13
  3.2 Univariate Classification methods ..................................... 15
     3.2.1 Bag-of-Patterns (BOP) .............................................. 15
     3.2.2 SAX-VSM ................................................................. 16
     3.2.3 Fast Shapelets (FS) ................................................... 18
     3.2.4 DTW Features .......................................................... 19
     3.2.5 Univariate Experiments ............................................ 20
  3.3 Multivariate Classification methods .................................... 25
     3.3.1 Time Series Bag-of-Features (TSBF) ....................... 25
     3.3.2 Symbolic Multivariate Time Series classifier (SMTS) ...... 27
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3.3</td>
<td>Multivariate Experiments</td>
<td>28</td>
</tr>
<tr>
<td>4</td>
<td>Proposed Method</td>
<td>33</td>
</tr>
<tr>
<td>4.1</td>
<td>MSAX</td>
<td>33</td>
</tr>
<tr>
<td>4.1.1</td>
<td>MSAX normalization</td>
<td>34</td>
</tr>
<tr>
<td>4.1.2</td>
<td>MSAX discretization</td>
<td>35</td>
</tr>
<tr>
<td>4.1.3</td>
<td>Dissimilarity definition</td>
<td>36</td>
</tr>
<tr>
<td>4.2</td>
<td>Method evaluation</td>
<td>39</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Case studies</td>
<td>39</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Experiments on benchmark datasets for MTS classification</td>
<td>48</td>
</tr>
<tr>
<td>4.3</td>
<td>Method discussion</td>
<td>53</td>
</tr>
<tr>
<td>5</td>
<td>Conclusions and Achievements</td>
<td>55</td>
</tr>
<tr>
<td></td>
<td>Bibliography</td>
<td>57</td>
</tr>
<tr>
<td>A</td>
<td>MSAX and SAX_INDP comparison trough a classification problem in multiple datasets</td>
<td>63</td>
</tr>
<tr>
<td>A.1</td>
<td>Accuracy values</td>
<td>63</td>
</tr>
<tr>
<td>A.2</td>
<td>Accuracy graphs</td>
<td>66</td>
</tr>
</tbody>
</table>
List of Tables

2.1 Confusion Matrix ................................................................. 7
3.1 The look up table used by the MINDIST function. .......................... 14
3.2 1-NN comparison between Euclidean Distance, SAX representation and DTW, in TS classification. ...................................................... 22
3.3 Comparison between BOP, SAX-VSM, Fast Shapelets, DTW Features and best classifier result from TSC, in TS classification. ..................... 24
3.4 Average and standard deviation values for the difference between the best TSC accuracy value and BOP, SAX-VSM, FS and DTW Features. .................. 25
3.5 Comparison between 1-NN Euclidean, 1-NN DTW, 1-NN SAX_INDP, TSBF and SMTS, in MTS classification. .............................................. 31
4.1 MSAX and SAX_INDP accuracy's for ECG dataset ............................ 50
4.2 MSAX and SAX_INDP accuracy's for PenDigits dataset ...................... 50
A.1 MSAX and SAX_INDP accuracy's with AUSLAN dataset ...................... 63
A.2 MSAX and SAX_INDP accuracy's with CharacterTrajectories dataset ........ 63
A.3 MSAX and SAX_INDP accuracy's with Epilepsy dataset ...................... 64
A.4 MSAX and SAX_INDP accuracy's with EthanolConcentration dataset ........ 64
A.5 MSAX and SAX_INDP accuracy's with LIBRAS dataset ...................... 64
A.6 MSAX and SAX_INDP accuracy's with LP1 dataset ............................ 64
A.7 MSAX and SAX_INDP accuracy's with LP2 dataset ............................ 64
A.8 MSAX and SAX_INDP accuracy's with LP3 dataset ............................ 65
A.9 MSAX and SAX_INDP accuracy's with LP4 dataset ............................ 65
A.10 MSAX and SAX_INDP accuracy's with LP5 dataset ......................... 65
A.11 MSAX and SAX_INDP accuracy's with StandWalkJump dataset ........... 65
A.12 MSAX and SAX_INDP accuracy's with UWAVE dataset ..................... 65
A.13 MSAX and SAX_INDP accuracy's with WAFER dataset ..................... 66
# List of Figures

2.1 Visual representation of the euclidean and dtw distance alignment on time series similarity measure. ......................................................... 9

3.1 PAA representation example. ......................................................... 13

3.2 Description of the SAX discretization process. ............................... 14

3.3 SAX representation phases and distance measures used on each one. ........ 15

3.4 Visual intuition of BOP representation example. ............................... 17

3.5 Sketch of the SAX-VSM algorithm. .................................................. 18

3.6 Visual intuition of an example analysis of a candidate shapelet in the Fast Shapelets method. 19

3.7 Visual representation of the classification accuracy rates of the Euclidean 1-NN and SAX representation 1-NN methods as 2-dimensional points. ......................................................... 21

3.8 Overview of the TSBF algorithm. ..................................................... 26

3.9 Visual representation of the SMTS representation steps through an example case. .... 28

3.10 Accuracy plot of the: 1-NN Euclidean, 1-NN DTW, 1-NN SAX_INDP, TSBF and SMTS, values in a MTS classification experiment. ......................................................... 29

4.1 Bivariate Gaussian curve plot. .......................................................... 37

4.2 Plot of the symbolic areas of the MSAX on an example case. ................. 37

4.3 Discretization example case of MSAX. ............................................. 38

4.4 PenDigits dataset samples ............................................................... 40

4.5 CharacterTrajectories dataset samples ............................................ 41

4.6 Confusion matrices of the classification results of 1-NN with MSAX and SAX_INDP (with an alphabet size of 10 and a TS length reduction ratio of 1) for the PenDigits dataset. 42

4.7 Correlation matrices of some classes of the PenDigits dataset. ............... 43

4.8 Confusion matrices of the classification results of 1-NN with MSAX and SAX_INDP (with an alphabet size of 10 and a TS length reduction ratio of 1/2) for the CharacterTrajectories dataset. ......................................................... 44

4.9 Correlation matrices of some classes of the CharacterTrajectories dataset. ........ 45

4.10 MSAX and SAX_INDP study on PenDigits and CharacterTrajectories datasets, accuracy against alphabet size plot and TS length reduction ratio plots. ......................................................... 46
4.11 MSAX and SAX,INDP study on PenDigits and CharacterTrajectories datasets, run time against alphabet size and TS length reduction ratio plots. .................................................. 47
4.12 MSAX and SAX,INDP accuracy’s comparison with: AUSLAN, CharacterTrajectories, ECG, Epilepsy, Libras and PenDigits datasets. ................................................................. 49
4.13 Accuracy plot of the 1-NN MSAX and 1-NN SAX,INDP values in a MTS classification experiment with 15 datasets. ................................................................. 51
4.14 Accuracy plot of the: 1-NN MSAX, 1-NN SAX,INDP 1-NN Euclidean, 1-NN DTW, TSBF and SMTS, accuracy values in a MTS classification experiment. ..................... 52
A.1 MSAX and SAX,INDP accuracy’s comparison with: EthanolConcentration, LP1, LP2, and LP3 datasets. ................................................................. 66
A.2 MSAX and SAX,INDP accuracy’s comparison with: LP4, LP5, Standwalkjump, UWAVE and WAFER datasets. ................................................................. 67
Nomenclature

1-NN  First Nearest Neighbour
BOP   Bag Of Patterns
DTW   Dynamic Time Warping
FS    Fast Shapelets
MSAX  Multivariate Symbolic Aggregate approximatio
MTS   Multivariate Time Series
PAA   Piecewise Aggregate Approximation
SAX   Symbolic Aggregate approximatio
SMTS  Symbolic Multivariate Time Series Classifier
SVM   Support Vector Machine
TSBF  Time Series Bag of Features
TS    Time Series
VSM   Vector Space Model
Chapter 1

Introduction

1.1 Motivation

For the last years Humans became great collectors of data in almost all fields of knowledge. Huge quantities of data are now available from which is desired to extract information like meaningful statistics and other characteristics such as patterns and regularities. This motives the emergency of the field of Machine Learning which, Tom Mitchell describes in [1] as, “The question of how to construct computer programs that automatically improve with experience”. Instead of the traditional approach of explicit programming the computers, they are made so that they are programmed to learn from data and modify themselves according to it.

One particular case of this data is temporal data, best know in the machine learning community as time series (TS), which is a series of values indexed by time in a chronological order obtained by repeated measures over time. This data type can be found in several meaning and important contexts such as bioinformatics, finances and weather forecasting. Machine learning techniques present the opportunity for new and more study types on changes and evolution of data overtime in these fields. Temporal Data Classification, as a sub field of temporal data mining, is a challenging and important task in many domains also. Taking a particular example of the medical domain, where in a multitude of different areas, cases where correct classification of TS data has immediate implications for diagnosis, quality assessment or prediction of meaningful outcomes [2–4], are especially essential.

Data representation takes a big focus in TS analyses. A rich wealth of data structures and algorithms for streaming data, where the data is assumed to be discrete, were developed especially by the text processing and bioinformatics communities in recent years [5–10]. Some examples of methods that are not defined for real-valued TS are: Markov models, suffix trees, decision trees and hashing. In order to make use of theses methods on the majority of time series where the data is real valued a need for good symbolic approaches has arise. Beside this, another very relevant factor is that these methods can also address the TS dimensionality problem arising from the fact that almost all TS datasets are very high dimensional and virtually all data mining algorithms scale poorly with dimensionality. In these ways, representation methods prove very useful in working with TS. Several state-of-the-art TS classification

1.2 Objectives and Contributions

This works proposes Multivariate Symbolic Aggregate Approximation (MSAX), a multivariate extension of the TS representation Symbolic aggregate approximation (SAX) algorithm [14] and evaluates its behaviour regarding classification tasks. Initially a study on the univariate algorithm was done, then several univariate TS classifications methods that make use of SAX were reviewed and their behaviour was analyzed in TS classification tasks. Through the results of these experiments the SAX utility was studied in order to confirm and motivate the idea of extending the method to multivariate TS. Following, a review on some state-of-the-art multivariate TS classification algorithms was done in order to compare MSAX in classification tasks against these. The MSAX was implemented and then its behaviour was analyzed through classification tasks where its results were compared other multivariate classification methods. In a brief manner the contributions of this thesis consists in:

- A review and analysis of the SAX algorithm influence on univariate TS classification tasks.
- The presentation of MSAX, a novelty extension of SAX for multivariate TS representation that takes in account the correlation between the multivariate TS attributes.
- A study of the behaviour of MSAX through classification tasks comparing it to the original SAX algorithm applied to multivariate TS. In these experiments several benchmark datasets from the multivariate TS classification community were used and so benchmarks values for SAX and MSAX were presented. An implementation of the MSAX algorithm is available in: https://github.com/ManuelMonteiro24/MSAX, also in this repository is present the code to reproduce all the experiments presented on this work.
- An analysis of the influence of SAX and MSAX on multivariate TS classification tasks.

The above findings resulted in a publication that was already presented orally at the conference CIBB 2019 (16th International Conference on Computational Intelligence methods for Bioinformatics and Biostatistics):


1.3 Document Outline

The remaining part of this report is organized as follows. On Chapter 2 a brief review on the field of machine learning, with focus on classification tasks, is regarded. Following a definition of TS and related
concepts is presented and in the end a brief overview of TS analysis tasks (Temporal Data mining) is done. In Chapter 3 TS classification is approached more deeply followed by a introduction to TS representation methods, where the SAX method is described, then some TS classification methods that make of use of SAX are presented and evaluated, in the end, multivariate TS classification is regarded with the presentation and evaluation of some state-of-the-art methods. In Chapter 4 the MSAX algorithm is proposed and described, following the behaviour of the method is addressed in several experiment classification tasks and finally a discussion on several topics related to the proposed method is done. The Chapter 5 addresses the conclusions withdraw from this work and points several directions to a continuous future work following this one.
Chapter 2

Temporal Data Mining

In this chapter a brief review of machine learning focusing on the classification task is done. Then, the definition of TS and other relevant characteristics of it are presented, followed by an overview on temporal data mining. Also in this chapter, several used concepts along the document, from the referenced areas, are explained here.

2.1 Machine Learning

Data mining is concerned with the analyses of large volumes of data, in order to automatically discover interesting regularities or relationships which in turn lead to a better understanding of the underlying processes of data creation and its relations. It is called machine learning to a group of statistical techniques that allow computer systems the ability to learn from the data without being explicit programmed to it. Depending on the output that is desired to extract from the data, several tasks can be distinguish in this field. Machine learning can be divided in several areas, three of the more popular ones are: clustering, regression and classification. Clustering focus on the task of grouping instances in a way that the instances in the same group are more similar to each others than to those which are in different groups. In regression the relationship of a dependent variable with other changing variables is estimated and studied. The classification tasks is addressed more deeply in the following subsection due to be in the scope of this work. One difference it is very common to regard in relation to machine learning tasks is the fact of being supervised or unsupervised. In supervised tasks, inputs and their respective outputs are presented to the system and the objective lies on discovering a pattern that maps the inputs to the outputs. On the other hand, on unsupervised tasks no outputs (or labels) are given with the input data, only with this information the algorithms try to find some structure in the data. Given the scope of this work, only the classification task is regarded.

2.1.1 Classification

Classification is a supervised task where a function that maps a data item into one of several predefined classes is produced by inputting a training data set and building a model of the classes based on the
data attributes. In other words, classification is a data mining (machine learning) technique used to predict group membership for data instances. In particular, this work is concerned with classification problems in which the output of instances admits only discrete, unordered values.

Several techniques have been proposed in order to solve different supervised machine learning problems [15], some of the most know are: decision trees, neural networks, support vector machines (SVM), instance-based classifiers and statistical learning algorithms as naive Bayes and Bayesian networks. In the following a description of a very popular instance-based classifier, \(K\)-nearest neighbor, that is several times referenced and used in work, is presented. This is done in order to show an example of a proposed solution to a classification problem.

The nearest neighbor classifiers are based on learning by analogy. The training samples are described by \(n\) dimensional numeric attributes. Each sample represents a point in an \(n\)-dimensional space. In this way, all of the training samples are stored in an \(n\)-dimensional space. When given an unknown sample, a \(k\)-nearest neighbor classifier searches this space for the \(k\) training samples that are closest to the unknown sample. “Closeness” is normally defined in terms of Euclidean distance, though other distance measures could be used instead. The unknown sample is assigned the most common class among its \(k\) nearest neighbors. When \(k = 1\), the unlabeled sample is assigned the class of the training sample that is closest to it.

These classifiers are instance-based learners in that they store all of the training samples and do not build a classifier until a new sample needs to be classified. This contrasts with eager learning methods, such as the popular decision tree induction [16], which construct a generalization model before receiving new samples to classify. Unlike decision tree induction, nearest neighbor classifiers assign equal weight to each attribute. This may cause confusion when there are many irrelevant attributes in the data. The \(k\)-nearest neighbors algorithm is among the simplest of all machine learning algorithms.

In the aforementioned algorithm description, the task was regarded through samples. In machine learning, a group of samples that is desired to discover a pattern in is regarded as a dataset. In classification, as a supervised task in a specific data it is common to distinguish between the training dataset and testing dataset. The training dataset gathers the instances and associated labels through which the classifying algorithm is to learn. Through the testing dataset it is given to the classifier instances that are unlabeled in order to be classified. Knowing the test instances labels the user can compare them to the labels given by the classifier to the testing instances and check if they are correct or not. Regarding to this the cross-validation concept that is several times mentioned along this work is introduced. Cross-validation is a model validation technique with the objective of evaluating how the results of classifier will generalize to an independent dataset (new data that was not used in the learning of the classifier), i.e, how accurately the predictive model will behaviour in practice. This is done in order to avoid a bias or overfitting to the classifier training data. This is normally achieved by doing multiple classification rounds using different training and testing datasets and combining the results (normally through average) over all the rounds to estimate the model predictive performance.

The aforementioned concepts leads to another important factor in classification tasks that is evaluation metrics. In classification there are four different types of outcomes, two of which are errors:
**False positive** This means that although instance \( I \) does not belong to class \( C \), it is classified as a member of class \( C \). **False negative** Although instance \( I \) belongs to class \( C \), it is not classified as a member of it. **True positive** Instance \( I \) belongs to class \( C \) and is classified as such. **True negative** The instance \( I \) does not belong to class \( C \) and it is not classified as a member of it.

When analyzing the performance of a classifier in a certain dataset the confusion matrix, depicted in Table 2.1, representing the number of occurrences of this four types of outcomes can be calculated in order to have better perception of the behaviour of the classifier.

<table>
<thead>
<tr>
<th>Prediction Outcome: Positive</th>
<th>Actual Value: Positive</th>
<th>Actual Value: Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>True positive</td>
<td>True positive</td>
<td>False positive</td>
</tr>
<tr>
<td>False negative</td>
<td>False negative</td>
<td>True negative</td>
</tr>
</tbody>
</table>

More straightforward measures like sensitivity, specificity and accuracy also can be employed to regard the performance of the classifier. Nonetheless the more common one and used in this work is the accuracy. This measures is a ratio between the correctly classified samples and all the classified samples. It can also be defined also as the probability of correctly classifying a randomly selected instance [17]. Following the function of this measure is also presented in terms of the possible classification outcomes.

\[
\text{Accuracy} = \frac{\#\text{true Pos.} + \#\text{true Neg.}}{\#\text{true Pos.} + \#\text{true Neg.} + \#\text{false Pos.} + \#\text{false Neg.}} = \frac{\#\text{correctly classified samples}}{\#\text{classified samples}}.
\]  

(2.1)

### 2.2 Time series definition and related concepts

A time series (TS) is a series of \( n \) observations each one being measured at a specific time \( t \in \{1, \ldots, T\} \) made sequentially and regularly through \( T \) instances of time. In this case, the \( i \)-th TS is given by \( \{x^i[t] \}_{t \in \{1, \ldots, T\}} \), where \( x^i[t] = (x^i_1[t], \ldots, x^i_n[t]) \). When \( n = 1 \) the TS is said to be univariate; otherwise, when \( n > 1 \), it is called Multivariate TS (MTS). In relation to the length of the TS, \( T \), it is normally classified as short when \( T \) ranges from 2 to a few dozens, else it is classified as long serie. A TS can also be continuous or discrete. A TS is continuous when its observations are measured at every moment of time in opposition to a discrete TS where observations are recorded at discrete points in time. For example, temperature readings or the concentration of a chemical propose can be record as a continuous TS. While exchange market rates or the production of an employee, recorded at certain intervals, represent a discrete TS. Nonetheless a continuous time serie can be put as discrete TS by merging data together over a specific time interval. Another relevant feature of time series is the time interval between observations, this can vary leading to unevenly spaced TS [18], nonetheless the most common are equally spaced TS and are the ones concerned in this work.

A relevant idea in the scope of this work is the subsequence (or subserie) concept. A subsequence is a continuous sequence of a TS with size minor than than the length \( T \) of the TS. Other relevant notion
to also have in mind is the similarity between two TS, which is used in almost every TS analysis task. In the following, the two most used distances measures are presented. The euclidean distance is the simplest and most intuitive one, is calculated through the sum of the differences between each pair of points corresponding to the same temporal index of two TS. Given $Q$ and $C$ as two TS examples the function is defined as follows:

$$
Euclidean\_distance(Q, C) = \sqrt{\sum_{i=1}^{T} (q[i] - c[i])^2}
$$

(2.2)

However simple and efficient in several temporal data mining problems the euclidean distance has some limitations, namely: it requires that the two series being measure have the same length and is very sensitive to small mis-matches like temporal shifts or delays, for two similar series with a small temporal delay between them the euclidean distance between them is unreasonably big. This disadvantages of the euclidean distance are override by the dynamic time warping (DTW) distance where, in contrast to the euclidean distance where the elements of two TS are aligned one to one in a rigid way, in DTW nonlinear alignments between the TS are possible, a visual intuition of both distance is presented in Figure 2.1. In [19] this distance was introduced in the temporal data mining field and due to its very good results has gained a lot of attention is the last years with a lot of methods implementing it and several extensions or modifications to it being proposed. Given $Q$ and $C$ as two examples TS the DTW distance is computed through finding the best alignment between the two TS. For this an $T_1$-by-$T_2$ matrix, where $T_1$ and $T_2$ represent the length of each TS respectively, is build, where the entry of each $(ith, jth)$ element corresponds to value of aligning the point $q[i]$ of TS $Q$ with the $c[j]$ point of TS $C$, which is equal to $(q[i] - c[j])^2$. The alignment between the series is represented by a warping path, $W = w_1, w_2, ..., w_k, ..., w_K$, that starts in the bottom-left corner and finish in e top-right corner of the matrix, this path was to be continuous. Then the chosen alignment is given by the path that minimizes the total cost of aligning its points, the DTW distance will correspond to this total cost. In the following the DTW function for the TS $Q$ and $C$ is presented:

$$
DTW(Q, C) = \sum_{k=1, w_k=(i,j)}^{K} (q[i] - c[j])^2
$$

(2.3)

The minimal cost alignment is computed through a dynamic programming algorithm. Being DTW an elastic distance measure it is more robust that the euclidean distance, but it is also more computationally expensive distance. Though several constrains that aim to speed up the computation [21, 22] and alternative approaches based on this distance have been proposed [23, 24], even so, DTW still performs several orders slower than the euclidean distance.

The distance measures referenced above are the two most used ones in TS classification tasks but beside them several other distance have been proposed, discussed and tested. In [25] a review on some state-of-the-art distances in TS classification is presented.
Figure 2.1: Visual representation of the euclidean (a) and DTW (b) distances alignment between two TS, one in red and the other in blue. The black lines represent the alignment between the points of the two TS through which the distance value is calculated. In the euclidean distance, a rigid linear alignment is done, where in DTW nonlinear alignments are possible. Considering the example TS in the figure is clear that they are very similar, the only difference resides on the fact that the two last spikes of the blue one are slight delayed in comparison to the red one. This is one clear example where the DTW achieves a better result than the euclidean distance. Figure from [20].

2.3 Temporal Data

In [26] three temporal data types are defined: time series, temporal sequences and semantic temporal data. This work focus only in the time series type. The main goal of temporal data mining is to automatically gather useful information from data of this type, such as TS. Through machine learning and others techniques, features such as regularities or relationships which can lead to a better understand of the underlying generating processes of this data. The usage and importance of temporal data is increasing in several relevant fields, such as biomedicine, finances, weather forecasting, internet site usage monitoring and more. Due to the motivating demand of solutions by these fields in the last years several temporal data mining techniques were proposed and shown to be useful in many applications. For example, in health care due to the evolution and availability of technology, more and more nowadays regular physiological analysis of patients, blood levels analysis for example, are performed and electronic saved associated to a patient medical profile. In these terms a data-adaptive monitization making use of a patient temporal data could be key in preventing deceases, improving the patient quality of life and relieve the hospital work balance. The study of patients data, since checking trends or general temporal changes to forecasting a future patient state, is a task in the scope of the temporal data mining field that motivate and increase its need.

In [27] the range of temporal data mining tasks is divided in 5 fields: prediction, classification, clustering, search and retrieval and pattern discovery. In this work only the classification field is approached, a deeper insight to it is presented in the following chapter.

It is important to note that classical machine learning techniques are designed to deal with static data and cannot simply be used with time series data, often they have to extended in appliance to the temporal domain. In [28] is stated that classical algorithms often do not consider the autocorrelation structure of time series data. The tasks on temporal data are differentiated from traditional data mining problems due to the fact that the attributes values are ordered, in this way, methods that take in cont the inter-temporal dependencies of the data must be used.

One particularly characteristic of temporal data mining is the large volume of data. Before this, efficient algorithms and representations methods for the data are key in this field.
One important concept to have in mind is data preprocessing. In order to process the data correctly and achieve meaningful results, the data must be first preprocessed (or clean) and be in agreement with the algorithm structure that will operate on it. Depending on the task and application, the preprocessing step may include several approaches. In the following a very brief description on three of the most used preprocessing techniques is done, in order to give a intuition on these techniques. For example, one of them is Discretization, this a technique where the real-valued instances of a TS are transform to symbols, the TS passes to be represented in a new symbolical space instead of its real-valued original one. This technique is relevant because the majority of the TS are real-valued [29] and a wealth of algorithms, techniques and data structures which had a lot of attention from the text retrieval community [5, 8], such as hashing, Markov models, suffix trees, etc... are not defined for real-valued sequences, only for symbolic approaches. With this transformation the range of possible techniques to be used increases, the algorithms aforementioned are then possible to apply for original real-valued TS.

Missing data is a common problem in several fields of research, the associated process of replacing this missing entries is known as imputation. Some methods can handle missing values while others can’t. The possibility to discard the TS where missing values occur is not an option when the they occur in the majority of the TS instances of the dataset, in this case, estimation methods based on know values of the TS are used to impute the missing ones.

Other preprocessing technique is noise removal. Noise is defined as a random error that can occur in the TS due to faulty measurement equipment or environmental factors for example. Binning is a method to deal with this noise. In this method the TS data is splitted into bins or buckets of equal size and then the data is smoothed by using the mean, median or the boundaries of the bucket. The objective of this smoothing is to eliminated the noise and leave only the real information of the TS.

In this line, in [29] it is pinpointed that “the high dimensionality, very high feature correlation, and the (typically) large amounts noise that characterize time series data” are the main issues facing classical machine learning algorithms when dealing with time series data.

Another desired property in the techniques proposed in this field is their interpretability, through an analysis of the algorithms it should be possible to infer the significance of the extracted pattern and rules in some statistical sense. Not only good results are desired but also an insight on the logic behind these.
Chapter 3

Time Series Classification

There are several alternative methods for time series classification. In the following a review on the TS classification categories according to [30], where the distinction of the algorithms is done by the type of discriminatory features the technique is attempting to find, is done.

Whole series In this methods, two TS are compared either as a vector (as in traditional classification) or by a distance measure that uses all the data. Two of the most used techniques of this type in TS classification are the 1-NN (first nearest neighbour) with DTW and euclidean distances. Most research effort has been directed at finding techniques that using elastic distance measures can compensate for small misalignments between series, an example of this is the derivative dynamic time warping method [24], a extension method of the DTW distance. Intervals In this one, rather than utilize the whole series, phase dependent intervals are selected. At its simplest, this involves a feature selection of a contiguous subset of time points [31]. Shapelets This family of algorithms focus on finding short patterns that can appear anywhere in the series and define its class. This patterns are commonly called shapelets. A class is then distinguished by the presence or absence of one or more shapelets somewhere in the series [32]. Dictionary based Some problems are distinguished by the frequency of repetition of subsequences rather than by their presence or absence. Dictionary-based methods form frequency counts of recurring patterns, then build classifiers based on the resulting histograms [33]. Combinations This type of algorithms combines two or more of the above approaches into a single classifier [34]. Model-based Model-based algorithms fit a generative model to each series then measure the similarity between series using similarity between models. Some of the approaches used include fitting auto-regressive models, hidden Markov models and kernel models [35–37].

In addition, it is relevant to regard that has long been known that Dynamic Time Warping (DTW) is superior to Euclidean distance for classification and clustering of time series [22], and the combination of DTW and the $k$-nearest neighbor classifier is known to be a very efficient approach as a golden standard for TS classification in the last years [25, 38–43].

Following in this chapter an introduction to TS representations methods is done and in this line the PAA and the SAX method are described. Then four univariate classification methods that implement the SAX algorithm are described and analyzed, in the following section they are asserted experimentally and
conclusions about them and SAX are withdraw. Then the MTS classification is approached, two state-of-the-art methods are presented, described and in the end their behaviour is evaluated experimentally.

### 3.1 TS Representations

As with most problems in computer science, the suitable choice of representation greatly affects the ease and efficiency of TS data mining. Many high level representations of TS have been proposed for data mining, in [44] several of the most common representations are described and discussed but till date some others more have been proposed.

There are primarily two desirable properties in a TS representation scheme. First, it must not allow any false dismissals. That is, if two series are to be found similar in the original space, they should also be found similar in the transformation space. This can be referenced as the lower bounding property, which means that we can define a distance measure on the representation-transformed data that is guaranteed to be less than or equal to the distance in the original data space. Second, due to a high number of TS datasets have a very high dimensionality, the representation must reduce the dimensionality of the similarity search problem. This is a challenging fact because all non-trivial data mining and indexing algorithms degrade exponentially with dimensionality [14].

Another relevant feature in TS representation is the discretization of the original data into symbolic strings. The fact that a representation is real valued limits the algorithms, data structures and definitions available for them. For example, in the anomaly detection field the definition of the probability of observing any particular set of wavelet coefficients cannot be meaningfully defined, since the probability of observing any real number is zero. Such limitations have lead researchers to consider using a symbolic representation of time series [45].

In the following subsections two representations methods are presented: PAA and SAX. The SAX method, is described in detail as the PAA method, this latter is referenced because it is used as a procedural step of the SAX algorithm.

#### 3.1.1 Piecewise Aggregate Composition (PAA)

In [46] the Piecewise Aggregate Composition (PAA) is presented. In this technique, the TS is divided into $N$ (method parameter) segments of equal length and then each segment is replaced with a constant value, which is the average value of the segment. Assuming that the TS $X$ of length $T$ is represented in a $N$ space by the vector $\bar{X} = \bar{x}_1, ..., \bar{x}_N$. The i-th element of $\bar{X}$ is calculated by the following equation:

$$\bar{x}_i = \frac{N}{T} \sum_{j=\frac{i-1}{N}+1}^{\frac{i}{N}} x_j.$$  

(3.1)

For simplicity and clarity, we assume that the size of the TS is divisible by $N$ but this assumption is relaxed in [45]. Then these average values are grouped in a vector, which represents the TS. The representation can be visualized as an attempt to approximate the original TS with a linear combination
of box basis functions as shown in Figure 3.1.

The TS are normalized in order to have temporal mean zero and a standard deviation of one before converting it to the PAA representation, since it is well understood that it is meaningless to compare TS with different offsets and amplitudes [29].

### 3.1.2 Symbolic Aggregate Approximation (SAX)

The Symbolic Aggregate Approximation (SAX) [14] is a TS representation method that performs discretization, dimensionality reduction and the distance measure in the symbolic space lower bounds the euclidean distance in the original TS space. The description of this method is done in the following but first is important to define the two algorithm parameters: (i) $a$ as the size of the symbolic alphabet, that represents the number of symbols which the discretized values can take and (ii) $w$ as the size of the symbolic representation, i.e., the number of segments in which the original TS is going to be divided.

As first step this technique normalizes the TS and applies the PAA method. Given the assumption that a TS normalized in this manner, to have temporal mean zero and a standard deviation of one ($\mu = 0$ and $\sigma = 1$), has a Gaussian distribution of $\mathcal{N}(0, 1)$ [45], it is possible to divide the area under the associated Gaussian curve into equal size spaces and produce symbols with equiprobability which is desirable. The breakpoints that perform this split may be determined by looking them up in a statistical table. For the sake of the explanation an example case where $a = 3$ is considered to continue the description of the algorithm through the discretization phase. Being calculated the PAA representation of the TS, all the PAA coefficients which value is below (smaller than) the smallest breakpoint are mapped to the symbol “a”, all coefficients greater than or equal to the smallest breakpoint and less than the second smallest breakpoint are mapped to the symbol “b”, etc. This idea is well demonstrated in Figure 3.2.

Regarding the distance measure in the new representation space two TS $Q$ and $C$ of the same length $T$ are considered. The MINDIST function returns the minimum distance between the original TS of the

![Figure 3.1: PAA representation of the TS $X$. The representation is given by $X'$ as a linear combination of "box" basis functions. Figure from [46].](image)
Figure 3.2: The TS (in heavy blue) is discretized by first applying the PAA technique and then using predetermined breakpoints to map the PAA coefficients into the symbols. In the example above, with $T = 128$, $w = 8$ and $a = 3$, the TS is mapped to the word baabccbc. Figure from [45].

correspondent two words:

$$MINDIST(Q, C) = \sqrt{\frac{T}{w} \sum_{i=1}^{w} \text{dist}(q[i], c[i])^2}.$$  \hspace{1cm} (3.2)

The function $\text{dist}()$ is implemented using a lookup table as illustrated in Table 3.1. The value for entry $<r,c>$ of the Table is obtained through the following function, where $\beta$ represents a breakpoint value:

$$cell_{r,c} = \begin{cases} 
0, & \text{if } |r - c| \leq 1 \\
\beta_{\text{max}(r,c)} - \beta_{\text{min}(r,c)} & \text{otherwise}
\end{cases}.$$  \hspace{1cm} (3.3)

For a given alphabet size $a$, the table needs only be calculated once, then stored for fast lookup. The intuition for the MINDIST distance function can be visualized in Figure 3.3.

An important characteristic of SAX is that it provides a lower-bounding distance measure. In [45] it is proven that MINDIST function lower-bounds the Euclidean distance. This is the only symbolic TS representation until date, in which the distance measure in the symbolic space lower bounds the distance in the original TS space. The fact that SAX guarantees this lower bound property while achieving dimensionality reduction is a big reason of its success. Beside this ones, another relevant feature of SAX is its possible numerosity reduction that is well explained in [45]. Regarding the choice of values for the method parameters, until date no formal formula or values are presented [14]. Nonetheless in this work the values used in [14, 45] are used as guideline. Using for $a$ values ranging from 3 to 20.

Table 3.1: The look up table used by the MINDIST function. This table is for an alphabet of cardinality of 4 ($a = 4$). The distance between two symbols can be read off by examining the corresponding row and column. For example, $\text{dist}(a, b) = 0$ and $\text{dist}(a, c) = 0.67$.

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>0.67</td>
<td>1.34</td>
</tr>
<tr>
<td>b</td>
<td>0.67</td>
<td>0</td>
<td>0</td>
<td>0.67</td>
</tr>
<tr>
<td>c</td>
<td>0.67</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d</td>
<td>1.34</td>
<td>0.67</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

14
Figure 3.3: A visual overview of the steps of the SAX algorithm, and the distance measures defined on them. A) The Euclidean distance between two TS can be visualized as the square root of the sum of the squared differences of each pair of corresponding points. B) The distance measure defined for the PAA approximation can be seen as the square root of the sum of the squared differences between each pair of corresponding PAA coefficients, multiplied by the square root of the compression rate. C) The distance between two SAX representations requires looking up the distances between each pair of symbols, squaring them, summing them, taking the square root and finally multiplying by the square root of the compression rate. Figure from [45].

3.2 Univariate Classification methods

In this section four univariate TS classification methods are described. This methods are referenced due to the fact that they implement the SAX algorithm and are between the most competitive methods on the univariate TS classification task [30]. After the algorithms description a section is presented where the results of an experimental evaluation of this methods is shown and discussed.

3.2.1 Bag-of-Patterns (BOP)

The Bag-of-Patterns (BOP) is a histogram representation method for TS proposed in [33]. This algorithm is based on the popular Bag-of-Words models used in the text mining and information retrieval communities [48]. In the Bag-of-Words representation, a text document is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping their multiplicity. Similarly, in the
BOP method a TS is splitted in subsequences, which a pattern from the TS is associated to, and the frequency of each pattern (or subsequence) in the TS is stored. One big feature of the bag methods is its high interpretability, for example: through a simple analysis of the representative histograms very often the user can derive what words are being used to define a certain class.

The algorithm works as follows. First the pattern dictionary for the TS database is constructed. For this a sliding window is used to extract subsequences of length \( t \) (a user-defined parameter) from the TS, each sub-sequence is normalized to have mean of zero and standard deviation of one and then discretized through SAX. A set of strings is obtained each one corresponding to a subsequence. After this process is done to all TS of the dataset, a matrix denoted by \( M \), similar to the popular vector-space-model [49] representation in the Bag-of-Words methods, can be constructed from the words associated to the subsequences. Each row \( i \) of the matrix \( M \) represents a SAX word (or a pattern) from the pattern dictionary, which is the collection of possible SAX string. Given \( a \) and \( w \) as parameters for SAX the number of possible words is given by \( a^w \). Each column \( j \) represents a TS from the dataset. Finally, the entries \( M_{i,j} \) denote the frequency of the word \( i \) in the TS \( j \). A visual representation of an \( M \) example can be seen in Figure 3.4.

After this process is applied, each TS is now represented as a histogram of SAX words that take in account global and local structures of the TS. To compute the similarity between two TS in this representation any applicable distance measures can be used. This is done by comparing the frequencies (or the histograms) of patterns in the TS. For example the Euclidean distance \( D \) between \( Q \) and \( C \), with \( j = 1 \) and \( j = 2 \) respectively, in this new space would be given by the following function:

\[
D(Q, C) = \sqrt{\sum_{i=1}^{a^w} (M_{i,1} - M_{i,2})^2}
\]

In [33] several distance measures are compared in a classification task concerning several datasets, it is concluded that the euclidean distance outperforms the others distances, based on this, the euclidean distance is the one that is used in this work. The BOP alone is a representation method but it is often used as classifier together with the 1-NN algorithm due to its very good results in classification tasks [33], in this work when referencing to it alone (not as a process step in other methods), it is the classifier that is being referenced. One factor that is important to have in mind using this algorithm is, since this approach determines similarity through the structure of the data, the TS should be long enough that the structures (or lack of them) could be meaningfully captured and summarized. For short TS this method is not recommended. In [50] is proposed a multivariate TS extension of the BOP representation method, due to the lack of information details on the implementation of this algorithm this method is not regarded in this work but it is pointed as a potential method for further investigations regarding this work proposal.

### 3.2.2 SAX-VSM

The SAX-vector-space-model (SAX-VSM) classification method proposed in [51] is very similar to the aforementioned BOP method. While BOP builds a bag for each TS, SAX-VSM constructs a bag for each
class. The representation is called SAX-VSM due to using, like in the BOP method, the SAX algorithm to discretize subsequences of the TS and because, now different to BOP, the vector-space-model [49] is exactly used as in the information retrieval community.

The algorithms starts exactly like BOP, but in the construction of matrix $M$ each column represents a class. So now an entry of matrix $M$ denotes the frequency of a SAX word in a class, instead of the frequency of a SAX word in a TS as in BOP. After $M$ is completed and all TS of the training set are processed a $tf * idf$ weighting scheme is applied to it in order to change the frequency values into a weight coefficients. Regarding to the variable $tf$ (term frequency), the concept "term" refers to the number of times a word appears in a class (what in bop was designated by a pattern) and $idf$ (inverse document frequency) is the inverse of the number of classes a word appears in. So the $tf * idf$ weight value for a term in a class bag is given by:

$$tf * idf(tf, df) = \begin{cases} \log(1 + tf) \cdot \log\left(\frac{c}{df}\right) \\ 0, otherwise \end{cases}$$

(3.5)

where $c$ is the total number of classes. After all weights are calculated, the algorithm is ready to start classifying unlabeled TS. For this the unlabeled TS is converted in a term frequency vector (BOP representation) and the Cosine similarity is calculated between it and the class frequency vectors, the TS is labelled in accordance to the class vector that maximizes this similarity distance. A visual representation of the entire process is represented in Figure 3.5.

The Cosine similarity between two vectors, $a$ and $b$ is based on their inner product and is given by the following function:

$$Cosine\_similarity(a, b) = \frac{a \cdot b}{\|a\| \cdot \|b\|}$$

(3.6)

For the same reasons of the BOP method this algorithm has an high interpretability of its results, is usage is not recommend for short TS and in [51] is demonstrated is robustness to noise.
Firstly, the labeled TS (on the top left of the figure), through the SAX discretization, are used to construct the matrix $M$ where each class is represented as a histogram of SAX words. Then frequency values of $M$ are changed to the $tf \cdot idf$ coefficient values. For the classification of the unlabeled TS (on the right top of the figure) the TS is transformed to a term histogram vector (BOP representation) and then is assigned the label of the class weight vector that maximizes the cosine distance to it. Figure from [51].

3.2.3 Fast Shapelets (FS)

The shapelet concept refers to a TS subsequence that is discriminatory of a class membership, in Figure 3.6 a visual representation of it in purple is shown. Two big features of this concept are: it generalizes the NN algorithm, a state-of-the-art technique in TS classification as referenced before, to a eager decision tree like classifier that allows a big improvement on classification performance time and also, they are very interpretable, giving a very clear insight to the user of what is defining the difference between classes. In the original algorithm proposed in [52] these ones are used as the splitting criteria for a decision tree classifier. Following a briefly description is made on how a shapelet is selected.

When a candidate shapelet is up for analysis, firstly the distances between this subsequence and all TS from the training dataset are calculated. The distance between a subsequence $S$ and a TS $X$ is defined as the minimum distance between between $S$ and any subsequence of $X$ that has the same length as $S$. All TS are then placed on an orderline according to the calculated distance. After this orderline is created the split point, separation gap and associated information gain are calculated for the candidate shapelet. All candidates shapelets (of all possible sizes) are processed in this manner. The one which is associated the most information gain will be set as the discriminant shapelet. In Figure 3.6 a visual example of this approach is shown. This approach is extended to multiclass situations through framing it as a decision tree. For a more detailed description of this algorithm the reader is forwarded to [52].

Alas, the big problem with this method is the fact that it takes a long time to run. In [52] it has proposed a speedup technique before the brute force shapelet search algorithm that uses a cheap-to-compute upper bound of the information gain and uses it to prune some candidates. Even with this technique the execution time remained a problem being an intractable method in large datasets. In
response to this in [32] the Fast Shapelets algorithm was proposed, a method that extends the approach in [52] speeding up the shapelet discovery by exploiting a random projection technique [10] on SAX representations. In short, instead of a full enumerative search at each node the shapelets are discretized and approximated. For each shapelet possible length, a dictionary of SAX words is formed and its dimensionality is reduced through random projection technique [10], several projections are done and a frequency count histogram is constructed for each class. Then, a score for each word is calculated based on how well the histograms discriminate the TS between classes. Finally the k-best SAX words are chosen and then mapped back to the original shapelets that are evaluated using the information gain in the same way as in [52]. This extension reduces the possible shapelets analyzed in [52] which results in a considerable reduction in the execution time of the algorithm with, as shown in [32], almost no difference in the accuracy of the classification tasks. A feature that is worth note is the fact that this is a heuristic algorithm that does not guarantee to find the same shapelet every run.

3.2.4 DTW Features

As referenced before, the DTW distance in combination with first nearest neighbor (1-NN) algorithm has proven itself to a very strong method on TS classification tasks. In [13] the DTW Features method has proposed, this exploits the DTW distance strength on this task but on the contrary of the 1-NN approach where the distance is used directly to find neighbors for a TS, in this algorithm the DTW distance is used to create new features that are further given to a support vector machine (SVM), in this advanced machine learning method a complex function is specifically tuned to maximize the classification accuracy over the training data. Given that the DTW is a distance measure between two TS, does not lend itself to be used as a feature representing a TS. So working around this, it is considered as feature of a TS its DTW distance to a given instance of the TS training dataset. Formally, considering $X$ as a TS example, and the training dataset $D = \{Q_1, Q_2, \ldots, Q_n\}$ the feature vector, $\text{FEATURE}_{DTW}(X)$, of $X$ will be given by:
This is the base feature vector of this algorithm but in [13] several others feature vectors are proposed, for example, using the euclidean distance or variations of DTW. Several features vectors can be used concatenating all of them into a single one in the end.

3.2.5 Univariate Experiments

Having knowledge about several state-of-art classification and discretization methods and with the endeavor to test the performance of the SAX method with TS classification tasks, the algorithms referenced before are going to be tested with benchmark datasets of the community of TS classification presented in [53]. In this evaluation, datasets with different proprieties are used: short or long TS, a ranging number of classes, different sizes on the train and test sets and with origin from different sources, i.e, ECG, images, simulations and on, but two proprieties are common to all of them, all are univariate and numeric.

In these tests and throughout this work the algorithms for 1-NN with euclidean, DTW and SAX representation used were based on the implementations of the tslearn framework [54].

SAX representation comparison with Euclidean distance on classification tasks

The first analysis that was done had the objective of confirming the results of SAX presented in [45]. In this document one of the ways of asserting the SAX behaviour was through classification tasks. With the usage of the first nearest-neighbour (1-NN) classification algorithm the accuracy of the classifications done was compared using the SAX representation and its symbolic distance measure \(MINDIST\) against the euclidean distance with the original, non-discretized, TS. The experiment was run in 22 datasets that are available in [53]. In regard to the value of the algorithm parameters: \(w\) (length of the symbolic TS) is set in a way that the dimensionality reduction takes, on average, a rate of \(1/4\) and \(a\) (alphabet size) is set to 10. In [45] the authors elaborate on the process of choosing the parameters values and the cross validation technique used, the reader is directed there if more information about the characteristics of the experiment is desired.

With the objective of trying to confirm these results and assert the SAX behaviour through a classification task, the following experienced was reproduced. In the tests done cross validation was not used, the \(w\) parameter was used in a way that for all datasets it achieves a rate of dimensionality reduction of \(1/4\) exactly and the alphabet size used was of \(a = 10\). It is assumed that the impact of differences on the \(w\) parameter and not using cross validation will not be significant on the results. In addition, the DTW distance, also with the 1-NN classifier, was included in the experiment of this project in order to compare the behaviour of SAX to this benchmark classifier. The results of the experiment are shown in Table 3.2.

As in [45] it is also summarized the results by plotting the accuracy for each dataset as a 2-dimensional point: (Euclidean\_accuracy, SAX\_accuracy). If the point falls within the lower triangle, then Euclidean is more accurate than the SAX, the opposite for the upper triangle. This plot is presented in Figure 3.7.
Figure 3.7: Accuracy rates for the Euclidean distance and SAX representation with the 1-NN classifier on 22 datasets. The upper region (upper white triangle) represents the zone where SAX representation is more accurate than the Euclidean distance and the lower region (lower cyan triangle) the zone where the euclidean distance is more accurate than the SAX representation.

from the 22 datasets the SAX representation has a better accuracy in 8 of them (that fall on upper white triangle), for the other 14 the Euclidean distance achieves a better performance. It is clear on the plot that most of the points fall around the diagonal line (border line between the two regions), this indicate that the accuracy results with both methods tends to have similar value, this can be translated as an equivalence on the accuracy performance of the two methods on these classification tasks.

Regarding the DTW distance is clear that this one outperforms both the Euclidean distance and the SAX representation. This was to be expected, as referenced before the 1-NN with DTW distance classifier is well know for its good results on TS classifications tasks. On the 22 datasets it performs better than the Euclidean distance and SAX representation for 16 of them.

As in [45] it is concluded that the SAX is competitive with the Euclidean distance with the advantage of requiring less space due to the dimensionality reduction feature. Beside the results of the experiments in this work arrive to a slight worse results for the SAX representation than the authors in [45], nonetheless their significance is not great because the same conclusions were achieved, and the slight difference on the results can be attributed to the small difference on the $w$ parameter value used and the fact that in this implementation cross validation was not used.
Table 3.2: 1-NN comparison between Euclidean Distance, SAX representation and DTW, in TS classification.

<table>
<thead>
<tr>
<th>Dataset Name</th>
<th>Dataset Train Size</th>
<th>Dataset Test Size</th>
<th>TS Length</th>
<th>No. of Classes</th>
<th>Dataset Type</th>
<th>Euclidean 1-NN accuracy</th>
<th>SAX 1-NN accuracy</th>
<th>DTW 1-NN accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>SyntheticControl</td>
<td>300</td>
<td>300</td>
<td>60</td>
<td>6</td>
<td>SIMULATED</td>
<td>0.88</td>
<td>0.9733</td>
<td>0.9933</td>
</tr>
<tr>
<td>GunPoint</td>
<td>50</td>
<td>150</td>
<td>150</td>
<td>2</td>
<td>MOTION</td>
<td>0.9133</td>
<td>0.8</td>
<td>0.9066</td>
</tr>
<tr>
<td>CBF</td>
<td>30</td>
<td>900</td>
<td>128</td>
<td>3</td>
<td>SIMULATED</td>
<td>0.8522</td>
<td>0.8977</td>
<td>0.9966</td>
</tr>
<tr>
<td>FaceAll</td>
<td>560</td>
<td>1690</td>
<td>131</td>
<td>14</td>
<td>IMAGE</td>
<td>0.7136</td>
<td>0.78409</td>
<td>0.8076</td>
</tr>
<tr>
<td>OSULeaf</td>
<td>200</td>
<td>242</td>
<td>427</td>
<td>6</td>
<td>IMAGE</td>
<td>0.5206</td>
<td>0.5289</td>
<td>0.5909</td>
</tr>
<tr>
<td>SwedishLeaf</td>
<td>500</td>
<td>625</td>
<td>128</td>
<td>15</td>
<td>IMAGE</td>
<td>0.7888</td>
<td>0.4752</td>
<td>0.792</td>
</tr>
<tr>
<td>FiftyWords</td>
<td>450</td>
<td>455</td>
<td>270</td>
<td>50</td>
<td>IMAGE</td>
<td>0.630</td>
<td>0.41714</td>
<td>0.6901</td>
</tr>
<tr>
<td>Trace</td>
<td>100</td>
<td>100</td>
<td>275</td>
<td>4</td>
<td>SENSOR</td>
<td>0.76</td>
<td>0.6</td>
<td>1.0</td>
</tr>
<tr>
<td>TwoPatterns</td>
<td>1000</td>
<td>4000</td>
<td>128</td>
<td>4</td>
<td>SIMULATED</td>
<td>0.90675</td>
<td>0.91825</td>
<td>1.0</td>
</tr>
<tr>
<td>Wafer</td>
<td>1000</td>
<td>6164</td>
<td>152</td>
<td>2</td>
<td>SENSOR</td>
<td>0.99545</td>
<td>0.91775</td>
<td>0.9798</td>
</tr>
<tr>
<td>FaceFour</td>
<td>24</td>
<td>88</td>
<td>350</td>
<td>4</td>
<td>IMAGE</td>
<td>0.78409</td>
<td>0.71707</td>
<td>0.8295</td>
</tr>
<tr>
<td>Lightning2</td>
<td>60</td>
<td>61</td>
<td>637</td>
<td>2</td>
<td>SENSOR</td>
<td>0.7540</td>
<td>0.80327</td>
<td>0.8688</td>
</tr>
<tr>
<td>Lightning7</td>
<td>70</td>
<td>73</td>
<td>319</td>
<td>7</td>
<td>SENSOR</td>
<td>0.5753</td>
<td>0.57534</td>
<td>0.7260</td>
</tr>
<tr>
<td>ECG200</td>
<td>100</td>
<td>100</td>
<td>96</td>
<td>2</td>
<td>ECG</td>
<td>0.88</td>
<td>0.89</td>
<td>0.77</td>
</tr>
<tr>
<td>Adiac</td>
<td>390</td>
<td>391</td>
<td>176</td>
<td>37</td>
<td>IMAGE</td>
<td>0.6112</td>
<td>0.0997</td>
<td>0.6035</td>
</tr>
<tr>
<td>Yoga</td>
<td>300</td>
<td>3000</td>
<td>426</td>
<td>2</td>
<td>IMAGE</td>
<td>0.83033</td>
<td>0.7796</td>
<td>0.8366</td>
</tr>
<tr>
<td>Fish</td>
<td>175</td>
<td>175</td>
<td>463</td>
<td>7</td>
<td>IMAGE</td>
<td>0.78285</td>
<td>0.41714</td>
<td>0.8228</td>
</tr>
<tr>
<td>Plane</td>
<td>105</td>
<td>105</td>
<td>144</td>
<td>7</td>
<td>SENSOR</td>
<td>0.9619</td>
<td>0.9333</td>
<td>1.0</td>
</tr>
<tr>
<td>Car</td>
<td>60</td>
<td>60</td>
<td>577</td>
<td>4</td>
<td>SENSOR</td>
<td>0.7333</td>
<td>0.65</td>
<td>0.7333</td>
</tr>
<tr>
<td>Beef</td>
<td>30</td>
<td>30</td>
<td>470</td>
<td>5</td>
<td>SPECTRO</td>
<td>0.6666</td>
<td>0.6333</td>
<td>0.6333</td>
</tr>
<tr>
<td>Coffee</td>
<td>28</td>
<td>28</td>
<td>286</td>
<td>2</td>
<td>SPECTRO</td>
<td>0.75</td>
<td>0.535714</td>
<td>1.0</td>
</tr>
<tr>
<td>OliveOil</td>
<td>30</td>
<td>30</td>
<td>570</td>
<td>4</td>
<td>SPECTRO</td>
<td>0.8666</td>
<td>0.1666</td>
<td>0.8333</td>
</tr>
</tbody>
</table>
BOP, SAX-VSM, Fast-Shapelets and DTW Features comparison with state-of-the-art classification algorithms

The second analysis was done in order to study the applications and implications of SAX in classification tasks. With this goal in mind the four classification methods presented before: BOP, SAX–VSM, Fast Shapelets (FS) and DTW Features were used. All this TS classification algorithms use SAX on their process and it was desired to test their behaviour against other TS state-of-the-art classification algorithms. On the case of this methods prove to be competitive with state-of-the-art classification methods, it serves as a good motivation to expand the SAX method to multivariate TS with the idea of, in this way the referenced methods that implement SAX can by applied to multivariate TS.

In [30] a review of the state-of-the-art TS classification algorithms was done in order to set some benchmark information about some of the most used TS classification algorithms, with simplification in mind this document it is referenced as TSC in this work. In TSC 39 algorithms were run in 85 datasets, between them are the 4 algorithms that use SAX and are desired to evaluate. In [53] a site for TSC is provided where the datasets, algorithm implementations and the results obtained are presented. In the experiment of this project, the implementations of three of the four algorithms presented were used on some of the datasets in order to compare their performance to others state-of-the-art TS classification algorithms. For the DTW Features algorithm none online implementation was found and due to that the results for this method were gather from the TSC results.

In the implementations used the algorithms parameters are set through cross validation on the training data. The implementations of BOP, SAX-VSM, and Fast Shapelets were ran on 22 of the 85 datasets. In order to compare to the results of TSC a cross-validation of 100 rounds were used. The results of the experiment are shown in Table 3.3. The “Best classification result from TSC” column represents the value of the highest accuracy rate from the total 39 algorithms run for each dataset according to the TSC results. In a broad analysis of this table is possible to see that the 4 algorithms in analysis prove to be competitive with the best classifications results achieved in TSC. Some particular examples of this are for example: the SAX–VSM and FS in the TRACE dataset, the BOP in the OliveOil dataset and the DTW Features in the TwoPatterns dataset.
### Table 3.3: Comparison between BOP, SAX-VSM, Fast Shapelets, DTW Features and best classifier result from TSC, in TS classification.

<table>
<thead>
<tr>
<th>Dataset Name</th>
<th>Dataset Train Size</th>
<th>Dataset Test Size</th>
<th>TS Length</th>
<th>No. of Classes</th>
<th>Dataset Type</th>
<th>BOP</th>
<th>SAX-VSM</th>
<th>FS</th>
<th>DTW Features</th>
<th>Best classification result from TSC</th>
</tr>
</thead>
<tbody>
<tr>
<td>SyntheticControl</td>
<td>300</td>
<td>300</td>
<td>60</td>
<td>6</td>
<td>SIMULATED</td>
<td>0.92</td>
<td>0.93</td>
<td>0.92</td>
<td>0.98583</td>
<td>0.9996</td>
</tr>
<tr>
<td>GunPoint</td>
<td>50</td>
<td>150</td>
<td>150</td>
<td>2</td>
<td>MOTION</td>
<td>0.96</td>
<td>0.826</td>
<td>0.9</td>
<td>0.9636</td>
<td>0.99866</td>
</tr>
<tr>
<td>CBF</td>
<td>30</td>
<td>900</td>
<td>128</td>
<td>3</td>
<td>SIMULATED</td>
<td>0.9628</td>
<td>0.97</td>
<td>0.946</td>
<td>0.9787</td>
<td>0.9994</td>
</tr>
<tr>
<td>FaceAll</td>
<td>560</td>
<td>1690</td>
<td>131</td>
<td>14</td>
<td>IMAGE</td>
<td>0.94497</td>
<td>0.968</td>
<td>0.75562</td>
<td>0.9626</td>
<td>0.99631</td>
</tr>
<tr>
<td>OSULeaf</td>
<td>200</td>
<td>242</td>
<td>427</td>
<td>6</td>
<td>IMAGE</td>
<td>0.68595</td>
<td>0.7768</td>
<td>0.69</td>
<td>0.809</td>
<td>0.97049</td>
</tr>
<tr>
<td>SwedishLeaf</td>
<td>500</td>
<td>625</td>
<td>128</td>
<td>15</td>
<td>IMAGE</td>
<td>0.7584</td>
<td>0.68</td>
<td>0.7744</td>
<td>0.885</td>
<td>0.96864</td>
</tr>
<tr>
<td>FiftyWords</td>
<td>450</td>
<td>455</td>
<td>270</td>
<td>50</td>
<td>IMAGE</td>
<td>0.534</td>
<td>0.4153</td>
<td>0.523</td>
<td>0.74837</td>
<td>0.82072</td>
</tr>
<tr>
<td>Trace</td>
<td>100</td>
<td>100</td>
<td>275</td>
<td>4</td>
<td>SENSOR</td>
<td>0.9769</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>TwoPatterns</td>
<td>1000</td>
<td>4000</td>
<td>128</td>
<td>4</td>
<td>SIMULATED</td>
<td>0.94925</td>
<td>0.90325</td>
<td>0.69725</td>
<td>0.99952</td>
<td>1</td>
</tr>
<tr>
<td>Wafer</td>
<td>1000</td>
<td>6164</td>
<td>152</td>
<td>2</td>
<td>SENSOR</td>
<td>0.99173</td>
<td>0.99318</td>
<td>0.9625</td>
<td>0.99615</td>
<td>0.9998</td>
</tr>
<tr>
<td>FaceFour</td>
<td>24</td>
<td>88</td>
<td>350</td>
<td>4</td>
<td>IMAGE</td>
<td>0.94497</td>
<td>0.92045</td>
<td>0.909</td>
<td>0.90920</td>
<td>0.9955</td>
</tr>
<tr>
<td>Lightning2</td>
<td>60</td>
<td>61</td>
<td>637</td>
<td>2</td>
<td>SENSOR</td>
<td>0.59016</td>
<td>0.70491</td>
<td>0.459</td>
<td>0.71016</td>
<td>0.8439</td>
</tr>
<tr>
<td>Lightning7</td>
<td>70</td>
<td>73</td>
<td>319</td>
<td>7</td>
<td>SENSOR</td>
<td>0.49315</td>
<td>0.63013</td>
<td>0.09589</td>
<td>0.67123</td>
<td>0.8110</td>
</tr>
<tr>
<td>ECG200</td>
<td>100</td>
<td>100</td>
<td>96</td>
<td>2</td>
<td>ECG</td>
<td>0.84</td>
<td>0.81</td>
<td>0.8</td>
<td>0.8185</td>
<td>0.8905</td>
</tr>
<tr>
<td>Adiac</td>
<td>390</td>
<td>391</td>
<td>176</td>
<td>37</td>
<td>IMAGE</td>
<td>0.60102</td>
<td>0.38618</td>
<td>0.5473</td>
<td>0.60498</td>
<td>0.8153</td>
</tr>
<tr>
<td>Yoga</td>
<td>300</td>
<td>3000</td>
<td>426</td>
<td>2</td>
<td>IMAGE</td>
<td>0.84</td>
<td>0.7793</td>
<td>0.6846</td>
<td>0.86347</td>
<td>0.91702</td>
</tr>
<tr>
<td>Fish</td>
<td>175</td>
<td>175</td>
<td>463</td>
<td>7</td>
<td>IMAGE</td>
<td>0.92</td>
<td>0.89714</td>
<td>0.7771</td>
<td>0.93114</td>
<td>0.9762</td>
</tr>
<tr>
<td>Plane</td>
<td>105</td>
<td>105</td>
<td>144</td>
<td>7</td>
<td>SENSOR</td>
<td>0.97143</td>
<td>0.9809</td>
<td>0.9619</td>
<td>0.995809</td>
<td>1</td>
</tr>
<tr>
<td>Car</td>
<td>60</td>
<td>60</td>
<td>577</td>
<td>4</td>
<td>SENSOR</td>
<td>0.83</td>
<td>0.8</td>
<td>0.7</td>
<td>0.851333</td>
<td>0.9255</td>
</tr>
<tr>
<td>Beef</td>
<td>30</td>
<td>30</td>
<td>470</td>
<td>5</td>
<td>SPECTRO</td>
<td>0.43</td>
<td>0.46</td>
<td>0.4</td>
<td>0.546</td>
<td>0.818666667</td>
</tr>
<tr>
<td>Coffee</td>
<td>28</td>
<td>28</td>
<td>286</td>
<td>2</td>
<td>SPECTRO</td>
<td>0.96428</td>
<td>0.8571</td>
<td>0.9642857</td>
<td>0.97285</td>
<td>0.999642857</td>
</tr>
<tr>
<td>OliveOil</td>
<td>30</td>
<td>30</td>
<td>570</td>
<td>4</td>
<td>SPECTRO</td>
<td>0.9</td>
<td>0.86</td>
<td>0.83</td>
<td>0.8979</td>
<td>0.906</td>
</tr>
</tbody>
</table>
For a simpler and clearer analysis, the accuracy values from the BOP, SAX-VSM, FS and DTW Features table were individually subtracted to the TSC result for the respective dataset and the average and standard deviation of its absolute (gathered by algorithm) was calculated, these values are presented in Table 3.4. Through this table it can be seen that, on average, the accuracy values differ from a value of 0, 1 to 0, 2 and the standard deviations are low which mean that the values not tend to differ much from the average mean. Assuming the TSC values, it is concluded that the four algorithms that implement SAX are in some way competitive with the state-of-the-art TS classifiers. This serves as a good motivation to expand the SAX to multivariate TS with the idea of applying these, proven to be good, methods that implement SAX to multivariate TS with some expectations of good results.

Table 3.4: Average and standard deviation values for the difference between the best TSC accuracy value and BOP, SAX-VSM, FS and DTW Features.

<table>
<thead>
<tr>
<th></th>
<th>BOP</th>
<th>SAX-VSM</th>
<th>FS</th>
<th>DTW Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>0,12001335</td>
<td>0,141117901</td>
<td>0,197972187</td>
<td>0,070627209</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>0,118199922</td>
<td>0,126048965</td>
<td>0,167366827</td>
<td>0,072831897</td>
</tr>
</tbody>
</table>

### 3.3 Multivariate Classification methods

In this section two multivariate classification methods are described. This methods were chosen due to their good results, references, well detailed documented descriptions and implementations. The analysis on these MTS classifiers is done regarding that is desired to analyse if the MSAX representation (with the use of the 1-NN classifier) is competitive with some state-of-the-art MTS classifiers. After this description a section is presented where the results of this methods on multivariate TS classification tasks are shown and discussed.

With the introduction of several attributes through time some constrains are added to the TS classification problem. MTS are characterized not only by serial correlations (auto-correlation) but also by relationships between the attributes measured at the same time point (intra-correlation). It is desired algorithms that considers all attributes of the MTS simultaneously, rather than separately, in order to extract information contained in their intra-correlations [55, 56]. A great part of the studies on MTS similarity that make use of univariate approaches and weight the distance between each attribute to generate a final similarity measure beside not considering the relationship between attributes come across with another problem, the high dimensionality introduced by many attributes and longer series [57, 58]. In these cases it becomes difficult to compute the similarity between MTS. This is another problem that is desired to approach in methods that work with MTS.

#### 3.3.1 Time Series Bag-of-Features (TSBF)

This method for TS classification was proposed in [12]. For a clear intuition of the algorithm, first the method is explained regarding univariate TS and after that, the extrapolation for the multivariate case is asserted.
This is a feature approach based on a bag-of-features (BoF) representation that lies on discovering both global and local properties of a TS that may define its class. The BoF approach, similar to the bag's method of BOP and SAX-VSM, is one in which complex objects are characterized by feature vectors of subobjects. In this method, from each TS multiple subsequences are extracted and from this ones shape-based features such as slope, mean and variance are calculated, this features measure properties at different locations and dilations when viewed from the original TS. When global information is too rigid to represent a good feature of the TS the partition into intervals allows one to detect patterns represented by a series of measurements over shorter time segments.

Figure 3.8: Overview of the TSBF algorithm. Firstly the subsequences are extracted from the TS training instances and the features are calculated from intervals on them (left side of the figure). Then the subsequences are class labeled and a classifier generates the class probability estimates that result on the Matrix depicted on the figure. After, the histograms of the class probability estimates are constructed and concatenated in order to summarize each subsequence information (right side of the figure). Then a random forest classifier is built on this representation in order to classify the unlabeled TS. Figure from [20].

Due to the extension and complexity of the method and regarding the document space available to explain it, a brief explanation of it is made capturing his key features. Firstly, a subsequence classification problem is set which passes by extracting random subsequences from the TS of the training set. Then the shape features referenced (mean, standard deviation and slope) are withdrawn over multiple intervals of the subsequences. Now a new dataset is generated where each subsequence becomes an instance, to which the class label of the correspondent TS is assigned. Then a random forest classifier [59] is used to generate a class probability estimation for each instance of the created dataset, this estimate regards on the strength of the assignment of the class label to the subsequence instance. After
this, the codebook is generated, this one represents the frequency of predicted subsequence classes for each TS, i.e., a histogram of the class probability estimates for each class. Global TS features can be also added to the codebook on this phase in order to give more information to the final classifier. Then, on this representation is built a random forest classifier. A unlabeled TS is then classified by following the subsequence extraction, feature calculation, e first classification problem and then is given to the final random forest classifier to it predict its label. A visual overview of the algorithm is demonstrated in Figure 3.8.

To extrapolate the TSBF algorithm to a multivariate TS classification problem, the authors in [12], propose two alternatives, only the one regarded in this work is described. For a multivariate approach, each univariate TS of the multivariate TS is handled separately and TSBF is run $M$ times, being $M$ the number of variables of the TS, generating $M$ codebooks. Then the codebooks are concatenated in one codebook to represent the multivariate TS.

### 3.3.2 Symbolic Multivariate Time Series classifier (SMTS)

In [60] the symbolic multivariate time series classifier (SMTS) was proposed. This is a classification method based on a symbolic representation for MTS that considers all attributes of the MTS simultaneously in order to extract information from the relationship between them. The symbols are learned in a supervised way. This method works both for numerical and nominal values due to its use of decision trees, it can handle MTS with attributes from both types which is a pretty good feature in TS classification. In comparison to the previous TSBF algorithm this method doesn’t require pre-defined features or time intervals, as parameters it only receives two values as information regarding the usage of decision trees.

For this representation firstly the TS training dataset is concatenated and handled in a matrix $d \times T$-by-$n$ denoted by $\Phi$, here the variable $d$ represents the training dataset size. The columns of the matrix corresponds to the attributes and the rows to the time index of each MTS. Each row will be denoted by instance. Following, it is added to the matrix values their time indexes and first differences values, which provide trend information. The instance corresponding to the MTS $x_i$, $i$-th TS of the dataset, at time index $t$ is represented as:

$$[t, x_i^1(t), x_i^1(t) - x_i^1(t - 1), ..., x_i^n(t), x_i^n(t) - x_i^n(t - 1)]$$  \hspace{1cm} (3.8)

Then it also added to each instance the class label of the associated MTS. After this, a random forest tree learner [59] is applied to this representation. Through this method, interactions between the attributes values over time are detected and the $\phi$ space is splitted in order in accordance to each instance class. In this process each instance is assigned to a terminal node of the tree and then a symbol is associated to each terminal node, so being $R$ the number of terminal nodes the symbolic alphabet size is also given by $R$. Due to the time index being an attribute of the representation, the time order of the data can be incorporated in the model. Following, the instances from each MTS are summarized by the frequency over the $R$ symbols. This process is repeated in order to generate a tree ensemble, which
is done to work around the greedy nature of a single tree [59]. Each MTS is then represented by a set of
distributions of its instances over the symbols. The representation process mentioned above is visually
represented in Figure 3.9 with a simple univariate example. Then, the distribution vectors of each MTS
are concatenated and used for the classification step of the process.

![Figure 3.9](image)

**Figure 3.9:** In (a) three univariate TS, each for one class are represented in a simple plot. In (b) is
represented the decision tree learned (for $R = 5$, 5 symbols) from the TS data through the matrix $\phi$. In
the box of the terminal nodes there is represented the number of instances associated to that node from
each class and in the top of it it is displayed the class that has more instances associated to it. Each
symbol is also associated to each terminal node. In (a) it is also represented in blue the partitions of
the space associated to each symbol available from the learning of the decision tree. Finally, in the bottom,
in (c) it is displayed the symbol distribution through the series, through which is obtained the symbolic
representation of each series. Figure from [60].

For the classification step a bag-of-word approach is applied where each symbol is taken as a word.
Briefly explaining, a random Forest classifier is learned on the basis of the frequency vectors from each
MTS of training dataset. In order to classify an unlabeled TS these vectors are used in its representation
process, after its frequency representation is obtained the random forest classifier assigns it to a class.

### 3.3.3 Multivariate Experiments

Having knowledge about this two MTS classification methods and with the endeavor to confirm their good
performance, the algorithms are tested in some experimental TS classification tasks with benchmark
datasets of the community of TS classification. As in the univariate experiments before, the datasets
used have different proprieties: short or long TS, a ranging number of classes, different sizes on the train
and test sets and with origin from different sources. Nonetheless some proprieties are common to all of
them: all are multivariate with all attributes numeric, all the attributes series have the same length and
the TS length is the same way throughout all the series of the dataset. For the datasets that originally
had TS with variable length, they were processed in order to set all TS to the same length. In these
cases, for each TS of the dataset, the time points, which the time index was bigger than the value of
the length of the smallest TS in the dataset, were discarded (cutted). This dataset cut was done due to,
although the two algorithms before can process datasets with TS with different lengths, the 1-NN with
SAX representation can’t due to the similarity measure can only be used with TS of the same length.
As in this experiment it is also desired to test the performance of the direct application of SAX to MTS
(applied independently to each variable of the MTS), which is referenced as SAX_INDP. The SAX_INDP
test is done in the same way as in the univariate previous experiments, through comparison with the
1-NN with euclidean distance and DTW classifiers. This is done with the objective of testing the SAX
performance in multivariate cases which the method proposed in this work pretends to improve. The
datasets can be found in [53, 61].

![Figure 3.10: Accuracy plot of the: 1-NN Euclidean, 1-NN DTW, 1-NN SAX_INDP, TSBF and SMTS, value in the MTS classification experiment.](image)

In these tests the implementations of TSBF and SMTS used are available in [61]. For both algorithms,
the parameters were chosen in accordance to the ones used in the experiments in [12, 60]. They are
presented in order to facilitated the reproducibility of the experiments done in this work. Both algorithms
are not deterministic, so, it was used a cross-validation technique of 10 rounds. For TSBF it was used
a minimal interval length (\(w_{min}\)) of 5, minimal subsequence length factor (\(z\)) of 0.5 and a codebook bin
size (\(binsize\)) of 10. For SMTS, the number of trees for symbol generation used was 50 (\(J_{ins}\)) and a
alphabet size of also 50 (\(R\)). For both algorithms, the number of trees to train per iteration used was
50 and the tolerance level for OOB rate improvement in the random forest trees was 0.05. In regard to
SAX_INDP it was used an alphabet size of 10 and a TS length reduction ratio of 1/2 for all datasets. The
results of the experiments are presented in Table 3.5, and in Figure 3.10 the accuracy’s obtained are
shown in a simple comparison plot.

In a first broad review of the results, it is clear that the TSBF, SMTS and 1-NN with DTW prove to be stronger than the 1-NN with Euclidean and SAX_INDP. In 8 of 13 of the datasets all the first three algorithms had higher accuracy’s than the last two, singularly the 1-NN with Euclidean or SAX_INDP achieves a higher accuracy than this three algorithms but in general their accuracy’s are lower. In line with this, the following analysis, is done by comparing the TSBF, SMTS and 1-NN with DTW and then separately the 1-NN with Euclidean and SAX_INDP, due to the general proximity of their accuracy’s. In regards to TSBF, SMTS and 1-NN with DTW, the results obtained are very similar for this three methods. Nonetheless, SMTS and TSBF perform better in a significant part of the datasets (8 in 13 of the datasets) in comparison to 1-NN with DTW. SMTS and TSBF have very close tied values for great part of the datasets, beside SMTS proves a better accuracy than TSBF in 10 of 13 of the datasets, in most of these cases the difference between both accuracy’s is very small. Now regarding 1-NN with Euclidean and SAX_INDP, the Euclidean distance achieves higher accuracy in almost all datasets, and in most of these cases the difference between the accuracy’s is significant. So by these experiments it is clear that the Euclidean distance over performs the SAX_INDP in multivariate cases. In the univariate experiments described before, the SAX algorithm with an alphabet size of 10 and TS length reduction ration of 1/4, has proven to be competitive with the Euclidean distance. In these multivariate experiments for the same alphabet size and for a higher TS length reduction ration of 1/2, for what the accuracy’s expected would be greater than with a 1/4 ratio, SAX in its linear application to multivariate TS (SAX_INDP) doesn’t maintains competitive with the Euclidean distance.

Concluding, from this multivariate experiments it was seen that the 1-NN with SAX_INDP doesn’t maintains his competitiveness with the 1-NN Euclidean distance, with its performance being lower in comparison to the SAX for univariate cases. It is desired, through the method proposed in this work, to achieve a better multivariate representation with SAX, and make that deduction through it achieving superior accuracy’s on MTS classification tasks. Through this experiment it also confirmed the good performance of the TSBF, SMTS and 1-NN with DTW classifiers that can be used to assert the behaviour of the proposed method if the results obtained from it improve significantly.
Table 3.5: Comparison between 1-NN Euclidean, 1-NN DTW, 1-NN SAX, IDP, TSBF and SMTS, in MTS classification.

<table>
<thead>
<tr>
<th>Dataset Name</th>
<th>Dataset Train Size</th>
<th>Dataset Test Size</th>
<th>TS Length</th>
<th>No. of Classes</th>
<th>Number of attributes</th>
<th>1-NN Euclidean</th>
<th>1-NN DTW</th>
<th>1-NN SAX, IDP</th>
<th>TSBF</th>
<th>SMTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUSLAN</td>
<td>1140</td>
<td>1425</td>
<td>45-136</td>
<td>95</td>
<td>22</td>
<td>0.84146</td>
<td>0.829</td>
<td>0.756</td>
<td>0.4337</td>
<td>0.925</td>
</tr>
<tr>
<td>CharacterTrajectories</td>
<td>300</td>
<td>2558</td>
<td>109-205</td>
<td>20</td>
<td>3</td>
<td>0.92806</td>
<td>0.967</td>
<td>0.8346</td>
<td>0.9433</td>
<td>0.951</td>
</tr>
<tr>
<td>ECG</td>
<td>100</td>
<td>100</td>
<td>39-152</td>
<td>2</td>
<td>2</td>
<td>0.86</td>
<td>0.81</td>
<td>0.74</td>
<td>0.776</td>
<td>0.805</td>
</tr>
<tr>
<td>Epilepsy</td>
<td>137</td>
<td>138</td>
<td>206</td>
<td>4</td>
<td>3</td>
<td>0.7318</td>
<td>0.949</td>
<td>0.3</td>
<td>0.981</td>
<td>0.958</td>
</tr>
<tr>
<td>EthanolConcentration</td>
<td>261</td>
<td>263</td>
<td>1751</td>
<td>4</td>
<td>3</td>
<td>0.21674</td>
<td>-</td>
<td>0.209</td>
<td>0.376</td>
<td>0.424</td>
</tr>
<tr>
<td>LIBRAS</td>
<td>180</td>
<td>180</td>
<td>45</td>
<td>15</td>
<td>2</td>
<td>0.706</td>
<td>0.81</td>
<td>0.705</td>
<td>0.829</td>
<td>0.909</td>
</tr>
<tr>
<td>Pen digits</td>
<td>300</td>
<td>10692</td>
<td>8</td>
<td>10</td>
<td>2</td>
<td>0.935</td>
<td>0.92</td>
<td>0.65</td>
<td>-</td>
<td>0.915</td>
</tr>
<tr>
<td>RobotFailure_LP1</td>
<td>38</td>
<td>50</td>
<td>15</td>
<td>4</td>
<td>6</td>
<td>0.6875</td>
<td>0.8125</td>
<td>0.5862</td>
<td>0.814</td>
<td>0.852</td>
</tr>
<tr>
<td>RobotFailure_LP2</td>
<td>17</td>
<td>30</td>
<td>15</td>
<td>5</td>
<td>6</td>
<td>0.379</td>
<td>0.62</td>
<td>0.48</td>
<td>0.587</td>
<td>0.75</td>
</tr>
<tr>
<td>RobotFailure_LP3</td>
<td>17</td>
<td>30</td>
<td>15</td>
<td>4</td>
<td>6</td>
<td>0.3</td>
<td>0.53</td>
<td>0.5</td>
<td>0.73</td>
<td>0.767</td>
</tr>
<tr>
<td>RobotFailure_LP4</td>
<td>42</td>
<td>75</td>
<td>15</td>
<td>3</td>
<td>6</td>
<td>0.67</td>
<td>0.746</td>
<td>0.52</td>
<td>0.911</td>
<td>0.903</td>
</tr>
<tr>
<td>RobotFailure_LP5</td>
<td>64</td>
<td>100</td>
<td>15</td>
<td>5</td>
<td>6</td>
<td>0.49</td>
<td>0.5</td>
<td>0.36</td>
<td>0.662</td>
<td>0.664</td>
</tr>
<tr>
<td>StandWalkJump</td>
<td>12</td>
<td>15</td>
<td>2500</td>
<td>3</td>
<td>4</td>
<td>0.73</td>
<td>0.26</td>
<td>0.467</td>
<td>0.353</td>
<td>0.427</td>
</tr>
<tr>
<td>Wafer</td>
<td>298</td>
<td>896</td>
<td>104–198</td>
<td>2</td>
<td>6</td>
<td>0.86</td>
<td>0.935</td>
<td>0.5208</td>
<td>0.9753</td>
<td>0.966</td>
</tr>
<tr>
<td>UWaveMTS</td>
<td>896</td>
<td>3582</td>
<td>315</td>
<td>8</td>
<td>3</td>
<td>0.901</td>
<td>0.929</td>
<td>0.82</td>
<td>0.899</td>
<td>0.941</td>
</tr>
</tbody>
</table>
Chapter 4

Proposed Method

In this chapter an extension of the SAX algorithm, MSAX, to multivariate cases is presented along with some key concepts to understand the process. Briefly, this method expands the original SAX method to MTS by using a multivariate Gaussian curve to associate the TS elements to symbols. Beside the motivation referenced on the chapter before that rests on the fact of the competitiveness of some TS classification algorithms that make of use of the SAX method, this proposal is greatly motivated also by the fact that considers the relations between the variables of the multivariate TS and on the fact that maintains the usual TS feature vector structure, which does not happen in several discretization methods, one example of this is the discretization proposed in SMTS algorithm described in the chapter before, this is an important factor because several machine learning algorithms are not able to process the TS if not in is form a vector of values index by time order.

4.1 MSAX

The original SAX method only works for univariate TS. The motivating idea behind the extension being proposed is to expand it to MTS in way that it considers the relationships between the several variables of the MTS. In literature [62–64] it is seen that applications of the SAX method to MTS are done by applying the algorithm to each variable independently (SAX_INDP) in this way the MTS intra-correlations between the variables aren’t considered in the discretization process. In the MSAX the relations of the attributes are taken into account with the goal of performing a discretization that better represents the original data and achieves better results in data mining tasks. In this study the evaluation of the MSAX representation is done through classification tasks, but others type of tasks like clustering or forecasting are also well suited for the purpose.

In the original method, firstly, the TS are normalized to have a temporal mean of zero and a standard deviation of one, given the assumption that a TS normalized in this manner have a Gaussian distribution [45], a Gaussian curve is associated to it. Then the PAA algorithm is applied in order to reduce the TS length. After, the discretizing process is done by associating the TS points to equal area intervals (that represent the discreet symbols) beneath the Gaussian curve associated to the TS to be discretized.
The idea proposed in this work focuses on expanding this process to MTS by normalizing the MTS to fit a multivariate Gaussian curve and doing the discretization process by associating the TS points to equal volume spaces under the multivariate Gaussian curve.

### 4.1.1 MSAX normalization

The first step of the extension proposed is to normalize the MTS to be discretized, to this it is key to have in notion that a vector-valued random variable $X = [X_1 \cdots X_n]^T$ is said to have a multivariate normal (or Gaussian) distribution with mean $\mu \in \mathbb{R}^n$ and Covariance matrix $\Sigma \in S_{++}^n$ if its probability density function is given by:

$$f_X(x; \mu; \Sigma) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu)\right).$$

where $x$ represents an instance of $X$ [65].

A TS that fits the Equation 4.1 it is said that have a normal distribution that is represented as $X \sim \mathcal{N}(\mu, \Sigma)$.

The objective of the normalization step is to transform the MTS in order to achieve a MTS with $X \sim \mathcal{N}(0, I)$, with $X$ representing the TS. In this distribution, the Gaussian curve associated is centered in 0 and it slope is the same for all variable axis. This is key because it is by knowing the Gaussian curve proprieties that the space beneath it is going to be splitted in equal volume spaces.

The normalization transformation [66] is done by the use of the following: Let $X \sim \mathcal{N}(\mu, \Sigma)$ for some $\mu \in \mathbb{R}^n$ and $\Sigma \in S_{++}^n$. Then, there exists a matrix $B \in \mathbb{R}^{n \times n}$ such that if we define $Z = B^{-1}(X - \mu)$, then $Z \sim \mathcal{N}(0, I)$.

To obtain the matrix $B$, the following properties of symmetric matrices from linear algebra are considered [67]:

1. Any real symmetric matrix $A \in \mathbb{R}^{n \times n}$ can always be represented as $A = U\lambda U^T$, where $U$ is a full rank orthogonal matrix containing the eigenvectors of $A$ as its columns, and $\lambda$ is a diagonal matrix containing $A$’s eigenvalues.

2. If $A$ is symmetric positive definite, all its eigenvalues are positive

Since the covariance matrix $\Sigma$ of any TS must be always symmetric positive semidefinite [66], using the first property referenced above, it can be written $\Sigma = U\lambda U^T$ for some appropriately defined matrices $U$ and $\lambda$. Using the second property, it can be defined $\lambda^{1/2} \in \mathbb{R}^{n \times n}$ to be the diagonal matrix whose entries are the square roots of the corresponding entries from $\lambda$. Since $\lambda = \lambda^{1/2}(\lambda^{1/2})^T$, it is given:

$$\Sigma = U\lambda U^T = U\lambda^{1/2}(\lambda^{1/2})^T U^T = U\lambda^{1/2}(U\lambda^{1/2})^T = BB^T,$$

where $B = U\lambda^{1/2}$. So to obtain the matrix $B$ it is first computed the covariance matrix of the MTS to be transformed, then from this one the eigenvalues and eigenvectors are withdraw and from these both the matrices $U$ and $\lambda$ are set and then $B$ is computed. Obtained the matrix $B$, the MTS $X$ to
be normalized is transformed by the following computation $Z = B^{-1}(X - \mu)$ where the $Z$ is the TS $X$ transformed with the distribution of $Z \sim \mathcal{N}(0, I)$.

Due to this normalization the method can’t process MTS where a value of an attribute along the time is constant. This happens because, for cases where an attribute is constant along the time the variance associated to the series of that attribute is 0. Which leads to, when trying to calculate the covariance matrix of the MTS in analysis, this one will have zero value for the entries associated to the constant attribute. In this way the covariance matrix isn’t positive definite and the $B$ matrix couldn’t be computed due to not all eigenvalues of the covariance matrix being positive. With this constrain the method can’t advance further. Because of a certain attribute being constant along the time, its influence to the other attributes values along the time is none, also as the attribute values are constant and no variation is present along the time, no useful information regarding temporal evolution can be extracted. So before running this method with a particular dataset one has to make sure that in all its MTS, none has attributes with a constant value along the time, for those cases the method can’t process the MTS. This point has the con of reducing the size of the dataset where this special cases may occur, as with the elimination of this MTS the information associated to them is lost. As aforementioned from the attributes with constant values no useful relation between the variables is extracted but the information of the other attributes of the MTS is lost.

### 4.1.2 MSAX discretization

After the normalization, the PAA process is applied in order to reduce the MTS length, this is done to each variable individually.

The following step of the algorithm is the discretization one. Here each $n$ dimensional point of the MTS is associated to a symbol, for this association the space under the multivariate Gaussian curve that fits the normalized MTS is split in accordance with the $a$ parameter, that represents the number of symbols to use per variable.

First, the volumes associated to each symbol beneath the multivariate Gaussian curve are defined. It is key to have in mind that is desired symbols with equal probability (similarly to the original method), for this the volumes of the symbolic spaces beneath the Gaussian curve associated to the MTS must be the same. Also, the distribution for this Gaussian curve will always be $\mathcal{N}(0, I)$ due to the normalization done before, which results in a curve centered in 0 with the same descend slope for all variables axis. With this in consideration the following reasoning as used to define the volumes.

Due to the Gaussian distribution of the curve associated with the MTS been $\mathcal{N}(0, I)$, the variables of the MTS are decorrelated, as the covariance matrix of the MTS is the identity matrix. Since the probability density function of the MTS is equal to the product of the probability density function of each variable when no correlation between the variables exist [65], a Gaussian distribution of $\mathcal{N}(0, 1)$ is associated to each variable of the TS, in the same way as in the original method. Then, each Gaussian curve associated to a variable will be split using breakpoints in order to the area of each space split beneath the Gaussian to be equal for all the spaces split regions beneath this curves, this is done in
the same way the original method in accordance with the \( a \) parameter. After, the split regions under the multivariate Gaussian curve are defined through the breakpoints intersection for each variable, this results in the variable space to be split in a grid way with each partition of the grid having the same volume under the multivariate Gaussian curve which results in symbols with equal probability. Being \( n \) the number of variables of the MTS, the breakpoints intersection results in a grid of \( a^n \) partitions that correspond to an alphabet of \( a^n \) possible symbols. Finally, the points of the normalized and PAA processed MTS are mapped to the multivariate splitted space beneath the Gaussian curve associated with MTS and in accordance with the area that they belong to, the symbol associated with that area is going to replace the numerical value of the TS attributes for a certain time index. As a result of the entire process a univariate discrete TS is obtained from the numerical MTS.

For a clearer explanation, an example of the discretization process is given with a bivariate TS \( X \) where \( x_1 \) and \( x_2 \) make reference to each variable respectively. The TS \( X \) has normal distribution of \( X \sim N(0, I) \) obtained by the normalization step, in Figure 4.1 the Gaussian curve associated to this distribution is illustrated. The curve has its center in \((0, 0)\) and has the same descend slope in both variables.

As an example \( a = 3 \) is consider, where three symbols per variable are used in the discretization process. The method splits the space under the bivariate Gaussian curve into equal volumes spaces in accordance to the value of \( a \), this space split is illustrated in Figure 4.2. In this figure, the grid symbolic division, beneath the multivariate Gaussian curve associated to the MTS to discretize can be seen. To each partition of the grid a symbol is associated, which will be called final symbol. The final symbol value is obtained by the concatenation of the symbols associated to each attribute (this symbols will be designated by symbol of variable to distinguish them from the final symbols). For example, the purple partition with final symbol of \( 'aB' \) is obtained through the concatenation of the symbol of variable of the attribute that delimits the purple partition, \( x_1 = 'a' \) and \( x_2 = 'B' \). In this case upper and small letters were used to distinguish the symbol of variable of each attribute, but this can also be done by index of sequence, numbers, etc.

In Figure 4.3 is illustrated the comparison between a TS original representation and its discretization result using MSAX.

### 4.1.3 Dissimilarity definition

Having introduced this new representation, it is now important to define the distance measure on it. Regarding two symbolic univariate TS \( Q \) and \( C \) of the same length \( T \) and obtained from a multivariate TS with \( n \) attributes, the distance measure between two TS using the MSAX representation is given by the sum of the distances between each two time points for all the indexes of the TS length, multiplied by the compression ratio:

\[
MINDIST_{MSAX} (Q, C) = \sqrt{\frac{T}{m} \sum_{i=1}^{m} \text{dist}^2_{sum}(q[i], c[i])}
\]

(4.3)
Figure 4.1: Plot of the probability density function or Gaussian curve with distribution $\mathcal{N}(0, I)$, for two variables $x_1$ and $x_2$.

Figure 4.2: Illustration of the MSAX areas associated to each symbol on the $x_1,x_2$ plane, for bivariate TS with $a = 3$. Each colored area is associated to a symbol. For example, a point situated in the area in orange, the $x_1$ variable value is associated to $b$ and the $x_2$ variable value is associated to $A$. To this example point will be associated final symbol of $bA$. 
Figure 4.3: Comparison between an original TS representation and its symbolic representation obtained through the application of MSAX. The example case is the bivariate TS $X$ with length 8. The SAX parameters used are $a = 3$ and $w = 4$ achieving a length reduction ratio of $1/2$.

The distance between two final symbols of MSAX is obtained by the sum of the difference between the symbol of variable associated to each attribute:

$$\text{dist}_{\text{sum}}(Q, C) = \sum_{i=1}^{n} \text{dist}(q[i], c[i]) .$$

(4.4)

The distance between two symbols of variable (associated to the same attribute) given by the $\text{dist}$ function is obtained through the same table used in the original SAX distance, this is due to the breakpoints used for each variable are the same than the one used in the original SAX (the number and value of the breakpoints will depend on the $a$ parameter in the same way). The $\text{dist}$ function is explained with detail in subsection 3.1.2.

Summarizing the MSAX method briefly, it starts by normalizing the MTS of $n$ dimensions in analysis in order to set the temporal mean for each attribute serie to zero and the covariance matrix of the whole series to the identity matrix. Then the PAA method is applied to each attribute series in order to reduce its length to a desired one given by the $w$ parameter. Following the discretization process begins, the space beneath the multivariate Gaussian curve with distribution of $\sim \mathcal{N}(0, I)$ associated to the MTS is splitted through breakpoints in accordance to $a$ (size of the symbolic alphabet per variable) in equal volume parts, each one associated to a symbol. The space split is done in a grid way, through the intersection of the breakpoints of each variable. Then the MTS points are mapped to the space beneath the multivariate Gaussian curve and each real-valued element of $n$ dimensions of the MTS is substituted by the symbol associated to the space part in which it is located. As a result, from a real-valued MTS a univariate discretized TS is achieved.

As an alternative to the final univariate symbolic TS representation an equivalent symbolic MTS representation can be achieved and the distance measure in the symbolic space adapted to it. This can
better suit algorithms or applications in which the symbolic representation is desired to be maintained multivariate instead of univariate. For this instead of concatenating all symbols of variable in a final symbol, each variable time point value is substituted by its symbol of variable. In this situation the distance measure used is:

\[ MINDIST_{MSAX}(Q,C) = \sqrt{T \sum_{i=1}^{n} \left( \sum_{j=1}^{m} \text{dist}(q[i][j], c[i][j])^2 \right)} \]  \hspace{1cm} (4.5)

### 4.2 Method evaluation

In this section the MSAX algorithm is tested. As referenced before, in this work the type of task used to assess the method behaviour is classification. This is done through the integration of the MTS representation algorithms with the 1-NN classifier. When setting the classifier as the 1-NN, the only variable in the classification process will be the TS representation method and the associated distance measure, that is exactly what is desired to study. Due to this, in all these studies it is assumed that a classification with a greater accuracy corresponds to a better representation performed by the TS representation algorithm. In these experiments, the following reasoning is applied, MSAX is compared with the direct application of SAX to MTS (SAX\text{\textunderscore}INDP) in order to understand if the proposed method achieves a better representation of MTS. First, the method evaluation is done through the use of two case studies datasets, in these, the algorithms comparison is performed vastly in multiple experiences that study several algorithm features, in turn, this is done through the analysis of metrics like, accuracy and confusion matrix. After, the MSAX and SAX\text{\textunderscore}INDP comparison is expanded to several datasets. In the end, the proposed method is compared with the MTS classifiers studied and tested in the chapter before, where its significance in MTS classification is approached.

#### 4.2.1 Case studies

As the great difference between MSAX and SAX\text{\textunderscore}INDP lies on the fact that MSAX takes in account the intra-correlation between the MTS variables, MTS with known and high intra-correlations between variables were desirable to understand if MSAX can take advantage of these, translating into a better MTS representation than SAX\text{\textunderscore}INDP. For an experimental first approach the creation of synthetic datasets was approached. It was desired a creation of datasets were the intra-correlation between the variables of a MTS could be set through a covariance matrix. Multivariate auto-regressive models were considered but with these, a specific intra-correlation of the variables couldn’t be set for a MTS, which was the desired. Generating multivariate points from a distribution with a desired covariance matrix and grouping them to form a MTS was also thought but in these cases the points of the created MTS would have no auto-correlation and it wouldn’t be achieve a real TS but a set of white noise. So, due to these difficulties, two case studies datasets were chosen for the first experimental approach of the method. Both were selected owing to their strong and clear to analyse, intra-correlation between the MTS attributes.
PenDigits dataset

The first dataset is the pen-based recognition of handwritten digits dataset created by E. Alpaydin and Fevzi. Alimoglu at the Department of Computer Engineering in Bogazici University [68].

This dataset consists in multiple labeled samples of pen trajectories recorded. The recording contemplates the writing of digits from 0 to 9. The dataset gathers 250 samples from each of the 44 writers which make a total of 10992 samples, the train set is composed of 300 while the test set of 10962. All samples are recorded with the same pressure sensitive tablet and hardware, this tablet sends the \( x \) and \( y \) table coordinates of the pen at fixed time intervals (sampling rate) of 100 milliseconds. In order to use this data in a classification task, is desired that the digits are represented through features vectors of constant length, for this, a spatial resampling to a length of 8 is used. As result a bivariate TS of length 8 is obtained. The data is z-normalized (each TS is transformed in order to have zero mean and standard deviation of 1 for each variable through time), all attributes are integers in the range 0..100 and the class label of TS corresponds to the digit written.

Figure 4.4: In this image 1500 samples of the handwritten digits (0 to 9) from the PenDigits dataset are plotted in grey organized by the class label. Their mean is represented in black. Figure from [69].

CharacterTrajectories dataset

The second case study dataset used is the CharacterTrajectories dataset developed by Ben H Williams from School of Informatics in University of Edinburgh [70].

This dataset is similar to the one referenced above but instead of digits the MTS represent characters from the English alphabet. The dataset consists of 2858 labeled character samples, the train set uses 300 instances and the test set 2558. All the instances recorded are from the same writer and the same tablet and hardware is used. Each character is represented by a MTS of three variables, two attributes represent the \( x \) and \( y \) tablet coordinates and the other one the pen force applied on the tablet screen. Not
all the alphabet characters are used, only characters with a single ‘pen-down’ segments are considered which correspond to 20 letters. The characters were obtained from a continuous writing through a character segmentation technique based on a pen-tip force cut-off point. The data is z-normalized. Originally the TS length of the MTS instances ranged from 109 to 205 but due to the distance measure of the SAX methods only works comparing TS of the same length, all the TS were set to the minimal length of 109 cutting the higher values from it.

![Graphical representation of character segmentation](image)

Figure 4.5: In this image, 20 examples, one from each character of the dataset are represented. The blue section of the characters represents the pen movements in contact with the tablet and the red section the pen movements getting of the tablet (the end of the character writing). Figure from [71].

**Case studies experiments**

The first experiments done were made with the goal of checking if the MSAX could take advantage of the intra-correlations of the MTS in order to achieve a better representation of them. For this the two cases studies datasets described before were used.

The PenDigits dataset study was performed first. MSAX and SAX_INDP were used in a classification task on this dataset. Several parameters combination for alphabet size and TS length reduction ratio were tested for both algorithms, but in the end, an alphabet size of 10 and a TS length reduction ratio of 1 were chosen due to being the parameters values that achieve the highest accuracy results for MSAX algorithm. The accuracy obtained for MSAX was 0.861 and for SAX_INDP 0.91. Just by the comparison of these accuracy’s MSAX seems not competitive with SAX_INDP, nonetheless the confusion matrices will be analyse in order to understand if for the digit classes with high correlations features MSAX can take advantage of it. After the classification results were obtained, the confusion matrices for each algorithm were draw, these can be seen in Figure 4.6.

For a confusion matrix comparison analysis, the correlations matrices for each class were draw with the objective to understand the intra-correlations features of each class. The correlation matrix
represents the correlation between all the variables at each time point for all the MTS of a specific class label. Each matrix cell represents the Pearson correlation coefficient between two variables at a certain time point. The Pearson correlation coefficient ranges between $-1$ and $1$, where $1$ represents a total positive linear correlation, $0$ is no linear correlation and $-1$ is total negative correlation. The idea behind the use of these matrices is to evaluate each class intra-correlations features through the presence (or not) of cell entries areas with high values (near $1$ or $-1$), which in turn correspond to high correlations features.

In Figure 4.7 the correlations matrices for several classes of the PenDigits dataset are plotted. The axis label reading is done in this way: the PenDigits dataset is composed of bivariate MTS of length 8, so for example, $X_{4,2}$ represents the second variable at the fourth time point index and $X_{8,1}$ represents the first variable at the eighth time point index. The matrices are represented through heat maps with a cyclic color bar so that for higher (through red) and for lower coefficients values (through blue) the color goes darker, which indicates high linear correlation. The values near $0$ are brighter and whiter, that in turn indicate low linear correlation.

The first aspect to point in the comparison analysis of the confusions matrices is that for 9 of the 10 classes SAX_INDP achieves a better prediction. Only for the 0 digit class MSAX achieves a better prediction, in Figure 4.7 the correlation matrix for this class can be observed. No specific area of correlation can be identified (in comparison to the digit 1 class matrix, where the middle red correlation area is well defined) but from the high presence of dark colors along the matrix, this class can be associated with high intra-correlations features. In contrast with, for example, the digit class 4 where a high presence of whiter colors along the matrix indicates a class with lower intra-correlations features.

Following this reasoning, the digit 5 class that in the same figure presents a very dark correlation matrix would make sense to also achieve a better prediction with the MSAX algorithm. But as it can be seen in the confusion matrices figure this doesn’t happen. So from this experiment is difficult to draw a conclusion if MSAX can take advantage from the MTS intra-correlations. Nonetheless, one clear aspect...
Figure 4.7: Correlation matrices of several classes of the PenDigits dataset. The PenDigits dataset is composed of bivariate MTS of length 8, so for example, $X_{4,2}$ represents the second variable at the fourth time point index.
that can be observed from the confusion matrices figure is that, constantly, MSAX prediction enhances the errors of SAX_INDP. For example, in the digit class 2 this can be easily observed, MSAX incorrectly labels some digit 2 class instances but always confusing them to the digits 1 or 7, that is exactly the same confusion observed with SAX_INDP but with a great number of occurrences.

Following, the CharacterTrajectories dataset was submitted to the same analysis. Both MSAX and SAX_INDP were used in a classification task on this dataset. The parameters chosen were an alphabet size of 10 and a TS length reduction ratio of 1/2. The accuracy obtained for MSAX was 0.745 and for SAX_INDP 0.93. In Figure 4.8 the confusion matrices for MSAX and SAX_INDP are presented.

![Confusion matrices for MSAX and SAX_INDP](image)

Figure 4.8: Confusion matrices of the classification results of 1-NN with MSAX and SAX_INDP (with an alphabet size of 10 and a TS length reduction ratio of 1/2) for the CharacterTrajectories dataset.

For the correlation matrices of the CharacterTrajectories classes, from the 3 attributes of the MTS, only the first two attributes that represent the x and y trajectories of the characters were used for the correlation matrices analysis. The pen tip force attribute was discarded in order to the analysis staying similar to the one done with the PenDigits dataset, focusing on the x and y digits trajectory correlations. In Figure 4.9 the correlations matrices of some classes of the CharacterTrajectories dataset are presented. An aspect that is worth to mention is that the first three times points values of the CharacterTrajectories MTS instances have 0 value, this corresponds to the time where the writing pen still didn’t touch the tablet. This makes the values associated to this time points being 0 and appear white in the correlations matrices, which in turn justifies the first white column and line of the correlations matrices of this dataset.

The first aspect to point in the analysis of the confusions matrices is that for all the classes, expect the classes p and q where MSAX and SAX_INDP achieve an equal prediction, SAX_INDP achieves a better prediction. For this reason, this two classes will be analysed. In Figure 4.9 the correlation matrices for this classes can be observed. Some specific area of correlation can be identified and from the high presence these dark colors areas along the matrix, these classes can be associated with high
Figure 4.9: Correlation matrices of several classes of the CharacterTrajectories dataset. For these matrices only the $x$ and $y$ trajectory representing attributes of the CharacterTrajectories dataset were used. So the MTS used are bivariate MTS of length 109, so for example, $X_{6}$ represents the variables at the sixth time point index. Not shown in the plot due to extension reasons, the two points following the $X_{6}$ axis label tick represent $X_{61}$ and $X_{62}$.
intra-correlations features. For example, comparing it to the character class \( c \) where a high presence of whiter colors along the matrix indicates a class with lower intra-correlations features.

Following this reasoning, the character \( v \) class that in the same figure presents a very dark correlation matrix would make sense that would also achieve a competitive or better prediction with the MSAX algorithm. But as it can be seen in the confusion matrices figure this doesn’t happen, with SAX, IND\( P \) achieving a better prediction. So from this experiment the same conclusion as in the PenDigits study is arrived, is difficult to draw a conclusion from these experiments if MSAX can take or not advantage from the MTS intra-correlations.

After the study of the confusion matrices it was desired to study the influence of the algorithms parameters in the representation obtained. For this, the accuracy of MSAX and SAX, IND\( P \) was studied on the case studies datasets. In this one the parameters of the algorithms: alphabet size and length of the symbolic TS (associated to the TS length reduction ratio), were varied in order to understand its influence in the TS representation achieved. For the alphabet size influence study, the TS length reduction ratio was fixed as \( \frac{3}{4} \) and the alphabet size was ranged from 5 to 20. In the variation of the TS length reduction ratio case, the alphabet size was fixed as 5 and the TS length reduction ratio was ranged from \( \frac{1}{4} \) to 1. The obtained values are presented in Figure 4.11.

![Figure 4.10: Accuracy against alphabet size and TS length reduction ratio](image-url)

(a) Accuracy against alphabet size (b) Accuracy against TS length reduction ratio

Figure 4.10: For the PenDigits and CharacterTrajectories datasets, the accuracy of 1-NN with MSAX and SAX, IND\( P \) is plotted, in the left graph, against the alphabet size values of 5, 10, 15 and 20 for the fixed TS length reduction ratio of \( \frac{3}{4} \). In the right plot the accuracy is plotted against the TS length reduction ratio of \( \frac{1}{4} \), \( \frac{1}{2} \), \( \frac{3}{4} \) and 1 for a fixed alphabet size of 5.

Starting by the analysis of the alphabet size variation plot. Regarding SAX, IND\( P \), in both datasets, as the alphabet size increases the accuracy also increases, stagnating after second \( a \) value around a value of 0.92. For MSAX, in both datasets the same increases is seen along the first two values but then the accuracy starts to decrease in the last two points, achieving a better accuracy with low alphabet
size values. Comparing both algorithms, SAX_INDP achieves a better accuracy for all $\alpha$ values, in the PenDigits dataset the accuracy achieved besides lower, is still competitive for the first two alphabet size values but in the CharacterTrajekories dataset the difference is significant for all points.

Now regarding the TS length reduction ratio variance plot. First it is relevant to point out that for the PenDigits dataset a TS length reduction ratio of $1/4$ was not used due to the small length of 8 of the MTS. With the usage of a TS length reduction ratio of $1/4$ the MTS is reduced to two temporal points which resulted in very low results in these experiments, which justifies itself due to the significant reduce of information. So due to this, the lowest TS length reduction ratio used for the PenDigits dataset was of $1/2$. In the PenDigits dataset as the TS length reduction ratio is increased the accuracy is also increased, this is clearly seen for both algorithms, nonetheless the SAX_INDP achieves a constant superior accuracy for all points. In the CharacterTrajekories dataset the behaviour of both algorithms is constant as the TS length is varied, regarding to the comparison between the algorithms the same as before is present the, SAX_INDP accuracy proves constantly superior to MSAX.

Concluding, from these two experiments the key ideas drawn were: in general SAX_INDP proves a better accuracy than MSAX; for SAX_INDP an increasing alphabet size tends to an higher accuracy, in the MSAX case this increase is only observed for the lowest alphabet size values, starting to decrease in the last two, this decreasing behaviour is not understood because an increase of the alphabet size, which in turn, is an increase of the information carried by the MTS representation, should lead to a better representation and increasing accuracy; regarding to the TS length reduction ratio the behaviour is the same for both algorithms, in one dataset there is an increase in the accuracy as the ratio increases and in the other there is no variation, so due to this no conclusions are withdraw.

![Run time against alphabet size](image1.png)

![Run time against TS length reduction ratio](image2.png)

Figure 4.11: The run time (in seconds) of the 1-NN with the MSAX and SAX_INDP representations is plotted, in the left graph against the $\alpha$ values from 3 to 20 for the fixed TS length reduction ratio of $3/4$. In the right graph, against a TS length reduction ratio of $1/4, 1/2, 3/4$ and 1 for the fixed alphabet size of 5.
In terms of performance of the algorithm it was also desired to compare MSAX and SAX_INDP in terms of algorithm running time. For this task the variation of the algorithms parameters and its influence on the algorithm running time was also studied. Similar to the tests presented before, it is plotted the run time (in seconds) against each parameter variation. The obtained values are presented in Figure 4.11.

First aspect to point out is the fact that the MSAX running time is always superior to SAX_INDP, this difference is seen in both plots in the two datasets, although its more significant in the PenDigits dataset. In regards to the influence of the algorithm parameters in the algorithm running time, the increase of the alphabet size in both algorithms and datasets doesn’t lead to any significant increase in the running time. In relation to the TS length reduction ratio its influence is clear, as the length of the final symbolic TS increases the algorithm takes more time to run. From these two experiments it is concluded that the MSAX is inferior to SAX_INDP in terms of algorithm running time performance and that the TS length reduction ratio has a great influence on the algorithm running time and that the same behaviour is not observed regarding the alphabet size parameter.

4.2.2 Experiments on benchmark datasets for MTS classification

After the analysis of the method through the case studies described before, an analysis with a broader range of 15 datasets datasets was done. The datasets used are the same ones that in the multivariate experiments of subsection 3.3.3 of chapter 3, the reader is redirected there for a more detailed description about them. In these experiences the algorithms were tested with an extensive parameter variation. The alphabet size was ranged from 5 to 20 with an interval of 5 and the TS length reduction ratio from \( \frac{1}{4} \) to 1 with an interval of \( \frac{1}{4} \). Both MSAX and SAX_INDP were ran on each dataset for each possible parameter combination. The accuracy results for the two algorithms are presented in Figure 4.12. Due to the extension of datasets, in this figure only some datasets are presented, these were chosen due to presenting more common and significant trends, nonetheless the graphs and results values for the rest of the datasets are present in Appendix A. Following an analysis on some datasets values from Figure 4.12 is done.

In the first dataset, AUSLAN, the SAX_INDP algorithm is much superior than MSAX for all parameter combinations. In regards to the parameter variation some small variations can be seen, but the accuracy is more or less constant both in SAX_INDP and MSAX. In the CharacterTrajectories dataset the behaviour is pretty similar to the one described before but in this one, MSAX is much more competitive with SAX_INDP. In the ECG dataset, MSAX is competitive with SAX_INDP achieving higher accuracy for several parameters combination. In relation to the parameter variation, for both algorithms, an increase alphabet size generates results with a better accuracy and the same is seen for the TS length reduction ratio. This is easy observed in Table 4.1 where the the accuracy values tend to be higher as the reader goes down diagonally to the right, reading the two tables. Regarding to the Epilepsy dataset, for the first alphabet size values MSAX is competitive with SAX_INDP but for the last values the accuracy values decrease achieving SAX_INDP a superior accuracy. This pattern where the MSAX accuracy values for the last alphabet size values tends to decrease, is also recurrent in others datasets. Also in this dataset
Figure 4.12: MSAX and SAX_INDP accuracy’s comparison for several parameters combinations with: AUSLAN, CharacterTrajectories, ECG, Epilepsy, Libras and PenDigits datasets.
it can be seen the accuracy of both algorithms increasing as the TS length reduction ratio increases. For the LIBRAS dataset the SAX_INDp proves superior and both algorithms present a more or less constant accuracy through the parameter variation. In the last dataset, PenDigits, the behaviour is similar to the one in the Epilepsy dataset, in the MSAX table in Table 4.2 this tendency is clearly seen for the last three values of the TS length reduction ratio where the accuracy of the first values of alphabet size increases, but for the last two decreases, the same doesn’t happen for the SAX_INDp algorithm where the increase is constant.

<table>
<thead>
<tr>
<th>len_red</th>
<th>alpha</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>0.69</td>
<td>0.72</td>
<td>0.76</td>
<td>0.79</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>0.74</td>
<td>0.8</td>
<td>0.85</td>
<td>0.87</td>
<td></td>
</tr>
<tr>
<td>0.75</td>
<td>0.69</td>
<td>0.78</td>
<td>0.83</td>
<td>0.83</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.73</td>
<td>0.83</td>
<td>0.86</td>
<td>0.84</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>len_red</th>
<th>alpha</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>0.59</td>
<td>0.72</td>
<td>0.79</td>
<td>0.71</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>0.71</td>
<td>0.76</td>
<td>0.77</td>
<td>0.79</td>
<td></td>
</tr>
<tr>
<td>0.75</td>
<td>0.68</td>
<td>0.77</td>
<td>0.84</td>
<td>0.74</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.73</td>
<td>0.8</td>
<td>0.76</td>
<td>0.79</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.1: MSAX and SAX_INDp accuracy’s for several parameters combinations with ECG dataset

<table>
<thead>
<tr>
<th>len_red</th>
<th>alpha</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>0.174</td>
<td>0.189</td>
<td>0.298</td>
<td>0.319</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>0.593</td>
<td>0.8</td>
<td>0.854</td>
<td>0.872</td>
<td></td>
</tr>
<tr>
<td>0.75</td>
<td>0.75</td>
<td>0.9</td>
<td>0.918</td>
<td>0.927</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.829</td>
<td>0.91</td>
<td>0.935</td>
<td>0.94</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>len_red</th>
<th>alpha</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>0.118</td>
<td>0.188</td>
<td>0.212</td>
<td>0.213</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>0.532</td>
<td>0.742</td>
<td>0.517</td>
<td>0.472</td>
<td></td>
</tr>
<tr>
<td>0.75</td>
<td>0.689</td>
<td>0.832</td>
<td>0.554</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.788</td>
<td>0.861</td>
<td>0.576</td>
<td>0.539</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.2: MSAX and SAX_INDp accuracy’s for several parameters combinations with PenDigits dataset

Through an analysis of the data from each dataset some significant trends were observed, the most relevant one is that in general SAX_INDp achieves a higher accuracy than MSAX, tough is some cases MSAX can still be competitive. Also, MSAX accuracy values tend to increase has TS length reduction increases, in regard to the alphabet size parameter is hard to withdraw a clear conclusion due to the variance of the obtained results, in some cases the accuracy tends to increase as the alphabet size increases but its also common in some datasets decreasing for the last two alphabet size values. The same behaviour is not observed in relation to SAX_INDp where for both parameters the accuracy tends to increase as they increase, this behaviour makes sense in the way that more information results in a higher accuracy, which in turn reveals a better MTS representation. Nonetheless is key to refer that these patterns aren’t present in all the datasets, the results obtained through out all datasets vary a lot, so even if this trends were found significant, they are not constant.

From the values presented before, for each dataset, the highest accuracy achieved from all the parameters combinations tested was selected from both algorithms and in Figure 4.13 thses values are plotted to comparison.

In the 15 datasets tested SAX_INDp achieves a significant higher accuracy in 8 of them, in 4 of them the accuracy’s obtained are very competitive and in 3 of them MSAX achieves significant higher accuracy. From these results it is clear to concluded that SAX_INDp, in general performs better than
MSAX in MTS classifications tasks, which points to the conclusion that its representation, in general, its preferable than MSAX.

In order to compare the performance of 1-NN MSAX with some state-of-the-art MTS classifiers, to the accuracy values of 1-NN MSAX and 1-NN SAX_INDP presented in the last figure plot (Figure 4.13) it was added the results of the state-of-the-art MTS classifiers studied before. As the datasets are the same, these values were withdraw from the multivariate experiment of subsection 3.3.3 of chapter 3, for a detailed description of the algorithms implementations and parameters used, the reader is redirected to this document section. The experiment results are presented in Figure 4.14.

This plot is presented in order to evaluate the significance of 1-NN MSAX on MTS classifications tasks. The comparison between: 1-NN SAX_INDP, 1-NN Euclidean, 1-NN DTW, TSBF and SMTS classifiers was already done in subsection 3.3.3 of chapter 3. By the comparison of the curves of the plot it can be seen that 1-NN MSAX behaviour is most similar to 1-NN SAX_INDP than to the others classifiers. Taking this in account the line of analysis of the last plot is enhanced, that 1-NN SAX_INDP proves to achieve an higher accuracy in the most cases. Nonetheless through the plot it can be seen that in some datasets 1-NN MSAX proves competitive and outperforms others classifiers achieving significant results. For example in the ECG, PenDigits and Wafer datasets 1-NN MSAX achieves a competitive accuracy against the others classifiers, in the EthanolConcentration, RobotFailure_LP3 and StandWalkJump datasets although not being competitive with the highest accuracy classifier it still outperforms half of the others classifiers.

Figure 4.13: Accuracy plot of the 1-NN MSAX and 1-NN SAX_INDP values in a MTS classification experiment with 15 datasets.
Figure 4.14: Accuracy plot of the: 1-NN MSAX, 1-NN SAX, INDP, 1-NN Euclidean, 1-NN DTW, TSBF and SMTS, accuracy values in a MTS classification experiment.
4.3 Method discussion

In regards to the MSAX behaviour several patterns were clearly extracted from the experiments before, others not so much. It is clear from all experiments that MSAX in general is not competitive with SAX_INDP in classification tasks, thought for some specific cases it can achieve good and competitive results. When translating the results of the experiments to the quality of the MTS representation algorithm it is concluded that SAX_INDP in general achieves a better MTS representation. Regarding to MSAX taking advantage of the MTS intra-correlations, this was not clearly seen in the cases studies datasets experiments where the classification confusion matrices and the classes correlations matrices were studied. A deeper investigation on the causes of this behaviour is pointed out as future investigation guideline. Now, concerning the the algorithm parameters influence on the quality of the TS representation, due to the great variance on the obtained results, the behaviour is not self evident. For the SAX_INDP, a higher alphabet size or TS length reduction ratio tends to a higher accuracy which in turn is associated to a MTS representation of better quality. For MSAX, the increase of accuracy associated to an increasing TS length reduction ratio can be also observed, but not so linearly as in SAX_INDP. Regarding the alphabet size influence, the most significant pattern was that for low alphabet size values, an increase as seen in SAX_INDP is also observed, but for higher values (example: 15 and 20) the accuracy starts to decrease. The cause of this behaviour would also be a relevant point to investigate in a continuous study of MSAX. In regard to the MSAX computational performance, through the experiences done it was concluded to be inferior to SAX_INDP in terms of running in both case studies datasets. Regarding the parameter influence on it, only the TS length reduction ratio prove to affect it significantly by increasing the running time as the TS length ratio increased.
Chapter 5

Conclusions and Achievements

In this work an extension of SAX for MTS, named MSAX, was proposed. Initially a study on the original SAX univariate algorithm was done, then several univariate TS classifications methods that make use of SAX were reviewed and their behaviour was analyzed. Through the results of these experiments the SAX utility was studied in order to confirm and motivate the idea of extending the method to MTS. Following, a review on some state-of-the-art MTS classification algorithms was done in order to compare MSAX in classification tasks against these.

MSAX behaviour was accessed in classifications tasks, comparing it with the SAX_INDP, the original SAX algorithm applied independently to each attribute in the MTS, and other state-of-the-art MTS classifiers. We concluded that the proposed method is overall not competitive with the SAX_INDP. Nonetheless, the obtained results have utility as benchmark values for SAX-based methods in multivariate classifications tasks. It is also noteworthy that for some datasets and specific cases, MSAX surpasses the other methods.

As future direction, MSAX could be investigated more deeply in order to understand some question that were left unanswered in this work about the MSAX behaviour, referenced before in section 4.3. Also MSAX applications could be studied in other data mining areas, such as clustering or forecasting, to check if it could be useful and achieve competitive performance with state-of-the-art methods.
Bibliography


Appendix A

MSAX and SAX_INDP comparison through a classification problem in multiple datasets

A.1 Accuracy values

<table>
<thead>
<tr>
<th>len</th>
<th>alpha</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td></td>
<td>0.756</td>
<td>0.793</td>
<td>0.793</td>
<td>0.793</td>
</tr>
<tr>
<td>0.5</td>
<td></td>
<td>0.756</td>
<td>0.756</td>
<td>0.768</td>
<td>0.768</td>
</tr>
<tr>
<td>0.75</td>
<td></td>
<td>0.768</td>
<td>0.756</td>
<td>0.756</td>
<td>0.756</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>0.78</td>
<td>0.756</td>
<td>0.768</td>
<td>0.768</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>len</th>
<th>redu</th>
<th>alpha</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td></td>
<td>0.146</td>
<td>0.219</td>
<td>0.183</td>
<td>0.17</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td></td>
<td>0.207</td>
<td>0.207</td>
<td>0.158</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.75</td>
<td></td>
<td>0.183</td>
<td>0.17</td>
<td>0.146</td>
<td>0.158</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>0.122</td>
<td>0.17</td>
<td>0.207</td>
<td>0.195</td>
<td></td>
</tr>
</tbody>
</table>

Table A.1: AUSLAN dataset.

<table>
<thead>
<tr>
<th>len</th>
<th>alpha</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td></td>
<td>0.889</td>
<td>0.928</td>
<td>0.932</td>
<td>0.934</td>
</tr>
<tr>
<td>0.5</td>
<td></td>
<td>0.9</td>
<td>0.93</td>
<td>0.935</td>
<td>0.935</td>
</tr>
<tr>
<td>0.75</td>
<td></td>
<td>0.859</td>
<td>0.919</td>
<td>0.929</td>
<td>0.935</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>0.9105</td>
<td>0.927</td>
<td>0.934</td>
<td>0.935</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>len</th>
<th>redu</th>
<th>alpha</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td></td>
<td>0.729</td>
<td>0.745</td>
<td>0.706</td>
<td>0.682</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td></td>
<td>0.729</td>
<td>0.745</td>
<td>0.745</td>
<td>0.699</td>
<td></td>
</tr>
<tr>
<td>0.75</td>
<td></td>
<td>0.702</td>
<td>0.727</td>
<td>0.687</td>
<td>0.657</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>0.731</td>
<td>0.743</td>
<td>0.747</td>
<td>0.709</td>
<td></td>
</tr>
</tbody>
</table>

Table A.2: CharacterTrajectories dataset.
### Table A.3: Epilepsy dataset.

<table>
<thead>
<tr>
<th>len_red</th>
<th>alpha</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>0.304</td>
<td>0.369</td>
<td>0.398</td>
<td>0.442</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>0.725</td>
<td>0.696</td>
<td>0.717</td>
<td>0.7</td>
<td></td>
</tr>
<tr>
<td>0.75</td>
<td>0.732</td>
<td>0.754</td>
<td>0.739</td>
<td>0.754</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.717</td>
<td>0.76</td>
<td>0.754</td>
<td>0.754</td>
<td></td>
</tr>
</tbody>
</table>

### Table A.4: EthanolConcentration dataset.

<table>
<thead>
<tr>
<th>len_red</th>
<th>alpha</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>0.247</td>
<td>0.22</td>
<td>0.258</td>
<td>0.25</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>0.236</td>
<td>0.236</td>
<td>0.243</td>
<td>0.247</td>
<td></td>
</tr>
<tr>
<td>0.75</td>
<td>0.247</td>
<td>0.217</td>
<td>0.262</td>
<td>0.243</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.243</td>
<td>0.224</td>
<td>0.247</td>
<td>0.25</td>
<td></td>
</tr>
</tbody>
</table>

### Table A.5: LIBRAS dataset.

<table>
<thead>
<tr>
<th>len_red</th>
<th>alpha</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>0.644</td>
<td>0.733</td>
<td>0.75</td>
<td>0.75</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>0.683</td>
<td>0.717</td>
<td>0.733</td>
<td>0.744</td>
<td></td>
</tr>
<tr>
<td>0.75</td>
<td>0.705</td>
<td>0.722</td>
<td>0.728</td>
<td>0.717</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.717</td>
<td>0.711</td>
<td>0.728</td>
<td>0.728</td>
<td></td>
</tr>
</tbody>
</table>

### Table A.6: LP1 dataset.

<table>
<thead>
<tr>
<th>len_red</th>
<th>alpha</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>0.448</td>
<td>0.448</td>
<td>0.448</td>
<td>0.483</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>0.379</td>
<td>0.483</td>
<td>0.483</td>
<td>0.483</td>
<td></td>
</tr>
<tr>
<td>0.75</td>
<td>0.276</td>
<td>0.241</td>
<td>0.276</td>
<td>0.276</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.241</td>
<td>0.172</td>
<td>0.172</td>
<td>0.172</td>
<td></td>
</tr>
</tbody>
</table>

### Table A.7: LP2 dataset.

<table>
<thead>
<tr>
<th>len_red</th>
<th>alpha</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>0.414</td>
<td>0.276</td>
<td>0.448</td>
<td>0.276</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>0.483</td>
<td>0.379</td>
<td>0.379</td>
<td>0.414</td>
<td></td>
</tr>
<tr>
<td>0.75</td>
<td>0.414</td>
<td>0.414</td>
<td>0.345</td>
<td>0.379</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.2</td>
<td>0.241</td>
<td>0.414</td>
<td>0.345</td>
<td></td>
</tr>
</tbody>
</table>
Table A.8: LP3 dataset.

<table>
<thead>
<tr>
<th>len</th>
<th>red</th>
<th>alpha</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>0.533</td>
<td>0.5</td>
<td>0.566</td>
<td>0.533</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>0.466</td>
<td>0.5</td>
<td>0.5</td>
<td>0.566</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.75</td>
<td>0.366</td>
<td>0.4</td>
<td>0.333</td>
<td>0.366</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.266</td>
<td>0.233</td>
<td>0.233</td>
<td>0.266</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table A.9: LP4 dataset.

<table>
<thead>
<tr>
<th>len</th>
<th>red</th>
<th>alpha</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>0.613</td>
<td>0.613</td>
<td>0.626</td>
<td>0.653</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>0.72</td>
<td>0.706</td>
<td>0.733</td>
<td>0.72</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.75</td>
<td>0.7</td>
<td>0.733</td>
<td>0.706</td>
<td>0.693</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.64</td>
<td>0.733</td>
<td>0.733</td>
<td>0.733</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table A.10: LP5 dataset.

<table>
<thead>
<tr>
<th>len</th>
<th>red</th>
<th>alpha</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>0.32</td>
<td>0.4</td>
<td>0.42</td>
<td>0.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>0.33</td>
<td>0.34</td>
<td>0.33</td>
<td>0.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.75</td>
<td>0.33</td>
<td>0.45</td>
<td>0.42</td>
<td>0.45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.34</td>
<td>0.3</td>
<td>0.266</td>
<td>0.266</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table A.11: StandWalkJump dataset.

<table>
<thead>
<tr>
<th>len</th>
<th>red</th>
<th>alpha</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>0.906</td>
<td>0.9</td>
<td>0.911</td>
<td>0.918</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>0.903</td>
<td>0.9</td>
<td>0.913</td>
<td>0.9122</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.75</td>
<td>0.883</td>
<td>0.884</td>
<td>0.888</td>
<td>0.889</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.906</td>
<td>0.91</td>
<td>0.914</td>
<td>0.913</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table A.12: UWAVE dataset.

<table>
<thead>
<tr>
<th>len</th>
<th>red</th>
<th>alpha</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>0.903</td>
<td>0.9</td>
<td>0.913</td>
<td>0.9122</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>0.903</td>
<td>0.9</td>
<td>0.913</td>
<td>0.9122</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.75</td>
<td>0.883</td>
<td>0.884</td>
<td>0.888</td>
<td>0.889</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.906</td>
<td>0.91</td>
<td>0.914</td>
<td>0.913</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### A.2 Accuracy graphs

<table>
<thead>
<tr>
<th>len_red \ alpha</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>0.877</td>
<td>0.858</td>
<td>0.863</td>
<td>0.918</td>
</tr>
<tr>
<td>0.5</td>
<td>0.903</td>
<td>0.9</td>
<td>0.913</td>
<td>0.9122</td>
</tr>
<tr>
<td>0.75</td>
<td>0.863</td>
<td>0.884</td>
<td>0.888</td>
<td>0.889</td>
</tr>
<tr>
<td>1</td>
<td>0.906</td>
<td>0.91</td>
<td>0.914</td>
<td>0.913</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>len_red \ alpha</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>0.432</td>
<td>0.435</td>
<td>0.238</td>
<td>0.258</td>
</tr>
<tr>
<td>0.5</td>
<td>0.428</td>
<td>0.436</td>
<td>0.21</td>
<td>0.259</td>
</tr>
<tr>
<td>0.75</td>
<td>0.38</td>
<td>0.395</td>
<td>0.226</td>
<td>0.2636</td>
</tr>
<tr>
<td>1</td>
<td>0.43</td>
<td>0.44</td>
<td>0.244</td>
<td>0.255</td>
</tr>
</tbody>
</table>

Table A.13: WAFER dataset.

Figure A.1: MSAX and SAX_INDP accuracy's comparison for several parameters combinations with: EthanolConcentration, LP1, LP2, and LP3 datasets.
Figure A.2: MSAX and SAX_INDP accuracy’s comparison for several parameters combinations with: LP4, LP5, Standwalkjump, UWAVE and WAFER datasets.