TIME SERIES DATA MINING: ENHANCEMENTS IN UNIVARIATE AND MULTIVARIATE REPRESENTATIONS, DISTANCE MEASURES AND TIME SERIES SIMILARITY SEARCH

ΛΕΩΝΙΔΑΣ ΚΑΡΑΜΗΤΟΠΟΥΛΟΣ

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Επιβλέπων Καθηγητής: Γεώργιος Ευαγγελίδης
Μέλη Τριμελούς Επιτροπής: Δημήτριος Α. Δέρβος
Κωνσταντίνος Μαργαρίτης

Τμήμα Εφαρμοσμένης Πληροφορικής

Πανεπιστήμιο Μακεδονίας
Θεσσαλονίκη

Δεκέμβριος 2008

Στους γονείς μου,

Τριαντάφυλλο και Δέσποινα,

και στη μνήμη του αδελφού μου,

Παναγιώτη

ΜΕΛΗ ΤΗΣ ΕΞΕΤΑΣΤΙΚΗΣ ΕΠΙΤΡΟΠΗΣ
(κατά βαθμίδα και αλφαβητικά)

Δ. Δέρβος, Καθηγητής (μέλος της τριμελούς συμβουλευτικής επιτροπής)

Ι. Μανωλόπουλος, Καθηγητής

Κ. Μαργαρίτης, Καθηγητής (μέλος της τριμελούς συμβουλευτικής επιτροπής)

Ι. Παπαδημητρίου, Καθηγητής

Δ. Παπαναστασίου, Καθηγητής

Γ. Ευαγγελίδης, Αναπληρωτής Καθηγητής (επιβλέπων)

Ν. Σαμαράς, Επίκουρος Καθηγητής

Ευχαριστίες

Θα ήθελα να εκφράσω τις ειλικρινείς ευχαριστίες μου σε όλους όσους συνέβαλαν με οποιοδήποτε τρόπο στην ερευνητική αυτή προσπάθεια.

Θερμά ευχαριστώ τα μέλη της συμβουλευτικής επιτροπής, τον Καθηγητή κ. Δημήτριο Δέρβο για τις συστηματικές υποδείξεις και συμβουλές του κατά την ερευνητική διαδικασία, όπως επίσης και τον Καθηγητή κ. Μαργαρίτη, κυρίως, για τις πολύτιμες παρατηρήσεις και προτάσεις του σχετικά με την διεύρυνση της ερευνητικής μου δραστηριότητας. Ιδιαίτερα, θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου, Αναπληρωτή Καθηγητή κ. Γεώργιο Ευαγγελίδη, για την πρόθυμη και αδιάλειπτη υποστήριξη και καθοδήγησή του σε όλη τη διάρκεια της εκπόνησης της διατριβής.

Επίσης, θα ήθελα να ευχαριστήσω τα υπόλοιπα μέλη της επταμελούς εξεταστικής επιτροπής, Ι. Μανωλόπουλο, Καθηγητή, Ι. Παπαδημητρίου, Καθηγητή, Δ. Παπαναστασίου, Καθηγητή, Ν. Σαμαρά, Επίκουρο Καθηγητή, για την επιστημονική τους κριτική και τις χρήσιμες υποδείξεις τους. Η εμπειρία της παρουσίασης της παρούσας διατριβής και της εξεταστικής διαδικασίας που ακολούθησε αποτελεί ένα σημαντικό εφόδιο για την περαιτέρω ερευνητική μου δραστηριότητα.

Μη εμφανής στο αποτέλεσμα της διατριβής, αλλά σημαντική για μένα ήταν η συνδρομή του φίλου μου, Νίκου Αγαθόνικου, ο οποίος αποτελεσματικά με εισήγαγε σε σχετικές με την έρευνά μου επιστημονικές έννοιες τις οποίες δεν κατείχα έως τότε.

Τέλος, αφιερώνω τη διατριβή αυτή στους γονείς μου, οι οποίοι με υποστήριξαν ηθικά και οικονομικά όλα αυτά τα χρόνια.

ΠΕΡΙΛΗΨΗ

Στην παρούσα διατριβή, διερευνούμε διάφορες τεχνικές για την αποτελεσματική εφαρμογή μεθόδων Εξόρυξης Πληροφορίας σε Χρονοσειρές από μεγάλες βάσεις δεδομένων. Οι κύριες ενέργειες που πραγματοποιούνται με την εφαρμογή των μεθόδων αυτών είναι οι εξής: συσταδοποίηση, κατηγοριοποίηση, εντοπισμός καινοτομιών, ανακάλυψη μοτίβων και ανακάλυψη κανόνων. Όλες οι ενέργειες αυτές εμπεριέχουν την έννοια της ομοιότητας, αφού απαιτούν την αναζήτηση όμοιων προτύπων. Η χρονική διάσταση των δεδομένων ανακύπτει δύο βασικά ζητήματα τα οποία θα πρέπει να λαμβάνονται υπόψη κατά την αναζήτηση ομοιοτήτων. Το πρώτο ζήτημα είναι η επιλογή ενός κατάλληλου μέτρου ομοιότητας, το οποίο θα πρέπει να επιτρέπει τον εντοπισμό όμοιων χρονοσειρών οι οποίες δεν ταυτίζονται απαραιτήτως. Το δεύτερο ζήτημα είναι η αναπαράσταση (μετασχηματισμός) των χρονοσειρών με στόχο την μείωση της υψηλής διαστατότητάς τους, η οποία είναι σύμφυτη σε αυτές.

Η παρούσα έρευνα εστιάζεται σε μονοδιάστατες και πολυδιάστατες χρονοσειρές. Στην πρώτη περίπτωση, η ομοιότητα αναζητείται σε χρονοσειρές μίας διάστασης, ενώ στην δεύτερη περίπτωση η ομοιότητα αναζητείται μεταξύ αντικειμένων κάθε ένα από τα οποία ορίζεται από ένα σύνολο μονοδιάστατων χρονοσειρών.

Οι σημαντικότερες συνεισφορές της παρούσας εργασίας αναφέρονται στην συνέχεια. Πρώτον, προτείνεται μία μέθοδος Εξόρυξης Πληροφορίας σε Χρονοσειρές για την Αναγνώριση Προτύπων σε Διαγράμματα Ελέγχου. Αναδεικνύουμε την ικανότητα των μεθόδων Εξόρυξης Πληροφορίας σε Χρονοσειρές στην αντιμετώπιση προβλημάτων τα οποία παραδοσιακά προσεγγίζονται με μεθόδους των οποίων η αποτελεσματικότητα περιορίζεται κάθε φορά στην συγκεκριμένη εφαρμογή για την οποία έχει σχεδιαστεί.

Δεύτερον, προτείνουμε μία πρωτότυπη αναπαράσταση μονοδιάστατων χρονοσειρών και ένα αντίστοιχο μέτρο ομοιότητας με σκοπό τη βελτίωση της ποιότητας της αναζήτησης ομοιοτήτων διατηρώντας την απαιτούμενη αποδοτικότητά της.

Η τρίτη συνεισφορά της εργασίας αυτής συνίσταται στον προσδιορισμό μίας νέας τεχνικής η οποία στοχεύει στην επιτάχυνση της διαδικασίας αναζήτησης του εγγύτερου γείτονα μίας χρονοσειράς. Η τεχνική αυτή περιλαμβάνει την αναπαράσταση των αρχικών χρονοσειρών και τον διαχωρισμό τους σε ένα πλήθος συστάδων.

Τέταρτη συνεισφορά αποτελεί η παρουσίαση μίας νέας προσέγγισης στην αναζήτηση όμοιων πολυδιάστατων χρονοσειρών. Η προσέγγιση αυτή περιλαμβάνει μία μέθοδο αναπαράστασης που βασίζεται στην τεχνική της Ανάλυσης σε Κύριες Συνιστώσες και σε μία πρωτότυπη τεχνική μέτρησης της ομοιότητας μεταξύ πολυδιάστατων αντικειμένων.

Πέμπτον, η παρούσα εργασία παρέχει μία εκτεταμένη βιβλιογραφική ανασκόπηση σε μεθόδους Εξόρυξης Πληροφορίας σε Πολυδιάστατες Χρονοσειρές.

Οι προτεινόμενες μέθοδοι της διατριβής αυτή έχουν αξιολογηθεί πειραματικά ως προς την ποιότητα της αναζήτησης ομοιότητας σε ένα ευρύ πλήθος πραγματικών και συνθετικών δεδομένων.

ABSTRACT

In this dissertation, we investigate various techniques for efficiently applying Time Series Data Mining methods in very large databases. The main tasks of these methods are: clustering, classification, novelty detection, motif discovery and rule discovery. At the core of these tasks lies the concept of similarity, since most of them require searching for similar patterns. The temporal nature of data arises two special issues to be considered in the process of similarity search. The first one is the definition of an appropriate similarity measure that allows imprecise matches among time series. The second issue is the representation of time series in order to reduce the intrinsically high dimensionality present in this type of data.

Our research focuses on univariate, as well as, on multivariate time series. In the first case, similarity is sought among one-dimensional time series, whereas in the latter case, similarity is sought among objects, which consist of a set of time series.

There are five major contributions of this work. First, we propose a Time Series Data Mining approach in the task of control chart pattern recognition. We demonstrate the capability of Time Series Data Mining techniques in handling tasks that traditionally are approached by application-specific methods.

Second, we present a novel representation for dimensionality reduction along with an appropriate measure in order to improve the quality of similarity search while retaining the required efficiency.

Third, we propose a new technique that aims at accelerating one-nearest neighbor similarity search. This technique involves the application of a representation on the original time series and, subsequently, the partition of the search space into a number of clusters.

Fourth, we present a novel approach in multivariate time series similarity search that includes a representation based on Principal Components Analysis and a new technique of measuring similarity among multivariate objects.

Fifth, we provide an extensive literature review of multivariate time series data mining.

All the proposed methods in this dissertation have been experimentally evaluated on the quality of similarity search with respect to a wide range of real-world and synthetic datasets.

CONTENTS

CHAPTER 1 Introduction 13
CHAPTER 2 Background on Time Series Data Mining 19
2.1 Time Series Data Ming Tasks 19
2.2 Time Series Data Mining Issues 21
2.3 The Pre-processing Phase 22
2.4 Similarity Measures 24
2.4.1 Characteristics of Similarity Measures 24
2.4.2 Euclidean Distance 26
2.4.3 Dynamic Time Warping 27
2.4.4 Additional Similarity Measures 31
2.5 Representations 31
2.5.1 Discrete Fourier Transform 32
2.5.2 Singular Value Decomposition 33
2.5.3 Piecewise Aggregate Approximation 37
2.5.4 A Feature Based Representation 38
2.5.5 Symbolic Aggregate Approximation 39
2.5.6 Additional Representations 40
2.6 Summary 43
CHAPTER 3 A TSDM Approach to Control Chart Pattern Recognition 45
3.1 Introduction 45
3.2 Control Chart Pattern Recognition 46
3.2.1 Review of Control Charts 46
3.2.2 Related Work 47
3.3 Data Mining & Control Chart Pattern Recognition 50
3.3.1 A Review of Decision Trees 50
3.3.2 The Proposed Approach 52
3.4 Experimental Framework 54
3.4.1 Description of the Datasets 54
3.4.2 Methods & Settings 55
3.5 Results 56
3.5.1 Comparisons among Representation Schemes 56
3.5.2 Comparisons Among SAX Representations Per Pattern 57
3.6 Conclusion 60
CHAPTER 4 A Hybrid Time Series Representation 63
4.1 Introduction 63
4.2 The D-PAA Representation 64
4.2.1 Description of D-PAA Representation 64
4.2.2 Graphical Description of D-PAA Representation 69
4.3 Framework of Experimentation 71
4.3.1 Datasets 71
4.3.2 Methods 72
4.3.3 Rival Approaches - Parameters 73
4.4 Results 74
4.4.1 Feature-based Representation (FB): Investigating Parameter D 74
4.4.2 D-PAA: Comparison of Distance Measures 77

4.4.3 Comparison between D-PAA and PAA 79
4.4.4 Comparison among D-PAA and other Rival Approaches..... 81
4.5 Conclusion..... 83
CHAPTER 5 Cluster Based Similarity Search 85
5.1 Introduction 85
5.2 Background 87
5.2.1 Clustering 87
5.2.2 Related Work 88
5.3 A Proposed Approach 90
5.4 Framework of Experimentation..... 95
5.5 Results..... 96
5.6 Conclusions 99
CHAPTER 6 Issues on Multivariate Time Series Data Mining – PCA..... 103
6.1 Introduction 103
6.2 Multivariate Time Series: Representations & Similarity Measures 104
6.3 Literature Review of Multivariate Time Series Data Mining 106
6.4 Review of Principal Component Analysis 109
6.4.1 Preliminaries 110
6.4.2 Mathematical Formulation of PCA..... 111
6.4.3 Intuition behind PCA 114
6.4.4 PCA & SVD 115
6.5 Principal Component Analysis for Time Series 116
6.6 Implications of PCA in Similarity Search..... 118
6.7 Conclusion..... 121
CHAPTER 7 PCA-based Measures for Similarity Search 123
7.1 Introduction 123
7.2 Related work..... 124
7.3 Proposed Approach..... 128
7.4 Experimental Methodology..... 133
7.4.1 Datasets 133
7.4.2 Evaluation Methods 134
7.4.3 Rival Measures 135
7.5 Results..... 136
7.5.1 1-NN Classification..... 136
7.5.2 k-NN Similarity Search 139
7.5.3 Speeding up Calculation of SPEdist 141
7.6 Conclusion..... 141
CHAPTER 8 Conclusions – Future Work 143
REFERENCES..... 147
APPENDICES 157

CHAPTER 1

Introduction

Technological advances in automated monitoring systems and storage devices have facilitated the generation of huge amounts of data in almost every domain such as in business, industry, government or medicine. Often, this volume of data remains unexploited, since the traditional methods of analyzing data have become inadequate. In the last decade, there has been an increasing interest in the Data Mining field, which involves techniques and algorithms capable of efficiently extracting patterns that can potentially constitute knowledge from very large databases. A special case is when data is in the form of time series that is, a collection of observations made sequentially through time. At each time point one or more measurements may be monitored corresponding to one or more attributes under consideration. The resulting time series is called univariate or multivariate respectively. Time Series Data Mining is the field that is comprised by data mining methods adjusted in a way that they take into consideration the temporal nature of data. According to the research in this field, the main tasks of TSDM methods are: clustering, classification, novelty detection, motif discovery and rule discovery. At the core of these tasks lies the concept of similarity, since most of them require searching for similar patterns. Although some of these tasks are similar to the data mining tasks, the temporal aspect arises two special issues to be considered and/or imposes some restrictions in the corresponding applications. First, in most of the above tasks, apparently, it is necessary to define a similarity measure between two time series. The second issue that arises and interrelates to the selection of a similarity measure is the representation of a time series. Since the amount of data may range from a few megabytes to terabytes, an appropriate representation of the time series is necessary in order to manipulate and analyze it efficiently. The main objective is to reduce the intrinsically high dimensionality of time series and thus, deal with the problem of the "dimensionality curse" that appears frequently within real world data mining applications. The rationale behind this procedure is to reduce the dimensionality of original data by representing it in a lower dimension, analyze it in this dimension and, finally, tune the results in order to obtain the same solution with one that would have been derived, if the original data had been used in the analysis.

The core work of this dissertation concentrates on similarity search of univariate and multivariate time series [79] [80]. The general objective is to provide methods for efficiently applying data mining techniques in very large databases.

Regarding the univariate case, our research is driven in three directions in order to address the issues stated in the previous paragraph. More specifically, we propose a novel representation for dimensionality reduction along with an appropriate similarity measure that can be tuned according to any specific application [85]. Note that the dimensionality of a time series is the number of time points at which it is recorded (i.e. time series length). We present a new method that partitions the search space for the purpose of accelerating similarity search. The main idea is to identify the most similar time series to a given query without necessarily searching over the whole database. Finally, we propose a Time Series Data Mining method in a specific task, namely the control chart pattern recognition [78] [83]. Our intention is to demonstrate the broad applicability of Time Series Data Mining techniques in handling tasks that traditionally are approached by application-specific methods. An application (part of our research work) that further demonstrates the broad applicability of Time Series Data Mining techniques and relates to meteorological data can be found in [145].

Regarding the multivariate case, our research is also focused on representations and similarity measures. A multivariate time series can be considered as a set of time series recorded at the same time interval. Contrary to the univariate case, similarity search is sought among objects that can be considered as matrices, where columns correspond to individual time series (variables) and rows correspond to time instances. Consequently, the problem of high dimensionality escalates, since it is not only the dimension that corresponds to the length of time series, but also the dimension that corresponds to the number of individual time series. Although, we provide an extensive literature review on the case of multivariate time series, our work is mainly focused on the application of Principal Component Analysis for the purpose of reducing dimensionality. This technique has been widely applied in many diverse cases, however, it has not been extensively explored with respect to similarity search. We provide a method that includes a novel similarity measure [82] [84] along with an appropriate technique for further speeding up similarity search [81].

All the proposed methods in this dissertation have been experimentally evaluated on the quality of similarity search. More specifically, the evaluation methods include classification accuracy and precision-recall graphs with respect to a wide range of real-world and synthetic datasets.

Figure 1.1 presents the areas of research contribution of this dissertation and the contributions of each chapter are concisely described below.

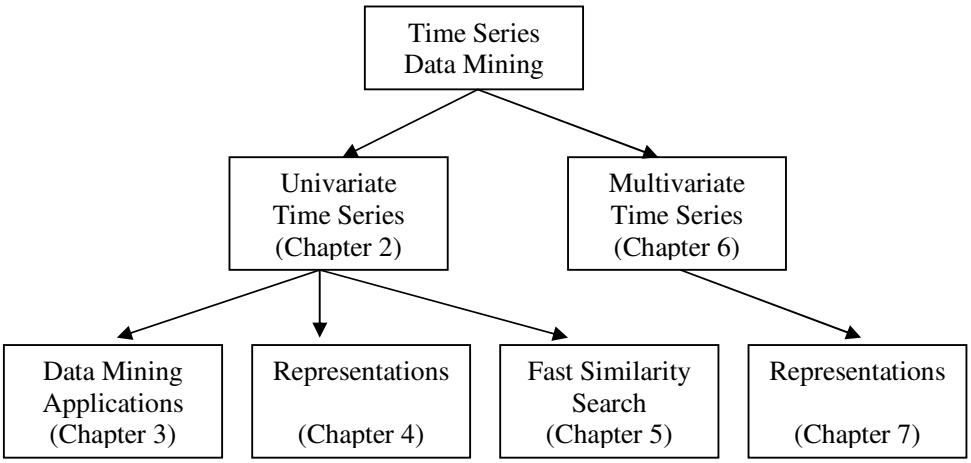


Figure 1.1 Diagram of the areas of contribution of this dissertation

Chapter 2 introduces the various aspects of Time Series Data Mining. We present the main tasks that this field involves, such as clustering, classification, novelty detection, motif discovery and rule discovery. We discuss the notion of similarity, which is central to most of these tasks, and we present in detail two of the most important similarity (distance) measures, namely the Euclidean distance and the Dynamic Time Warping. Although the first measure has been utilized in the majority of research in this area, it is highly sensitive to some "distortions" in the data. For example, two series may be similar in shape but out of phase with respect to the time axis. The technique of Dynamic Time Warping allows an elastic matching of two series even when they present time misplacements. Another issue that is discussed in this chapter is the need for transforming the original time series in order to reduce the inherently high dimensionality of time series. For this purpose, many representation schemes that have been proposed in the literature are briefly presented. Nevertheless, we analytically describe five representations that have been utilized in the subsequent chapters, namely Discrete Fourier Transform, Singular Value Decomposition, Piecewise Aggregate Approximation, Symbolic Aggregate Approximation and Feature Based representation.

In Chapter 3, we investigate the possible contribution of Time Series Data Mining in control chart pattern recognition. The control chart is the primary tool in Statistical Process Control that aids in monitoring a process and identifying deviations from its normal operating conditions. There are several possible patterns that indicate the existence of

specific causes, which lead the process into an out of control state. The objective is to automatically recognize and classify a control chart into one of several pre-determined patterns. We introduce a Time Series Data Mining approach in control chart pattern recognition that is based on Decision Trees. The scope of our work is to indicate the flexibility that Time Series Data Mining techniques offer in dealing with tasks that traditionally are approached either manually or by application-specific methods. We also propose the representation of the original data prior to the construction of the decision tree. Two different representations are selected for their different characteristics: the Feature Based representation and the Symbolic Aggregate Approximation. Experiments are conducted on synthetic datasets taking into consideration many parameters such as noise, magnitude of deviations and presence of in-control data.

Chapter 4 introduces a novel representation (D-PAA) that can be considered as a variation of Piecewise Aggregate Approximation. A time series is segmented into a series of equal length sections and the corresponding mean and standard deviation are recorded for each one of them. The difference with Piecewise Aggregate Approximation is that D-PAA takes into consideration not only the central tendency but also the dispersion present in each section. We propose an appropriate distance measure under this representation, which includes a parameter that assigns different weights to means and standard deviations. This setting allows the user to adjust the proposed distance measure according to the characteristics of the data under consideration. In addition to that, it is proved that this measure lower bounds the Euclidean distance. In this part of the work, extensive experiments are conducted on twenty widely utilized datasets in the literature and appropriate comparisons are made among several representations and distance measures.

In Chapter 5, we introduce a new approach to similarity search that reduces the search space by applying clustering in a database. The first step is to group time series into clusters by applying the k-means method. When the most similar series to a given query is required, the search is performed hierarchically starting from the cluster that lies most closely to the query. According to the proposed procedure, we may reach the most similar series without searching all clusters. Although this approach has been proposed in the literature and evaluated for non time series data, in this chapter, we experimentally evaluate it in the context of time series. In addition to that, we investigate the effectiveness of applying a specific representation scheme before clustering. We select as a representation the Piecewise Aggregate Approximation for its simplicity and efficiency. Experiments are conducted on fifteen real-world and synthetic datasets covering a wide range of applications.

Chapter 6 presents the case of multivariate time series, that is, objects that consist of a set of time series being generated at the same time interval. Each time series corresponds to a variable that is being recorded through time. For example, suppose that we monitor weather conditions in a particular city for one month. If we record the values of more than one variable, such as the minimum temperature, the humidity and the wind speed, we generate a multivariate time series. The objective is to identify cities that have similar weather conditions. In general, this problem is different than the one of univariate time series because the dimensionality of an object is defined not only with respect to the length of the series but also to the number of variables. To our knowledge, there is no work that reviews thoroughly this case. The first objective of this chapter is to provide literature review of multivariate time series data mining covering suggested representations and similarity measures. The second objective is to investigate the implications of applying Principal Component Analysis in the context of similarity search. This is a well known statistical technique that aims at reducing the number of variables while retaining as much as possible of the variation (information) present in the original data. We provide a thorough discussion on the corresponding representations and similarity/distance measures, as well as, on the pre-processing phase with respect to the most frequently appeared "distortions" in data.

In Chapter 7, we introduce a novel approach in multivariate time series similarity search that is based on Principal Component Analysis (PCA). This approach utilizes a novel distance measure, which is an extension of the Squared Prediction Error, a well-known statistic in the Statistical Process Control community. More specifically, we propose to apply PCA on every object (multivariate time series) in a database and to retain the appropriate information in order to be able to reconstruct the original series as accurately as possible. This information may be considered as a model representation that adequately describes the original series. For each object in a database, we quantify how well the corresponding model representation fits to the query object. Contrary to other PCA-based measures proposed in the literature, this measure does not require applying the computationally expensive PCA technique on the query. Moreover, we provide a method that further speeds up the calculations without affecting substantially the quality of similarity search. According to this method, Piecewise Aggregate Approximation is applied on each one of the time series that form the query object during the pre-processing phase. This technique segments a time series into consecutive sections of equal-width and calculates the corresponding mean for each one. The series of these means is the new representation of the original series. In this chapter, we also review several PCA-based similarity/distance measures that have been proposed by researchers from diverse scientific fields, and we test several of them against our approach.

CHAPTER 2

Background on Time Series Data Mining

2.1 Time Series Data Ming Tasks

A time series is a collection of observations made sequentially through time. At each time point one or more measurements may be monitored corresponding to one or more variables under consideration. The resulting time series is called univariate or multivariate respectively.

The Data Mining (DM) field involves techniques and algorithms capable of efficiently extracting patterns from large databases that can potentially constitute knowledge. Time Series Data Mining (TSDM) is a relatively new field that is comprised by DM methods adjusted in a way that they take into consideration the temporal nature of data. According to the research in this field, the main tasks of TSDM methods are: query by content, clustering, classification, novelty detection, motif discovery and rule discovery [128].

Query by content, one of the earliest TSDM tasks that was investigated, involves finding the most similar time series in a large database to a query time series [4]. For example, given the closing prices of a stock during a specific period, find those stocks that showed similar evolvement through time.

Classification involves assigning a given time series to one of several predefined groups (Figure 2.1). The purpose is to find that group, whose time series are the most similar to the given one. The task of classification is often referred as supervised classification, in a sense that the groups are predefined. A thorough research on relative methods and techniques can be found in [61].

Clustering refers to the task of finding groups of time series in a database such that, time series of the same group are similar to each other, whereas time series from different groups are dissimilar to each other. Although clustering constitutes classification, it is examined separately because the number of groups (clusters) to be formed is not determined a priori. For this reason, the task of clustering is also referred as unsupervised classification. A recent survey is provided by Liao [103].

The novelty detection task can be defined as follows: find all sections of a time series that contain a different behavior than the expected one. Many problems of finding

periodic patterns can be considered also as similar problems. Several alternative terms for "novelty" have been used, such as, "anomaly", "interestingness" and "surprising" to name a few. The techniques that have been proposed rely heavily on the definition of the term "novelty". One approach is to consider as novel any unexpected pattern with respect to the change in variation of data [37] [135], whereas another approach is to take into consideration the change in the frequency with which a pattern appears [89].

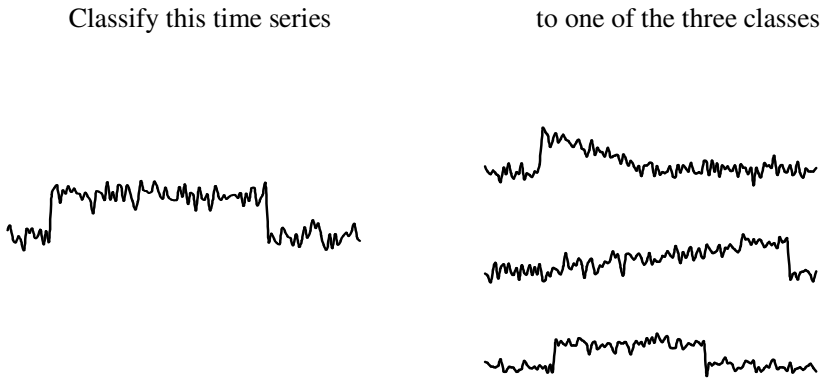


Figure 2.1 A classification example

Motif discovery refers to detecting previously unknown repeated patterns in a time series database (Figure 2.2). These discovered motifs can also be utilized in a preprocessing step of another TSDM task such as in clustering by determining initial cluster representatives. Motif discovery is a well-known task in the bioinformatics community but only recently attracted the interest of the data mining community [104].

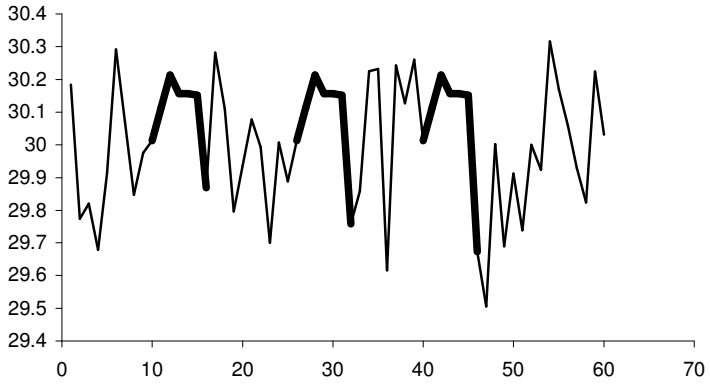


Figure 2.2 A motif example

Rule discovery aims at inferring rules from one or more time series, which describe the most possible behavior that they might present at a specific time point (or interval). In other words, these rules describe the way that several patterns correlate to each other. An example of such a rule could be: when the mean temperature in the city of Thessaloniki remains above 38^0 for five consecutive days, it is highly possible that a heavy thunder will occur in the city of Veroia after one day. These patterns can be predefined by expert knowledge or can be derived by applying another TSDM task, such as clustering [35].

2.2 Time Series Data Mining Issues

At the core of all the TSDM tasks described in the previous Section lies the notion of similarity. Two time series can be considered similar when they exhibit similar shape or pattern. The presence of noise demands allowing imprecise matches among sequences. Consequently, it is necessary to define an appropriate similarity measure, since the notion of similarity involves a degree of subjectivity that might affect the final result. Most often, this measure constitutes a distance measure, that is, it measures how dissimilar two time series are. A further discussion on similarity measures is provided in Section 2.4.

Another important issue that arises from the temporal nature of data is the intrinsic high dimensionality, which affects substantially the efficiency of data mining techniques. When the time series under consideration are univariate, the dimensionality refers to their length. For example, a sequence of length 100 can be considered as an instance of 100 attribute values. High dimensionality affects the calculation speed of similarity measure among series and, moreover, prohibits the construction of an efficient indexing structure. Suppose that there is a query for which the most similar sequence in a database is to be found. This search can be performed by either sequentially scanning the database or by indexing the database for the purpose of accelerating the search. However, when the dimensionality increases, indexing performance deteriorates and eventually reduces to sequential searching. This phenomenon is known as the dimensionality curse. It has been shown experimentally that this effect may occur for as few as 10-15 dimensions [18]. Consequently, both sequential scanning and indexing require performing a dimensionality reduction technique in order to speed-up calculations.

Indexing approaches are mostly influenced by the pioneer work of Agrawal et al. [4] and generalized by Faloutsos et al. [44]. The emerged framework from these papers, referred as GEMINI, can be summarized in the following steps [45]:

1. extract k essential features from the time series
2. map into a point in k-dimension feature space

3. organize points with off-the-shelf spatial access method ('SAM')
4. discard false alarms

The rationale behind this procedure is to reduce the dimensionality of original data by representing it in a lower dimension, analyze it in this dimension and, finally, tune the results in order to obtain the same solution with one that would have been derived, if the original data had been used in the analysis. The first and second step suggests the application of a representation scheme in order to reduce the dimensionality. However, this representation should guarantee that there will not be any false dismissals. This property, known as Lower Bounding Lemma, can be described as follows.

Suppose that t_1 and t_2 are two time series that need to be investigated for similarity and R denotes a representation scheme. Given a distance function D between two time series t_1 and t_2, R should satisfy the following property in order to guarantee no false dismissals:

$$D(R(t_1), R(t_2)) \leq D(t_1, t_2) \qquad (2.1)$$

This property states that the distance measure in the k-dimension feature space should lower bound the corresponding distance measure in the original space.

The third step is an open selection. Two of the most common indexing structures are the R-trees [58] and the vp-trees [20] [150].

Finally, the fourth step is a consequence of the fact that this approach introduces false hits in the results that need to be discarded in a post processing phase. Besides the Lower Bounding Lemma, it is important for the representation to lower bound the true distance as tightly as possible. That is, the distance measure in the k-dimension feature space should be as close as possible to the corresponding distance measure in the original space in order to reduce the number of false hits and consequently the post-processing time.

Apparently, similarity measures and representation schemes are interrelated to each other and play an important role in efficiently applying any time series data mining task.

2.3 The Pre-processing Phase

As it was mentioned in the previous Section, two time series can be considered similar when they exhibit similar shape or pattern. However, prior to define a similarity measure, a clarification of what constitutes similar shape or pattern should be provided with respect to the corresponding application. Two main distortions that should be taken into consideration are the vertical shift and the amplitude scaling. For example, the time series in Figure 2.3

seem to exhibit a similar pattern. However, the first one (solid line) is vertically shifted (upward) with respect to the second one (dotted line). In addition to that, they differ in amplitude scaling, that is in the magnitude of their fluctuations. If a specific application considers them as similar, these two distortions should be eliminated, prior to computing a similarity measure. Vertical shift can be eliminated by subtracting the mean value of the time series from each one of its individual values (Figure 2.4). Scaling can be eliminated by dividing each value of the time series by its standard deviation (Figure 2.5).

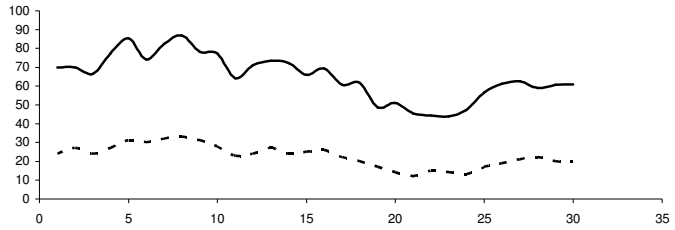


Figure 2.3 Time series of similar shape, but with different vertical shift and amplitude scaling

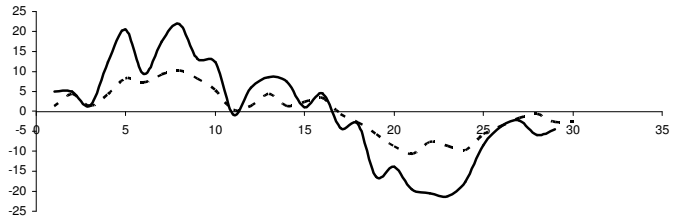


Figure 2.4 Time series adjusted to vertical shift ($x_{i,new} = x_{i,old} - \bar{X}$)

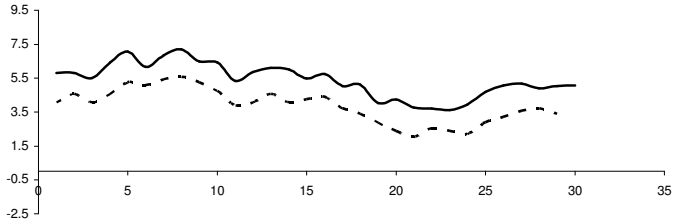


Figure 2.5 Time series adjusted to amplitude scaling ($x_{i,new} = x_{i,old} / s_X$)

According to human intuition, both of these two adjustments are required in most of the cases. In this case, the transformation that is needed is expressed in the following equation:

$$x_{i,new} = \frac{x_{i,old} - \overline{X}}{s_X}$$

where, $\overline{X}$ and s_X are the mean and the standard deviation of the time series values respectively (Figure 2.6).

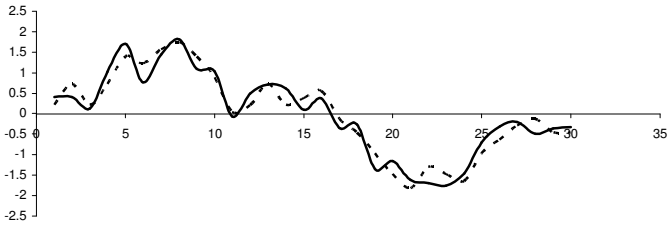


Figure 2.6 Time series adjusted to both vertical shift and amplitude scaling ($x_{i,new} = (x_{i,old} - \overline{X}) / s_X$)

2.4 Similarity Measures

2.4.1 Characteristics of Similarity Measures

The definition of novel similarity measures has been one of the most researched areas in the TSDM field. Most of the times these measures are referred as distance measures, since they constitute distance functions, which represent a way of quantifying the closeness of objects. When the objects under consideration are time series, such functions must be capable of a) efficient computation, b) managing time series of different lengths, c) handling outliers and d) representing similarity as close to human intuition as possible.

A distance function can be either metric or non-metric. A distance function d defined on a domain of objects D is called metric, when it holds all of the following properties:

- Non-negativity $\forall x, y \in D, d(x,y) \geq 0$
- Symmetry $\forall x, y \in D, d(x,y)=d(y,x)$
- Identity $\forall x, y \in D, x=y \Leftrightarrow d(x,y)=0$
- Triangle Inequality $\forall x, y, z \in D, d(x,z) \leq d(x,y)+d(y,z)$

The distance function is considered as non-metric, when any of the above properties is violated.

Within TSDM context, metric distances are important, mainly, because of the triangular inequality that allows easier indexing and more efficient calculations. For example, suppose that there are three time series X, Y and Z and a query Q for which the most similar time series is to be found among them. Let assume that the distances among X, Y and Z are known (Table 2.1).

Table 2.1 Distances between three time series

d	X	Y	Z
X		20	30
Y			10
Z			

First, the distance between Q and X is calculated ($d(Q,X)=80$) and X becomes the closest series to Q. Then, the distance between Q and Y is calculated ($d(Q,Y)=100$) and X remains the closest series to Q. According to the triangular inequality the following inequality should hold:

$$d(Q,Y) \le d(Q,Z) + d(Z,Y) \Rightarrow d(Q,Z) \ge d(Q,Y) - d(Z,Y) = 100 - 10 = 90$$

As it is shown in the above equation, the distance between Q and Z, although unknown, cannot be less than the smallest so far distance (80). Thus, it is not required to retrieve time series Z and calculate its distance from Q. In large databases, this property may result in a substantial improvement in computational efficiency.

Another approach in speeding up the calculations of a distance measure and/or dealing with the violation of the triangular inequality is lower-bounding. In Section 2.2, the Lower Bounding Lemma has been described as a desirable property of a representation scheme. Regarding distance measures, this concept refers to a function that lower bounds the original measure (Eq. 2.2) and its computation is much faster.

$$\forall X,Y: lower\_bound\_function(X,Y) \le original\_distance\_measure(X,Y) \qquad (2.2)$$

According to this approach, the lower-bounding function is calculated between two series. If this distance is larger than the "*best so far*" distance, it is not necessary to calculate the original distance, since the latter will also be larger than the "*best so far*". Hopefully,

there will be few calculations of the original distance measure, which is much slower than the lower-bounding one.

2.4.2 Euclidean Distance

In the TSDM field, most of the researchers select the Euclidean distance as the distance measure between two time series. More specifically, for two univariate time series

$$X = x_1, x_2, \cdots, x_n$$

$$Y = y_1, y_2, \cdots, y_n,$$

the Euclidean distance is defined as follows:

$$D(X,Y) = \left(\sum_{i=1}^{n} (x_i - y_i)^2 \right)^{\frac{1}{2}}.$$

In the case where the time series X and Y are multivariate, that is

$$x_i = x_{i1}, x_{i2}, \cdots, x_{im}$$

$$y_i = y_{i1}, y_{i2}, \cdots, y_{im},$$

the Euclidean distance is generalized as follows:

$$D(X,Y) = \left(\sum_{i=1}^{n} \left\| x_i - y_i \right\|^2 \right)^{\frac{1}{2}} \text{ where, } \left\| x_i - y_i \right\| = \left(\sum_{i=1}^{m} (x_{i1} - y_{i1})^2 \right)^{\frac{1}{2}}.$$

The Euclidean distance belongs to the family of the L_p norms, which defined as follows:

$$L_p(X,Y) = \left(\sum_{i=1}^{n} \left| x_i - y_i \right|^p \right)^{\frac{1}{p}}.$$

When $p = 1$, it is called the Manhattan or city-block norm, whereas when $p = 2$, it is the Euclidean norm. Yi and Faloutsos [161] present a novel and fast indexing scheme when the distance function is any of the arbitrary L_p norms ($p = 1, 2, \cdots, \infty$).

The main advantages of the Euclidean distance are the following:

- It constitutes a metric, and thus, it satisfies the triangular inequality.

- It can be efficiently computed.
- It is preserved under transformations that may be applied on the original data, such as the Discrete Fourier Transform or the Discrete Wavelet Transform.

In several applications, however, the effectiveness of this metric may deteriorate due to the following reasons.

- It is not defined on series of unequal length.
- It is not robust to outliers and noise.
- It does not take into consideration the different rate with which two series evolve through time (locally or globally).

2.4.3 Dynamic Time Warping

Many real world applications require searching for similar time series, which evolve with different rates through time. For example, two persons may read the same text with different speed or the closing prices of two stocks may evolve similarly within a time period of different duration. Contrary to the Euclidean distance (Figure 2.7), the Dynamic Time Warping (DTW) distance takes into consideration the different rate with which two time series evolve by locally compressing or decompressing them (Figure 2.8). This distance has been utilized in many applications from diverse scientific fields such as, in cardiology [152], speech recognition [126] and bioinformatics [1], to name a few.

Berndt and Clifford [16] introduced the DTW technique in the Data Mining field and presented the dynamic programming approach in calculating the corresponding distance. A description of this technique is provided below.

Suppose that there are two time series X and Y, of length m and n respectively:

$$X = x_1, x_2, \cdots, x_m$$

$$Y = y_1, y_2, \cdots, y_n$$

DTW aims at finding an alignment of the two series, which minimizes a distance function between them. Contrary to the Euclidean distance, DTW may map two elements that correspond to different time points and calculate their distance. An element of one series may map to more than one elements of another series, resulting to compression or decompression in time. Any possible alignment is called a warping path (W) and is subject to the following three constraints:

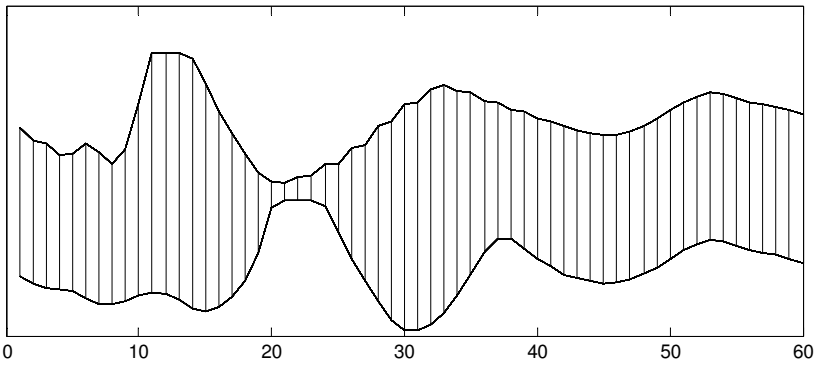


Figure 2.7 Euclidean distances between two time series

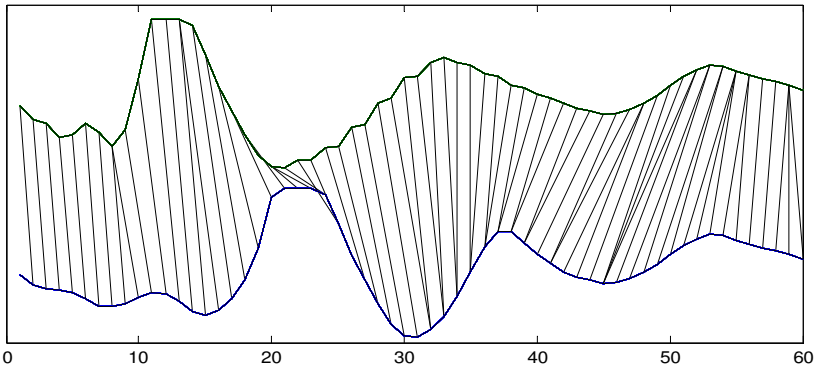


Figure 2.8 DTW distances between two time series

1. Monotonicity. The points in a warping path $W = w_1, w_2, \cdots, w_p$ are monotonically ordered with respect to time, that is, for consecutive pairs $w_k = (x_{i_k}, y_{j_k})$ and $w_{k-1} = (x_{i_{k-1}}, y_{j_{k-1}})$, the following relations should hold:

$$i_k - i_{k-1} \geq 0 \text{ and } j_k - j_{k-1} \geq 0$$

2. Continuity. The allowable steps in a warping path $W = w_1, w_2, \cdots, w_p$ are restricted to neighboring points, that is, for consecutive pairs $w_k = (x_{i_k}, y_{j_k})$ and $w_{k-1} = (x_{i_{k-1}}, y_{j_{k-1}})$, the following relations should hold:

$$i_k - i_{k-1} \leq 1 \text{ and } j_k - j_{k-1} \leq 1$$

3. Boundary conditions. The endpoints in a warping path $W = w_1, w_2, \cdots, w_p$ are forced to be

$$w_1 = (x_1, y_1) \text{ and } w_p = (x_m, y_n)$$

The first step in DTW technique is the construction of an $n \times m$ matrix where the (i^{th}, j^{th}) element of the matrix contains the cumulative distance $\gamma(i, j)$, which is defined by the following recurrence relation:

$$\gamma(i, j) = d(i, j) + min\left[\gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1)\right] \qquad (2.3)$$

where $d(x_i, y_j)$ is a metric distance. This cumulative distance is the sum of the distance between current points and the minimum of the cumulative distances of the neighboring points.

During the second step the optimal warping path is determined by moving backwards in the table and choosing the previous points with the lowest cumulative distance.

Example: Suppose that the DTW distance is required for the two time series X and Y presented in Table 2.2. For simplicity, the selected distance between two elements of the time series is their absolute difference $d(x_i, y_j) = \left|x_i - y_j\right|$.

Table 2.2 An example of two time series

X	20	40	60	70	80	60	40	20
Y	10	30	40	50	40	30	10	

The first step is to construct the table of the corresponding cumulative distances (Table 2.3), by applying the recursive relation (Eq. 2.3). The computation starts from the lower left corner and ends at the upper right corner of the table where the optimal distance lies. Then, the optimal warping path is found by moving backwards and choosing the previous points with the lowest cumulative distance. Table 2.4 shows the alignment of the two series and the distances between their corresponding points. This alignment is also presented graphically in Figure 2.9.

The main advantage of DTW is that it allows acceleration and/or deceleration of a series along the time dimension. In addition to that, it can be defined on series of unequal length. On the other hand, DTW does not constitute a metric, because it does not satisfy the triangular inequality and moreover, it involves heavy computation cost. Research has been focused on improving the efficiency by providing a lower bounding distance measure [132], imposing global constraints on the warping path [67] [131], or by indexing [92] [162].

Keogh and Ratanamahatana introduced a novel technique for the exact indexing of DTW in [90].

Table 2.3 The cumulative distances between two time series

	7	110	70	90	120	160	160	130	110
	6	100	40	60	90	110	120	100	100
	5	90	30	50	60	90	100	90	110
Y	4	70	30	30	50	80	90	100	130
	3	40	20	40	70	110	130	130	150
	2	20	20	50	90	140	170	180	190
	1	10	40	90	150	220	270	300	310
	time	1	2	3	4	5	6	7	8
					X				

Table 2.4 The alignment of two series and the distances between their corresponding points

t_X	1	2	2	3	4	5	6	7	7	8	
X	20	40	40	60	70	80	60	40	40	20	
t_Y	1	2	3	4	4	4	4	5	6	7	
Y	10	30	40	50	50	50	50	40	30	10	
Distances	10	10	0	10	20	30	10	0	10	10	110

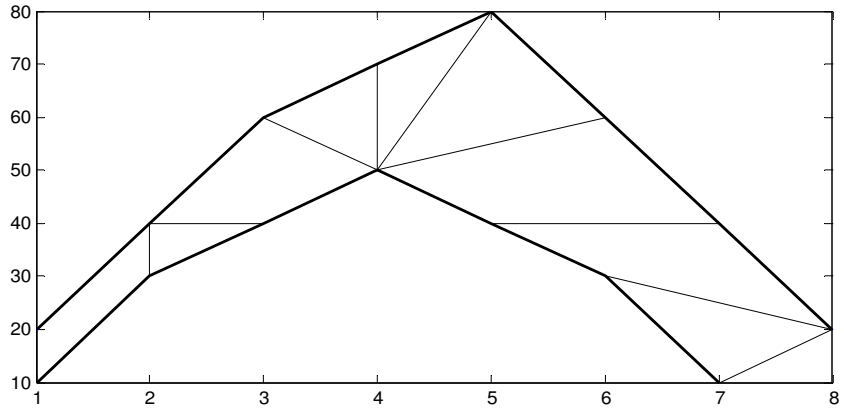


Figure 2.9 The alignment of two series and the distances between their corresponding points

2.4.4 Additional Similarity Measures

DTW and Euclidean distance are the most frequently chosen approaches in measuring similarity among time series. However, there are many other proposed measures in the literature. One important measure is the Longest Common Subsequence (LCSS), which is a variation of the Levenshtein or Edit distance [98]. Intuitively, LCSS approach *matches two sequences by allowing them to stretch, without rearranging the sequence of the elements but allowing some elements to be unmatched* [150]. The distance measure is defined as the length of the matched subsequence [35] [160]. Agrawal et al. [6] define *two sequences as similar when they have enough, non-overlapping, time-ordered pairs of subsequences that are similar*. The main advantage of this measure is that it allows elastic matching in time, as DTW does, and furthermore it is robust to outliers, since it allows elements to remain unmatched. One disadvantage is that it requires specifying the value of a parameter, which expresses the matching threshold between two elements.

Probabilistic approaches (probabilistic generative modeling) to measuring similarity have been also proposed [52] [77]. Here, similarity between two sequences S and S´ is measured by calculating the likelihood that S´ is generated from a model, which was constructed from S. Markov models have been utilized and experimented.

Furthermore, there is the expected contribution to defining similarity measures by papers that propose novel representation schemes for the purpose of dimensionality reduction, since these two tasks are interrelated to each other. For instance, some representation-specific measures are provided for in [104] [110].

Gunopoulos and Das [57] provide a thorough tutorial on similarity measures whereas Keogh and Kasetty [88] provide classification results by implementing 11 different distance measures on two datasets.

2.5 Representations

As it was described in Section 2.2, it is necessary to apply a representation scheme on data, mainly for the purpose of reducing the intrinsically high dimensionality of time series. In many cases also, the objective is to take advantage of the specific characteristics of a representation that make specific methods applicable (i.e. inducing rules, Markov models).

A representation may be considered as a transformation technique that maps a time series from the original space to a feature space, retaining the most important features. Since the distance measures are applied on the feature space, a representation scheme should guarantee that:

1. there would be no false dismissals (two similar series declared as dissimilar)

2. the number of false hits (two dissimilar series declared as similar) will be minimized
3. the dimensionality reduction will be substantial

There have been several time series representations proposed in the literature, mainly for the purpose of reducing the intrinsically high dimensionality of time series. A hierarchy of various time series representations is presented in a tree diagram in [104]. In the following subsections, a discussion of some commonly used representations is provided.

2.5.1 Discrete Fourier Transform

Discrete Fourier Transform (DFT) was one of the first representation schemes proposed within data mining context [4]. DFT transforms a time series from the time domain into the frequency domain by expressing the time series as a linear combination of trigonometric functions (i.e. sines and cosines).

The n-point DFT [120] of a signal $\left[x_t\right], t = 0,\ 1,\ ...,\ n-1$ is defined to be a sequence $\left[X_f\right]$ of n complex numbers given by:

$$X_f = \frac{1}{\sqrt{n}} \sum_{t=0}^{n-1} x(t)\, e^{\frac{-j2\pi ft}{n}} \quad ,\ f = 0,\ 1,...,\ n-1$$

where f is the frequency, X_f is the corresponding Fourier coefficient and j is the imaginary unit $j = \sqrt{-1}$. Note that X_0 is a real number.

Given the Fourier coefficients, the original series can be completely recovered by applying the Inverse Fourier Transform:

$$x_t = \frac{1}{\sqrt{n}} \sum_{f=0}^{n-1} X_f\ e^{\frac{j2\pi ft}{n}} \quad t = 0,\ 1,...,\ n-1$$

The main importance of DFT is that the reconstruction of the original series based only on a few coefficients can be very accurate in many cases. This is the fact that makes DFT an important dimensionality reduction technique. As an example, for a time series of length 64, DFT generates 64 complex numbers. If 4 of them are considered to be important and retained, the transformed series is expressed by 8 numbers (the corresponding real and imaginary parts). This would result in a dimensionality reduction of 87.5%.

Another important property of DFT is that it preserves the Euclidean distance in the frequency domain according to Parseval's Theorem.

Theorem 2.1 (Parseval): If X is the Discrete Fourier Transform of a sequence x, then the following holds:

$$\sum_{t=0}^{n-1}\left|x_t\right|^2 = \sum_{f=0}^{n-1}\left|X_f\right|^2$$

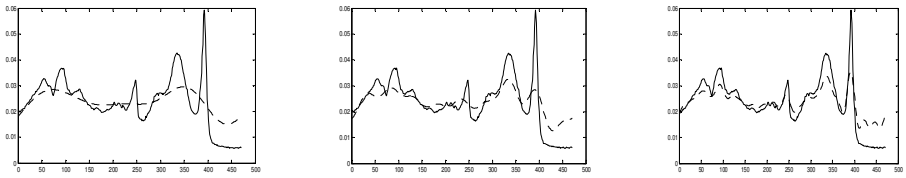
$$\sum_{t=0}^{n-1}\left|x_t - y_t\right|^2 = \sum_{f=0}^{n-1}\left|X_f - Y_f\right|^2$$

This means that the Euclidean distance between two series in the time domain is the same as their Euclidean distance in the frequency domain. Thus, by retaining only a few (k) coefficients, the Euclidean distance in the frequency domain lower bounds the Euclidean distance in the time domain (Eq. 2.4).

$$\sum_{t=0}^{n-1}\left|x_t - y_t\right|^2 \geq \sum_{f=0}^{k}\left|X_f - Y_f\right|^2 \quad \text{where } k < n \tag{2.4}$$

This fact implies that the lower bounding Lemma (Section 2.2) is satisfied and thus this representation guarantees that there will be no false dismissals. The choice of which k coefficients to retain relies on the shape of the time series (random walk, periodicity etc.). Most of the researchers [4] [44] [127] choose the first k coefficients, whereas others choose the largest k coefficients [150].

Finally, an example of how DFT representation may reconstruct a time series is presented in Figure 2.10. The term reconstruction refers to the case where a time series is approximated by a representation scheme.



5 coefficients 10 coefficients 20 coefficients

Figure 2.10 An example of how DFT representation reconstructs a time series for various number of retained coefficients

2.5.2 Singular Value Decomposition

The Singular Value Decomposition (SVD) is a powerful technique of Linear Algebra, and it has been used in many diverse fields, such as in statistics [71], in text retrieval [17], and in gene expression analysis [153], to name a few.

The intuition behind SVD is provided with the following example. Suppose that there is a 5×2 matrix X.

$$X = \begin{bmatrix} 4 & 1.5 \\ 3 & 2 \\ 3.5 & 3.5 \\ 5 & 4.5 \\ 5.5 & 3 \end{bmatrix}$$

Each row can be represented graphically as a point in a 2-dimensional space (Figure 2. 11a). SVD transforms X by rotating the original axes X_1 and X_2 and deriving a new set of axes Y_1 and Y_2, which are called components (Figure 2. 11b). The rotation is performed in such a manner that the first axis (Y_1) will account for the largest portion of variance present in the data, while the second one (Y_2) will account for the largest portion of the remaining variance subject to being orthogonal to the first one.

The SVD theorem states that a $m \times n$ real matrix X can be expressed in the following form:

$$X_{m \times n} = U_{m \times m} \cdot S_{m \times n} \cdot V^T_{n \times n}$$

where S is a diagonal matrix that contains the singular values; the matrices U and V are orthonormal and their columns are the left and right singular vectors respectively.

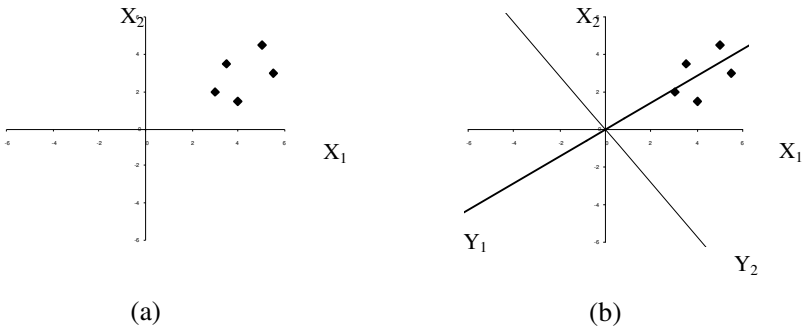


Figure 2. 11 The graphical representation of a time series (a) and the corresponding SVD rotation of the original axes (b)

Without loss of generality, it is assumed that the singular values are sorted in non-increasing order. Geometrically, these values provide the importance of each component (axis) with respect to the amount of the variation that explains. The square root of a singular

value (i.e the eigenvalue), is proportional to the variance that the corresponding axis explains. The matrix V provides information about the direction of the components in correspondence to S matrix. More specifically, the first column of V provides the direction of the first most important component, which corresponds to the greatest singular value. Accordingly, the second column provides the direction of the second most important component and so on. The matrix $U \cdot S$ provides the coordinates on the new set of axes (components).

Within TSDM context, the matrix X consists of all the time series data. Hereafter, it is assumed that each row corresponds to a time series and each column corresponds to a specific time instance. SVD is applied on the whole dataset and the transformed values $U \cdot S$ are provided by multiplying X by V:

$$X_{m \times n} \cdot V_{n \times n} = U_{m \times m} \cdot S_{m \times n}$$

Until now, there has not been any dimensionality reduction, since the transformed values form a matrix of the same dimensions as X ($m \times n$). In order to achieve dimensionality reduction, the k most important components should be retained ($k \ll n$) and the original data should be projected on these axes. Since SVD determines those axes that describe the variation in data in an optimal way, the approximation is expected to be adequate for further analysis. The SVD representation will result in a $m \times k$ matrix:

$$\mathrm{X}_{\mathrm{m \times n}} \cdot \mathrm{V}_{\mathrm{m \times k}}$$

Following the previous example, the matrix X can be decomposed as follows:

$$\begin{bmatrix} 4 & 1.5 \\ 3 & 2 \\ 3.5 & 3.5 \\ 5 & 4.5 \\ 5.5 & 3 \end{bmatrix} = \begin{bmatrix} -0.3528 & 0.6225 & 0.1798 & 0.0659 & -0.6719 \\ -0.3077 & 0.0567 & -0.6317 & -0.7088 & -0.0245 \\ -0.4165 & -0.4808 & 0.6380 & -0.4229 & -0.0976 \\ -0.5703 & -0.4524 & -0.3738 & 0.5503 & -0.1658 \\ -0.5312 & 0.4164 & 0.1477 & 0.1075 & 0.7148 \end{bmatrix} \cdot$$

$$\cdot \begin{bmatrix} 11.7139 & 0 \\ 0 & 1.7418 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} -0.8165 & 0.5773 \\ -0.5773 & -0.8165 \end{bmatrix}^T$$

In order to reduce the dimensionality of the matrix X, only the first ($k=1$) component is retained out of two ($n=2$), and the corresponding SVD representation is obtained as follows:

$$\mathrm{X}_{5\times 2}\cdot \mathrm{V}_{2\times 1}=\begin{bmatrix}4 & 1.5\\ 3 & 2\\ 3.5 & 3.5\\ 5 & 4.5\\ 5.5 & 3\end{bmatrix}\cdot\begin{bmatrix}-0.8165\\ -0.5773\end{bmatrix}=\begin{bmatrix}-4.1321\\ -3.6042\\ -4.8784\\ -6.6805\\ -6.2228\end{bmatrix}$$

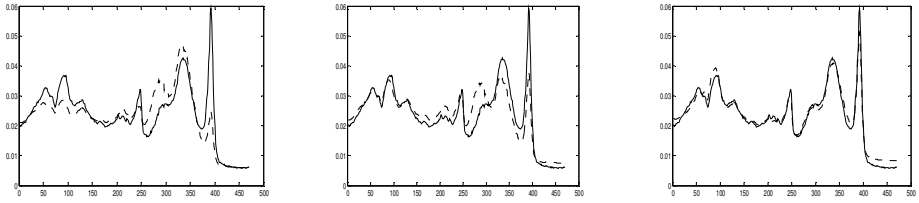
The original matrix X can be reconstructed approximately by retaining only the first component as in Eq. 2.5.

Similarly to DFT representation, the lower bounding Lemma (Section 2.2) is satisfied and this representation guarantees that there will be no false dismissals. Moreover, SVD is the linear transform that minimizes the reconstruction error [77] [127]. Graphical examples of reconstructions for various values of k are shown in Figure 2.12.

On the other hand, the main disadvantage of SVD is that it is data dependent, meaning that it is required to have historical data, since SVD is applied on whole datasets and not on individual time series. Practically, each time a new series arrives, SVD is applied on the whole dataset augmented by this series. Another disadvantage is that it is computationally expensive with complexity of $O(mn^2)$ in time. It is worth noting however that several enhancements of SVD-based representation have been proposed for an efficient similarity search in large databases [77] [94].

$$\begin{aligned}\tilde{X}_{5\times 2} &= U_{5\times 5}\cdot S_{5\times 1}\cdot V_{2\times 1}^{T}\\ &=\begin{bmatrix}-0.3528 & 0.6225 & 0.1798 & 0.0659 & -0.6719\\ -0.3077 & 0.0567 & -0.6317 & -0.7088 & -0.0245\\ -0.4165 & -0.4808 & 0.6380 & -0.4229 & -0.0976\\ -0.5703 & -0.4524 & -0.3738 & 0.5503 & -0.1658\\ -0.5312 & 0.4164 & 0.1477 & 0.1075 & 0.7148\end{bmatrix}\cdot\begin{bmatrix}11.7139\\ 0\\ 0\\ 0\\ 0\end{bmatrix}\cdot\begin{bmatrix}-0.8165\\ -0.5773\end{bmatrix}^{T}\\ &=\begin{bmatrix}3.3741 & 2.3854\\ 2.9430 & 2.0806\\ 3.9834 & 2.8162\\ 5.4549 & 3.8565\\ 5.0813 & 3.5923\end{bmatrix}\end{aligned} \tag{2.5}$$

The final issue to be discussed relates to the number (k) of components to be retained. There are several criteria for determining the value of k , such as the scree graph or the cumulative percentage of total variation [71]. According to the latter criterion, one could select that value for k , for which the first k components retain more than a predetermined proportion (e.g. 90%) of the total variation present in the original data.



1 component 2 components 4 components

Figure 2.12 An example of how SVD representation reconstructs a time series for various number of retained components

2.5.3 Piecewise Aggregate Approximation

The Piecewise Aggregate Approximation (PAA) is a very simple dimensionality reduction technique that was proposed independently by Keogh et al. [77] and Yi and Faloutsos [161]. The PAA technique segments a time series of length n into k consecutive sections of equal-width and calculates the corresponding mean for each one. The series of these means is the new representation of the original data.

More formally, a time series $X = x_1, x_2, \cdots, x_n$ can be represented by a series $\tilde{X} = \overline{x}_1, \overline{x}_2, \cdots, \overline{x}_k$, where:

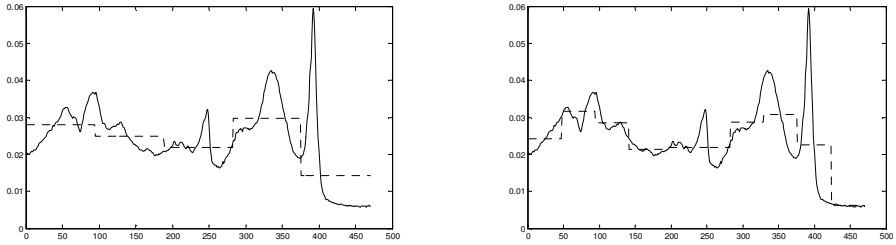
$$\overline{x}_i = \frac{\sum_{j=(i-1)\frac{n}{k}+1}^{i\frac{n}{k}} x_j}{\frac{n}{k}}$$

The quantity n / k expresses the number of points that each section consists and it is assumed that it is an integer.

PAA is simple, fast to calculate and it has been shown empirically that it is as efficient as other approaches. Moreover, it can handle time series of different lengths [77]. Regarding the lower bounding Lemma (Section 2.2), the Euclidean distance it is provably lower-bounded by the following distance in the reduced space:

$$D(\tilde{X},\tilde{Y}) = \sqrt{\frac{n}{k}}\sqrt{\sum_{i=1}^{k}\left(\bar{x}_i - \bar{y}_i\right)^2}$$

Similarly to the previous representations, PAA involves the determination of the parameter k. Since it is desirable to have a representation that copes with the dimensionality curse, this parameter should not take a value much greater than 15 (Section 2.2). When $k=1$, the time series is mapped to its mean value, whereas when $k=n$, it remains untransformed. An example of PAA representation for $k=5$ and $k=10$ is shown in Figure 2.13.



(a) k=5 (b) k=10

Figure 2.13 An example of how PAA representation reconstructs a time series for various number of sections

2.5.4 A Feature Based Representation

A different approach in representation is the extraction of global features from the time series. In this case, a time series is represented by a feature vector. For example, Alcock and Manolopoulos [7] proposed the use of statistical features in similarity search and they experimented with first-order and second-order features in order to determine their corresponding effectiveness. In a continuing work, Nanopoulos et al. [117] investigate further the use of statistical features in time series classification in conjunction with a multi-layer perceptron neural network. In their work, the original time series $X = x_1, x_2, \cdots, x_n$ of length n is represented by a feature vector of length 8. The first 4 features are the mean value (μ), the standard deviation (σ), the skewness (SKEW) and the kurtosis (KURT), which are defined in the following equations:

$$\mu = \frac{\sum_{i=1}^{n} x_i}{n}$$

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}\left(x_i - \mu\right)^2}{n}}$$

$$SKEW = \frac{\sum_{i=1}^{n}\left(x_i - \mu\right)^3}{n\sigma^3}$$

$$KURT = \frac{\sum_{i=1}^{n}\left(x_i - \mu\right)^4}{n\sigma^4} - 3$$

The next 4 features are extracted by calculating the same statistical measures on the transformed time series:

$$x'_t = (x_{t+D} - x_t), \quad 1 \le t \le n - D$$

where D is a user-defined parameter.

These are statistical measures, which attempt to describe a time series with respect to the central tendency and dispersion of its values along with the shape of their distribution.

This approach is simple, fast to calculate and capable of dealing with noise and series of different length. However, the fact that the extracted features are global may be a drawback in specific datasets. In addition to that, it has been tested only in conjunction with a multi-layer perceptron neural network and in the context of control chart pattern recognition. In Chapter 4, this feature-based representation will be further investigated and tested in several diverse datasets.

2.5.5 Symbolic Aggregate Approximation

Lin et al. [104] proposed a representation of time series, called Symbolic Aggregate approXimation (SAX) that achieves dimensionality reduction and handles streaming data efficiently. The symbolic nature of the representation allows the application of methods and data structures appropriate for discrete data, such as Decision Trees. Last but not least, a similarity measure can be defined that lower bounds a distance measure defined on the original time series. The procedure that SAX method proposes is as follows. First, the time series is normalized to have a mean of zero and a standard deviation of one. Second, the time series is transformed further by applying Piecewise Aggregate Approximation (Section 2.5.3). Third, taking advantage of the fact that the transformed series follows the normal

probability distribution, each element is mapped to a symbol using the properties of this distribution. The user must assign a value in a parameter a, which defines the alphabet size (number of symbols to be used). Then, the area under the normal curve is divided into a areas of equal size (meaning that the corresponding probabilities will be equal for each symbol) and each one of them is assigned to a symbol [39]. Finally, an element of the series, which falls into an interval that corresponds to a specific area, is mapped to the area's symbol (Figure 2.14).

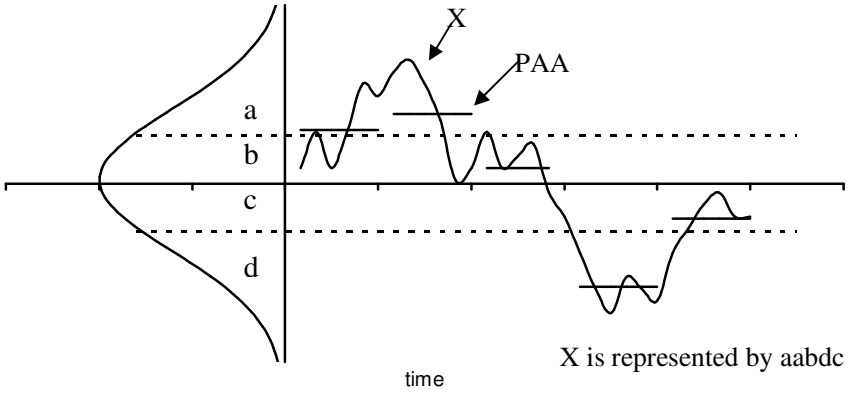


Figure 2.14 The SAX representation of a time series for an alphabet size equal to 4

2.5.6 Additional Representations

There is a wealth of representation schemes proposed in the literature, the detailed description of which is beyond the scope of this thesis. In this section, a brief description of some additional representations, which are either frequently used or recently proposed, is provided.

Discrete Wavelet Transform (DWT) transforms a time series into the time/frequency domain by decomposing it into a series of wavelet basis functions. The linear combination of these functions constitutes the original series and the corresponding (wavelet) coefficients constitute the DWT representation. A significant dimensionality reduction is achieved by retaining only the first few coefficients, which hold the most significant information of the original series. Contrary to Discrete Fourier Transform (DFT), DWT determines not only the important frequencies of a time series, but also the time locations where they occur [27] [158]. This property allows the investigation of long segments with low resolution and short segments with high resolution. Thus, DWT analysis provides multi-resolution results, meaning that it provides all the analyses from coarse to fine resolution. This procedure is similar to cartography, where choosing a large-scale, the resolution is high and vice versa.

Although there are lot wavelet transformations, such as Daubechies, Coiflets etc, Haar wavelet is the most widely used in the data-mining community, due to its simplicity and effectiveness. A comparative study on different wavelets can be found in [123] and applications within Data Mining context in [102].

Another approach is to represent a time series as a sequence of real-valued functions [136]. In the simplest case, called Piecewise Linear Approximation (PLA), a time series is approximated by a sequence of linear segments [101]. According to this approach, there are three basic decisions to be made; the number of segments, the function that generates the linear segments and the corresponding algorithm. The number of segments may be user-specified or algorithmically determined with respect to minimization of some error function. Linear segments may be generated by either interpolation or linear regression. Finally, there is a wealth of segmentation algorithms, which can be grouped into three approaches, namely the Sliding Window, the Top-Down and the Bottom-Up. An extensive review of segmentation algorithms, in the context of Time Series Data Mining, can be found in [88].

Chakrabarti et. al [26] proposed Adaptive Piecewise Constant Approximation (APCA), which constitutes a significant variation of Piecewise Aggregate Approximation (PAA). As it was described in Section 2.5.3, the PAA technique segments a time series of length n into k consecutive sections of equal-width and calculates the corresponding mean for each one. The variation in APCA is that these sections are allowed to have arbitrary lengths. The transformed series consists of the mean values and the lengths of the corresponding sections. In comparison with PAA, APCA has the advantage of representing a long section of low activity by one segment and a short section of high activity with more than one segment. On the other hand, PAA has the advantage of representing a series by twice as many segments, since it retains only the mean value for each segment.

A broad class of time series representations involves symbolic representations. Persist [115] is a new unsupervised discretization method of time series, which results into a sequence with symbols that retain their temporal aspect. This method requires that the time series does not change behavior fast and does not contain a long-term trend. The basic idea of this approach is that a time series is a sequence of states generated by an underlying process. Persist is based on the Kullback_Leibler divergence between the marginal and the self-transition probability distributions of the states (the discretization symbols) in order to discover these states. In [9] the affects of clipping original data on the clustering of time series is assessed. Each point of a series is mapped to 1 when it is above the population mean and to 0 when it is below. This representation is called clipping and has many advantages especially when the original series is long enough. It achieves adequate accuracy in

clustering, it efficiently handles outliers and it provides the ability to employ algorithms developed for discrete or categorical data. The authors evaluate the effects of this representation on clustering. Another novel dimensionality reduction technique, called Piecewise Vector Quantized Approximation (PVQA), is introduced in [110] and extended further in [110]. This technique is based on vector quantization that partitions each series into segments of equal length and uses vector quantization to represent each segment by the closest codeword from a codebook. The original time series is transformed to a lower dimensionality series of symbols. This approach requires a training phase in order to construct the codebook (the Generalized Lloyd Algorithm is applied), a data-encoding scheme and a distance measure.

Kontaki et al. [93] present a method of continuously classifying streaming time series based on their trends. The first step of this method includes the application of Piecewise Linear Approximation (PLA) on smoothed time series under the sliding window paradigm and results in a sequence of pairs (t, trend), where t denotes the first time point of the corresponding segment and trend denotes the current trend of it (UP or DOWN). An incremental computation of this representation is presented in order to support the application on streaming data. The classification task is performed based on the identified trends and a set of a priori known classifiers. Fu et al. [49] propose an application-oriented representation scheme with respect to financial time series. They take advantage of the specific head-and –shoulder pattern of interest within stock analysis domain and identify the perceptually important points (PIPs). These critical points of higher importance are identified by the vertical distance between the points of interest to their current pair of adjacent PIPs. According to the proposed representation, the points of the original time series are reordered in a way that a data point identified as perceptually important in an earlier stage is considered being more important than those points identified afterwards and are stored in a specialized binary tree. Authors provide comparative experiments on various methods of updating where new values arrive in a streaming fashion.

A novel representation scheme for the purpose of efficiently identifying similar time series is introduced in [144]. Their technique is based on the assumption that similar time series correspond to centroids that are close to each other, after properly pre-processing raw data in order to deal with different shifts and scales. The determination of the centroids is based on the calculation of the second moments as they provide weights to the locations of the measured parameter along the time axis.

Finally, there are several feature-based representations proposed, other than the one discussed in the previous section, such as, the ARIMA models [74] and the Autocorrelation Functions [154].

2.6 Summary

As it was stated in Section 2.2, Time Series Data Mining tasks rely heavily on the concept of similarity among time series. In any relevant application, a thorough definition of similarity should be provided, since the subsequent decisions may depend significantly on it. More specifically, there are four decisions to be made regarding pre-processing, similarity measure, representation and indexing. These issues are strongly interrelated to each other and thus, it is of high importance to have a concrete view of what similarity implies. In addition to that, the time series dataset should be thoroughly described with respect to any features that might be present (e.g. periodicities, noise, peaks etc). The choice of the representation scheme is mainly based on its capability in capturing the most important features present in a time series. Also, this choice should be made in conjunction with the choice of the similarity measure, since it is highly desirable to lower-bound the true distance between two series. In Data Mining applications, all the above decisions should form a framework that allows indexing.

CHAPTER 3

A TSDM Approach to Control Chart Pattern Recognition

3.1 Introduction

The need for constantly monitoring and measuring the level of quality in manufacturing industries led to the development and application of specific procedures, one of which is the Statistical Process Control (SPC). SPC constitutes the basic method of quality control not only in manufacturing industries but also in services. Contrary to the traditional methods of quality testing of the final product, SPC focuses in the whole process, from the raw materials to the final product. The main goals of SPC are: (a) monitoring (usually by sampling) the main parameters of the process, (b) detecting process deviation, and, (c) diagnosing and taking corrective action [114]. Although SPC was technically and theoretically developed in 1931 [137], it was adopted from the industries worldwide during the last two decades being one of the most effective tools in Total Quality Management (TQM) [40] [41].

Control chart, which is the graphical representation of a time series, is the most important tool of SPC for the purpose of monitoring a process. The interpretation of this chart is based not only on the individual values but also on its shape, which indicates the current status of a process. According to the shape of a control chart, the process is classified into one of several categories, which indicate the probable cause of a deviation (if exists) of this process from its normal status. This procedure is often called Control Chart Pattern Recognition (CCPR). Although the traditional SPC methods have been proven to be efficient, advances in technology of generating and storing huge amount of data emerged the need of developing new techniques and tools capable of intelligent process and analysis in real time. In this chapter, we investigate the possible contribution of Time Series Data Mining (TSDM) techniques in CCPR. Although one of the basic techniques is classification and the control charts are constructed by time series, there is not much research in the possible contribution of TSDM to control chart pattern recognition. We discuss the general issues of a Data Mining approach within the SPC context namely, the choice of representation, similarity measures and techniques. The technique of Decision Trees is selected as the classification method to be applied for its simplicity and ability to produce

understandable rules. Experiments are conducted on synthetic datasets taking into consideration many parameters such as noise, magnitude of deviations and presence of in-control data.

The main contributions of this chapter are the following:

- the introduction of a TSDM approach in CCPR that is based on Decision Trees,
- the investigation of the various aspects of TSDM with respect to CCPR, and
- the experimental evaluation of SAX representation in conjunction with Decision Trees within CCPR context.

The rest of this chapter is divided into five further sections. In Section 3.2, a brief review of SPC is provided along with the related work in CCPR. Section 3.3 discusses the basic components of TSDM methods with respect to CCPR and describes the proposed approach. In Section 3.4, the settings of the experiments are described, whereas in Section 3.5 the corresponding results are presented and discussed. Finally, conclusions are provided in Section 3.6.

3.2 Control Chart Pattern Recognition

Control Chart Pattern Recognition (CCPR) is performed within Statistical Process Control (SPC) for the purpose of identifying possible causes that result in deviation of a process from its normal operating conditions. A brief review of Control Charts is provided in Section 3.2.1, whereas related work is presented in Section 3.2.2.

3.2.1 Review of Control Charts

Although there are several tools that should be utilized together, control chart is the most powerful tool in order to achieve SPC goals. The three fundamental uses of a control chart are: (a) reduction of process variability, (b) monitoring and surveillance of a process and (c) estimation of product or process parameters. Basically, a control chart is the graphical representation of a time series. An important (in quality context) variable of a process is being measured continuously for a specific time interval and the corresponding measurements are being plotted in a diagram over time (Figure 3.1). The unique characteristic of a control chart is that at the same diagram a lower control limit (LCL) and an upper control limit (UCL) are being drawn. The calculation of these limits is based on statistical theory. As long as the points (measurements) fall randomly within these limits the process is said to be in statistical control. The variation is due only to chance causes (common causes). When one or more points fall beyond the control limits or the plotted points exhibit some non-random pattern of behavior, the process is said to be out of control.

In this case, variation is due also to the presence of assignable causes (special causes). Western Electric [156] determined fifteen different patterns that may be encountered in practice. Some of them are shown in Figure 3.2. The traditional methods of SPC have been proven to be very effective in detecting out of control conditions, especially when one or more points fall beyond the control limits. Problems arise when non-random (unnatural) patterns appear, since this involves the ability or experience of an operator to identify and classify them into the appropriate class of assignable causes. The SPC approach to this problem was the development of sensitizing rules (run rules or zone tests) in order to assist operators in identifying these patterns [118] [119]. However, the more rules are applied to a chart, the more complicated the decision process becomes and the number of false alarms may be increased resulting in a less effective SPC procedure.

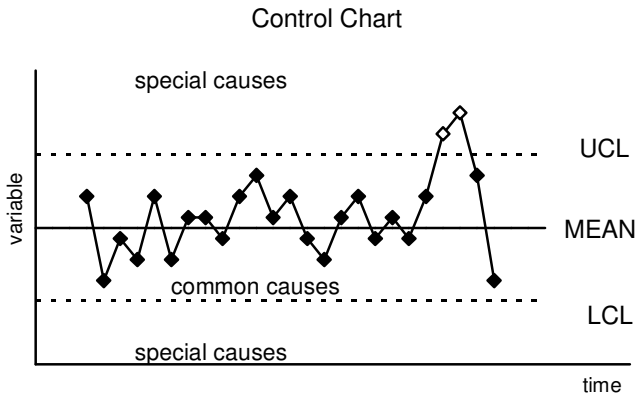


Figure 3.1 An example of Control Chart

3.2.2 Related Work

Artificial intelligence (AI) techniques have been widely investigated in order to assist in CCPR and further automate the process control procedure. In particular, Expert Systems have been developed for the purpose of automating the interpretation and diagnosis of control chart information. The main advantages are the ease of development and transparent reasoning. Several Expert Systems have been proposed [43] [63] [140] that primarily perform the tasks of selecting, constructing and interpreting an appropriate control chart, and diagnosing the possible assignable causes (if present). For interpretation and diagnosis, sensitizing rules are incorporated to these systems resulting into the problem of false alarms (misclassification) as stated in the previous section.

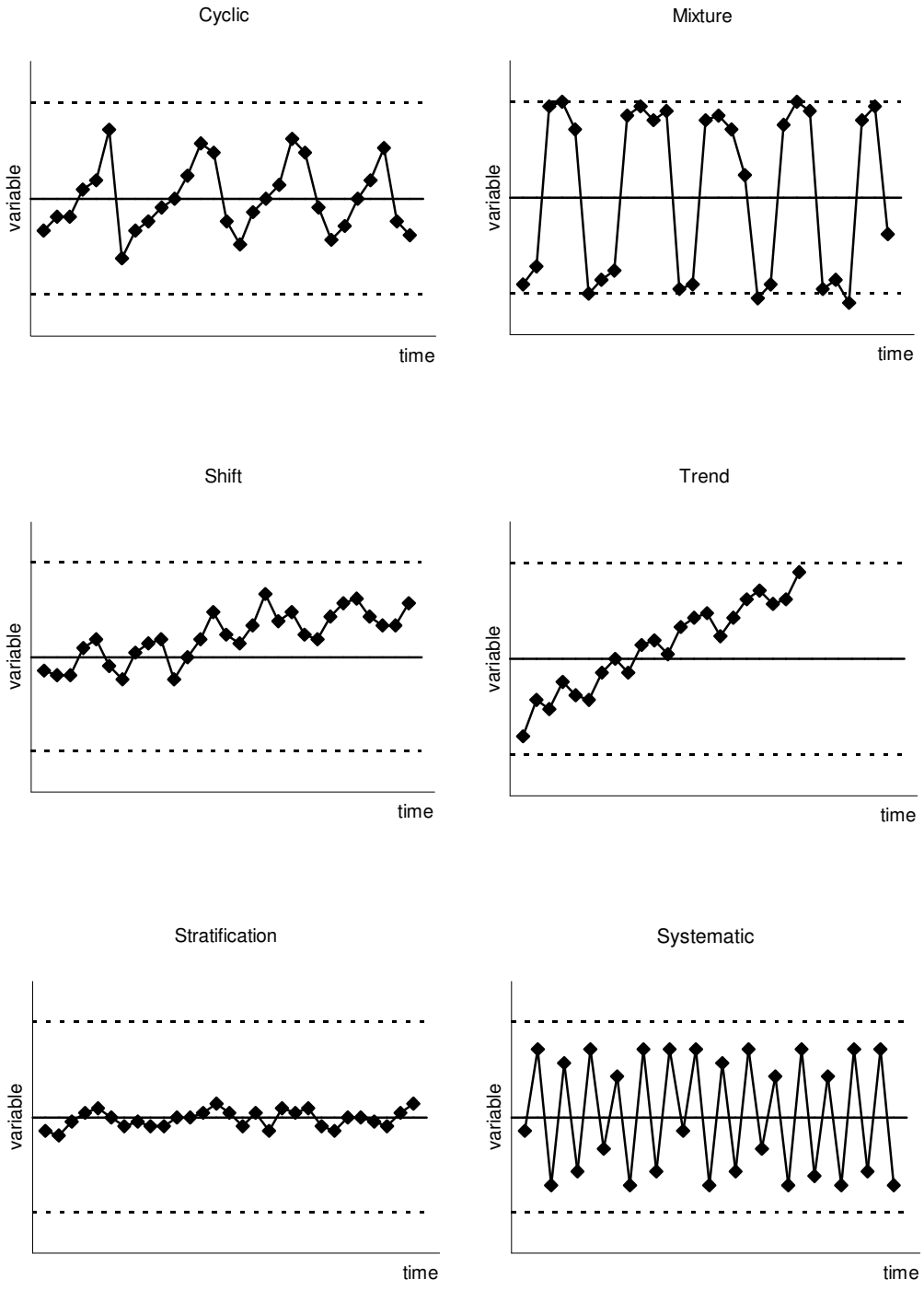


Figure 3.2 Six Control Charts that indicate out-of-control patterns

Neural Networks (NN) have also been investigated within SPC context, mainly for the purpose of identifying unnatural patterns in control charts. The main tasks that Neural Networks can accomplish are classification, clustering and prediction [129]. Since the problem of control chart pattern recognition can be assumed as a classification problem, NN technology has been investigated for its usefulness in improving automation in SPC systems. NNs rely more on historic data produced by a process than on prior human knowledge about the distribution of this data. These models mimic the operation of the neural connections in human brain by learning from data and developing an ability to generalize, similarly to the way humans learn from experience. An artificial neural network consists of a number of interconnected processing units called neurons, since the basic modeling is based on biological neurons. Neurons are organized into layers: input, hidden (one or more layers) and output. Each unit receives a number of input signals and combines them into a single output value by using an activation function. Each input signal is assigned a relative weight, and the weighed sum of these input signals constitute the effective input to the processing unit. From this result, an output value is calculated and is transferred to the output of the unit. Depending on the interconnection architecture, the output signal may be sent to other processing units as input signal. The network starts learning (adjusting the weights) by receiving training data with known outputs and by following a specific learning strategy.

There has been a lot of research in control chart pattern recognition using neural networks [32] [65] [65]. Different models have been implemented and tested in the context of pattern recognition in SPC. The procedure of the construction of these models consists of common steps and decisions to be made, namely, the representation of input data, the selection of the architecture and topology of the network, the choice of the activation function, the learning and training strategies. These decisions often interrelate to each other.

The first step is the representation and transformation of input data. In control chart pattern recognition, there are many different approaches such as feeding the network with raw data [124] or with features extracting from data [117]. Several transformations also may be performed such as standardization, zoning, scaling and using continuous (as opposed to binary) representation [164]. The second step is the selection of the NN architecture and topology with respect to the number of layers, the number of nodes in each layer and the corresponding interconnections. Some popular NN architectures are: Multi-Layer Perceptron (MLP), Learning Vector Quantization (LVQ), Radial Basis Function (RBF), Adaptive Resonance Theory (ART) and Kohonen Self-Organizing Maps (SOM). In control chart pattern recognition literature, MLP feed forward fully connected networks is the most frequently selected model. The third step is the selection of the learning rule. There are many

different rules but the Back Propagation Network (BP) rule is preferred in order to train a network. The fourth step is the choice of the activation function, more specifically that part that corresponds to the transfer function. Although there are several types of transfer functions, the most commonly preferred function is the Sigmoid. The fifth step is the selection of the training strategies with respect to the window size under investigation, the size of the training dataset and the presentation order of the training patterns. The evaluation of the performance is mainly assessed by the classification accuracy, the average run length (ARL), and, the Type I and II errors. An analytical review of Neural Networks in control chart pattern recognition can be found in [164].

3.3 Data Mining & Control Chart Pattern Recognition

The Data Mining field involves techniques and algorithms capable of efficiently extracting patterns from large databases that can potentially constitute knowledge. According to Fayyad et al. [46], the basic data mining tasks are: classification, regression, clustering, summarization, dependency modeling and change and deviation detection. The case of control chart pattern recognition can be considered as a time series classification problem. As it was mentioned before, one possibility of a process being out of control is when the plotted points exhibit some non-random pattern of behavior, specifically, one of the fifteen patterns determined in [156]. Thus, given a set of pre-defined classes (possible patterns), the goal is to assign a time series into the most appropriate class. In this chapter, we propose the technique of Decision Trees [38] for classification of control charts. In Section 3.3.1, a brief review on Decision Trees is provided, whereas Section 3.3.2 presents our proposed approach.

3.3.1 A Review of Decision Trees

Decision Trees (DT) are constructed for the purpose of classification [125] [116]. An internal node is mapped to an attribute and denotes a test on this attribute, a branch is mapped to an outcome of the test and a leaf node is mapped to a class (Fig.3.3). The attribute values of an unknown instance are tested against the DT in a top-down manner, and when a leaf is reached the instance is classified according to the class that is mapped to this leaf. The tree is constructed on a training set for which the class of each instance is known.

The basic principle of DT is to partition the search space into rectangular regions and classifying an instance based on the region into which it falls. The generation of a DT is developed in two phases: tree construction and tree pruning.

The tree construction phase constitutes a recursive procedure. First, we select an attribute and assign it to the root node. For each possible value of this attribute, we construct a single branch. Then, the process is repeated recursively for each branch, using only those instances that actually traverse the branch. When all instances for a given node belong to the same class the construction of that part of the tree is terminated. This node becomes a leaf and it is assigned with the corresponding class. When there are no remaining attributes for further partitioning, majority voting is employed for assigning a class to the leaf. Regarding this procedure, there are two issues that need to be considered. The first one refers to the type of attribute that is tested in a node. When the attribute is nominal, the number of children is usually the number of all possible values of the attribute. However, when the attribute is numeric, it is usually discretized prior to tree construction [4] or its value is tested against some threshold. Moreover, a numeric attribute may appear in more than one node (Fig. 3.3). The second issue refers to the decision of which attribute to assign to a specific node [112]. Most often the selection of an attribute is based on a measure that is called *information gain*. The concept of *information* is quantified by calculating the average entropy of each attribute at a given node (Eq. 3.1). It represents the expected amount of information that would be needed in order to specify the class of a new instance.

$$entropy(Y,N) = -\frac{y}{y+n}log_2\frac{y}{y+n} - \frac{n}{y+n}log_2\frac{n}{y+n} \qquad (3.1)$$

where Y and N denote the existing classes, y and n denote the number of their elements respectively at a given node.

The fact that the tree is constructed on a training set of instances may introduce the problem of overfitting, meaning that the constructed tree may classify almost perfectly the training set but inadequately any other independently chosen test set. One common approach in handling overfitting is to prune the tree [42] by removing subtrees under some criterion. In this case, a subtree is replaced by a leaf node. This replacement will certainly decrease the accuracy on the training set but it may increases the accuracy on a test set.

Finally, each path of a Decision Tree corresponds to a classification rule. An example of a rule that is derived from the DT shown in Figure 3.3 is the following:

If the age of a customer is less than 50 years and she is married and her age is less than 30 years, then she is considered as a "good" customer.

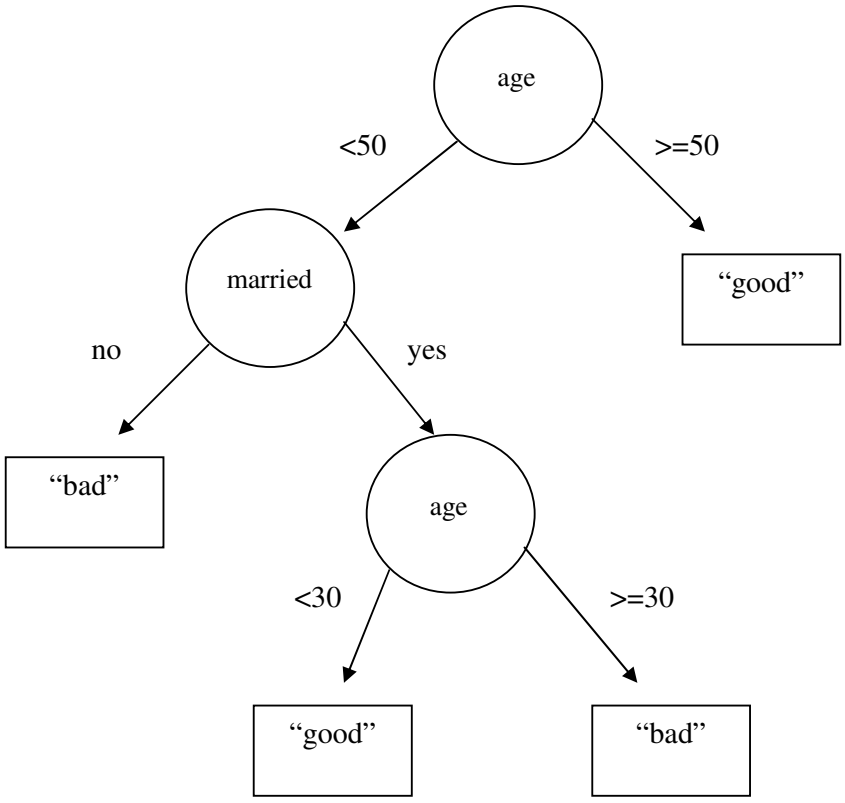


Figure 3.3 An example of a Decision Tree with 2 attributes (age and marital status) and two classes ("good" and "bad" customer).

3.3.2 The Proposed Approach

A TSDM approach should involve the selection of (a) a representation scheme, (b) a similarity measure, and, (c) a data mining method for classification. These three decisions interrelate to each other and, thus, they should be considered concurrently. However, it is not our intension to identify the "best" possible classification procedure, but to indicate the potential usefulness of TSDM methods in CCPR.

Within Statistical Process Control context, an important issue that should be taken into consideration while making the above decisions is how efficiently a classification procedure can be applied to an on-line environment. Another issue is the required sensitivity of the SPC system, in other words, the magnitude of deviations that ought to be detected. Finally, it would be desirable if the selected classification procedure was flexible to changes in the configuration of a process, with fewer parameters to be determined by the developer and/or user.

Besides Neural Networks that are described in the previous section, there are many other classification methods available such as tree models, linear discriminants, logistic

discriminant analysis, nearest neighbor methods and the naïve Bayes model, to name a few. In this chapter, we propose Decision Trees (DT) in order to classify control charts, since it is considered as the most simple classification method. The main advantages of DTs are their ability to produce understandable rules, their low computation requirements, and their ability to handle both continuous and categorical variables, although in the first case transformations are usually needed. Moreover, the fact that there is no need to determine a similarity measure simplifies further the procedure. All these advantages address the issues that are mentioned in the previous paragraph. In addition to that, the property of producing understandable rules constitutes DTs a powerful tool in SPC during off-line analysis.

When the instance is a time series, as in our case, each time point corresponds to an (numerical) attribute. The interpretation of the derived rules is more difficult, especially if the DT is constructed on the original values of time series. One approach is to rephrase these rules in order to accommodate the temporal nature of data. Another approach is to implement an appropriate representation scheme on data prior to the construction of the DT, which may produce more understandable rules.

There are some disadvantages though. DTs may be quite large and since they are constructed from training data, overfitting is possible. Tree pruning usually overcomes these problems. Also, the decision tree approach involves dividing the search space into rectangular regions and classifying an object based on the region into which it falls. This implies that it is not an appropriate approach for all classification problems.

Our proposed approach includes also the representation of time series for the purpose of reducing their intrinsically high dimensionality. Although there are several representations introduced in the literature, in this work we consider only two of them, namely, the Feature-based (FB) (Section 2.5.4) and the Symbolic Aggregate Approximation (SAX) (Section 2.5.5). This choice is made due to the fact that these two representations differ significantly in their approach. FB is based on the calculation of statistical measures and transforms the original data into a vector of real values where the transformed values do not retain the time aspect of the original data. In order for this representation to be implemented the whole series should be obtained. On the other hand, SAX is a symbolic representation that transforms the original data into a vector of discrete symbols where the transformed values are of ordinal level retaining the time aspect of the original data. Also, this representation can be implemented on streaming data and thus, it can be very useful in on-line SPC. Besides that, SAX is a very popular representation scheme in the TSDM community.

Finally, the implementation of FB and SAX representations on data in conjunction with DT may result into classification rules that that are understandable. For example, when FB is applied, one rule could be stated as follows:

If the mean value of the time series is less than 30 and the standard deviation is greater than 2, then the time series has a cyclic pattern.

Another example of a classification rule with respect to SAX representation could be the following:

If the mean value of the first five values of the time series is between 28 and 32 and the mean value of the next five values is between 32 and 36, then the time series has a trend pattern.

3.4 Experimental Framework

3.4.1 Description of the Datasets

The aim of this chapter is to investigate the possibility that TSDM techniques can contribute to control chart pattern recognition rather than apply and compare all potential approaches from this field. In order to conduct experiments, six control chart patterns (the most common types encountered in practice) are selected: natural, cyclic, upward trend, downward trend, upward shift and downward shift. Two datasets, A and B, are generated each consisting of 1800 time series of length 56 (300 from each one of the six patterns), by using the following equations:

$$x_t = \mu + r_t \qquad \text{(natural)}$$

$$x_t = \mu + r_t + A\sin(2\pi t / \Omega t) \qquad \text{(cyclic)}$$

$$x_t = \mu + r_t \pm bt \qquad \text{(trend)}$$

$$x_t = \mu + r_t \pm ps \qquad \text{(shift)}$$

where,

x_t: time series value

μ: process mean ($\mu = 30$)

σ: standard deviation when process is in-control ($\sigma = 2$)

r_t: common cause variation, following a normal distribution with mean of zero and standard deviation in terms of σ (0.1σ, 0.3σ and 0.5σ)

A: cyclic amplitude in terms of σ (1σ, 2σ and 3σ)

Ω: period of cycle ($\Omega = 8$)

b: trend slope in terms of σ (0.1σ, 0.2σ and 0.3σ)

p: parameter that determines the shift position $p = 0$ before shifting, $p = 1$ after shifting) while the shift position is set to be uniformly distributed between the 21st and 40th time points

s: shift magnitude in terms of σ (1σ, 2σ and 3σ)

The difference between the two datasets is in the presence of in-control data. In the first dataset (A), where in-control data is not included, the disturbance starts at the first time point for the cyclic and trend patterns, whereas in the second dataset (B), where in-control data is included, cyclic and trend patterns are generated by setting the starting time that the disturbance occurred uniformly distributed between the 21^{st} and 40^{th} time points. There is no clear-cut answer as to the most appropriate number of in-control data and the starting position of a disturbance [31]. The choice of the time series length is application-dependent. The process itself and the monitoring measurements affect the sampling frequency and, thus, the window size to be selected. In this chapter, we consider time series of individual measurements and the length is determined to be 56. One reason is that it is a rational choice for applications where the frequency of sampling is high (e.g., every one second). Another reason is that it facilitates the specific experiments where we have included in-control data. The above parameters are selected to ensure that the majority of the generated patterns fluctuated within $\pm 3\sigma$, since the identification of a deviation beyond these limits can be easily identified as out of control incidence [56]. MATLAB is utilized in order to generate these datasets.

3.4.2 Methods & Settings

In Section 3.3.2, two different representation schemes are selected for the classification task. The SAX scheme has two parameters: the length of the transformed series (k) and the alphabet size (α). In our experiments, SAX representation is applied using different values of these two parameters. In particular, k is set equal to 4, 6 and 8, and α is set equal to 6, 8 and 10 resulting into nine different representations. In order to distinguish among the different set of parameters, the notation SAX(k,a) will be used. The corresponding code for this representation is freely available by Keogh et. al. (Copyright (c) 2003, Eamonn Keogh, Jessica Lin, Stefano Lonardi, Pranav Patel). The FB scheme has one parameter D, which is set equal to one. In our experiments, two different FB schemes are applied. In the first one, FB(8), the original time series is transformed to a vector of eight

real values (statistical features of the original and the transformed series), whereas in the second one, FB(4), the transformation results into a vector of four real values (statistical features of only the original series). MATLAB is also utilized in order to apply these two FB representations.

Moreover, DT has been chosen as the Data Mining classification method to be applied. Three of the most popular algorithms have been applied in both datasets: CHAID (Chi-squared Automatic Interaction Detector) [19], CRT [21] (a version of the Classification and Regression Trees algorithm) and QUEST (Quick, Unbiased, Efficient Statistical Tree) [108]. The aim of these experiments is to compare the performance of various classification algorithms using different representations rather than determine the optimal performance. SPSS 13.0 is utilized in order to perform these experiments and its defaults are selected. Specifically, the maximum number of levels in the tree below the root node is set to 3 for CHAID and 5 for CRT and QUEST, while the minimum number of examples is set to 100 in the parent node and 50 in the child node. The corresponding trees are pruned with the maximum difference in risk to be of one standard error. These growth limits criteria and the pruning procedure aim at constructing simpler trees, consequently simpler rules, and dealing with the problem of overfitting. The evaluation measure is the classification accuracy for the testing dataset. The validation is based on randomly splitting (approximately in half) the dataset into training and testing samples. Each algorithm is repeated ten times and the corresponding values of mean and standard deviation are calculated for classification accuracy.

3.5 Results

3.5.1 Comparisons among Representation Schemes

The overall classification rates of our experiments are presented at Table 3.1. One important observation is that the classification rates are (in all cases except CRT with raw data and CRT with feature-based representations) lower than 90%, while the corresponding rates in the literature are above 90%. This is due to the strict criteria we used for the growth of the tree, as it was described earlier. These experiments result in fairly simple trees since the number of nodes is ranging from 5 to 15, the number of leaf nodes from 4 to 10 (the maximum depth ranges from 3 to 5 by default). Obviously, the number of leaf nodes and the depth of the tree determine the number of the rules that will be generated and their complexity. Our aim is not to investigate the trade-off between accuracy and complexity of

the generated trees. In this chapter, the classification rates serve more as a comparison tool among different representation schemes and algorithms.

Regarding dataset A, the most accurate results are obtained when CRT algorithm is applied to FB transformed data (99.9%) and to raw data (95.75%). Moreover, CRT algorithm provides more accurate results than the other two algorithms in every representation (above 80%). Although these differences have not been statistically validated, they provide an indication of the possible usefulness of this algorithm. On the other hand, CHAID provides comparable accuracy rates to CRT when it is applied on SAX transformed data. The corresponding best results are obtained when SAX(6,10) and SAX(8,10) representations are used (86%). This fact may imply the appropriateness of using larger alphabet size. QUEST algorithm provides the less accurate results in most of the cases. This algorithm shows a noticeable instability in results within each experiment having a standard deviation that reaches 18.93 units. Thus, even when QUEST results into comparable classification rates, they cannot be considered adequately reliable.

Regarding dataset B, the most accurate results are obtained when CRT algorithm is applied to FB(8) transformed data (88.54%) and to FB(4) transformed data (88.30%). As it was expected, accuracy rates for datasets with in-control data not included are higher than those for datasets with in-control data included. A few exceptions appear when QUEST algorithm is applied but due to its instability they cannot be considered important. CRT and CHAID algorithms provide similar results when they are applied on raw data or on SAX represented data..

Regarding SAX representation in dataset A, the best results are provided when an alphabet size of 8 or 10 is selected, regardless the length of the transformed series. In dataset B, the best results are provided when an alphabet size of 10 is selected with an exception of SAX(6,6) which provides a comparable accuracy rate of 75.95%.

3.5.2 Comparisons Among SAX Representations Per Pattern

Focusing on SAX representations and CRT algorithm, further comparisons are performed with respect to the classification rates per pattern.

There are two well-known difficulties in classifying control charts to appropriate patterns. The first problem is in distinguishing natural from cyclic patterns and the second one is in distinguishing trends from shifts. Table 3.2 shows the mean classification rates per pattern per SAX representation resulted from dataset A.

Regarding the first problem, the best results are obtained when data are transformed to a series of length 4 with the classification rates to range from 64.89% to 74.50%. The

corresponding rates for the other cases range from 41.82% to 63.90%. On the other hand, representations of length 6 and 8 seem to perform better in distinguishing trends from shifts with classification rates reaching 98.05%. In addition to that, when the alphabet size increases the classification rates improve. The representations SAX(6,10) and SAX(8,10) provide accuracy rates for all patterns above 90% except for the natural pattern where the corresponding rates are 47.79% and 42.89%.

Table 3. 1 Mean Accuracy Rates (Standard Deviations)

	DATASET A			DATASET B		
	CHAID	CRT	QUEST	CHAID	CRT	QUEST
RAW	78.90 (5.09)	95.75 (0.49)	84.62 (18.93)	66.20 (1.70)	75.23 (2.48)	62.88 (8.60)
FB(4)	81.02 (2.48)	99.90 (0.08)	78.33 (13.99)	67.69 (4.99)	88.30 (0.51)	55.33 (16.39)
FB(8)	80.64 (2.34)	99.90 (0.08)	77.85 (13.63)	64.59 (0.93)	88.54 (0.74)	69.42 (10.87)
SAX(4,6)	79.59 (2.58)	82.67 (2.74)	61.08 (5.71)	66.43 (1.72)	72.80 (1.03)	71.60 (1.72)
SAX(4,8)	84.71 (0.89)	89.48 (2.42)	72.55 (7.78)	72.77 (1.09)	71.44 (1.25)	68.83 (7.92)
SAX(4,10)	81.66 (2.55)	86.42 (2.78)	78.75 (8.17)	77.69 (0.90)	75.15 (2.07)	72.30 (5.84)
SAX(6,6)	79.83 (2.53)	82.09 (1.62)	56.38 (4.24)	75.95 (0.77)	71.41 (4.04)	48.83 (1.43)
SAX(6,8)	84.53 (1.26)	88.66 (2.60)	73.07 (10.66)	71.98 (2.08)	68.94 (0.97)	68.46 (1.45)
SAX(6,10)	86.43 (1.57)	88.47 (1.45)	54.54 (6.74)	75.62 (2.31)	74.46 (1.47)	55.03 (6.76)
SAX(8,6)	78.00 (2.83)	81.18 (2.17)	54.87 (2.31)	69.07 (1.30)	65.72 (0.99)	48.40 (0.86)
SAX(8,8)	80.51 (2.32)	84.81 (1.60)	62.94 (9.93)	70.62 (1.60)	73.22 (0.82)	68.90 (6.31)
SAX(8,10)	86.26 (2.22)	88.06 (0.77)	52.67 (3.57)	76.24 (1.34)	75.63 (1.76)	68.53 (7.78)

Table 3.2 Mean Accuracy Rates Per Pattern (Standard Deviations) – Dataset A

	Natural	Cyclic	Upward Trend	Downward Trend	Upward Shift	Downward Shift
SAX(4,6)	64.89 (16.35)	92.70 (3.68)	100.00 (0.00)	100.00 (0.00)	70.30 (2.91)	68.76 (3.00)
SAX(4,8)	74.50 (15.22)	91.29 (3.10)	99.20 (0.62)	100.00 (0.00)	84.91 (1.42)	87.17 (1.52)
SAX(4,10)	70.80 (17.47)	91.84 (3.86)	98.93 (0.57)	100.00 (0.00)	78.12 (1.55)	78.74 (3.33)
SAX(6,6)	47.47 (7.84)	90.66 (3.63)	99.23 (1.25)	100.00 (0.00)	74.68 (2.21)	81.26 (1.87)
SAX(6,8)	63.90 (17.44)	89.44 (3.84)	100.00 (0.00)	100.00 (0.00)	88.87 (2.57)	89.05 (2.04)
SAX(6,10)	47.79 (9.11)	90.79 (2.95)	98.20 (0.42)	98.68 (0.89)	98.05 (0.66)	97.82 (1.62)
SAX(8,6)	41.82 (3.55)	93.76 (1.92)	100.00 (0.00)	99.86 (0.29)	75.30 (10.34)	76.17 (11.84)
SAX(8,8)	49.15 (10.87)	93.52 (2.44)	100.00 (0.00)	100.00 (0.00)	81.93 (2.34)	84.23 (2.40)
SAX(8,10)	42.89 (3.58)	93.55 (2.55)	98.80 (0.60)	100.00 (0.00)	96.48 (0.91)	96.82 (2.98)

Regarding results from dataset B (Table 3.3), similar observations can be made in general. Series of length 4 performs better in distinguishing natural from cyclic patterns, the classification accuracy rates for trends and shifts are comparable though. Regardless the length of the transformed series, an alphabet size of 10 provides greater accuracy rates for shifts (above 70%).

Table 3.3 Mean Accuracy Rates Per Pattern (Standard Deviations) - Dataset B

	Natural	Cyclic	Upward Trend	Downward Trend	Upward Shift	Downward Shift
SAX(4,6)	55.45 (17.27)	61.14 (18.24)	100.00 (0.00)	100.00 (0.00)	58.56 (2.85)	62.10 (2.34)
SAX(4,8)	28.86 (34.28)	77.79 (31.27)	99.27 (0.85)	98.34 (0.68)	62.56 (2.74)	63.31 (2.48)
SAX(4,10)	39.15 (45.69)	64.03 (45.76)	94.48 (1.81)	93.90 (1.37)	80.42 (2.45)	80.64 (1.78)
SAX(6,6)	3.47 (10.97)	94.46 (10.58)	98.62 (3.00)	99.60 (0.34)	62.80 (16.27)	72.05 (16.74)
SAX(6,8)	9.93 (31.40)	90.00 (31.62)	100.00 (0.00)	100.00 (0.00)	54.67 (2.93)	59.69 (2.30)
SAX(6,10)	18.28 (38.53)	79.79 (42.05)	98.52 (0.66)	98.96 (3.29)	72.79 (1.89)	78.25 (5.92)
SAX(8,6)	3.70 (11.71)	92.03 (11.39)	83.35 (14.25)	81.28 (18.44)	69.20 (17.25)	66.92 (17.72)
SAX(8,8)	11.00 (30.68)	89.20 (31.35)	99.59 (0.36)	99.72 (0.36)	69.72 (2.00)	74.35 (1.72)
SAX(8,10)	3.78 (7.98)	96.45 (1.90)	99.87 (0.42)	100.00 (0.00)	77.14 (2.62)	78.93 (2.30)

3.6 Conclusion

In this chapter, the possible contribution of Time Series Data Mining in control chart pattern recognition has been investigated. We approach this problem by selecting fairly simple options, among a wealth of TSDM representations, similarity measures and data mining methods. Feature-based and Symbolic Aggregate Approximation representations, although different to each other, are both easy to be implemented. Moreover, in conjunction with Decision Trees for the classification task, they can produce understandable rules that may facilitate the procedure of interpreting control charts and taking a corrective action as fast as possible. The experiments have been performed on synthetically generated datasets, taking into account many different parameters such as noise, magnitude of deviations and presence of in-control data. The evaluation is based on the classification accuracy rates and comparisons arae made among three different algorithms (CHAID, CRT and QUEST) and

between two different datasets (with and without in-control data present in training and testing examples). Results show that although the generated models are simple the accuracy rates are adequately high, especially when feature-based representation is implemented and CRT algorithm is utilized for classification.

SAX representation performs significantly better overall when the alphabet size is greater than four regardless the length of the transformed series. However, the length of the transformed series seems to affect the discrimination capability of the classification procedure. Series of length 4 performs better in distinguishing natural from cyclic patterns, whereas series of length 6 or 8 performs better in distinguishing trends from shifts. The alphabet size also affects the discriminating capability with sizes larger than 6 to improve performance.

Decision trees generated by raw data provide comparable accuracy rates to the trees generated by transformed data. The role representation schemes play in classification within SPC context will be further investigated in future work. As mentioned previously, the accuracy rates reported are affected by the desired complexity of the generated decision trees and, thus, they cannot be compared to other experiments in the literature. In addition to that, these results have not been statistically validated. Nevertheless, they provide some indications of the possible strengths and weaknesses of the various representation schemes and algorithms. Finally, these results indicate that Time Series Data Mining approaches may be comparable to other approaches such Neural Networks.

CHAPTER 4

A Hybrid Time Series Representation

4.1 Introduction

There is a large number of representations proposed in the literature, which aimed at reducing the dimensionality of a time series dataset, while retaining as much information as possible. As it is expected, there is no representation that can be considered as the most appropriate for all cases. The application under consideration, the data mining method and the specific characteristics of data play an important role in the selection of a representation scheme.

In this chapter, a new representation scheme is introduced, named D-PAA, which can be considered as a variation of Piecewise Aggregate Approximation (PAA) (Section 2.5.3). A time series is segmented into a series of equal length sections and the corresponding mean and standard deviation are recorded for each one of them. The difference with PAA is that D-PAA takes into consideration not only the central tendency but also the dispersion present in each section. This additional information may improve the performance of the dimensionality reduction in the context of time series similarity search. In addition to that, a distance measure is proposed that is proved to lower-bound the Euclidean distance between two series. This measure involves a parameter that assigns different weights in means than in standard deviations. Standard deviation is also utilized, along with other statistical measures, in the Feature-based (FB) representation proposed by Nanopoulos et. al [117], however in their approach it is calculated for the whole time series. More specifically, the authors propose to calculate four statistical measures (mean, standard deviation, skewness, kurtosis) for the whole time series and the same measures for the transformed values of the original series (Section 2.5.4). Another similar approach [107] uses two additional values, besides the mean value, for representing a segment of a time series, namely, the corresponding minimum and maximum values. These values are mapped to an alphabet to produce a symbolic representation. This approach is an extension to SAX representation [104] and aims at improving the representation of financial time series data.

Extensive experiments on 20 widely utilized datasets are conducted in order to evaluate D-PAA and compare it to PAA and FB, as well as with other commonly applied

representations in Time Series Data Mining applications. This evaluation is based on 1-NN classification and k-NN similarity search.

The main contributions made in this Chapter are:

- The introduction of a new representation (D-PAA) along with appropriate experimental evaluation. Several representations have been tested for comparison reasons along with the application of Euclidean distance and Dynamic Time Warping on raw/original data.
- The definition of a distance measure under D-PAA representation that lower-bounds the Euclidean distance along with the corresponding proof.
- The extensive experimental evaluation of FB representation in the context of kNN similarity search and 1NN classification.
- The presentation of classification results under different representations on twenty datasets, which have been used extensively in the literature for testing classification algorithms and techniques.

In Section 4.2, the proposed representation (D-PAA) is presented and contrasted with FB and PAA representations. In Section 4.3, the experimentation framework is described, whereas in Section 4.4 the corresponding results are presented. Finally, a further discussion on D-PAA is provided in Section 4.5 along with a final conclusion in Section 4.6.

4.2 The D-PAA Representation

The D-PAA technique is a variation of the PAA representation, which is described in Section 2.5.5. The PAA technique segments a time series of length n into k consecutive sections of equal-width and calculates the corresponding mean for each one. In addition to that, D-PAA calculates the corresponding standard deviations and the resulting representation is a series of these means and standard deviations. These statistical measures can be also referred as features.

4.2.1 Description of D-PAA Representation

Formally, a time series $X = x_1, x_2, \cdots, x_n$ can be represented by a series $\tilde{X} = (\overline{x}_1, s_{x,1}), (\overline{x}_2, s_{x,2}), \cdots, (\overline{x}_k, s_{x,k})$, where:

$$\overline{x}_i = \frac{\sum_{j=(i-1)\frac{n}{k}+1}^{i\frac{n}{k}} x_j}{\frac{n}{k}} \tag{4.1}$$

$$s_{x,i} = \sqrt{\frac{\sum_{j=(i-1)\frac{n}{k}+1}^{i\frac{n}{k}} (x_j - \overline{x}_i)^2}{\frac{n}{k} - 1}} \tag{4.2}$$

The quantity n/k expresses the number of points that each section consists of and it is assumed that it is an integer. Otherwise, an appropriate number of zeros are added at the end of original series prior to representation. As it is stated in [161], this modification does not affect query results.

The dimensionality of the transformed time series is equal to $2 \cdot k$, since for each of the k segments two values are recorded.

In order to determine the distance between two series in the feature space, a distance measure is needed. In this work, a weighted Euclidean distance is proposed, as follows. Suppose that there are two time series $X = x_1, x_2, \cdots, x_n$ and $Y = y_1, y_2, \cdots, y_n$ along with their representations

$$\tilde{X} = (\overline{x}_1, s_{x,1}), (\overline{x}_2, s_{x,2}), \cdots, (\overline{x}_k, s_{x,k})$$

$$\tilde{Y} = (\overline{y}_1, s_{y,1}), (\overline{y}_2, s_{y,2}), \cdots, (\overline{y}_k, s_{y,k}).$$

The distance between $\tilde{X}$ and $\tilde{Y}$ is defined in the following equation.

$$D_{D-PAA}(\tilde{X}, \tilde{Y}) = \sqrt{w \cdot \sum_{i=1}^{k} (\overline{x}_i - \overline{y}_i)^2 + (1-w) \cdot \sum_{i=1}^{k} (s_{x,i} - s_{y,i})^2}, \tag{4.3}$$

$$0 \le w \le 1$$

where w is a user-specified parameter that assigns different weights to the differences of means and standard deviations. This parameter enables the user to assign relative importance to mean values and standard deviations, according to the specific application and/or data type under consideration. In the extreme case of $w = 1$, the D-PAA representation is equivalent to PAA, since only the mean values are taken into account. As w decreases, more weight is assigned to standard deviations. In the extreme case of $w = 0$, the D-PAA representation ignores the mean values. Note that the case of $w = 0.5$ does not imply that each feature has equal absolute influence on the distance function value [61]. One way of giving all features equal influence in characterizing overall dissimilarity between time series is to normalize the values of each feature prior to calculating the weighted distance with $w = 0.5$. However, this normalization may incur an extra computation cost, since it must be realized each time a new query arrives.

Lemma 4.1. The distance between $\tilde{X}$ and $\tilde{Y}$, as it is defined in Eq. 4.3, lower-bounds the Euclidean distance between X and Y, that is,

$$D_{D-PAA}(\tilde{X},\tilde{Y}) \le D(X,Y)$$

or equivalently

$$\sqrt{w\cdot\sum_{i=1}^{k}(\bar{x}_i-\bar{y}_i)^2+(1-w)\cdot\sum_{i=1}^{k}(s_{x,i}-s_{y,i})^2} \le \sqrt{\sum_{i=1}^{n}(x_i-y_i)^2} \qquad (4.4)$$

where k is the number of segments. It is assumed that X and Y have the same length n. If they have unequal lengths, one of them should be interpolated.

Proof

The fact that the parameter w takes on values between 0 and 1 implies that

$$w\cdot\sum_{i=1}^{k}(\bar{x}_i-\bar{y}_i)^2+(1-w)\cdot\sum_{i=1}^{k}(s_{x,i}-s_{y,i})^2 \le \sum_{i=1}^{k}(\bar{x}_i-\bar{y}_i)^2+\sum_{i=1}^{k}(s_{x,i}-s_{y,i})^2 .$$

Thus, it is equivalent to prove that

$$\sum_{i=1}^{k}(\bar{x}_i-\bar{y}_i)^2+\sum_{i=1}^{k}(s_{x,i}-s_{y,i})^2 \le \sum_{i=1}^{n}(x_i-y_i)^2 \qquad (4.5).$$

The left hand side of the above inequality (Eq. 4.5) is the summation over k segments, whereas the right hand side is the summation over all points of the series. It is sufficient to prove this inequality for one segment only:

$$(\bar{x}-\bar{y})^2+(s_x-s_y)^2 \le \sum_{i=1}^{n/k}(x_i-y_i)^2 \qquad (4.6)$$

where n/k is the number of points that comprise a segment, $\bar{x}$, $\bar{y}$ are the mean values of the corresponding segments of X and Y, and s_x, s_y are the corresponding standard deviations. The proof can be concluded by applying the same proof of Eq. 4.6 to every segment and adding the corresponding parts of the inequalities.

Although the following proof is straightforward, there are two relations that need to be mentioned. The first one (Eq. 4.7) is a well-known property of the mean value, whereas the second one (Eq. 4.8) constitutes a property of Pearson's correlation coefficient r (Eq. 4.9), whose value is always less than or equal to one.

$$\sum_{i=1}^{n}(x_i-\bar{x})=0 \qquad (4.7)$$

$$\sqrt{\sum_{i=1}^{n}(x_i-\overline{x})^2}\cdot\sqrt{\sum_{i=1}^{n}(y_i-\overline{y})^2}\geq\sum_{i=1}^{N}(x_i-\overline{x})(y_i-\overline{y}) \tag{4.8}$$

$$r=\frac{\sum_{i=1}^{N}(x_i-\overline{x})(y_i-\overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\overline{x})^2}\cdot\sqrt{\sum_{i=1}^{n}(y_i-\overline{y})^2}}\leq 1 \tag{4.9}$$

The proof of the lower-bounding property, when one segment is taken into consideration (Eq. 4.6), is as follows:

$$\sum_{i=1}^{n/k}(x_i-y_i)^2=$$ the right hand side of Eq. 4.6

$$=\sum_{i=1}^{n/k}(x_i-\overline{x}+\overline{x}-y_i+\overline{y}-\overline{y})^2=$$ add and subtract the mean values

$$=\sum_{i=1}^{n/k}[(\overline{x}-\overline{y})+(x_i-\overline{x})-(y_i-\overline{y})]^2=$$

$$=\sum_{i=1}^{n/k}[(\overline{x}-\overline{y})^2+(x_i-\overline{x})^2+(y_i-\overline{y})^2+$$ developing the square of the quantity within brackets

$$+2(x_i-\overline{x})(\overline{x}-\overline{y})-2(\overline{x}-\overline{y})(y_i-\overline{y})-$$

$$-2(x_i-\overline{x})(y_i-\overline{y})]=$$

$$=\sum_{i=1}^{n/k}(\overline{x}-\overline{y})^2+\sum_{i=1}^{n/k}(x_i-\overline{x})^2+\sum_{i=1}^{n/k}(y_i-\overline{y})^2+$$ determine the sum of each term

$$+\sum_{i=1}^{n/k}2(x_i-\overline{x})(\overline{x}-\overline{y})-\sum_{i=1}^{n/k}2(\overline{x}-\overline{y})(y_i-\overline{y})-$$

$$-\sum_{i=1}^{n/k}2(x_i-\overline{x})(y_i-\overline{y})=$$

$$=(\frac{n}{k})(\overline{x}-\overline{y})^2+\sum_{i=1}^{n/k}(x_i-\overline{x})^2+\sum_{i=1}^{n/k}(y_i-\overline{y})^2$$

$$+2(\overline{x}-\overline{y})\sum_{i=1}^{n/k}(x_i-\overline{x})-2(\overline{x}-\overline{y})\sum_{i=1}^{n/k}(y_i-\overline{y})-$$

$$-2\sum_{i=1}^{n/k}(x_i-\bar{x})(y_i-\bar{y})]=$$

$$=(\frac{n}{k})(\bar{x}-\bar{y})^2+\sum_{i=1}^{n/k}(x_i-\bar{x})^2+\sum_{i=1}^{n/k}(y_i-\bar{y})^2- \qquad \text{apply Eq. 4.7}$$

$$-2\sum_{i=1}^{n/k}(x_i-\bar{x})(y_i-\bar{y})\geq$$

$$\geq(\bar{x}-\bar{y})^2+\sum_{i=1}^{n/k}(x_i-\bar{x})^2+\sum_{i=1}^{n/k}(y_i-\bar{y})^2- \qquad \text{subtract } (\frac{n}{k}-1)(\bar{x}-\bar{y})^2$$

$$-2\sum_{i=1}^{n/k}(x_i-\bar{x})(y_i-\bar{y})\geq$$

$$\geq(\bar{x}-\bar{y})^2+\sum_{i=1}^{n/k}(x_i-\bar{x})^2+\sum_{i=1}^{n/k}(y_i-\bar{y})^2- \qquad \text{apply Eq. 4.8}$$

$$-2\sqrt{\sum_{i=1}^{n/k}(x_i-\bar{x})^2}\sqrt{\sum_{i=1}^{n/k}(y_i-\bar{y})^2}=$$

$$=(\bar{x}-\bar{y})^2+\left[\sqrt{\sum_{i=1}^{n/k}(x_i-\bar{x})^2}-\sqrt{\sum_{i=1}^{n/k}(y_i-\bar{y})^2}\right]^2=$$

$$=(\bar{x}-\bar{y})^2+\left[(\frac{n}{k}-1)s_x-(\frac{n}{k}-1)s_y\right]^2= \qquad \text{apply Eq. 4.2}$$

$$=(\bar{x}-\bar{y})^2+(\frac{n}{k}-1)^2\left(s_x-s_y\right)^2\geq$$

$$\geq(\bar{x}-\bar{y})^2+\left(s_x-s_y\right)^2 \qquad \text{the left hand side of Eq. 4.6}$$

As it was mentioned in Introduction, the D-PAA can be contrasted with PAA (Section 2.5.3) and FB (Section 2.5.4). One aspect is the expected quality of these representations. FB has the disadvantage that describes a time series globally, whereas D-PAA and PAA describe a specific number of segments that constitute a time series. An important point with respect to PAA and D-PAA is that the first utilizes twice as many segments than the latter. D-PAA aims at describing each segment more thoroughly than PAA does, at the expense of describing fewer segments. Another aspect of the proposed representations is the dimensionality reduction. In the case of FB, the representation is of constant length 8, since there are four statistical measures that calculated for the values of the original time series and the transformed one. On the other hand, D-PAA representation can

be of length that is a multiple of 2, since two measures are calculated for each segment, whereas there is no such restriction for PAA.

4.2.2 Graphical Description of D-PAA Representation

As it is mentioned in the previous section, D-PAA requires the segmentation of a time series into a series of equal length sections. D-PAA representation is the vector that consists of the corresponding means and standard deviations of each one of these sections. Suppose that we have a time series X that is segmented into 10 sections. In Figure 4.1 the graph of this time series is presented along with the corresponding PAA representation (the graph of PAA series is shifted for presentation purposes). Each segment represents the mean value of the corresponding section.

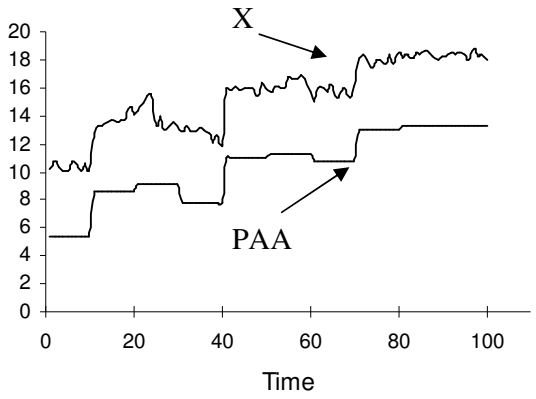


Figure 4.1 The graph of the time series X along with the corresponding PAA representation

However, D-PAA representation takes into consideration not only the mean values but also the standard deviations of each section. In order to present this additional information graphically (Figure 4.2), we add two more series that correspond to the upper and lower bounds of each section i defined by $\overline{x}_i \pm s_{x,i}$ (the graph of D-PAA series is shifted for presentation purposes). The portion of the values that lies within these intervals depends on the distribution of the values, which is unknown in general. For example, if we assume normality, the Empirical Rule states that approximately 68% of the values lie within these intervals. Nevertheless, these bounds indicate the dispersion of the values to some extent.

Suppose now that we have another time series Y that we wish to compare with X (Figure 4.3). The proposed distance measure between two time series under D-PAA representation is based on the differences of the corresponding means and standard

deviations. Figure 4.4 presents the upper bounds, $\overline{x}_i + s_{x,i}$ and $\overline{y}_i + s_{y,i}$, for each one of X and Y. The vertical distances of these bounds represent the summation of the differences between the corresponding means and standard deviations. More formally,

$$(\overline{y}_i + s_{y,i}) - (\overline{x}_i + s_{x,i}) = (\overline{y}_i - \overline{x}_i) + (s_{y,i} - s_{x,i}).$$

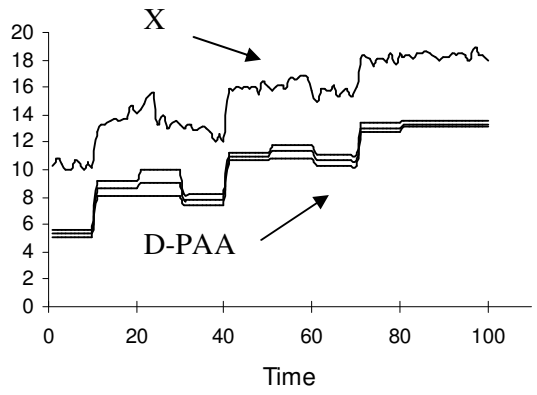


Figure 4.2 The graph of the time series X along with the corresponding D-PAA representation

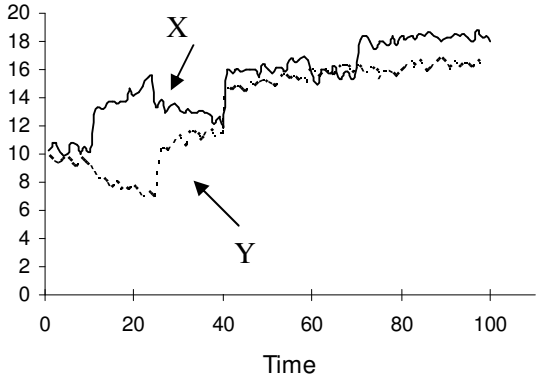


Figure 4.3 The graph of the time series X and Y

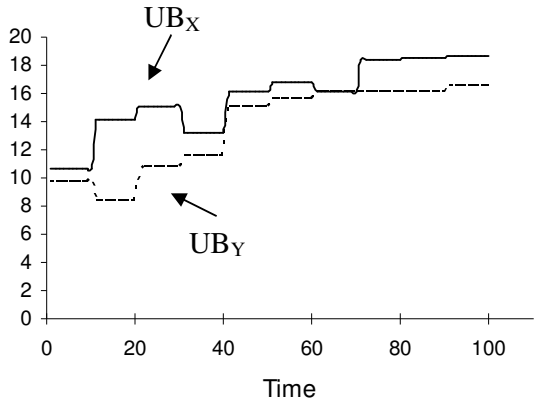


Figure 4.4 The graph of the Upper Bounds of two time series (X , Y). Solid line corresponds to X , whereas dotted line corresponds to Y .

4.3 Framework of Experimentation

The experiments have been conducted on 20 real-world and synthetic datasets that have been used extensively in the literature and are described in Section 4.3.1. Section 4.3.2 presents the evaluation methods and Section 4.3.3 discusses the rival measures along with their corresponding settings.

4.3.1 Datasets

The experiments have been conducted on 20 real world and synthetic datasets, which are available upon request in [64]. Most of them have been used extensively in the TSDM literature, for the purpose of testing the performance of novel representation schemes and similarity measures with respect to specific Data Mining tasks and/or indexing. They have been utilized extensively as benchmark datasets for testing classification algorithms and for this reason, they are separated in training and testing sets. All series are labeled according to the class they belong to.

The majority of these datasets (17) consist of normalized time series, that is, time series of mean equal to zero and standard deviation equal to one. In this work, the remaining three (3) datasets have been also normalized. A description of these datasets is presented in

Table 4.1.

When 1-NN classification is applied, the datasets are utilized in the form they are provided. On the other hand, k-NN similarity search is applied on datasets generated by merging the corresponding train and test sets.

Table 4.1 Description of Datasets

ID	DATASET	CLASSES	TRAIN*	TEST*	LENGTH
1	Adiac	37	390 (4-15)	391 (6-16)	176
2	50words	50	450 (1-52)	455 (1-57)	270
3	CBF	3	30 (8-12)	900 (298-302)	128
4	ECG200	2	100 (31-69)	100 (36-64)	96
5	FaceAll	14	560 (40)	1690 (8-233)	131
6	FaceFour	4	24 (3-8)	88 (14-26)	350
7	Fish	7	175 (21-28)	175 (22-29)	463
8	GunPoint	2	50 (24-26)	150 (76-74)	150
9	Lighting2	2	60 (20-40)	61 (28-33)	637
10	Lighting7	7	70 (5-19)	73 (6-19)	319
11	OSULeaf	6	200 (15-53)	242 (23-55)	427
12	SwedishLeaf	15	500 (26-42)	625 (33-49)	128
13	SyntheticControl	6	300 (50)	300 (50)	60
14	Trace	4	100 (21-31)	100 (19-29)	275
15	TwoPatterns	4	1000 (237-271)	4000 (959-1035)	128
16	Wafer	2	1000 (97 , 903)	6164 (665 ,5499)	152
17	Yoga	2	300 (137 , 163)	3000 (1393 , 1607)	426
18	Beef	5	30 (6)	30 (6)	470
19	Coffee	2	28 (14)	28 (15 , 13)	286
20	OliveOil	4	30 (4-13)	30 (4-12)	570

* Numbers in parentheses show the range of sizes of classes.

4.3.2 Methods

The proposed representation D-PAA is evaluated by two methods, namely, 1-NN classification and k-NN similarity search. These are the most commonly used methods in

TSDM literature for the purpose of evaluating novel representation schemes and/or similarity measures [88] [110].

Regarding 1-NN classification for a given dataset, the procedure is as follows:

1. Obtain a series (q) from the test set
2. Find its distance from every series (X_i) of the train set
3. Select the closest series (minimum distance) and obtain its class
4. If this class is the same as the class of q , then the classification is correct
5. Repeat this procedure for each series in the test set

The error rate of classification is the percentage of incorrect classifications over the total number of series in the test set.

Regarding k-NN search, the parameter k is set equal to the number of all time series that belong to the same class with the query. Consequently, the parameter k is set equal to a value that may be different among queries. This setting implies that for a given query, all relevant (belonging to the same class) series are requested.

More specifically, k-NN similarity search is performed as follows:

1. Obtain a series (q) from the dataset along with its class and set it as the query
2. Find the number of series that belong to the same class with q and denote it k
3. Find the distance of q from every series (X_i) of the dataset
4. Select the k closest series (minimum distances) and obtain their classes
5. Compute the number of these classes that are the same with the class of q .
6. The error rate for q is the percentage of the k closest series with different class than q
7. Repeat this procedure for each series in the test set

The error rate of similarity search is the average of the error rates for every q .

4.3.3 Rival Approaches - Parameters

In the first part of the experiments, the parameter (D) of the FB representation is examined. The FB approach involves the calculation of mean, standard deviation, skewness and kurtosis for the original values x_t of a time series and for their transformed values $x'_t = (x_{t+D} - x_t)$. This representation is tested on the datasets and methods described in the previous sections for varying values of D. The value of the parameter D is set equal to 1 up to 20. Intuitively, a time period of length less than or equal to 20 is assumed to be a reasonable one in order to differencing the corresponding values. Alcock and Manolopoulos present results for values of D up to 9 [7]. Since time series are normalized, means and

standard deviations of untransformed data are constants across all series and equal to 0 and 1 respectively, and thus they are excluded from the feature vector. In this case, the length of the feature vector is equal to 6.

In the second part of the experiments, the proposed distance measure for D-PAA is examined. As it is described in Section 4.2, the proposed distance measure can be applied on the original features that D-PAA representation has generated or on the normalized features. These two cases are tested on the datasets and methods described in the previous section.

The third part of the experiments aims at comparing D-PAA with PAA. Each time series is first segmented into a number of sections, which constitutes a parameter. As it is stated in Section 4.2, D-PAA representation can be of length that is a multiple of 2, whereas there is no such restriction for PAA. In order to compare these representations on the same dimensionalities, the number of segments is set equal to multiples of 2, ranging from 4 to 16.

Another parameter that needs to be specified is the weight (w) that is used in the distance function of D-PAA. The values tested in experiments range from 0 to 1 in increments of 0.1. Again, these representations are tested on the same datasets with respect to 1-NN classification and k-NN search.

In the final part of the experiments, D-PAA is compared with alternative representations and methods with respect to 1-NN classification. In particular, the alternative methods that are taken into consideration refer to the cases where Euclidean distance and Dynamic Time Warping are applied on raw data (when no representation is applied) along with FB, SVD and DFT.

4.4 Results

4.4.1 Feature-based Representation (FB): Investigating Parameter D

The complete results of these experiments are presented in Appendix D.

In Figure 4.5, a summarization of results is presented as the average error rate of k-NN search of all datasets across the varying values of D. It is clear that the best average performance is achieved when D equals to 1, whereas for larger values there is a relatively stable performance.

In Table 4.2 the minimum error rate (the best) is presented for each dataset along with the corresponding value of D at which it is achieved. In 9 out of 20 datasets, the best performance is achieved when D equals to 1, whereas in the remaining datasets the corresponding values of D vary from 2 to 20. However, when D is different than 1, the

improvement seems to be essential in 4 datasets, specifically in OSULeaf, Swedishleaf, Wafer and Beef.

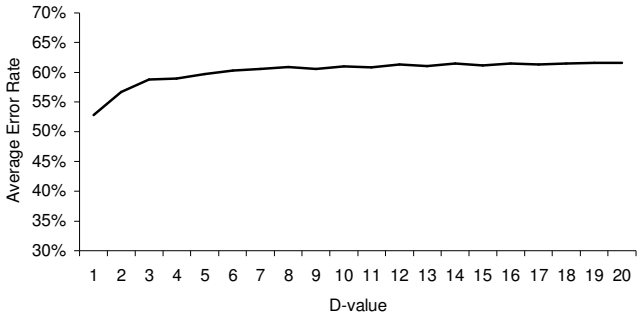


Figure 4.5 k-NN Search: Average Error-Rate for 20 datasets / D-value

Table 4.2 k-NN Search: Minimum Error-Rate

Dataset	Min. Error Rate*		D
Adiac	64.87		1
50words	79.79	(80.16)	4
CBF	38.83		1
ECG200	38.90		1
FaceAll	68.71		1
FaceFour	60.07	(60.43)	18
Fish	79.25	(79.65)	2
GunPoint	48.32	(48.97)	2
Lighting2	47.56	(48.96)	17
Lighting7	65.98	(66.36)	2
OSULeaf	44.62	(62.59)	2
SwedishLeaf	48.90	(54.17)	2
SyntheticControl	43.43		1
Trace	27.20		1
TwoPatterns	62.07		1
Wafer	12.65	(16.42)	3
Yoga	46.95	(48.68)	20
Beef	64.70	(70.00)	11
Coffee	25.25		1
OliveOil	50.21		1

*Numbers in parentheses present the error rate when D=1.

Similar observations can be made, when 1-NN classification is applied. When D equals 1, the average error rate for all datasets is not only the lowest but it seems to be much lower than the average error rate observed for other values of D (Figure 4.6). In addition to that, it seems that when D increases, the performance deteriorates.

In Table 4.3 the minimum error rate (the best) is presented for each dataset along with the corresponding value of D at which it is achieved. In 14 out of 20 datasets, the best performance is achieved when D equals to 1, whereas in the remaining datasets the corresponding values of D vary from 2 to 13.

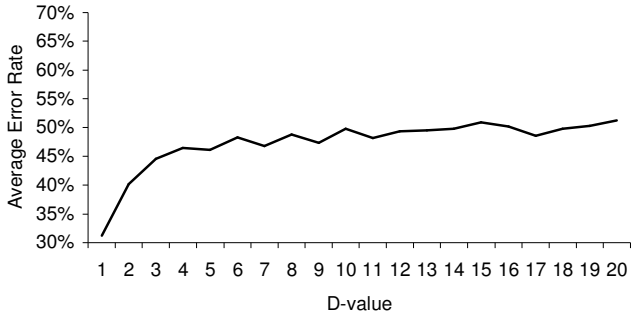


Figure 4.6 1-NN Classification: Average Error-Rate for 20 datasets / D-value

Table 4.3 1-NN Classification: Minimum Error-Rate

Dataset	Min. Error Rate*	D
Adiac	46.55	1
50words	69.23	1
CBF	16.56	1
ECG200	23.00	1
FaceAll	40.36	1
FaceFour	37.50	1
Fish	39.43	1
GunPoint	22.00	1
Lighting2	34.43	1
Lighting7	57.53 (64.38)	2
OSULeaf	19.01 (45.45)	2
SwedishLeaf	18.72 (23.20)	2
SyntheticControl	13.33	1
Trace	11.00	1
TwoPatterns	29.08	1
Wafer	0.26 (2.55)	3
Yoga	21.53 (25.67)	2
Beef	26.67 (43.33)	13
Coffee	3.57	1
OliveOil	33.33	1

*Numbers in parentheses present the error rate when D=1.

4.4.2 D-PAA: Comparison of Distance Measures

In Table 4.4, the minimum error rates are presented along with the corresponding weights (w) and dimensionalities (dim), when k-NN search is performed.

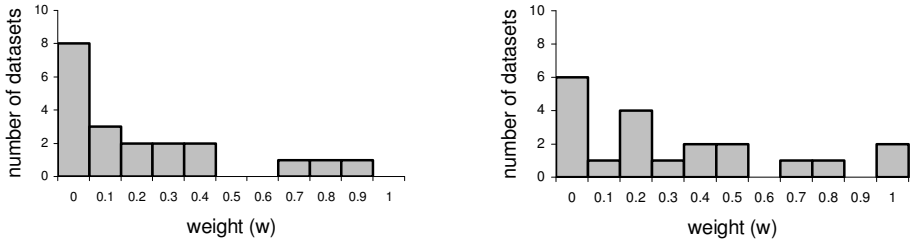
The first observation is that, normalization of features does not affect the error rate either consistently across datasets or significantly (more than 1 unit) in the majority (16) of datasets. There are four datasets, however, where the error rate is affected by more than 1 unit, when the normalized features are utilized. More specifically, in SwedishLeaf dataset the error rate is decreased by 1.29 units, in Trace by almost 8 units and in wafer by 2.75 units. There is only one dataset, Coffee, where the error rate is increased by almost 6 units.

The second observation is that the minimum error rates are achieved for weight values, which are lower than 0.5 in the majority of datasets (Figure 4.7). Moreover, the weight value of zero appears with the highest frequency in both cases of original (Figure 4.7a) and normalized features (Figure 4.7b). This implies that in most of the cases, when standard deviations are assigned greater weight than means, the error rates are improved (even by little). Note that assigning a weight value equal to 0.5 is equivalent to no weight. In these experiments, the value of 0.5 appears twice as the optimal weight, only when normalized features are utilized.

Table 4.4 k-NN Search: Minimum Error Rates (%)

Features	Original			Normalized		
Dataset	D-PAA	w	dim	D-PAA	w	dim
Adiac	63.27	0.0	14	63.31	0.2	14
50words	59.40	0.3	12	59.57	0.4	12
CBF	30.12	0.0	6	30.09	0.2	4
ECG200	31.56	0.0	12	32.77	0.0	12
FaceAll	49.30	0.4	14	48.92	0.2	14
FaceFour	33.80	0.1	14	34.97	0.4	12
Fish	59.23	0.0	14	59.07	0.2	14
GunPoint	44.38	0.0	16	44.92	0.1	8
Lighting2	40.30	0.0	4	40.53	0.0	4
Lighting7	52.04	0.7	10	52.75	1.0	12
OSULeaf	70.67	0.4	14	70.80	0.7	14
SwedishLeaf	52.64	0.2	14	51.35	0.5	16
SyntheticControl	33.74	0.9	10	35.47	0.5	6
Trace	46.36	0.1	10	38.40	0.0	10
TwoPatterns	67.30	0.8	10	66.13	0.8	10
Wafer	6.70	0.0	8	3.95	0.0	8
Yoga	48.03	0.0	4	48.03	0.0	4
Beef	65.45	0.3	8	64.70	1.0	16
Coffee	24.34	0.1	16	30.28	0.0	16
OliveOil	29.89	0.2	12	29.42	0.3	14

Finally, normalization does not affect essentially dimensionality reduction. In particular, there are only three datasets, where the minimum error rate is achieved for a significantly different dimensionality (GunPoint, SyntheticControl and Beef).



a) original features

b) normalized features

Figure 4.7 k-NN Search: Distribution of Weights

Similar observations are made, when 1-NN classification is performed. Table 4.5 presents the minimum error rates along with the corresponding weights (w) and dimensionalities (dim).

Table 4.5 1-NN Classification: Minimum Error-Rates

Features	Original			Normalized		
Dataset	D-PAA	w	dim	D-PAA	w	dim
Adiac	40.15	0.1	14	39.13	0.1, 0.2	14
50words	31.43	0.5	16	29.45	0.6	16
CBF	0.11	0.1, 0.2	6	0.33	0.2, 0.3	6
ECG200	8.00	0.2, 0.3	8	8.00	0.2, 0.9	8, 14
FaceAll	25.03	0.7	14	25.09	0.4	14
FaceFour	13.64	0.1-0.3, 0.9	12-16	9.09	0.3	14
Fish	25.71	0.3	12	25.14	0.2	14
GunPoint	6.00	0.1, 0.2	14	7.33	0.0	8
Lighting2	11.48	0.3	6	14.75	0.1, 0.2, 0.3	6, 8
Lighting7	27.40	0.2-0.5	14, 16	23.29	0.4, 0.5	16
OSULeaf	47.11	0.1	14	47.11	0.4	16
SwedishLeaf	12.48	0.2	14	13.92	0.5	12
SyntheticControl	3.67	0.2, 0.3, 0.5, 0.6	16	3.33	0.7	16
Trace	4.00	0.1, 0.3	12, 14	2.00	0.1-0.6	16
TwoPatterns	5.42	0.7	16	5.52	0.8	16
Wafer	0.36	0.7	14	0.39	0.6, 0.8	14
Yoga	16.67	0.2	16	16.13	0.8	16
Beef	26.67	0.7-1.0	12	23.33	0.8-1.0	16
Coffee	0.00	0.1-0.7	16	0.00	0.1-0.7	14, 16
OliveOil	13.33	0.0-0.2, 1.0	10, 12	13.33	0.3, 0.4	16

Again, normalization of features does not affect the error rate either consistently across datasets or significantly. Although there are more datasets (9) where the difference is greater than 1 unit, this differences ranges from 1 to 4.6 units. Also, normalization does not affect essentially dimensionality reduction. In particular, there are only three datasets, where the minimum error rate is achieved for a significantly different dimensionality (GunPoint, Beef and OliveOil).

On the other hand, the minimum error rates are achieved at weight values that are not unique for each dataset. The distribution of weights is not as skewed to the lower values, as it was in the previous set of experiments. Moreover, their values often change significantly with respect to the application of normalization. Again, the value of 0.5 (no weight) appears in only 2 datasets.

4.4.3 Comparison between D-PAA and PAA

Table 4.6 presents the minimum error rates and dimensionalities (dim) of a k-NN search, when PAA and D-PAA representations are applied on data.

Table 4.6 k-NN Search: Minimum Error Rates (%)

Features			Original		Normalized	
Dataset	PAA	dim	D-PAA	dim	D-PAA	dim
Adiac	70.81	14	63.27	14	63.31	14
50words	62.83	12	59.40	12	59.57	12
CBF	40.61	6	30.12	6	30.09	4
ECG200	32.63	14	31.56	12	32.77	12
FaceAll	66.54	12	49.30	14	48.92	14
FaceFour	41.97	6	33.80	14	34.97	12
Fish	64.97	16	59.23	14	59.07	14
GunPoint	45.73	6	44.38	16	44.92	8
Lighting2	44.49	4	40.30	4	40.53	4
Lighting7	53.53	6	52.04	10	52.75	12
OSULeaf	69.99	14	70.67	14	70.80	14
SwedishLeaf	59.36	14	52.64	14	51.35	16
SyntheticControl	31.23	10	33.74	10	35.47	6
Trace	49.70	4	46.36	10	38.40	10
TwoPatterns	68.76	6	67.30	10	66.13	10
Wafer	15.24	4	6.70	8	3.95	8
Yoga	49.62	8	48.03	4	48.03	4
Beef	66.06	6	65.45	8	64.70	16
Coffee	27.51	16	24.34	16	30.28	16
OliveOil	30.45	14	29.89	12	29.42	14

The first observation is that D-PAA representation produces better results than PAA in the majority of datasets. In 16 out of 20 datasets, the improvement in error rate ranges approximately from 0.5 to 17 units, independently of whether normalization is applied or not. When PAA is compared with D-PAA with normalization, the largest differences appear in FaceAll (17.62 units), Trace (11.30 units), Wafer (11.29 units) and CBF (10.52 units). In 2 out of 20 datasets (ECG200 and OSULeaf) PAA performs similarly with D-PAA, whereas in the remaining 2 datasets, SyntheticControl and Coffee, PAA has an error rate that is lower by 4.24 and 2.77 units respectively.

The fact that, PAA may produce the "best" results in a lower dimensionality than D-PAA, cannot be always considered as an advantage, since in higher dimensionalities may produce worse results than D-PAA. In these experiments, there are only 4 datasets (GunPoint, Lighting7, TwoPatterns and Beef), where PAA produces similar error rates with D-PAA in significantly lower dimensionality.

Table 4.7 presents the minimum error rates and dimensionalities (dim) of a 1-NN classification, when PAA and D-PAA representations are applied on data.

Table 4.7 1-NN Classification: Minimum Error-Rates

Features			Original		Normalized	
Dataset	PAA	dim	D-PAA	dim	D-PAA	dim
Adiac	51.66	14	40.15	14	39.13	14
50words	33.63	12	31.43	16	29.45	16
CBF	3.11	10	0.11	6	0.33	6
ECG200	11.00	12	8.00	8	8.00	8, 14
FaceAll	32.19	16	25.03	14	25.09	14
FaceFour	18.18	16	13.64	12-16	9.09	14
Fish	28.00	14	25.71	12	25.14	14
GunPoint	8.00	6	6.00	14	7.33	8
Lighting2	18.03	8	11.48	6	14.75	6, 8
Lighting7	27.40	16	27.40	14, 16	23.29	16
OSULeaf	46.69	16	47.11	14	47.11	16
SwedishLeaf	20.48	14	12.48	14	13.92	12
SyntheticControl	1.00	12	3.67	16	3.33	16
Trace	24.00	4	4.00	12, 14	2.00	16
TwoPatterns	6.50	16	5.42	16	5.52	16
Wafer	0.47	8	0.36	14	0.39	14
Yoga	17.27	12	16.67	16	16.13	16
Beef	26.67	6	26.67	12	23.33	16
Coffee	3.57	14	0.00	16	0.00	14, 16
OliveOil	13.33	6	13.33	10, 12	13.33	16

The first observation is that D-PAA representation produces better results than PAA in the majority of datasets. In 15 out of 20 datasets, the improvement in error rate ranges from approximately 0.08 to 20 units, independently of whether normalization is applied or

not. When PAA is compared with D-PAA with normalization, the largest differences appear in Trace (22 units), Adiac (12.53 units), FaceFour (9.09 units) and FaceAll (7.10 units). In 1 out of 20 datasets (OliveOil) PAA performs similarly with D-PAA, whereas in 2 datasets OSULeaf and SyntheticControl, PAA has an error rate lower by 0.42 and 2.33 units respectively.

Regarding the dimensionalities, there are 5 datasets (CBF, ECG200, FaceAll, FaceFour and Lighting2), where D-PAA produces considerably better results than PAA with fewer dimensions. On the other hand, there are 4 datasets (Wafer, Yoga, Beef and OliveOil), where PAA produces similar error rates with D-PAA in lower dimensionality.

4.4.4 Comparison among D-PAA and other Rival Approaches

Table 4.8 presents the minimum error rates recorded in 1-NN classification, when several representations are applied.

Table 4.8 1-NN Classification: Minimum Error Rates (%)

Features					Original	Normalized
Dataset	SVD	DFT	FB	PAA	D-PAA	D-PAA
Adiac	39.1	43.7	46.6	51.7	40.2	39.1
50words	34.3	33.9	69.2	33.6	31.4	29.5
CBF	4.9	3.3	16.6	3.1	0.1	0.3
ECG200	12.0	11.0	23.0	11.0	8.0	8.0
FaceAll	31.9	27.3	40.4	32.2	25.0	25.1
FaceFour	20.5	17.1	37.5	18.2	13.6	9.1
Fish	21.1	22.9	39.4	28.0	25.7	25.1
GunPoint	8.0	8.7	22.0	8.0	6.0	7.3
Lighting2	18.0	19.7	34.4	18.0	11.5	14.8
Lighting7	32.9	28.8	57.5	27.4	27.4	23.3
OSULeaf	46.3	48.4	19.0	46.7	47.1	47.1
SwedishLeaf	17.4	17.8	18.7	20.5	12.5	13.9
SyntheticControl	1.0	1.0	13.3	1.0	3.7	3.3
Trace	25.0	27.0	11.0	24.0	4.0	2.0
TwoPatterns	5.7	5.5	29.1	6.5	5.4	5.5
Wafer	0.5	0.4	0.3	0.5	0.4	0.4
Yoga	17.2	17.4	21.5	17.3	16.7	16.1
Beef	33.3	33.3	26.7	26.7	26.7	23.3
Coffee	0.0	0.0	3.6	3.6	0.0	0.0
OliveOil	10.0	16.7	33.3	13.3	13.3	13.3

The first observation is that D-PAA produces the minimum error rate in the majority of the datasets either when normalization is applied or not. In the first case, D-PAA improves error rate in 12 datasets, performs exactly the same in 3 datasets and worse in 5 datasets.

When it is compared with the next "best" representation, the improvement ranges from 0.7 to 9 units, whereas, the deterioration ranges from 0.1 to 28.1 units. Similarly, in the latter case, D-PAA improves error rate in 11 datasets, performs exactly the same in 3 datasets and worse in 6 datasets. When it is compared with the next "best" representation, the improvement ranges from 0.1 to 7 units, whereas, the deterioration ranges from 0.1 to 28.1 units.

The second observation is that all other representations perform better only in much fewer cases. The SVD and FB representations perform better than all the other representations in two datasets each, whereas, DFT and PAA provide similar or worse results than SVD, FB and D-PAA in all datasets.

Table 4.9 presents the minimum error rates recorded in 1-NN classification, when Euclidean and DTW distances are applied on raw (no representation) data.

The first observation is that D-PAA (with or without normalization) and DTW perform much better than Euclidean distance, which produces similar or negligibly better results in only three datasets.

Table 4.9 1-NN Classification: Minimum Error Rates (%)

Features			Original	Normalized
Dataset	ED	DTW	D-PAA	D-PAA
Adiac	38.9	39.6	40.2	39.1
50words	36.9	31.0	31.4	29.5
CBF	14.8	0.3	0.1	0.3
ECG200	12.0	23.0	8.0	8.0
FaceAll	28.6	19.2	25.0	25.1
FaceFour	21.6	17.0	13.6	9.1
Fish	21.7	16.7	25.7	25.1
GunPoint	8.7	9.3	6.0	7.3
Lighting2	24.6	13.1	11.5	14.8
Lighting7	42.5	27.4	27.4	23.3
OSULeaf	47.9	40.9	47.1	47.1
SwedishLeaf	21.1	21.0	12.5	13.9
SyntheticControl	12.0	0.7	3.7	3.3
Trace	24.0	0.0	4.0	2.0
TwoPatterns	9.3	0.0	5.4	5.5
Wafer	0.5	2.0	0.4	0.4
Yoga	17.0	16.4	16.7	16.1
Beef	33.3	36.7	26.7	23.3
Coffee	0.0	0.0	0.0	0.0
OliveOil	13.3	16.7	13.3	13.3

The second observation is that, overall, the performances of D-PAA and DTW seem to be comparable to each other. The first produces the minimum error rate in the majority of the datasets when normalization. D-PAA improves error rate in 11 datasets, performs exactly

the same in 2 datasets and worse in 7 datasets. The improvement ranges from 0.3 to 15 units, whereas, the deterioration ranges from 1.7 to 8.4 units. On the other hand, when normalization is not applied, D-PAA improves error rate in 9 datasets, performs exactly the same in 2 datasets and worse in 9 datasets. The improvement ranges from 0.2 to 15 units, whereas, the deterioration ranges from 0.3 to 9 units.

4.5 Conclusion

In this chapter, we introduce a novel time series representation, named D-PAA, along with a distance measure that lower bounds the Euclidean distance.

The first conclusion is that D-PAA, when compared to the most similar representation PAA, performs considerably better in the majority of the datasets without necessarily sacrificing the required dimensionality. In addition to that, assigning a weight to the features improves the performance of D-PAA with respect to k-NN search and 1-NN classification. Moreover, experiments indicate that local dispersion present in a time series possesses a further discriminating power in conjunction with the corresponding central tendency.

A second conclusion is that D-PAA, when compared to other representations, performs consistently better in the majority of datasets with respect to classification accuracy. The minimum error rate is consistently similar or better throughout the majority of datasets. In addition to that, D-PAA performance is comparable to DTW, which constitutes a state of the art approach in measuring similarity among time series. In fact, there are datasets where D-PAA outperforms DTW and vice versa.

A third conclusion refers to the Feature-based representation, which also utilizes the standard deviation. The results of the conducted experiments indicate that FB performs considerably better when the required parameter is set equal to one. Also, FB outperforms other representations in 2 out of 20 datasets in the context of kNN similarity search and 1NN classification.

An expected conclusion that can be drawn from the experiments of the previous section is that there is no representation scheme that outperforms all the others in every dataset. This conclusion refers mainly to their capability of capturing the most essential features of a time series, that is, to minimize the loss of important information inherent in a time series, while reducing dimensionality. The D-PAA representation is proposed for the purpose of improving the quality of a time series that has been transformed into a space of reduced dimensionality.

CHAPTER 5

Cluster Based Similarity Search

5.1 Introduction

Similarity search is an important task in many applications, such as content-based retrieval, exploratory data analysis, predictive modeling and data mining. The basic problem can be stated as follows: given a set of objects, find the most similar ones to a given query object. For example, one may be interested in retrieving the most similar images to a given one from a database or in identifying those stocks whose prices evolved similarly to a specific one over the last year. Retrieval of these objects is based on "similarity" rather than on "exactness".

The main research in this area is focused on the development of methods that can efficiently support similarity search, since common applications involve a very large amount of data. The volume of data depends not only on the number of objects but also on their dimensionality. Typically, an object can be considered as a point in an n-dimensional Euclidean space.

One major class of methods for efficient similarity search comprises of multidimensional indexing schemes that can be used for fast access of these points [51]. Examples include the R-tree [58], TV-tree [106], X-tree [14] and R+-tree [134]. In general, the tree-like schemes divide an N-dimensional space into overlapping subregions that contain subsets of objects. When a query object arrives, it is driven to one of these subregions, and then, a nearest neighbor search is performed in order to locate similar objects that may reside not only in this subregion but also in neighboring ones. Although the indexing approach can be extremely fast, its efficiency degrades rapidly with the increase of the dimensionality. There are several research results that demonstrate the negative effects of increasing dimensionality on index structures [62] [155]. For instance, White & Jain [157] report that as the dimensionality increases from 5 to 10, the performance of a nearest neighbor search degrades by a factor of 12 for various multidimensional index structures. Beyer et al. [18] show that nearest neighbor search can become unstable with as few as 10-20 dimensions. This phenomenon, known as *dimensionality curse*, implies that a simple sequential scan usually performs better at higher dimensionalities than index structures.

One solution for achieving efficient similarity search in the presence of high dimensionality is to condense the data by applying a dimensionality reduction technique (i.e. Principal Component Analysis). The idea is to map the original data into a lower dimension domain without losing substantial amount of information. The approach of dimensionality reduction can be very helpful in several ways. The fact that it reduces the storage requirements directly affects the performance scalability. Also, this approach may result into a number of dimensions that allow efficient implementation of multidimensional indexing structures. In addition, there are many applications where the dimensionality reduction improves the performance of similarity search with respect to the quality of the results. For example, a dimensionality reduction technique that is capable of removing high levels of noise present in data may improve the quality of similarity search.

In this chapter, we examine the case of time series similarity search within the context of one nearest neighbor (1NN) classification. Time series data differ from other domains in that they have an intrinsically high dimensionality, which necessitates the application of a dimensionality reduction technique. Moreover, the results from Chapter 4 directly indicate that the performance of similarity search is improved when the original time series is mapped into a lower dimension domain. The method of 1NN classification requires searching a database for the most similar object (time series) to a given one. The main drawback of this method is that we have to compare a query object to every object in a database in order to find the most similar one [59]. This approach becomes prohibitive, when the reference database is extremely large. The efficiency of this method is affected by the number of objects in the database, as well as, by the dimensionality of these objects, since a distance measure is calculated for measuring the closeness of the corresponding objects.

We propose to reduce the dimensionality of the original time series for the purpose of accelerating the computation of the chosen distance measure and improving the quality of results. In addition to that, we propose to partition the dimensionality reduced database into clusters of similar objects, and apply 1NN searching on each cluster in a sequential manner. The rationale behind our approach is that we may reach the nearest neighbor without searching the whole database. A multidimensional index structure can be built on each one of the clusters in order to efficiently apply similarity search. However, we investigate the effectiveness of our approach with respect to sequential searching, since the required dimensionality reduction in order to ensure desired quality performance may require a multidimensional indexing that is not essentially faster than sequential scanning.

The main contributions made in this Chapter are:

- The introduction of a new similarity search method for time series that is faster than sequential search, and potentially, than indexing-based search.
- The experimental evaluation of a clustering based approach on data that is in the form of time series.

In Section 5.2, the background of clustering is presented along with related work. In Section 5.3, our approach is presented in detail. The experimentation framework is described in Section 5.4, whereas in Section 5.5 the corresponding results are presented. Finally, conclusions are provided in Section 5.6.

5.2 Background

In this section we present a brief review of clustering and we discuss related work in similarity search.

5.2.1 Clustering

Clustering is the process of constructing meaningful partitions (clusters) of a large set of objects based on the characteristics they possess. These clusters should exhibit high within-cluster homogeneity and high between-cluster heterogeneity. In other words, objects in the same cluster should be as similar as possible to each other, while being as different as possible from objects in other clusters. There is a huge number of clustering algorithms proposed in the literature, since cluster analysis has been studied for several decades in statistics [60] [8] and machine learning [68].

When applying clustering on a dataset, one should address three key issues: the definition of a similarity measure, the method of forming the clusters and the number of clusters. Since objects are multidimensional, an appropriate similarity measure should be defined in order to capture the "closeness" between two objects. Two broad categories of clustering algorithms are the hierarchical and partitioning methods. According to the first category, the procedure starts with an initial clustering in which each cluster consists of a single object. At each step, the two closest clusters are merged, until just one cluster, of all objects, is formed. This procedure is called agglomerative in contrast with the divisive one, which starts with a single cluster composed of all objects and proceeds to divide it into successively smaller clusters. The process ends with a partition in which each cluster consists of a single object. On the other hand, the partitioning methods are nonhierarchical procedures that require the specification of the number of clusters to be formed. The procedure starts with the selection of cluster seeds as the initial cluster centers. The next task is the assignment of each object to the nearest seed and the recalculation of the cluster

centers. Objects then may be reassigned if they are closer to another cluster than the one originally assigned. Our work involves the k-means method, which is one of the most popular partitioning methods, and it is analytically described in Section 5.3.

Clustering algorithms should follow specific properties within data mining, one of which is the capability of working with high dimensional data. The objects in data mining could have hundreds of attributes. Clustering in such high dimensional spaces presents two basic problems. First, with an increase in the number of attributes that characterize an object, the likelihood of presence of irrelevant attributes increases. The second problem, known as the dimensionality curse, is the increasing difficulty of distinguishing objects based on their distances. Beyer et al. [18] show that the distance of a query to the nearest neighbor approaches the distance to the farthest point as the dimensionality increases. For interesting insights into complications of high dimensional data, see [2].

There are several approaches in dealing with high dimensional data. Traditional methods of dimensionality reduction, such as Fourier Transforms or Singular Value Decomposition, transform original data into a lower dimension domain by applying some function of the initial attributes. On the other hand, algorithms of subspace clustering try to circumvent high dimensionality by building clusters in appropriate subspaces of original attribute space. Another approach, co-clustering, divides attributes into similar groups and comes up with new derived attributes representing each group. Co-clustering can be considered as a procedure that simultaneously clusters both objects and their attributes. Our work follows the approach of dimensionality reduction because the data under consideration is in the form of time series, which have an intrinsically high dimensionality and high attribute correlation. A general survey on clustering analysis can be found in [15].

5.2.2 Related Work

As it is mentioned earlier, the dimensionality curse has serious effects on the effectiveness of high-dimensional similarity search from the performance perspective. The most obvious solution for achieving scalable performance is to reduce the dimensionality of data. Aggarwal [3] provides a model of the effects of this reduction on high dimensional problems for the purpose of improving the quality of similarity search.

Most of the research is focused on reducing the dimensionality of data in order to apply an effective multidimensional indexing structure. Chakrabarti and Mehrotra [25] propose to discover correlated clusters in the dataset and reduce the corresponding dimensionalities by applying Principal Component Analysis to each one of them. They also provide a technique to individually index these clusters that guarantees no false positive or

false negative answers to be returned to the user. This approach applies local dimensionality reduction assuming that there are subsets of locally correlated data.

Bennett et al. [13] propose to apply an EM (Expectation Maximization) algorithm to cluster data using a mixture-of-Gaussians pdf (probability density function) model. Each cluster corresponds to a specific Gaussian pdf, which is parameterized with a mean vector and a covariance matrix. This approach (DBIN) utilizes the derived density model of the data in order to introduce an indexing scheme that produces a mapping between a query point and an ordering on the clustered index values. The authors provide stability conditions under which DBIN performs optimally and means of detecting when their indexing structure is unlikely to be useful.

Ferhatosmanoglou et al. [47] introduce a new technique based on clustering that reduces the size of dataset and the dimensionality of each data object. The authors propose to transform the original d-dimensional data by using the Karhunen-Loève Transformation (KLT). The derived dataset has the same dimensionality with the original one; however, most of the information is accumulated in the first few dimensions. After dimensionality reduction, a modified K-means clustering algorithm is applied in the low dimensional domain. The number of dimensions to be retained can be determined based on a statistical analysis of data. The procedure of approximate nearest neighbor searching starts with the transformation of the query object, the retention of the first r dimensions, and the identification of the cluster this query falls into. Then, similarity search is performed in this cluster and the k nearest neighbors are retrieved. The authors propose a progressive refinement of the process by increasing the number of the retained dimensions and/or by searching neighboring points that fall in other clusters.

Another cluster-based approach is introduced in [143]. The authors propose an indexing method called Clustering with Singular Value Decomposition (CSVD) that efficiently supports approximate nearest neighbor queries. The construction of a CSVD index involves three steps: partitioning the dataset using a clustering technique (i.e. K-means), applying SVD separately on each cluster to reduce the dimensionality of data, and constructing an index for the transformed space. The authors provide a recursive extension of CSVD in order to deal with the expensive computation cost of constructing the proposed index. Similar approaches with CSVD are introduced in [139] [24].

Li et al. [99] provide a clustering and indexing paradigm (called CLINDEX) for high dimensional spaces. The CLINDEX partitions the dataset into clusters. Each cluster is represented by a separate file and all files are sequentially stored on disk. The authors introduce a new technique for constructing clusters of objects. In particular, the algorithm

starts by dividing each dimension of the d-dimensional vector space into 2^n segments, so every segment can be identified using an n-bit number. This process forms $(2^n)^d$ cells in the data space. The clustering algorithm aggregates these cells into clusters. Once the clusters are obtained, an indexing structure is built that is based on a simple encoding scheme which maps an object to a cell and a cell to its corresponding cluster. Approximate similarity search is processed by first identifying the cluster to which the query object belongs, and then by searching the corresponding file.

Zezula et al. presents various approaches in developing index structures for searching complex data modeled as instances in a metric space [163].

5.3 A Proposed Approach

In this chapter, we examine the case of time series similarity search within the context of one nearest neighbor (1NN) classification. The objective of our work is to provide a method for the purpose of improving the efficiency of 1NN classification without sacrificing the quality of the results. We introduce an approach that integrates the dimensionality reduction of data and the reduction of the search space by applying clustering on the transformed data. We call this approach CLUREP (Clustering on Representation). In particular, the original data is represented in a compressed form by applying any dimensionality reduction technique, such as Discrete Fourier Transform, Singular Value Decomposition or Piecewise Aggregate Approximation. Then, the transformed dataset is partitioned by applying a clustering algorithm. The similarity search proceeds at each cluster sequentially according to specific criteria. The idea is that we may reach the nearest neighbor without necessarily visiting all clusters. The proposed method consists of three phases that are analytically described hereafter.

The first phase involves the application of a dimensionality reduction technique on the original data for two reasons. First, we expect to improve the quality (accuracy) of the classification results, as it has been experimentally shown in the previous chapter. Second, the subsequent clustering analysis becomes more effective, since the problem of high dimensional data is alleviated. Virtually, any technique can be selected at this step; however, this choice is application dependent, since different techniques may result into representations of higher quality in different applications. In our method, the quality of the representation of the original data affects the quality of the subsequent clustering analysis, which in turn affects the efficiency of similarity search. However, in this work we propose to apply the Piecewise Aggregate Approximation (PAA) because it is simple, fast to calculate,

and it has been shown empirically that it is as efficient as other more sophisticated approaches.

The second phase involves the application of a clustering algorithm on the transformed data. In this work, we propose the application of K-means [109], which is one of the most researched and popular clustering methods [68]. Although there are many variations of this algorithm, it can be generally defined as follows.

1. Decide the desired number of clusters k .
2. Select k points to be the seeds for the centroids of the k clusters.
3. Assign each object to the cluster whose centroid is closest to the object, forming in this way k exclusive clusters of objects. Most often, the distance measure that is utilized is the Euclidean distance.
4. Recalculate the new centroids of the clusters.
5. Repeat Steps 3 and 4 until convergence is achieved, that is until there are no new assignments.

Note that the two previous phases constitute the pre-processing that is executed off-line. The third phase involves the procedure of similarity search in the derived clusters. The input to this phase is the cluster membership of each object, the centroid of each cluster (c_i) and the corresponding radius (r_i). The radius of a cluster is defined as the distance of the farthest object of a cluster to the corresponding centroid. Given a query object (q), the method proceeds as follows:

1. Calculate the distances of the query object to the centroids of the clusters ($d\left(q,c_i\right)$).
2. Set the cluster with the closest centroid as the current cluster. Let denote this cluster $C^{(i)}$, where $i=1$.
3. Calculate the distance of the query object to each one of the other clusters. This distance is defined to be the difference between the distance of the query to the centroid and the corresponding radius. If the query object lies within the cluster, then this distance is set equal to zero (Eq. 5.1).

$$d\left(q,C^{(i)}\right)=max\left\{0\ ,\ d\left(q,c_i\right)-r_i\right\},\ \ i=2,3,...,k \tag{5.1}$$

4. The clusters are sorted in an increasing order with respect to their distances from the query object. Ties may be broken according to the distances of the centroids of the clusters to the query object. Let denote these clusters $C^{(i)}$, where $i = 2,3,...,k$ with k corresponding to the cluster that is farthest from the query.
5. Search the current cluster sequentially in order to locate the current nearest neighbor (*nn*) to the query object. Record the corresponding distance $d(q,nn)$
6. If the distance of the query object to the current nearest neighbor is less than or equal to the distance of the query object to the next cluster (Eq. 5.2), then the actual nearest neighbor is found and the algorithm stops.

$$d(q,nn) \le d\left(q,C^{(i+1)}\right) \quad \text{(Eq 5.2)}$$

   Otherwise, let $i = i+1$ and move to the next step.
7. Calculate the difference between the distance of the query object to the centroid of $C^{(i)}$ and the distance of the query object to the current nearest neighbor. If this difference is positive, then search the objects of $C^{(i)}$ that their distance to c_i is greater than this difference. Otherwise, search the whole cluster. After this, the current nearest neighbor (*nn*) has been changed. If $i = k$, the algorithm stops. In this step, we determine a lower bound on the distances of objects from the centroid in order to reduce the search space in the current cluster (Eq. 5.3). In other words, we search the nearest neighbor among the objects that lie within the current cluster and their distances from the corresponding centroid are greater than the lower bound.

$$lower\_bound = max\left\{0\ ,\ d(q,nn) - d(q,c_i)\right\} \quad \text{(Eq. 5.3)}$$

8. Go to step 6.

An example of this procedure is presented in Figures 5.1-5.3. We consider 17 time series of 2 dimensions (two time instances). For presentation purposes, there are two special cases in this example. First, the resulting clustering derives three clusters that do not overlap to each other. In other words, the distance between the centroids is greater than the sum of the corresponding radii, for any pair of clusters. Second, the query object lies outside of the clusters. In general, we do not expect to have such kind of placement of the clusters and the query, especially when the dimensionality is high and the size of the dataset is large. The proposed method works for any placement of clusters and queries.

Our work is very similar to the work of Thomasian et al. [143] that is briefly presented in Section 5.2.2. Nevertheless, we examine the case of time series data, which differ from other domains because they exhibit high dimensionality, high feature correlation and high levels of noise. Also, we investigate the effectiveness of our approach with respect to sequential searching, since the required dimensionality reduction in order to ensure desired quality performance may require a multidimensional indexing that is not essentially faster than sequential scanning. The selection of the specific technique for dimensionality reduction is an open choice; however, its application precedes the clustering task. Finally, the search of the clusters (except from the primary one) is restricted only to that space where the nearest neighbor (if exist) lies.

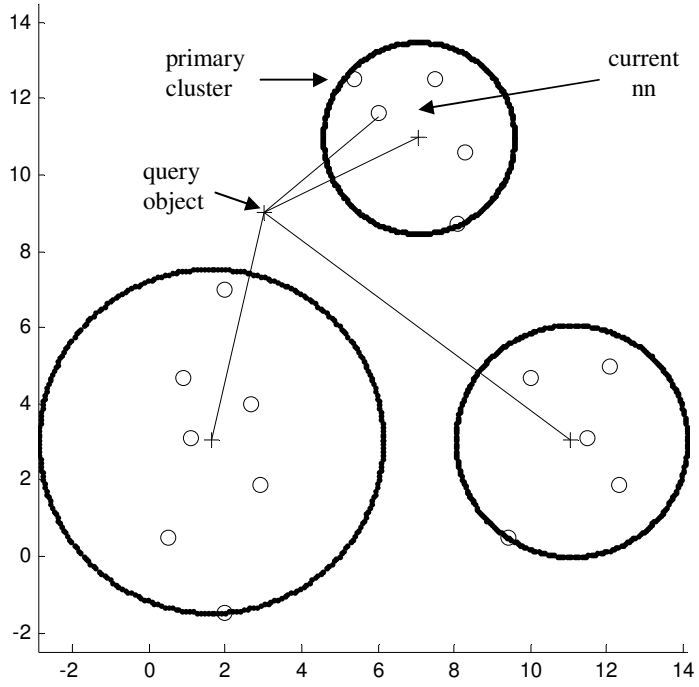


Figure 5.1 The determination of the primary cluster and the location of the current nearest neighbor

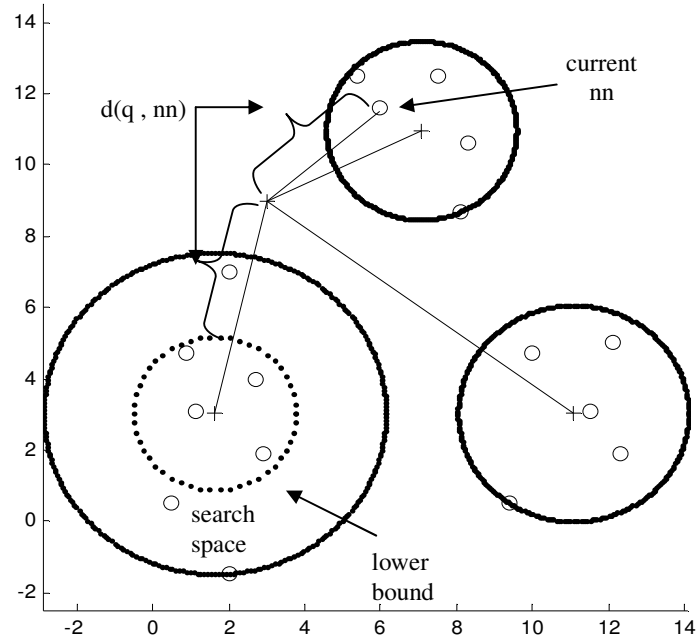


Figure 5.2 The determination of the next closest cluster, the check of whether a better nearest neighbor may exist (in this case, it exists) and the determination of the corresponding search space

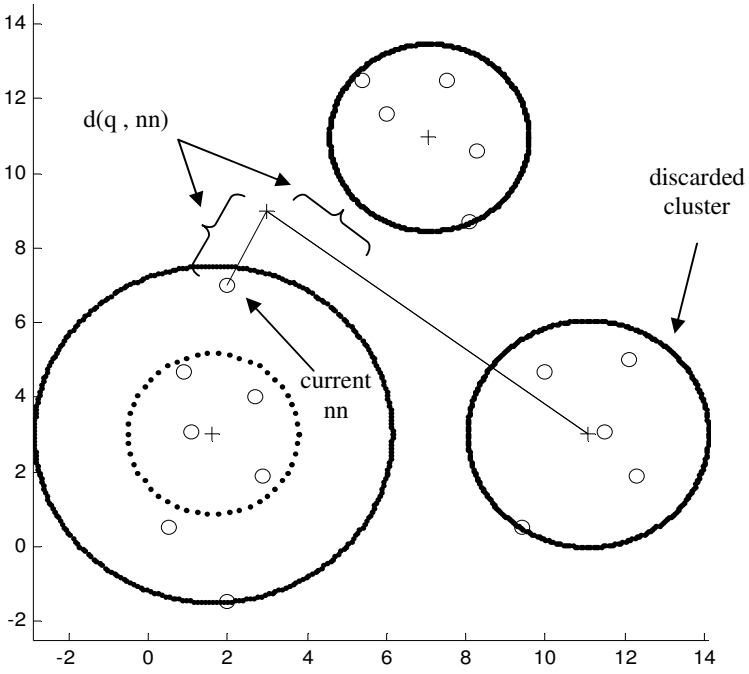


Figure 5.3 The location of a better nearest neighbor and the check of whether a better nearest neighbor may exist (in this case, it does not exist and the algorithm ends without searching the next cluster)

5.4 Framework of Experimentation

In order to evaluate the performance of the proposed approach, we perform one-nearest neighbor classification (1-NN) and validate it with the leave-one-out procedure. Each series of a dataset is considered as the query object and is classified according to the class of its nearest neighbor. We record the classification accuracy and the percentage of the volume of data that is searched. The corresponding average values are calculated over the total number of series in the dataset.

The experiments are conducted on 12 real world and synthetic datasets, which are available upon request in [64]. They are utilized extensively as benchmark datasets for testing classification algorithms and for this reason, they are separated in training and testing sets. In this work, we merge these two sets in order to increase the size of each dataset. All datasets consist of normalized time series, that is, time series of mean equal to zero and standard deviation equal to one. All series are labeled according to the class they belong to. A description of these datasets is presented in Table 5.1.

As it is stated in the previous section, our method involves reducing the dimensionality of data by applying an appropriate technique. We apply Piecewise Aggregate Approximation (PAA), which segments a time series into a number of sections and records the corresponding means. The number of sections is set equal to multiples of 2, ranging from 4 to 16. In this work, we do not experiment with other dimensionality reduction techniques, since there is not a single technique that outperforms all the others in every application. The dimensionality reduction technique affects the quality of clustering, and consequently, the quality of clustering affects the performance of our method. However, this method aims at improving the efficiency of 1-NN classification rather than the quality of clustering.

We compare our method (CLUREP) with the complete sequential search and the method that does not restrict the search space of the visited clusters [143]. According to the latter method, the whole clusters are scanned once visited. We call this method CLUREP_all. We do not provide any comparison with index-based methods because the required dimensionality for achieving the best accuracy rate is experimentally shown that is greater than 10 for the majority of the selected datasets (11 out of 12). This observation implies that a multidimensional indexing may not be essentially faster than sequential scanning.

Finally, the algorithm of K-means is applied with a number of clusters that ranges from 2 to 20 in increments of 2. Intuitively, the more the number of clusters, the less the fraction of dataset that need to be searched.

All the necessary codes and experiments are developed in MATLAB.

Table 5.10 Description of Datasets

ID	DATASET	CLASSES	SIZE (# of time series)	DIMENSIONALITY (length of time series)
1	50words	50	905	270
2	CBF	3	930	128
3	ECG200	2	200	96
4	FaceAll	14	2250	131
5	GunPoint	2	200	150
6	OSULeaf	6	442	427
7	SwedishLeaf	15	1125	128
8	SyntheticControl	6	600	60
9	Trace	4	200	275
10	TwoPatterns	4	5000	128
11	Wafer	2	7164	152
12	Yoga	2	3300	426

5.5 Results

In the first part of the results, we provide the classification error rates and the percentage of the volume of data that is searched for a constant number of clusters ($k = 10$). The objective is to examine their relation under varying dimensionalities.

Table 5.2 presents the classification error rates for varying dimensionalities when the number of the generated clusters is set equal to 10. The main observation is that the lowest error rate is achieved at high dimensionalities. In 8 out of 12 datasets, the required dimensionality is the highest (16). The lowest dimensionality is 8 and it is observed in only one dataset. The minimum error rate ranges from 0.00% to 35.75%.

Table 5.3 presents the percentages of the volume of data that is searched when the number of the generated clusters is set equal to 10. In 8 out of 12 datasets, there is an increasing trend in the volume of data searched as the dimensionality increases. On the other hand, there are 4 datasets (GunPoint, Trace, Wafer, Yoga) where the corresponding volume of data remains fairly stable across the dimensionalities. The minimum volume of data ranges from 12.5% to 19.2%.

The key observation that arises by combining both tables (Table 5.2 and Table 5.3) is that as the dimensionality increases, the classification error rate decreases with the cost of an increasing percentage of the volume of data that need to be searched. We also observe that when the dimensionality is set equal to 16, the corresponding volume of data ranges from 13% to 58.3%. In addition to that, the three datasets with the "worst" error rates also

present poor results in the volume of data that need to be searched (50words, OSULeaf, SwedishLeaf). Similar observations can be made, if we generate fewer or more clusters than 10. For brevity, results for other values of k are not presented.

Table 5.2 1-NN classification error rates (%) ($k = 10$)

		Dimensionality						
ID	Dataset	4	6	8	10	12	14	16
1	50words	65.30	43.87	34.92	31.05	29.61	29.28	27.85
2	CBF	10.54	0.43	0.00	0.11	0.00	0.00	0.00
3	ECG200	25.00	16.50	12.50	12.50	10.50	9.00	10.00
4	FaceAll	59.82	31.73	24.04	13.38	10.00	9.56	7.42
5	GunPoint	17.50	10.50	13.50	7.50	8.00	6.00	5.50
6	OSULeaf	71.95	53.85	44.12	40.27	36.65	36.65	35.75
7	SwedishLeaf	63.38	41.42	27.47	23.73	18.84	18.84	17.33
8	SyntheticControl	14.67	9.00	5.00	1.17	1.17	1.17	0.67
9	Trace	20.00	20.00	22.00	22.50	15.00	20.50	18.50
10	TwoPatterns	45.40	16.84	8.00	3.50	2.82	2.32	0.78
11	Wafer	0.38	0.43	0.18	0.20	0.17	0.14	0.15
12	Yoga	24.36	11.30	9.12	8.30	6.70	7.03	6.58

*Gray areas correspond to the minimum error rate of the corresponding dataset

Table 5.3 Percentages of the volume of data that is searched ($k = 10$)

		Dimensionality						
ID	Dataset	4	6	8	10	12	14	16
1	50words	16.4	21.0	25.6	30.1	33.8	36.7	40.0
2	CBF	14.2	15.6	16.3	17.3	18.8	18.3	20.4
3	ECG200	17.7	20.9	21.9	22.9	24.3	25.8	25.2
4	FaceAll	17.3	26.2	31.8	28.8	28.9	29.5	31.0
5	GunPoint	16.7	16.0	17.5	16.7	17.6	17.1	17.4
6	OSULeaf	19.2	28.8	33.6	43.9	48.5	53.2	58.3
7	SwedishLeaf	19.1	23.7	29.6	32.0	35.3	35.7	39.0
8	SyntheticControl	16.8	19.6	21.3	21.5	25.0	24.6	28.4
9	Trace	14.1	13.4	13.5	12.5	13.7	13.8	13.0
10	TwoPatterns	19.2	19.1	25.1	32.5	37.4	40.9	46.2
11	Wafer	14.5	15.2	14.9	15.2	15.5	15.3	15.5
12	Yoga	15.3	15.1	16.4	16.6	16.9	16.8	17.1

*Gray areas correspond to the minimum volume of data of the corresponding dataset

In the second part of the results, we provide the percentage of the volume of data that is searched for varying number of clusters. The dimensionality is set equal to that

number for which the lowest error rate is achieved (Table 5.2). In general, this number varies across different datasets.

In this set of experiments, we provide results for two methods, namely the CLUREP and the CLUREP_all. Note that the latter differs from the first in that it searches the whole cluster once visited. By definition, CLUREP is expected to provide better results than CLUREP_all. However, one of our objectives is to experimentally quantify the expected improvement.

In Table 5.4 and Table 5.5, the performances of CLUREP and CLUREP_all are presented with respect to the percentage of volume of data that is searched. The first observation is that when the number of the generated clusters (k) increases, the corresponding volume decreases. In general, the rate of decrease is higher when the value of k increases from 2 to 10 than when k increases from 10 to 20 (Fig. 5.4, Fig. 5.5).

The second observation is that both methods perform better than sequential scan. However, CLUREP performs consistently better than CLUREP_all across datasets for every number of clusters. The only excemption is the TRACE dataset, where the two methods seem to perform similarly (Fig. 5.5).

More specifically, when the number of clusters is equal to 20, CLUREP requires searching a fraction of the original datasets that ranges from 7.2% to 45.3% whereas the corresponding fraction of CLUREP_all ranges from 8.0% to 69.9%. On the average across datasets, CLUREP requires searching the 19.6% of the original data volume, whereas the corresponding percentage for CLUREP_all is 39.5%.

Table 5.4 Percentage of the volume of data that is searched (CLUREP)

		Number of Clusters									
ID	Dataset	2	4	6	8	10	12	14	16	18	20
1	50words	87.1	67.3	53.9	45.7	40.1	35.9	32.9	30.4	28.5	26.7
2	CBF	57.9	34.5	25.4	19.4	16.1	13.7	12.0	10.4	9.6	9.0
3	ECG200	68.8	43.1	32.8	29.2	25.5	23.5	21.2	20.1	18.7	17.4
4	FaceAll	67.4	48.2	39.3	34.2	30.9	28.4	26.4	24.9	23.5	22.2
5	GunPoint	61.3	36.0	26.8	20.5	17.3	15.5	14.3	11.6	10.9	10.6
6	OSULeaf	85.7	74.6	67.5	62.9	58.5	55.2	52.1	49.5	47.5	45.3
7	SwedishLeaf	77.6	58.9	49.6	43.5	39.0	35.7	33.3	31.1	29.3	27.8
8	SyntheticControl	59.2	42.3	35.6	31.8	28.5	25.8	24.1	22.7	21.5	20.3
9	Trace	51.2	31.1	21.2	17.2	14.2	12.1	10.3	9.0	8.3	7.2
10	TwoPatterns	86.4	68.2	58.2	51.3	46.1	42.0	38.7	35.7	33.2	31.1
11	Wafer	55.9	33.7	24.3	18.6	15.2	12.9	11.1	9.8	8.8	8.0
12	Yoga	61.1	34.1	25.2	20.2	17.0	14.8	13.1	11.7	10.5	9.8

Table 5.5 Percentage of the volume of data that is searched (CLUREP_all)

		Number of Clusters									
ID	Dataset	2	4	6	8	10	12	14	16	18	20
1	50words	99.0	94.4	86.5	79.1	72.9	67.7	63.3	60.0	56.1	53.8
2	CBF	79.5	54.8	43.9	34.1	27.3	22.8	18.9	16.8	14.6	13.0
3	ECG200	97.0	75.4	60.8	49.8	44.5	39.4	35.2	31.4	28.9	27.4
4	FaceAll	98.7	93.6	88.9	84.2	80.7	77.7	74.2	71.1	68.0	65.3
5	GunPoint	82.7	60.1	43.0	30.3	25.7	22.0	18.2	16.6	15.5	13.6
6	OSULeaf	96.9	94.3	90.6	86.9	83.9	80.7	77.5	74.6	72.4	69.9
7	SwedishLeaf	99.7	97.6	94.5	90.3	85.4	81.6	77.7	74.6	71.7	69.3
8	SyntheticControl	66.5	59.5	53.6	47.7	43.7	40.1	37.6	35.0	33.2	32.0
9	Trace	49.7	34.7	23.7	19.0	15.5	13.4	10.8	9.6	8.9	8.0
10	TwoPatterns	99.8	97.2	93.3	89.6	85.5	81.5	77.3	73.6	69.8	66.3
11	Wafer	58.3	55.6	49.2	43.1	39.1	35.8	33.1	30.4	28.2	26.0
12	Yoga	85.0	59.5	53.9	48.0	43.5	39.6	36.7	34.1	31.6	29.3

5.6 Conclusions

In this chapter, we propose a cluster-based method (CLUREP) for similarity search for the purpose of 1NN classification. Part of this method can be utilized in conjunction with a multidimensional indexing structure. However, we examine the effectiveness of this approach in sequential searching because the dimensionality reduction technique that is applied on data may not be adequate for providing a multidimensional indexing the means for outperforming sequential scanning.

The main conclusion is that, when compared to sequential searching, CLUREP is at least 5 times faster in the majority of the datasets considered in the experiments. In addition to that, CLUREP substantially improves the performance of a similar approach (CLUREP_all) by a factor of 2. A secondary conclusion is that when the number of generated clusters increases, the performance of CLUREP improves. However, experiments show that the rate of improvement "slows down" after the generation of approximately 8 to 12 clusters.

A more general conclusion is that, as the dimensionality of the transformed data increases, the classification error rate decreases with the cost of an increasing percentage of the volume of data that need to be searched. In the data mining context, one can trade-off between accuracy and efficiency with respect to the requirements of the application under consideration.

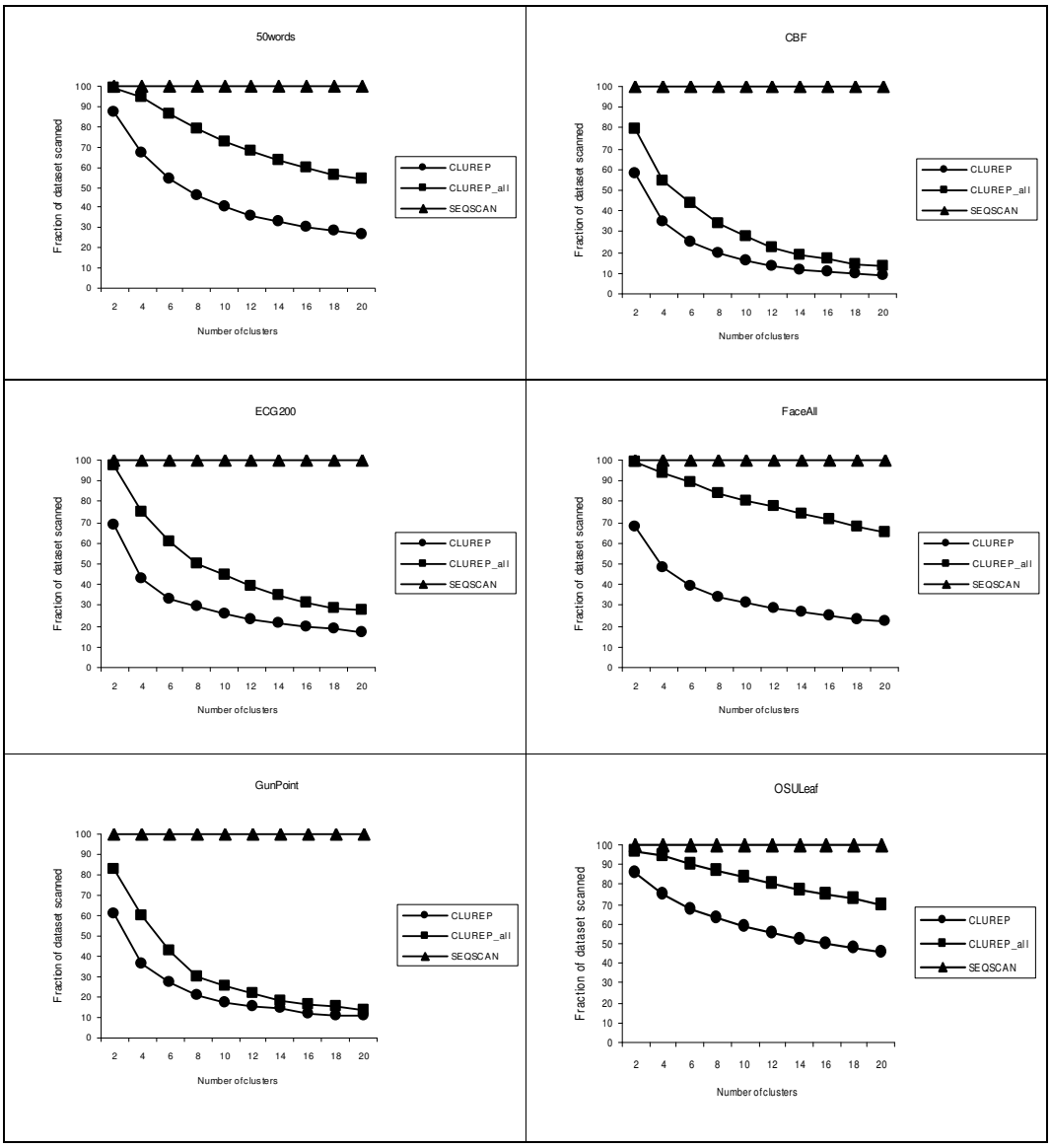


Figure 5.4 Percentage of the volume of data that is searched across varying number of clusters (datasets 1-6) (d = 10)

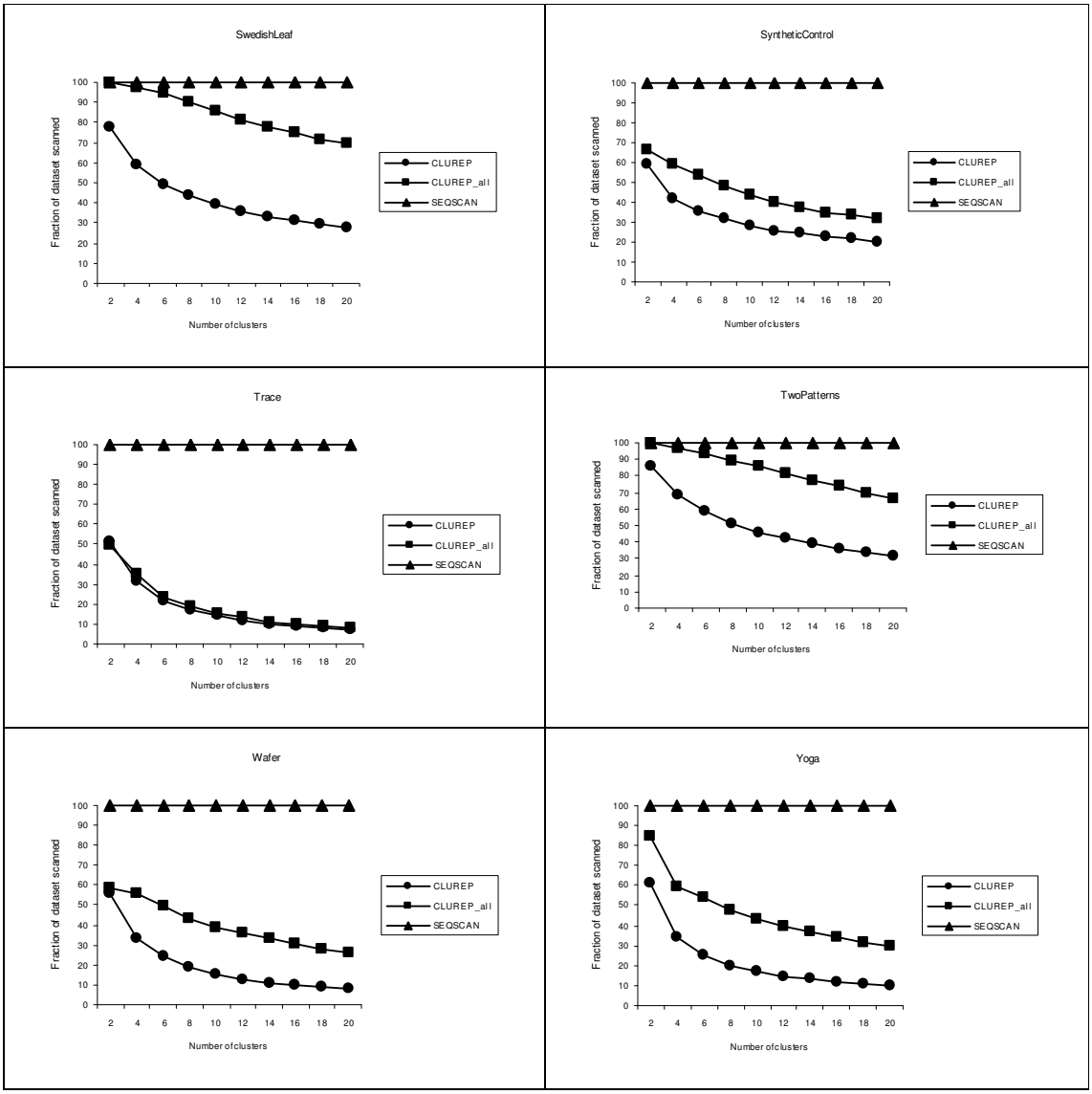


Figure 5.5 Percentage of the volume of data that is searched across varying number of clusters (datasets 7-12) (d = 10)

CHAPTER 6

Issues on Multivariate Time Series Data Mining – PCA

6.1 Introduction

Multivariate time series appear frequently in several diverse applications. Examples include human motion capture [113], geographical information systems [28], statistical process monitoring [75], or intelligent surveillance systems [146]. Time Series Data Mining (TSDM) techniques have been investigated for the purpose of analyzing efficiently this type of data. As it was stated in previous chapters, the notion of similarity is of great importance in almost every TSDM task. For instance, it is of interest to form clusters of objects that move similarly by analyzing data from surveillance systems or classify current operating conditions in a manufacturing process into one of several operational states. In the field of computer graphics, an animator needs to search efficiently a motion database for similar motions to a desired one [48].

A multivariate time series is generated by recording the values of p attributes that co-evolve through time ($p > 1$). These attributes are also called variables. For example, suppose that we analyze stock market data. If we record the closing prices of a particular stock within a month, we generate a univariate time series ($p = 1$). On the other hand, if we additionally record the volume exchanged and the P/E index within the same month, we generate a multivariate time series ($p = 3$). Another application that generates multivariate time series is the monitoring of a human motion, where there are several sensors that record the position of various body joints. Several researchers refer to this type of series as multi-dimensional or multi-attribute. These terms may be used interchangeably through this chapter. Note that the notion of dimensionality is different than the one encountered in univariate time series. In the latter case, the dimensionality is defined as the length of the series, whereas in the multivariate case, most often, it refers to the number of individual time series (variables). Within Data Mining context, it is important to reduce the dimensionality of a series not only by reducing its length but also the number of the individual series.

Although there is a vast literature in univariate time series similarity search, mainly focused on the interrelated issues of representation, distance/similarity measure and indexing (Chapter 2), the case of multivariate time series has not been extensively explored with

respect to these issues. Most of the papers concentrate on indexing multidimensional time series and provide appropriate extensions of already known representations and/or similarity measures. In addition to that, most of the research interest lays on trajectories, which usually consist of 2 or 3 dimensional time series. A different direction in research is based on the dimensionality reduction of a series with respect to the number of individual time series (variables). The driving tools in this approach are the interrelated techniques of Principal Component Analysis (PCA) and Singular Value Decomposition (SVD). The main idea is to condense a large number of interrelated variables into a smaller set of variates, while retaining as much as possible of the variation present in the original data. These two techniques act as representations with special features that allow the definition of different kind of similarity measures than the ones proposed in the univariate case.

The main contributions of this chapter are the following:

- We provide an extensive literature review with respect to multivariate time series data mining.
- We thoroughly explore the implications of applying PCA on multivariate time series in the context of similarity search.

In section 6.2, a formal definition of Multivariate Time Series is provided along with the special issues that arise with respect to representation and similarity measure. In Section 6.3, we present an extensive literature review of multivariate time series data mining. PCA is presented in Section 6.4, and the corresponding assumptions with respect to time series are discussed in Section 6.5. The implications of applying PCA on time series similarity search are investigated in Section 6.6. Finally, conclusions are included in Section 6.7.

6.2 Multivariate Time Series: Representations & Similarity Measures

A Multivariate Time Series (MTS) can be represented as a matrix $X_{n\times p}$, where p is the number of individual time series and n is their length (Eq. 6.1).

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix} \tag{6.1}$$

Each column corresponds to the values of a time series, which can be considered as the values of a variable. Each row corresponds to the values of all variables at a particular

time instance. We assume that each time series has the same length n, since they are generated by the same procedure within the same time interval.

As it was stated in previous chapters, the notion of similarity is of great importance in almost every TSDM task. Regarding similarity search, the first issue is the representation of a multivariate time series for the purpose of reducing the dimensionality. Contrary to the univariate case, the notion of dimensionality does not refer only to the length of the series but also to the number of variables. One could approach this issue by considering X as a univariate series that consists of the concatenation of the corresponding individual series (Eq. 6.2). Then, any representation technique could be applied, such as DFT or PAA.

$$X' = \left[x_{11}, x_{21}, ..., x_{n1}, x_{12}, x_{22}, ..., x_{n2}, ..., x_{1p}, x_{2p}, ..., x_{np} \right] \quad (6.2)$$

A different approach is to reduce the length n of each one of the individual time series (Eq. 6.3), and then concatenate them into a univariate time series (Eq. 6.4).

$$X' = \begin{bmatrix} x'_{11} & x'_{12} & \cdots & x'_{1p} \\ x'_{21} & x'_{22} & \cdots & x'_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ x'_{N1} & x'_{N2} & \cdots & x'_{Np} \end{bmatrix} \quad (6.3)$$

$$X'' = \left[x'_{11}, x'_{21}, ..., x'_{N1}, x'_{12}, x'_{22}, ..., x'_{N2}, ..., x'_{1p}, x'_{2p}, ..., x'_{Np} \right] \quad (6.4)$$

where $N \ll n$.

The second issue, with respect to similarity search, is the definition of a distance measure between two MTS. The most commonly applied measure is the Euclidean distance, which is defined between two MTS (X, Y) as in Eq. 6.5.

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix} \quad \text{and} \quad Y = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1p} \\ y_{21} & y_{22} & \cdots & y_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ y_{n1} & y_{n2} & \cdots & y_{np} \end{bmatrix}$$

$$D(X,Y) = \left(\sum_{i=1}^{n} \left\| x_i - y_i \right\|^2 \right)^{\frac{1}{2}} \text{ where, } \left\| x_i - y_i \right\| = \left(\sum_{i=1}^{p} (x_{i1} - y_{i1})^2 \right)^{\frac{1}{2}} \quad (6.5)$$

Similarly, several researchers extend other distance measures for multivariate time series, such as the Dynamic Time Warping [150].

6.3 Literature Review of Multivariate Time Series Data Mining

There is a vast literature in univariate time series Data Mining with respect to the interrelated issues of representation, similarity measure and indexing (Chapter 2). However, the case of multivariate time series has not been extensively explored with respect to these issues. In addition to that, most of the research interest lays on trajectories, which usually consist of 2 or 3 dimensional time series. The literature review has indicated that there are three main classes of papers. The first one concentrates on indexing multidimensional time series and provides an appropriate representation scheme and/or a similarity measure, whereas the second one is driven by specific data mining applications, such as clustering or novelty detection. A third class of papers focuses on mining streaming multiple time series. This case is different than the multivariate case because there is only one MTS under consideration that evolves through time, whereas in the multivariate case there is a large number of MTS that need to be compared to each other. Nevertheless, we have included a brief review of papers from this class, since particular tasks may require treating time series in a multivariate way.

Regarding the first class of papers, the authors of [147] introduce the problem of similarity search on multi-dimensional time series, such as trajectories, under different orientations. They propose a novel method of mapping trajectories into a space that is invariant of translation, scaling and rotation. In addition to that, they define a DTW-based (Dynamic Time Warping) similarity measure on this new space. More specifically, their approach involves transforming the spatial coordinates of a trajectory into a sequence of angle / arc-length pairs and normalizing the transformed series by applying a novel iterative modulo normalization technique, which deals with the possible problem of vertical shifts. The proposed similarity measure allows elastic matches and can be efficiently lower bounded.

Vlachos et. al [148] propose an indexing framework that supports multiple distance functions, without the need to rebuild the index. Although the primary measure is based on the Longest Common Subsequence (LCSS) model, the proposed framework is able to

support other measures, such as the Euclidean distance and the Dynamic Time Warping. Their work is mainly focused on trajectories.

In [22], an extension of LCSS is provided for the purpose of clustering motion trajectories. The authors define a LCSS-based distance between two 2-dimensional trajectories as a pair of numbers. Each number represents the similarity between the projections of the trajectories on the coordinate axes. The clustering is performed by applying an agglomerative algorithm.

The authors of [30] introduce a novel distance function, namely the Edit Distance for Real sequence (EDR), for the purpose of similarity search of moving object trajectories. EDR is robust to noise, shifts and scaling of this type of data and is accompanied by three pruning techniques in order to improve the retrieval efficiency.

Kahveci et al. [73] extend the Euclidean distance to handle shift and scale invariance and propose a novel index structure called CS-index. The authors investigate separately the cases of dealing with dependent and independent variables. In the first case, they propose to convert a p-dimensional time series of length n to a univariate time series of length np by concatenating the individual series, and then apply a representation for the purpose of dimensionality reduction. In the latter case, they propose to reduce the dimensionality of each individual time series separately, and then to concatenate the resulting series.

On the other hand, Lee et al. [97] propose a scheme for searching a database, which, in the pre-processing phase includes the representation (e.g. DFT) of each one of the individual time series separately. They provide distance measures between two series along with appropriate lower bounds and extend the similarity search method on univariate time series in order to support multidimensional time series.

Cai & Ng [23] approximate and index multidimensional time series with Chebyshev polynomials and prove the Lower Bounding Lemma for this representation. That is, that the true distance between two time series is lower-bounded by the distance in the space of Chebyshev coefficients. In the previous three papers, the Euclidean distance is applied as a distance measure.

Bakalov et al. [10] extend the *Symbolic Aggregate Approximation* (SAX) [104] and the corresponding distance measure for moving objects. They provide an indexing scheme for efficiently and accurately discovering similarities among trajectories. Their work is focused on the specific problem of identifying all pairs of similar trajectories between two datasets.

Regarding the second class of papers that are driven by Data Mining applications, Fujimaki et al. [50] present a novel anomaly detection method for spacecrafts, based on

Kernel Feature Space and directional distribution. Part of their work is to define an anomaly metric. Although this is an application-oriented method, the fact that it requires little a priori knowledge makes it potentially useful to other applications too.

In [69] a dimensionality reduction technique is proposed for the purpose of revealing spatio-temporal structures from human motion streams and therefore deriving primitive behaviors (vocabulary). They propose an extension of *ISOMAP*, a non-linear dimensionality reduction technique, which incorporates not only the extraction of spatial structures but also the corresponding temporal dependencies. This is a PCA-based method (Principal Component Analysis) that performs the decomposition on a feature space similarity matrix instead of an input space covariance matrix. The extended technique takes into account the temporal dependencies by adjusting the similarity matrix through Weighted Temporal Neighbours. The combination of this type of dimensionality reduction with clustering is proposed in order to determine a human motion vocabulary.

The work of Tanaka et. al. [141] introduces a new method for identifying motifs from multi-dimensional time series. It applies Principal Component Analysis to reduce dimensionality and perform a symbolic representation. Then, the motif discovery procedure starts by calculating a description length of a pattern based on the "Minimum Description Length" principle.

A different approach, which is mainly motivated from research on human genome sequences, is introduced in [54]. However, this approach is more general and involves multivariate time series. The notion behind their approach is that, the high variability that some time series very often exhibit may be explained by the existence of several different sources that affect different segments of this series. More specifically, the task is to find a proper way to segment a time series into k segments, each of which comes from one of h different sources ($k >> h$). This task is analogous to clustering the points of a time series in h clusters with the additional constraint that a cluster may change at most $k-1$ times.

Finally, Geurts [53] presents formally the problem of multivariate time series classification and presents a novel tool that is based on a piecewise constant modeling of temporal signals by regression tress. The main objective is to generate interpretable results while retaining classification accuracy at a high level.

The third class of papers aims at identifying patterns among streaming multiple time series. Although each one of the time series is univariate, the fact that they are being generated concurrently results in recording multiple measurements at each time point. Depending on the task, this fact may require treating time series in a multivariate way. *SPIRIT* is introduced in [122] as a new approach to discovering patterns from streaming

multiple time series. This approach identifies correlations and hidden variables among time series by applying Principal Component Analysis, in order to summarize the entire set of streams and provide useful means in efficient forecasting. The *SPIRIT* also satisfies the important requirements of an efficient streaming pattern discovery procedure, that is, it is streaming, it scales linearly with the number of time series, it is adaptive to changes and it is automatic.

In [33] the task of discovering correlated windows of time series (synchronously or with lags) over streaming data is addressed. The authors concentrate on the case where the time series are "uncooperative", meaning that there does not exist a fundamental degree of regularity that would allow an efficient implementation of DFT or DWT transformations. The proposed method involves a combination of several techniques – sketches (random projections), convolution, structured random vectors, grid structures, and combinatorial design – in order to achieve high performance.

In [133] a novel method for extracting time-correlations among multiple time-series is introduced. The time-correlations are expressed in the form of correlation rules such as: if A increases more than 5% then B decreases more than 10% on the next day. In order to reduce the search space, the proposed method includes summarization of the original data by aggregation at different time granularities and detection of the change points upon which the desirable comparisons will be based. For the latter task, CUSUM, a well known statistical method, is used in order to convert continuous time series data to discrete data. The resulting representation includes the change points along with a label (UP or DOWN) that indicates the direction of the change and the amount of change.

6.4 Review of Principal Component Analysis

Principal Component Analysis (PCA) is a well-known statistical method for multivariate data analysis. It is applied on a set of interrelated quantitative variables in order to derive a new set of variables, which are called components. Each one of these components represents a group of the initial variables that share common features. In addition to that, these components are determined to be uncorrelated to each other and ordered in a way that the first few of them will account for the largest variance present in the original dataset.

The PCA technique serves two objectives. First, it identifies the underlying patterns or relationships for a large number of variables and determines whether this information can be condensed or summarized in a smaller set of components. Second, it can be utilized in

order to reduce the dimensionality of a dataset for subsequent analysis. The latter objective is of great importance when there is a huge amount of data that requires efficient analysis.

6.4.1 Preliminaries

Prior of presenting the PCA method, we provide definitions of the main concepts of eigenvalues, eigenvectors variance, covariance, correlation coefficient.

Definition 6.1. The eigenvalue of a square matrix $A_{p\times p}$ is a scalar value λ, which satisfies the following property:

$$A \cdot u = \lambda \cdot u$$

where u is a unit vector. The vector u that corresponds to an eigenvalue is called eigenvector.

Definition 6.2. The variance of a variable X is a dispersion measure that is based on the deviations of each value x_i of the variable from their mean, and it is defined by the following equations:

$$\sigma^2 = \frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n} \text{ , when } \mu \text{ is known}$$

$$s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1} \text{ , when } \mu \text{ is unknown}$$

where μ and $\bar{x}$ are the population and sample means respectively.

Definition 6.3. The standard deviation of a variable is defined as the square root of the variance and is denoted by the symbols σ (population standard deviation) or s (sample standard deviation).

Definition 6.4. The covariance of two variables X_i and X_j indicates the degree of linear association between two variables and it is defined by the following equations

$$cov(X_i, X_j) = \frac{\sum_{k=1}^{n}(x_{ki} - \mu_i)(x_{kj} - \mu_j)}{n}, \text{ when } \mu_i \text{ and } \mu_j \text{ are known}$$

$$cov(X_i, X_j) = \frac{\sum_{k=1}^{n}(x_{ki} - \bar{x}_i)(x_{kj} - \bar{x}_j)}{n-1}, \text{ when } \mu_i \text{ and } \mu_j \text{ are unknown}$$

Definition 6.5. The Pearson product-moment coefficient of correlation is a measure that indicates the strength of the linear relationship between two variables that is independent of their respective scales of measurement. In the case that the corresponding population means are unknown, the coefficient of correlation is defined by the following equation:

$$r = \frac{cov(X_i, X_j)}{s_{X_i} \cdot s_{X_j}}$$

where s_{X_i} and s_{X_j} are the standard deviations of X_i and X_j respectively.

6.4.2 Mathematical Formulation of PCA

Table 6.1 provides a list of symbols and definitions that will be used throughout this chapter.

Table 6.1 Symbols and Definitions

X	the $n \times p$ data matrix
X^T	the transpose of X
X_i	the i^{th} variable
x_{ij}	the i^{th} value of the j^{th} variable
$Y_{(i)}$	the i^{th} principal component
A	the $p \times p$ matrix of component weights
$a_{\cdot j}$	the j^{th} column vector of A
a_{ij}	the weight of X_i in $Y_{(j)}$
Σ	the variance – covariance matrix
S	the sample variance – covariance matrix
$\Lambda_{p\times 1}$	the matrix of the eigenvalues (variances) in decreasing order

Y the $n \times p$ matrix the matrix of the coordinates of data on the new set of axes

Principal Component Analysis is applied on a multivariate dataset, which can be represented as a matrix $X_{n\times p}$ (Eq. 6.6), where p is the number of variables X_i and n is number of observations that have been recorded for each variable.

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix} \tag{6.6}$$

The objective of PCA is to determine a new set of orthogonal and uncorrelated composite variates $Y_{(i)}$ (Eq. 6.7), which are called principal components:

$$Y_{(i)} = a_{1i}X_1 + a_{2i}X_2 + ... + a_{pi}X_p \tag{6.7}$$

where i = 1, 2, …, p and X_i denotes the i^{th} variable. The coefficients a_{ij} are also called component weights.

The first step is the determination of the first component $Y_{(1)}$, which is a linear combination of the original variables (Eq. 6.7) and will account for the maximum portion of their variance. In other words, we must determine those values for a_{i1} for which the variance of $Y_{(1)}$ (Eq. 6.8) will be maximized. This can be achieved by utilizing the variance – covariance matrix Σ, where the element (i , j) corresponds to the covariance between the variables X_i and X_j when $i \neq j$, and to the variance of the variable X_i when $i = j$. In the case that the matrix Σ is unknown, the variances and covariances are estimated from the sample and the resulting matrix is denoted by S.

$$Y_{(1)} = a_{11}X_1 + a_{21}X_2 + ... + a_{p1}X_p \tag{6.7}$$

$$var(Y_{(1)}) = \alpha_{.1}^{T} \cdot \Sigma \cdot \alpha_{.1} \tag{6.8}$$

Apparently, the determination of the coefficients a_{i1} is not feasible, since the variance always increases for greater values of a_{i1}. This fact necessitates the restriction of the allowable values that these coefficients can take. Most often we impose the following restriction:

$$\alpha_{\cdot 1}^{T} \cdot \alpha_{\cdot 1} = 1 .$$

It can be proved that the vector of the required coefficients is the eigenvector of Σ that corresponds to the largest eigenvalue λ_1. This eigenvalue represents the (maximum) variance of the first component (Eq. 6.9).

$$var(Y_{(1)}) = \alpha_{\cdot 1}^{T} \cdot \Sigma \cdot \alpha_{\cdot 1} = \lambda_1 \qquad (6.9)$$

The next step is the determination of the second component $Y_{(2)}$, which will account for the maximum portion of the remaining variance subject to being orthogonal to the first one.

Generally, it can be proved that the k^{th} component has variance equal to λ_k, where λ_k is the k^{th} largest eigenvalue of Σ and $a_{\cdot k}$ is the corresponding eigenvector.

The number of the components is equal to the number of the original variables (p). The original data are transformed by utilizing the derived components $Y_{(k)}$. When the total number of components is retained, the transformed data can be represented as a matrix Y (Eq. 6.10), which has the same dimensions with X and holds the initial information. The objective of PCA is to retain the first m components, which retain most of the variation present in all of the original variables (p). Thus, an essential dimensionality reduction may be achieved by projecting the original data on the new m-dimensional space, as long as, $m \ll p$.

$$Y = \begin{bmatrix} \sum_{i=1}^{p} a_{i1}x_{1i} & \sum_{i=1}^{p} a_{i2}x_{1i} & \cdots & \sum_{i=1}^{p} a_{ip}x_{1i} \\ \sum_{i=1}^{p} a_{i1}x_{2i} & \sum_{i=1}^{p} a_{i2}x_{2i} & \cdots & \sum_{i=1}^{p} a_{ip}x_{2i} \\ \vdots & \vdots & \vdots & \vdots \\ \sum_{i=1}^{p} a_{i1}x_{ni} & \sum_{i=1}^{p} a_{i2}x_{ni} & \cdots & \sum_{i=1}^{p} a_{ip}x_{ni} \end{bmatrix} \quad (6.10)$$

As it is obvious from the above analysis, PCA generates two matrices, namely the matrix of the component weights $A_{p\times p}$ and the matrix of the corresponding variances $\Lambda_{p\times 1}$ in decreasing order. The matrix Y that holds the transformed values can be easily obtained as follows:

$$Y = X \cdot A$$

As it was mentioned earlier, Principal Component Analysis is based on the variance-covariance matrix Σ. Alternatively, the determination of the components can be based on the correlation matrix S, where the element $(i\ ,\ j)$ is the correlation coefficient of the variables X_i and X_j when $i \neq j$, and the standard deviation of the variable X_i when $i = j$. The final result is different than the one obtained when Σ is utilized. These two approaches are equivalent when PCA is applied on the normalized values (z_{ij}) of the original data (Eq. 6.11).

$$z_{ij} = \frac{x_{ij} - \bar{x}_{\cdot j}}{s_{\cdot j}} \quad (6.11)$$

where $\bar{x}_{\cdot j}$ και $s_{\cdot j}$ are the mean value and the standard deviation of the variable X_j. These transformed values have a mean value equal to zero and standard deviation equal to one. This approach is very popular because the normalized values are independent of the scale of measurement, which is usually different among variables.

6.4.3 Intuition behind PCA

Suppose that we have two variables X_1 and X_2 ($p = 2$) and their corresponding values at ten instances ($n = 10$). Figure 6.1 presents this dataset as a set of points on a 2-

dimensional space. We observe that there is considerable variation in both variables, though rather more in the direction of X_2 than X_1. Intuitively, PCA transforms the dataset X by rotating the original axes and deriving a new set of axes $Y_{(1)}$ and $Y_{(2)}$ (components). It is clear that under this rotation there is greater variation in the direction of $Y_{(1)}$ than in either of the original variables, but very little variation in the direction of $Y_{(2)}$. If only $Y_{(1)}$ is retained and data is projected on it, there will be little loss of information, whereas a dimensionality reduction will be achieved by retaining only one of the two dimensions.

As it was shown in the previous section, PCA generates the matrices $A_{p\times p}$ and $\Lambda_{p\times 1}$. The component weights a_{ij} provide the directions of the components [55], whereas the variances λ_i provide their corresponding strengths. In addition to that, the matrix $X \cdot A$ provides the coordinates of the original data on the new set of axes.

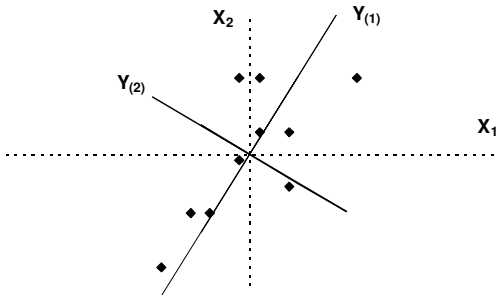


Figure 6.1 A multivariate time series consisting of two variables (X_1 and X_2) and ten instances. Dots represent the instances, while solid lines represent the principal components that have been derived by PCA. A dimensionality reduction can be achieved, if only the first component $Y_{(1)}$ is retained and data is projected on it.

6.4.4 PCA & SVD

Singular Value Decomposition (SVD) is of great importance to PCA, since it provides a computationally efficient method of determining the principal components. This technique is presented analytically in Section 2.5.2. The SVD theorem states that an $n \times p$ real matrix X can be expressed in the following form:

$$X_{n\times p} = U_{n\times n} \cdot S_{n\times p} \cdot V^T_{p\times p} \qquad (6.12)$$

where S is a diagonal matrix that contains the singular values; the matrices U and V are orthonormal and their columns are the left and right singular vectors respectively.

There is a direct relation between PCA and SVD in the case where principal components are calculated from the variance-covariance matrix. Suppose that the matrix X consists of the deviations of the original values from their corresponding mean. In other words, suppose that for each column of X we subtract its mean from every element. Under this condition, the matrix $(1/(n-1))\cdot X^T \cdot X$ is the variance –covariance matrix of X. When X is decomposed according to Eq. 6.12, the matrix V provides the eigenvectors of $X^T \cdot X$, meaning that V is equal to A. The matrix S provides the eigenvalues of $X^T \cdot X$, which are the square roots of the eigenvalues that represent (proportional) the variances of the principal components. Finally, $U \cdot S$ provides the coordinates of the observations in the new space.

Conclusivelly, SVD provides all the necessary information for the determination of the principal components. In addition to that, it provides the matrix U, which plays the same role with V when we reverse the roles of variables and observations.

6.5 Principal Component Analysis for Time Series

In Section 6.4, we reviewed Principal Component Analysis without referring explicitly to the type of data under consideration. In case of time series, PCA is applied on a multivariate dataset $X_{n\times p}$, where n represents their length (number of time instances) and p is the number of variables being measured (number of individual time series). As it was mentioned in Section 6.4, the first objective of PCA is the identification of the underlying patterns or relationships for a large number of variables. However, there are two assumptions that should be met in order to provide statistical inferences from this analysis, specifically, the independence and the multivariate normality of the values of variables. When the variables are time series the first assumption does not hold, since each value of a series cannot be considered as independent to the value that was observed in the previous time instance. On the other hand, these assumptions are not required when the objective is to reduce the dimensionality of a dataset. The PCA method, as it was described in the previous section, does not take into consideration the dependencies that might exist among the values of a series.

In Time Series Analysis, the notion of stationarity plays a central role. Intuitively, a time series is said to be stationary if there is no systematic change in mean, if there is no

systematic change in variance and if periodic variations are not present. In other words, the properties of one section of the series are much like those of any other section [29]. More formally, a time series X_t is called stationary when the following equations hold.

$$\mu = \mathrm{E}(X_t), \quad \forall t \in T \tag{6.13}$$

$$\gamma_k = \mathrm{E}\left[\left(X_t - \mu\right)\left(X_{t+k} - \mu\right)\right] = cov\left(X_t, X_{t+k}\right), \quad \forall t \in T \tag{6.14}$$

where T is a time interval.

These two equations (Eq. 6.13) (Eq. 6.14) imply that the mean value and the covariance are independent of the time instance. On the other hand, the covariance between two terms of the series depends on their lag (k).

The above definition can be generalized for a multivariate time series, which consists of p variables (Eq. 6.15) (Eq. 6.16).

$$\mu = \mathrm{E}(X_t), \quad \forall t \in T \tag{6.15}$$

$$\Gamma_k = \mathrm{E}\left[\left(X_t - \mu\right)\left(X_{t+k} - \mu\right)^T\right], \quad \forall t \in T \tag{6.16}$$

where μ is the vector of the mean values of the p variables, and Γ_k is a matrix $p \times p$ that holds the covariances of the variables with lag equal to *k*.

According to the previous section, PCA is applied on the variance – covariance matrix, which in this section is denoted by Γ_0. However, in the case of time series PCA can be applied to variance-covariance matrices Γ_k between variables that their values are recorded with lag $k \neq 0$.

Literature provides several modified versions of PCA that take into consideration the dependencies among the values of one series and/or more series, even in the presence of lag [71]. The main objective of these versions is the identification of patterns for the purpose of interpreting specific phenomena (e.g. in meteorology). However, the fact that these methods take into consideration the lag of the values results into a substantial increase in the number of variables. This means that the application of PCA becomes computationally expensive, and not appropriate within Data Mining context.

6.6 Implications of PCA in Similarity Search

Similarity search is based on shapes, meaning that two time series are considered similar when their shapes are considered to be "close enough". Apparently, the notion of "close enough" depends heavily on the application itself, a fact that affects the decision of the pre-processing phase steps to be followed, the similarity measure to be utilized and the representation to be applied on the raw data.

Two MTS can be considered similar when the corresponding individual time series are similar. The application of PCA on two similar MTS will provide similar results. More specifically, the corresponding principal components will be close enough, that is, the corresponding angles will be close to zero degrees, and the corresponding variances will be almost equal. When these two requirements are fulfilled, the transformed values will be also close to each other. In addition to that observation, two MTS with dissimilar analysis in principal components, they are expected to be dissimilar. However, this does not hold vice versa. When two MTS have similar analysis in principal components, they may be similar or dissimilar. The reason is that PCA takes into consideration the covariances among variables. Nevertheless, two covariances may be equal while corresponding to different patterns (shapes). Figure 6.2 presents two dissimilar 2-dimensional MTS, which have exactly the same analysis in principal components. This observation implies that PCA-based similarity search requires a further filtering step in order to discard this kind of false hits.

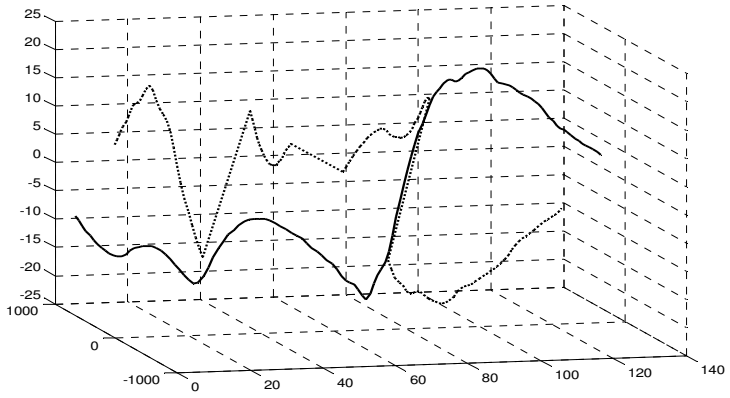


Figure 6.2 Two dissimilar 2-dimensional time series that have the same analysis in principal components

A second implication of applying PCA on MTS is that the temporal nature of data is not taken into account while deriving the principal components, since this procedure is based on the covariances among variables. Note that the order of the time instances of two variables does not affect the value of their covariance. In order to further clarify this

comment, let take the extreme case of two 2-dimensional MTS that consist of exactly the same values in reverse order (Figure 6.3). Their corresponding variance-covariance matrices are exactly the same and thus the analysis in principal components is the same. The only difference that may appear is the direction of the principal components. The corresponding principal components lay on the same axes but their direction may be opposite.

When PCA is utilized for the purpose of similarity search, the definition of similarity should be fully clarified. For example, it should be clarified whether the MTS in Figure 6.3 can be considered similar or not in order to appropriately evaluate the application of PCA.

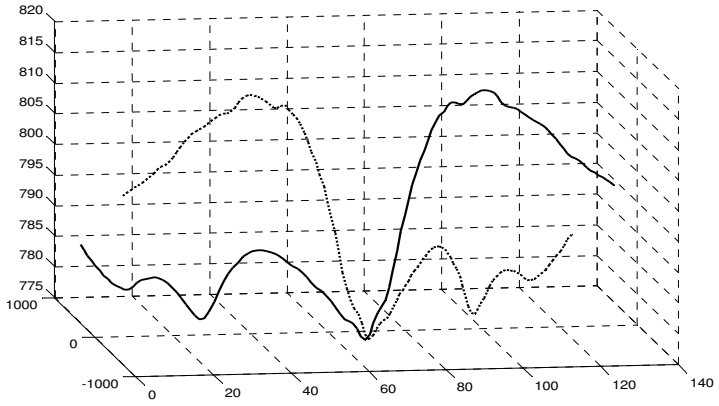


Figure 6.3 Two 2-dimensional time series evolving in reverse order

Regarding the pre-processing phase (Section 2.3), there are four main distortions that may be existed in raw data, namely, offset translation, amplitude scaling, time warping and noise. Distance measures may be affected seriously by the presence of any of these distortions, resulting most of the times in missing similar shapes.

Offset translation refers to the case where there are differences in the magnitude of the values of two time series, while the general shape remains similar. This distortion is inherently handled by PCA, since it is based on covariances, which are not affected by the magnitude of the values. It can be proved that the covariance of two variables is not affected by the addition of a constant on each value (Eq. 6.17). Therefore, if we transform each value by subtracting the corresponding mean, the covariance will remain the same (Eq. 6.18). This is a potential disadvantage of PCA, if similarity search is to be based also on the magnitude of the values.

$$cov(X_i, X_j) = cov(X_i + a, X_j + b), \quad a,b \in \mathbb{R} \tag{6.17}$$

$$cov(X_i, X_j) = cov(X_i - \bar{x}_i, X_j - \bar{x}_j), \quad a,b \in \mathbb{R} \tag{6.18}$$

Amplitude scaling refers to the case where there are differences in the scaling of the values of two time series, while the general shape remains similar. In this case, PCA representation can be based on the correlations among variables, instead of the covariances. This is an alternative way of deriving the principal components, which produce slightly different results, but not essentially different in the context of dimensionality reduction.

Time warping, which may be global or local, refers to the acceleration or deceleration of the evolvement of a time series through time. In the case of global time warping (i.e. two multivariate time series evolve in different rates), PCA representation is expected to be similar, since the shorter time series can be considered as a systematic random sample of the longer one, resulting to similar covariance matrix. On the other hand, the existence of local time warping distortions may be captured by the covariances of the corresponding variables.

Finally, noise is intrinsically handled by PCA, since the discarded principal components account mainly for variations due to noise.

Another issue in the pre-processing phase is analyzing time series of different lengths. PCA requires variables (time series) of equal length for the same object; therefore, this is a limitation of this technique. However, similarity search is performed on objects, and thus, it is based on the produced matrices $A_{p\times p}$ and $\Lambda_{p\times 1}$, which are independent of the corresponding lengths.

Similarity search also requires a measure that quantifies the similarity or dissimilarity between two objects. Under PCA transformation, this measure should be based on at least one of the produced matrices, mentioned in the previous paragraph, $A_{p\times p}$, $\Lambda_{p\times 1}$, and $Y_{n\times p}$. The central concept is that, if two multivariate time series are similar, their PCA representation will be similar, that is, the produced matrices will be close enough. Searching similarity based on $A_{p\times p}$, means to compare the angles of principal components derived from two multivariate time series, whereas searching based solely on $Y_{n\times p}$ is useless, since these values are coordinates in different spaces. The matrix $\Lambda_{p\times 1}$ contains information about

the shape of the time series and it may be used in conjunction with $A_{p\times p}$ for further distinguishing power.

The PCA representation of a dataset $X_{n\times p}$ consists of the component weight matrix $A_{p\times p}$ and the variances matrix $\Lambda_{p\times 1}$. The data reduction may be substantial, as long as, the number of time instances n is much greater than the number of variables p. More specifically, the original matrix X comprises $n \cdot p$ data elements, whereas the PCA representation comprises $p \cdot p$ data elements for the component weight matrix and p data elements for the corresponding variances. Thus, the ratio of the required space for the PCA representation to the original space is equal to

$$\frac{p \cdot p + p}{n \cdot p} = \frac{p+1}{n}$$

Moreover, a further data reduction can be achieved, if only m components are retained, where $m \le p$. In this case, the corresponding ratio of the required space is equal to

$$\frac{p \cdot m + m}{n \cdot p} \le \frac{p \cdot m + p}{n \cdot p} = \frac{m+1}{n}$$

There are several criteria for determining the number of components to retain, such as the scree graph or the cumulative percentage of total variation [71]. According to the latter criterion, one could select that value for m, for which the first m components retain more than 90% of the total variation present in the original data.

Although PCA-based similarity search is complicated and usually requires expensive computations, it may improve the quality of similarity search providing at the same time useful information for post hoc analysis.

6.7 Conclusion

In this chapter, we discussed the various issues that arise in Data Mining when data is in the form of multivariate time series. We provided a literature review of Multivariate Time Series Data Mining with respect to similarity search and relative applications. Principal Component Analysis was described and the corresponding implications in similarity search were presented and discussed in detail.

There are four key observations. First, there is a trend to adjust existed techniques to data mining tasks performed on multivariate time series. Nevertheless, most of the research is focused on trajectories, which consist of 2 or 3 dimensional time series. Second, there is an increasing demand for analyzing streaming data, which also results into modifying existing methods and providing new algorithms. Third, there has been a lot of work in exploiting Time Series Data Mining techniques from diverse application areas, besides computing science. This fact suggests a closer, interdisciplinary cooperation among research communities. A final observation is that Principal Component Analysis has not been extensively explored in the context of similarity search in multivariate time series and hence, it has the potential to offer more in the Data Mining field.

CHAPTER 7

PCA-based Measures for Similarity Search

7.1 Introduction

The problem of similarity search in multivariate time series has attracted the interest of many researchers from diverse scientific fields. As it was noted in the previous chapter, one direction in research is based on the dimensionality reduction of a series with respect to the number of individual time series (variables). The driving tools in this approach are the interrelated techniques of Principal Component Analysis (PCA) and Singular Value Decomposition (SVD). The central concept is that, if two Multivariate Time Series (MTS) are similar, their PCA representation will be similar. There are several PCA-based measures that have been proposed in order to quantify similarity between two MTS. Most of them take into consideration the angles between the principal components of two MTS and/or their corresponding variances.

In this chapter, a novel PCA-based approach is introduced for the purpose of similarity search. This approach utilizes as a distance measure an extension of the Squared Prediction Error (SPE), a well-known statistic in the Statistical Process Control community. Contrary to other PCA-based measures proposed in the literature, SPE does not require applying the computationally expensive PCA technique on the query. In addition to that, we provide a method that further speeds up the calculation of SPE without affecting substantially the quality of similarity search. This method involves the application of Piecewise Aggregate Approximation (PAA) on the query object during the pre-processing phase, for the purpose of reducing its dimensionality.

In order to evaluate this approach, we conducted experiments on four datasets, which have been used extensively in the literature. More specifically, we performed one-nearest neighbor classification (1-NN) and evaluate it by means of classification error rate. In addition to that, we performed leave-one-out k-NN similarity search and evaluate it by plotting the recall-precision graph [34]. Finally, we evaluated the trade-off between classification accuracy and speed of calculating the proposed measure by applying 1-NN classification. Several measures have also been tested for comparison reasons.

The main contributions of this chapter are the following:

- the introduction of a novel approach in multivariate time series similarity search that is based on Principal Component Analysis,
- a thorough review of several PCA-based similarity/distance measures that have been proposed by researchers from diverse scientific fields, not necessarily within data mining context.

In Section 7.2, we provide related work. Section 7.3 introduces our novel approach and provides a distance measure that is based on Multivariate Statistical Process Control. In Section 7.4, we describe the experimental settings with respect to the datasets, the methods and the rival measures. The results of these experiments are presented and discussed in Section 7.5. Finally, conclusions are provided in Section 7.6.

7.2 Related work

There are several PCA-based measures that have been proposed in order to compare two objects, which are in the form of multivariate time series. The main idea is to derive the principal components for each one and then to compare the produced matrices.

Suppose that we have two multivariate time series denoted $X_{n\times p}$ and $Q_{n\times p}$. Applying PCA on each one results in the matrices of component weights A_X and A_Q, and variances Λ_X and Λ_Q respectively. All the following measures assume that the number of variables p is the same for all series. This is a rational assumption, since these series are usually generated by the same process within a specific application.

One of the earliest measures has been proposed by Krzanowski [96]. This measure (Eq. 7.1) is applicable to time series, although originally it was not applied on such type of data. The proposed approach is to retain m principal components ($m \ll p$) and compare the angles between all the combinations of the first m components of the two objects.

$$Sim_{PCA}(X,Q) = trace(A_X^{\mathrm{T}} \cdot A_Q \cdot A_Q^{\mathrm{T}} \cdot A_X) = \sum_{i=1}^{m}\sum_{j=1}^{m} cos^2\theta_{ij}, \quad 0 \le Sim_{PCA} \le m \qquad (7.1)$$

where θ_{ij} is the angle between the i^{th} principal component of X and the j^{th} principal component of Q.

Johannesmeyer [70] modified the previous measure by weighting the angles with the corresponding variances as in Eq. 7.2.

$$S^{\lambda}_{PCA}(X,Q)=\sum_{i=1}^{m}\sum_{j=1}^{m}(\lambda_{X_i}\cdot\lambda_{Q_j}\cdot cos^2\theta_{ij})/\sum_{i=1}^{m}\lambda_{X_i}\cdot\lambda_{Q_j},\quad 0\leq\quad S^{\lambda}_{PCA}\leq 1 \tag{7.2}$$

Singhal & Seborg [138] extend the previous measure by incorporating an extra term in Eq. 7.2, which expresses the distance between the original values of the two objects. This distance factor (S_{dist}) can be particularly useful in case the two objects have similar principal components but the values of their variables are essentially different. In order to find the distance between the two objects, it is required to set one of them as the reference dataset. Then, the Mahalanobis distance of a dataset from the reference is computed as in Eq. 7.3. Q is assumed to be the reference dataset.

$$\Phi=\sqrt{(\bar{x}-\bar{q})^T\cdot\Sigma_Q^{*-1}\cdot(\bar{x}-\bar{q})} \tag{7.3}$$

where $\bar{x}$ and $\bar{q}$ are vectors that contain the mean values of the variables that consist the datasets X and Q, whereas, Σ_Q^{*-1} is the pseudo-inverse of the covariance matrix of Q.

Assuming Gaussian distribution, the authors propose as a distance factor the probability that the distance is at least Φ units (Eq. 7.4).

$$S_{dist}(X,Q)=\sqrt{\frac{2}{\pi}}\int_{\Phi}^{\infty}e^{-z^2/2}dz,\quad 0\leq S_{dist}\leq 1 \tag{7.4}$$

Note that, although S_{dist} represents distance between two objects, it is a similarity measure. As the distance Φ increases, the corresponding probability decreases.

Finally, the proposed measure (Eq. 7.5) is the weighted summation of two similarity measures.

$$SF=w_1S^{\lambda}_{PCA}+w_2S_{dist},\quad 0\leq SF\leq 1\ \ and\ \ w_1+w_2=1 \tag{7.5}$$

Yang & Shahabi [159] propose a similarity measure, Eros, which is based on the acute angles between the corresponding components from two objects X and Q (Eq. 7.6). Contrary to the previous measures, all components are retained from each object and their variances form a weight vector w. More specifically, the variances obtained from all the objects in a database are aggregated into one weight vector, which is updated when objects

are inserted or removed from the database. Finally, the authors provide lower and upper bounds for this measure.

$$Eros(X,Q,w)=\sum_{i=1}^{p} w(i)\cdot\left|cos\,\theta_i\right|,\quad 0\le Eros\le 1 \tag{7.6}$$

Otey & Parthasarathy [121] define a distance measure in terms of three dissimilarity functions that take into account the differences among the original values, the angles between the corresponding components and the difference in variances. For the first term, authors propose to use either the Euclidean (Eq. 7.7) or the Mahalanobis distance (Eq. 7.8).

$$D_d(X,Q)=\sqrt{(\mu_X-\mu_Q)\cdot(\mu_X-\mu_Q)^{\mathrm{T}}} \tag{7.7}$$

$$D_d(X,Q)=\sqrt{(\mu_X-\mu_Q)\cdot\Sigma_{XQ}^{-1}\cdot(\mu_X-\mu_Q)^{\mathrm{T}}} \tag{7.8}$$

where μ_X and μ_Q are the vectors that contain the mean values of the variables that consist the datasets X and Q, whereas Σ_{XQ} is the covariance matrix of the combination of datasets X and Q.

The second term is defined as the summation of the acute angles between the corresponding components, given that all components are retained (Eq. 7.9).

$$D_r(X,Q)=trace(cos^{-1}(\left|A_X^T\cdot A_Q\right|)),\quad 0\le \mathrm{D_r}\le p\pi/2 \tag{7.9}$$

The third term accounts for the differences in the distributions of the variance over the derived components. Consider the random variable V_X having the probability mass function $p_X=P(V_X=i)=\lambda_i^X/trace(\Lambda_X)$, where $P(V_X=i)$ represents the proportion of the variance in the direction of the i^{th} principal component.

The difference between the distributions V_X and V_Y is defined as the symmetric relative entropy (Eq. 7.10).

$$D_v(X,Q)=\frac{1}{2}(H(p_X\|p_Q)+H(p_Q\|p_X)) \tag{7.10}$$

The proposed distance measure can be defined in the following two forms:

$$D_{\Pi}(X,Q) = D_d \cdot D_r \cdot D_v \tag{7.11}$$

$$D_{\Sigma}(X,Q) = \beta_0 + \beta_d \cdot D_d + \beta_r \cdot D_r + \beta_v \cdot D_v . \tag{7.12}$$

Otey & Parthasarathy refer to Eq. 7.11 as their basic formulation, and to Eq. 7.12 as a more flexible form that allows the terms to be weighted differently according to the needs of a given application.

Li & Prabhakaran [100] propose a similarity measure for recognizing distinct motion patterns in motion streams in real time. This measure, which is called k Weighted Angular Similarity (kWAS), can be obtained by applying singular value decomposition on the transformed datasets, $X^T \cdot X$ and $Q^T \cdot Q$, and retaining the first m components. kWAS is based on the acute angles between the corresponding components weighted by the corresponding eigenvalues (Eq. 7.13).

$$\Psi(X,Q) = (1/2)\sum_{i=1}^{m}((\sigma_i / \sum_{i=1}^{n}\sigma_i + \lambda_i / \sum_{i=1}^{n}\lambda_i)\left|u_i \cdot v_i\right|), \quad 0 \le \Psi(X,Q) \le 1 \tag{7.13}$$

where σ_i and λ_i are the i^{th} eigenvalues corresponding to i^{th} eigenvectors u_i and v_i of the matrices $X^T \cdot X$ and $Q^T \cdot Q$.

When the original datasets are mean centered, the above procedure is equivalent to applying PCA on the original data. The eigenvectors u_i and v_i are the corresponding principal components, while the eigenvalue-based weight in Eq. 7.13 is equal to the one obtained, if σ_i and λ_i are replaced by the variances of the corresponding components. The absolute value implies that the cosine of the acute angles is computed.

In the context of Statistical Process Control, Kano et al. [75] propose a distance measure for the purpose of monitoring process and identifying deviations from normal operating conditions. This measure is based on the Karhunen-Loeve expansion, which is mathematically equivalent to PCA. According to their approach, there is one multivariate time series $X_{n1\times p}$ that is defined as the reference dataset and represents a process under normal operating conditions. When data arrives from this process that corresponds to a

different time interval $Q_{n2\times p}$, the objective is to identify whether there is any substantial deviation from the normal operating conditions. The authors propose a dissimilarity measure between X and Q that requires applying eigenvalue decomposition on a new matrix C that comprises both X and Q (Eq. 7.14).

$$C = \begin{bmatrix} X \\ Q \end{bmatrix} \tag{7.14}$$

Moreover, their measure requires applying eigenvalue decomposition for second time at a subsequent step. Although this measure may be appropriate for monitoring processes, it involves applying eigenvalue decomposition twice during its calculation, which is the most computationally expensive part.

7.3 Proposed Approach

In this chapter, we propose a novel approach in multivariate time series similarity search that is based on Principal Component Analysis. The main difference with other proposed methods is that it does not require applying PCA on the query object. More specifically, PCA is applied on each object $X_{n\times p}$ of a database and the derived matrix of component weights $A_{p\times m}$ is stored (where m is the number of the retained components and ($m < p$). Each time instance q_i of a query object $Q_{n\times p}$ is projected on the plane derived by PCA and its new coordinates (q_i') are obtained (Eq. 7.15).

$$q_i' = q_i \cdot A, \quad i = 1,\ 2,\ \ldots n \tag{7.15}$$

In order to determine the error that this projection introduces to the new values, first we calculate the predicted values ($\hat{q}_i$) of q_i (Eq. 7.16), and then we compute the Squared Prediction Error (SPE) (Eq. 7.17).

$$\hat{q}_i = q_i' \cdot A^T, \quad i = 1,\ 2,\ \ldots n \tag{7.16}$$

$$SPE_i = \sum_{j=1}^{p} (q_{ij} - \hat{q}_{ij})^2, \quad i = 1,\ 2,\ \ldots n \tag{7.17}$$

SPE or Q is a well-known statistic in Multivariate Statistical Process Control [95] representing the squared perpendicular distance of a time instance from the plane. In this chapter, we propose the summation of these errors (over all time instances) to be used as a distance measure between X and Q (Eq. 7.18).

$$SPE_{dist}(X, Q) = \sum_{i=1}^{n} \sum_{j=1}^{p} (q_{ij} - \hat{q}_{ij})^2 \quad (7.18)$$

The main concept is that, the most similar object in a database is defined to be the one, whose principal components describe more adequately the query object with respect to reconstruction error. A similar approach can be found in the work of Barbic et al. [11], who propose a technique for the purpose of segmenting motion capture data into distinct motions. However, the authors utilize the squared error of the projected values and not the predicted values, as we propose in our work. Moreover, they focus on an application that involves one multivariate time series, which should be segmented.

As it was mentioned earlier, the proposed approach does not apply the computationally expensive PCA technique on the query object. Moreover, we provide a method that further speeds up the calculation of SPE_{dist}, hopefully, without affecting substantially the quality of similarity search. This method involves applying a dimensionality reduction technique on each one of the time series that form the query object, as a pre-processing step. The proposed technique is the Piecewise Aggregate Approximation (PAA) that was introduced independently by Keogh et al. [77] and, Yi and Faloutsos [161]. PAA is a well-known representation in the data mining community that can be extremely fast to compute. This technique segments a time series of length n into N consecutive sections of equal-width and calculates the corresponding mean for each one. The series of these means is the new representation of the original series. According to this approach, a query object that consists of p time series of length n is transformed to an object of p time series of length N. Under this transformation, we only need a fraction (N / n) of the required calculations in order to compute SPE_{dist}. Equivalently, the required calculations will be executed n / N time faster than the original ones. The consequence of this method in the quality of similarity search depends mainly on the quality of PAA representation within a specific dataset. Intuitively, SPE_{dist} is computed on a set of time instances, which may be considered as representatives of the original ones.

Example 7.1

Table 7.1 presents three MTS objects ($O1$, $O2$, Q) of two dimensions (X_1, X_2). Suppose that Q is the query object and the objective is to determine which one of $O1$ and $O2$ is the most similar object to Q. Figure 7.1 presents the graphs of these objects with respect to time.

Table 7.1 An example of three 2-dimensional MTS objects

	$O1$		$O2$		Q	
Time	X_1	X_2	X_1	X_2	X_1	X_2
1	1	2	6	-3	4	2
2	3	3	8	-5	5	3
3	3	5	5	-3	4	4
4	2	7	7	-6	5	7
5	2	4	6	-4	6	4

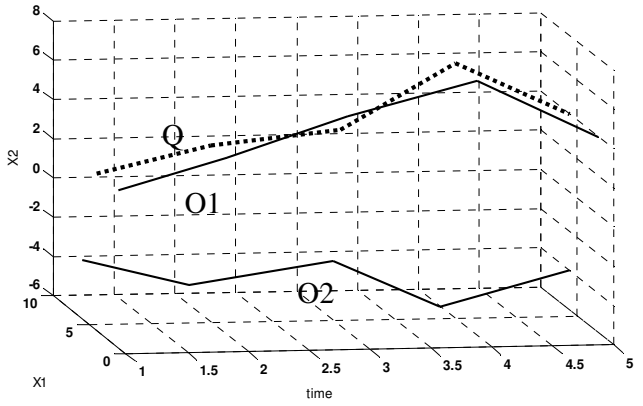


Figure 7.1 The 3-d graph of the three 2-dimensional MTS objects with respect to time. Dotted line corresponds to the query object, whereas solid lines correspond to two objects from the database.

The first step is to transform the original values of these objects by subtracting the corresponding (column) means from each one of them (Table 7.2). Then, Principal Component Analysis is applied only on $O1$ and $O2$ and the corresponding component weight matrices A_{O1} and A_{O2} (Table 7.3) are derived. The first columns of A_{O1} and A_{O2} determine the first principal components, PC_{O1} and PC_{O2}, of $O1$ and $O2$ respectively. In Figure 7.2, we

see the scatter diagram of $O1$ and $O2$ along with their first principal components. Black color corresponds to object $O1$ and gray color corresponds to object $O2$.

Table 7.2 The MTS objects centered about the corresponding means

	$O1$		$O2$		Q	
Time	X_1	X_2	X_1	X_2	X_1	X_2
1	-1.2	-2.2	-0.4	1.2	-0.8	-2
2	0.8	-1.2	1.6	-0.8	0.2	-1
3	0.8	0.8	-1.4	1.2	-0.8	0
4	-0.2	2.8	0.6	-1.8	0.2	3
5	-0.2	-0.2	-0.4	0.2	1.2	0

Table 7.3 The component weight matrices of $O1$ and $O2$. The i[th] column corresponds to the i[th] component.

A_{O1}		A_{O2}	
0.1452	-0.9894	0.6437	-0.7653
0.9894	0.1452	-0.7653	-0.6437

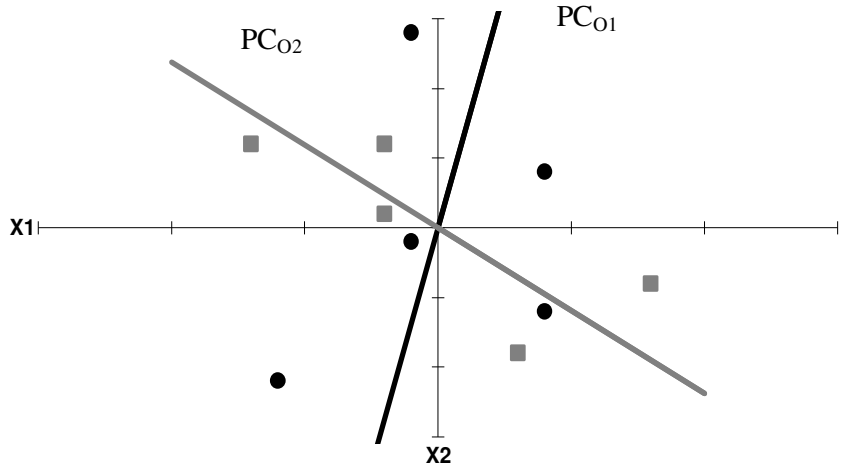


Figure 7.2 The scatter graph of the 2-dimensional MTS objects from the database along with their corresponding first principal components. Black color indicates object O1 and gray color indicates object O2.

According to the proposed approach, m components should be retained ($m < p$). In this example, m is set equal to one. The objects $O1$ and $O2$ are represented by the

component weight matrices, where only the first column is retained. Let denote these two matrices A'_{O1} and A'_{O2}.

First, we will find the distance of Q from $O1$ as follows. Each time instance q_i of Q is projected on the plane derived by PC_{O1} and its new coordinates (q'_i) are obtained by applying Eq. 7.15. The projected object Q' is given in Eq. 7.19.

$$Q' = Q \times A'_{O1} = \begin{pmatrix} -0.8 & -2 \\ 0.2 & -1 \\ -0.8 & 0 \\ 0.2 & 3 \\ 1.2 & 0 \end{pmatrix} \times \begin{pmatrix} 0.1452 \\ 0.9894 \end{pmatrix} = \begin{pmatrix} -2.0950 \\ -0.9604 \\ -0.1162 \\ 2.9972 \\ 0.1743 \end{pmatrix} \tag{7.19}$$

In order to determine the error that this projection introduces to the new values, we calculate the predicted values of q_i (Eq. 7.16). The predicted object $\hat{Q}$ with respect to object $O1$ is given in Eq. 7.20.

$$\hat{Q} = Q' \times A'^{\,T}_{O1} = \begin{pmatrix} -2.0950 \\ -0.9604 \\ -0.1162 \\ 2.9972 \\ 0.1743 \end{pmatrix} \times \begin{pmatrix} 0.1452 & 0.9894 \end{pmatrix} = \begin{pmatrix} -0.3042 & -2.0738 \\ -0.1395 & -0.9502 \\ -0.0169 & -0.1149 \\ 0.4352 & 2.9655 \\ 0.0253 & 0.1724 \end{pmatrix} \tag{7.20}$$

Finally, the computation of the proposed measure SPE_{dist} involves the summation of the squared errors, that is, the squared differences between the actual values (Q) and the predicted values ($\hat{Q}$):

$$SPE_{dist}(O1, Q) = \sum_{i=1}^{5} \sum_{j=1}^{2} (q_{ij} - \hat{q}_{ij})^2 = 2.4615 \tag{7.21}$$

Similarly, the distance of Q from $O2$ is found to be:

$$SPE_{dist}(O2, Q) = 9.4109 \tag{7.22}$$

According to Eq. 7.21 and Eq. 7.22, the object $O1$ is more similar to Q than the object $O2$. This conclusion is justified also from Figure 7.1. Intuitively, the proposed measure represents the sum of the perpendicular distances of each time instance of the query object from the first principal component of the corresponding object. Figure 7.3 shows the scatter graph of Q and the first principal components of $O1$ and $O2$. The most similar object to Q is the one whose principal component is "closer" to the time instances of Q.

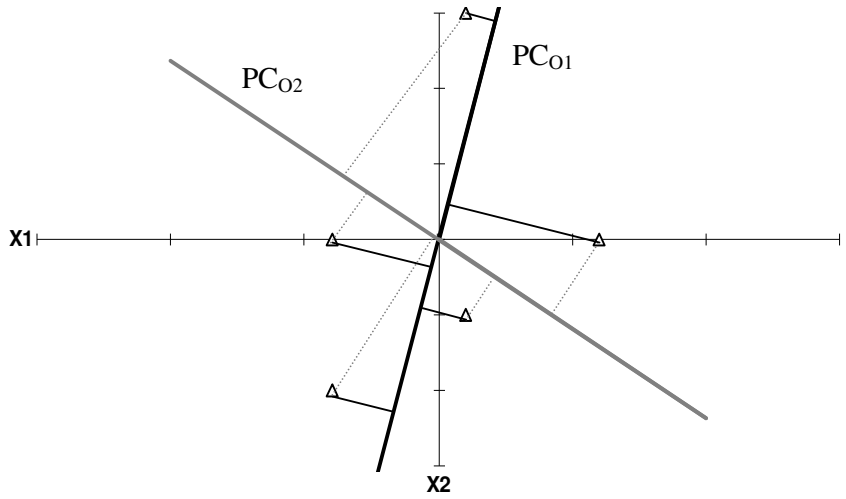


Figure 7.3 The scatter graph of the query object along with its corresponding distances from the first principal components of the 2-dimensional MTS objects.

7.4 Experimental Methodology

The experiments have been conducted on three real-world datasets and one synthetically created dataset that have been used extensively in the literature and are described in Section 7.4.1. Section 7.4.2 presents the evaluation methods and Section 7.4.3 discusses the rival measures along with their corresponding settings.

7.4.1 Datasets

Table 7.4 provides a summary of the main characteristics of the utilized datasets in the experiments. The first dataset relates to Australian Sign Language (AUSLAN), which contains sensor data gathered from 22 sensors placed on the hands (gloves) of a native AUSLAN speaker. The objective is the identification of a distinct sign. There are 95 distinct

signs, each one performed 27 times. In total, there are 2,565 signs in the dataset. More technical information can be found in [72].

The second dataset, HUMAN GAIT, involves the task of identifying a person at a distance. Data are captured using a Vicon 3D motion capture system, which generates 66 values at each time instance. 15 persons participated in this experiment and were required to walk in 3 sessions, at 4 different speeds, 3 times for each speed. In total, there are 540 walk sequences. More technical information can be found in [142].

The third dataset relates to EEG data that arises from a large study to examine EEG correlates of genetic predisposition to alcoholism [12]. It contains measurements from 64 electrodes placed on the scalp and sampled at 256 Hz (3.9-msec epoch) for 1 second. The experiments were conducted on 10 alcoholic and 10 control subjects. Each subject was exposed to 3 different stimuli, 10 times for each one. This dataset is provided in the form of a train and a test set, both consisting of 600 EEG's. The test data was gathered from the same subjects as with the training data, but with 10 out-of-sample runs per subject per paradigm.

Finally, the transient classification benchmark (TRACE) is a synthetic dataset designed to simulate instrumentation failures in a nuclear power plant [130]. There are 4 process variables, which generate 16 different operating states, according to their co-evolvement through time. There is an additional variable, which initially takes on the value of 0, until the start of the transient occurs and its value changes to 1. We retain only that part of data, where the transient is present. For each state, there are 100 examples. The dataset is separated into train and test sets each consisting of 50 examples per state.

Table 7.4 Description of datasets

DATASET	# of variables	mean length	# of classes	size of class	size of dataset
AUSLAN	22	57	95	27	2565
HUMAN GAIT	66	133	15	36	540
EEG	64	256	2	600	1200
TRACE	4	250	16	100	1600

7.4.2 Evaluation Methods

In order to evaluate the performance of the proposed approach in similarity search, we conducted several experiments in three phases.

First, we perform one-nearest neighbor classification (1-NN) and evaluate it by means of classification error rate. We use 9-fold cross validation for AUSLAN and HUMAN GAIT datasets taking into account all the characteristics of the experiments, while creating the subsets. The observed differences in the error rates among the various methods were

statistically tested. Due to the small number of subsets and to the violation of normality assumption in some cases, Wilcoxon Signed-Rank tests were performed at 5% significance level. For the EEG and TRACE datasets, we use the existing train and test sets.

Second, we perform leave-one-out k-NN similarity search and evaluate it by plotting the recall-precision graph [34]. In particular, every object in the dataset is considered as a query. Then the r most similar objects are retrieved, where r is the smallest number of objects that should be retrieved in order to obtain k objects of the same class with the query ($1 \le k \le size\_of\_class - 1$). The precision and recall pairs corresponding to the values of k are calculated. Finally, the average values of precision and recall are computed for the whole dataset. Precision is defined as the proportion of retrieved objects that are relevant to the query, whereas Recall is defined as the proportion of relevant objects that are retrieved relative to the total number of relevant objects in the dataset. In these experiments, the training and testing datasets of EEG and TRACE are merged.

Third, we evaluate the trade-off between classification accuracy and speed of calculating the proposed measure SPE_{dist} by applying 1-NN classification on objects that have been pre-processed as described in Section 7.3.

All the necessary codes and experiments were developed in MATLAB, whereas the statistical analysis was performed in SPSS.

7.4.3 Rival Measures

The similarity measures that were tested on our experiments are Sim_{PCA}, S^{λ}_{PCA}, Eros, kWAS, and SPE_{dist}. For comparison reasons, we also included in the experiments the Euclidean distance. Since this measure requires datasets of equal number of time instances, we decided to apply linear interpolation on the original datasets and set the length of the time series equal to the corresponding mean length (Table 7.4). The transformed datasets were utilized only when Euclidean distance was applied. The rest of the measures we reviewed in Section 7.2 are not included in these experiments because they take into consideration the differences among the original values, whereas in our experiments, the measures are calculated on the mean centered values.

Regarding Eros, the weight vector w was computed by averaging the variances of each component across the objects of the training dataset and normalizing them so that $\sum w_i = 1,$ for $i = 1, 2, ..., p$. The authors propose alternative ways of computing the weight vector, which have not been tested here.

All other measures require determining the number of components m to be retained. For AUSLAN, HUMAN GAIT and EEG, we have conducted classification for

consecutive values of m between 1 and 20. For m = 20, at least 99% of the total variation is retained for all objects in AUSLAN and HUMAN GAIT, whereas at least 90% of the total variation is retained for all objects in EEG. For TRACE, we have conducted classification for all possible values of m (m = 1, 2, 3, 4). Precision-Recall graphs are plotted for the "best" value of m that it was observed in the classification experiments. In general, this value is different for each measure.

Principal Component Analysis is performed on the covariance matrices. For comparison reasons, the similarity measures kWAS, and SPE_{dist} were computed on the mean centered values.

7.5 Results

We provide and discuss the results of 1-NN classification for each dataset separately in Section 7.5.1. In particular, we present the classification error rates that the tested measures achieved across various values of m (the number of components retained), and we also report the m that corresponds to the lowest error rate for each measure. In Section 7.5.2, the results of performing leave-one-out k-NN similarity search are presented in precision-recall graphs for each dataset. Finally, in Section 7.5.3, we provide and discuss the effect SPEdist with PAA has on the classification accuracy for various degrees of speed up.

7.5.1 1-NN Classification

In the following figures, the classification error rates are presented graphically for various values of m (the number of components retained) for each dataset. For the first three datasets, we show the error rates up to that value of m beyond which the behavior of similarity measures does not change significantly. For the TRACE dataset, which has only four variables, we show error rates for values of m up to three. Note that Euclidean Distance (ED) and Eros do not require this parameter, thus their corresponding rates are constants across m.

Regarding the first three datasets (Figure 7.4, Figure 7.5, Figure 7.6), we observe that all measures seem to achieve the lowest error rate, when only a few components are retained. Moreover, as the number of components is further increased, the improvement in error rates seems to be negligible. In AUSLAN dataset (Figure 7.4), the performance of SPE_{dist} and Sim_{PCA} deteriorates with the increase of m. Note that these two measures do not take into account the variance that each component explains, contrary to the other three PCA-based measures. A second observation is that the performance of SPE_{dist} is comparable,

if not better, to the "best" measure in each one of the three datasets. Regarding TRACE, which consist of only 4 variables, ED achieves considerably lower error rates than any other measure (Figure 7.7).

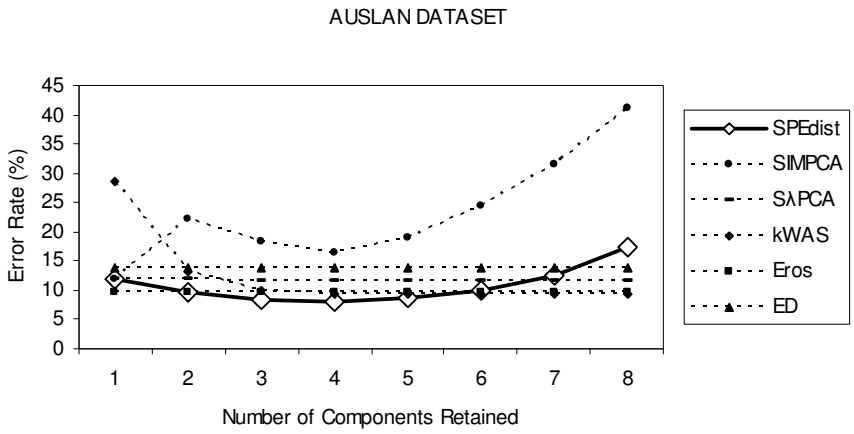


Figure 7.4 1-NN classification error rates (AUSLAN dataset)

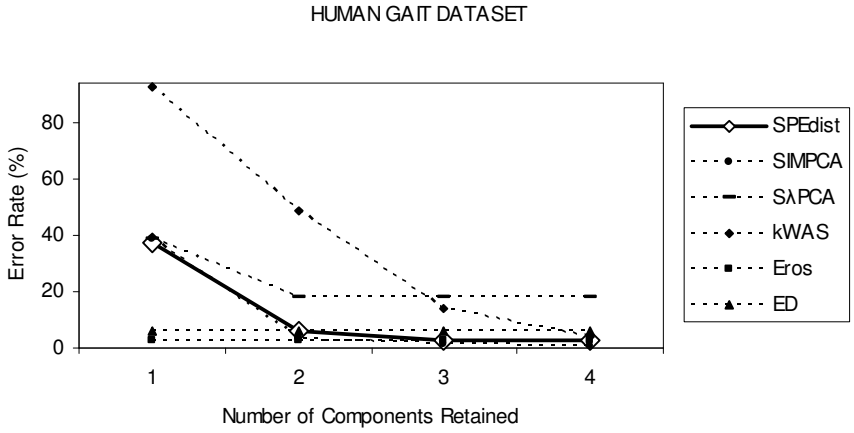


Figure 7.5 1-NN classification error rates (HUMAN GAIT dataset)

In Table 7.5, the lowest classification error rates are presented along with the corresponding number of the retained components. First, we will compare similarity/distance measures with respect to each dataset separately. For the AUSLAN dataset, it seems that SPE_{dist} produces at least similar results to Eros and kWAS. Statistically testing their differences across the specific subsets, SPE_{dist} produces better results than all ($p < 0.01$), except from kWAS ($p = 0.066$).

Regarding the HUMAN GAIT dataset, Sim_{PCA}, Eros, kWAS and SPE_{dist} seems to provide the best results. Statistically testing their differences across the specific subsets, Sim_{PCA} produces better results than all ($p < 0.05$), whereas the performance of Eros, kWAS and SPE_{dist} is statistically similar ($p > 0.05$).

For the EEG dataset, SimPCA and SPEdist seem to provide considerably better results than other measures, with classification error rates of 0.00% and 1.50% respectively, when the next best performing measure, Eros, presents a classification error rate of 14.83%.

Finally, for the TRACE dataset that consists of only 4 variables, Euclidean distance, a non-PCA-based measure, performs essentially better than all measures with 3.9% classification error rate. The next best performing measures are Eros and kWAS with classification error rates of 21.38% and 21.88% respectively.

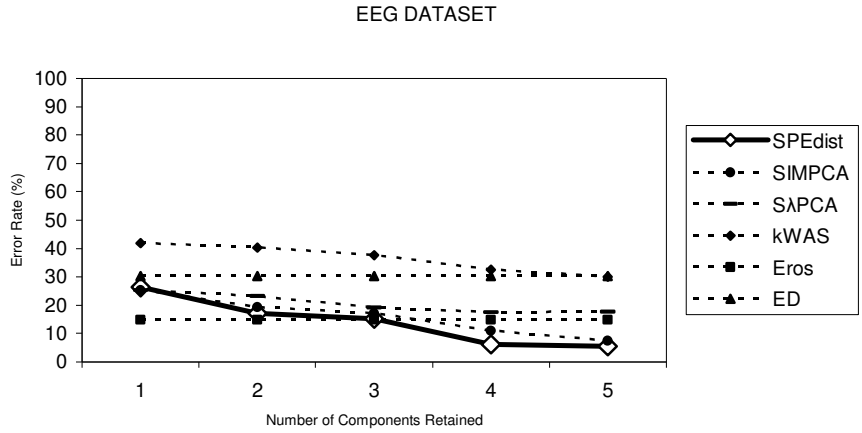


Figure 7.6 1-NN classification error rates (EEG dataset)

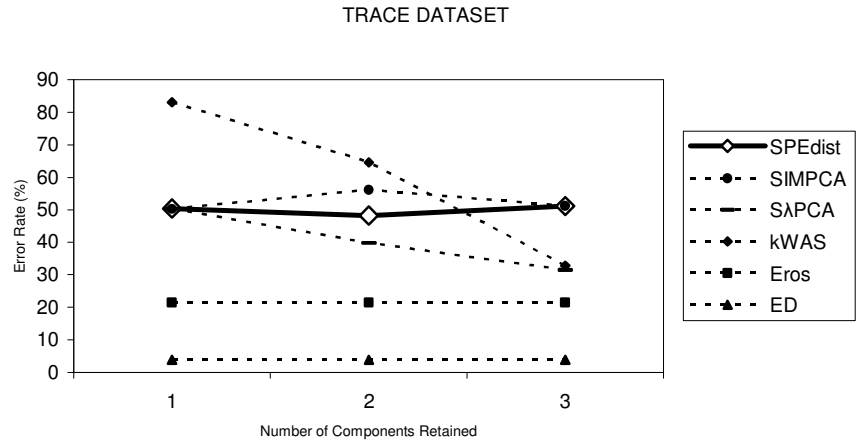


Figure 7.7 1-NN classification error rates (TRACE dataset)

Table 7.5 1-NN classification error rates (%)

Dataset / Measure	AUSLAN	HUMAN GAIT	EEG	TRACE
ED	13.76	5.74	30.17	3.88
Sim_{PCA}	(1) 12.05	(8) 0.00	(14) 0.00	(1) 50.25
S^{λ}_{PCA}	(4) 11.46	(3) 17.78	(10) 16.50	(3) 31.38
Eros	9.71	2.96	14.83	21.38
kWAS	(6) 9.24	(12) 2.59	(17) 25.33	(4) 21.88
SPE_{dist}	(4) 7.95	(6) 2.59	(14) 1.50	(2) 48.25

Numbers in parentheses indicate the number of principal components retained. Lack of number indicates measures that exploit all components.

7.5.2 k-NN Similarity Search

In the following figures, the precision-recall graphs are presented for each dataset separately. The number of retained components is set equal to the one for which the corresponding measure provided the lowest classification error rates (

Table 7.5). Regarding the AUSLAN dataset (Figure 7.8), all measures seem to perform similarly to each other and better than the Euclidean distance. In HUMAN GAIT (Figure 7.9) and EEG (Figure 7.10), however, Sim_{PCA} and SPE_{dist} perform better than all. As mentioned in the previous section, these two measures do not take into consideration the explained variance of the retained components. This fact may imply that for these specific datasets, the "variance" information may not be significant. On the other hand, in AUSLAN dataset, where this information may be important, SPE_{dist} provides comparable results to other measures. In the final dataset, TRACE, Euclidean distance performs better for recall values up to 0.3, whereas kWAS performs better for greater recall values (Figure 7.11).

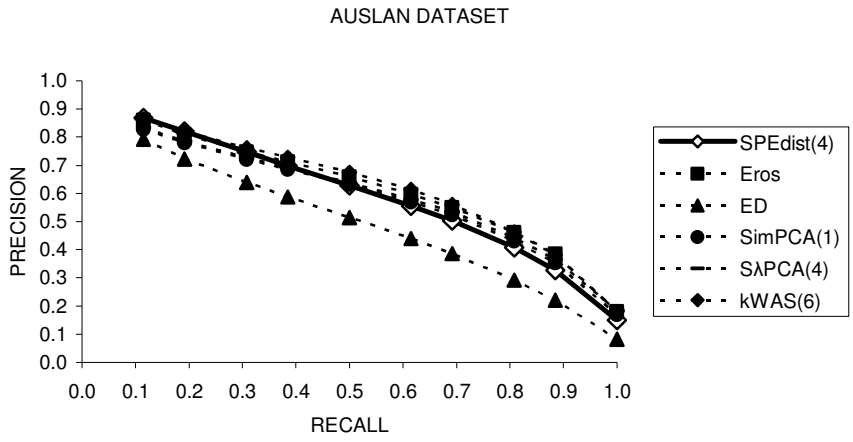


Figure 7.8 Precision-Recall graph for various measures (AUSLAN dataset)

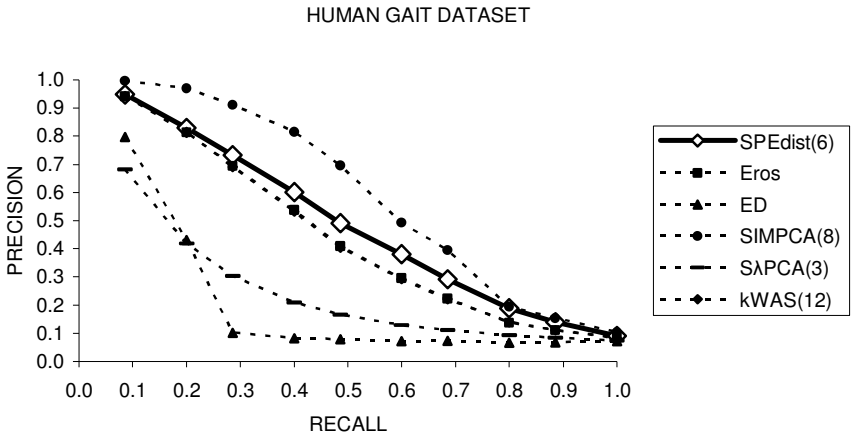


Figure 7.9 Precision-Recall graph for various measures (HUMAN GAIT dataset)

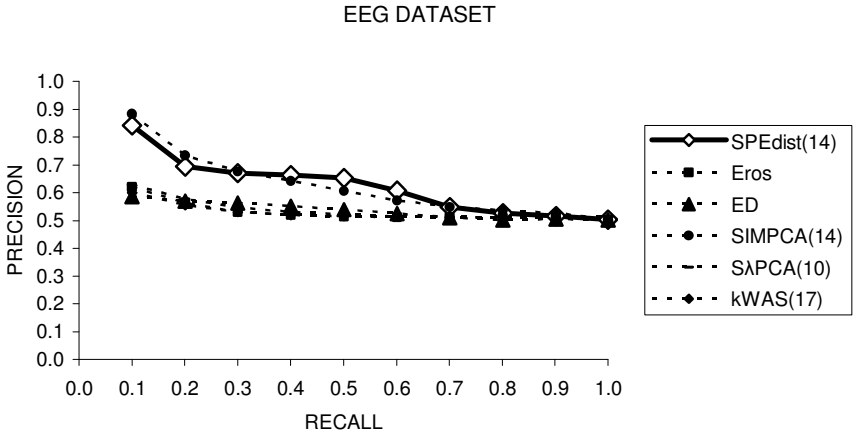


Figure 7.10 Precision-Recall graph for various measures (EEG dataset)

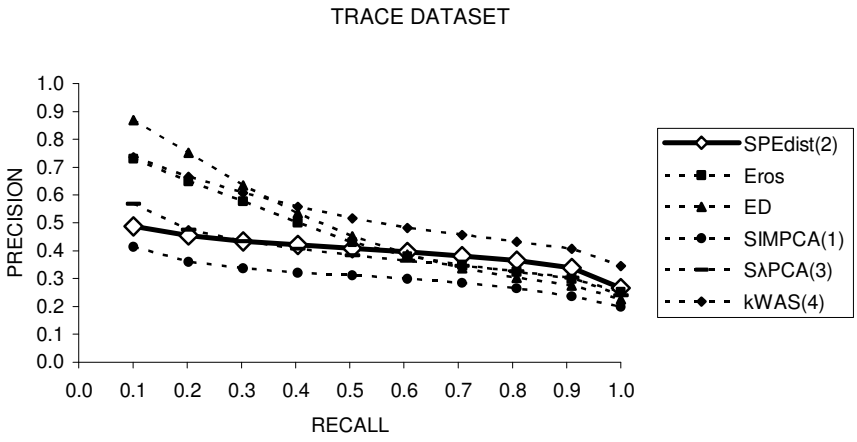


Figure 7.11 Precision-Recall graph for various measures (TRACE dataset)

7.5.3 Speeding up Calculation of SPEdist

The idea is to apply PAA on each one of the time series that comprise the query object (Section 7.3), in order to speed up the calculations of SPEdist. We experimented with various degrees of dimensionality reduction by using PAA to retain 10%, 20%, and 30% of the original dimensions of the query object, thus, expecting a 10x, 5x, and 3.33x speed up of the calculations, respectively.

Table 7.6 presents the effect the speed up has on the classification error rate. The number of the retained components is different among datasets and is set equal to the optimal value obtained in Section 7.5.1 (Table 7.5).

As it was expected, the classification error rate increases as the speed up increases. Nevertheless, in all datasets, we are able to achieve similar classification error rates by doing at most 20% of the required calculations (a 5x speed up). More specifically, for the AUSLAN dataset, even a 5x speed up provides better results than rival measures (Table 7.6). Regarding HUMAN GAIT, a 10x speed up results into exactly the same classification error rate as the one observed when full calculations were applied. In the EEG dataset, although the error rates differ significantly for the various degrees of speed up, the 10x speed up provides lower error rate than rival measures (except from SimPCA). Regarding TRACE, a 5x speed up results into almost the same classification error rate as the one observed when full calculations were applied.

Table 7.6 1-NN classification error rates for various degrees of dimensionality reduction on the query object

Percentage of retained dimensions	10%	20%	30%	100%
Speed up	10×	5×	3.33×	1×
AUSLAN	12.48	8.58	8.03	7.95
HUMAN GAIT	2.59	2.59	2.59	2.59
EEG	8.17	3.17	1.67	1.50
TRACE	59.63	50.13	49.38	48.25

Numbers in parentheses indicate the reduced length of the time series that form an object under classification.

7.6 Conclusion

The main contribution of this chapter is the introduction of a novel PCA-based approach in multivariate time series similarity search, which does not require for the query object to be analyzed in principal components. The calculation of the proposed distance measure can be further sped up by reducing the dimensionality of each one of the time series

that form the query object. Experiments were conducted on four widely utilized datasets and various measures were tested with respect to 1-NN classification and precision/recall.

There are three key observations with respect to the results of these experiments. First, there is no measure that that can be clearly considered as the most appropriate one for any dataset. Second, in three datasets, our approach provided significantly better results than Euclidean distance, whereas its performance was at least comparable to the other four PCA-based measures that they were tested. Third, there is strong evidence that the application of the proposed approach can be accelerated with little cost in the quality of similarity search. In all datasets, one tenth up to one third of the required calculations was adequate in order to achieve similar results to the full computation case.

CHAPTER 8

Conclusions – Future Work

In this dissertation, we propose several enhancements in the various aspects of similarity search within the Data Mining context. The objective of our work is to improve the efficiency of data mining techniques without substantially affecting the quality of the obtained results. All the proposed methods can be virtually applied on other types of data. However, we focus on time series data because it is generated in huge amounts by several procedures in almost every domain, such as in business, industry, medicine, meteorology etc. Time series data differs from other domains because it exhibits high dimensionality, high correlation feature and high levels of noise.

Similarity search is central to the majority of Time Series Data Mining tasks, since these tasks often require identifying series with similar shape or pattern. However, this identification requires the definition of a similarity/distance measure that is takes into consideration the distortions, such as noise and local misplacements in time, which are frequently present in data. In addition to that, when we encounter data of such high dimensionality, we inevitably have to consider the problem of scalability, that is, the way of applying data mining methods efficiently. This problem is often approached by representing original data in a domain of lower dimensionality and analyzing it in this domain.

In Chapter 3, we propose a Data Mining approach to control chart pattern recognition and we demonstrate that fairly simple representations and data mining methods produce comparable results to other more complex approaches such as Neural Networks. In addition to that, experimental results show the importance of the decision that we have to take with respect to the representation. Different representation schemes may affect substantially the quality of the results.

In Chapter 4, we propose a novel representation scheme for univariate time series, named D-PAA, along with an appropriate distance measure that lower bounds the Euclidean distance. D-PAA representation takes into consideration the local central tendencies and the corresponding dispersions of a time series. There are four main conclusions that are drawn from this work. The first conclusion is that the local dispersion present in a time series possesses a further discriminating power in conjunction with the corresponding central tendency. Moreover, in almost half of the datasets that are utilized in the experiments, the

information of local dispersions has greater power than the information of local central tendencies in identifying similar time series. The third conclusion is D-PAA performs consistently better than other popular representations with respect to classification accuracy. The minimum error rate is consistently similar or better throughout the majority of datasets. In addition to that, D-PAA performance is comparable to DTW, which constitutes a state of the art approach in measuring similarity among time series. Finally, an expected conclusion that can be drawn is that there is no representation scheme that outperforms all the others in every dataset. This conclusion refers mainly to their capability of capturing the most essential features of a time series, that is, to minimize the loss of important information inherent in a time series, while reducing dimensionality.

In Chapter 5, we introduce an improved method, named CLUREP, for performing one-nearest neighbor similarity search with respect to efficiency. Although this method can be applied in conjunction with a multidimensional indexing structure, we investigate its effectiveness the general case of sequential searching. The main conclusion is that CLUREP is considerably faster than sequential searching. More specifically, when the number of the generated clusters is equal to 20, CLUREP is at least 5 times faster in the majority of the datasets considered in the experiments. The fraction of the searched space ranges from 7% to 45% with an average value of 20%. A secondary conclusion is that when the number of the generated clusters increases, the performance of CLUREP improves. However, experiments show that the rate of improvement "slows down" after the generation of approximately 8 to 12 clusters. A final conclusion is that one can trade-off between accuracy and efficiency with respect to the requirements of the application under consideration.

In Chapter 6, we discuss several aspects of multivariate time series in the context of Data Mining and we provide a literature review of Multivariate Time Series Data Mining with respect to similarity search and relative applications. In addition to that, we investigate the implications of Principal Component Analysis in similarity search. The main conclusion is that there is an increasing interest on adjusting existed techniques to multivariate time series. However, the majority of the research papers focus on trajectories, which constitute at most four-dimensional time series. A second conclusion is that there has been a lot of work in exploiting Time Series Data Mining techniques from diverse application areas. This fact suggests a closer, interdisciplinary cooperation among research communities. A final conclusion is that Principal Component Analysis has not been extensively explored in the context of similarity search in multivariate time series and hence, it has the potential to offer more in the Data Mining field.

In Chapter 7, we propose a novel distance measure, named SPEdist, between multivariate time series, which requires a representation that is based on Principal Component Analysis. The main conclusion is that SPEdist provides significantly better results than Euclidean distance in three out of four datasets utilized in the experiments, whereas its performance is at least comparable to the best of the four other PCA-based measures that they are tested. The novelty of our approach is that it does not require the application of Principal Component Analysis on the query object. There is strong evidence that the application of the proposed approach can be further accelerated with little cost in the quality of similarity search. In all datasets, one tenth up to one third of the required calculations is adequate in order to achieve similar results to the full computation case.

There are two directions for our future research. The first direction is the improvement of the methods that are proposed in this dissertation. Regarding the distance measure that is based on D-PAA representation, we will focus on providing tighter lower bounds on the Euclidean distance and we will further investigate the weight of means and standard deviations for the purpose of providing a distance measure that is free of parameters. We will further examine the CLUREP method for the purpose of improving its efficiency. Regarding the Principal Component Analysis, we will develop a framework for the pre-processing phase with respect to similarity search. We will also focus on improving the speed up of the SPE_{dist} distance measure during the pre-processing stage by exploiting the features of other dimensionality reduction techniques.

The second direction for our future research is “applications”. There is an increasing demand for demonstrating the effectiveness of Time Series Data Mining techniques through real-world situations in order to validate their importance. Although there are a lot of real-world datasets that have been utilized as benchmarks datasets, the number of publications that describe and analyze thoroughly the corresponding applications is extremely limited.

REFERENCES

[1] Aach, J., Church, G.: Aligning gene expression time series with time warping algorithms. Bioinformatics **17**, pp. 495-508 (2001)

[2] Aggarwal, C.C., Hinneburg, A., Keim, D.A.: On the surprising behavior of distance metrics in high dimensional space. In: Database Theory – ICDT. Springer, Berlin / Heidelberg, pp. 420-434 (2001)

[3] Aggarwal, C.C.: On the effects of dimensionality reduction on high dimensional similarity search. In: Proc. 20th ACM SIGMOD-SIGACT-SIGART Symp. (PODS'01), pp. 256-266 (2001)

[4] Agrawal, R., Ghosh, S., Imielinski, T., Iyer, B., Swami, A.: An interval classifier for database mining applications. In: Proc. 18th Int'l. Conf. VLDB, pp. 560-573 (1992)

[5] Agrawal, R., Faloutsos, C., Swami, A.: Efficient similarity search in sequence databases. In: Proc. 4th Int'l. Conf. FODO, pp. 69-84 (1993)

[6] Agrawal, R., Lin, K., Sawhney, H.S., Shim, K.: Fast similarity search in the presence of noise, scaling and translation in time series databases. In: Proc. VLDB, pp. 490-501 (1995)

[7] Alcock, R.-J., Manolopoulos, Y.: Time-series Similarity Queries Employing a Feature-Based Approach. In: Proc. 7th Panhellenic Conference in Informatics (PCI'99), pp. III.1-9, (1999)

[8] Arabie, P., Hubert, L.J.: An overview of combinatorial data analysis. In: Clustering and Classification. World Scientific Publishing Co., New Jersey, pp. 5-63 (1996)

[9] Bagnall, A.-J., Janacek, G.J.: Clustering time series from ARMA models with clipped data. In: Proc. 10th ACM SIGKDD, pp. 49-58 (2004)

[10] Bakalov, P., Hadjieleftheriou, M., Keogh, E., Tsotras, V.J.: Efficient trajectory joins using symbolic representations. In: Proc. 6th Int. Conf. on Mobile Data Management, pp. 86-93 (2005)

[11] Barbic, J., Safonova, A., Pan, J.Y., Faloutsos, C., Hodgins, J.K., Pollard, N.S.: Segmenting motion capture data into distinct behaviors. In: Proc. Graphics Interface Conf., pp. 185-194 (2004)

[12] Begleiter, H.: The UCI KDD Archive [http://kdd.ics.uci.edu]. Irvine, CA: University of California, Department of Information and Computer Science, (1999)

[13] Bennett, K.P., Fayyad, U., Geiger, D.: Density-based indexing for approximate nearest-neighbor queries. In: Proc. 5th ACM SIGKDD, pp. 233-243 (1999)

[14] Berchtold, S., Keim, D.A., Kriegel, H.P.: The X-Tree: An index structure for high-dimensional data. In: Proc. 22nd VLDB, pp. 28-39 (1996)

[15] Berkhin, P.: A survey of clustering data mining techniques. In: Grouping Multidimensional Data. Springer, Berlin / Heidelberg, pp. 25-71 (2006)

[16] Berndt, D., Clifford, J.: Using dynamic time warping to find patterns in time series. AAAI-94 Workshop on Knowledge Discovery in Databases, pp. 229-248 (1994)

[17] Berry, M.W., Dumais, S.T., O'Brien, G.W.: Using linear algebra for intelligent information retrieval. SIAM Review, **37**(4), pp. 573-595 (1995)

[18] Beyer, K., Goldstein J., Ramakrishnan, R., Shaft, U.: When is Nearest Neighbor Meaningful?. In: Database Theory – ICDT'99. Springer, Berlin / Heidelberg, pp. 217-235 (1999)

[19] Biggs, D., De Ville, B., Suen, E.: A method of choosing multiway partitions for classification and decision trees. Journal of Applied Statistics **18**(1), pp. 49-62 (1991)

[20] Bozkaya, T., Yazdani, N., Ozsoyoglu, M.: Matching and indexing sequences of different lengths. In: Proc. CIKM'97, pp. 128-135 (1997)

[21] Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Wadsworth Int. Group, (1984)

[22] Buzan, D., Sclaroff, S., Kollios, G.: Extraction and clustering of motion trajectories in video. In: Proc. 17th ICPR, **(2)**, pp. 521-524 (2004)

[23] Cai, Y., Ng, R.: Indexing spatio-temporal trajectories with Chebyshev polynomials. In: Proc. ACM SIGMOD, pp. 599-610 (2004)

[24] Cappelli, R., Maio, D., Maltoni, D.: Similarity search using multi-space K-L. In: Proc. Int. Workshop on Similarity Search, IWOSS'99 -DEXA, pp. 155-160 (1999)

[25] Chakrabarti, K., Mehrotra, S.,: Local dimensionality reduction: A new approach to indexing high dimensional spaces. In: Proc. 26th VLDB'00, pp. 89-100 (2000)

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

[27] Chan K., Fu, A.W.: Efficient time series matching by wavelets. In: Proc. ICDE, pp. 126-133 (1999)

[28] Chapman, L., Thornes, J.E.: The use of geographical information systems in climatology and meteorology. Progress in Physical Geography, **27**(3), pp. 313-330 (2003)

[29] Chatfield, C.: The analysis of time series: An Introduction. Chapman & Hall / CRC, (2004)

[30] Chen, L., Ozsu, M.T., Oria, V.: Robust and fast similarity search for moving object trajectories. In: Proc. ACM SIGMOD, , Baltimore, MD, pp. 491-502 (2005)

[31] Cheng, C.S.: A multi-layer neural network model for detecting changes in the process mean. Computers & Industrial Engineering, **28**(1), pp. 51-61 (1995)

[32] Cheng, C.S.: A neural network approach for the analysis of control chart patterns. International Journal of Production Research, **35**(3), pp. 667-697 (1997)

[33] Cole, R., Shasha, D., Zhao, X.: Fast window correlations over uncooperative time series. In: Proc. 11th Int'l. Conf. ACM SIGKDD, pp. 743-749 (2005)

[34] Cover, T.M., Thomas, J.A.: Elements of Information Theory. JohnWiley & Sons Inc., (1991)

[35] Das, G., Gunopoulos, D., Mannila H.: Finding similar time series. In: Proc. 1st PKDD Symp., pp. 88-100 (1997)

[36] Das G., Lin, K.I., Mannila, H., Ranganathan, G., Smyth, P.: Rule discovery from time series, In: Proc. KDD 1998, pp. 16-22 (1998)

[37] Dasgupta, D., Forrest, S.: Novelty detection in time series data using ideas from immunology. In: Proc. International Conference on Intelligent Systems, pp 19-21 (1996)

[38] Dattatreya, G.R., Kanal, L.N.: Decision Trees in Pattern Recognition. Progress in Pattern Recognition, **2**, pp. 189-239 (1985)

[39] Daw, C.S., Finney, C.E.A., Tracy, E.R.,: A review of symbolic analysis of experimental data. Review of Scientific Instruments, **74**(2), pp. 915-930 (2003)

[40] Deming, W.E.,: Quality Productivity and Competitive Position. Cambridge, MA: Massachusetts Institute of Technology, Center for Advanced Engineering Study, (1982)

[41] Deming, W.E.: Out of the crisis. Cambridge, MA: Massachusetts Institute of Technology, Center for Advanced Engineering Study (1986)

[42] Esposito, F. Malerba, D. Semeraro, G. Kay, J.: A comparative analysis of methods for pruning decision trees. IEEE Transactions on Pattern Analysis and Machine Intelligence, **19** (5), pp.476-491 (1997)

[43] Evans, J.R., Lindsay, W.M.: A framework for expert system development in Statistical Quality Control. Computers & Industrial Engineering, **14**(3), pp. 335-343 (1988)

[44] Faloutsos, C., Ranganathan, M., Manolopoulos, Y.: Fast subsequence matching in time series databases. In: Proc. ACM SIGMOD, pp. 419-429 (1994)

[45] Faloutsos, C.: Mining Time Series Data. Tutorial ICML'03, (2003)

[46] Fayyad, U.-M., Piatetsky-Shapiro, G,. Smyth, P.: From Data Mining to Knowledge Discovery: An Overview. In: Advances in Knowledge Discovery and Data Mining, AAAI / MIT Press, pp. 1-30 (1996)

[47] Ferhatosmanoglu, H., Tuncel, E., Agrawal, D., El Abbadi, A.: Approximate nearest neighbor searching in multimedia databases. In: Proc. 17th IEEE Int'l Conf. Data Eng., pp. 503-511 (2001)

[48] Forbes, K., Fiume, E.: An efficient search algorithm for motion data using weighted PCA. In: Proc. ACM SIGGRAPH/ Eurographics Symp. on Computer Animation, pp. 67-76 (2005)

[49] Fu, T.C., Chung, F.L., Luk, R. Ng, C.M.: A specialized binary tree for financial time series representation. 10th ACM SIGKDD Data Mining workshop on Temporal Data Mining, (2004)

[50] Fujimaki, R., Yairi, T., Machida, K.: An approach to spacecraft anomaly detection problem using kernel feature space. In: Proc. 9th PAKDD'05, pp. 785-790 (2005)

[51] Gaede, V., Gunther, O.: Mutidimensional access methods. In: ACM Computing Surveys. **30**(2), pp. 170–231 (1998)

[52] Ge, X., Smyth, P.: Deformable Markov model templates for time series pattern matching. In: Proc. ACM SIGKDD, pp. 81-90 (2000)

[53] Geurts, P.: Pattern extraction from time series for classification. In: Proc. 5th European Conference on Principles of Data Mining and Knowledge Discovery, pp. 115-127 (2001)

[54] Gionis, A., Mannila H.: Finding recurrent sources in sequences. In: Proc. 7th RECOMB'03, pp. 123-130 (2003)

[55] Gower, J.C.: Multivariate Analysis and Multidimensional Geometry. The Statistician, **17**(1), pp. 13–28 (1967).

[56] Guh, R.S.: A hybrid learning based model for on-line detection and analysis of control chart patterns. Computers & Industrial Engineering, **49**(1), pp. 35-62 (2005)

[57] Gunopulos, D., Das, G.: Time series similarity measures and time series indexing. In: Proc. ACM SIGMOD, pp. 624 (2001)

[58] Guttman, A.: R-trees: A dynamic index structure for spatial searching. In: Proc. ACM SIGMOD, pp. 47-57 (1984)

[59] Hand, D., Mannila, H., Smyth, P.: Principles of Data Mining. Cambridge, Mass., MIT Press, (2001)

[60] Hartigan, J.: Clustering Algorithms. John Wiley & Sons, New York, NY (1975)

[61] Hastie, T., Tibshirani, R., Friedman, J.: The elements of statistical learning. Springer series in Statistics (2001)

[62] Hellerstein, Papadimitriou, Koutsoupias,: On the Analysis of Indexing Schemes. In: Proc. 16th Symp. ACM SIGACT-SIGMOD-SIGART, pp. 249-256 (1997)

[63] Hosni, Y.A., Elshennawy, A.K.: Quality control & inspection: Knowledge-based quality control system. Computers & Industrial Engineering, **15**(1-4), pp. 331-337 (1988)

[64] http://www.cs.ucr.edu/~eamonn/time_series_data/

[65] Hwarng, H.B., Hubele, N.F.: X-bar control chart pattern identification through efficient off-line neural network training. IIE Transactions, **25**, pp. 27-39 (1993a)

[66] Hwarng, H.B., Hubele, N.F.: Back-propagation pattern recognizers for x-bar control charts: Methodology and performance. Computers & Industrial Engineering, **24**(2), pp. 219-235 (1993b)

[67] Itakura, F.: Minimum prediction residual principle applied to speech recognition. IEEE Trans. Accoustics, Speech and Signal Processing, **23**, pp. 55-72 (1975)

[68] Jain, A., Dubes, R.: Algorithms for Clustering Data. Prentice-Hall, Englewood Cliffs, NJ, (1988)

[69] Jenkins, O.C., Mataric, M.J.: Automated derivation of behavior vocabularies for autonomous humanoid motion. In: Proc. Int'l. Conf. AAMAS'03, pp. 225-232 (2003)

[70] Johannesmeyer, M.C.: Abnormal situation analysis using pattern recognition techniques and historical data. M.S. thesis, UCSB, Santa Barbara, CA, (1999).

[71] Jolliffe, I.T.: Principal Component Analysis. Springer, New York (2004)

[72] Kadous, M.W.: Temporal Classification: Extending the classification paradigm to multivariate time series. Ph.D. Thesis, School of Computer Science and Engineering, University of New South Wales, (2002)

[73] Kahveci, T., Singh, A., Gurel, A.: Similarity searching for multi-attribute sequences. In: Proc. 14th SSDBM, pp. 175-184 (2002)

[74] Kalpakis, K., Gada, D., Puttagunta, V.: Distance measures for effective clustering of ARIMA time-series. In: Proc. IEEE Int. Conf. Data Mining, pp 273-280 (2001)

[75] Kano, M., Nagao, K., Ohno, H., Hasebe, S., Hashimoto, I.: Dissimilarity of process data for statistical process monitoring. In: Proc. IFAC Symp. ADCHEM, **I**, pp. 231-36 (2000)

[76] Kano, M., Nagao, K., Hasebe, S., Hashimoto, I., Ohno, H., Strauss, R., Bakshi, B.R.: Comparison of multivariate statistical process monitoring methods with applications to the Eastman challenge problem. Computers & Chemical Engineering, **26**(2) pp. 161-174 (2002)

[77] Kanth Ravi, K.V., Agrawal, D., Abbadi, A.E.: Dimensionality reduction for similarity searching dynamic databases. ACM SIGMOD, 27 (2), pp 166-176 (1998)

[78] Karamitopoulos L., Evangelidis G., Dervos D., "Control Chart Pattern Recognition - A Time Series Data Mining Approach", In Proc. 18th National Conference on Planning, Information Retrieval and Knowledge Management, Hellenic Operations Research Society, pp. 195-204, Kozani, Greece, June 15-17 (2006)

[79] Karamitopoulos L., Evangelidis G., "Recent Advances in Time-Series Data Mining", In Proc. 1st International Scientific Conference, eRA: The Contribution of Information Technology to Science, Economy, Society and Education, Tripolis, Greece, September 16-17 (2006)

[80] Karamitopoulos L., Evangelidis G., "Current Trends in Time Series Representation", In Proc. Panhellenic Conference in Informatics (PCI), Patras, Greece, May 18-20, 2007

[81] Karamitopoulos L., Evangelidis G., "PCA-based Similarity Search: Pre-processing & Distance Measures", In Proc. 2nd International Scientific Conference, eRA: The Contribution of Information Technology to Science, Economy, Society and Education, Athens, Greece, September 22-23, (2007)

[82] Karamitopoulos L., Evangelidis G., Dervos D., "Multivariate Time Series Data Mining: PCA-based Measures for Similarity Search", In Proc. 4th International Conference in Data Mining (DMIN), Las Vegas, Nevada, USA, July 14-17 (2008)

[83] Karamitopoulos L., Evangelidis G., Dervos D., "Time Series Data Mining and Control Chart Pattern Recognition", Operational Research An International Journal (ORIJ), Springer, (2009), to appear

[84] Karamitopoulos L., Evangelidis G., Dervos D., "PCA-based Time Series Similarity Search", Annals of Information Systems (AIS) Springer, (2009), to appear

[85] Karamitopoulos L., Evangelidis G., "A Dispersion-based PAA Representation for Time Series", World Congress on Computer Science and Information Engineering (CSIE 2009), Los Angeles/Anaheim, USA, March 31 - April 2, (2009)

[86] Keogh, E., Smyth, P.: A probabilistic approach to fast pattern matching in time series databases. In: Proc. KDD'97, pp. 24-30 (1997)

[87] Keogh, E., Chakrabarti, K., Pazzani, M., Mehrotra, S.: Dimensionality reduction for fast similarity search in large time series databases. Knowledge and Information Systems, **3**(3), pp. 263-286 (2001)

[88] Keogh, E., Chu, S., Hart, D., Pazzani, M.: An online algorithm for segmenting time series. In: Proc. IEEE Int'l. Conf. Data Mining, pp 289-296 (2001)

[89] Keogh, E., Lonardi, S., Chiu, B.Y.: Finding surprising patterns in a time series database in linear time and space. In: Proc. ACM SIGKDD, pp. 550-556 (2002)

[90] Keogh, E., Ratanamahatana, C.: Exact Indexing of Dynamic Time Warping. In: Proc. 28th VLDB, pp. 406-417 (2002)

[91] Keogh, E., Kasetty, S.: On the need for time series data mining benchmarks: A survey and empirical demonstration. Data Mining and Knowledge Discovery. **7**(4), pp. 349-371 (2003)

[92] Kim, S., Park, S., Chu, W.: An index-based approach for similarity search supporting time warping in large sequence databases. In: Proc. 17th ICDE, pp. 607-614 (2001)

[93] Kontaki, M., Papadopoulos, A.N., Manolopoulos, Y.: Continuous trend-based classification of streaming time series. In: Proc. 9th ADBIS, (2005)

[94] Korn, F., Jagadish, H., Faloutsos, C.: Efficiently supporting ad hoc queries in large datasets of time sequences. In: Proc. ACM SIGMOD, pp. 289-300 (1997)

[95] Kresta, J., MacGregor, J.F., Marlin, T.E.: Multivariate statistical monitoring of process operating performance. The Canadian Journal of Chemical Engineering, **69**, pp. 35-47 (1991)

[96] Krzanowski, W.: Between-groups comparison of Principal Components. JASA, **74**(367), pp. 703-707 (1979)

[97] Lee, S.L., Chun, S.J., Kim, D.H., Lee, J.H., Chung, C.W.: Similarity search for multidimensional data sequences. In: Proc. ICDE, pp. 599-608 (2000)

[98] Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady, **10**, pp 707-710 (1966)

[99] Li, C., Garcia-Molina, H., Wiederhold, G.: Clustering for approximate similarity search in high-dimensional spaces. IEEE Trans. Knowl. Data Eng., **14**(4), pp. 792–808 (2002)

[100] Li, C., Prabhakaran, B.: A similarity measure for motion stream segmentation and recognition. In: Proc.6th Int. Workshop MDM/KDD, pp. 89-94 (2005)

[101] Li, C.S., Yu, P.S., Castelli, V.: MALM: A framework for mining sequence database at multiple abstraction levels. In: Proc. 9th CIKM, pp. 267-272 (1998)

[102] Li, T., Li, Q, Zhu, S, Ogihara, M.: A survey on wavelet applications in data mining. SIGKDD Explorations, **4**(2), pages 49-68 (2002)

[103] Liao, T.W.: Clustering of time series data – a survey. Pattern Recognition, **38**, pp. 1857-1874 (2005)

[104] Lin, J., Keogh, E., Lonardi, S., Patel, P.: Finding motifs in time series. In: Proc. 2nd workshop on Temporal Data Mining, ACK SIGKDD, pp. 53-68 (2002)

[105] Lin, J., Keogh, E., Lonardi, S., Chiu, B.: A Symbolic representation of time series with implications for streaming algorithms. In: Proc. 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, pp 2-11 (2003)

[106] Lin, K.L., Jagadish, H.V., Faloutsos, C.: The TV-tree: an index structure for high-dimensional data. VLDB Journal, **3**(4), pp. (1994)

[107] Lkhagva, B., Suzuki, Y., Kawagoe, K.: New time series data representation ESAX for financial applications. In: Proc. 22nd ICDEW'06, pp. x115 (2006)

[108] Loh, W. Y., Shih, Y.S.: Split selection methods for classification trees. Statistica Sinica, **7**, pp. 815-840 (1997)

[109] MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: L.M. LeCam, J. Neyman (Eds.), Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, **1**, pp. 281-297 (1996)

[110] Megalooikonomou, V., Li G., Wang Q.: A dimensionality reduction technique for efficient similarity analysis of time series database. In: Proc. 13th CIKM'04, pp. 160-161 (2004)

[111] Megalooikonomou, V,, Qiang Wang1 Guo Li1, Faloutsos, C.: A multiresolution symbolic representation of time series. In: Proc. 21st ICDE'05, pp. 668-679 (2005)

[112] Mingers, J.: An empirical comparison of selection measures for decision-tree induction. Machine Learning, **3** (4), pp. 319-342 (1989)

[113] Moeslund, T.B., Granum, E.: A survey of computer vision-based human motion capture. Computer Vision and Image Understanding, **81**(3), pp. 231-268 (2001)

[114] Montgomery, D.C.: Introduction to statistical quality control. Wiley, New York, (1996)

[115] Morchen, F., Ultsch, A.: Optimizing time series discretization for knowledge discovery. In: Proc. 11th ACM SIGKDD, pp. 660-665 (2005)

[116] Murthy, S.K.: Automatic Construction of decision trees from data: A multidisciplinary survey. Data Mining and Knowledge Discovery, **2**(4), pp. 345-389 (1998)

[117] Nanopoulos, A., Alcock, R., Manolopoulos, Y.: Feature-based classification of time-series data. International Journal of Computer Research, Nova Science, pp. 49-61 (2001)

[118] Nelson, L.S.: The Shewhart control chart – Tests for special causes. Journal of Quality Technology, **16** (4), pp. 237-239 (1984)

[119] Nelson, L.S.: Interpreting Shewhart x-bar control charts. Journal of Quality Technology, **17** (2), pp. 114-116 (1985)

[120] Oppenheim, A.V., Schafer, R.W.: Digital Signal Processing. Prentice Hall, Englewood Cliffs, N.J., (1975)

[121] Otey, M.E., Parthasarathy, S.: A dissimilarity measure for comparing subsets of data: Application to multivariate time series. In: Proc. ICDM Workshop on Temporal Data Mining, (2005)

[122] Papadimitriou, S., Sun J., Faloutsos, C.: Streaming pattern discovery in multiple time series. In: Proc. 31st VLDB'05, pp. 697-708 (2005)

[123] Popivanov, I., Miller, R.: Similarity search over time series data using wavelets. In: Proc. 18th ICDE'02, pp. 212-221 (2002)

[124] Pugh, G.A.: Synthetic Neural Networks for Process Control. Computers & Industrial Engineering, **17**(1-4), pp. 24-26 (1989)

[125] Quinlan J.R.: Induction of Decision Trees. Machine Learning, **1**(1), pp. 81-106 (1986)

[126] Rabiner, L. Juang, B.-H.: Fundamentals of Speech Recognition. Chapter 4, Prentice Hall, U.S.A., (1993)

[127] Rafiei, D., Mendelzon, A.: Efficient retrieval of similar time sequences using DFT. In: Proc. 5th FODO, pp. 249-257 (1998)

[128] Ratanamahatana, C.A., Lin, J., Gunopulos, D., Keogh, E., Vlachos, M., Das, G.: Mining Time Series Data. In: Data Mining and Knowledge Discovery Handbook, Springer US, pp. 1069-1103 (2005)

[129] Ripley, B.D.: Pattern Recognition and Neural Networks. Canbridge University Press (1996)

[130] Roverso, D.: Plant diagnostics by transient classification: The Aladdin approach. International Journal of Intelligent Systems, **17** (8), pp. 767-790 (2002)

[131] Sakoe, H. Chiba, S.: Dynamic programming algorithm optimization for spoken word recognition. IEEE Transactions Accoustics, Speech and Signal Processing, **26**, pp. 43-49 (1978)

[132] Sakurai, Y., Yoshikawa, M., Faloutsos, C.: FTW: Fast search under the time warping distance. In: Proc. 24th Symp. PODS'05, pp. 326-337 (2005)

[133] Sayal, M.: Detecting time correlations in time-series data streams. Technical Report, Intelligent Enterprise Technologies Laboratory HP Laboratories Palo Alto HPL-2004-103, (2004)

[134] Sellis, T., Roussopoulos, N., Faloutsos, C.: The R+-Tree: A dynamic index for multidimensional objects. In: Proc. VLDB, pp. 507-518 (1987)

[135] Shahabi, C., Xiaoming, T., Wugang, Z., TSA-Tree: A wavelet-based approach to improve the efficiency of multi-level surprise and trend queries on time-series data. In: Proc. 12th SSDBM'00, pp. 55-68 (2000)

[136] Shatkay, H.: Approximate queries and representations for large data sequences. Technical Report cs-95-03, Department of Computer Science, Brown University, (1995)

[137] Shewhart, W.A.: Economic control of quality of manufactured product. Van Nostrand: reprinted American Society for Quality Control, 1980, Milwaukee, WI (1931)

[138] Singhal, A., Seborg, D.E.: Clustering multivariate time-series data. Journal of Chemometrics, **19** (8), pp. 427-438 (2005)

[139] Swets, D., Weng, J.: Hierarchical discriminant analysis for image retrieval. IEEE Trans. Pattern Analysis and Machine Intelligence, **21**, pp. 386–401 (1999)

[140] Swift, J.A., Mize, J.H.: Out-of-control pattern recognition and analysis for quality control charts using LISP-based systems. Computers & Industrial Engineering, **28** (1), pp. 81-91 (1995)

[141] Tanaka, Y., Iwamoto, K., Uehara, K.: Discovery of time series motif from multi-dimensional data based on MDL principle. Machine Learning, **58** (2-3) pp. 269-300 (2005)

[142] Tanawongsuwan, R., Bobick, A.: Performance analysis of time-distance gait parameters under different speeds. In: Proc. 4th Int. Conf. AVBPA, pp. 715-724 (2003)

[143] Thomasian, A., Castelli, V., Li, C.S.: Clustering and singular value decomposition for approximate indexing in highdimensional spaces. In: Proc. CIKM'98, pp. 267-272 (1998)

[144] Toshniwal, D., Ramesh, C.J.: Finding similarity in time series data by method of time weighted moments. In: Proc. 16th ADC'05, pp. 155-164 (2005)

[145] Tsagalidis E., Karamitopoulos L., Evangelidis G., Dervos D., "Extraction of the Convective Day Category Index Using Data Mining Techniques", 3rd IEEE International Workshop on Intelligent Data Acquisition and Advanced Computer Systems: Technology and Applications (IDAACS'2005), Sofia, Bulgaria, Sep 2005.

[146] Valera, M., Velastin, S.A.: Intelligent distributed surveillance systems: A review. In: Proc. IEE Vision Image and Signal Processing, **152** (2) pp. 192-204 (2005)

[147] Vlachos, M., Kollios, G., Gunopoulos, D.: Discovering similar multidimensional trajectories. In: Proc. 18th ICDE, pp. 673-684 (2002)

[148] Vlachos, M., Hadjieleftheriou, M., Gunopoulos, D., Keogh, E.: Indexing multidimensional time-series with support for multiple disatance measures. In: Proc. 9th ACM SIGKDD, pp. 216–225 (2003)

[149] Vlachos, M., Gunopulos, D., Das, G.: Rotation invariant distance measures for trajectories. In: Proc. 10th ACM SIGKDD, pp. 707-712 (2004)

[150] Vlachos, M., Meek, C., Vagena, Z., Gunopoulos, D.: Identifying similarities, periodicities and bursts for online search queries. In: Proc. ACM SIGMOD, pp. 131-142 (2004)

[151] Vlachos, M., Hadjieleftheriou, M., Gunopoulos, D., Keogh, E,: Indexing multidimensional time-series. The VLDB Journal, **15** (1) pp. 1-20 (2006)

[152] Vullings, H.J, Verhaegen, L.M, M.H.G. Verbruggen, H.-B.: Automated ECG Segmentation with Dynamic Time Warping. In: Proc. 20th Annual International Conference of the IEEE in Engineering in Medicine and Biology Society, **1**, pp. 163-166 (1998)

[153] Wall, M., Rechtsteiner, E., Luis, A., Rocha, M.: Singular value decomposition and principal component analysis. In: A practical approach to microarray data analysis. Springer US, pp. 91-109 (2003)

[154] Wang, C., Wang, X.S.: Supporting content-based searches on time series via approximation. In: Proc. 12th Int'l Conference on Scientific and Statistical Database Management, pp 69-81 (2000)

[155] Weber, R., Schek, H., Blott, S.: A quantitative analysis and performance study for similarity search methods in high-dimensional spaces. In: Proc. 24th VLDB, pp. 194-205 (1998)

[156] Western Electric,: Statistical Quality Control Handbook. Princeton, NJ: AT&T, (1956)

[157] White, D.A., Jain, R.: Similarity indexing with the SS-Tree. In: Proc. 12th ICDE, pp. 516-523 (1996)

[158] Wu, Y.L., Agrawal, D., Abbadi, A.E.: A comparison of DFT and DWT based similarity search in time series databases. In: Proc. CIKM'00, pp. 488-495 (2000)

[159] Yang, K., Shahabi, C.: A PCA-based similarity measure for multivariate time series. In: Proc. 2nd ACM MMDB, pp. 65-74 (2004)

[160] Yazdani, N, Ozsoyogin, Z.M.: Sequence matching of images. In: Proc. 8th International Conference on Scientific and Statistical Databases, (1996)

[161] Yi, B.K., Faloutsos, C.: Fast time sequence indexing for arbitrary Lp Norms. In: Proc. 26th VLDB'00, pp. 385-394 (2000)

[162] Yi, B.K., Jagadish, H., Faloutsos, C.: Efficient retrieval of similar time sequences under time warping. In: Proc. ICDE'98, pp. 23-27 (1998)

[163] Zezula, P., Amato, G., Dohnal, V., Batko, M.: Similarity Search: The Metric Approach. Springer (2006)

[164] Zorriassatine, F. Tannock, J.D.T.: A review of neural networks for statistical process control. Journal of Intelligent Manufacturing, **9** (3), pp. 209-224 (1998)

APPENDICES

APPENDIX A. MATLAB codes– univariate time series

```
% Discrete Fourier Transform Representation
% x: m*(n+1) array of m time series of length n. The first column contains
the class label of each time series
% coefficients: the number of Fourier coefficients to be retained. Note that
the dimensionality of the resulting time series is 2*coefficients

function [y] = dft(x,coefficients)
xf = (1/sqrt(size(x,2)-1)) * fft(x(:,2:end),[],2);
% ignore the first coefficient (=0)
xfc = xf(:,2:coefficients+1);
y = [x(:,1) real(xfc) imag(xfc)];
end
```

Figure A1. Discrete Fourier Transform

```
% Piecewise Aggregate Approximation
% x: m*(n+1) array of m time series of length n. The first column contains
the class label of each time series
% seg: the number of segments that will be generated
% When (n/seg) is not an integer, the time series is augmented by adding
zeros and then represented by PAA

function [y] = paa(x,seg)
% create a vector of the class labels
x_label = x(:,1);
% pull out the class labels from the dataset
x(:,1) = [];
% find the number of points that will consist each segment
points = size(x,2)/seg;
% if (n/segments) is an integer
if mod(size(x,2),seg) == 0
    for j= 1:size(x,1)
        % PAA representation
        y(j,:) = mean(reshape(x(j,:),points,seg));
    end
% if (n/seg) is not an integer
else
    for j= 1:size(x,1)
        % create a temporarily
        a = x(j,:);
        % find the number of zeros to be added
        pad_points = (ceil(points) * seg) - length(a);
        % create the augmented time series
        a = [a zeros(1,pad_points)];
        % PAA representation
        y(j,:) = mean(reshape(a,ceil(points),seg));
    end
end
% add the class labels to the tranformed dataset
y = [x_label y];
end
```

Figure A2. Piecewise Aggregate Approximation

```
% Feature Based Representation
% x: m*(n+1) array of m time series of length n. The first column contains
the class label of each time series
% The parameter "lag" determines the time position of the values to be
subtracted in order to compute the second order features
% The code assumes that the values of the time series are normalizized

function [y] = fb(x,lag)
for i=1:size(x,1)
    y(i,1:7) = [x(i,1) ,
                skewness(x(i,2:size(x,2))),kurtosis(x(i,2:size(x,2)))
                ,mean(diff(x(i,2:size(x,2)),lag)),std(diff(x(i,2:size(x,2)),l
                ag)),skewness(diff(x(i,2:size(x,2)),lag)),kurtosis(diff(x(i,2
                :size(x,2)),lag))];
end
end
```

Figure A3. Feature-based representation

```
% Piecewise Aggregate Approximation with Dispersion
% x: m*(n+1) array of m time series of length n. The first column contains
the class labels of each time series
% The parameter "segments" determines the number of sections that a time
series is segmented into.
% When (n/segments) is not an integer, the time series is augmented by
adding zeros and then represented by DPAA

function [y] = dpaa(x,segments)
% create a vector of the class labels
x_label = x(:,1);
% pull out the class labels from the dataset
x(:,1) = [];
% find the number of points that consist each segment
points = size(x,2)/segments;
% if (n/segments) is an integer
if mod(size(x,2),segments) == 0
    for j= 1:size(x,1)
        % calculate the means of the derived segments
        y(j,1:segments) = mean(reshape(x(j,:),points,segments));
        % calculate the standard deviations of the derived segments
        y(j,segments+1:2*segments) = std(reshape(x(j,:),points,segments));
    end
% if (n/segments) is not an integer
else
    for j= 1:size(x,1)
        % create a temporary vector
        a = x(j,:);
        % find the number of zeros to be added
        pad_points = (ceil(points) * segments) - length(a);
        % create the augmented time series
        a = [a zeros(1,pad_points)];
        % calculate the means of the derived segments
        y(j,1:segments) = mean(reshape(a,ceil(points),segments));
        % calculate the standard deviations of the derived segments
        y(j,segments+1:2*segments) = std(reshape(a,ceil(points),segments));
    end
end
% add the class labels to the tranformed dataset
y = [x_label y];
end
```

Figure A4. Dispersion-based Piecewise Aggregate Approximation

```
% ONE NEAREST NEIGHBOR CLASSIFICATION
% train: m1*(n+1) array of m1 time series of length n. The first column
contains the class labels of each time series
% test: m2*(n+1) array of m2 time series of length n. The first column
contains the class labels of each time series

function [error_rate] = nn1(train,test)

% Pull out the class labels.
train_lab = train(:,1);
% Remove class labels from training set.
train(:,1) = [];
% Pull out the class labels.
test_lab = test(:,1);
% Remove class labels from testing set.
test(:,1) = [];
%%%%%%%%%%%%%% LOOP OVER EVERY INSTANCE IN THE TEST SET %%%%%
correct = 0;
for i = 1 : length(test_lab)
    object = test(i,:);
    object_class = test_lab(i);
    pred_class = classify(train,train_lab,object);
    if pred_class == object_class
        correct = correct + 1;
    end;
end;
error_rate = (length(test_lab)-correct )/length(test_lab);

end
%%%%%%%%%%%%%% END %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%% CLASSIFICATION ALGORITHM %%%%%%%%%%%%%%%%%%%%%%
function pred_class = classify(train, train_lab, obj)

best_so_far = inf;
% Loop over every instance in the train set
for i = 1 : length(train_lab)
    % Euclidean distance
    distance = sqrt(sum((train(i,:) - obj).^2));
    if distance < best_so_far
        pred_class = train_lab(i);
        best_so_far = distance;
    end
end

end
%%%%%%%%%%%%%% END CLASSIFICATION %%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Figure A5. 1-NN classification (univariate time series)

```
% ONE NEAREST NEIGHBOR CLASSIFICATION - DPAA REPRESENTATION
% train: m1*(n+1) array of m1 time series of length n. The first column
contains the class labels of each time series
% test: m2*(n+1) array of m2 time series of length n. The first column
contains the class labels of each time series
% The parameter "seg" determines the number of sections that a time series
is segmented into. Note that the dimensionality of the resulting series is
2*segments.
% The parameter "w" is the weight assigned to the means while (1-w) is  the
weight assigned to standard deviations (w is assumed to be within [0,1]).

function [error_rate] = nn1_dpaa(train,test,seg,w)
%%%%%%%%%%%%%%% DPAA REPRESENTATION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
train = dpaa(train,seg);
test = dpaa(test,seg);
%%%%%%%%%%%%%%% END REPRESENTATION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Pull out the class labels.
train_lab = train(:,1);
% Remove class labels from training set.
train(:,1) = [];
% Pull out the class labels.
test_lab = test(:,1);
% Remove class labels from testing set.
test(:,1) = [];
%%%%%%%%%%%%%%% LOOP OVER EVERY INSTANCE IN THE TEST SET %%%%%%%%%%%%
correct = 0;
for i = 1 : length(test_lab)
    object = test(i,:);
    object_class = test_lab(i);
% Add the following 4 lines and make the appropriate change to the next
line, if the object needs to be standardized along with the train set
% train_test = [object ; train];        %augment train set by the
% train_test_z = zscore(train_test);    % standardize variables (columns)
% object = train_test_z(1,:);           % standardized object
% train_z = train_test_z(2:end,:);      % standardized train set
    % change "train" to "train_z"
    pred_class = classify(train, train_lab, object, seg, w);
    if pred_class == object_class
        correct = correct + 1;
    end;
end;
error_rate = (length(test_lab)-correct )/length(test_lab);

end
%%%%%%%%%%%%%%% END %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%% CLASSIFICATION ALGORITHM %%%%%%%%%%%%%%%%%%%%%%%%%
function pred_class = classify(train, train_lab, obj, seg, w)
best_so_far = inf;
% Loop over every instance in the train set
for i = 1 : length(train_lab)
    distance = sqrt((w/seg)*(sum((train(i,1:seg)-obj(1:seg)).^2))+((1-w)
               /seg)*(sum((train(i,seg+1:end) - obj(seg+1:end)).^2)));
    if distance < best_so_far
        pred_class = train_lab(i);
        best_so_far = distance;
    end
end
end
%%%%%%%%%%%%%%% END CLASSIFICATION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Figure A6. 1-NN classification under DPAA representation

```
% ONE NEAREST NEIGHBOR - SVD REPRESENTATION
% train: m1*(n+1) array of m1 time series of length n. The first column
contains the class labels of each time series
% test: m2*(n+1) array of m2 time series of length n. The first column
contains the class labels of each time series
% The parameter "comp" determines the number of components to be retained.

function [error_rate] = nn1_svd(train,test,comp)

% Pull out the class labels.
train_lab = train(:,1);
% Remove class labels from training set.
train(:,1) = [];
 % Pull out the class labels.
test_lab = test(:,1);
% Remove class labels from testing set.
test(:,1) = [];
%%%%%%%%%%%%%%% LOOP OVER EVERY INSTANCE IN THE TEST SET %%%%%%%%%%%%%
correct = 0;
for i = 1 : length(test_lab)

    %%%%%%%%%%%%%%% SVD REPRESENTATION %%%%%%%%%%%%%%%
    % Perform SVD on augmented train dataset
    [U S] = svd([train ; test(i,:)],'econ');
    % Dimensionality reduction
    train_svd = U(:,1:comp) * S(1:comp,1:comp);
    % Test object
    object = train_svd (size(train_svd,1),:);
    % Test object's class
    object_class = test_lab(i);
    % Train dataset
    train_svd = train_svd (1:size(train_svd,1)-1,:);
    %%%%%%%%%%%%%%% END PRE-PROCESSING %%%%%%%%%%%%%%%

    pred_class = classify(train_svd, train_lab, object);
    if pred_class == object_class
        correct = correct + 1;
    end;
end;
error_rate = (length(test_lab)-correct )/length(test_lab);
end
%%%%%%%%%%%%%%% END %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%% CLASSIFICATION ALGORITHM %%%%%%%%%%%%%%%%%%%%%%
function pred_class = classify(train_svd, train_lab, obj)
best_so_far = inf;
% Loop over every instance in the train set
for i = 1 : length(train_lab)
    distance = sqrt(sum((train_svd(i,:) - obj).^2));
    if distance < best_so_far
        pred_class = train_lab(i);
        best_so_far = distance;
    end
end
end
%%%%%%%%%%%%%%% END CLASSIFICATION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Figure A7. 1-NN classification under SVD representation

```
% kNN search
% k is set equal to the actual size of each class

function [mean_error_rate] = knn(dataset)

% find all distinct classes and their corresponding frequencies
size_of_classes = tabulate(dataset(:,1));
% the (i,j) element is the Euclidean distance between the ith and jth time
series
all_distances = squareform(pdist(dataset(:,2:end)));
% Loop over every query in the dataset
for i=1:size(dataset,1)
    % Select the query
    query = dataset(i,:);
    % Obtain query's class
    query_label = query(1);
    % Find total number of instances of the same class with the query
    size_query_label = size_of_classes (find(size_of_classes(:,1)==
                        query_label),2);
    % exclude query from the count
    size_query_label = size_query_label - 1;
    % distances of the query from each series of the database along with the
      corresponding labels
    distances = [dataset(:,1) all_distances(i,:)'];
    % Sort distances in ascending order
    distances = sortrows(distances,2);
    % Keep the k closest instances (labels) - exclude the first instance
    % because it is the distance of the query from itself
    distances = distances(2:1+size_query_label,1);
    correct = 0;
    for k = 1:length(distances)
        if distances(k) == query_label
            correct = correct + 1;
        end
    end
    % error rate for the specific query
    error_rate(i) = 1 - (correct/length(distances));
end
% mean error rate across all queries
mean_error_rate = mean(error_rate);

end
```

Figure A8. k-NN similarity search (univariate time series)

```
% kNN search - It requires data to be transformed by DPAA
% k is set equal to the actual size of each class
% w is the weight for the distances among means, whereas (1-w) is the weight
for the distances among standard deviations
% the distance function is the weighted distance of means and standard
deviations

function [mean_error_rate] = knn_dpaa(dataset,w)
% Add the following line, if the query needs to be standardized along with
the dataset
% dataset = [dataset(:,1) zscore(dataset(:,2:end))];

% find all distinct classes and their corresponding frequencies
size_of_classes = tabulate(dataset(:,1));
% half the number of elements of the transformed series
c = ( size(dataset,2) - 1 ) / 2;
% table of means
m = dataset( : , 2:c+1 )   ;
% table of std's
s = dataset( : , c+2:end )   ;
distances_of_means = pdist(m);
distances_of_stds =  pdist(s);
all_distances = sqrt((w* distances_of_means.^2 + (1-
w)*distances_of_stds.^2));
% the (i,j) element is the distance between the ith and jth time series
all_distances = squareform(all_distances);
% Loop over every query in the dataset
for i=1:size(dataset,1)
    % Select the query
    query = dataset(i,:);
    % Obtain query's class
    query_label = query(1);
    % Find total number of instances of the same class with the query
    size_query_label = size_of_classes(find(size_of_classes(:,1)==
                       query_label),2);
    % exclude query from the count
    size_query_label = size_query_label - 1;
    % distances of the query from each series of the database along with the
      corresponding labels
    distances = [dataset(:,1) all_distances(i,:)'];
    % Sort distances in ascending order
    distances = sortrows(distances,2);
    % Keep the k closest instances (labels) - exclude the first instance
      because it is the distance of the query from itself
    distances = distances(2:1+size_query_label,1);
    correct = 0;
    for k = 1:length(distances)
        if distances(k) == query_label
            correct = correct + 1;
        end
    end
    % error rate for the specific query
    error_rate(i) = 1 - (correct/length(distances));
end
% mean error rate across all queries
mean_error_rate = mean(error_rate);
end
```

Figure A9. k-NN similarity search under DPAA representation

APPENDIX B. MATLAB codes – Cluster-based 1-NN search

```
% The function "nn1_clurep_all" applies the algorithm k-means in order to
partition
% the dataset into clusters, and then similarity search proceeds at each
cluster sequentially.
% The algorithm searches the whole cluster, when it reaches it.
% k-means: k is the number of clusters.


function [error, average_space , bsf] = nn1_clurep_all(dataset,k)

correct = 0;
for j = 1 : size(dataset,1)
    query = dataset(j,:);
    % Pull-out label from the query
    query_label = query(1);
    % Remove label from the query
    query(1) = [];
    train = dataset;
    % Remove the query from the dataset
    train(j,:) = [];
    % Pull-out labels from the training set
    train_labels = train(:,1);
    % Remove labels from the training set
    train(:,1) = [];
    [cluster_id centroids sumd distances] = kmeans(train,k)
    distances = sqrt(distances);
    % insert 1) cluster_id, 2) distances fron centroids and 3) class_labels
to
      the training set
    temp = [cluster_id distances train_labels train];
    % sort training set with respect to cluster_id
    temp = sortrows(temp,1);
    for i = 1 : k
       first = find(temp(:,1) == i , 1 , 'first');
       last  = find(temp(:,1) == i , 1 , 'last');
       cluster(i).id       = i;
       cluster(i).dist     = temp (first:last , 1+i);
       cluster(i).class    = temp (first:last , 2+k);
       cluster(i).data     = temp (first:last , 3+k:end);
       cluster(i).radius   = max( cluster(i).dist );
       cluster(i).mean     = mean( cluster(i).dist );
       cluster(i).std      = std( cluster(i).dist );
       cluster(i).ub       = cluster(i).mean + 3 * cluster(i).std;
    end

   %%%% calculate distances of the query from each centroid and sort them%%%
   for i=1:k
        dqc(i,1) = i;
        dqc(i,2) = sqrt(sum((query-centroids(i,:)).^2));
   end
   dqc = sortrows(dqc,2);
   primary_cluster = dqc(1,1);
   % sort by cluster id (1,2,3,...k)
   dqc = sortrows(dqc,1);
   %%%%%%%%%%%%%%%%%%%%%%%%
end%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

(continues)

```
(continued)

    %%%% calculate distances of the query from each cluster and sort them
%%%%
    for i=1:k
        dqcl(i,1) = i;
        dqcl(i,2) = max( 0 , dqc(i,2)-cluster(i).radius );
    end
    dqcl(primary_cluster,:) = [];
    % sort by distance from cluster
    dqcl = sortrows(dqcl,2);
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%% end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%%%%%% 1-NN Classification for the Primary Cluster %%%%%%%%%%%%
    vis = 0;
    best_so_far = inf;
    for i =  1 : size (cluster(primary_cluster).data , 1)
        vis = vis +1;
        d = sqrt( sum ( (query - cluster(primary_cluster).data(i,:) ).^2));
        if d < best_so_far
            best_so_far = d;
            predicted_class = cluster(primary_cluster).class(i);
        end
    end
    %%%%%%%% 1-NN Classification for the Rest Clusters %%%%%%%%%%%%
    for s = 1 : k-1
        if dqcl(s,2) < best_so_far
            for i = 1 : size (cluster(dqcl(s,1)).data , 1)
                vis = vis +1;
                d = sqrt( sum ((query - cluster(dqcl(s,1)).data(i,:)).^2));
                if d < best_so_far
                    best_so_far = d;
                    predicted_class = cluster(dqcl(s,1)).class(i);
                end
            end
        end
    end
    if predicted_class == query_label
        correct = correct+1;
    end
    visited(j) = vis * 100 / size (train , 1);
    bsf(j) = best_so_far;
end
accuracy = correct / size(dataset,1);
error = (1 - accuracy);
average_space = mean (visited) ;

end
```

Figure B1. Cluster-based 1-NN similarity search (whole cluster search)

```
% The function "nn1_clurep" applies the algorithm k-means in order to
partition
% the dataset into clusters, and then similarity search proceeds at each
% cluster sequentially.
% The algorithm reduces the search space for each cluster.
% k-means: k is the number of clusters

function [error, average_space ] = nn1_clurep(dataset,k)

correct = 0;
for j = 1 : size(dataset,1)
    query = dataset(j,:);
    % Pull-out label from the query
    query_label = query(1);
    % Remove label from the query
    query(1) = [];
    train = dataset;
    % Remove the query from the dataset
    train(j,:) = [];
    % Pull-out labels from the training set
    train_labels = train(:,1);
    % Remove labels from the training set
    train(:,1) = [];
    [cluster_id centroids sumd distances] = kmeans(train,k);
    distances = sqrt(distances);
    % insert 1) cluster_id, 2) distances fron centroids and 3) class_labels
to
      the training set
    temp = [cluster_id distances train_labels train];
    % sort training set with respect to cluster_id
    temp = sortrows(temp,1);
    for i = 1 : k
       first = find(temp(:,1) == i , 1 , 'first');
       last  = find(temp(:,1) == i , 1 , 'last');
       cluster(i).id       = i;
       cluster(i).dist     = temp (first:last , 1+i);
       cluster(i).class    = temp (first:last , 2+k);
       cluster(i).data     = temp (first:last , 3+k:end);
       cluster(i).radius   = max( cluster(i).dist );
       cluster(i).mean     = mean( cluster(i).dist );
       cluster(i).std      = std( cluster(i).dist );
       cluster(i).ub       = cluster(i).mean + 3 * cluster(i).std;
    end

    %%%%%%%% calculate distances of the query from each centroid and sort
them%
    for i=1:k
        dqc(i,1) = i;
        dqc(i,2) = sqrt(sum((query-centroids(i,:)).^2));
    end
    dqc = sortrows(dqc,2);
    primary_cluster = dqc(1,1);
    % sort by cluster id (1,2,3,...k)
    dqc = sortrows(dqc,1);
    %%%%%%% end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%% calculate distances of the query from each cluster and sort them
%%%%%%
    for i=1:k
        dqcl(i,1) = i;
        dqcl(i,2) = max( 0 , dqc(i,2)-cluster(i).radius );
    end
```

(continues)

```
(continued)

    dqcl(primary_cluster,:) = [];
    % sort by distance from cluster
    dqcl = sortrows(dqcl,2);
    %%%%% %%%%%%%%%%%%%%%%%%%%%%%% end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%%%%%% 1-NN Classification for the Primary Cluster %%%%%%%%%%%%
    vis = 0;
    best_so_far = inf;
    for i =  1 : size (cluster(primary_cluster).data , 1)
        vis = vis +1;
        d = sqrt( sum ((query - cluster(primary_cluster).data(i,:)).^2));
        if d < best_so_far
            best_so_far = d;
            predicted_class = cluster(primary_cluster).class(i);
        end
    end
    %%%%%%%% 1-NN Classification for the Rest Clusters %%%%%%%%%%%%
    for s = 1 : k-1
        if dqcl(s,2) < best_so_far
            if dqcl(s,2) > 0
                x = best_so_far - dqcl(s,2);
                upperb = max (cluster(dqcl(s,1)).radius - x , 0);
            else
                x = best_so_far - dqcl(s,2);
                upperb = max (  dqc(dqcl(s,1),2) - x , 0);
            end;
            for i = 1 : size (cluster(dqcl(s,1)).data , 1)
                if cluster(dqcl(s,1)).dist(i) >= upperb
                    vis = vis +1;
                    d = sqrt(sum((query -
cluster(dqcl(s,1)).data(i,:)).^2));
                    if d < best_so_far
                        best_so_far = d;
                        predicted_class = cluster(dqcl(s,1)).class(i);
                    end
                end
            end
        end
    end
    if predicted_class == query_label
        correct = correct+1;
    end
    visited(j) = vis * 100 / size (train , 1)  ;
end
accuracy = correct / size(dataset,1);
error = (1 - accuracy)*100;
average_space = mean (visited);

end
```

Figure B2. Cluster-based 1-NN similarity search (partial cluster search)

APPENDIX C. MATLAB codes – Multivariate time series

```
% The function shift subtracts from each element of a matrix (data) the mean
of the corresponding column
% db (dataset) should be a structured array comprised by at least 1 field:
data

function  new_db = shift(db)

new_db = db;
for j = 1:size(db,2)
    new_db(j).data = db(j).data -
                     repmat(mean(db(j).data),size(db(j).data,1),1);
end
end
```

Figure C1. Center time series about their corresponding means

```
% ONE NEAREST NEIGHBOR CLASSIFICATION
% train and test should be structured arrays comprised by at least 2 fields:
data & class

function [error_rate] = nn1_ed(train,test)
%Use [] if class is numeric
train_lab = {train.class};
% Use [] if class is numeric
test_lab = {test.class};

%%%%%%%%%%%%%%% LOOP OVER EVERY INSTANCE IN THE TEST SET %%%%%%%%%%%%%
correct = 0;
for i = 1 : length(test_lab)
    object = [test(i).data];
    object_class = test_lab(i);
    pred_class = classify(train,train_lab, object);
    % write - (pred_class == object_class) - if class is string
    if strcmp(pred_class,object_class)
       correct = correct + 1;
   end;
end;
error_rate = (length(test_lab)-correct )/length(test_lab);
%%%%%%%%%%%%%%%%%% END %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%% CLASSIFICATION ALGORITHM %%%%%%%%%%%%%%%%%%%%%%
function pred_class = classify(train, train_lab, obj)

best_so_far = inf;
% Loop over every instance in the train set
for i = 1 : length(train_lab)
    % Euclidean distance
    distance = sqrt(sum(sum(([train(i).data] - obj).^2,2)));
    % change to: distance > best_so_far, in case the measure represents
similarity
    if distance < best_so_far
         pred_class = train_lab(i);
         best_so_far = distance;
     end
end;
%%%%%%%%%%%%%%% END CLASSIFICATION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Figure C2. 1-NN classification - Euclidean Distance (multivariate time series)

```
% ONE NEAREST NEIGHBOR CLASSIFICATION
% train and test should be structured arrays comprised by at least 2 fields:
data & class

function [error_rate] = nn1_eros(train,test)
% PCA representation
pca_train = train;
pca_train = rmfield(pca_train,'data');
for k = 1:size(train,2)
    data = [train(k).data];
    [coef scores latent] = princomp(data);
    pca_train(k).data = [coef];
    w(:,k) = latent;
end
% the weight vector may be computed on raw or normalized eigenvalues
% if it is desired to compute w from normalized eigenvalues, then add the
following line: % w = w ./ repmat(sum(w),size(w,1),1)
%using "mean" as an aggregating function of raw eigenvalues
w = mean(w,2)/sum(mean(w,2));
pca_test = test;
pca_test = rmfield(pca_test,'data');
for k = 1:size(test,2)
    data = [test(k).data];
    [coef scores latent] = princomp(data);
    pca_test(k).data = [coef];
end
train = pca_train;
test = pca_test;
%use {} if class is numeric
train_lab = {train.class};
%use {} if class is numeric
test_lab = {test.class};
%%%%%%%%%%%%%% LOOP OVER EVERY INSTANCE IN THE TEST SET %%%%%%%%%%%%
correct = 0;
for i = 1 : length(test_lab)
    object = [test(i).data];
    object_class = test_lab(i);
    pred_class = classify(train,train_lab, object,w);
    % write - (pred_class == obj_class) - if class is string
    if strcmp(pred_class,object_class)
       correct = correct + 1;
   end;
end;
error_rate = (length(test_lab)-correct )/length(test_lab);

function pred_class = classify(train, train_lab, obj, w)
%Similarity Measure
best_so_far = 0;
% Loop over every instance in the train set
 for i = 1 : length(train_lab)
     distance = sum(abs(diag([train(i).data]'* obj)).*w);
     % change to: distance< best_so_far, in case the measure represents
       distance
     if distance > best_so_far
         pred_class = train_lab(i);
         best_so_far = distance;
     end
end;
%%%%%%%%%%%%%% END CLASSIFICATION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Figure C3. 1-NN classification – Eros similarity measure (Yang & Shahabi 2004)

```
% ONE NEAREST NEIGHBOR CLASSIFICATION
% train and test should be structured arrays comprised by at least 2 fields:
data & class

function [error_rate] = nn1_kwas(train,test,components)
% SVD representation
svd_train = train;
svd_train = rmfield(svd_train,'data');
for k = 1:size(train,2)
    data = [train(k).data];
    [UTRAIN STRAIN VTRAIN] = svd(data'*data);
    STRAIN = diag(STRAIN);
    svd_train(k).data = VTRAIN(:,1:components);
    svd_train(k).lat = STRAIN(1:components)/sum(STRAIN);
end
svd_test = test;
svd_test = rmfield(svd_test,'data');
for k = 1:size(test,2)
    data = [test(k).data];
    [UTEST STEST VTEST] = svd(data'*data);
    STEST = diag(STEST);
    svd_test(k).data = VTEST(:,1:components);
    svd_test(k).lat = STEST(1:components)/sum(STEST);
end
train = svd_train;
test = svd_test;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%use {} if class is numeric
train_lab = {train.class};
 %use {} if class is numeric
test_lab = {test.class};
%%%%%%%%%%%%%%% LOOP OVER EVERY INSTANCE IN THE TEST SET %%%%%%%%%%%%%
correct = 0;
for i = 1 : length(test_lab)
    object = [test(i).data];
    object_class = test_lab(i);
    object_lat = test(i).lat;
    pred_class = classify(train,train_lab, object,object_lat);
    % write - (pred_class == object_class) - if class is string
    if strcmp(pred_class,object_class)
       correct = correct + 1;
   end;
end;
error_rate = (length(test_lab)-correct )/length(test_lab);
%%%%%%%%%%%%%%%%%% END %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function pred_class = classify(train, train_lab, obj, object_lat)
%Similarity Measure
best_so_far = 0;
% Loop over every instance in the train set
 for i = 1 : length(train_lab)
     distance =
(1/2)*abs(diag(train(i).data'*obj))'*(train(i).lat+object_lat);
     % change to: distance< best_so_far, in case the measure represents
       distance
     if distance > best_so_far
         pred_class = train_lab(i);
         best_so_far = distance;
     end
end;
%%%%%%%%%%%%%%% END CLASSIFICATION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Figure C4. 1-NN classification – kWAS similarity measure (Li & Prabhakaran 2005)

```
% ONE NEAREST NEIGHBOR CLASSIFICATION
% train and test should be structured arrays comprised by at least 2 fields:
data & class

function [error_rate] = nn1_sim(train,test,components)
% PCA representation
pca_train = train;
pca_train = rmfield(pca_train,'data');
for k = 1:size(train,2)
    data = [train(k).data];
    [coef scores latent] = princomp(data);
    pca_train(k).data = [coef(:,1:components)];
end
pca_test = test;
pca_test = rmfield(pca_test,'data');

for k = 1:size(test,2)
    data = [test(k).data];
    [UTEST STEST VTEST] = svd(data'*data);
    [coef scores latent] = princomp(data);
    pca_test(k).data = [coef(:,1:components)];
end
train = pca_train;
test = pca_test;
%use {} if class is numeric
train_lab = {train.class};
%use {} if class is numeric
test_lab = {test.class};
%%%%%%%%%%%%%% LOOP OVER EVERY INSTANCE IN THE TEST SET %%%%%%%%%%%%%
correct = 0;
for i = 1 : length(test_lab)
    object = [test(i).data];
    object_class = test_lab(i);
    pred_class = classify(train,train_lab, object);
    % write - (pred_class == object_class) - if class is string
    if strcmp(pred_class,object_class)
       correct = correct + 1;
   end;
end;
error_rate = (length(test_lab)-correct )/length(test_lab);
%%%%%%%%%%%%%%%%%% END %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function pred_class = classify(train, train_lab, obj)
% Similarity Measure
best_so_far = 0;
% Loop over every instance in the train set
 for i = 1 : length(train_lab)
     distance = trace(train(i).data'* obj * obj' * train(i).data);
     % change to: distance< best_so_far, in case the measure represents
       distance
     if distance > best_so_far
         pred_class = train_lab(i);
         best_so_far = distance;
     end
end;
```

Figure C5. 1-NN classification – PCA similarity measure (Krzanowski 1999)

```
% ONE NEAREST NEIGHBOR CLASSIFICATION
% train and test should be structured arrays comprised by at least 2 fields:
data & class

function [error_rate] = nn1_sim_vars(train,test,components)
% PCA representation
lattrain = [];
pca_train = train;
pca_train = rmfield(pca_train,'data');
for k = 1:size(train,2)
    data = [train(k).data];
    [coef scores latenttrain] = princomp(data);
    pca_train(k).data = [coef(:,1:components)]*
                        sqrt(diag(latenttrain(1:components)));
    lattrain = [lattrain latenttrain(1:components)];
end
lattest = [];
pca_test = test;
pca_test = rmfield(pca_test,'data');

for k = 1:size(test,2)
    data = [test(k).data];
    [coef scores latenttest] = princomp(data);
    pca_test(k).data = [coef(:,1:components)]*
                        sqrt(diag(latenttest(1:components)));
    lattest = [lattest latenttest(1:components)];
end
train = pca_train;
test = pca_test;
%the element (i,j) is the sum of the products of the variances of the
%corresponding components, for train_i and test_j datasets
lat = lattrain'*lattest;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%use {} if class is numeric
train_lab = {train.class};
%use {} if class is numeric
test_lab = {test.class};

%%%%%%%%%%%%%%% LOOP OVER EVERY INSTANCE IN THE TEST SET %%%%%%%%%%%%%
correct = 0;
for i = 1 : length(test_lab)
    object = [test(i).data];
    object_class = test_lab(i);
    %the element (i,j) is the sum of the products of the variances of the
    %corresponding components, for train_i and the specific test_j datasets
    object_lat = lat(:,i);
    pred_class = classify(train,train_lab, object, object_lat);
    % write - (pred_class == object_class) - if class is string
    if strcmp(pred_class,object_class)
       correct = correct + 1;
   end;
end;
error_rate = (length(test_lab)-correct )/length(test_lab);
%%%%%%%%%%%%%%%%%% END %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

(continues)

```
(continued)

function pred_class = classify(train, train_lab, obj, object_lat)
% Similarity Measure
best_so_far = 0;
% Loop over every instance in the train set
 for i = 1 : length(train_lab)
      distance =
(trace(train(i).data'*obj*obj'*train(i).data))/object_lat(i);
      % change to: distance< best_so_far, in case the measure represents
        distance
     if distance > best_so_far
         pred_class = train_lab(i);
         best_so_far = distance;
     end
end;
%%%%%%%%%%%%%% END CLASSIFICATION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Figure C6. 1-NN classification – PCA similarity measure (Johannesmeyer 1999)

```
% ONE NEAREST NEIGHBOR CLASSIFICATION
% train and test should be structured arrays comprised by at least 2 fields:
data & class

function [error_rate] = nn1_spe(train,test,components)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PCA representation

pca_train = train;
pca_train = rmfield(pca_train,'data');
for k = 1:size(train,2)
    data = [train(k).data];
    [coef scores latent] = princomp(data);
    pca_train(k).data = [coef(:,1:components)];
end


train = pca_train;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%use {} if class is numeric
train_lab = {train.class};
%use {} if class is numeric
test_lab = {test.class};
%%%%%%%%%%%%%%% LOOP OVER EVERY INSTANCE IN THE TEST SET %%%%%%%%%%%%%%
correct = 0;
for i = 1 : length(test_lab)
    object = [test(i).data];
    object_class = test_lab(i);
    pred_class = classify(train,train_lab, object);
    % write - (pred_class == obj_class) - if class is string
    if strcmp(pred_class,object_class)
       correct = correct + 1;
   end;
end;
error_rate = (length(test_lab)-correct )/length(test_lab);
%%%%%%%%%%%%%%%%%%% END %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function pred_class = classify(train, train_lab, obj)
% distance Measure
best_so_far = inf;
% Loop over every instance in the train set
 for i = 1 : length(train_lab)
     distance = sum(sum((obj*train(i).data*train(i).data' - obj).^2));
     % change to: distance> best_so_far, in case the measure represents
       similarity
     if distance < best_so_far
         pred_class = train_lab(i);
         best_so_far = distance;
     end
end;
%%%%%%%%%%%%%%% END CLASSIFICATION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Figure C7. 1-NN classification – SPE distance measure (Chapter 7)

```
% PRECISION - RECALL
% The values of Precision are calculated for values of Recall that range
from 0.1 to 1.0 in increments of 0.1 (approximately).
% X (dataset) should be structured array comprised by at least 2 fields:
data & class
% The distance function is the Euclidean distance

function [AP] = pr_ed(X,size_class)

size_class = size_class - 1;

for k=1:10
    % Loop for each query
    for i=1:size(X,2);
        query = X(i);
        dataset = X;
        % Remove the query from the dataset
        dataset(i) = [];
        % Loop for every object in the dataset
        for j=1:size(dataset,2);
            % Record the class of each object in the dataset
            distances(j,1) = dataset(j).class;
            % Calculate the distance of each object in the dataset from the
            % query object (Euclidean distance)
            distances(j,2) = sqrt(sum(sum((dataset(j).data -
                                query(1).data).^2,2)));
        end;
        % sort the distances in increasing order
        distances_sort = sortrows(distances,2);
        retrieved = 0;
        relevant = 0;
        % Loop for a given recall
        while (relevant < round(k*0.1*size_class)) && (retrieved <
               size(dataset,2))
            retrieved = retrieved+1;
            if distances_sort(retrieved,1) == query(1).class
                relevant = relevant+1;
            end
        end
        % Calculate the Precision for a given Recall
        Precision(i) = round(k*0.1*size_class)/retrieved;
    end
    % Calculate the average precision across all queries
    AP(k,1) = mean(Precision);
    % Record the corresponding Recall
    AP(k,2) = round(k*0.1*size_class)/size_class;
end

end
```

Figure C8. Precision-Recall – Euclidean Distance

```
% PRECISION - RECALL: The values of Precision are calculated for values of
Recall that range from 0.1 to 1.0 in increments of 0.1 (approximately).
% X should be structured array comprised by at least 2 fields: data, class
% Eros similarity measure (Yang & Shahabi 2004)

function [AP] = pr_eros(X,size_class)
% PCA representation
pca_X = X;
pca_X = rmfield(pca_X,'data');
for k = 1:size(X,2)
    data = [X(k).data];
    [coef scores latent] = princomp(data);
    pca_X(k).data = [coef];
    w(:,k) = latent;
end
X = pca_X;
% the weight vector may be computed on raw or normalized eigenvalues
% if it is desired to compute w from normalized eigenvalues, then add the
following line: % w = w ./ repmat(sum(w),size(w,1),1)
%using "mean" as an aggregating function of raw eigenvalues
w = mean(w,2)/sum(mean(w,2));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%

size_class = size_class - 1;
for k=1:10
    % Loop for each query
    for i=1:size(X,2);
        query = X(i);
        dataset = X;
        % Remove the query from the dataset
        dataset(i) = [];
        % Loop for every object in the dataset
        for j=1:size(dataset,2);
            % Record the class of each object in the dataset
            distances(j,1) = dataset(j).class;
            distances(j,2) = sum(abs(diag(dataset(j).data'*
                                query(1).data)).*w);
        end;
        % sort the distances (similarities) in decreasing order
        distances_sort = sortrows(distances,-2);
        retrieved = 0;
        relevant = 0;
        % Loop for a given recall
        while (relevant < round(k*0.1*size_class)) && (retrieved <
               size(dataset,2))
            retrieved = retrieved+1;
            if distances_sort(retrieved,1) == query(1).class
                relevant = relevant+1;
            end
        end
        % Calculate the Precision for a given Recall
        Precision(i) = round(k*0.1*size_class)/retrieved;
    end
    % Calculate the average precision across all queries
    AP(k,1) = mean(Precision);
    % Record the corresponding Recall
    AP(k,2) = round(k*0.1*size_class)/size_class;
end
end
```

Figure C9. Precision-Recall – Eros similarity measure (Yang & Shahabi 2004)

```
% PRECISION - RECALL: The values of Precision are calculated for values of
Recall that range from 0.1 to 1.0 in increments of 0.1 (approximately).
% X should be structured array comprised by at least 2 fields: data & class
% size_class: the number of objects within a class. The code assumes that
all classes have the same size.
% components: the number of components to be retained under svd
representation

function [AP] = pr_kwas(X,size_class,components)
% SVD representation
svd_X = X;
svd_X = rmfield(svd_X,'data');

for k = 1:size(X,2)
    data = [X(k).data];
    [UX SX VX] = svd(data'*data);
    SX = diag(SX);
    svd_X(k).data = VX(:,1:components);
    svd_X(k).lat = SX(1:components)/sum(SX);
end
X = svd_X;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%

size_class = size_class - 1;

for k=1:10
    % Loop for each query
    for i=1:size(X,2);
        query = X(i);
        dataset = X;
        % Remove the query from the dataset
        dataset(i) = [];
        % Loop for every object in the dataset
        for j=1:size(dataset,2);
            % Record the class of each object in the dataset
            distances(j,1) = dataset(j).class;
            distances(j,2) = (1/2)*abs(diag(dataset(j).data'*
                              query(1).data))'
                              *(dataset(j).lat+query(1).lat);
        end;
        % sort the distances (similarities) in decreasing order
        distances_sort = sortrows(distances,-2);
        retrieved = 0;
        relevant = 0;
        % Loop for a given recall
        while (relevant < round(k*0.1*size_class)) && (retrieved <
               size(dataset,2))
            retrieved = retrieved+1;
            if distances_sort(retrieved,1) == query(1).class
                relevant = relevant+1;
            end
        end
        % Calculate the Precision for a given Recall
        Precision(i) = round(k*0.1*size_class)/retrieved;
    end
    % Calculate the average precision across all queries
    AP(k,1) = mean(Precision);
    % Record the corresponding Recall
    AP(k,2) = round(k*0.1*size_class)/size_class;
end
end
```
Figure C10. Precision - Recall – kWAS similarity measure (Li & Prabhakaran 2005)

```
% PRECISION - RECALL: The values of Precision are calculated for values of
Recall that range from 0.1 to 1.0 in increments of 0.1 (approximately).
% X should be structured array comprised by at least 2 fields: data & class
% size_class: the number of objects within a class. The code assumes that
all classes have the same size.
% components: the number of components to be retained under pca
representation

function [AP] = pr_sim(X,size_class,components)
% PCA representation
pca_X = X;
pca_X = rmfield(pca_X,'data');

for k = 1:size(X,2)
    data = [X(k).data];
    [coef scores latent] = princomp(data);
    pca_X(k).data = [coef(:,1:components)];

end
X = pca_X;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

size_class = size_class - 1;
for k=1:10
    % Loop for each query
    for i=1:size(X,2);
        query = X(i);
        dataset = X;
        % Remove the query from the dataset
        dataset(i) = [];
        % Loop for every object in the dataset
        for j=1:size(dataset,2);
            % Record the class of each object in the dataset
            distances(j,1) = dataset(j).class;
            distances(j,2) = trace(dataset(j).data' * query(1).data *
                                   query(1).data' * dataset(j).data);
        end;
        % sort the distances (similarities) in decreasing order
        distances_sort = sortrows(distances,-2);
        retrieved = 0;
        relevant = 0;
        % Loop for a given recall
        while (relevant < round(k*0.1*size_class)) && (retrieved <
               size(dataset,2))
            retrieved = retrieved+1;
            if distances_sort(retrieved,1) == query(1).class
                relevant = relevant+1;
            end
        end
        % Calculate the Precision for a given Recall
        Precision(i) = round(k*0.1*size_class)/retrieved;
    end
    % Calculate the average precision across all queries
    AP(k,1) = mean(Precision);
    % Record the corresponding Recall
    AP(k,2) = round(k*0.1*size_class)/size_class;
end

end
```

Figure C11. Precision - Recall – PCA similarity measure (Krzanowski 1999)

```
% PRECISION - RECALL: The values of Precision are calculated for values of
Recall that range from 0.1 to 1.0 in increments of 0.1 (approximately).
% X should be structured array comprised by at least 2 fields: data & class
% size_class: the number of objects within a class. The code assumes that
all classes have the same size.
% components: the number of components to be retained under PCA
representation

function [AP] = pr_sim_vars(X,size_class,components)
% PCA representation
latX = [];
pca_X = X;
pca_X = rmfield(pca_X,'data');
for k = 1:size(X,2)
    data = [X(k).data];
    [coefX scoresX latentX] = princomp(data);
    pca_X(k).data =
[coefX(:,1:components)]*sqrt(diag(latentX(1:components)));
    latX = [latX latentX(1:components)];
end
lat = latX'*latX;
X = pca_X;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
size_class = size_class - 1;
for k=1:10
    % Loop for each query
    for i=1:size(X,2);
        query = X(i);
        dataset = X;
        % Remove the query from the dataset
        dataset(i) = [];
        query_lat = lat(:,i);
        query_lat(i) = [];
        % Loop for every object in the dataset
        for j=1:size(dataset,2);
            % Record the class of each object in the dataset
            distances(j,1) = dataset(j).class;
            distances(j,2) = trace(dataset(j).data'* query(1).data *
                                   query(1).data'*dataset(j).data) /
                                   query_lat(j);
        end;
        % sort the distances (similarities) in decreasing order
        distances_sort = sortrows(distances,-2);
        retrieved = 0;
        relevant = 0;
        % Loop for a given recall
        while (relevant < round(k*0.1*size_class)) && (retrieved <
               size(dataset,2))
            retrieved = retrieved+1;
            if distances_sort(retrieved,1) == query(1).class
                relevant = relevant+1;
            end
        end
        % Calculate the Precision for a given Recall
        Precision(i) = round(k*0.1*size_class)/retrieved;
    end
    % Calculate the average precision across all queries
    AP(k,1) = mean(Precision);
    % Record the corresponding Recall
    AP(k,2) = round(k*0.1*size_class)/size_class;
end
end
```

Figure C12. Precision – Recall – PCA similarity measure (Johannesmeyer 1999)

```
% PRECISION - RECALL: The values of Precision are calculated for values of
Recall that range from 0.1 to 1.0 in increments of 0.1 (approximatelly).
% X (dataset) should be structured array comprised by at least 2 fields:
data & class
% size_class: the number of objects within a class. The code assumes that
all classes have the same size.
% components: the number of components to be retained under pca
representation

function [AP] = pr_spe(X,size_class,components)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PCA representation
pca_X = X;
pca_X = rmfield(pca_X,'data');

for k = 1:size(X,2)
    data = [X(k).data];
    [coef scores latent] = princomp(data);
    pca_X(k).data = [coef(:,1:components)];

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
size_class = size_class - 1;

for k=1:10
    % Loop for each query
    for i=1:size(X,2);
        query = X(i);
        dataset = pca_X;
        % Remove the query from the dataset
        dataset(i) = [];
        % Loop for every object in the dataset
        for j=1:size(dataset,2);
            % Record the class of each object in the dataset
            distances(j,1) = dataset(j).class;
            distances(j,2) = sum(sum((query(1).data * dataset(j).data *
                             dataset(j).data' - query(1).data).^2));
        end;
        % sort the distances in increasing order
        distances_sort = sortrows(distances,2);
        retrieved = 0;
        relevant = 0;
        % Loop for a given recall
        while (relevant < round(k*0.1*size_class)) && (retrieved <
               size(dataset,2))
            retrieved = retrieved+1;
            if distances_sort(retrieved,1) == query(1).class
                relevant = relevant+1;
            end
        end
        % Calculate the Precision for a given Recall
        Precision(i) = round(k*0.1*size_class)/retrieved;
    end
    % Calculate the average precision across all queries
    AP(k,1) = mean(Precision);
    % Record the corresponding Recall
    AP(k,2) = round(k*0.1*size_class)/size_class;
end

end
```

Figure C13. Precision – Recall – SPE distance measure (Chapter 7)

```
% db (dataset) should be a structured array comprised by at least 1 field:
data
% segments: the desired number of segments

function [paa_db,db_zeros] = paa_m(db,segments)
paa_db = rmfield(db,'data');
db_zeros = rmfield(db,'data');
% add zeros to make each data-piece of length that is divided by the number
of segments
for j=1:size(db,2)
    points = ceil(size([db(j).data],1)/segments);
    db_zeros(j).data = [db(j).data ; zeros(abs(size(db(j).data,1)-
                       (segments*points)) , size(db(j).data,2))];
end
% apply PAA on extended dataset
for m = 1:size(db_zeros,2)
    k = 1;
    points_new = size([db_zeros(m).data],1)/segments;
    for i =1 : size([db_zeros(m).data],1)/segments :
size([db_zeros(m).data],1)
        paa_db(m).data(k,:) = mean(db_zeros(m).data(i:i+points_new-1,:),1);
        k = k+1;
    end
end
end
```

Figure C14. Piecewise Aggregate Approximation (multivariate time series)

```
% The function linerp performs linear interpolation on a time series to
change its length
% The function interp1 does not guarantee that the new time series will have
the desired length
% The following code enforces the desired length
% db (dataset) should be a structured array comprised by at least 1 field:
data

function  new_db = linerp(db,new_length)

new_db = rmfield(db,'data');
for j = 1:size(db,2)
    step = size(db(j).data,1)/new_length;
    a = [1:step:size(db(j).data,1)];
    if (length(a) < new_length)
        while (length(a) < new_length)
            step = step-0.0001;
            a = [1:step:size(db(j).data,1)];
        end
    else
        while (length(a) > new_length)
            step = step+0.0001;
            a = [1:step:size(db(j).data,1)];
        end
    end
    for i=1:size(db(j).data,2)
        new_db(j).data(:,i) = interp1(db(j).data(:,i),a)';
    end
end
end
```

Figure C15. Linear interpolation for transforming a time series to a given length

APPENDIX D. FEATURE-BASED REPRESENTATION RESULTS

Table D1. 1-NN classification error rates for feature-based representation (gray areas indicate the minimum error rate across various values of lag)

LAG / DATASET	1	2	3	4	5	6	7	8	9	10
Adiac	0.47	0.51	0.61	0.63	0.64	0.63	0.67	0.64	0.64	0.66
50words	0.69	0.74	0.75	0.75	0.70	0.75	0.73	0.75	0.78	0.75
CBF	0.17	0.37	0.31	0.38	0.35	0.39	0.38	0.40	0.44	0.43
ECG200	0.23	0.29	0.31	0.30	0.32	0.30	0.31	0.27	0.36	0.27
FaceAll	0.40	0.45	0.52	0.60	0.69	0.64	0.73	0.70	0.73	0.71
FaceFour	0.38	0.49	0.56	0.47	0.47	0.64	0.50	0.65	0.59	0.69
Fish	0.39	0.64	0.70	0.69	0.69	0.70	0.67	0.67	0.62	0.65
GunPoint	0.22	0.24	0.35	0.32	0.32	0.40	0.36	0.37	0.36	0.37
Lighting2	0.34	0.39	0.39	0.41	0.46	0.43	0.46	0.39	0.49	0.43
Lighting7	0.64	0.58	0.71	0.67	0.67	0.63	0.70	0.66	0.68	0.62
OSULeaf	0.45	0.19	0.44	0.35	0.50	0.41	0.48	0.44	0.52	0.48
SwedishLeaf	0.23	0.19	0.37	0.29	0.36	0.33	0.39	0.36	0.40	0.37
SyntheticContro	0.13	0.50	0.47	0.55	0.53	0.52	0.51	0.53	0.55	0.59
Trace	0.11	0.35	0.28	0.26	0.33	0.30	0.30	0.33	0.23	0.32
TwoPatterns	0.29	0.74	0.32	0.74	0.35	0.74	0.38	0.74	0.46	0.74
Wafer	0.03	0.01	0.00	0.01	0.01	0.03	0.02	0.04	0.02	0.04
Yoga	0.26	0.22	0.25	0.27	0.27	0.27	0.27	0.30	0.28	0.29
Beef	0.43	0.40	0.33	0.47	0.37	0.50	0.37	0.47	0.37	0.47
Coffee	0.04	0.29	0.50	0.54	0.61	0.54	0.46	0.54	0.32	0.46
OliveOil	0.33	0.47	0.73	0.60	0.60	0.53	0.67	0.53	0.63	0.60
LAG / DATASET	11	12	13	14	15	16	17	18	19	20
Adiac	0.62	0.64	0.65	0.67	0.66	0.67	0.68	0.64	0.70	0.69
50words	0.79	0.78	0.81	0.79	0.81	0.82	0.82	0.83	0.82	0.82
CBF	0.41	0.44	0.43	0.46	0.42	0.44	0.43	0.43	0.44	0.42
ECG200	0.36	0.33	0.31	0.33	0.30	0.34	0.28	0.32	0.29	0.35
FaceAll	0.71	0.72	0.72	0.74	0.73	0.74	0.75	0.73	0.74	0.74
FaceFour	0.63	0.66	0.66	0.67	0.63	0.58	0.53	0.58	0.51	0.59
Fish	0.63	0.69	0.65	0.67	0.69	0.69	0.61	0.69	0.67	0.67
GunPoint	0.39	0.38	0.36	0.34	0.40	0.34	0.37	0.40	0.37	0.37
Lighting2	0.46	0.38	0.44	0.41	0.46	0.43	0.39	0.36	0.43	0.46
Lighting7	0.59	0.60	0.67	0.60	0.64	0.64	0.63	0.67	0.62	0.59
OSULeaf	0.51	0.48	0.54	0.53	0.56	0.52	0.51	0.49	0.56	0.56
SwedishLeaf	0.37	0.38	0.42	0.41	0.42	0.42	0.42	0.42	0.43	0.44
SyntheticContro	0.57	0.58	0.53	0.56	0.61	0.58	0.55	0.59	0.58	0.59
Trace	0.25	0.35	0.30	0.31	0.37	0.39	0.29	0.40	0.36	0.43
TwoPatterns	0.56	0.75	0.68	0.74	0.72	0.74	0.74	0.74	0.75	0.75
Wafer	0.03	0.04	0.04	0.04	0.06	0.05	0.06	0.06	0.06	0.06
Yoga	0.27	0.28	0.30	0.29	0.31	0.29	0.30	0.28	0.29	0.29
Beef	0.33	0.50	0.27	0.53	0.27	0.50	0.37	0.43	0.40	0.50
Coffee	0.43	0.39	0.43	0.36	0.39	0.32	0.39	0.39	0.36	0.36
OliveOil	0.70	0.50	0.70	0.50	0.73	0.53	0.60	0.50	0.67	0.57

Table D2. k-NN similarity search error rates for feature-based representation (gray areas indicate the minimum error rate across various values of lag)

LAG / DATASET	1	2	3	4	5	6	7	8	9	10
Adiac	0.65	0.67	0.78	0.78	0.77	0.78	0.79	0.78	0.79	0.79
50words	0.80	0.83	0.84	0.80	0.82	0.82	0.83	0.83	0.84	0.84
CBF	0.39	0.49	0.47	0.50	0.49	0.51	0.50	0.52	0.51	0.52
ECG200	0.39	0.39	0.42	0.40	0.41	0.43	0.42	0.42	0.42	0.41
FaceAll	0.69	0.70	0.72	0.76	0.78	0.81	0.82	0.83	0.83	0.83
FaceFour	0.60	0.62	0.64	0.61	0.64	0.63	0.64	0.63	0.63	0.61
Fish	0.80	0.79	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82
GunPoint	0.49	0.48	0.48	0.49	0.49	0.48	0.49	0.48	0.49	0.49
Lighting2	0.49	0.50	0.49	0.50	0.49	0.50	0.50	0.49	0.49	0.48
Lighting7	0.66	0.66	0.69	0.66	0.70	0.67	0.69	0.68	0.69	0.68
OSULeaf	0.63	0.45	0.61	0.53	0.63	0.57	0.64	0.60	0.64	0.60
SwedishLeaf	0.54	0.49	0.66	0.61	0.67	0.64	0.68	0.66	0.68	0.67
SyntheticContro	0.43	0.66	0.67	0.68	0.68	0.69	0.69	0.69	0.70	0.69
Trace	0.27	0.44	0.45	0.48	0.48	0.50	0.49	0.52	0.48	0.53
TwoPatterns	0.62	0.75	0.63	0.75	0.64	0.75	0.65	0.75	0.67	0.75
Wafer	0.16	0.14	0.13	0.14	0.14	0.17	0.17	0.17	0.18	0.18
Yoga	0.49	0.48	0.48	0.48	0.48	0.48	0.47	0.47	0.47	0.47
Beef	0.70	0.72	0.67	0.68	0.68	0.68	0.70	0.70	0.66	0.72
Coffee	0.25	0.44	0.50	0.50	0.51	0.50	0.50	0.50	0.50	0.51
OliveOil	0.50	0.62	0.62	0.62	0.64	0.63	0.62	0.62	0.62	0.60
LAG / DATASET	11	12	13	14	15	16	17	18	19	20
Adiac	0.79	0.79	0.78	0.79	0.79	0.80	0.80	0.81	0.81	0.80
50words	0.86	0.86	0.86	0.86	0.87	0.87	0.87	0.87	0.88	0.87
CBF	0.51	0.52	0.52	0.52	0.52	0.52	0.53	0.53	0.53	0.53
ECG200	0.41	0.40	0.41	0.41	0.42	0.42	0.42	0.42	0.42	0.42
FaceAll	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.83
FaceFour	0.63	0.62	0.63	0.62	0.63	0.61	0.62	0.60	0.62	0.61
Fish	0.82	0.82	0.82	0.82	0.80	0.80	0.80	0.80	0.80	0.80
GunPoint	0.49	0.49	0.49	0.48	0.49	0.49	0.49	0.49	0.49	0.49
Lighting2	0.49	0.49	0.49	0.49	0.48	0.48	0.48	0.48	0.48	0.49
Lighting7	0.69	0.68	0.68	0.67	0.67	0.66	0.68	0.67	0.67	0.67
OSULeaf	0.65	0.63	0.65	0.64	0.65	0.64	0.65	0.64	0.65	0.64
SwedishLeaf	0.68	0.68	0.69	0.70	0.69	0.70	0.69	0.69	0.69	0.69
SyntheticContro	0.70	0.70	0.70	0.70	0.70	0.70	0.70	0.70	0.70	0.70
Trace	0.49	0.53	0.51	0.53	0.52	0.53	0.51	0.52	0.51	0.53
TwoPatterns	0.71	0.75	0.74	0.75	0.75	0.75	0.75	0.75	0.75	0.75
Wafer	0.18	0.17	0.18	0.18	0.18	0.19	0.19	0.19	0.19	0.19
Yoga	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47
Beef	0.65	0.71	0.66	0.71	0.66	0.72	0.67	0.72	0.70	0.72
Coffee	0.51	0.51	0.51	0.51	0.51	0.51	0.51	0.51	0.51	0.51
OliveOil	0.62	0.62	0.61	0.62	0.61	0.63	0.62	0.61	0.62	0.61