ABSTRACT

VASHEGHANI FARAHANI, IMAN. Time Series Motifs in Practice. (Under the direction of Dr. Russell E. King and Dr. Michael G. Kay.)

This dissertation addresses two data mining tasks: *time series classification* and *anomaly detection*. The first problem is investigated from three perspectives, namely (a) by using motifs (repeated pattern in a time series) of variable lengths, (b) by studying the symbolic aggregate approximation (SAX) dimension reduction technique and its similarity measure, and (c) by utilizing two shape descriptor functions in shape dynamic time warping (shapeDTW). The first approach attempts to label a time series by finding the class signatures—the subsequences that are likely to exist in a time series, if it belongs to a certain class. Motif instances of variable lengths are found by a new motif discovery method called *deep search*. In the second approach, the trend of SAX letter segments in addition to the segment average is considered. A new distance measure between any two time series is found based on the minimal distance of their SAX words and trends. In the third approach, two new shape descriptor functions namely *normalized subsequence* and *local derivatives* are employed to determine the optimal alignment between any two time series. All three models are applied to the UCR time series datasets. Classification accuracies of the models ensure the quality of the suggested features. The decision tree classification method is used in the first approach. This classifier improved the classification accuracy of 5 out of 6 datasets, compared to one-nearest neighbor classifiers of Euclidean distance and dynamic time warping similarity measures. In three datasets, the improvement was over 10%. The second approach improved the classification accuracy of 7 out of 20 datasets. The comparison baselines in this method are one-nearest neighbor classifiers using Euclidean distance, SAX, extended SAX and SAX with trend

distance measures. In the third approach, one-sided Wilcoxon signed-rank test indicated that median error rates of one-nearest neighbor classifier of shapeDTW by normalized subsequence and local derivatives were significantly lower than that of dynamic time warping in 12 out of 24 and 13 out of 24 datasets (*p-value* less than 0.05). Moreover, among the 24 datasets, the test indicated that the median error rates of shapeDTW using normalized subsequence and local derivatives in 7 and 8 datasets, respectively, are statistically lower than those of shapeDTW using raw-subsequence. The second problem, *anomaly detection*, is studied by utilizing Markov chains. The purpose of this problem is to find segments (rather than a single data point in classic anomaly detection problems) of a time series whose behaviors do not meet the expected behavior of the underlying system. Markov chain modeling is used to learn the behavior of the consecutive subsequences on a dataset without anomalies. The trained model is used to assess how well the testing dataset suits the baseline behavior. The designed *anomaly detection* framework is capable of discovering unusual patterns in both streaming data (online) and stored data (offline). The performance of the methodology is evaluated by applying it to two datasets from different fields, electrocardiogram (ECG) data and utility usage data. The detected anomaly in the ECG data agrees with the results of the studies in the literature. For the utility usage data, the domain expert confirmed the accuracy of the results.

© Copyright 2018 by Iman Vasheghani Farahani

All Rights Reserved

Time Series Motifs in Practice

by
Iman Vasheghani Farahani

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Industrial Engineering

Raleigh, North Carolina

2018

APPROVED BY:

Dr. Thom J. Hodgson	Dr. Donald P. Warsing
Dr. Russell E. King Co-chair of Advisory Committee	Dr. Michael G. Kay Co-chair of Advisory Committee

DEDICATION

Dedicated to my lovely parents *Manizheh* and *Ata*, this is for you.

BIOGRAPHY

Iman Vasheghani Farahani is a Ph.D. student in the Edward P. Fitts Department of Industrial and Systems Engineering at North Carolina State University. He was born in Tehran, Iran in 1989. He received his B.Sc. and M.Sc. degrees in Industrial Engineering and Socio-Economic Systems Engineering from Sharif University of Technology in Tehran, Iran in 2011 and 2013, respectively. In 2014, he joined the graduate program at NC State University. He is currently working as a graduate intern in the Forecasting R&D department at SAS Institute in Cary, NC and will continue his work as a Research Statistician Developer upon graduation.

ACKNOWLEDGEMENTS

First and foremost, I am grateful for all my advisors' guidance and their confidence in me: Dr. Alex Chien, Dr. Michael G. Kay and Dr. Russell E. King, who provided opportunities for me to express my ideas and learn from my mistakes. I thank Dr. Thom J. Hodgson and Dr. Donald P. Warsing who agreed to be on my committee. Their thoughtful comments on my work helped me to improve this research.

This dissertation is the summary of my year-round internship at SAS Institute, Inc. I would like to thank the Forecasting R&D department who supported my Ph.D. education: Jerzy Brzezicki, Alex Chien, and Michael Leonard. Alex helped me to find my way and to conduct the research in a highly professional environment. This invaluable experience was not possible without his help. He was the best mentor anyone could ever ask for.

I dedicate this work to my parents Manizheh and Ata and my siblings Amir, Elham, and Ehsan. I appreciate their endless support and encouragement throughout my education. This dissertation is the result of the hard days far away from home. My wish is to see them soon.

Lastly, my friends have always been there for me, through my ups and downs, and supported me as I traveled along this journey. Their friendship is so valuable to me and so deserve my wholehearted thanks: Parinaz, Bahman, Sajjad, and Pooya.

TABLE OF CONTENTS

List of Tables . . . . . **vii**

List of Figures . . . . . **ix**

Chapter 1 Introduction **1**
1.1 Background 1
1.2 Time Series Motif 3
1.3 Similarity Measures 5
1.3.1 Euclidean Distance 5
1.3.2 Dynamic Time Warping 5
1.3.3 Hamming Distance 8
1.4 Dimension Reduction 9
1.4.1 Piecewise Aggregate Approximation 10
1.4.2 Symbolic Aggregate Approximation 11
1.5 Motif Applications 14
1.5.1 Discovering Motifs 14
1.5.2 Anomaly Detection 15
1.5.3 Temporal Data Feature Generation 16

Chapter 2 Time Series Classification by Motifs **18**
2.1 Introduction 18
2.2 Background 21
2.3 Methodology 22
2.3.1 Motif Discovery Algorithms 25
2.3.2 Time Series Classification 31
2.4 Numerical evaluations 34
2.5 Conclusion 34

Chapter 3 Symbolic Aggregate Approximation Distance Measure **40**
3.1 Introduction 40
3.2 Literature Review 41
3.2.1 Symbolic Aggregate Approximation 41
3.2.2 Related Work 44
3.3 Optimization Problem: First SAX Model 48
3.3.1 Initial Solution 56
3.3.2 Initial Solution Feasibility Condition 58
3.3.3 Non-Feasible Initial Solution 59
3.3.4 Initial Solution Optimality Condition 59
3.3.5 Non-Optimal Initial Solution 62

3.4 Optimization Problem: SAX with Trend Model . . . 63
3.5 Numerical Experiments . . . 67
3.6 Conclusion . . . 71

Chapter 4 Anomaly Detection . . . 72
4.1 Introduction . . . 72
4.2 Literature review . . . 73
4.3 Methodology . . . 74
4.3.1 Local descriptor functions . . . 75
4.3.2 Clustering . . . 76
4.3.3 Discrete-time Markov chain . . . 78
4.3.4 Testing phase . . . 80
4.4 Empirical evaluation . . . 80
4.5 Conclusion . . . 85

Chapter 5 New Features with Shape Dynamic Time Warping . . . 86
5.1 Introduction . . . 86
5.2 Literature Review . . . 90
5.3 Methodology . . . 95
5.4 Numerical Experiments . . . 99
5.4.1 ShapeDTW vs. DTW . . . 99
5.4.2 ShapeDTW comparisons . . . 101
5.5 Conclusion . . . 101

Chapter 6 Conclusions and Future Work . . . 103

References . . . 106

LIST OF TABLES

Table 1.1 Data points of two series, signal 1 and signal 2. Two signals are plotted in figure 1.3. NaN is a missing value. 6
Table 1.2 Breakpoints of standard Normal distribution having equiprobable regions based on the alphabet size, α [34]. 12
Table 1.3 Equiprobable intervals for Normal distribution with associated letters, average of data points at each bin and assigned letters are shown. 12

Table 2.1 This table illustrates the SAX dimension-reduction with numerosity reduction. The column t refers to the starting point of the subsequence with length 70. Words are represented by sequence of numbers and each number refers to the ordinal positions of letters in the alphabet set. For example, the number 12544 and the word *ABEDD* are equal. 30
Table 2.2 Results of the motif tree classifier error rate vs. 1-NN ED and 1-NN DTW classifiers error rate. The parameters for the motif search is presented. α and w are the size of the alphabet set and the word length, respectively. In each dataset, the best performance is highlighted in bold and an improvement over 10% is shown by †. 36

Table 3.1 Breakpoints of the standard Normal distribution that has equiprobable regions based on the alphabet size, α are shown [34]. 43
Table 3.2 The statistics used for *ESAX*, *SAX with trend* and *SAX with standard deviation* representations are reported. PAA, Min, Max, SD and TD stand for piecewise aggregate approximation, minimum, maximum, standard deviation and trend distance. 49
Table 3.3 *SAX with trend* and *SAX with standard deviation* representations for two time series in figure 3.5 are shown. 50
Table 3.4 *ESAX* representation for sequences in figure 3.5 are shown. 50
Table 3.5 The summary of datasets used for numerical experiments is given. 68
Table 3.6 The summary of 1-NN classifier error rate using different methods is presented. For each dataset, the method that has the smallest error rate is highlighted in bold. ED and CR stand for Euclidean distance and compression rate, respectively. 69

Table 4.1 The symbols used in the Markov chain modeling is defined. 78

Table 5.1 1-NN classification error rates in shapeDTW technique is reported as $best[Q_1, Q_3]$, where *best* is the minimum error rate and Q_1 and Q_3 are the error rates first and third quartiles, respectively. Symbols † and ‡ indicate significant difference in median values of classifier errors when compared with DTW and shapeDTW using *raw-subsequence* descriptor function, respectively. 100

LIST OF FIGURES

Figure 1.1 Red, blue and green colored subsequences are three instances of a motif. The length of each subsequence is 70. 3
Figure 1.2 Three instances of the motif in figure 1.1 are normalized and plotted on top of each other. 4
Figure 1.3 Lengths of signal 1 and signal 2 are 21 and 19, so DTW is used to compute their distance. 6
Figure 1.4 The path starts at the point $(1, 1)$ and finishes at $(21, 19)$. The path is continuous and corresponds the optimal alignment. 7
Figure 1.5 The Euclidean distance or DTW between these two vectors is meaningless. The suitable alternative is the frequency-domain analysis. 9
Figure 1.6 A time series with length 150 and its PAA representation with length 6 are plotted here. The value of each bin is corresponding to 25 data points of the original time series. 10
Figure 1.7 SAX representation of the time series in figure 1.6 is the word *BBADEE*. 13

Figure 2.1 This figure shows the two closest matches of the red shapelet in a time series. The time series is the 5^{th} instance of Bird/Chicken dataset obtained from UCR Archive [10]. 20
Figure 2.2 A class motif of length 75 with 4 instances from *Beetle/Fly* dataset of UCR Archive [10] are shown. 24
Figure 2.3 Normalized form of instances in figure 2.2 are plotted here. 24
Figure 2.4 Plot (a): A time series with length 360 is given. The goal is to discover motif instances with a base length of 70. Plot (b): Two instances of the motif $\langle 12354, 12454\rangle$ are found. Their lengths are 72 (left) and 74 (right). Plot (c): Two instances of the motif are expanded to $\langle 12354, 12454, 12453\rangle$. Their lengths are 75. 29
Figure 2.5 The optimal motif instances that are found in class 1 of Beetle/Fly dataset. The instances represent the motif $\langle EECBBAB, EECBBBB\rangle$. 37
Figure 2.6 The optimal motif instances that are found in class 2 of Shapelet-Sim dataset. The instances represent the motif $\langle ABDFHF, ABDGHF\rangle$. 38

Figure 3.1 A time series with length 150 and its PAA representation with the length of 6 are plotted here. The value of each bin is corresponding to 25 consecutive data points of the original time series. 42
Figure 3.2 SAX representation of the time series in figure 3.1 is the word *BBADEE*. 44
Figure 3.3 SAX representation cannot distinguish between the four segments as they all have the same average. The average is 0.7. 46

Figure 3.4 The average of the first bin is not representative of the behavior of the sequence. ESAX records the minimum and maximum of each segment in addition to the average. 47
Figure 3.5 Four breakpoints are $-0.84, -0.25, 0.25$ and 0.25. The original SAX representation for sequence 1 and 2 are $CBCBDD$ and $BADDDD$. 49
Figure 3.6 Examples of three different sequences with the same SAX representation $BDEBC$ are provided. 51
Figure 3.7 Examples of three different sequences with the same SAX representation $BEBEB$ are provided. 51
Figure 3.8 In both plots, the red dotted line is the average. In the left plot, all data points are equal to their average. In the right plot, the distance between the first point and the last point from the average are computed to be -1.2 and -0.6. 66
Figure 3.9 The summary of 1-NN classifier error rate using different methods is presented. 70

Figure 4.1 The anomaly detection procedure is performed in two phases. . . 74
Figure 4.2 300 two-dimensional data points are grouped into three clusters using $k - means$ technique. 77
Figure 4.3 In this figure, a time series of length 100 and 4 cluster centroids associated with subsequences of length 20 are plotted. 77
Figure 4.4 The first plot shows the path Log-likelihood and the second one shows anomaly detected areas using *normalized subsequence* descriptor function. In this example, the window length, number of clusters and score threshold are set to 30, 7 and -3.5360. 82
Figure 4.5 The first plot shows the path Log-likelihood and the second one shows anomaly detected areas using *normalized subsequence* descriptor function. In this example, the window length, number of clusters and score threshold are set to 30, 6 and -3.1298. 83
Figure 4.6 The first plot shows the path Log-likelihood and the second one shows anomaly detected areas using *local derivatives* descriptor function. In this example, the window length, number of clusters and score threshold are set to 45, 8 and -4.2328. 84
Figure 4.7 This figure shows that if the criteria for detecting anomalies was only the distance of subsequences from the centroid of their cluster, then many false positives were produced. 85

Figure 5.1 Plot (a): Two example time series taken from the Beef Dataset in UCR archive. Plot (b): It is the scaled time warp plot which shows *singularity* in several locations. Plot (c): This plot shows the basic Euclidean distance path alignment, where $w_i = (i, i)$ for $i = 1, ..., m$. 89

Figure 5.2 Plot (a): Two subsequences of larger time series. Plot (b): Shapes of the sequences are similar to each other; however, DDTW justifies the variability in Y-axis towards time warping that produces several singularities. 91
Figure 5.3 Examples of (a) Sakoe-Chiba and (b) Itakura parallelogram constraints for warping path. The path cannot contain any white squares. 92
Figure 5.4 Plot (a): The scaled time warp plot based on DDTW. Plot (b): The scaled time warp plot based on Sakoe-Chiba constrained DTW. The alignment created by DDTW is more intuitive compared to that of Sakoe-Chiba constrained DTW. 94
Figure 5.5 The original sequence and the neighborhood w has the length of 6 and 5, respectively. The embedded sequence, 6 subsequences and their associated normalized form are plotted. 97
Figure 5.6 Error rates of 1-NN classifier of shapeDTW using *normalized-subsequence* (in blue) and *local derivatives* (in brown) as well as DTW error rates (in black dash-dot line) on 24 datasets. 102

Chapter 1

Introduction

1.1 Background

It has been more than a decade that people are surrounded by technologies involving large amounts of data [52]. For example, wearable devices are predicted to sell over 126 million units by 2019 [43], mainly due to the growth of the Internet of Things (IoT) and the benefits one can get from the useful and continuous data provided by these devices regarding their health status and activities [8, 21]. In response to the growth of these technologies, this dissertation addresses the new analyses for problems that are in form of a time series. Specifically, the goal of this research is to present the applications of dimension-reduction techniques and *motifs* (similar subsequences that repeat in a time series) in real-world problems and to focus on two data mining tasks: *time series classification* and *anomaly detection*. The *time series classification* problem is studied from three perspectives: (a) by taking advantage of motifs and the data in a lower-dimensional space form, which also led to introducing a new motif discovery method, (b) by studying a dimension-reduction technique called symbolic aggregate approximation and the asso-

ciated similarity measure, and (c) by employing two shape descriptor functions in shape dynamic time warping. The *anomaly detection* problem is studied by utilizing Markov chain modeling.

The following examples show what type of information can be extracted from data. **Example 1:** Fitbit Alta, an activity tracker and a fitness wristband, can count the steps and detect sports activities, such as running on a treadmill or using an elliptical [1]. Detecting activities by a smartwatch is a *classification* problem. In most cases, the information from different sensors are indexed in time order, can be transferred to smartphones via Bluetooth, and can be used for a deeper analysis. These devices are capable of producing a massive amount of data every second and the advanced statistical tools for analyzing them are required now more than ever. **Example 2:** Sensors collect an enormous amount of data in real time in many domains such as telecommunications, network traffic, power usage, financial monitoring, valve pressure monitoring, electrocardiogram (ECG) monitoring, etc. [13, 52]. Quite a few algorithms have been designed to detect anomalous/unusual behavior in the collected data. Some of them are online-based methods that provide analysis on *streaming data* while the sensor is continuously collecting data [4, 13], and some others are offline that works with stored data (on disk) [52]. Once anomalous behaviors detected, domain experts can identify the event(s) associated with them as a post-processing action. The reason for such unusual behaviors varies based on the source of the input data. For instance, some might represent a machine breakdown in a factory, some might represent a sale on Black Friday, and some might indicate a security attack to the system [3].

The rest of this chapter is organized as follows. First, the time series motif and similarity measures that are used in the analysis are introduced. Then, there is a discussion about dimension-reduction techniques that are used in the big-data. The chapter is fin-

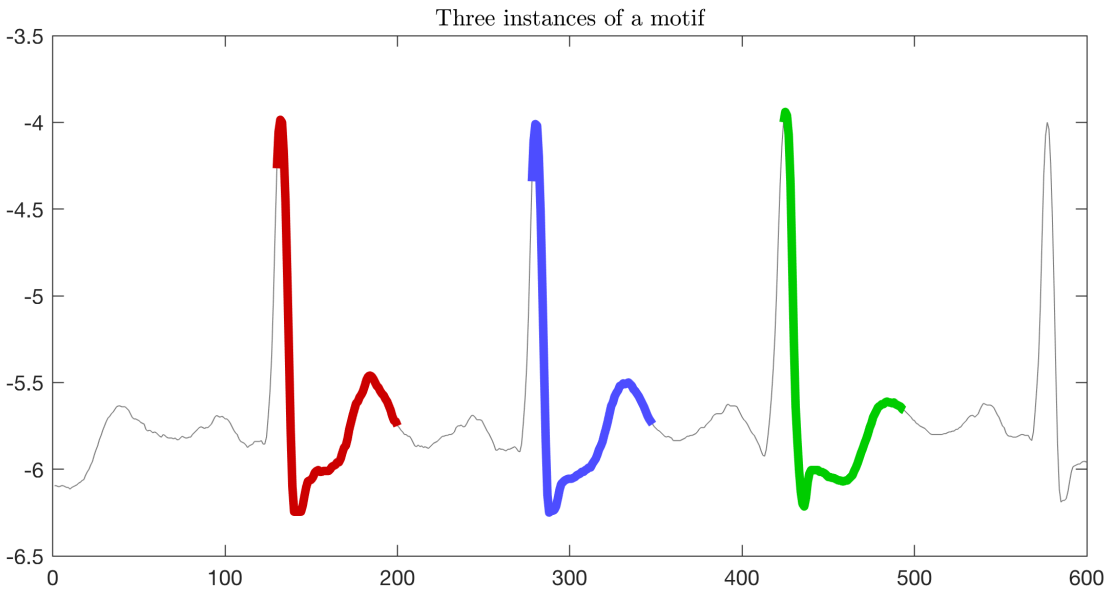

Figure 1.1: Red, blue and green colored subsequences are three instances of a motif. The length of each subsequence is 70.

ished by providing some applications of motifs.

1.2 Time Series Motif

A time series motif is a set of repeating subsequences of a time series that are similar to each other in shape. The two terms *time series motif* and *motif* have the same meaning and can be used interchangeably. Each member of the motif set is an instance of that motif. In figure 1.1, three instances of a motif are shown. The term motif originally referred to a sequence of amino-acids in a DNA sequence, but it is now used in different areas such as genetics, medicine, mathematics, music, etc. [50]. In every area of application, motifs carry useful information that can be explained by experts in that domain. However, experts are not always available, meaning that motifs are generally used in

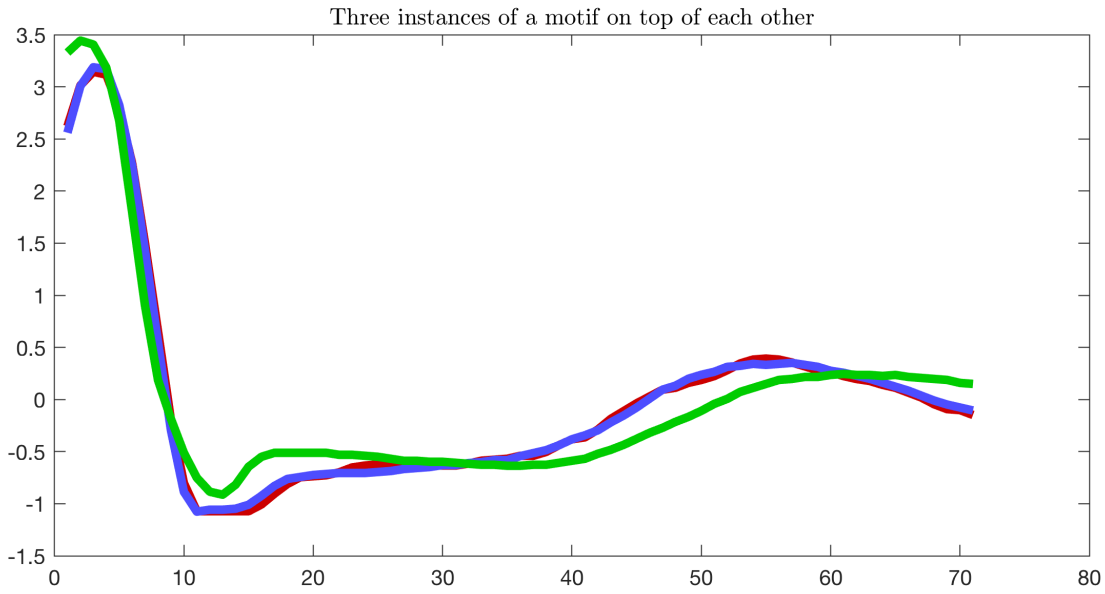

Figure 1.2: Three instances of the motif in figure 1.1 are normalized and plotted on top of each other.

unsupervised data exploration and analytics [38]. Examples of such tasks include *time series classification*, *clustering* and *anomaly detection*.

A motif is basically a pattern. Before analyzing and using motifs, mathematical tools are needed to quantify them, which depend on the domain they are presented in. Dimension-reduction algorithms will be discussed later, but as a simple example, if motifs are presented in their original domain, then a logical way to show them is by their normalized form and their similarity can then be measured by Euclidean distance. In order to depict how similar the instances of the motif in figure 1.1 are, their normalized form are plotted in figure 1.2.

In what follows, similarity measures that are often used in the literature when working with motifs are provided.

1.3 Similarity Measures

In this section, several similarity (distance) functions from literature that are used throughout this research are introduced. They are Euclidean distance, dynamic time warping and Hamming distance. It is discussed why each one of them is used.

1.3.1 Euclidean Distance

Euclidean distance is a distance measure that takes two vectors of the same size as their input. Formally, the Euclidean distance between two vectors $X = (x_1, x_2, ..., x_n)$ and $Y = (y_1, y_2, ..., y_n)$ is $ED(X, Y)$ and computed as:

$$ED(X, Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{1.1}$$

1.3.2 Dynamic Time Warping

Dynamic time warping (DTW) is a distance measure that quantifies the similarity of two time series. It does not necessarily need the lengths of two vectors be the same. As an example, plot (a) of figure 1.3 and table 1.1 show two signals that do not have the same length. Specifically, given two time-ordered sequences $X = x_1, ..., x_m$ and $Y = y_1, ..., y_n$, DTW computes the minimum warping cost function:

$$Z = \sum_{k=1}^{K} dist(w_k) \tag{1.2}$$

where w_k is the k^{th} element of the warping path that represents the k^{th} alignment (x_i, y_j). DTW has three criteria for creating the path $W = w_1, ..., w_K$, they are:

1. $w_1 = (1, 1)$; the boundary condition for the first element of the warping path.

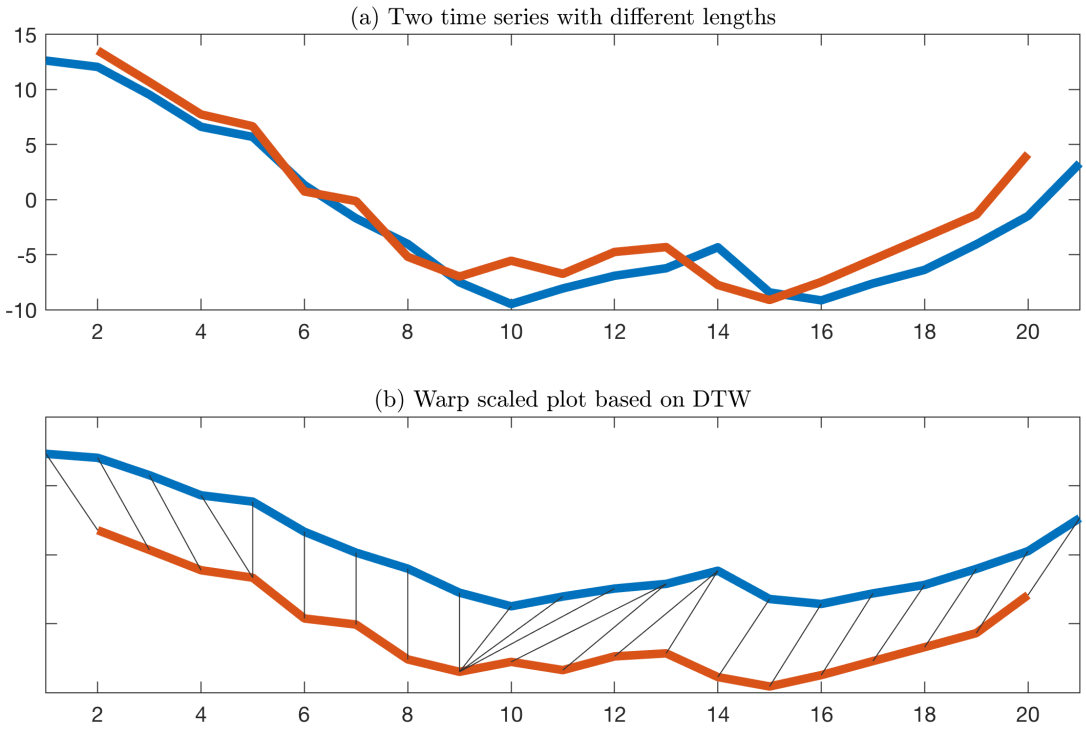

Figure 1.3: Lengths of signal 1 and signal 2 are 21 and 19, so DTW is used to compute their distance.

Table 1.1: Data points of two series, signal 1 and signal 2. Two signals are plotted in figure 1.3. NaN is a missing value.

time	signal 1	signal 2	time	signal 1	signal 2
1	12.63	NaN	12	−6.92	−4.76
2	12.05	13.55	13	−6.24	−4.31
3	9.53	10.68	14	−4.33	−7.76
4	6.61	7.74	15	−8.43	−9.12
5	5.69	6.66	16	−9.15	−7.47
6	1.32	0.74	17	−7.62	−5.44
7	−1.71	−0.14	18	−6.38	−3.42
8	−4.03	−5.2	19	−4.06	−1.38
9	−7.52	−6.99	20	−1.49	4.14
10	−9.49	−5.55	21	3.29	NaN
11	−8.07	−6.75			

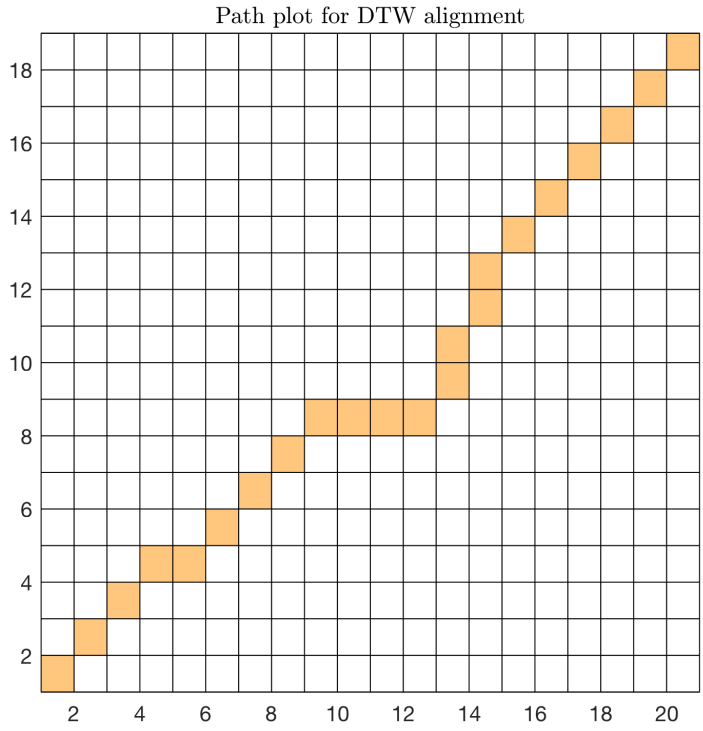

Figure 1.4: The path starts at the point $(1, 1)$ and finishes at $(21, 19)$. The path is continuous and corresponds the optimal alignment.

2. If $w_k = (i, j)$, then w_{k+1} can take either $(i, j+1)$, $(i+1, j)$ or $(i+1, j+1)$, when i and j are in the appropriate range. This condition ensures that the path is continuous and a monotonically increasing function.

3. The last element of the warping path is (m, n).

The squared or absolute deviation between elements of the two series are popular functions to use for the $dist(\cdot, \cdot)$ function. Plot (b) of figure 1.3 and figure 1.4 show other representation of the optimal path for two series with squared deviation as $dist$ function. DTW is more computationally intensive compared to Euclidean distance. However, in many cases, it results in a better representation of a distance between two vectors when there exists a little shifting or when their lengths are not equal. From an application per-

spective, the superiority of *time series classification* with DTW over the Euclidean distance in combination with the one-nearest neighbor method has been empirically shown in several studies [25, 53]. DTW is not a metric distance as it does not satisfy the triangle inequality. That is, if A, B, C are three vectors, then there is no relation between two quantities $DTW(A, B) + DTW(A, C)$ and $DTW(B, C)$.

Chapter 5 provides more details on DTW variants and its application in *time series classification*.

1.3.3 Hamming Distance

The Hamming distance between two strings is the number of elements at which the corresponding values are different. The size of two strings must be equal. For example, the Hamming distance between two strings $AABCBB$ and $EABCBD$ is 2, as their first and last elements are different.

This section is finished by pointing out a remark. Different similarity measures should be used based on the context of the problem, size of the problem, and how one wants to interpret the resulting distance. Some studies showed that for large classification problems, the error rate of Euclidean distance converges to DTW, meaning that Euclidean distance is more efficient due its computation complexity [53, 47]. With the same logic, for small size classification problems, DTW is preferred. There are some problems that neither DTW nor Euclidean distance is suitable. For example, consider two vectors of figure 1.5 (the example is obtained from [54]). Both vectors have the same length of 1000. However, one has 14 cycles of a sine wave and the other has 9. Since DTW and Euclidean distance measures connect every point of one vector to the other vector, it does not make sense to use them in this example. Two other alternatives are *uniform scaling* [54] and

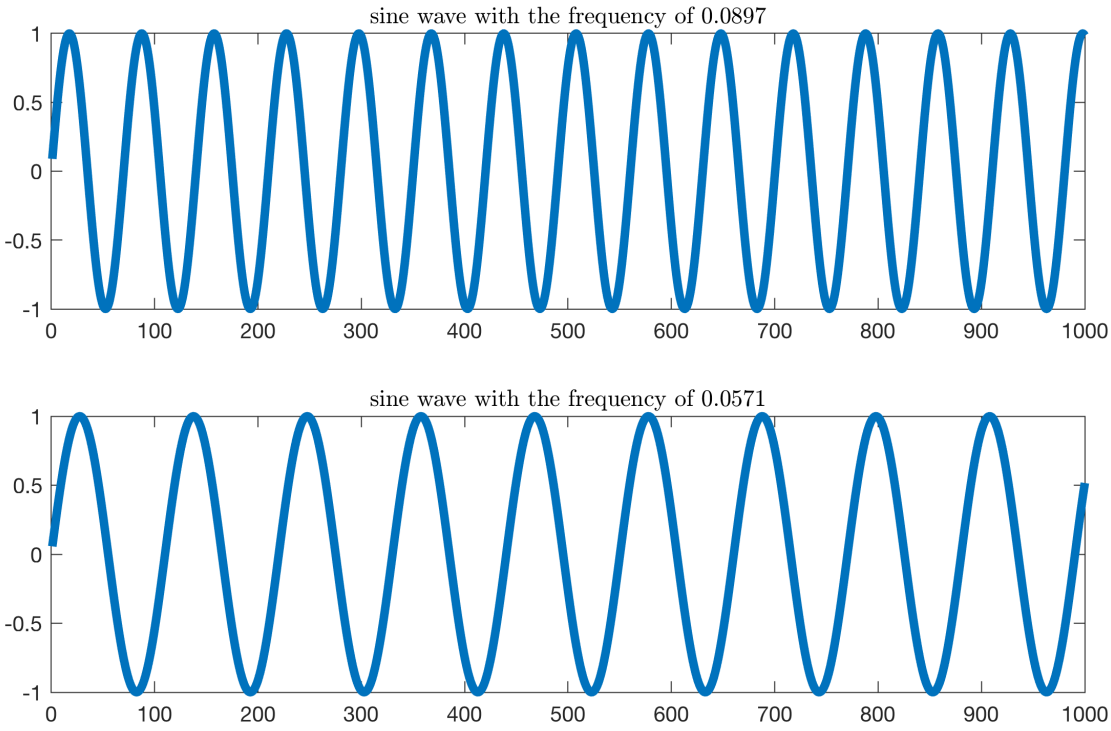

Figure 1.5: The Euclidean distance or DTW between these two vectors is meaningless. The suitable alternative is the frequency-domain analysis.

frequency-domain analysis, whose discussion is beyond the scope of this project.

1.4 Dimension Reduction

Many studies analyze motifs in different spaces. The reasons are discussed later, but for now, it suffices to say that the computational complexity of finding different types of motifs makes the hope of working directly with time series in the time-domain unreasonable. In what follows, two popular time series dimension-reductions are introduced. Piecewise Aggregate Approximation (PAA) and Symbolic Aggregate Approximation (SAX) have been widely used to represent the original time series in simpler ways requiring smaller

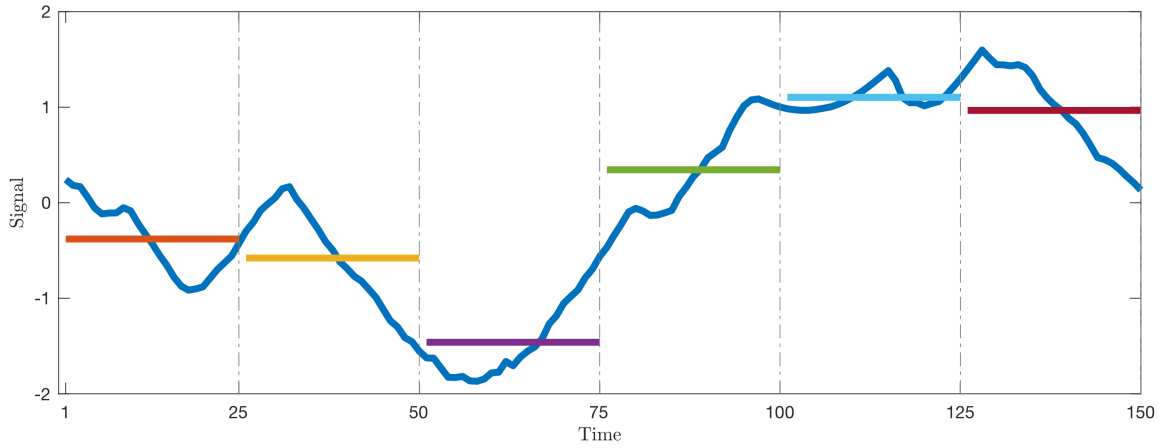

Figure 1.6: A time series with length 150 and its PAA representation with length 6 are plotted here. The value of each bin is corresponding to 25 data points of the original time series.

system space [5, 36, 34, 12].

1.4.1 Piecewise Aggregate Approximation

A detailed review of PAA is provided in [34]. Suppose $X = x_1, ..., x_n$ is a time series. One can aggregate each $\frac{n}{w}$ points together (suppose n is divisible by w) and transform X to $\bar{X} = \bar{x}_1, ..., \bar{x}_w$ as follows:

$$\bar{x}_i = \frac{\sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} x_j}{n/w} \quad \text{for } i = 1, ..., w. \tag{1.3}$$

By doing this, PAA tries to reduce the size of the time series from n to w. Figure 1.6 shows a time series and its PAA representation. In this example, $n = 150$ and $w = 6$.

1.4.2 Symbolic Aggregate Approximation

SAX is another dimension reduction algorithm similar to PAA that tries to reduce the dimensions of a time series. However, SAX takes more steps to represent the original sequence with some words. A detailed review of SAX is also provided in [34]. Following definitions help us build the structure of the SAX transformation:

Definition *(Word)* A word with size w is a sequence of letters represented by $\hat{X} = \hat{x}_1, ..., \hat{x}_w$, where each element $\hat{x}_i$ can take a symbol from the set *Alphabet*.

Definition *(Breakpoints)* Based on the size of alphabet set, α, breakpoints are a sequence of $\alpha - 1$ points obtained from the standard Normal distribution along with two points $-\infty$ and ∞ such that the area under the Normal curve between every two consecutive points is $1/\alpha$.

Table 1.2 from [34] introduces the breakpoints based on the alphabet size ranging from 3 to 10 elements. For example, if the alphabet size is 3, then the breakpoints are -0.43 and 0.43 and if it is 4, then the breakpoints are -0.67, 0 and 0.67. To obtain a SAX representation of a given time series X, one should take the following steps:

1. Determine the word size, w, and alphabet size, α
2. Normalize the sequence (subtract the mean and divide by the standard deviation)
3. Apply the PAA method, based on the word size, to obtain $\bar{X} = \bar{x}_1, ..., \bar{x}_w$
4. For each element $\bar{x}_i$, assign a letter from the alphabet set based on the intervals created by consecutive breakpoints to obtain $\hat{x}_i$

Figure 1.7 shows the time series in figure 1.6 and its SAX representation. Word size and alphabet size are set to 6 and 5, respectively. Note that each interval $(-\infty, -0.84)$,

Table 1.2: Breakpoints of standard Normal distribution having equiprobable regions based on the alphabet size, α [34].

α	3	4	5	6	7	8	9	10
β_0	-0.43	-0.67	-0.84	-0.97	-1.07	-1.15	-1.22	-1.28
β_1	0.43	0	-0.25	-0.43	-0.57	-0.67	-0.76	-0.84
β_2		0.67	0.25	0	-0.18	-0.32	-0.43	-0.52
β_3			0.84	0.43	0.18	0	-0.14	-0.25
β_4				0.97	0.57	0.32	0.14	0
β_5					1.07	0.67	0.43	0.25
β_6						1.15	0.76	0.52
β_7							1.22	0.84
β_8								1.28

..., $(0.84, \infty)$ has the equal-area 1/5 under the Normal distribution curve and because of that, every letter in the alphabet set $\{A, B, C, D, E\}$ has the same probability to appear at each position of the word. The output of the algorithm is the word *BBADEE*. Table 1.3 shows the letter assignments for this example. This method converts the whole time series into one SAX word. In other words, it only shows the global structure of the time series. In order to capture local structures as well, which are lost in converting the whole

Table 1.3: Equiprobable intervals for Normal distribution with associated letters, average of data points at each bin and assigned letters are shown.

Interval	Letter	Bin 1	Bin 2	Bin 3	Bin 4	Bin 5	Bin 6
$(-\infty, -0.84)$	A			-1.459			
$(-0.84, -0.25)$	B	-0.379	-0.578				
$(-0.25, +0.25)$	C						
$(+0.25, +0.84)$	D				0.346		
$(+0.84, +\infty)$	E					1.103	0.966
Assigned Letter		*B*	*B*	*A*	*D*	*E*	*E*

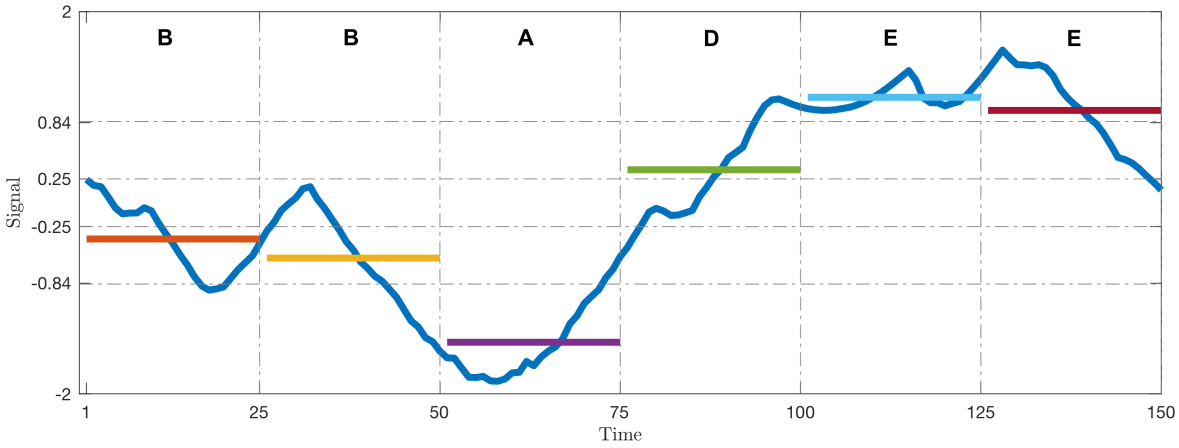

Figure 1.7: SAX representation of the time series in figure 1.6 is the word *BBADEE*.

time series into one word, one can use a sliding window of size q, generate $n - q + 1$ subsequences out of X by sliding the window over the time series and then apply SAX transformation to each subsequence. Consecutive subsequences might map to the same SAX word, so a technique called *numerosity reduction* is often used, which is to keep only the first occurrence of repeated words [34]. The SAX method with a sliding window has been used in some motif discovery algorithms [12, 31, 30].

This section is finished by pointing out some benefits of the dimension-reduction process. They are:

1. *Similarity computation speed up*

   The most important benefit is speeding up similarity computation between series. Either PAA or SAX, reduces the dimension of series from n to w.

2. *Noise reduction*

3. *Outlier effect reduction*

   The effect of outliers in similarity computation is reduced, though not completely

removed.

Chapter 3 provides more details on variants of SAX representation of time series and their associated similarity measures.

1.5 Motif Applications

Earlier in this chapter, the importance of studying motifs was pointed out: it has many applications in genetics, medicine, mathematics, music, etc. [50]. This section attempts to be more specific; thus, a brief review of data mining tasks associated with motifs is provided.

1.5.1 Discovering Motifs

This topic addresses the problem of discovering similar subsequences that repeat several times in a time series. The earliest algorithm attempting to discover motifs is called EMMA (Enumeration of Motifs through Matrix Approximation) introduced in 2002 [33]. This attracted many researchers. A good reference for understanding the progress of this topic is the survey by Torkamani and Lohweg [50].

Discovering motifs is an unsupervised process. The general challenge of the motif discovery task is that one does not know what the baseline subsequence to search for. An important factor is the size of a motif. Two cases are considered:

1. *Length of the motif is known*

   By knowing the length of a motif, the search space is limited to subsequences of that length. An exact brute force search and quite a few approximation algorithms are provided for this task. If the length of the time series is n and the length of the

motif is m, brute force search makes $(n - m + 1)^2$ comparisons. The comparison function can be either Euclidean distance or DTW. Based on a threshold ϵ provided by the user, similar subsequences are put in different clusters. Triangular inequality and early abandoning are often used to improve the search speed [26, 33]. EMMA [33], random projection (based on SAX transformation and hashing tables) [12] and probabilistic model [51] are famous approximation methods. VizTree, a motif discovery software that uses SAX dimension reduction, belongs to this category when no numerosity reduction option is used [31].

2. *Length of the motif is not known*

   This problem is harder than discovering the motifs when the length of a motif is fixed. A trivial solution for this problem is to take a motif finding algorithm and try it with all ranges of feasible lengths. The problem can get even harder if instances of a motif can have different lengths. Not many studies belong to this category. However, a visualization tool has been developed for this type of problem using a grammar-based compression algorithm [30]. VizTree software with numerosity reduction finds variable length motifs, whose lengths are larger than the predefined window size [31].

1.5.2 Anomaly Detection

Pattern-wise outlier detection for temporal data is one of the popular topics in data mining. To the best of our knowledge, detection of the subsequences in a time series having different behaviors from the rest of the time series was introduced in 2005 [26]. Applications of such a topic include healthcare, fraud detection, telecommunications, network traffic, power usage, financial monitoring, valve pressure monitoring, ECG monitoring,

etc. [13, 52]. *Anomaly detection* problem is discussed in chapter 4.

1.5.3 Temporal Data Feature Generation

Features generated by aforementioned dimension reductions can be used in *classification*, *clustering*, and *anomaly detection*. To the best of our knowledge, *bag of patterns* is the only motif-related tool that has been used in these tasks [36, 35]. The definition of the bag of patterns is as follows:

Definition *(Bag of Patterns)* It is a matrix, M, used to summarize information from multiple time series in terms of their SAX words (patterns) when the concept of sliding window is applied. The matrix M is formed as follows: each row i represents a SAX word and each column j represents a time series. Element $M_{i,j}$ denotes the frequency of occurrence of SAX word i in time series j.

The idea of using the bag of patterns comes from *bag of words* representation in text mining [36]. Bag of patterns and its histogram matrix representation are the features to use for classification. It has been empirically shown that all time series belonging to the same class share similar behavior in their patterns histogram [36]. The superiority of this feature definition along with one-nearest neighbor classifier is reported in [35, 36].

The organization of this dissertation is as follows. In chapter 2, *time series classification* problem is studied by introducing a new motif discovery method called *deep search*. Chapter 3 discusses the variants of SAX dimension reduction and their associated similarity measures. In that chapter, a new form of a similarity measure and its application in *time series classification* problem are presented. Chapter 4 addresses the use of Markov chains in the *anomaly detection* problem. Finally, chapter 5 presents new analysis on shape dynamic time warping with the application in *time series classification*. Chapter

6 is the conclusion of this dissertation.

Chapter 2

Time Series Classification by Motifs

2.1 Introduction

Techniques that are often used for classifying time series are either distance based or feature based [25]. In distance-based methods, all distances of a given time series from time series of the training set are computed and its label is predicted by the vector of computed distances using machine learning methods such as one-nearest neighbor (1-NN) or decision trees. The distance function can be any form of similarity/dissimilarity measures. Clearly, if the lengths of two time series are not equal, the Euclidean distance cannot be used and Dynamic Time Warping (DTW) is the other option.

The time series shapelet concept is introduced in [55]. The assumption is that there are specific subsequences (or just one subsequence) that are very likely to exist in a time series if the time series belongs to a certain class. These subsequences are called *shapelets*. It is extremely helpful to have prior information about shapelets. In case that such information is available, a time series can be labeled simply by seeing if it has similar subsequences to the shapelets. This approach, subsequence matching, is potentially more

accurate (in terms of classification accuracy) in some applications and gives more insight why a certain label is assigned. However, prior information is not always available. One solution to overcome this issue is to use brute force search on all time series in the training dataset and generate shapelet candidates of all lengths. Let S be the number of all shapelet candidates. In that setting, one can represent a given time series by an S-dimensional vector, the i^{th} element of which represents the minimum distance of the i^{th} shapelet from that time series that is found by sliding the shapelet over the time series. Figure 2.1 shows an example of this process for the red shapelet. The two most similar subsequences are illustrated in the original time series. The next step would be using a classifier to select the best shapelet or set of shapelets in order to differentiate between classes. This approach has been used in a few studies [18, 55, 20]. Note that using the above-mentioned approach by itself is very time-consuming. Although entropy pruning and early abandoning rules help to speed up the computation process (99.9% of candidates are pruned off), this approach might take several days to complete in some problems [20].

In feature-based methods, structural and statistical features are used to represent a time series. As a good example close to this study, DTW distance was used as a feature for a time series [25]. The approach is briefly explained here, as a similar method is used in this study later on. Let Q be a time series and the training set be $\mathcal{D} = \{T_1, ..., T_n\}$, having n time series with potentially different lengths. The question of whether Q belongs to the set $\mathcal{D}$ is not a concern here. The feature-based representation of Q, based on DTW is the vector of its DTW distance from each training time series in $\mathcal{D}$. That is, Q is represented by $\mathcal{F}(Q) = (DTW(Q, T_1), ..., DTW(Q, T_n))$. This feature definition led to the classification accuracy improvement in 31 out of 47 UCR time series benchmark [10]. However, note that DTW distance is most suitable to compare two time series of similar

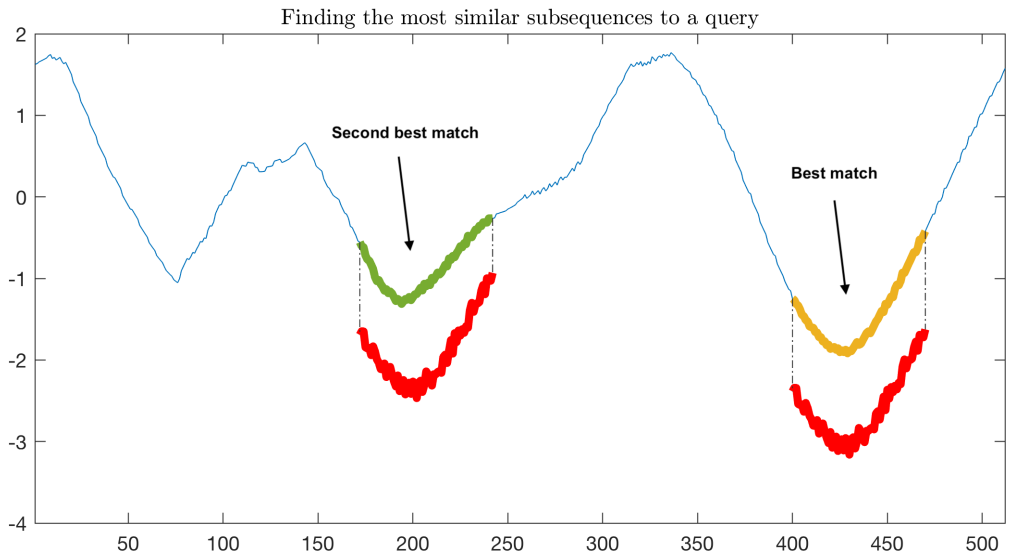

Figure 2.1: This figure shows the two closest matches of the red shapelet in a time series. The time series is the 5^{th} instance of Bird/Chicken dataset obtained from UCR Archive [10].

lengths. That is, one drawback of this feature definition is that if the length of Q and a time series T_i are significantly different, then $DTW(Q, T_i)$ does not really make sense.

The studies introduced herein have one characteristic in common: they attempt to measure similarities between different time series and use that information to build appropriate classifiers. The first two [55, 20] capture similarities between subsequences (which are also called candidates) as a discriminatory feature, while the third one [25] used whole series similarities. This study attempts to relate these two approaches. "Class Motifs" are used as the candidates. Class motifs, informally, are similar subsequences/shapes that repeat in time series of a certain class.

2.2 Background

In this section, the mathematical tools that are used throughout this section are provided.

Definition *(Time series)* A time series T of length n is an ordered sequence of n-real valued numbers $\langle t_1, t_2, ..., t_n \rangle$.

Time series applications cover temporal measurements in many domains such as medicine (electrocardiogram and electroencephalogram recordings), economics (stock price and inventory level) and environmental variables (temperature) among others.

Definition *(Time series motif)* A time series motif, or motif, is a set of subsequences, which are similar in shape and repeat in a time series.

Due to the vast applications of motifs in genetics, medicine, mathematics, music, etc. [50], the task of discovering motifs has attracted many researchers in the area of computer science in the past decade [50, 33, 26, 12, 51, 31, 5, 38]. Piecewise Aggregate Approximation (PAA) and Symbolic Aggregate Approximation (SAX) are two dimension-reduction techniques that are often used to find motifs of certain lengths [34]. SAX extends PAA and is a form of representing a time series with a word (or a set of words depending on the settings of the algorithm). To better understand how SAX and PAA work, the pseudocode for the SAX function, $SAXMaker(T, w, \alpha)$, is given here as algorithm 1, where T is a time series, w is the number of bins and α is the size of alphabet set. The function $Map2Str(\cdot, \alpha)$ converts a single number to a letter based on the equiprobable intervals of the Normal distribution that the number falls into. Each interval has $1/\alpha$ of the area under the Normal distribution curve.

Algorithm 1: SAX dimension reduction function: $SAXMaker(T, w, \alpha)$

Require $T \leftarrow$ *a time series,* $w \leftarrow$ *word size and* $\alpha \leftarrow$ *alphabet size*
if $|T| \bmod w \neq 0$ **then**
  abort
else
  $BinSize \leftarrow |T| / w$
end
$T \leftarrow (T - Mean(T)) / Std(T)$
for $i \leftarrow 1$ **to** w **do**
  $T_{PAA}[i] \leftarrow Mean(T[(i-1) * BinSize + 1 : i * BinSize])$
end
for $i \leftarrow 1$ **to** w **do**
  $Word[i] \leftarrow Map2Str(T_{PAA}[i], \alpha)$
end
return $Word$

2.3 Methodology

This section is started by stating the motivation, a definition, and pseudocodes of the proposed algorithm.

Shapelets can provide a good insight about data [55] and, based on that, can be used for time series classification; see [18, 55, 20] for details. The key idea in using shapelets is that there exists a subsequence (or subsequences) in each class that is unique in shape and their existence in a time series can predict its label. Finding the best shapelet, in general, is a difficult task, as one does not know its length and the search space grows with the number of time series at each class [18]. The efficiency of each shapelet is measured by how well it can perform as a differentiating factor. It means that one should first compute the minimum distance between all time series in the dataset and the candidate shapelet and then search for a distance threshold, d, to assign each time series label (i.e. in a two-class dataset, those having values smaller than d are assigned to class 1 and the

rest are assigned to class 2) [18]. However, there are a few drawbacks and limitations to this approach:

1. There is no intuition behind the candidates search process. All possible subsequences are first selected in a brute-force manner, and then they are either pruned off by entropy pruning or early abandoning rules, or they are chosen to be in the list of good candidates [55].

2. One cannot use distance measures other than Euclidean distance. For instance, if DTW is to be used, one should modify the process of finding the closest match of a shapelet in a time series.

3. The lengths of shapelets and the closest matches should be equal.

These limitations are the motivations to use class motifs in this study, instead of shapelets. First, the definition a class motif is needed:

Definition *(Class motif)* A class motif is a time series motif whose instances appear in different time series of the same class.

Note that instances of a class motif need not be presented in all time series of the class. As an example for class motifs, consider the *Beetle/Fly* dataset from UCR Archive [10] that has 20 time series with 2 classes (each class has 10 time series). Only the first 4 time series of class 1, $T_1, ..., T_4$ are used. Each time series has the equal length of 512. In figure 2.2, 4 instances of a class motif of length 75 are presented. Each instance is found in one time series. Figure 2.3 shows the normalized form of 4 instances.

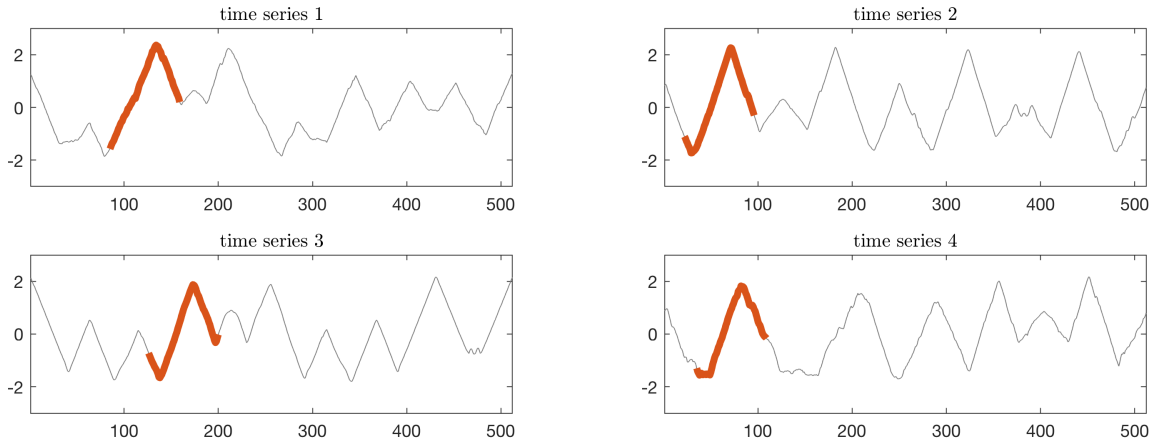

Figure 2.2: A class motif of length 75 with 4 instances from *Beetle/Fly* dataset of UCR Archive [10] are shown.

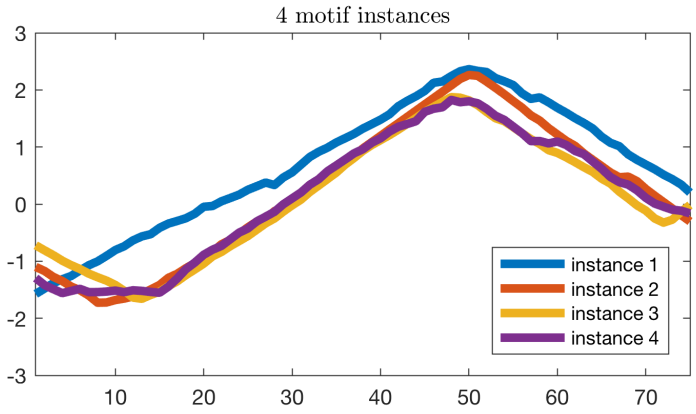

Figure 2.3: Normalized form of instances in figure 2.2 are plotted here.

2.3.1 Motif Discovery Algorithms

In order to discover class motifs, it is needed to start with a motif finding algorithm. In this regard, a few methods were introduced in chapter 1; *Brute-Force* search, *random projection* [12], *EMMA* (Enumeration of Motifs through Matrix Approximation) [33], and *probabilistic model*[51] are well-known algorithms. In this section, two methods will be explained. One is the proposed method, called *deep search*, that allows finding instances with variable length based on SAX dimension-reduction and the other is the well-known *random projection* [12].

Deep Search Method

The motivation to create this method is to find instances of a motif with variable lengths. The *deep search* method relies on SAX dimension-reduction and numerosity reduction technique. The numerosity reduction rule states that if any consecutive subsequences (in the time-domain) map to the same SAX word, then only the first occurrence is kept, yet the overall length of the transformation is increased by the number of repetitions. Algorithm 2 is the pseudocode of the proposed motif discovering process.

The *deep search* method is explained with an example. Plot (a) of figure 2.4 shows the time series T whose motifs with the base length of 70 will be discovered. To do so, $SAXMaker(\cdot, w, \alpha)$ function is run with appropriate parameters w and α over subsequences with length 70 extracted from T. The user can decide the parameters w and α based on previous experience and the accuracy they desire. The parameters are set to be $w = \alpha = 5$. Table 2.1 summarizes the output of several calls to the $SAXMaker(\cdot, w, \alpha)$ function in terms of the SAX words and their respective starting point. The first subsequence of T (starting at $t = 1$) is mapped to the word 12544 (equivalent to $ABEDD$).

Algorithm 2: Finding Motifs with variable length using SAX dimension-reduction

Require $T \leftarrow$ *a time series,* $n \leftarrow$ *window size,* $w \leftarrow$ *word size and* $\alpha \leftarrow$ *alphabet size*
$k \leftarrow 1$
for $i \leftarrow 1$ **to** $|T| - n + 1$ **do**
    $S \leftarrow T[i : i + n - 1]$
    $SAX_S \leftarrow SAXMaker(S, w, \alpha)$
    if $i == 1$ **then**
        $SAX_{List}[k] \leftarrow SAX_S$
        $Index[k] \leftarrow i$
    end
    if $SAX_{List}[k] \neq SAX_S$ **then**
        $k \leftarrow k + 1$
        $SAX_{List}[k] \leftarrow SAX_S$
        $Index[k] \leftarrow i$
    end
end
for $k \leftarrow 1$ **to** $|SAX_{List}|$ **do**
    $Ord_{List}[k] \leftarrow$ Index order of $SAX_{List}[k]$
end
$U \leftarrow Unique\,(Ord_{List})$
$ColM \leftarrow [0]_{|U|*|U|}$
for $i \leftarrow 1$ **to** $|Ord_{List}| - 1$ **do**
    $ColM[Ord_{List}[i], Ord_{List}[i + 1]] \leftarrow ColM[Ord_{List}[i], Ord_{List}[i + 1]] + 1$
end
$m \leftarrow 1$
for $i \leftarrow 1$ **to** $|ColM|$ **do**
    for $j \leftarrow 1$ **to** $|ColM|$ **do**
        if $ColM[i, j] > 1$ **then**
            $Motif[m] \leftarrow \{i, j\}$
            $m \leftarrow m + 1$
        end
    end
end

```
for i ← 1 to m do
    Current ← Motif[i]
    for j ← 1 to |Ord_List| − 1 do
        if Current == {Ord_List[i], Ord_List[i + 1]} then
            Loc[i] ← Append(Loc[i], j)
        end
    end
end
i ← 1
S ← True
while S do
    Current ← Motif[i]
    Expand(Current, Ord_List, Loc, |U| , i)
    i ← i + 1
    if i > |Motif| then
        S ← False
    end
end
return Motif
```

Table 2.1 shows that the starting point of the next SAX word is at $t = 3$. It means that the subsequence starting at $t = 2$ must have also mapped to the word 12544. Due to the numerosity reduction rule, only the first occurrence is kept.

The next step of the *deep search* is to put the SAX words in Ord_{List}. The vector Ord_{List} saves the index order of the words in the previous step. For example, the word 12354 (equivalent to $ABCED$) is the smallest generated SAX word (which appears two times at $t = 137$ and $t = 285$), so its index is 1. Now, the roots of each motif set are ready to be built. This is done by looping over generated words and seeing if any two consecutive words are repeated. For example, the word 12354 is followed by the word 12454 two times in positions $\{137, 138\}$ and $\{285, 286\}$. The combination of two consecutive words $\langle 12354, 12454 \rangle$ is considered to be a motif. The length of the first instance of this motif

Algorithm 3: Expanding a given motif function:
$Expand(Current, Ord_{List}, Loc, |U|, j)$

$MSize \leftarrow |Current| - 1$
$MFreq \leftarrow [0]_{|U|}$
$a \leftarrow Loc[j]$
for $i \leftarrow 1$ **to** $|a|$ **do**
  if $a[i] + MSize \leq |Ord_{List}|$ **then**
    $MFreq[Ord_{List}[a[i] + MSize]] \leftarrow MFreq[Ord_{List}[a[i] + MSize]] + 1$
  end
end
for $i \leftarrow 1$ **to** $|U|$ **do**
  if $MFreq[i] > 1$ **then**
    $TempCurrent \leftarrow Append(Current, i)$
    for $n \leftarrow 1$ **to** $|a|$ **do**
      if $a[n] + MSize \leq |Ord_{List}|$ **then**
        if $Ord_{List}[a[n] : a[n] + MSize] == TempCurrent$ **then**
          $TempLoc \leftarrow Append(TempLoc, a[n])$
        end
      end
    end
    $Loc \leftarrow Append(Loc, TempLoc)$
    $Motif \leftarrow Append(Motif, TempCurrent)$
  end
end
return $Motif, Loc$

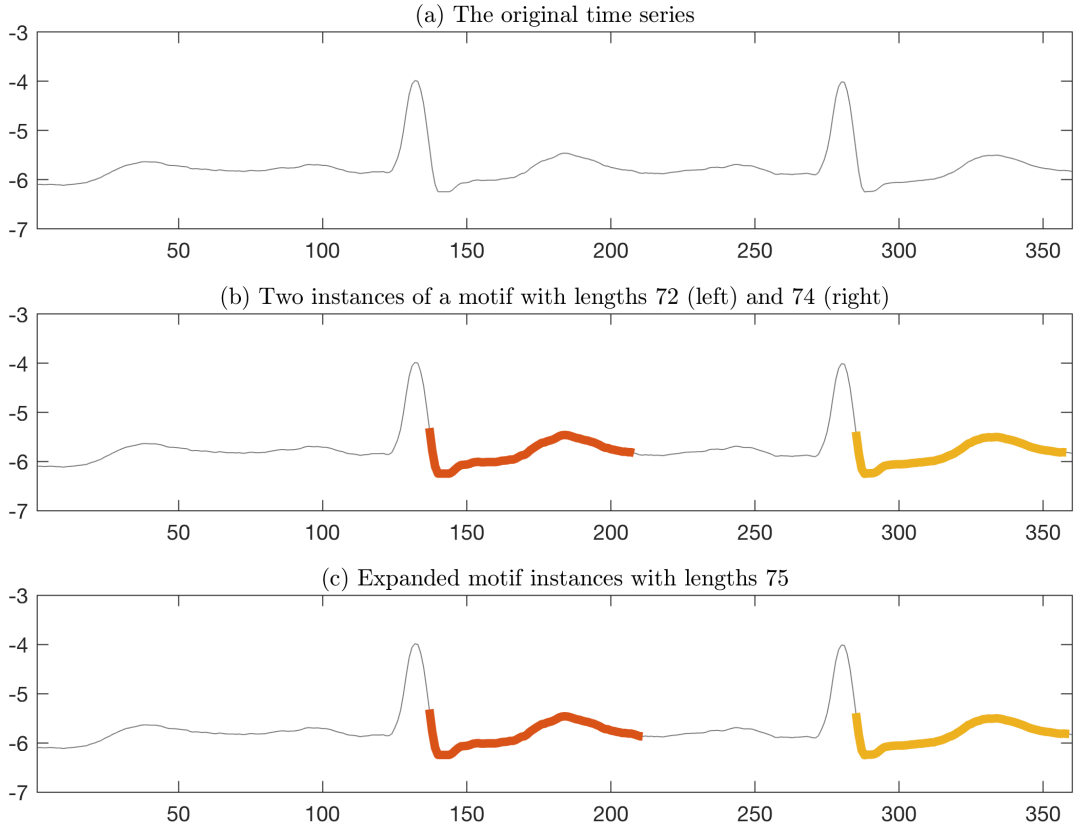

Figure 2.4: Plot (a): A time series with length 360 is given. The goal is to discover motif instances with a base length of 70. Plot (b): Two instances of the motif $\langle 12354, 12454 \rangle$ are found. Their lengths are 72 (left) and 74 (right). Plot (c): Two instances of the motif are expanded to $\langle 12354, 12454, 12453 \rangle$. Their lengths are 75.

Table 2.1: This table illustrates the SAX dimension-reduction with numerosity reduction. The column t refers to the starting point of the subsequence with length 70. Words are represented by sequence of numbers and each number refers to the ordinal positions of letters in the alphabet set. For example, the number 12544 and the word *ABEDD* are equal.

t	Word	t	Word	t	Word	t	Word	t	Word
1	12544	61	23325	131	42233	191	51242	241	32522
3	12543	63	22325	132	42243	193	51252	246	33522
5	13543	67	22225	134	32244	194	41352	251	34422
9	14543	72	23225	135	22354	196	41351	254	35323
11	14533	73	23235	137	12354	197	41451	255	35223
14	15433	75	23234	138	12454	198	32451	256	25223
18	15423	76	23244	140	12453	201	32551	261	35223
20	15323	77	23243	143	12553	202	22541	265	34223
21	25323	78	23253	146	12543	204	23541	266	44223
23	35323	79	33252	147	12542	205	23532	267	43223
24	35213	82	33251	150	13542	206	23533	268	53224
25	45213	86	33352	152	13532	207	23424	269	52224
28	54113	88	32352	153	14532	209	23324	271	52233
29	54114	89	32442	157	15522	210	23325	272	51233
31	53114	91	32432	160	15422	213	22325	276	52233
32	53124	92	32532	162	25422	215	22225	279	42233
35	52124	93	32522	165	25312	218	23225	280	42243
38	52134	98	33522	166	35312	221	23235	282	32244
41	52143	103	34422	168	35212	223	23234	283	22354
43	52243	105	34322	169	45212	224	23244	285	12354
44	52252	106	35322	171	55212	225	33243	286	12454
45	42252	107	35223	173	54212	226	33253	290	12453
47	42351	110	25223	174	54113	227	33252	291	12553
51	32451	112	35223	177	54123	230	33251		
54	32551	117	44223	178	53123	233	33252		
55	32541	119	43223	182	52123	234	33352		
57	22532	120	53223	185	52133	235	32352		
58	22433	121	52223	188	52233	237	32442		
59	23424	124	51233	189	52243	239	32432		
60	23324	128	52233	190	51243	240	32532		

is 72 (starting at $t = 137$ and ending at $t = 140 - 1 + 70 - 1 = 208$) and length of the second instance is 74 (starting at $t = 285$ and ending at $t = 290 - 1 + 70 - 1 = 358$). Plot (b) Figure 2.4 shows these two subsequences.

After this step, a list of motif roots is created. It is possible to expand each motif root by seeing if the next word (if there is any) at each instance is repeated or not. This process is handled by the $Expand\,(\cdot,\cdot,\cdot,\cdot,\cdot)$ function (see algorithm 3). Back to the example, the motif root was $\langle 12354, 12454\rangle$ and the next word at each instance of the motif is 12453. Therefore, the motif $\langle 12354, 12454\rangle$ is expanded to $\langle 12354, 12454, 12453\rangle$. Two instances of the new motif have the same length of 75. The instances of the extended motif are presented in the plot (c) of figure 2.4. The *deep search* algorithm finishes when there exists no further extension to the list of root motifs and their extended forms.

Random Projection Method

The *random projection* motif discovery algorithm for time series data was introduced in [12]. It is briefly explained in the next section, as it provides a very good insight into the problem and can be used in time series classification problem. Algorithm 4 is the pseudocode of *random projection*.

2.3.2 Time Series Classification

The goal was to find class motifs. This task can be easily achieved by concatenating the SAX words of all time series in the same class and then running the proposed motif discovery algorithm. The pseudocode of this process is presented as algorithm 5. So far, it is explained how to discover class motifs. The next steps of the classification problem are to use class motifs as shapelet candidates, find the closest subsequences of each time

Algorithm 4: Random Projection motif discovery method

Require $T \leftarrow$ Time series data points, $n, w, \alpha \leftarrow$ window size, word size and alphabet size, $I \leftarrow$ number of iterations, $H \leftarrow$ hash function, $p \leftarrow$ overlap parameter, $R \leftarrow iter$-by-2 random integer numbers between 1 and w, such that the values of each row not be the same, $\gamma \leftarrow$ threshold to discover a motif
$k \leftarrow 1$
for $i \leftarrow 1$ **to** $|T| - n + 1$ **do**
    $S \leftarrow T[i : i + n - 1]$
    $SAX_S \leftarrow SAXMaker(S, w, \alpha)$
    if $i == 1$ **then**
        $SAX_{List}[k] \leftarrow SAX_S$
        $Index[k] \leftarrow i$
        $k \leftarrow k + 1$
    end
    if $SAX_{List}[k] \neq SAX_S$ **then**
        $k \leftarrow k + 1$
        $SAX_{List}[k] \leftarrow SAX_S$
        $Index[k] \leftarrow i$
    end
end
$C \leftarrow [0]_{|SAX_{List}|*|SAX_{List}|}$
for $iter \leftarrow 1$ **to** I **do**
    Map the starting point of SAX_{List} words into *buckets* using the hash function H and $R(iter, :)$
    Update C using *buckets*
end
for $i \leftarrow 1$ **to** $|SAX_{List}|$ **do**
    for $j \leftarrow \max(1, [i - np])$ **to** $\min(|SAX_{List}|, [i + np])$ **do**
        $C[i, j] \leftarrow 0$
    end
end
for $i \leftarrow 1$ **to** $|C| - 1$ **do**
    for $j \leftarrow i + 1$ **to** $|C|$ **do**
        if $C[i, j] > \gamma$ **then**
            $Motif \leftarrow Append(Motif, [i, j])$
        end
    end
end
return $Motif$

series to the candidates, and finally compute their minimum distance. An approach to finding the closest subsequences to each candidate is shown in algorithm 6. Hamming distance is used for SAX word matching, which gives some potential closest matches in each time series. Applying the numerosity reduction rule allows us to find matches with different lengths. The output of this algorithm is a number of closest subsequences to each class motif.

The pseudocode for creating the feature matrix is presented in algorithm 7. In this algorithm, the indices i and j loop over the set of class motifs and instances of the i^{th} class motif, respectively. In the inner for-loops, indices m and n loop over all time series in the dataset and closest match(es) of the i^{th} class motif in the m^{th} time series, respectively. Once the feature representation of all time series is built, one can create decision tree classifiers and find the best class motif(s).

Algorithm 5: Finding class motifs

```
Require D ← Dataset, L ← labels, n ← window size, w ← word size and α ← alphabet size
D ←Sort D based on labels
N ← Number of time series in D
C ← Unique(L)
for i ← 1 to N do
    for j ← 1 to |C| do
        if L[i] == C[j] then
            SAX[i] ← SAXMaker(D[i], n, w, α)
            ConcSAX[j] ← Concatenate(ConcSAX[j], SAX[i])
        end
    end
end
Motif ← Find motifs using the Motif discovery algorithm 2
Motif ← Update the Motif list and make sure each instance of a motif completely falls in one time series
return Motif
```

2.4 Numerical evaluations

Deep search has the advantage of finding motif instances with different lengths, due to (a) numerosity reduction and (b) word expansion. In fact, *deep search* is capable of taking the motifs that are discovered by other algorithms and attempts to expand them. The classification approach that is presented in this study is applied to datasets of UCR Archive [10]. The experiments have been performed using the statistical software SAS 9.4. *TSA.SIMILARITY* function of *PROC TSMODEL* was used for time domain similarity computation to get exact distances (based on DTW.) This procedure is operated by Cloud Analytic Services (CAS). *PROC HPSPLIT* and *PROC IML* are used to create classification trees and for general programming, respectively. In this set of numerical experiments, *deep search* is used to discover the class motifs. Table 2.2 shows the results of the experiments on 6 datasets. The results are compared to 1-NN Euclidean distance (ED) and DTW classifiers [10].

Motifs provide an explanation as to why a certain label is predicted. The optimal motif can be regarded as the signature of a certain class. For example, figures 2.5 and 2.6 show the optimal motif instances in Beetle/Fly (class 1) and ShapeletSim datasets (class 2).

2.5 Conclusion

This study presented a new motif discovery algorithm, *deep search*, and an important application of motifs in time series classification. *Deep search* is based on the SAX representation of a time series and is capable of finding motif instances with different lengths.

This study extended the idea of shapelets, similar subsequences with a fixed length,

Algorithm 6: Finding the potential closest subsequence match(es) to class motifs

Require $Motif \leftarrow$ class motifs, $SAX \leftarrow$ SAX words of the dataset D, $Index \leftarrow$ positions associated with SAX words for each time series
$N \leftarrow$ Number of time series in D
for $i \leftarrow 1$ **to** $|Motif|$ **do**
    $L \leftarrow$ Number of SAX words making $Motif(i)$
    for $j \leftarrow 1$ **to** N **do**
        $s \leftarrow$ SAX word size of time series j
        $Distance \leftarrow \infty$
        for $k \leftarrow 1$ **to** $s - L + 1$ **do**
            if $Distance < HamDist\,(SAX[j, k : k + L - 1], Motif(i))$ **then**
                $Distance \leftarrow HamDist\,(SAX[j, k : k + L - 1], Motif(i))$
                $StartPt[i, j] \leftarrow \{k\}$
                if $k < s - L + 1$ **then**
                    $EndPt[i, j] \leftarrow \{Index(k + L) - 1\}$
                else
                    $EndPt[i, j] \leftarrow \{|D[j]|\}$
                end
            else if $Distance == HamDist(SAX[j, k : k + L - 1])$ **then**
                $StartPt[i, j] \leftarrow Append(StartPt[i, j], k)$
                if $k < s - L + 1$ **then**
                    $EndPt[i, j] \leftarrow Append(EndPt[i, j], Index(k + L) - 1)$
                else
                    $EndPt[i, j] \leftarrow Append(EndPt[i, j], |D[j]|)$
                end
            end
        end
    end
end
return $StartPt, EndPt$

Algorithm 7: Creating feature matrix by class motifs and DTW distance

Require $CMotif \leftarrow$ class motifs, $SAX \leftarrow$ SAX words of the dataset D, $Index \leftarrow$ positions associated with SAX words for each time series
$Matches \leftarrow$ Find the potential closest subsequence match(es) to class motifs based on Hamming distance
for $i \leftarrow 1$ **to** $|CMotif|$ **do**
  for $m \leftarrow 1$ **to** $|D|$ **do**
    if *Any instance of* $CMotif(i)$ *exists in* $D(m)$ **then**
      $dist \leftarrow 0$
    else
      $dist \leftarrow \infty$
      for $j \leftarrow 1$ **to** $|CMotif(i)|$ **do**
        for $n \leftarrow 1$ **to** $|Matches(i, m)|$ **do**
          if $DTW(CMotif(i, j), Matches(i, m, n)) < dist$ **then**
            $dist \leftarrow DTW(CMotif(i, j), Matches(i, m, n))$
          end
        end
      end
    end
    $Feature(i, m) \leftarrow dist$
  end
end
return $Feature$

Table 2.2: Results of the motif tree classifier error rate vs. 1-NN ED and 1-NN DTW classifiers error rate. The parameters for the motif search is presented. α and w are the size of the alphabet set and the word length, respectively. In each dataset, the best performance is highlighted in bold and an improvement over 10% is shown by †.

Dataset	1-NN ED	1-NN DTW	Motif tree	Motif(s)	Base length	α	w
Beetle/Fly	0.250	0.300	**0.000** †	$\langle EECBBAB, EECBBBB\rangle$	42	5	7
Bird/Chicken	0.450	0.250	**0.200**	$\langle BAABCEE, BABBDEE\rangle$	84	5	7
Coffee	**0.000**	**0.000**	0.036	$\langle CDECHGA, CEECHGA\rangle$	84	8	7
Gun-Point	0.087	0.093	**0.060**	$\langle GGFCBBA, GGECBBA\rangle$	70	7	7
				$\langle AABCFGG, AABDFGG\rangle$			
Ham	0.400	0.533	**0.248** †	$\langle DDDDDDH, DDDCDDH\rangle$	140	8	7
ShapeletSim	0.461	0.350	**0.000** †	$\langle ABDFHF, ABDGHF\rangle$	30	8	6

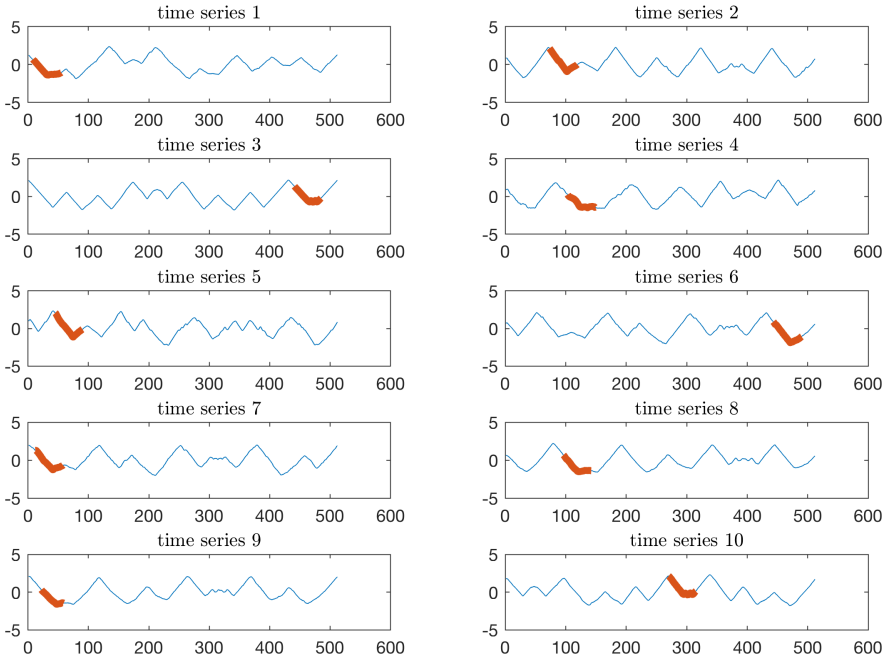

Figure 2.5: The optimal motif instances that are found in class 1 of Beetle/Fly dataset. The instances represent the motif $\langle EECBBAB, EECBBBB \rangle$.

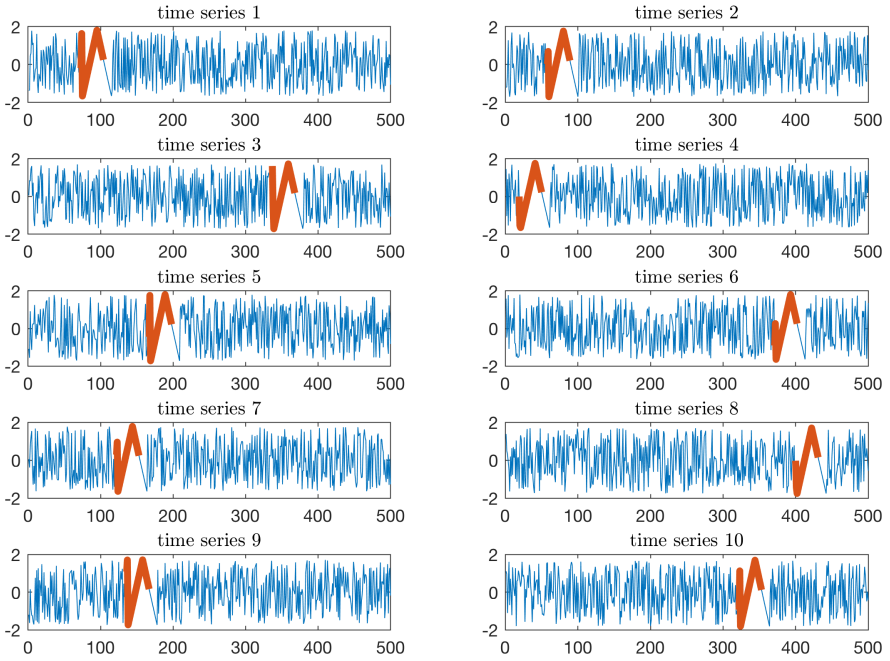

Figure 2.6: The optimal motif instances that are found in class 2 of ShapeletSim dataset. The instances represent the motif $\langle ABDFHF, ABDGHF \rangle$.

to similar subsequences with variable lengths. In this study, classification tree method is used for classification. For the future work, a combination of more advanced methods such as support vector machine and Bayesian network with features created by class motifs is suggested.

Chapter 3

Symbolic Aggregate Approximation Distance Measure

3.1 Introduction

Symbolic Aggregate Approximation (SAX) is a well-known dimension reduction technique that represents a time series with a word (that consists of some symbols) in a lower dimensional space [34]. This technique is widely accepted in the data mining community since its introduction [7, 16, 26, 32, 34, 37, 49]. The space-saving gained by SAX comes at a cost of losing some information from the original time series. Specifically, SAX takes the average values of different segments of a time series and maps them to a set of symbols. The big challenge is how to measure similarity/dissimilarity between SAX words and how to relate it to that of the original two time series, as it highly impacts the quality of the underlying task output. For instance, failure in this process causes false classification for some *time series classification* problems [48].

SAX distance provides a numerical similarity measure between two time series $C =$

$c_1, ..., c_n$ and $Q = q_1, ..., q_n$ using the information provided by SAX words $\hat{C}$ and $\hat{Q}$. The aims of this chapter are as follows:

1. Present a new similarity measure between $\hat{C}$ and $\hat{Q}$ that not only provides lower bounds on the Euclidean distance between C and Q, but also is a tighter bound compared to that of the original SAX procedure [34].

2. Observe this problem from a mathematical optimization perspective. A quadratic optimization program is used to model the Euclidean distance between C and Q. Lagrangian multipliers and Karush-Kuhn-Tucker (KKT) conditions are common tools to analyze this type of program.

The rest of this chapter is organized as follows. First, in section 3.2 a review of existing methods in this domain is presented. In sections 3.3 and 3.4, two optimization models for *first SAX distance* and *SAX with trend distance* are presented. In section 3.5, numerical experiments based on the initial solution for optimized *SAX with trend distance* are provided followed by the conclusion in section 3.6.

3.2 Literature Review

In what follows, SAX representation and several distance measures are briefly reviewed.

3.2.1 Symbolic Aggregate Approximation

A detailed review of SAX representation is provided in [34]. The whole idea behind SAX is to represent the input normalized time series, $C = c_1, ..., c_n$, with a small number of symbols. This is achieved in two steps. First, every n/w continuous points (with no

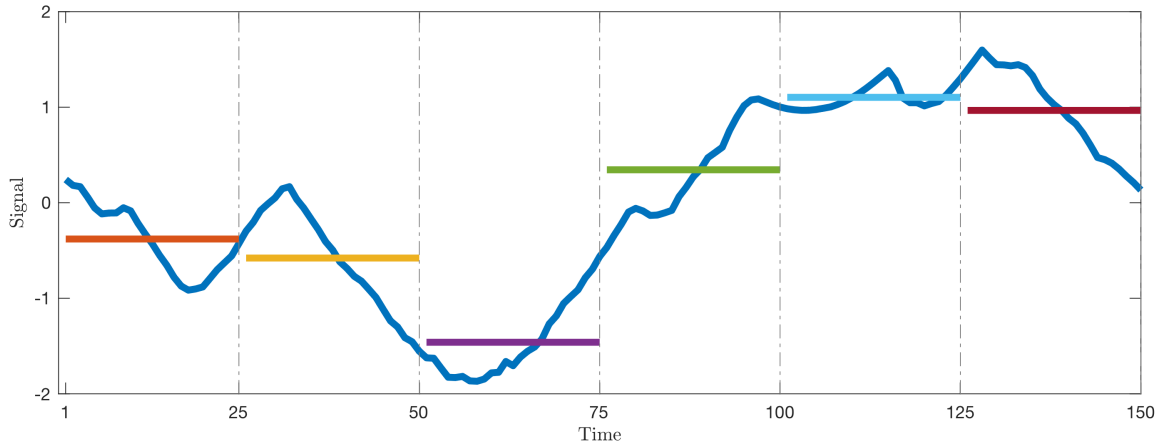

Figure 3.1: A time series with length 150 and its PAA representation with the length of 6 are plotted here. The value of each bin is corresponding to 25 consecutive data points of the original time series.

overlap and suppose n is divisible by w) are aggregated in order to transform C to $\bar{C} = \bar{c}_1, ..., \bar{c}_w$ as follows:

$$\bar{c}_j = \frac{\sum_{i=\frac{n}{w}(j-1)+1}^{\frac{n}{w}j} c_i}{n/w} \quad \text{for } j = 1, ..., w. \tag{3.1}$$

By doing this, the size of the time series is reduced from n to w. This representation is called piecewise aggregate approximation (PAA). Figure 3.1 shows a time series and its PAA representation. In this example, $n = 150$ and $w = 6$. In the second step, a symbol is assigned to each computed average. Following definitions provide more details regarding the SAX technique:

Definition *(Word)* A word with size w is a sequence of letters represented by $\hat{C} = \hat{c}_1, ..., \hat{c}_w$, where each element can take a symbol from the set *Alphabet*.

Definition *(Breakpoints)* Based on the size of alphabet set, α, breakpoints are a sequence of $\alpha - 1$ points obtained from the standard Normal distribution along with two points

Table 3.1: Breakpoints of the standard Normal distribution that has equiprobable regions based on the alphabet size, α are shown [34].

α	3	4	5	6	7	8	9	10
β_0	-0.43	-0.67	-0.84	-0.97	-1.07	-1.15	-1.22	-1.28
β_1	0.43	0	-0.25	-0.43	-0.57	-0.67	-0.76	-0.84
β_2		0.67	0.25	0	-0.18	-0.32	-0.43	-0.52
β_3			0.84	0.43	0.18	0	-0.14	-0.25
β_4				0.97	0.57	0.32	0.14	0
β_5					1.07	0.67	0.43	0.25
β_6						1.15	0.76	0.52
β_7							1.22	0.84
β_8								1.28

$-\infty$ and ∞ such that the area under the Normal curve between each two consecutive points is $1/\alpha$.

Table 3.1, obtained from [34], introduces the breakpoints based on the alphabet size α ranging from 3 to 10. For example, if $\alpha = 3$, then the breakpoints are -0.43 and 0.43 and if it is 4, then the breakpoints are -0.67, 0 and 0.67. To obtain a SAX representation of a given time series C, one should take the following steps:

Step 1: Determine the word size, w, and alphabet size, α

Step 2: Normalize the sequence (subtract the mean and divide by the standard deviation)

Step 3: Apply the PAA technique, based on the word size, to obtain $\bar{C} = \bar{c}_1, ..., \bar{c}_w$

Step 4: For each element $\bar{c}_j$ $(j = 1, ..., w)$, assign a letter from the alphabet set based on the intervals created by consecutive breakpoints to obtain $\hat{c}_j$

Figure 3.2 shows the time series in figure 3.1 and its SAX representation. In this example, word size and alphabet size are set to 6 and 5, respectively. Note that each

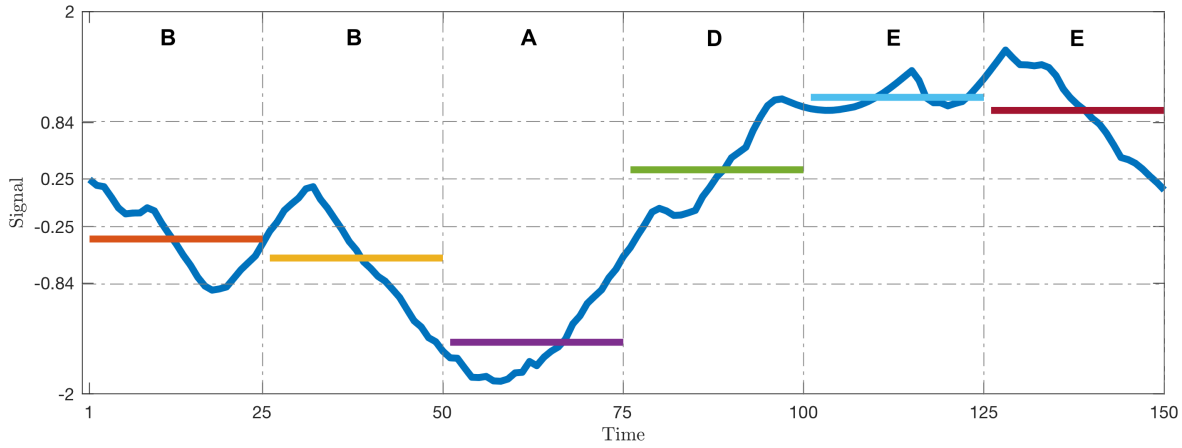

Figure 3.2: SAX representation of the time series in figure 3.1 is the word *BBADEE*.

interval $(-\infty, -0.84)$, ..., $(0.84, \infty)$ has the equal area under the Normal distribution curve and because of that, every letter in the alphabet set $\{A, B, C, D, E\}$ has the same probability to appear at each position of the word. The output of the algorithm is the word *BBADEE*. This method converts the whole time series into one SAX word.

3.2.2 Related Work

Euclidean distance and dynamic time warping are widely used similarity measures in time series analysis. However, it is impossible to use them directly to evaluate the similarity between SAX words, due to the lack of information after taking averages at each bin. In what follows, the studies that attempt to provide similarity measures based on SAX representation are summarized.

First SAX Distance

The first similarity measure for SAX words was introduced in [34]. Motivated by Euclidean distance, *first SAX distance* computes the lower bound distance between letters

and sums over their squared values. The following function is the proposed minimal distance between two letters r and c:

$$dist\,(r,c) = \begin{cases} 0, & \text{if } |r-c| \leq 1 \\ \beta_{\max(r,c)-1} - \beta_{\min(r,c)}, & \text{otherwise} \end{cases} \tag{3.2}$$

Accordingly, the distance between $\hat{C}$ and $\hat{Q}$ is:

$$MINDIST(\hat{C},\hat{Q}) = \sqrt{\frac{n}{w}}\sqrt{\sum_{j=1}^{w} dist\,(\hat{c}_j,\hat{q}_j)^2} \tag{3.3}$$

The factor n/w is the number of points at each bin. The distance between letters is suggested in a way that the $MINDIST$ function lower-bounds both PAA distance and Euclidean distance [34].

SAX with Trend Distance

One of the limitations in SAX dimension reduction is that the trend of data points at each bin is not taken into account in the formulation [48]. That is, the only feature extracted from each bin is the average of the associated data points. This drawback makes the $MINDIST$ function (from *first SAX distance*) not able to distinguish between two of the same letters in different words and assigns their distance to zero. Figure 3.3 shows four sequences with the same average, 0.7, whose shapes are obviously different from each other. The contribution of *SAX with trend distance* is to collect the distance of the starting point and the average value at each bin (in addition to the average of data points) and present another distance function. Accordingly, the similarity between $\hat{C}$ and

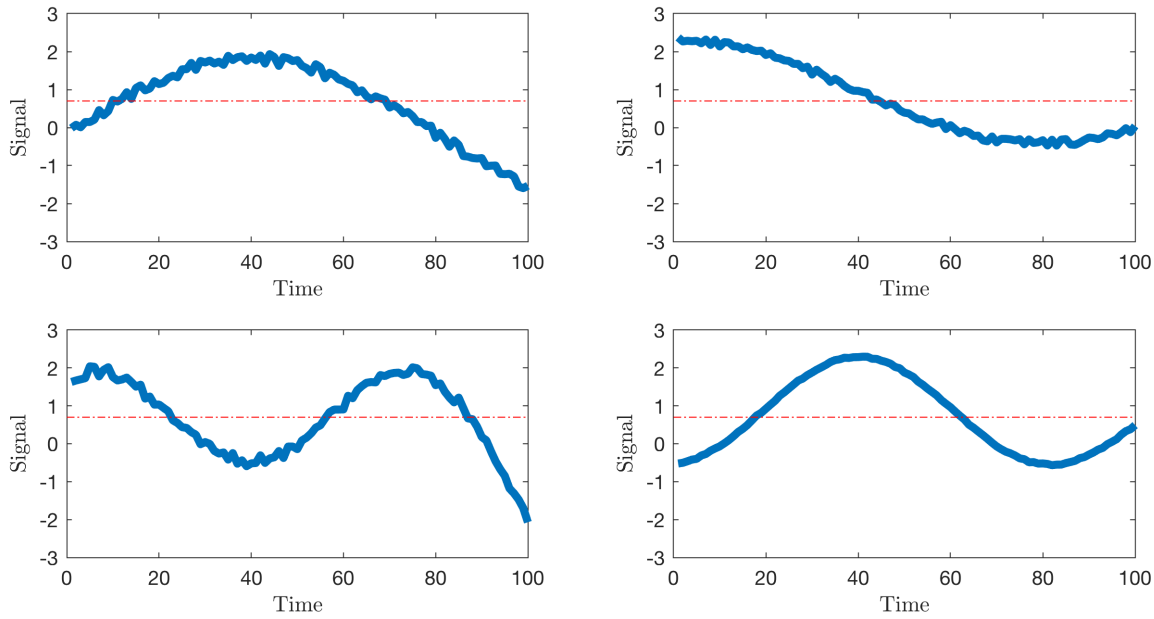

Figure 3.3: SAX representation cannot distinguish between the four segments as they all have the same average. The average is 0.7.

$\hat{Q}$ is:

$$TDIST\left(\hat{C},\hat{Q}\right)=\sqrt{\frac{n}{w}}\sqrt{\sum_{j=1}^{w}\left(dist\left(\hat{c}_j,\hat{q}_j\right)^2+\frac{w}{n}td(q_j,c_j)^2\right)}, \tag{3.4}$$

where

$$td(q_j,c_j)=\sqrt{(\Delta q_s^j-\Delta c_s^j)^2+(\Delta q_e^j-\Delta c_e^j)^2}, \tag{3.5}$$

and Δq_s^j and Δc_s^j are the differences between the time series value Q and C at the starting point of bin j and their average values. Similarly, Δq_e^j and Δc_e^j are the differences between the time series value Q and C at the ending point of bin j and their average values.

Similar to *first SAX distance*, *SAX with trend distance* has the property of lower bounding the PAA and Euclidean distance.

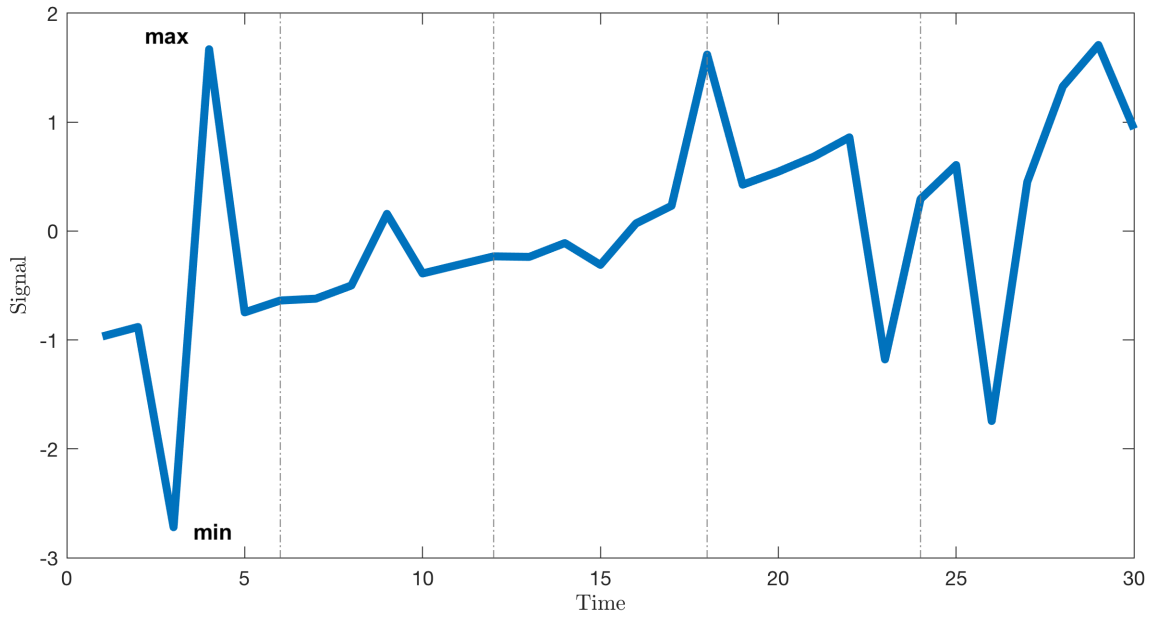

Figure 3.4: The average of the first bin is not representative of the behavior of the sequence. ESAX records the minimum and maximum of each segment in addition to the average.

Extended SAX Distance

The original SAX dimension reduction has another disadvantage in that it can miss some important patterns. Examples of such important patterns are extreme values (minimum and maximum) at each equal segment regions. This makes the SAX representation less efficient in some domains such as finance and economics [37]. Figure 3.4 shows an example, where the maximum and minimum in a segment are the global maximum and minimum of the sequence. To overcome this, the *Extended SAX* (ESAX) suggests recording the maximum and the minimum of each bin in addition to the average. The extra recorded features are also converted to letters based on the original breakpoints. Hence, the size of recorded string increases from w to $3w$. The distance between two ESAX words is computed similar to that of *first SAX distance*.

SAX Distance Using Standard Deviation

For this method, the average and the standard deviation of the values at each segment are computed [59]. The average values are converted to the symbols of the alphabet set, while the standard deviations are recorded as a vector of features. The distance between two SAX words $\hat{C}$ and $\hat{Q}$ along with their vector of standard deviation is computed as follows:

$$SAXSD\left(\hat{C},\hat{Q}\right)=\sqrt{\frac{n}{w}}\sqrt{\sum_{j=1}^{w}\left(dist\left(\hat{c}_j,\hat{q}_j\right)^2+SD\left(\hat{c}_j,\hat{q}_j\right)^2\right)}, \tag{3.6}$$

where

$$SD\left(\hat{c}_j,\hat{q}_j\right)=\sqrt{\frac{w}{n}\sum_{i=(j-1)\frac{n}{w}+1}^{j\frac{n}{w}}(c_i-\bar{c}_j)^2}-\sqrt{\frac{w}{n}\sum_{i=(j-1)\frac{n}{w}+1}^{j\frac{n}{w}}(q_i-\bar{q}_j)^2}\quad \text{for } j=1,...,w \tag{3.7}$$

This similarity measure has been shown to provide a lower bound on the actual Euclidean distance between C and Q. It has also achieved the highest *classification accuracy* compared to *first SAX distance*, *SAX with trend distance* and *ESAX distance* on UCR datasets [10, 59].

Figure 3.5 and table 3.2 show two time series and the statistics used to create different representations, namely *ESAX*, *SAX with trend* and *SAX with standard deviation*. Tables 3.3 and 3.4 show the different representations.

3.3 Optimization Problem: First SAX Model

In this section, an optimization framework for computing words similarity is presented. The following is the statement of the problem:

Problem Statement: *Let* $\hat{C}=\hat{c}_1,...,\hat{c}_w$ *and* $\hat{Q}=\hat{q}_1,...,\hat{q}_w$ *be two SAX words.*

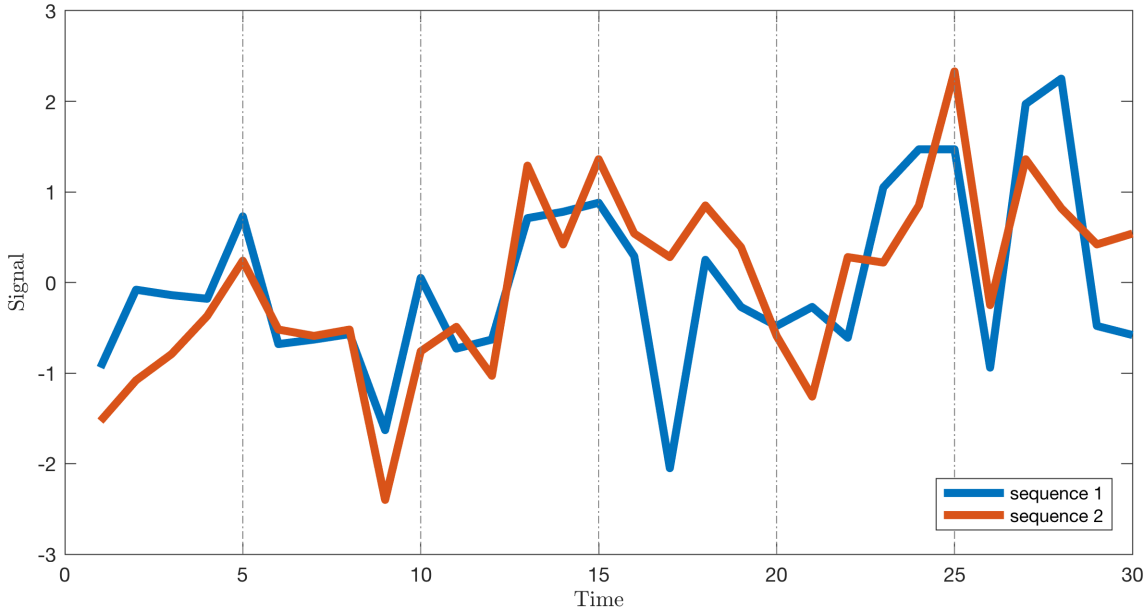

Figure 3.5: Four breakpoints are $-0.84, -0.25, 0.25$ and 0.25. The original SAX representation for sequence 1 and 2 are *CBCBDD* and *BADDDD*.

Table 3.2: The statistics used for *ESAX*, *SAX with trend* and *SAX with standard deviation* representations are reported. PAA, Min, Max, SD and TD stand for piecewise aggregate approximation, minimum, maximum, standard deviation and trend distance.

Bin	PAA 1	PAA 2	SD 1	SD 2	Min 1	Min 2	Max 1	Max 2	TD 1	TD 2
1	-0.12	-0.71	0.53	0.61	-0.94	-1.53	0.73	0.24	-0.82	-0.82
2	-0.69	-0.96	0.54	0.73	-1.63	-2.40	0.05	-0.52	0.01	0.44
3	0.20	0.31	0.72	0.95	-0.73	-1.03	0.88	1.36	-0.93	-0.80
4	-0.45	0.29	0.85	0.48	-2.05	-0.59	0.29	0.85	0.74	0.24
5	0.62	0.48	0.89	1.16	-0.61	-1.26	1.47	2.33	-0.89	-1.74
6	0.44	0.58	1.37	0.53	-0.94	-0.25	2.25	1.36	-1.38	-0.83
									-1.02	-0.04

Table 3.3: *SAX with trend* and *SAX with standard deviation* representations for two time series in figure 3.5 are shown.

Sequence	*SAX with trend*	*SAX with standard deviation*
1	$_{-0.82}C_{0.01}B_{-0.93}C_{0.74}B_{-0.89}D_{-1.38}D_{-1.02}$	$C_{0.53}B_{0.54}C_{0.72}B_{0.85}D_{0.89}D_{1.37}$
2	$_{-0.82}B_{0.44}A_{-0.80}D_{0.24}D_{-1.74}D_{-0.83}D_{-0.04}$	$B_{0.61}A_{0.73}D_{0.95}D_{0.48}D_{1.16}D_{0.53}$

Table 3.4: *ESAX* representation for sequences in figure 3.5 are shown.

Sequence	*ESAX*
1	*ACD,BAC,BCE,DAB,BDE,ADE*
2	*ABC,BAA,ADE,EDB,ADE,BED*

Also let α be the alphabet size and $\beta_0, ..., \beta_{\alpha-2}$ be the breakpoints associated with the SAX representation. Find the minimum Euclidean distance between C and Q of length n, which are the unknown and original normalized time series associated with $\hat{C}$ and $\hat{Q}$.

Example: Figure 3.6 and 3.7 show examples of different time series whose SAX representations are $\hat{C} = BDEBC$ and $\hat{Q} = BEBEB$, respectively. In this section, it is aimed to find a pair of time series, one for $\hat{C}$ and one for $\hat{Q}$ that corresponds to the minimum Euclidean distance.

Among all time series of length n that map to the words $\hat{C}$ and $\hat{Q}$, the following quadratically constrained quadratic program (QCQP) finds the two that have minimum possible Euclidean distance. It means that any other two time series that map to $\hat{C}$ and $\hat{Q}$ have equal or larger Euclidean distance. In order to follow the QCQP definition, the objective function is squared. It does not change the optimal values of two time series

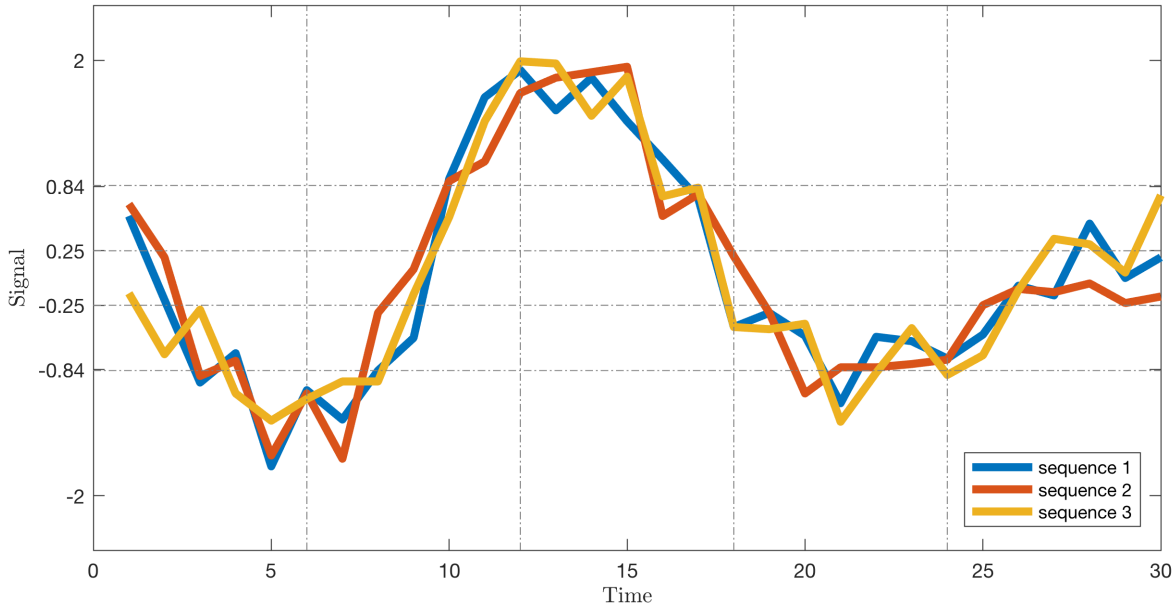

Figure 3.6: Examples of three different sequences with the same SAX representation $BDEBC$ are provided.

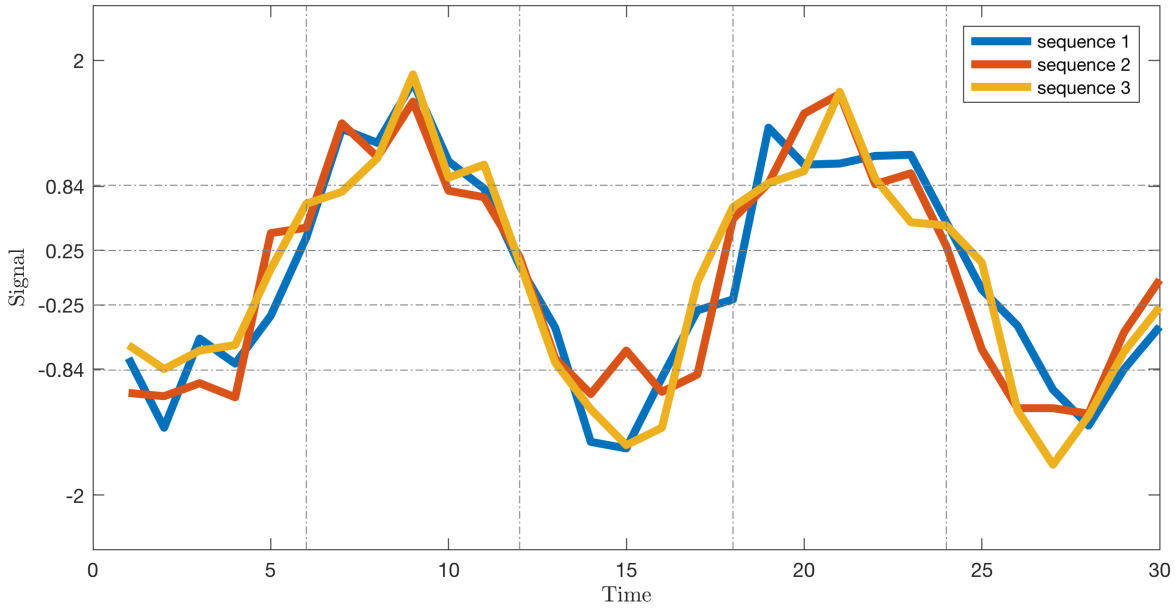

Figure 3.7: Examples of three different sequences with the same SAX representation $BEBEB$ are provided.

data points. This program is labeled as **P1**.

$$\min Z = \sum_{i=1}^{n} (q_i - c_i)^2 \tag{3.8}$$

subject to:

$$\sum_{i=1}^{n} c_i = 0 \tag{3.9}$$

$$\sum_{i=1}^{n} q_i = 0 \tag{3.10}$$

$$\sum_{i=1}^{n} c_i^2 = n - 1 \tag{3.11}$$

$$\sum_{i=1}^{n} q_i^2 = n - 1 \tag{3.12}$$

$$\frac{w}{n} \sum_{i=\frac{n}{w}(j-1)+1}^{\frac{n}{w}j} c_i \geq \beta_{LB}(\hat{c}_j) \qquad \text{for } j = 1, ..., w \tag{3.13}$$

$$\frac{w}{n} \sum_{i=\frac{n}{w}(j-1)+1}^{\frac{n}{w}j} q_i \geq \beta_{LB}(\hat{q}_j) \qquad \text{for } j = 1, ..., w \tag{3.14}$$

$$\frac{w}{n} \sum_{i=\frac{n}{w}(j-1)+1}^{\frac{n}{w}j} c_i \leq \beta_{UB}(\hat{c}_j) \qquad \text{for } j = 1, ..., w \tag{3.15}$$

$$\frac{w}{n} \sum_{i=\frac{n}{w}(j-1)+1}^{\frac{n}{w}j} q_i \leq \beta_{UB}(\hat{q}_j) \qquad \text{for } j = 1, ..., w \tag{3.16}$$

$$q_i, c_i \in R \qquad \text{for } i = 1, ..., n \tag{3.17}$$

In this formulation, index $i = 1, ..., n$ refers to data points and index $j = 1, ..., w$ indicates the bin numbers. Note that $\hat{q}_j$ and $\hat{c}_j$ are known letters, while c_i and q_i are decision variables. Inequalities 3.13, 3.14, 3.15, 3.16 provide conditions that the two time series C

and Q convert to the words $\hat{C}$ and $\hat{Q}$, respectively. Let the optimal value of the problem **P1** be $z^*_{\mathbf{P1}}$.

The first lower bound for SAX was introduced in [34]. They provided the following function as a distance between two letters r and c:

$$dist\,(r,c) = \begin{cases} 0, & \text{if } |r-c| \leq 1 \\ \beta_{\max(r,c)-1} - \beta_{\min(r,c)}, & \text{otherwise} \end{cases} \tag{3.18}$$

and their proposed lower bound was:

$$MINDIST(\hat{C},\hat{Q}) = \sqrt{\frac{n}{w}}\sqrt{\sum_{j=1}^{w} dist\,(\hat{c}_j,\hat{q}_j)^2} \tag{3.19}$$

They proved that $MINDIST$ function is a lower bound for both the PAA distance and the Euclidean distance. However, they did not take into account the two properties of a normalized time series, that the mean is zero and the standard deviation is 1. In terms of mathematical programming, they tried to find the optimal solution for the following quadratic program (**P2**):

$$\min Z = \sum_{i=1}^{n} (q_i - c_i)^2 \tag{3.20}$$

subject to:

$$\frac{w}{n} \sum_{i=\frac{n}{w}(j-1)+1}^{\frac{n}{w}j} c_i \geq \beta_{LB}(\hat{c}_j) \qquad \text{for } j = 1, ..., w \tag{3.21}$$

$$\frac{w}{n} \sum_{i=\frac{n}{w}(j-1)+1}^{\frac{n}{w}j} q_i \geq \beta_{LB}(\hat{q}_j) \qquad \text{for } j = 1, ..., w \tag{3.22}$$

$$\frac{w}{n} \sum_{i=\frac{n}{w}(j-1)+1}^{\frac{n}{w}j} c_i \leq \beta_{UB}(\hat{c}_j) \qquad \text{for } j = 1, ..., w \tag{3.23}$$

$$\frac{w}{n} \sum_{i=\frac{n}{w}(j-1)+1}^{\frac{n}{w}j} q_i \leq \beta_{UB}(\hat{q}_j) \qquad \text{for } j = 1, ..., w \tag{3.24}$$

$$q_i, c_i \in R \qquad \text{for } i = 1, ..., n \tag{3.25}$$

It can be verified that **P2** is an additively separable optimization problem and its variables optimal values are those found in [34]. It is clear that $z^*_{\mathbf{P2}}$, the optimal value of the problem **P2**, is less than or equal to $z^*_{\mathbf{P1}}$ since four constraints of **P1** (equations 3.9, 3.10, 3.11 and 3.12) are relaxed in **P2**. Now, the quadratic program **P3** is built as follows:

$$\min Z = \sum_{i=1}^{n} (q_i - c_i)^2 \tag{3.26}$$

subject to:

$$\sum_{i=1}^{n}(q_i - c_i) = 0 \tag{3.27}$$

$$\frac{w}{n}\sum_{i=\frac{n}{w}(j-1)+1}^{\frac{n}{w}j} c_i \geq \beta_{LB}(\hat{c}_j) \qquad \text{for } j = 1, ..., w \tag{3.28}$$

$$\frac{w}{n}\sum_{i=\frac{n}{w}(j-1)+1}^{\frac{n}{w}j} q_i \geq \beta_{LB}(\hat{q}_j) \qquad \text{for } j = 1, ..., w \tag{3.29}$$

$$\frac{w}{n}\sum_{i=\frac{n}{w}(j-1)+1}^{\frac{n}{w}j} c_i \leq \beta_{UB}(\hat{c}_j) \qquad \text{for } j = 1, ..., w \tag{3.30}$$

$$\frac{w}{n}\sum_{i=\frac{n}{w}(j-1)+1}^{\frac{n}{w}j} q_i \leq \beta_{UB}(\hat{q}_j) \qquad \text{for } j = 1, ..., w \tag{3.31}$$

$$q_i, c_i \in R \qquad \text{for } i = 1, ..., n \tag{3.32}$$

The optimal value of this program is $z^*_{\mathbf{P3}}$. The difference between **P2** and **P3** is that in **P3**, a weak form of constraints 3.9 and 3.10 is added back to **P2**. Therefore, $z^*_{\mathbf{P2}} \leq z^*_{\mathbf{P3}} \leq z^*_{\mathbf{P1}}$. The optimal solution of **P3** can be found using Karush-Kuhn-Tucker (KKT) conditions. Let $\mathbf{Q} = (q_1, ..., q_n)$, $\mathbf{C} = (c_1, ..., c_n)$, $\mathbf{L^q} = (l^q_1, ..., l^q_w)$, $\mathbf{U^q} = (u^q_1, ..., u^q_w)$, $\mathbf{L^c} = (l^c_1, ..., l^c_w)$ and $\mathbf{U^c} = (u^c_1, ..., u^c_w)$. The Lagrangian function of **P3** is constructed as

follows:

$$\begin{aligned}\mathcal{L}\left(\mathbf{Q},\mathbf{C},\mathbf{L^q},\mathbf{U^q},\mathbf{L^c},\mathbf{U^c},\lambda\right) = &\sum_{i=1}^{n}\left(q_i-c_i\right)^2 \\ &+\lambda\sum_{i=1}^{n}(q_i-c_i) \\ &+\sum_{j=1}^{w} l_j^c\left(\beta_{LB}(\hat{c}_j)-\frac{w}{n}\sum_{i=\frac{n}{w}(j-1)+1}^{\frac{n}{w}j} c_i\right) \\ &+\sum_{j=1}^{w} l_j^q\left(\beta_{LB}(\hat{q}_j)-\frac{w}{n}\sum_{i=\frac{n}{w}(j-1)+1}^{\frac{n}{w}j} q_i\right) \\ &+\sum_{j=1}^{w} u_j^c\left(\frac{w}{n}\sum_{i=\frac{n}{w}(j-1)+1}^{\frac{n}{w}j} c_i-\beta_{UB}(\hat{c}_j)\right) \\ &+\sum_{j=1}^{w} u_j^q\left(\frac{w}{n}\sum_{i=\frac{n}{w}(j-1)+1}^{\frac{n}{w}j} q_i-\beta_{UB}(\hat{q}_j)\right)\end{aligned} \tag{3.33}$$

3.3.1 Initial Solution

The initial closed-form solution for **P3**, labeled as $\bar{c_i}$ and $\bar{q_i}$ for $i = 1, ..., n$ is provided. Based on the parameters β_{LB}, β_{UB} for all pairs of $\hat{c}_j$ and $\hat{q}_j$, this solution may or may not be optimal. The optimality criteria are provided at the end of this section, and in case the initial solution is not optimal, an algorithm is presented to find it. Let $\Delta = \{j \mid |\hat{q}_j - \hat{c}_j| > 1\}$, the indices of letters in two SAX words that are at least different by two and $\Lambda^u = \{j \mid \hat{q}_j - \hat{c}_j = u\}$ where $u \in \{-1, 0, +1\}$. Also, let the function $j(i)$ be the bin number that i falls into, i.e. $j(i) = \left\lceil \frac{i-1}{w} \right\rceil + 1$ for each index i (where $i = 1, ..., n$). The initial solution is:

Step 1: For every $i = 1, ..., n$, if $j(i) \in \Delta$, two cases happen:

1. If $\hat{q}_{j(i)} > \hat{c}_{j(i)}$, then

$$\bar{q}_i = \beta_{LB}(\hat{q}_{j(i)}) \quad \text{and} \quad \bar{c}_i = \beta_{UB}(\hat{c}_{j(i)}) \tag{3.34}$$

2. If $\hat{q}_{j(i)} < \hat{c}_{j(i)}$, then

$$\bar{q}_i = \beta_{UB}(\hat{q}_{j(i)}) \quad \text{and} \quad \bar{c}_i = \beta_{LB}(\hat{c}_{j(i)}) \tag{3.35}$$

Step 2: Compute $\sum_{i=1}^{n} \mathcal{A}(j(i) \in \Delta)(\bar{q}_i - \bar{c}_i)$ where the function $\mathcal{A}(x)$ takes 1 if x is true, and 0 otherwise.

Step 3: For every $i = 1, ..., n$, if $j(i) \notin \Delta$, three cases happen:

1. If $\sum_{i=1}^{n} \mathcal{A}(j(i) \in \Delta)(\bar{q}_i - \bar{c}_i) > 0$, then

$$\bar{q}_i - \bar{c}_i = \begin{cases} -\left(\frac{w}{n}\right) \frac{\sum_{i=1}^{n} \mathcal{A}(j(i) \in \Delta)(\bar{q}_i - \bar{c}_i)}{|\Lambda^0| + |\Lambda^{-1}|} & \text{if } j(i) \notin \Lambda^{+1} \\ 0 & \text{otherwise} \end{cases} \tag{3.36}$$

2. If $\sum_{i=1}^{n} \mathcal{A}(j(i) \in \Delta)(\bar{q}_i - \bar{c}_i) < 0$, then

$$\bar{q}_i - \bar{c}_i = \begin{cases} -\left(\frac{w}{n}\right) \frac{\sum_{i=1}^{n} \mathcal{A}(j(i) \in \Delta)(\bar{q}_i - \bar{c}_i)}{|\Lambda^0| + |\Lambda^{+1}|} & \text{if } j(i) \notin \Lambda^{-1} \\ 0 & \text{otherwise} \end{cases} \tag{3.37}$$

3. If $\sum_{i=1}^{n} \mathcal{A}(j(i) \in \Delta)(\bar{q}_i - \bar{c}_i) = 0$, then

$$\bar{q}_i - \bar{c}_i = 0 \quad \text{for } j(i) \notin \Delta \tag{3.38}$$

where $|X|$ is the number of elements in the set X.

3.3.2 Initial Solution Feasibility Condition

The initial solution provided in the last section is feasible if it satisfies all constraints of **P3**, that is:

1. It is obvious that $\sum_{i=1}^{n}(\bar{q}_i - \bar{c}_i) = 0$. In fact, the initial solution is designed to satisfy this constraint.

2. For every $j \in \Delta$, the solutions $\bar{q}_i$ and $\bar{c}_i$ (where $i = (j-1)\frac{n}{w}+1, ..., j\frac{n}{w}$) are constants whose averages at bin $j(i)$ lie on one of their associated boundaries, either the lower-bound or the upper-bound. They are feasible.

3. In order to satisfy the rest of the constraints, several conditions must hold. Without loss of generality, it is assumed that $\sum_{i=1}^{n}\mathcal{A}(j(i) \in \Delta)(\bar{q}_i - \bar{c}_i) > 0$. For every $j \notin \Delta$, three cases happen:

   (a) If $j \in \Lambda^{-1}$, then for every $i = (j-1)\frac{n}{w} + 1, ..., j\frac{n}{w}$, the condition is:

   $$-(\bar{q}_i - \bar{c}_i) < \beta_{UB}(\hat{c}_j) - \beta_{LB}(\hat{q}_j) \tag{3.39}$$

   (b) If $j \in \Lambda^{+1}$, then for every $i = (j-1)\frac{n}{w} + 1, ..., j\frac{n}{w}$, the feasibility condition holds by setting:

   $$\bar{q}_i = \bar{c}_i = \beta_{UB}(\hat{c}_j) = \beta_{LB}(\hat{q}_j) \tag{3.40}$$

   (c) If $j \in \Lambda^{0}$, then for every $i = (j-1)\frac{n}{w} + 1, ..., j\frac{n}{w}$, the condition is:

   $$-(\bar{q}_i - \bar{c}_i) < \beta_{UB}(\hat{q}_j) - \beta_{LB}(\hat{q}_j) \tag{3.41}$$

Therefore, the feasibility conditions of the initial solution are summarized into two

inequalities 3.39 and 3.41.

3.3.3 Non-Feasible Initial Solution

Suppose $\lambda > 0$ and for some $j \in \Lambda^{-1} \cup \Lambda^0$, the feasibility conditions 3.39 and 3.41 do not hold. Increasing $\bar{q}_i - \bar{c}_i$ when $j(i) \in \Delta$ does not make the solution feasible; however, decreasing $\bar{q}_i - \bar{c}_i$ when $j(i) \in \Delta$ and $\hat{q}_{j(i)} < \hat{c}_{j(i)}$ increases $\bar{q}_i - \bar{c}_i$ for all $j(i) \in \Lambda^{-1} \cup \Lambda^0$. Accordingly, creating the feasible solution, based on the current infeasible solution is quite straightforward. The approach is to take the largest $\bar{q}_i - \bar{c}_i$ for $j(i) \in \Delta$ and to decrease it (while updating $\bar{q}_i - \bar{c}_i$ for all $j(i) \in \Lambda^{-1} \cup \Lambda^0$) until the feasibility conditions 3.39 and 3.41 are met.

3.3.4 Initial Solution Optimality Condition

To check other KKT optimality conditions for the provided solution, the derivatives of the Lagrangian function are computed:

$$\frac{\partial \mathcal{L}}{\partial q_i} = 2(q_i - c_i) + \lambda - \frac{w}{n} l^q_{j(i)} + \frac{w}{n} u^q_{j(i)} \tag{3.42}$$

$$\frac{\partial \mathcal{L}}{\partial c_i} = 2(c_i - q_i) - \lambda - \frac{w}{n} l^c_{j(i)} + \frac{w}{n} u^c_{j(i)} \tag{3.43}$$

These two derivatives should be zero in the optimal solution. Suppose $\sum_{i=1}^{n} \mathcal{A}(j(i) \in \Delta)(\bar{q}_i - \bar{c}_i) > 0$. Due to complementary slackness, if a constraint is not satisfied at equality at the proposed solution, its Lagrangian multiplier should be zero. If $j(i) \in \Lambda^{-1}$ or $j(i) \in \Lambda^0$, we have $l^q_{j(i)} = u^q_{j(i)} = l^c_{j(i)} = u^c_{j(i)} = 0$ and:

$$\frac{\partial \mathcal{L}}{\partial q_i} = 2(\bar{q}_i - \bar{c}_i) + \lambda = 0 \Rightarrow \lambda = \frac{2w}{n} \left(\frac{\sum_{i=1}^{n} \mathcal{A}(j(i) \in \Delta)(\bar{q}_i - \bar{c}_i)}{|\Lambda^0| + |\Lambda^{-1}|} \right) \tag{3.44}$$

$\frac{\partial \mathcal{L}}{\partial c_i} = 0$ gives the same result. The variable λ is an unrestricted-in-sign variable as it is the Lagrangian multiplier associated with the equality constraint 3.27. If $j(i) \in \Lambda^{+1}$, variables $u^q_{j(i)} = l^c_{j(i)} = 0$ (due to the complementary slackness) and:

$$\frac{\partial \mathcal{L}}{\partial q_i} = \lambda - \frac{w}{n} l^q_{j(i)} = 0 \Rightarrow l^q_{j(i)} = \frac{n}{w}\lambda = \frac{2\sum_{i=1}^{n} \mathcal{A}(j(i) \in \Delta)(\bar{q}_i - \bar{c}_i)}{|\Lambda^0| + |\Lambda^{-1}|} \tag{3.45}$$

Similarly, the relation $\frac{\partial \mathcal{L}}{\partial c_i} = 0$ gives:

$$\frac{\partial \mathcal{L}}{\partial c_i} = -\lambda + \frac{w}{n} u^c_{j(i)} = 0 \Rightarrow u^c_{j(i)} = \frac{2\sum_{i=1}^{n} \mathcal{A}(j(i) \in \Delta)(\bar{q}_i - \bar{c}_i)}{|\Lambda^0| + |\Lambda^{-1}|} \tag{3.46}$$

For any $i = 1, ..., n$, if $j(i) \in \Delta$ and $\hat{q}_{j(i)} > \hat{c}_{j(i)}$ the initial solution is $\bar{q}_i = \beta_{LB}(\hat{q}_{j(i)})$ and $\bar{c}_i = \beta_{UB}(\hat{c}_{j(i)})$, leading to $u^q_{j(i)} = l^c_{j(i)} = 0$. Similarly, for any $i = 1, ..., n$, if $j(i) \in \Delta$ and $\hat{q}_{j(i)} < \hat{c}_{j(i)}$ we have $l^q_{j(i)} = u^c_{j(i)} = 0$ (as $\bar{q}_i = \beta_{UB}(\hat{q}_{j(i)})$ and $\bar{c}_i = \beta_{LB}(\hat{c}_{j(i)})$). If the values of $l^q_{j(i)}, l^c_{j(i)}, u^q_{j(i)}, u^c_{j(i)}$ obtained from 3.42 and 3.43 and satisfying the complementary slackness condition, were non-negative, then the initial solution $\bar{c}_i$ and $\bar{q}_i$ for $i = 1, ..., n$ is optimal (by KKT conditions). For concreteness of the discussion, optimality conditions are provided in two cases:

Case 1: $\hat{q}_{j(i)} > \hat{c}_{j(i)}$

In this case, the variables $u^q_{j(i)} = l^c_{j(i)} = 0$. Equations 3.42 and 3.43 will turn into:

$$\frac{\partial \mathcal{L}}{\partial q_i} = 2(\bar{q}_i - \bar{c}_i) + \lambda - \frac{w}{n} l^q_{j(i)} = 0 \Rightarrow l^q_{j(i)} = \frac{n}{w}\left(2(\bar{q}_i - \bar{c}_i) + \lambda\right) \tag{3.47}$$

$$\frac{\partial \mathcal{L}}{\partial c_i} = 2(\bar{c}_i - \bar{q}_i) - \lambda + \frac{w}{n} u^c_{j(i)} = 0 \Rightarrow u^c_{j(i)} = \frac{n}{w}\left(2(\bar{q}_i - \bar{c}_i) + \lambda\right) \tag{3.48}$$

For optimality, the relation $l^q_{j(i)} = u^c_{j(i)} \geq 0$ should hold. This gives:

$$\lambda \geq 2(\bar{c}_i - \bar{q}_i) = 2(\beta_{UB}(\hat{c}_{j(i)}) - \beta_{LB}(\hat{q}_{j(i)})) \tag{3.49}$$

Case 2: $\hat{q}_{j(i)} < \hat{c}_{j(i)}$

The relations are very similar to the previous case, $\hat{q}_{j(i)} > \hat{c}_{j(i)}$. The variables $u^c_{j(i)} = l^q_{j(i)} = 0$ and the optimality condition is:

$$\lambda \leq 2(\bar{c}_i - \bar{q}_i) = 2(\beta_{LB}(\hat{c}_{j(i)}) - \beta_{UB}(\hat{q}_{j(i)})) \tag{3.50}$$

At the beginning of the optimality discussion, it was assumed that $\sum_{i=1}^{n} \mathcal{A}(j(i) \in \Delta)(\bar{q}_i - \bar{c}_i) > 0$. Note that the above discussion is valid for the case $\sum_{i=1}^{n} \mathcal{A}(j(i) \in \Delta)(\bar{q}_i - \bar{c}_i) < 0$ as well.

Example 1: Consider two SAX words $Word_1 = AEBEB$ and $Word_2 = BACDE$. The alphabet size is 5 (the letters to use are A, B, C, D and E) and the breakpoints are $-0.84, -0.25, +0.25$ and $+0.84$. Other parameters are $n = 50$ and $w = 5$. In this example, $\Delta = \{2, 5\}$, $\Lambda^{+1} = \{4\}$, $\Lambda^{-1} = \{1, 3\}$ and $\Lambda^0 = \{\}$. First, the following is computed:

$$\sum_{i=1}^{50} \mathcal{A}(j(i) \in \Delta)(\bar{q}_i - \bar{c}_i) = (8.4 + 8.4) + (-2.5 - 8.4) = 5.9 \tag{3.51}$$

Since $\sum_{i=1}^{50} \mathcal{A}(j(i) \in \Delta)(\bar{q}_i - \bar{c}_i) > 0$, the difference $\bar{q}_i - \bar{c}_i = 0$ for all i such that $j(i) \in \Lambda^{+1} = \{4\}$ and $\bar{q}_i - \bar{c}_i = \left(\frac{5}{50}\right)\left(\frac{5.9}{2+0}\right) = 0.295$ for all i such that $j(i) \in \Lambda^{-1} \cup \Lambda^0 = \{1, 3\}$. The solution is optimal as it satisfies the relations 3.49 and 3.50. The new SAX distance based on the optimization problem is 6.469. The distance was computed 6.333 by the formula 3.19.

If for any $i = 1, ..., n$, two conditions 3.49 and 3.50 are not satisfied, then the initial

solution 3.34, 3.35, 3.36, 3.37 and 3.38 is not optimal and a new solution should be found.

3.3.5 Non-Optimal Initial Solution

Suppose $\lambda > 0$ (which is equivalent to $\sum_{i=1}^{n} \mathcal{A}(j(i) \in \Delta)(\bar{q}_i - \bar{c}_i) > 0$) is derived from 3.44 when forming the initial solution. Inequality 3.49 automatically holds, as its right hand side is always negative. However, the right hand side of 3.50 is positive and as the initial solution is not optimal, the relation 3.50 does not hold for some $i = 1, ..., n$ where $j(i) \in \Delta$. Let $\bar{\Delta}$ be the set of indices such that the optimality condition does not hold. Also, let $\bar{\Delta}_{(j)}$ be the j^{th} smallest element of $\bar{\Delta}$. Algorithm 8 finds the optimal solution in at most $|\bar{\Delta}|$ iterations. The intuition is that decreasing $\bar{q}_i - \bar{c}_i$ for $j(i) \in \bar{\Delta}$ makes λ decrease. In another words, λ and $\bar{c}_i - \bar{q}_i$ (for the elements that do not satisfy 3.50) move in the opposite direction, which at some point makes the two become equal and 3.50 becomes valid.

Algorithm 8: Finding the optimal solution

Require $\bar{\Delta} \leftarrow$ *the set of indices such that the optimality condition does not hold*
$j \leftarrow 1$
status $\leftarrow$ True
$\mathcal{S} \leftarrow \bar{\Delta}_{(1)}$
while *status* **do**
    Set $\lambda = 2(\bar{c}_i - \bar{q}_i)$ for all i whose bin is in $\mathcal{S}$
    if $\lambda > 2(\bar{c}_i - \bar{q}_i)$, *where i is the elements of the bin* $\bar{\Delta}_{(j+1)}$ **then**
        $j \leftarrow j + 1$
        $\mathcal{S} \leftarrow \mathcal{S} \cup \bar{\Delta}_{(j)}$
    else
        status $\leftarrow$ False
    end
end
return $\bar{q}_j - \bar{c}_j$ for $j \in \bar{\Delta}$

3.4 Optimization Problem: SAX with Trend Model

SAX with trend distance has been studied in [48]. This method not only computes the average of data points at each equal segment, but also records the distance between the first data point from the respective average. This procedure can be expanded by recording the distance between the last point of each segment (as well as the first point) from the average. In this way, the behavior of each segment is described more realistically. The optimization framework that was discussed in the last section can also be applied in this model. Formally, it is aimed to solve the following problem in this section:

Problem Statement: *Let* $\hat{C} = \hat{c}_1, ..., \hat{c}_w$ *and* $\hat{Q} = \hat{q}_1, ..., \hat{q}_w$ *be two SAX words. Also, let* α *be the alphabet size and* $\beta_0, ..., \beta_{\alpha-2}$ *be the breakpoints associated with the SAX representation. The values* $(\Delta c_1^s, \Delta c_1^e), ..., (\Delta c_w^s, \Delta c_w^e)$ *and* $(\Delta q_1^s, \Delta q_1^e), ..., (\Delta q_w^s, \Delta q_w^e)$ *are trends of* $\hat{C}$ *and* $\hat{Q}$ *at each segment* $j = 1, ..., w$*, defined by:*

$$\Delta c_j^s = c_{(j-1)\frac{n}{w}+1} - \frac{w}{n} \sum_{i=(j-1)\frac{n}{w}+1}^{j\frac{n}{w}} c_i \quad for\ j = 1, ..., w \tag{3.52}$$

$$\Delta c_j^e = c_{j\frac{n}{w}} - \frac{w}{n} \sum_{i=(j-1)\frac{n}{w}+1}^{j\frac{n}{w}} c_i \quad for\ j = 1, ..., w \tag{3.53}$$

$$\Delta q_j^s = q_{(j-1)\frac{n}{w}+1} - \frac{w}{n} \sum_{i=(j-1)\frac{n}{w}+1}^{j\frac{n}{w}} q_i \quad for\ j = 1, ..., w \tag{3.54}$$

$$\Delta q_j^e = q_{j\frac{n}{w}} - \frac{w}{n} \sum_{i=(j-1)\frac{n}{w}+1}^{j\frac{n}{w}} q_i \quad for\ j = 1, ..., w \tag{3.55}$$

The indices s and e refer to the starting point and ending point in a segment. Find the minimum Euclidean distance between C *and* Q *of length* n*, which are the unknown and original normalized time series associated with* $\hat{C}$ *and* $\hat{Q}$*.*

In the last section, the optimization program **P3** discussed a drawback with the *MINDIST* function, in that the mean is zero for a normalized series. The solution to **P3** was found using Lagrangian multipliers, which resolved the issue. Mimicking the proposed solution to **P3**, the initial solution to the new problem is:

Step 1: For every $j = 1, ..., w$, if $j \in \Delta$, two cases happen:

1. If $\hat{q}_j > \hat{c}_j$, then

$$\bar{c}_{(j-1)\frac{n}{w}+1} = \beta_{UB}(\hat{c}_j) + \Delta c_j^s \tag{3.56}$$

$$\bar{q}_{(j-1)\frac{n}{w}+1} = \beta_{LB}(\hat{q}_j) + \Delta q_j^s \tag{3.57}$$

$$\bar{c}_{j\frac{n}{w}} = \beta_{UB}(\hat{c}_j) + \Delta c_j^e \tag{3.58}$$

$$\bar{q}_{j\frac{n}{w}} = \beta_{LB}(\hat{q}_j) + \Delta q_j^e \tag{3.59}$$

$$\bar{c}_i = \beta_{UB}(\hat{c}_j) - \frac{\left(\Delta c_j^s + \Delta c_j^e\right)}{\frac{n}{w} - 2} \quad \text{for } i = (j-1)\frac{n}{w} + 2, ..., j\frac{n}{w} - 1 \tag{3.60}$$

$$\bar{q}_i = \beta_{LB}(\hat{q}_j) - \frac{\left(\Delta q_j^s + \Delta q_j^e\right)}{\frac{n}{w} - 2} \quad \text{for } i = (j-1)\frac{n}{w} + 2, ..., j\frac{n}{w} - 1 \tag{3.61}$$

2. If $\hat{q}_j < \hat{c}_j$, then

$$\bar{c}_{(j-1)\frac{n}{w}+1} = \beta_{LB}(\hat{c}_j) + \Delta c_j^s \tag{3.62}$$

$$\bar{q}_{(j-1)\frac{n}{w}+1} = \beta_{UB}(\hat{q}_j) + \Delta q_j^s \tag{3.63}$$

$$\bar{c}_{j\frac{n}{w}} = \beta_{LB}(\hat{c}_j) + \Delta c_j^e \tag{3.64}$$

$$\bar{q}_{j\frac{n}{w}} = \beta_{UB}(\hat{q}_j) + \Delta q_j^e \tag{3.65}$$

$$\bar{c}_i = \beta_{LB}(\hat{c}_j) - \frac{\left(\Delta c_j^s + \Delta c_j^e\right)}{\frac{n}{w} - 2} \quad \text{for } i = (j-1)\frac{n}{w} + 2, ..., j\frac{n}{w} - 1 \tag{3.66}$$

$$\bar{q}_i = \beta_{UB}(\hat{q}_j) - \frac{\left(\Delta q_j^s + \Delta q_j^e\right)}{\frac{n}{w} - 2} \quad \text{for } i = (j-1)\frac{n}{w} + 2, ..., j\frac{n}{w} - 1 \tag{3.67}$$

Step 2: Compute $\sum_{i=1}^{n} \mathcal{A}(j(i) \in \Delta)(\bar{q}_i - \bar{c}_i)$.

Step 3: For every $j = 1, ..., w$, if $j \notin \Delta$, three cases happen:

1. If $\sum_{i=1}^{n} \mathcal{A}(j(i) \in \Delta)(\bar{q}_i - \bar{c}_i) > 0$, then

$$u = -\left(\frac{w}{n}\right) \frac{\sum_{i=1}^{n} \mathcal{A}(j(i) \in \Delta)(\bar{q}_i - \bar{c}_i)}{|\Lambda^0| + |\Lambda^{-1}|} \tag{3.68}$$

$$\bar{q}_i - \bar{c}_i = \begin{cases} u + \Delta q_j^s - \Delta c_j^s & \text{if } j \notin \Lambda^{+1} \wedge i = (j-1)\frac{n}{w} + 1 \\ u + \Delta q_j^e - \Delta c_j^e & \text{if } j \notin \Lambda^{+1} \wedge i = j\frac{n}{w} \\ u - \frac{\Delta q_j^s + \Delta q_j^e - \Delta c_j^s - \Delta c_j^e}{\frac{n}{w} - 2} & \text{if } j \notin \Lambda^{+1} \wedge i = (j-1)\frac{n}{w} + 2, ..., j\frac{n}{w} - 1 \\ \Delta q_j^s - \Delta c_j^s & \text{if } j \in \Lambda^{+1} \wedge i = (j-1)\frac{n}{w} + 1 \\ \Delta q_j^e - \Delta c_j^e & \text{if } j \in \Lambda^{+1} \wedge i = j\frac{n}{w} \\ -\frac{\Delta q_j^s + \Delta q_j^e - \Delta c_j^s - \Delta c_j^e}{\frac{n}{w} - 2} & \text{if } j \in \Lambda^{+1} \wedge i = (j-1)\frac{n}{w} + 2, ..., j\frac{n}{w} - 1 \end{cases} \tag{3.69}$$

2. If $\sum_{i=1}^{n} \mathcal{A}(j(i) \in \Delta)(\bar{q}_i - \bar{c}_i) < 0$, then

$$u = -\left(\frac{w}{n}\right) \frac{\sum_{i=1}^{n} \mathcal{A}(j(i) \in \Delta)(\bar{q}_i - \bar{c}_i)}{|\Lambda^0| + |\Lambda^{+1}|} \tag{3.70}$$

$$\bar{q}_i - \bar{c}_i = \begin{cases} u + \Delta q_j^s - \Delta c_j^s & \text{if } j \notin \Lambda^{-1} \wedge i = (j-1)\frac{n}{w} + 1 \\ u + \Delta q_j^e - \Delta c_j^e & \text{if } j \notin \Lambda^{-1} \wedge i = j\frac{n}{w} \\ u - \frac{\Delta q_j^s + \Delta q_j^e - \Delta c_j^s - \Delta c_j^e}{\frac{n}{w} - 2} & \text{if } j \notin \Lambda^{-1} \wedge i = (j-1)\frac{n}{w} + 2, ..., j\frac{n}{w} - 1 \\ \Delta q_j^s - \Delta c_j^s & \text{if } j \in \Lambda^{-1} \wedge i = (j-1)\frac{n}{w} + 1 \\ \Delta q_j^e - \Delta c_j^e & \text{if } j \in \Lambda^{-1} \wedge i = j\frac{n}{w} \\ -\frac{\Delta q_j^s + \Delta q_j^e - \Delta c_j^s - \Delta c_j^e}{\frac{n}{w} - 2} & \text{if } j \in \Lambda^{-1} \wedge i = (j-1)\frac{n}{w} + 2, ..., j\frac{n}{w} - 1 \end{cases} \tag{3.71}$$

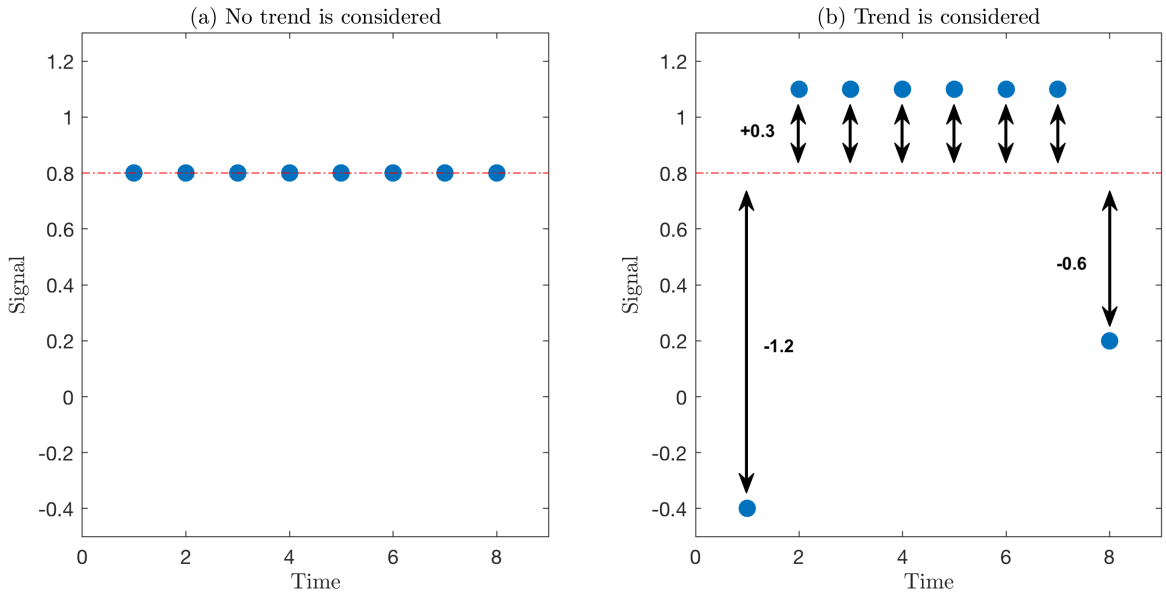

Figure 3.8: In both plots, the red dotted line is the average. In the left plot, all data points are equal to their average. In the right plot, the distance between the first point and the last point from the average are computed to be -1.2 and -0.6.

3. If $\sum_{i=1}^{n} \mathcal{A}(j(i) \in \Delta)(\bar{q}_i - \bar{c}_i) = 0$, then

$$\bar{q}_i - \bar{c}_i = \begin{cases} \Delta q_j^s - \Delta c_j^s & \text{if } i = (j-1)\frac{n}{w} + 1 \\ \Delta q_j^e - \Delta c_j^e & \text{if } i = j\frac{n}{w} \\ -\frac{\Delta q_j^s + \Delta q_j^e - \Delta c_j^s - \Delta c_j^e}{\frac{n}{w} - 2} & \text{if } i = (j-1)\frac{n}{w} + 2, ..., j\frac{n}{w} - 1 \end{cases} \tag{3.72}$$

Figure 3.8 shows the intuition behind the new optimized model. The points in each bin are moved away from the average (while the average remains the same) so as to satisfy the two trend constraints. In this example, the average at each bin is 0.8. The distance of the first and last point from the average is -1.2 and -0.6. In order for the average to remain 0.8, the rest of the points in the bin should be equal to 1.1.

3.5 Numerical Experiments

This section provides the results of the experimental evaluation using the SAX distance methods in the literature and the optimized SAX model with trend. For the optimized SAX model, the results are based on the equations provided in section 3.4. Table 3.5 summarizes the datasets that were used for this purpose. The performance of the new similarity measure is evaluated by 1-NN classifier. The classification error rates and associated parameters (word length w and alphabet size α) are reported based on the optimal classifier (see table 3.6). Figure 3.9 shows the pairwise classification error rate of all models for all 20 datasets. In order to compare the efficiency of the proposed method to those reported in the literature, the same range of w and α parameters used in [34, 48] are considered. The range of parameters are evaluated based on the following rules:

1. Word length w ranges from $2, 2^2, ...$ up to half of the time series length in each dataset.

2. Alphabet size α ranges from 3 to 10.

3. In case of the same error rate at each experiment, the smaller parameters are reported.

The compression rate (CR) for the optimized SAX model with trend is calculated by $\frac{w\lceil \log_2 \alpha \rceil + 64w}{32n}$, as each letter requires $\lceil \log_2 \alpha \rceil$ bits and each number requires 4 bytes (32 bits) to store in the memory [34].

Table 3.5: The summary of datasets used for numerical experiments is given.

No.	Dataset	Number of classes	Size of training set	Size of testing set	Length of time series
1	50Words	50	450	455	270
2	Adiac	37	390	391	176
3	Beef	5	30	30	470
4	CBF	3	30	900	128
5	Coffee	2	28	28	286
6	ECG200	2	100	100	96
7	FaceFour	4	24	88	350
8	FaceAll	14	560	1690	131
9	Fish	7	175	175	463
10	GunPoint	2	50	150	150
11	Lightning2	2	60	61	637
12	Lightning7	7	70	73	319
13	Olive Oil	4	30	30	570
14	OSULeaf	6	200	242	427
15	Swedish Leaf	15	500	625	128
16	Synthetic Control	6	300	300	60
17	Trace	4	100	100	275
18	Two Patterns	4	1000	4000	128
19	Wafer	2	1000	6174	152
20	Yoga	2	300	3000	426

Table 3.6: The summary of 1-NN classifier error rate using different methods is presented. For each dataset, the method that has the smallest error rate is highlighted in bold. ED and CR stand for Euclidean distance and compression rate, respectively.

Dataset No.	ED	SAX	SAX-TD	ESAX	Optimal SAX-TD	w	α	CR
1	0.369	0.341	0.338	**0.321**	0.341	64	8	0.50
2	0.389	0.890	**0.273**	0.890	0.286	16	8	0.19
3	0.467	0.567	0.200	0.533	**0.167**	64	9	0.29
4	0.148	0.104	**0.088**	0.138	0.117	4	10	0.07
5	0.250	0.464	**0**	0.179	**0**	8	3	0.06
6	0.120	0.120	0.090	0.090	**0.080**	16	4	0.34
7	0.216	**0.170**	0.181	0.182	0.171	32	4	0.19
8	0.286	0.330	**0.215**	0.333	0.225	16	8	0.26
9	0.217	0.474	**0.154**	0.469	**0.154**	32	3	0.14
10	0.087	0.180	**0.073**	0.193	0.087	4	3	0.06
11	0.246	0.213	0.229	**0.180**	0.197	32	7	0.11
12	0.425	0.397	**0.329**	0.370	**0.329**	32	10	0.21
13	0.133	0.833	**0.067**	0.833	0.133	4	3	0.01
14	0.483	0.467	0.446	0.446	**0.434**	32	7	0.16
15	0.211	0.483	0.213	0.400	**0.210**	16	7	0.26
16	0.120	**0.020**	0.077	0.157	0.117	8	10	0.28
17	0.240	0.460	0.210	**0.160**	0.240	32	3	0.24
18	0.093	0.081	**0.071**	0.129	0.077	16	10	0.27
19	0.005	0.003	0.004	**0.003**	0.004	32	6	0.44
20	**0.170**	0.195	0.179	0.200	0.185	4	10	0.02

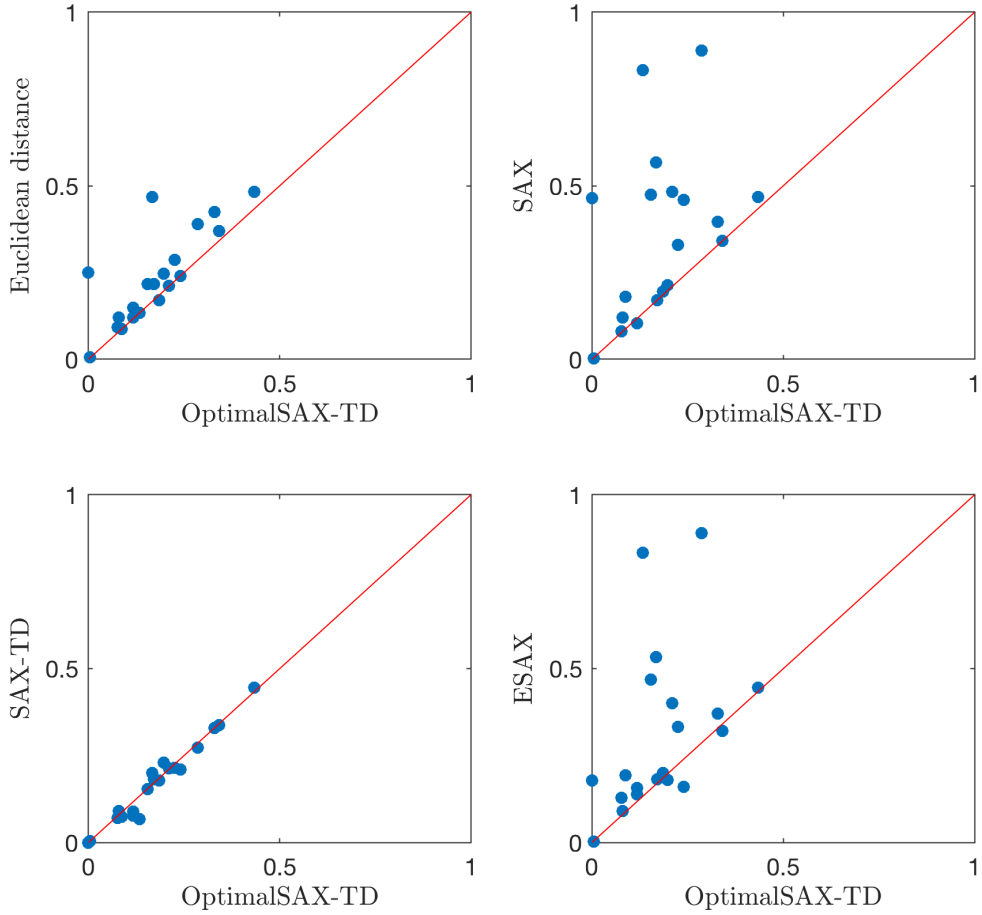

Figure 3.9: The summary of 1-NN classifier error rate using different methods is presented.

3.6 Conclusion

This study attempted to create a bridge between two big topics *data mining* and *operations research* by providing a new similarity measure using SAX representation. The analysis provided in this study created a new line of research and provided a framework for analyzing SAX distance measures. The optimization program presented takes into account an important property of a normalized sequence, that the average is zero. The new distance measure is used to evaluate the classification accuracy on the UCR Archive datasets [10].

Chapter 4

Anomaly Detection

4.1 Introduction

Nowadays, sensors are capable of collecting an enormous amount of data in real time in many domains such as telecommunications, network traffic, power usage, financial monitoring, valve pressure monitoring, electrocardiogram (ECG) monitoring, etc. [13, 52]. Quite a few algorithms have been designed to detect anomalous/unusual behavior in the data. Some of them are online-based methods that provide analysis while the sensor is continuously collecting data [4, 13]. The rest of the methods are considered to be offline that only work with the stored data (on disk) [52]. Once an anomalous behavior is detected, domain experts can identify the event(s) associated with them as a post-processing action. The reason for such unusual behaviors varies based on the source of the underlying process. For instance, some might represent a machine breakdown in a factory, some might represent a sale on Black Friday, and some might indicate a security attack to the system [3]. The purpose of this chapter is to study the anomaly detection problem. In particular, the use of Markov chains is considered as a framework to detect

anomalous behaviors of the underlying system in an online manner.

4.2 Literature review

Anomaly, intrusion or discord detection are general terms that address the problem of finding any unusual patterns, activity or behavior that do not fit the normal and expected behaviors in some data stream. A variety of techniques have been developed to approach this problem in different domains such as server systems [46], network [6, 56], wildfire in Earth science data [11], and electronic systems [14]. Different data mining techniques including symbolic aggregate approximation (SAX) [26, 45], support vector machines [15, 60] and neural networks [39] have been used to study the problem. For example, SAX is used in finding a subsequence with the largest distance to its nearest neighbor as shown in [26] and is combined with grammar induction to find the most structurally different subsequence in [45]. Studying the anomaly detection problem using classic Markov chains and hidden Markov models perspectives is also feasible. They are used in several studies with the idea that not only a single event, but also a series of consecutive events might represent an odd behavior [14, 17, 24, 46, 56, 57, 58]. Clustering methods have been effective in detecting anomalies. The models with clustering techniques have one of the following assumptions regarding the relation between anomaly object and normal clusters: (a) anomalies do not belong to any normal cluster, (b) the distance of anomalies from all normal cluster centroids is large, and (c) the clusters with small number of objects represent anomalies [9].

The contribution of this work is to use a clustering technique to segment the test data based on the normal clusters and use Markov chains to determine how likely it is to observe consecutive transitions between clusters associated with consecutive test

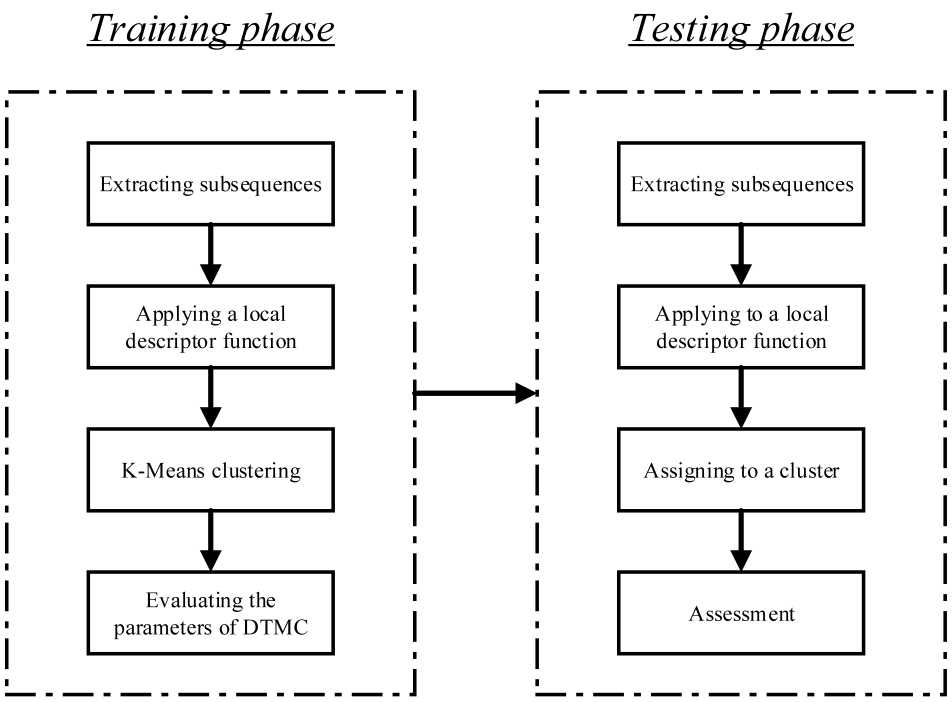

Figure 4.1: The anomaly detection procedure is performed in two phases.

subsequences.

4.3 Methodology

In this study, it is assumed that the *normal* behavior of the system is known. Specifically, a time series X_{Good} is given that represents a normal system. The anomaly detection procedure is summarized in figure 4.1. This process is performed in two general steps. The first step, the training phase, is for parameter evaluations using the normal time series X_{Good}. The second step is to examine the test time series X. Parameters that are learned in the first step are used in the testing phase. The following sections provide the details of the process.

4.3.1 Local descriptor functions

Given a time-ordered sequence $X = x_1, ..., x_n$, the problem is to find areas in X whose *behaviors* do not fit the behavior of the rest of the time series. It is crucial to use a function that can fully describe the features of a subsequence, as this step plays a significant role in modeling the anomaly detection system. The following *local descriptor functions* are used to explain the fluctuations of $X_j = x_j, ..., x_{j+w-1}$, a subsequence of X with length w that starts at time j:

1. *Normalized Subsequence*

   Normalizing the subsequences allows consideration of the shape but not the shifts or scales in the actual values. The descriptor function for the *normalized subsequence* is $\mathcal{D}_{NS}\colon R^w \to R^w$ and is computed by $\mathcal{D}_{NS}(X_j) = \frac{X_j - \mu_j}{\sigma_j}$, where:

   $$\mu_j = \frac{\sum_{i=j}^{j+w-1} x_i}{w} \tag{4.1}$$

   $$\sigma_j = \sqrt{\frac{\sum_{i=j}^{j+w-1} (x_i - \mu_j)^2}{w-1}} \tag{4.2}$$

2. *Local Derivatives*

   Motivated by [28], the descriptor function for this case is $\mathcal{D}_{LD}\colon R^w \to R^w$, where it performs the estimated local derivatives on the elements of its input. The function $\mathcal{D}_{LD}$ is:

   $$\mathcal{D}_{LD}(X_j) = \hat{x}_j, ..., \hat{x}_{j+w-1} \tag{4.3}$$

   $$\hat{x}_i = \frac{x_i - x_{i-1} + \frac{x_{i+1} - x_{i-1}}{2}}{2} \quad \text{for } i = j+1, ..., j+w-2 \tag{4.4}$$

   $$\hat{x}_j = \hat{x}_{j+1} \quad \text{and} \quad \hat{x}_{j+w-1} = \hat{x}_{j+w-2} \tag{4.5}$$

The selected local descriptor function needs to be evaluated for the subsequences of both X_{Good} and X.

4.3.2 Clustering

The clustering step belongs to the training phase. It uses the normal time series X_{Good}. Each subsequence shows the behavior of a system at a certain time. Clustering of subsequences helps to understand what the state of the system at a certain time is and to move the modeling of the underlying system towards a discrete-time Markov chain. *k-means* clustering is a powerful unsupervised learning technique that partitions the dataset into k distinct groups in which elements of each cluster are similar to each other. Figure 4.2 shows an example of the *k-means* clustering technique on 300 randomly generated 2-dimensional data that are split into three clusters. As another example, see figure 4.3, a time series of length 100. Subsequences of length 20 are extracted, applied to a *normalized sequence* descriptor function and based on their similarity are grouped into four clusters. Figure 4.3 also shows the centroids of each cluster. The new representation for the original time series is the vector $2, 2, 1, 1, 3, 3, 4, 4, 2, 2, ...$, where each value represents the cluster number of the associated subsequence. This example showed the application of *k-means* clustering technique on high-dimensional data.

In this study, all subsequences of length w are first applied to a local descriptive function (either *normalized subsequence* or *local derivatives*) and then partitioned into k clusters. A cluster representation for time series X_{Good} is C_{Good}. The k cluster centroids are used in the testing phase.

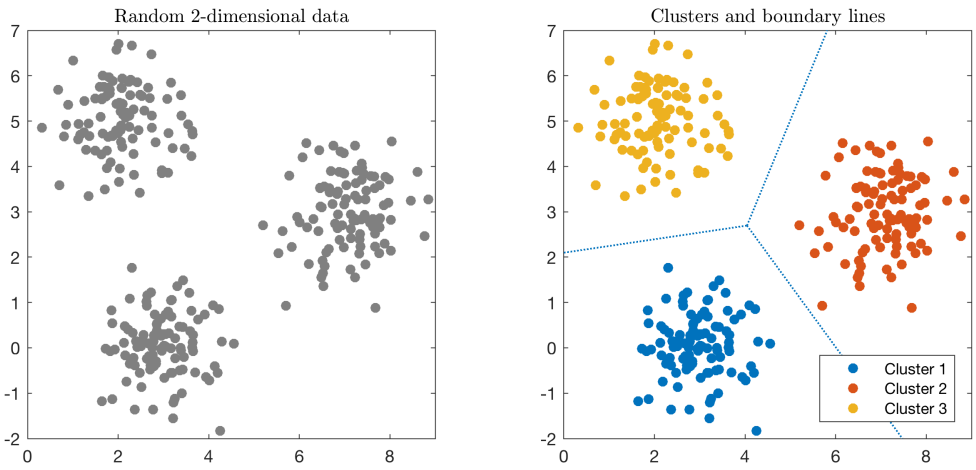

Figure 4.2: 300 two-dimensional data points are grouped into three clusters using $k-means$ technique.

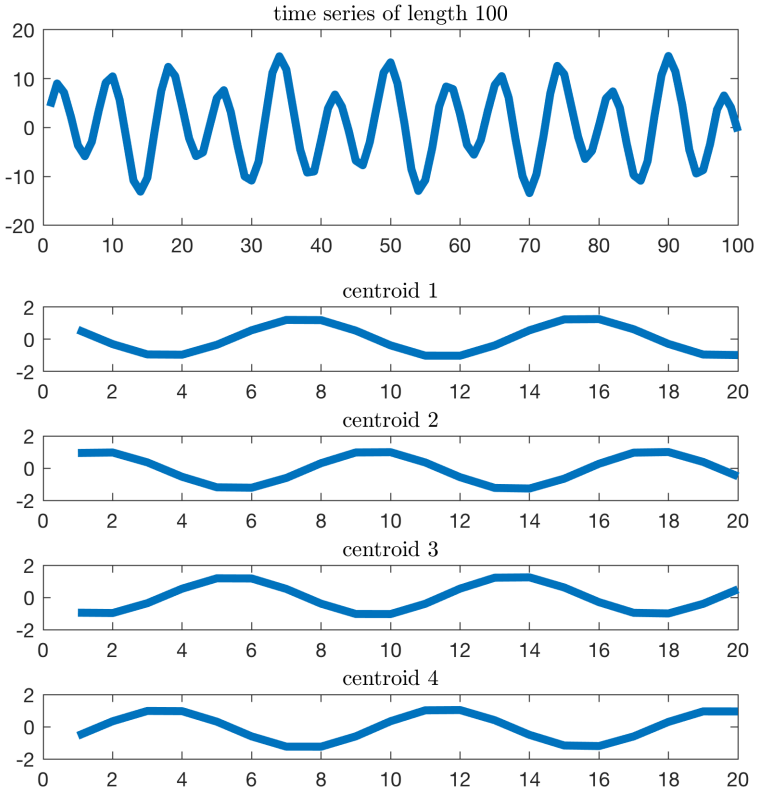

Figure 4.3: In this figure, a time series of length 100 and 4 cluster centroids associated with subsequences of length 20 are plotted.

Table 4.1: The symbols used in the Markov chain modeling is defined.

Symbol	Definition
$\mathcal{S}$	The set of Markov chain states
Y_t	The random variable that represents the state of the system at time step t
N_i	The number of times that the Markov chain visited state i $(i = 1, ..., k)$
N_{ij}	The number of times that the Markov chain made a transition from state i to state j $(i, j = 1, ..., k)$
N	The total number of time steps
p_{ij}	The one-step transition probability from state i to state j $(i, j = 1, ..., k)$
π_i	The long-run probability of the Markov chain for state i $(i = 1, ..., k)$

4.3.3 Discrete-time Markov chain

Background

This section provides a brief background on Markov chains. The symbols that are used in this section are summarized in table 4.1. A sequence of discrete-time random variables Y_t $(t \geq 1)$ defines the behavior of a system throughout time, where the value of Y_t is called a state. States are elements of a discrete set $\mathcal{S}$. The stochastic process created by the random variable Y_t is said to be a Markov chain, if it satisfies the following condition for all states in $\mathcal{S}$ and all $t \geq 1$:

$$\mathbb{P}(Y_{t+1} = j | Y_t = i) = \mathbb{P}(Y_{t+1} = j | Y_t = i, Y_{t-1} = i_{t-1}, ..., Y_1 = i_1) = p_{ij} \tag{4.6}$$

where p_{ij} is called a one-step transition probability from state i to state j. For each state $i \in \mathcal{S}$, the transition probability has the property of $\sum_{j \in \mathcal{S}} p_{ij} = 1$. The estimate of p_{ij} is

computed as:

$$p_{ij} = \frac{N_{ij}}{N_i} \tag{4.7}$$

where N_{ij} and N_i are the number of transitions from state i to state j and the number of times that the Markov chain is in state i, respectively.

Long-run probabilities of the Markov chain is the long-run average of times that Y_t is in a specific state. Let N be the total number of time steps that the random variable Y_t is recorded. π_i is computed as:

$$\pi_i = \lim_{N\to\infty} \frac{1}{N} \sum_{t=1}^{N} \mathcal{A}(Y_t = i) \tag{4.8}$$

where $\mathcal{A}(\gamma)$ takes 1 if the statement γ is true and takes zero otherwise. In this study, the short-time estimate of π_i is used, as N is finite. That is:

$$\pi_i = \frac{N_i}{N} \tag{4.9}$$

The last piece of Markov chain modeling is the probability of observing a sequence of transitions. The likelihood of observing a path $S = s_1, ..., s_m$ that the Markov chain visits is computed as follows:

$$\mathbb{P}(S = s_1, ..., s_m) = \pi_{s_1} \prod_{i=1}^{m-1} p_{s_i, s_{i+1}} \tag{4.10}$$

Markov chain model for clusters

The next step of the anomaly detection process is to create a Markov chain model for C_{Good}, the cluster representation of a normal time series X_{Good}. The parameters of the Markov chain are evaluated as follows. The size of state space $\mathcal{S}$ is equal to the number

of clusters, the one-step transition probabilities are computed based on equation 4.7 and the long-run probabilities are computed by equation 4.9.

4.3.4 Testing phase

In the testing phase, subsequences of length w are extracted from the time series X. The same descriptor function that is used in the training phase should be applied on the subsequences of X. Then, each subsequence is assigned to one of the clusters $1, ..., k$, based on its similarity to the cluster centroids. That is, the Euclidean distance between each subsequence and centroids are computed and the subsequence is assigned to a cluster with the minimum distance. A cluster representation of X is called C.

Assessment

The assessment step answers this question: *"Is the subsequence $X_j = x_j, ..., x_{j+w-1}$ an anomaly?"* This step uses equation 4.10 to evaluate how likely it is to observe the s-step transitions $C_j, ..., C_{j+s}$. If this probability is less than the predetermined threshold h, then the sequence X_j is an anomaly. For numerical considerations, the logarithm of the likelihood is used in this study. As some of the transition probabilities might be zero, a relatively small number (i.e., 10^{-4}) is replaced with them so the log function is well-defined for those transitions.

4.4 Empirical evaluation

Numerical experiments are performed on two datasets from different domains. The first dataset is a medical data, QTDB 0606 ECG, from PhysioNet [19, 29]. The second dataset contains the utility usage data that was recorded at SAS Institute Inc., Cary, NC. The

sensors recorded the data at the campus every 5 minutes from June 1, 2017 to September 20, 2017. For the privacy of this dataset, data is normalized before analysis. The weekend data is eliminated from the computational experiment. Moreover, the average value of every 6 data points is taken to reduce the dimensionality and noise in the dataset. The window length and path size are set to be $\{30, 45\}$ and 6. The number of clusters ranges from 5 to 8. In order to determine the cut-off point for detecting the anomaly areas, the assessment step is performed on the training data. The threshold is considered to be the minimum score multiplied by 1.1.

Figures 4.4, 4.5 and 4.6 show the path Log-likelihood and anomaly detected areas. In figures 4.4 and 4.5, the *normalized subsequence* descriptor function is used for the computations. The third heartbeat in figure 4.4 represent the true anomaly as shown in [45]. Anomaly areas in figures 4.5 and 4.6 are due to the national holidays July 4 (independence day) and September 4 (labor day) and some regular pre-scheduled maintenance on campus buildings. In figure 4.6, the *local derivatives* descriptor function is used.

As mentioned in the literature review section, some clustering techniques identify an anomaly based on their distances from cluster centroids. That assumption is not suitable for this study. Figure 4.7 shows the areas whose distances from their cluster centroid is larger than those of the normal subsequences. This approach produces many *false positives*. The evaluation measure of the Markov model presented in the current study relies on the consecutive transitions between clusters and not the performance of the clustering step.

This method is also very useful for streaming data. The new data points that come to the system create new subsequences. The new subsequences need to be only compared to the cluster centroids. Once the clusters are identified, the assessment step is straightforward.

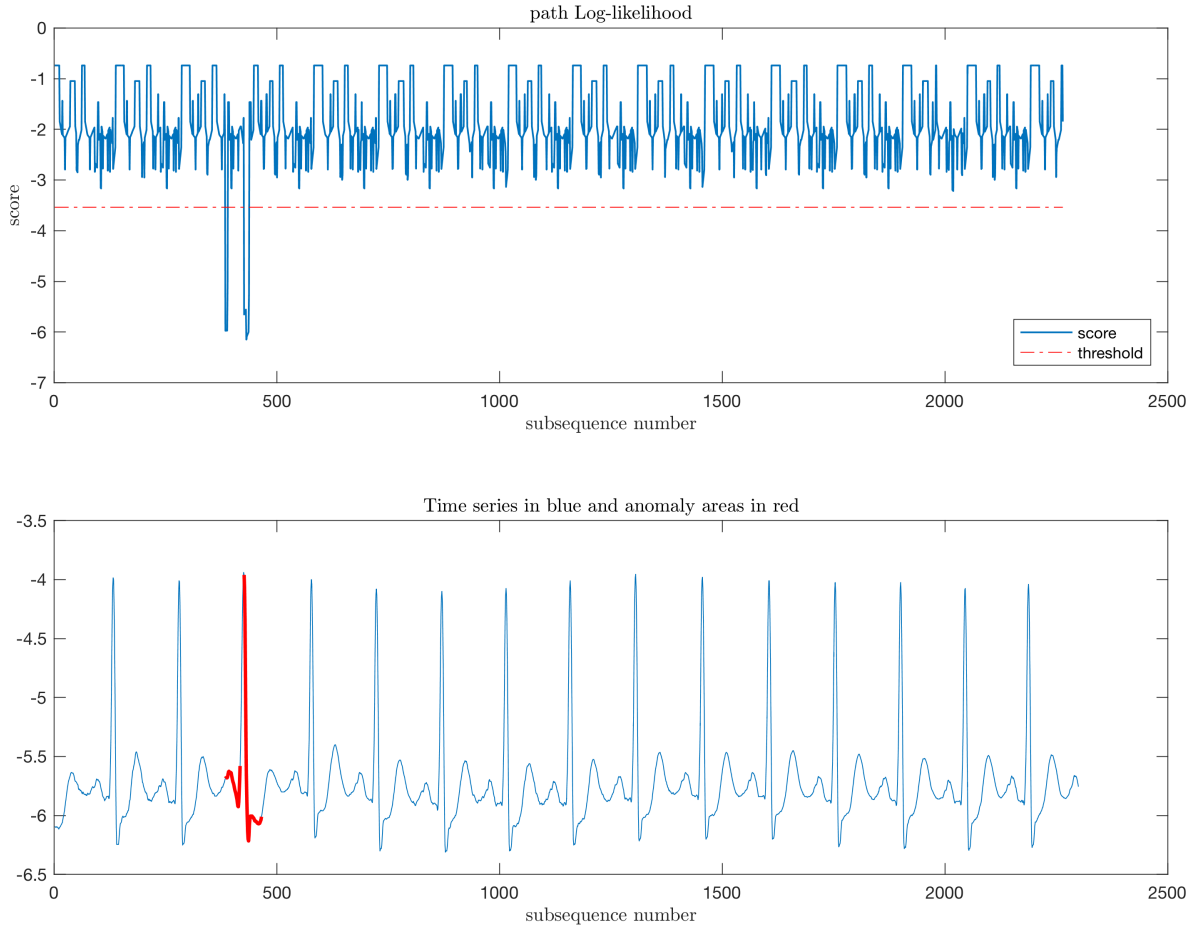

Figure 4.4: The first plot shows the path Log-likelihood and the second one shows anomaly detected areas using *normalized subsequence* descriptor function. In this example, the window length, number of clusters and score threshold are set to 30, 7 and -3.5360.

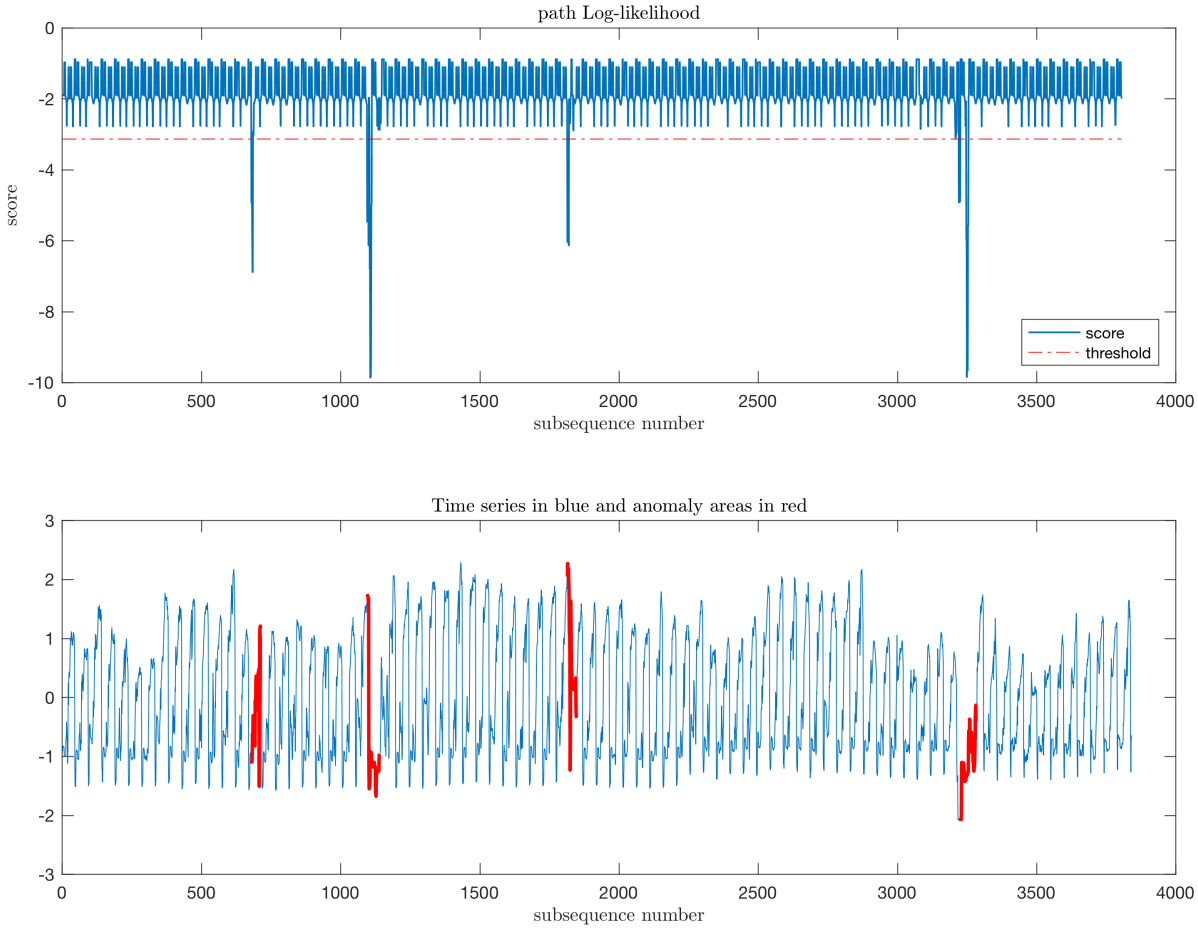

Figure 4.5: The first plot shows the path Log-likelihood and the second one shows anomaly detected areas using *normalized subsequence* descriptor function. In this example, the window length, number of clusters and score threshold are set to 30, 6 and -3.1298.

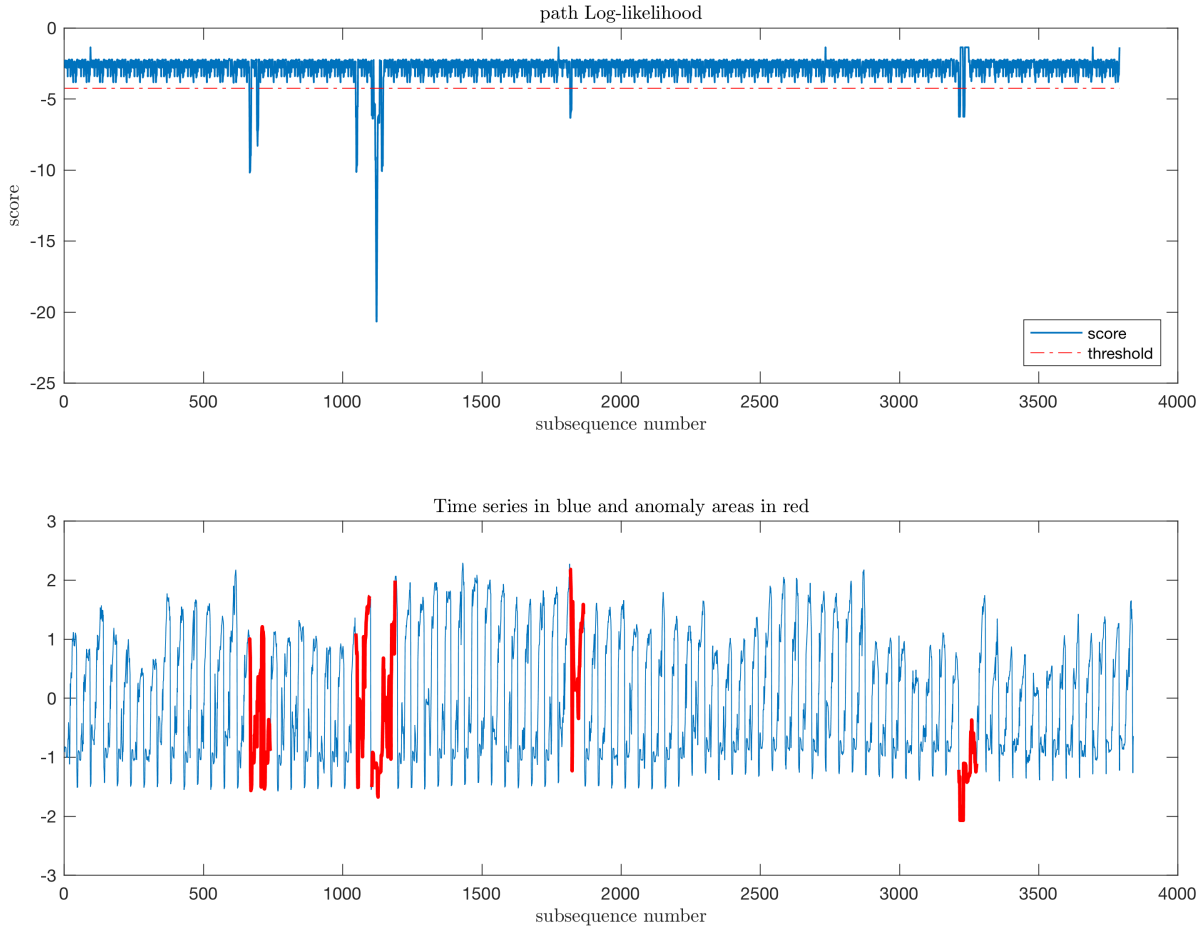

Figure 4.6: The first plot shows the path Log-likelihood and the second one shows anomaly detected areas using *local derivatives* descriptor function. In this example, the window length, number of clusters and score threshold are set to 45, 8 and -4.2328.

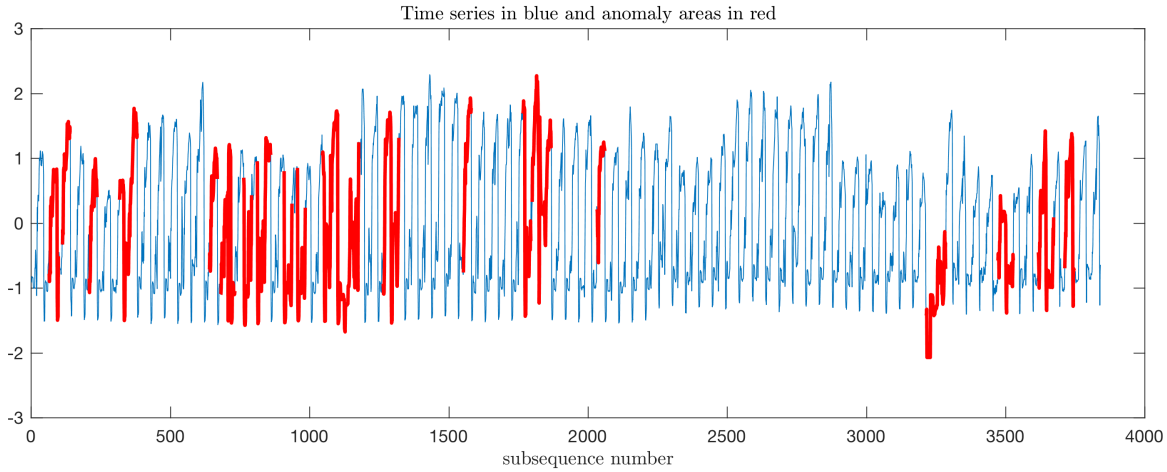

Figure 4.7: This figure shows that if the criteria for detecting anomalies was only the distance of subsequences from the centroid of their cluster, then many false positives were produced.

4.5 Conclusion

This study showed how to merge two distinct tools, clustering techniques, and Markov chains, for detecting anomalies in a univariate time series. The proposed method was successfully applied on two different datasets, one in the medical domain and one utility usage data. Several extensions can be made on this topic. They include:

1. the study of anomalies in multivariate time series
2. the use of hidden Markov models
3. to use the variety of shape descriptor functions
4. to study the assessment step by taking into account the quality of clustering step and distances of each subsequence from cluster centroids as a feature.

Chapter 5

New Features with Shape Dynamic Time Warping

Note: This chapter is inspired by the recent article [62] and intends to present, analyze and discuss some unpublished results of that article.

5.1 Introduction

It has been more than a decade since the introduction of a well-known time series distance measure, dynamic time warping (DTW). The objective of DTW is to quantify the similarity of two time-ordered sequences of real-valued numbers (not necessarily of the same size) by finding the optimal alignment between the two. The alignment need not be linear, which makes this distance measure very powerful in detecting time distortions. This technique has quite a few applications in various data mining tasks such as classification [23, 25, 41, 61], clustering [40, 42] and indexing [27].

Specifically, given two time-ordered sequences $X = x_1, ..., x_m$ and $Y = y_1, ..., y_n$, DTW

computes the minimum warping cost function:

$$Z = \sum_{k=1}^{K} dist(w_k) \tag{5.1}$$

where w_k is the k^{th} element of the warping path that represents the k^{th} alignment (x_i, y_j). DTW has three criteria for creating the path $W = w_1, ..., w_K$, they are:

1. $w_1 = (1, 1)$, that is the boundary condition for the first element of the warping path.

2. If $w_k = (i, j)$, then w_{k+1} can take either $(i, j+1)$, $(i+1, j)$ or $(i+1, j+1)$, when i and j are in the appropriate range. This condition ensures that the path is continuous and a monotonically increasing function.

3. The last element of the warping path is (m, n).

Dynamic Programming finds the optimal warping path in a time complexity of $\mathcal{O}(mn)$ (given that the length of two time series are m and n). If $f(u, v)$ is the DTW of subsequences $X^u = x_1, ..., x_u$ and $Y^v = y_1, ..., y_v$, then the following is the backward induction relation to find the DTW of two sequences X and Y:

$$f(u, v) = dist(x_u, y_v) + \min\{f(u-1, v), f(u, v-1), f(u-1, v-1)\} \tag{5.2}$$

with the following initializations:

$$f(1, 1) = dist(x_1, y_1) \tag{5.3}$$

$$f(1, v) = \sum_{j=1}^{v} dist(x_1, y_j) \tag{5.4}$$

$$f(u,1) = \sum_{i=1}^{u} dist(x_i, y_1) \tag{5.5}$$

Figure 5.1 displays an example of the optimal alignment. In plot 5.1(a), two time series from the Beef dataset in the UCR archive are presented [10]. In plot 5.1(b), the optimal warp scaled plot based on DTW is shown. For this specific example, one can also show the Euclidean alignment, plot 5.1(c), since the lengths of two sequences are equal. Euclidean alignment ensures that $w_i = (i, i)$ for $i = 1, ..., m$, meaning that the i^{th} element of the first sequence is aligned with the i^{th} element of the second sequence.

As part of computing the DTW distance of two sequences X and Y, it is essential to define the distance function $dist(x_i, y_j)$ between every elements x_i $(i = 1, ..., m)$ and y_j $(j = 1, ..., n)$ of the two series. The most common used distance function $dist(x, y)$ are the squared difference $(x - y)^2$ and absolute difference $|x - y|$.

Two common drawbacks of DTW are as follows: (a) it does not prevent *singularities* by itself and (b) it might fail to align similar local structures. Singularity is the state of mapping a singledata point in one sequence to a large number of consecutive data points in the other. Figure 5.1(b) shows an example of these failures. In that figure, several singularities as well as subsequences mismatches are observed. It is difficult to relate the two concepts mathematically; however, failure in the alignment of similar local patterns is likely to occur in the presence of a singularity. Needless to mention that although the above-mentioned issues frequently happen, DTW finds the optimum alignment between the two sequences. Numerous efforts have been made to prevent singularity, whereas only a few studies have shown how to adjust DTW formulation to consider the local behavior of sequences [28, 62]. The purpose of this study is to extend two powerful variants of DTW, derivative DTW (DDTW) [28] and shapeDTW [62]. DDTW uses the local derivatives of the data instead of their actual values. To some extent, it prevents the unnecessary

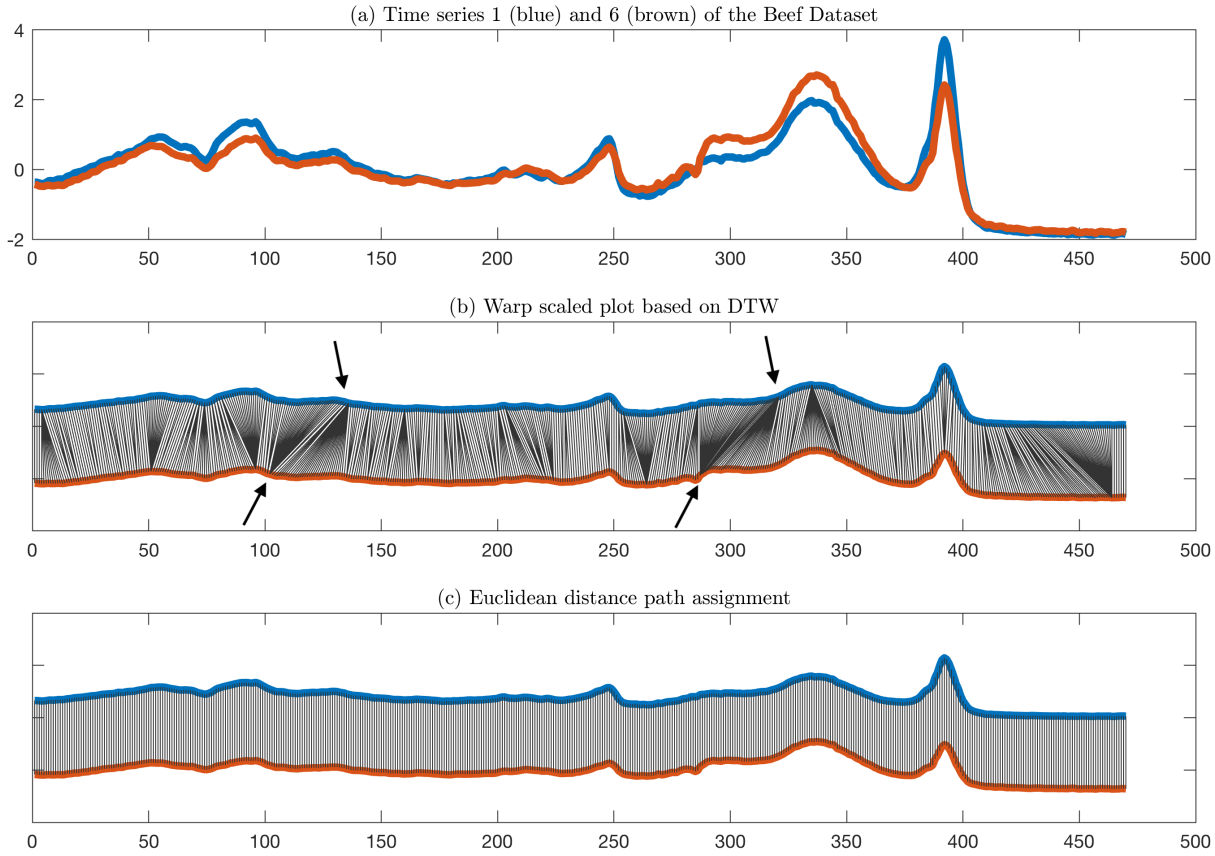

Figure 5.1: Plot (a): Two example time series taken from the Beef Dataset in UCR archive. Plot (b): It is the scaled time warp plot which shows *singularity* in several locations. Plot (c): This plot shows the basic Euclidean distance path alignment, where $w_i = (i, i)$ for $i = 1, ..., m$.

warping in the *time*-axis in the presence of variability in the Y-axis values. Similar to DDTW, shapeDTW does not use the actual values either. It is a methodology that uses local structure descriptor functions to explain the fluctuations in the neighborhood of each data point. ShapeDTW has achieved an excellent performance on classification task [62].

This study uses two local structure descriptor functions: the local derivatives and the normalized form of subsequences. Figure 5.2 shows the rationale behind choosing these functions. Plot 5.2(a) displays two time series that are parts of larger time series. Figure 5.2(b) shows a few singularities in the classic DTW. As obvious in plot 5.2(c), DDTW fails to find a logical alignment. In some cases, it actually attempts to warp the *time*-axis to adjust variability in the Y-axis. The normalized form of subsequences, due to its Y-axis invariant property, ignore shifts and scales. In plots 5.2(d) and (e), shapeDTW finds the proper alignments by considering local derivatives and normalized form of sequences (neighborhood of length $w = 25$). In this example, the values of two time series satisfy the relation $y_{brown} = 2.98y_{blue} + 0.06$, that makes the alignment in figure 5.2(e) the best.

The rest of this study is organized as follows. In section 5.2, a short review of different studies that addressed DTW drawbacks is presented. In section 5.3, the adjustment of DTW based on the normalized form of subsequences and local derivatives as local structure descriptors is discussed followed by section 5.4 that provides numerical experiments on the *UCR time series classification archive* [10].

5.2 Literature Review

Different modifications of DTW mostly address the singularity problem, a few of which are presented in this section. Sakoe-Chiba band [44] and Itakura parallelogram [22] are

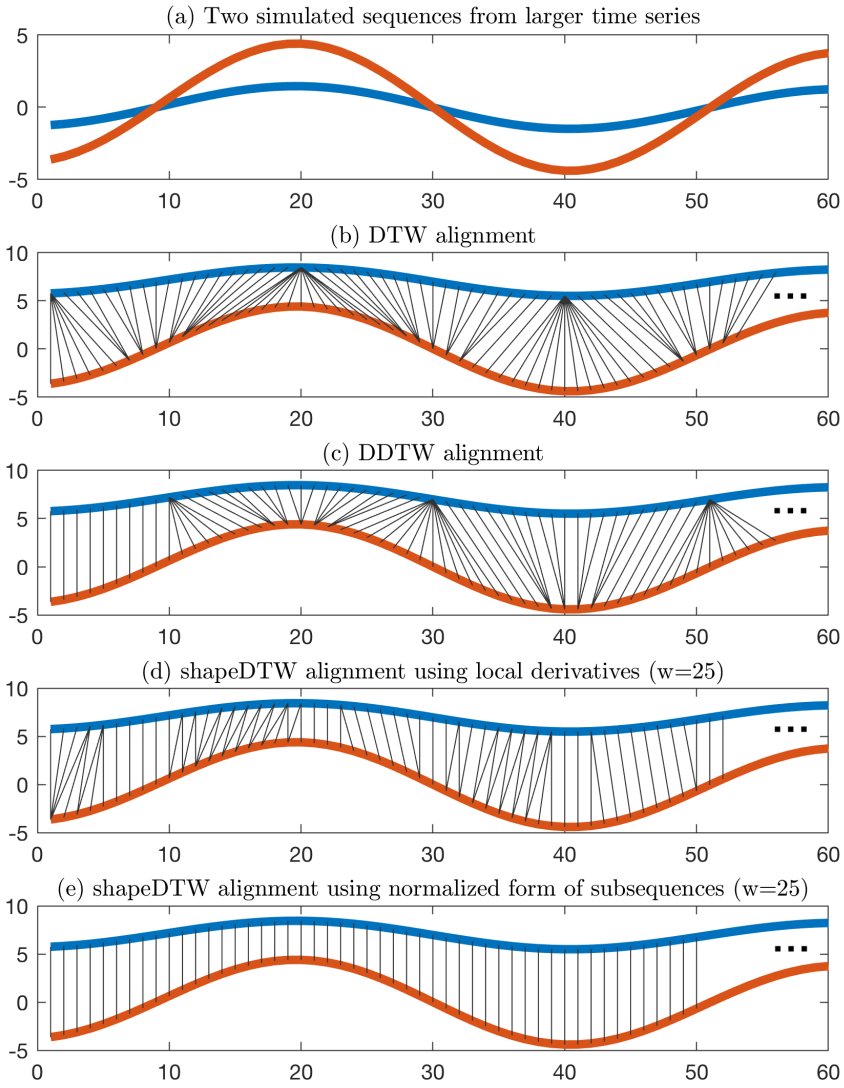

Figure 5.2: Plot (a): Two subsequences of larger time series. Plot (b): Shapes of the sequences are similar to each other; however, DDTW justifies the variability in Y-axis towards time warping that produces several singularities.

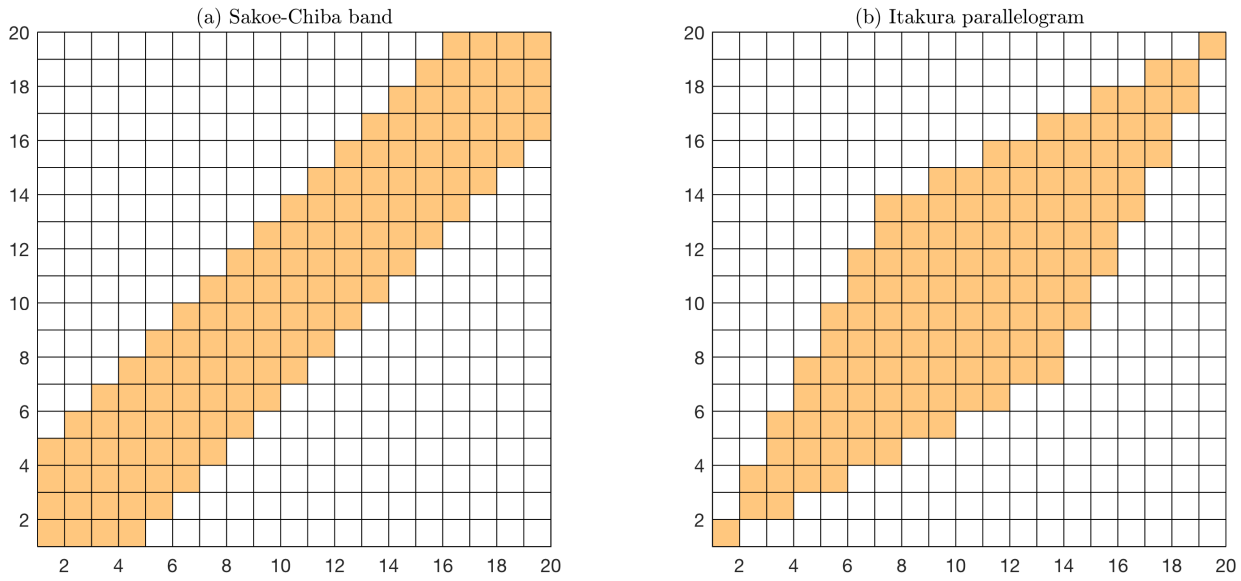

Figure 5.3: Examples of (a) Sakoe-Chiba and (b) Itakura parallelogram constraints for warping path. The path cannot contain any white squares.

two classic region constraints that were designed to restrict the phase difference in the warping path; i.e., x_i cannot be aligned with y_j unless $|i - j|$ satisfies some constraints. Figure 5.3 shows the examples of limited warping path constraints by the above techniques. In this figure, white squares are illegal alignments. One mathematical interpretation of these models is to set $dist(x_i, y_j) = \infty$ for the pairs associated with white squares. Sakoe-Chiba band and Itakura parallelogram are the oldest solutions for the singularity problem. These two methods impose local constraints on the phase difference alignments and the warping path.

Recently, the effect of restricting the total length of the warping path has been studied [61]. Limiting the total length of the warping path is considered to be a global constraint on the problem. The advantage of the global-constraint model compared to local-constraints models is that it 'potentially' allows the optimization problem to find the correct alignment between the two sequences, that could be easily missed by im-

posing local constraints. Similar to the regular DTW, the technique used to solve DTW under this policy is dynamic programming, which has a time complexity of $\mathcal{O}(L_{UB}mn)$ (given that the length of two time series are m and n and the limit on the total length of the warping path is L_{UB}). This similarity measure has been experimentally shown to be more effective than both no-constrained DTW and locally-constrained DTW models on the time series classification task.

As mentioned before, the second weakness of DTW is that it does not consider local structures of a sequence when it performs the alignment. A few studies attempt to address this problem [28, 62]. Derivative dynamic time warping (DDTW) is inherently a regular DTW with a minor modification that uses the (estimated) local derivatives of the sequences as the input data. The data preprocessing is performed in the following manner. If the input time series is $Q = q_1, ..., q_n$, then the modified time series is $\hat{Q} = \hat{q}_1, ..., \hat{q}_n$, where:

$$\hat{q}_i = \frac{q_i - q_{i-1} + \frac{q_{i+1} - q_{i-1}}{2}}{2} \quad \text{for } i = 2, ..., n-1 \tag{5.6}$$

$$\hat{q}_1 = \hat{q}_2 \quad \text{and} \quad \hat{q}_n = \hat{q}_{n-1} \tag{5.7}$$

Figure 5.4 illustrates a comparison between DDTW (plot Figure 5.4(a)) and Sakoe-Chiba band (Figure 5.4(b)) alignment for the same input as in figure 5.1. It is observed that Sakoe-Chiba constrained DTW produces quite a few singularities, while DDTW finds a logical alignment between two sequences.

Another solution to consider local structures of a sequence is shapeDTW, a variant of DTW [62]. The idea is to see the input sequence $X = x_1, ..., x_m$ as vectors of features, the j^{th} one of which represented by $\mathcal{F}_X(j, :)$ describes the point x_j and its neighborhood points. Definition of the features (or local structure descriptor) relies on the context of the

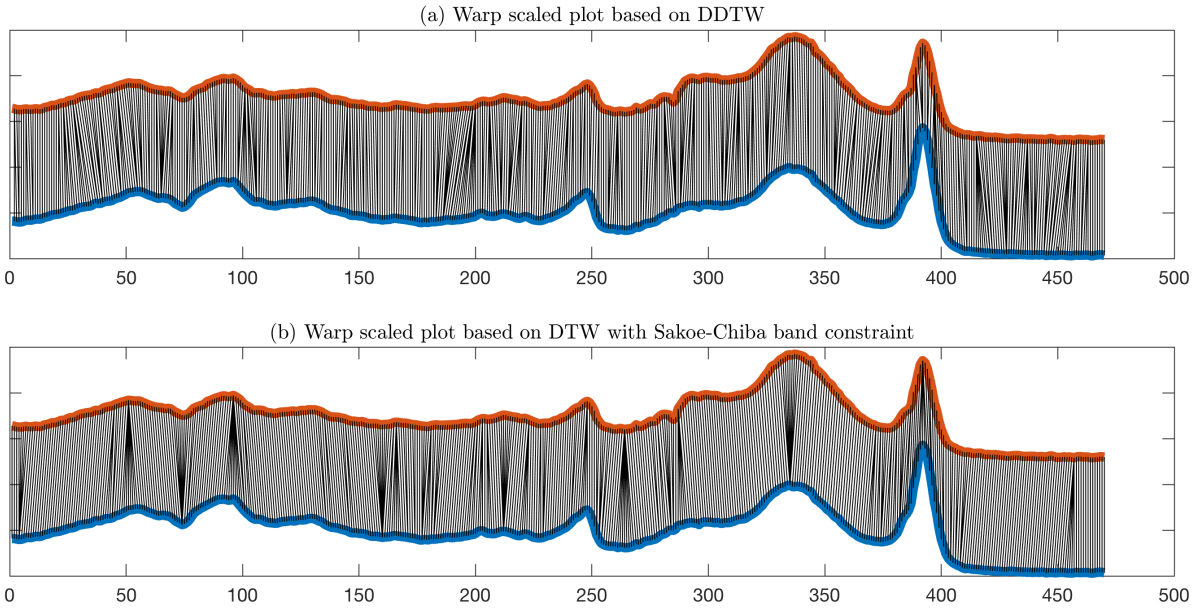

Figure 5.4: Plot (a): The scaled time warp plot based on DDTW. Plot (b): The scaled time warp plot based on Sakoe-Chiba constrained DTW. The alignment created by DDTW is more intuitive compared to that of Sakoe-Chiba constrained DTW.

problem, yet several options were introduced in [62], namely *raw-subsequence*, *piecewise aggregate approximation*, *discrete wavelet transform*, *slope*, *derivative* and *histogram of oriented gradients*. The rest of shapeDTW algorithm is similar to the regular DTW, where $dist(\mathcal{F}_X(.,:), \mathcal{F}_Y(.,:))$ (the first or the second norm of the vector $\mathcal{F}_X(.,:) - \mathcal{F}_Y(.,:)$) is used instead of $dist(.,.)$.

In section 5.3, two local structure descriptors based on (estimated) local derivatives (from DDTW) and normalized form of *raw-subsequence* descriptor are introduced. The two new feature definitions are then applied to the shapeDTW algorithm.

5.3 Methodology

A thorough introduction of shapeDTW is presented in [62]. A brief summary of it is also presented here, as it is crucial to understand the steps taken in the algorithm. For each sequence $X = x_1, ..., x_m$, a feature matrix $\mathcal{F}_X$ is computed based on a function that summarizes the neighborhood (of length w) of each point x_j (where $j = 1, ..., m$). In order to create the row j of the matrix $\mathcal{F}_X$, the j^{th} subsequence of X, $X_j = x_{j-[\frac{w}{2}]}, ..., x_j, ..., x_{j+[\frac{w-1}{2}]}$, is passed through a function $\mathcal{D}\colon R^w \rightarrow R^s$. The output of $\mathcal{D}(X_j)$ is the j^{th} row of $\mathcal{F}_X$. Therefore, the matrix $\mathcal{F}_X$ has m rows and s columns. For example, in the case of the *raw-subsequence* descriptor, $\mathcal{D}(X_j) = X_j$ and $s = w$. Note that the first and last element of X are replicated enough so that X_j is well defined over all values of j. ShapeDTW uses the feature matrices $\mathcal{F}_X$ and $\mathcal{F}_Y$ to find the alignment between the sequences X and Y. Specifically, it uses the vectorized form of equation 5.2 as follows:

$$f(u, v) = dist(\mathcal{F}_X(u, :), \mathcal{F}_Y(v, :)) + \min\{f(u-1, v), f(u, v-1), f(u-1, v-1)\} \quad (5.8)$$

where $dist(\mathcal{F}_X(u, :), \mathcal{F}_Y(v, :))$ is the similarity value for the neighborhood of points u and v in two sequences X and Y, respectively. In this study, the $\mathcal{L}2$-norm is used.

The two local structure descriptors introduced in this study are as follows:

1. *Normalized Subsequence*

   In the case of *raw-subsequence* descriptor, the DTW alignment may suffer from Y-axis shifts in pairwise subsequence matching. Normalizing the subsequences allows to consider the shape and not the shifts or scales in the actual values. The descriptor function for *normalized subsequence* is $\mathcal{D}_{NS}\colon R^w \rightarrow R^w$ and is computed

by $\mathcal{D}_{NS}(X_j) = \frac{X_j - \mu_j}{\sigma_j}$, where:

$$\mu_j = \frac{\sum_{i=j-[\frac{w}{2}]}^{j+[\frac{w-1}{2}]} x_i}{w} \tag{5.9}$$

$$\sigma_j = \sqrt{\frac{\sum_{i=j-[\frac{w}{2}]}^{j+[\frac{w-1}{2}]} (x_i - \mu_j)^2}{w-1}} \tag{5.10}$$

Figure 5.5 provides an example of creating a *normalized subsequence* descriptor for a sequence of length 6. The neighborhood w has the length of 5. For this example, the embedded sequence (the one whose first and last elements are duplicated), 6 subsequences and their associated normalized form are plotted.

2. *Local Derivatives*

Motivated by [28], the descriptor function for this case is $\mathcal{D}_{LD}\colon R^w \to R^w$, where it performs the estimated local derivatives on the elements of its input. Specifically, given a sequence $Q = q_1, ..., q_w$, the function $\mathcal{D}_{LD}$ is:

$$\mathcal{D}_{LD}(Q) = \hat{q}_1, ..., \hat{q}_w \tag{5.11}$$

$$\hat{q}_i = \frac{q_i - q_{i-1} + \frac{q_{i+1} - q_{i-1}}{2}}{2} \quad \text{for } i = 2, ..., w-1 \tag{5.12}$$

$$\hat{q}_1 = \hat{q}_2 \quad \text{and} \quad \hat{q}_w = \hat{q}_{w-1} \tag{5.13}$$

The following is the feature matrix based on the *local derivatives* descriptor function

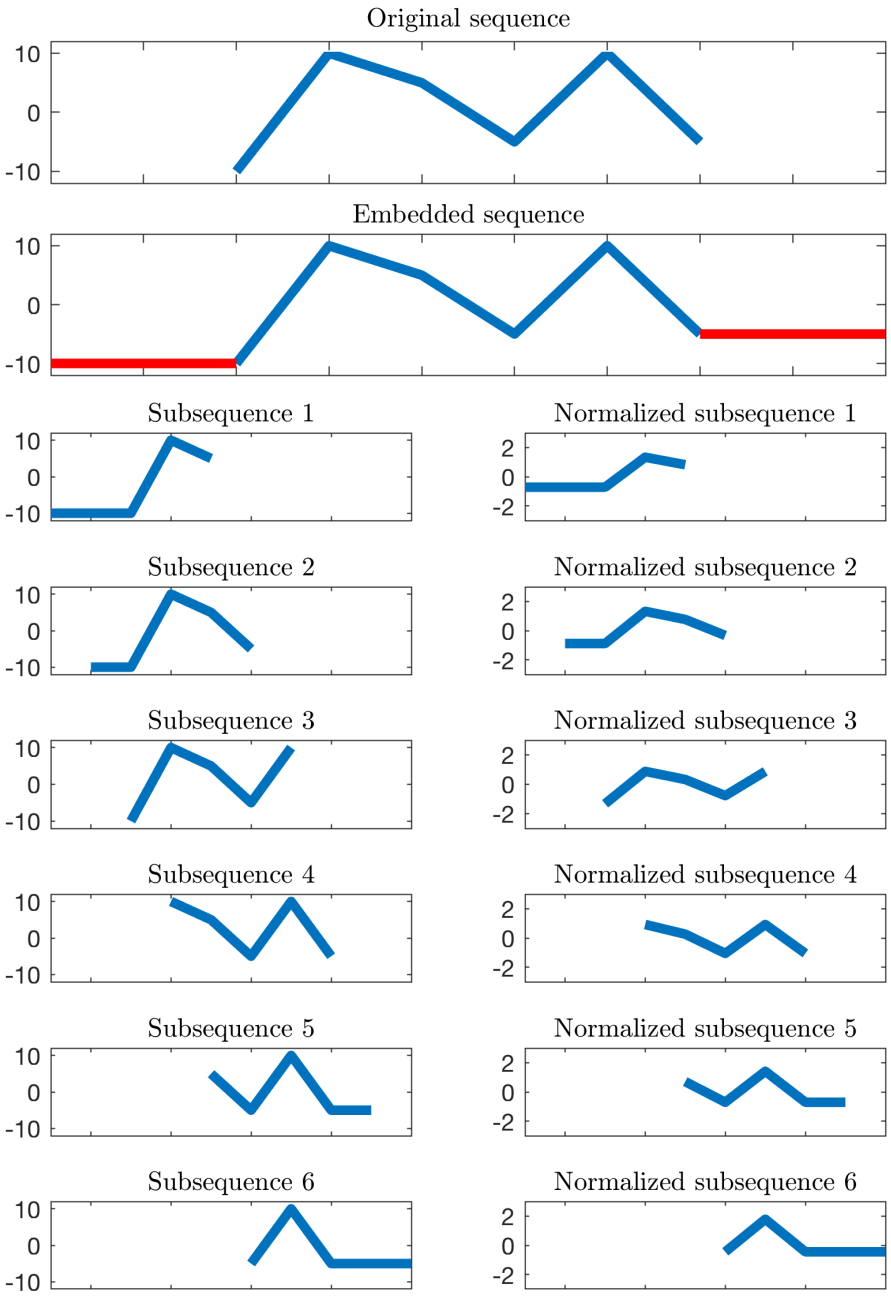

Figure 5.5: The original sequence and the neighborhood w has the length of 6 and 5, respectively. The embedded sequence, 6 subsequences and their associated normalized form are plotted.

for the example in figure 5.5. Each row is associated with a subsequence.

$$\begin{bmatrix} 0 & 0 & 5 & 13.75 & 13.75 \\ 5 & 5 & 13.75 & -6.25 & -6.25 \\ 13.75 & 13.75 & -6.25 & -3.75 & -3.75 \\ -6.25 & -6.25 & -3.75 & 7.5 & 7.5 \\ -3.75 & -3.75 & 7.5 & -11.25 & -11.25 \\ 7.5 & 7.5 & -11.25 & 0 & 0 \end{bmatrix} \tag{5.14}$$

DTW recursive formulation is used to find the alignment between two sequences X and Y, based on their associated feature matrices $\mathcal{F}_X$ and $\mathcal{F}_Y$. Algorithm 9 is used to compute the value of similarity between two sequences X and Y, based on the alignment matrix *path*. *PROC SIMILARITY* of SAS 9.4 supports the classic DTW whose implementation is explained in [2].

Algorithm 9: Computing the similarity of two sequences X and Y given the alignment *path*

Require $X \leftarrow$ *sequence 1,* $Y \leftarrow$ *sequence 2,* *path* $\leftarrow$ *the alignment.*
$dist \leftarrow 0$
for $k \leftarrow 1$ **to** $Length(path)$ **do**
  $dist \leftarrow dist + (x_{path(k,1)} - y_{path(k,2)})^2$
end
return *dist*

5.4 Numerical Experiments

In order to show the quality of the proposed descriptor functions, time series classification is performed on the UCR Archive datasets [10]. One nearest-neighbor (1NN) classifier is used due to its direct capability of showing the performance of a similarity measure. Neighborhood length w is the only parameter to tune for the descriptor functions. In this set of computations, w is set to vary from 5 to 100 with increments of 5. The classification error rates for each dataset is reported as $best[Q_1, Q_3]$, where *best* is the minimum error rate and Q_1 and Q_3 are the error rates first and third quartiles, respectively. Table 5.1 summarizes the results of this experiment. The error rates associated with DTW (with no warping window limits) [10] and shapeDTW with *raw-subsequence* (based on [62]), *normalized-subsequence* and *local derivatives* descriptor functions are presented. Wilcoxon signed-ranks test is used to compare the performance of the three descriptor functions. A *p-value* less than 0.05 is considered to be significant.

5.4.1 ShapeDTW vs. DTW

In general, there is no statistical relationship between classification performance and neighborhood length w. Figure 5.6 shows the error rates in all datasets as a function of w for *normalized subsequence* (in blue) and *local derivatives* (in brown) descriptor functions. The error rates of the DTW classifier are also presented in black dash-dot line. One-sided Wilcoxon signed-rank test indicates that median error rates of shapeDTW classifier by *normalized subsequence* and *local derivatives* are significantly lower than that of DTW in 12/24 and 13/24 datasets (*p-value* less than 0.05). In addition, all error rate values of shapeDTW using *normalized subsequence* are lower than their DTW error rate in *ECGFiveDays*, *Face (all)*, *Ham* and *Wine* datasets. Similarly, shapeDTW using

Table 5.1: 1-NN classification error rates in shapeDTW technique is reported as $best[Q_1, Q_3]$, where *best* is the minimum error rate and Q_1 and Q_3 are the error rates first and third quartiles, respectively. Symbols † and ‡ indicate significant difference in median values of classifier errors when compared with DTW and shapeDTW using *raw-subsequence* descriptor function, respectively.

Dataset	DTW	Raw-Subsequence	Normalized-Subsequence	Local Derivatives
50Words	0.310	0.209 [0.217, 0.237]	0.202 [0.218, 0.277] †	0.204 [0.212, 0.241] †
Adiac	0.396	0.340 [0.349, 0.358]	0.312 [0.352, 0.379] †	0.348 [0.362, 0.385] †
ArrowHead	0.297	0.177 [0.186, 0.223]	0.171 [0.203, 0.266] †	0.166 [0.189, 0.217] †
Beef	0.367	0.333 [0.333, 0.367]	0.233 [0.333, 0.367] †	0.300 [0.333, 0.333] †‡
BeetleFly	0.300	0.150 [0.250, 0.375]	0.050 [0.200, 0.350] ‡	0.050 [0.250, 0.350] ‡
BirdChicken	0.250	0.050 [0.200, 0.250]	0.150 [0.250, 0.300]	0.050 [0.125, 0.200] †‡
CBF	0.003	0.003 [0.014, 0.034]	0.031 [0.042, 0.068]	0.024 [0.045, 0.100]
Coffee	0.000	0.000 [0.000, 0.000]	0.000 [0.000, 0.000]	0.000 [0.000, 0.000]
ECG200	0.230	0.110 [0.120, 0.150]	0.100 [0.120, 0.138] †‡	0.070 [0.083, 0.108] †‡
ECGFiveDays	0.232	0.015 [0.020, 0.145]	0.010 [0.025, 0.145] †	0.013 [0.014, 0.121] †‡
Face (all)	0.192	0.195 [0.209, 0.221]	0.172 [0.205, 0.217] ‡	0.143 [0.184, 0.221] ‡
Face (four)	0.170	0.170 [0.239, 0.284]	0.193 [0.222, 0.284]	0.170 [0.216, 0.267]
Fish	0.177	0.034 [0.051, 0.132]	0.097 [0.109, 0.175]	0.086 [0.097, 0.203]
Gun-Point	0.093	0.013 [0.013, 0.020]	0.013 [0.020, 0.050] †	0.007 [0.013, 0.027] †
Ham	0.533	0.410 [0.429, 0.476]	0.371 [0.405, 0.438] †‡	0.381 [0.434, 0.481] †
Lightning-2	0.131	0.148 [0.148, 0.222]	0.197 [0.230, 0.295]	0.131 [0.156, 0.189]
Lightning-7	0.274	0.260 [0.301, 0.329]	0.219 [0.254, 0.301] ‡	0.205 [0.260, 0.377]
OliveOil	0.167	0.133 [0.133, 0.133]	0.133 [0.133, 0.133] †	0.133 [0.133, 0.133] †
OSU Leaf	0.409	0.306 [0.324, 0.354]	0.281 [0.302, 0.333] †‡	0.269 [0.312, 0.347] †‡
ShapeletSim	0.350	0.356 [0.406, 0.442]	0.294 [0.395, 0.444]	0.356 [0.409, 0.486]
Swedish Leaf	0.208	0.115 [0.139, 0.156]	0.115 [0.127, 0.155] †	0.118 [0.136, 0.164] †
Synthetic Control	0.007	0.040 [0.068, 0.080]	0.017 [0.068, 0.092]	0.023 [0.057, 0.072] ‡
Trace	0.000	0.000 [0.000, 0.000]	0.010 [0.010, 0.050]	0.000 [0.010, 0.010]
Wine	0.426	0.389 [0.389, 0.389]	0.278 [0.389, 0.389] †‡	0.370 [0.389, 0.389] †

local derivatives with all parameters w had lower error rates compared to DTW in *Beef*, *ECGFiveDays*, *Gun-Point*, *Ham*, *Swedish Leaf* and *Wine* datasets.

DTW works better than any form of shapeDTW in *CBF*, *Face (four)*, *Lightning-2*, *ShapeletSim* and *Synthetic Control*. All time series of these datasets have one characteristic in common and that is they have at least one abrupt change in form of a jump. This might cause misalignment and thus misclassification in few cases when a local descriptor function is used.

5.4.2 ShapeDTW comparisons

Among the 24 datasets, the test indicated that the median error rates of shapeDTW using *normalized-subsequence* and *local derivatives* in 7 and 8 datasets, respectively, are statistically lower than those of shapeDTW using *raw-subsequence*. ShapeDTW using *raw-subsequence* had better results in 10 and 6 datasets when compared with shapeDTW using *normalized-subsequence* and *local derivatives*.

5.5 Conclusion

DTW distance measure is a heuristic to find the most proper alignment between two time series. The quality of the alignment (and the similarity measure value) highly impacts the performance of data mining tasks including time series classification, clustering, indexing, etc. This study showed the efficiency of the shapeDTW distance measure when combined with two descriptive functions: normalized subsequences and local derivatives. Several extension can be performed on this study. The first one can be a model that takes into account a penalty when aligned indices of two sequences are not equal, similar to [23]. Another extension can be done by limiting the warping path length [61].

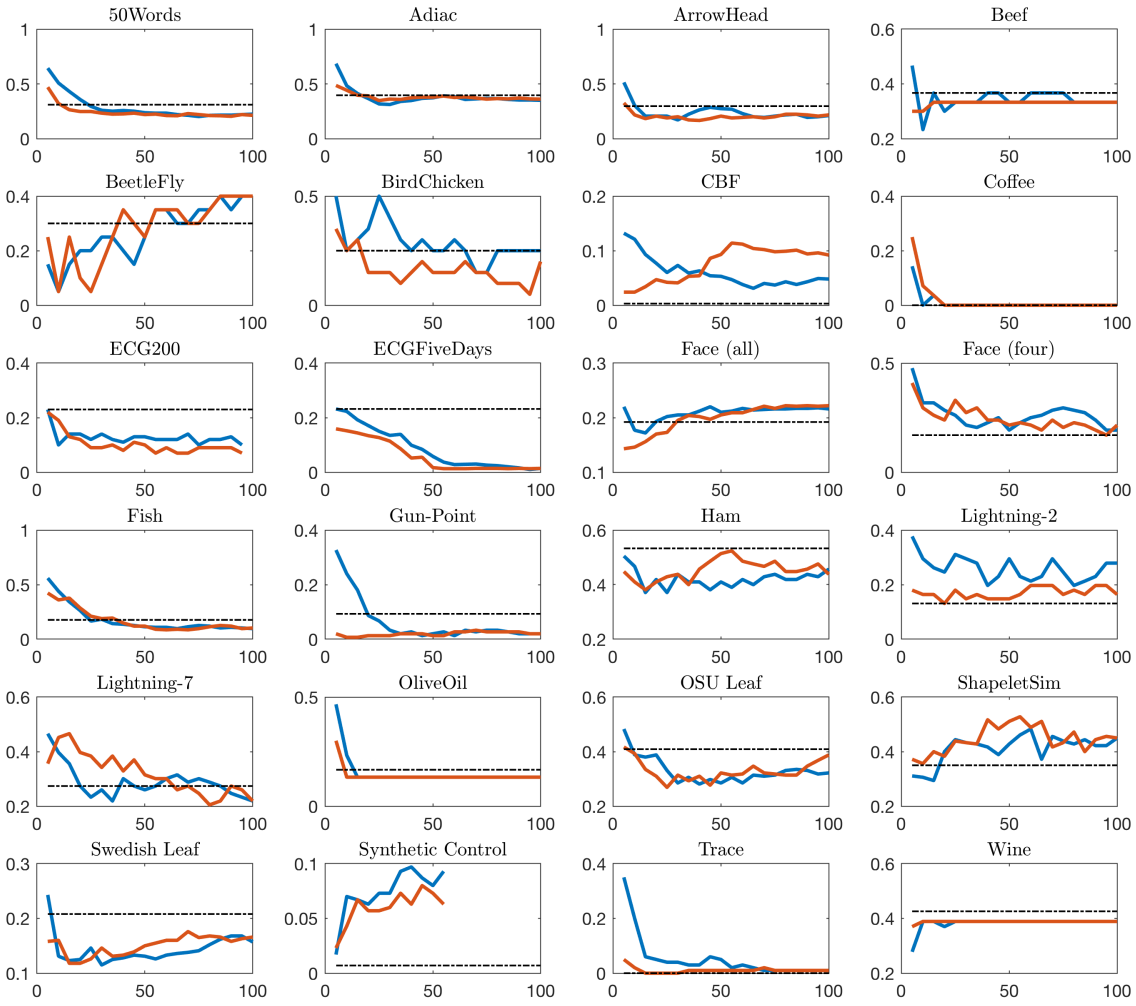

Figure 5.6: Error rates of 1-NN classifier of shapeDTW using *normalized-subsequence* (in blue) and *local derivatives* (in brown) as well as DTW error rates (in black dash-dot line) on 24 datasets.

Chapter 6

Conclusions and Future Work

This dissertation addresses two data mining tasks, *time series classification* and *anomaly detection*. The first problem is studied from three different perspectives. They are:

1. **Motifs with variable lengths (chapter 2)**

   The first approach attempts to label a time series by finding the class signatures, the subsequences that are likely to exist in a time series if it belongs to a certain class. Motif instances are found by a new motif discovery method called *deep search*. *Deep search* is based on the SAX representation of a time series and is capable of finding motif instances with different lengths. This approach extended the concept of shapelets, similar subsequences with a fixed length, to similar subsequences with variable lengths. Classification tree method is used for classification. For the future work, a combination of more advanced methods such as support vector machine and Bayesian network with features created by class motifs is suggested.

2. **SAX representation and its similarity measure (chapter 3)**

   In the second approach, the trend of each SAX letter segments in addition to the

segment average is taken into account. A new distance measure between any two time series is found based on the minimal distance of their SAX words and trends. This study attempted to create a bridge between two big topics *data mining* and *operations research* by presenting a new similarity measure using SAX representation. The analysis provided in this study created a new line of research and provided a framework for analyzing SAX distance measures. The optimization program took into account an important property of a normalized sequence, that the average is zero. For the future work, imposing some constraints concerning the standard deviation of different segments is suggested.

3. **Shape dynamic time warping with normalized subsequence and local derivatives (chapter 5)**

   DTW distance measure is a heuristic to find the most proper alignment between two time series. The quality of the alignment (and the similarity measure value) highly impacts the performance of data mining tasks including time series classification, clustering, indexing, etc. This study showed the efficiency of shape dynamic time warping distance measure when combined with two descriptive functions *normalized subsequence* and *local derivatives*. Several extensions can be considered for this study. They are:

   - Taking into account a penalty in the objective function when aligned indices of two sequences are not equal, similar to [23]
   - Limiting the warping path length as shown in [61]

In chapter 4, the *anomaly detection* problem is studied by employing Markov chains. The purpose of this problem is to find segments (rather than a single data point in classic anomaly detection problems) of a time series whose behaviors do not fit the expected

behavior of the underlying system. The Markov chains modeling is used to learn the behavior of the consecutive subsequences on a normal dataset. The trained model is used to assess how well the testing dataset suits the baseline behavior. The designed *anomaly detection* framework is capable of discovering unusual patterns in both streaming data (online) and stored data (offline). Several extensions can be made on this topic. They include:

- Studying anomalies in multivariate time series;
- Using hidden Markov models;
- Using the variety of shape descriptor functions; and
- Studying the *assessment* step by taking into account the quality of the clustering step and distances of each subsequence from cluster centroids as a feature.

REFERENCES

[1] Fitbit alta fitness wristband, November 2017. https://www.fitbit.com/alta.

[2] The similarity procedure, July 2018. http://support.sas.com/documentation/cdl/en/etsug/63939/HTML/default/viewer.htm#etsug_similarity_sect001.htm.

[3] Shikha Agrawal and Jitendra Agrawal. Survey on anomaly detection using data mining techniques. *Procedia Computer Science*, 60:708–713, 2015.

[4] Subutai Ahmad, Alexander Lavin, Scott Purdy, and Zuha Agha. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, 262:134–147, 2017.

[5] Arvind Balasubramanian, Jun Wang, and Balakrishnan Prabhakaran. Discovering multidimensional motifs in physiological signals for personalized healthcare. *IEEE Journal of Selected Topics in Signal Processing*, 10(5):832–841, 2016.

[6] S Brindasri and K Saravanan. Evaluation of network intrusion detection using markov chain. *International Journal on Cybernetics & Informatics (IJCI)*, 3(2):11–20, 2014.

[7] Jeremy Buhler and Martin Tompa. Finding motifs using random projections. *Journal of computational biology*, 9(2):225–242, 2002.

[8] Nicola Bui and Michele Zorzi. Health care applications: a solution based on the internet of things. In *Proceedings of the 4th International Symposium on Applied Sciences in Biomedical and Communication Technologies*, page 131. ACM, 2011.

[9] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

[10] Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista. The ucr time series classification archive, July 2015. www.cs.ucr.edu/~eamonn/time_series_data/.

[11] Haibin Cheng, Pang-Ning Tan, Christopher Potter, and Steven Klooster. Detection and characterization of anomalies in multivariate time series. In *Proceedings of the 2009 SIAM International Conference on Data Mining*, pages 413–424. SIAM, 2009.

[12] Bill Chiu, Eamonn Keogh, and Stefano Lonardi. Probabilistic discovery of time series motifs. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 493–498. ACM, 2003.

[13] Zhiguo Ding and Minrui Fei. An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window. *IFAC Proceedings Volumes*, 46(20):12–17, 2013.

[14] Enkhjargal Dorj, Chaochao Chen, and Michael Pecht. A bayesian hidden markov model-based approach for anomaly detection in electronic systems. In *IEEE Aerospace Conference Proceedings*, pages 1–10. IEEE, 2013.

[15] Sarah M Erfani, Sutharshan Rajasegarar, Shanika Karunasekera, and Christopher Leckie. High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning. *Pattern Recognition*, 58:121–134, 2016.

[16] Muhammad Marwan Muhammad Fuad. Genetic algorithms-based symbolic aggregate approximation. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 105–116. Springer, 2012.

[17] Takashi Fuse and Keita Kamiya. Statistical anomaly detection in human dynamics monitoring using a hierarchical dirichlet process hidden markov model. *IEEE Transactions on Intelligent Transportation Systems*, 18(11):3083–3092, 2017.

[18] Pierre Geurts. Pattern extraction for time series classification. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 115–127. Springer, 2001.

[19] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. Physiobank, physiotoolkit, and physionet. *Circulation*, 101(23):e215–e220, 2000.

[20] Jon Hills, Jason Lines, Edgaras Baranauskas, James Mapp, and Anthony Bagnall. Classification of time series by shapelet transformation. *Data Mining and Knowledge Discovery*, 28(4):851–881, 2014.

[21] SM Riazul Islam, Daehan Kwak, MD Humaun Kabir, Mahmud Hossain, and Kyung-Sup Kwak. The internet of things for health care: a comprehensive survey. *IEEE Access*, 3:678–708, 2015.

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

[23] Young-Seon Jeong, Myong K Jeong, and Olufemi A Omitaomu. Weighted dynamic time warping for time series classification. *Pattern Recognition*, 44(9):2231–2240, 2011.

[24] Wen-Hua Ju and Yehuda Vardi. A hybrid high-order markov chain model for computer intrusion detection. *Journal of Computational and Graphical Statistics*, 10(2):277–295, 2001.

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

[26] Eamonn Keogh, Jessica Lin, and Ada Fu. Hot sax: Efficiently finding the most unusual time series subsequence. In *Proceedings of the 5th IEEE international conference on data mining*, pages 226–233. IEEE, 2005.

[27] Eamonn Keogh and Chotirat Ann Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and information systems*, 7(3):358–386, 2005.

[28] Eamonn J Keogh and Michael J Pazzani. Derivative dynamic time warping. In *Proceedings of the 2001 SIAM International Conference on Data Mining*, pages 1–11. SIAM, 2001.

[29] Pablo Laguna, Roger G Mark, A Goldberg, and George B Moody. A database for evaluation of algorithms for measurement of qt and other waveform intervals in the ecg. In *Computers in Cardiology 1997*, pages 673–676. IEEE, 1997.

[30] Yuan Li, Jessica Lin, and Tim Oates. Visualizing variable-length time series motifs. In *Proceedings of the 2012 SIAM international conference on data mining*, pages 895–906. SIAM, 2012.

[31] Jessica Lin, Eamonn Keogh, and Stefano Lonardi. Visualizing and discovering non-trivial patterns in large time series databases. *Information visualization*, 4(2):61–82, 2005.

[32] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 2–11. ACM, 2003.

[33] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Pranav Patel. Finding motifs in time series. In *Proceedings of 2nd workshop on temporal data mining*, pages 53–68, 2002.

[34] Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. Experiencing sax: a novel symbolic representation of time series. *Data Mining and knowledge discovery*, 15(2):107–144, 2007.

[35] Jessica Lin, Rohan Khade, and Yuan Li. Rotation-invariant similarity in time series using bag-of-patterns representation. *Journal of Intelligent Information Systems*, 39(2):287–315, 2012.

[36] Jessica Lin and Yuan Li. Finding structural similarity in time series data using bag-of-patterns representation. In *Scientific and statistical database management*, pages 461–477. Springer, 2009.

[37] Battuguldur Lkhagva, Yu Suzuki, and Kyoji Kawagoe. New time series data representation esax for financial applications. In *Data Engineering Workshops, 2006. Proceedings. 22nd International Conference on*, pages x115–x115. IEEE, 2006.

[38] Abdullah Mueen and Nikan Chavoshi. Enumeration of time series motifs of all lengths. *Knowledge and Information Systems*, 45(1):105–132, 2015.

[39] Srinivas Mukkamala, Guadalupe Janoski, and Andrew Sung. Intrusion detection using neural networks and support vector machines. In *Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on*, volume 2, pages 1702–1707. IEEE, 2002.

[40] Tim Oates, Laura Firoiu, and Paul R Cohen. Clustering time series with hidden markov models and dynamic time warping. In *Proceedings of the IJCAI-99 workshop on neural, symbolic and reinforcement learning methods for sequence learning*, pages 17–21. Citeseer, 1999.

[41] François Petitjean, Germain Forestier, Geoffrey I Webb, Ann E Nicholson, Yanping Chen, and Eamonn Keogh. Dynamic time warping averaging of time series allows faster and more accurate classification. In *Data Mining (ICDM), 2014 IEEE International Conference on*, pages 470–479. IEEE, 2014.

[42] François Petitjean, Alain Ketterlin, and Pierre Gançarski. A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recognition*, 44(3):678–693, 2011.

[43] Felix Richter. The predicted wearables boom is all about the wrist. *Statista, April*, 8, 2015.

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

[45] Pavel Senin, Jessica Lin, Xing Wang, Tim Oates, Sunil Gandhi, Arnold P Boedihardjo, Crystal Chen, and Susan Frankenstein. Time series anomaly discovery with grammar-based compression. In *EDBT*, pages 481–492, 2015.

[46] Wenyao Sha, Yongxin Zhu, Tian Huang, Meikang Qiu, Yan Zhu, and Qiannan Zhang. A multi-order markov chain based scheme for anomaly detection. In *Computer Software and Applications Conference Workshops (COMPSACW), 2013 IEEE 37th Annual*, pages 83–88. IEEE, 2013.

[47] Jin Shieh and Eamonn Keogh. i sax: indexing and mining terabyte sized time series. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 623–631. ACM, 2008.

[48] Youqiang Sun, Jiuyong Li, Jixue Liu, Bingyu Sun, and Christopher Chow. An improvement of symbolic aggregate approximation distance measure for time series. *Neurocomputing*, 138:189–198, 2014.

[49] Hossein Tayebi, Shonali Krishnaswamy, Augustinus Borgy Waluyo, Abhijat Sinha, and Mohamed Medhat Gaber. Ra-sax: resource-aware symbolic aggregate approximation for mobile ecg analysis. In *Mobile Data Management (MDM), 2011 12th IEEE International Conference on*, volume 1, pages 289–290. IEEE, 2011.

[50] Sahar Torkamani and Volker Lohweg. Survey on time series motif discovery. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 7(2), 2017.

[51] Jagannadan Varadarajan, Rémi Emonet, and Jean-Marc Odobez. Probabilistic latent sequential motifs: Discovering temporal activity patterns in video scenes. Technical report, Idiap, 2010.

[52] Li Wei, Nitin Kumar, Venkata Nishanth Lolla, Eamonn J Keogh, Stefano Lonardi, and Chotirat (Ann) Ratanamahatana. Assumption-free anomaly detection in time series. In *SSDBM*, volume 5, pages 237–242, 2005.

[53] Xiaopeng Xi, Eamonn Keogh, Christian Shelton, Li Wei, and Chotirat Ann Ratanamahatana. Fast time series classification using numerosity reduction. In *Proceedings of the 23rd international conference on Machine learning*, pages 1033–1040. ACM, 2006.

[54] Dragomir Yankov, Eamonn Keogh, Jose Medina, Bill Chiu, and Victor Zordan. Detecting time series motifs under uniform scaling. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 844–853. ACM, 2007.

[55] Lexiang Ye and Eamonn Keogh. Time series shapelets: a new primitive for data mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 947–956. ACM, 2009.

[56] Nong Ye et al. A markov chain model of temporal behavior for anomaly detection. In *Proceedings of the 2000 IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop*, volume 166, page 169. West Point, NY, 2000.

[57] Nong Ye, Xiangyang Li, Qiang Chen, Syed Masum Emran, and Mingming Xu. Probabilistic techniques for intrusion detection based on computer audit data. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 31(4):266–274, 2001.

[58] Nong Ye, Yebin Zhang, and Connie M Borror. Robustness of the markov-chain model for cyber-attack detection. *IEEE Transactions on Reliability*, 53(1):116–123, 2004.

[59] Chaw Thet Zan and Hayato Yamana. An improved symbolic aggregate approximation distance measure based on its statistical features. In *Proceedings of the 18th International Conference on Information Integration and Web-based Applications and Services*, pages 72–80. ACM, 2016.

[60] Xueqin Zhang, Chunhua Gu, and Jiajun Lin. Support vector machines for anomaly detection. In *Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on*, volume 1, pages 2594–2598. IEEE, 2006.

[61] Zheng Zhang, Romain Tavenard, Adeline Bailly, Xiaotong Tang, Ping Tang, and Thomas Corpetti. Dynamic time warping under limited warping path length. *Information Sciences*, 393:91–107, 2017.

[62] Jiaping Zhao and Laurent Itti. shapedtw: Shape dynamic time warping. *Pattern Recognition*, 74:171–184, 2018.