Time Series Distance Measures

Segmentation, Classification, and Clustering of Temporal Data

vorgelegt von
Dipl.-Inf.
Stephan Spiegel

Von der Fakultät IV — Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
— Dr.-Ing. —

genehmigte Dissertation

Promotionsauschuss

Vorsitzender: Prof. Dr. Manfred Opper (TU Berlin)
Berichter : Prof. Dr. Dr. h.c. Sahin Albayrak (TU Berlin)
Berichter : Prof. Dr. Dr. h.c. mult. Jürgen Kurths (PIK)
Berichter : Prof. Dr. Friedemann Mattern (ETH Zürich)

Tag der wissenschaftlichen Aussprache: 27. Juli 2015

Berlin 2015

D 83


TIME
SERIES
DISTANCE
DATA
CLASSIFICATION
RECURRENCE
PATTERNS
CLUSTERING
WARPING
ENERGY
ALGORITHM
PROPOSED
APPROACH
APPLIANCE
CONSUMPTION
INTRODUCED
SEGMENTS
CHAPTER
MEASURE
MULTIVARIATE
ACCURACY
SIGNAL
MATRIX
SIMILAR
LENGTH
RESULTS
CONTEXT
RECURRING
SENSOR
MODEL
GIVEN
USE
SECTION
SHOWN
FIGURE
LEARNING
ONE
TWO
STATE
ORDER
HEATING
NUMBER
BASED
MINING
INDIVIDUAL
SIMILARITY
MEASURES
DTW
SEGMENTATION
WORK
ANALYSIS
USED
COST
TECHNIQUES
CLUSTER
STATES
SIGNALS
EVALUATION
VISUALIZATION
APPLICATIONS
FEATURES
POWER
DIFFERENT
PATTERN
USING
SINCE
POINTS
PROBLEM
REAL-LIFE
FURTHERMORE
COMPUTATIONAL
DATASET
MACHINE
SEGMENT
SEQUENCES
CONSIDERED
INTRODUCED
CONSUMPTION
APPROACH
APPLIANCE
PROPOSED
ALGORITHM
ENERGY
MODEL
WARPING
SENSOR
CLUSTERING
PATTERNS
SEGMENTS
RECURRING
CONTEXT
LENGTH
RESULTS
CHAPTER
RECURRENCE
MATRIX
SIMILAR
SIGNAL
HOWEVER
FIGURE
LEARNING
ONE
TWO
STATE
ORDER
ACCURACY
CLASSIFICATION
MULTIVARIATE
MEASURE
HEATING
NUMBER
BASED
MINING
INDIVIDUAL
SIMILARITY
MEASURES
DTW
SEGMENTATION
WORK
ANALYSIS
USED
COST
TECHNIQUES
CLUSTER
STATES
SIGNALS
EVALUATION
VISUALIZATION
APPLICATIONS
FEATURES
POWER
DIFFERENT
PATTERN
USING
SINCE
POINTS
PROBLEM
REAL-LIFE
FURTHERMORE
COMPUTATIONAL
DATASET
MACHINE
SEGMENT

Acknowledgement

First of all I want to express my gratitude to my family, which always believed in me and gave me the strength to follow my path - no matter what. Secondly I want to thank my close friends for lending an open ear at all times. I am very grateful to have such a wonderful family and true friends.

Furthermore, I appreciate the continued support of my colleagues at the TU Berlin. Over the last couple of years we had endless conversations about interesting scientific problems and found common solutions that eventually resulted in joint publications. In particular I wish to thank Dr. B.-J. Jain for giving me constant feedback and professional advise on my research, which ultimately led to improved results. Moreover, I give our student research assistants credit for their support in software prototyping.

In addition, I feel obliged to show my appreciation to our associates of the Potsdam Institute for Climate Impact Research, which were willing to share their scientific knowledge of nonlinear data analysis, thereby bridging gaps between theoretical physics and computer science.

Finally I want to thank our industry partners, which provided us with real-life sensor data and contributed their expert knowledge to the development of our proposed time series segmentation, classification, and clustering approaches. In collaboration with our industry partners we were able to make a contribution to important environmental issues, such as emission reduction and energy efficiency.

Abstract

Time series can be found in domains as diverse as medicine, astronomy, geophysics, engineering, and quantitative finance. In general, a time series is a sequence of data points, measured at successive points in time and spaced at uniform time intervals. This thesis is concerned with time series mining, including segmentation, classification, and clustering of temporal data. Many algorithms for these tasks depend upon pairwise (dis)similarity comparisons of (sub)sequences, which accounts for the continued research on time series distance measures as an important subroutine.

In the course of this work we introduce several novel distance measures, which describe time series characteristics that may distinguish the individual classes contained in the data. Our proposed time series distance measures address frequently encountered issues, such as the processing of multivariate data, the computational complexity of pairwise (dis)similarity comparisons, the invariance required for temporal data with distortions, the separation of mixed signals, and the analysis of nonlinear systems.

Our work contributes to the time series community by introducing novel approaches to pattern recognition in temporal data, presenting miscellaneous sensor fusion techniques for multivariate measurements, offering efficient and robust distance measures for fast time series classification, introducing previously disregarded invariance and proposing corresponding distance measures, comparing various machine learning algorithms for signal separation, and providing nonlinear models for time series mining.

In addition to our theoretical contributions, we furthermore demonstrate that our proposed time series distance measures are beneficial in real-world applications, including the optimization of vehicle engines with regard to exhaust emission and the optimization of heating control in terms of energy efficiency. Furthermore, we present several specifically developed time series mining tools, which implement our introduced distance measures and provide graphical user interfaces for straightforward parameter setting as well as exploratory data analysis.

Zusammenfassung

Zeitreihen kommen unter anderem in vielzähligen Bereichen der Medizin, Astronomie, Geophysik, Konstruktion, und Finanzwirtschaft vor. Im Allgemeinen bezeichnet man eine Zeitreihe als Sequenz von Datenpunkten die mit fortlaufender Zeit in gleichmäßigen Zeitabständen gemessen wurde. Diese These beschäftigt sich hauptsächlich mit der Auswertung von Zeitreihen, was die Segmentierung, Klassifikation, und Gruppierung von temporalen Daten beinhaltet. Viele Algorithmen die diese Aufgaben lösen bedingen den paarweisen Vergleich von Sequenzen, was das fortwährende Forschungsinteresse an Distanzmaßen als entscheidende Subroutine begründet.

Im Verlauf dieser Arbeit führen wir mehrere neue Distanzmaße ein welche die wesentlichen Charakteristiken von Zeitreihen erfassen und die Unterscheidung von verschiedenen, in einem Datensatz vorkommenden, Klassen ermöglichen. Unsere vorgeschlagenen Distanzmaße adressieren häufige, bei der Auswertung von Zeitreichen auftretende, Herausforderung. Dazu gehören die Untersuchung von multivariaten Daten, der Rechenaufwand von paarweisen Ähnlichkeitsberechnungen, die Messstörungen und Verzerrungen von temporalen Daten, das Trennen von gemischten Signalen, sowie die Analyse von nicht linearen Systemen.

Unsere Arbeit leistet einen Betrag im Gebiet der Zeitreihenanalyse indem wir neue Ansätze zur Erkennung von Mustern in temporalen Daten einführen, robuste Distanzmaße für die effiziente Klassifikation von Zeitreihen bereitstellen, zuvor unbeachtete Invarianz betrachten und entsprechende Distanzberechnungen vorschlagen, unterschiedliche Methoden des Maschinellen Lernens für die Trennung von Signalen vergleichen, und nicht lineare Model für die Untersuchung von Zeitreichen adaptieren.

Des Weiteren demonstrieren wir die Einsetzbarkeit unserer vorgeschlagen Distanzmaße in praktischen Anwendungen, wie z.B. bei der Optimierung von Fahrzeugmotoren in Bezug auf den Schadstoffausstoß sowie die Optimierung von Heizplänen unter Betrachtung des Energieverbrauches. Darüber hinaus präsentieren wir mehrere eigens entwickelte Zeitreihenanalysewerkzeuge die unsere eingeführten Distanzmaße anwenden.

List of Publications

The content of this thesis builds on the following publications by the author:

I. (Ch. 2) Stephan Spiegel and Brijnesh-Johannes Jain and Ernesto De Luca and Sahin Albayrak - Pattern Recognition in Multivariate Time Series - Dissertation Proposal. Proceedings of 4th Workshop for Ph.D. Students in Information and Knowledge Management (PIKM), 2011. [184]

II. (Ch. 2) Stephan Spiegel and Julia Gaebler and Andreas Lommatzsch and Ernesto De Luca and Sahin Albayrak - Pattern Recognition and Classification for Multivariate Time Series. Proceedings of the 5th International Workshop on Knowledge Discovery from Sensor Data (SensorKDD), 2011. [182]

III. (Ch. 3) Stephan Spiegel and Brijnesh-Johannes Jain and Sahin Albayrak - Fast Time Series Classification under Lucky Time Warping Distance. Proceedings of 29th Symposium on Applied Computing (SAC), 2014. [183]

IV. (Ch. 4) Stephan Spiegel and Sahin Albayrak - An Order-Invariant Time Series Distance Measure - Position on Recent Developments in Time Series Analysis. Proceedings of 4th International Conference on Knowledge Discovery and Information Retrieval (KDIR), 2012. [179]

V. (Ch. 4) Stephan Spiegel and Brijnesh-Johannes Jain and Sahin Albayrak - A Recurrence Plot-based Distance Measure. Springer Proceedings in Mathematics - Translational Recurrences: From Mathematical Theory to Real-World Applications, 2014. [185]

VI. (Ch. 4) Stephan Spiegel - Discovery of Driving Behavior Patterns. Advances in Computer Vision and Pattern Recognition, Smart Information Services - Computational Intelligence for Real-Life Applications, Springer, 2015. [177]

VII. (Ch. 5) Stephan Spiegel and Sahin Albayrak - Energy Disaggregation meets Heating Control. Proceedings of 29th Symposium on Applied Computing (SAC), 2014. [180]

VIII. (Ch. 5) Stephan Spiegel - Optimization of In-House Energy Demand. Advances in Computer Vision and Pattern Recognition, Smart Information Services - Computational Intelligence for Real-Life Applications, Springer, 2015. [178]

IX. (Ch. 6) Stephan Spiegel and Veit Schwartze and Marie Schacht and Sebastian Ahrndt and Sahin Albayrak - Heating Control via Energy Disaggregation: A Practical Demonstration [189] [In Preparation: Demo Paper for INTERACT 2015]

X. (Ch. 6) Stephan Spiegel and David Schultz and Sahin Albayrak - BestTime: Finding Representatives in Time Series Datasets. Lecture Notes in Artificial Intelligence (LNAI) Series, Springer, 2014. [187]

XI. (Ch. 6) Julia Gaebler and Stephan Spiegel and Sahin Albayrak - MatArcs: An Exploratory Data Analysis of Recurring Patterns in Multivariate Time Series. Proceedings of Workshop on New Frontiers in Mining Complex Patterns (NFMCP), 2012. [51]

Other publications by the author outside the scope of this thesis:

- David Schultz and Stephan Spiegel and Norbert Marwan and Sahin Albayrak - Approximation of Diagonal Line based Measures in Recurrence Quantification Analysis. Physics Letters A, Elsevier, 2015. [171]
- Stephan Spiegel and Jan Clausen and Sahin Albayrak and Jerome Kunegis - Link Prediction on Evolving Data Using Tensor Factorization. Proceedings of the 15th International Conference on New Frontiers in Applied Data Mining (PAKDD), 2011. [181]
- Stephan Spiegel and Jerome Kunegis and Fang Li - Hydra: A Hybrid Recommender System [Cross-linked Rating and Content Information]. Proceedings of the 1st International Workshop on Complex Networks Meet Information and Knowledge Management (CNIKM), 2009. [186]
- Esra Acar and Stephan Spiegel and Sahin Albayrak - MediaEval 2011 Affect Task: Violent Scene Detection combining audio and visual Features with SVM. Proceedings of CEUR Workshop, 2011. [4]

Contents

List of Figures

List of Tables

List of Abbreviations and Notations

	Abbreviations
1NN	1-nearest-neighbor classifier
1NN-DTW	1-nearest-neighbor classifier with DTW distance
kNN	k-nearest-neighbor classifier
ACC	amortized computational cost
ANN	artificial neural network
BU	bottom-up approach (for segmentation)
BUCP	bottom-up approach with critical point
cDTW	constraint DTW (with Sakoe-Chiba band)
cDTW_LB	constraint DTW in combination with LB
cLTW	constraint LTW (with Sakoe-Chiba band)
CP	Critical Point
CRP	Cross Recurrence Plot
CT	Classification Tree
DTW	Dynamic Time Warping (distance measure)
EDA	Exploratory Data Analysis
ED	Euclidean Distance
FHMM	Factorial Hidden Markov Model
FN	False Negative
FP	False Positive
JCRPs	Joint Cross Recurrence Plots
JRPs	Joint Recurrence Plots
LB	Lower Bound(ing)
LOI	Line Of Identity
LTW	Lucky Time Warping
MDT	Multi-Dimensional Time series
MFCCs	Mel Frequency Cepstral Coefficients
NB	Naive Bayes (classifier)
NILM	Non-Intrusive Load Monitoring

OID	Order-Invariant Distance
PAA	Piecewise Aggregate Approximation
PCA	Principal Component Analysis
RPs	Recurrence Plots
RQA	Recurrence Quantification Analysis
RRR	RecuRRence Plot-based distance
SVD	Singular Value Decomposition
TN	True Negative
TP	True Positive

Notations

$\|\| \cdot \|\|$	a norm
a	start point of time series segment
b	end point of time series segment
c	object in certain class or cluster, so that $c \in C$
$c(\cdot)$	mapping function $c : \mathbb{X} \to \mathbb{C}$ maps time series $\mathbb{X} = \{X_1, \ldots, X_t\}$ to corresponding class $\mathbb{C} = \{C_1, \ldots, C_k\}$ so that $X_i \mapsto c(X_i)$ for $i = 1 \ldots t$
$cos(\cdot, \cdot)$	cosine distance between feature vectors, so that $cos : F \times F \to \mathbb{R}^+$
$cost_Q(S)$	homogeneity of time series segment S according to Q-measure
C	class or cluster
$\|C\|$	number of objects in certain class or cluster
$\mathbb{C}$	set of (k) classes or clusters, so that $\mathbb{C} = \{C_1, \ldots, C_k\}$
$CR^{d,\epsilon}$	cross recurrence plot, comparing two trajectories in d-dimensional phase space regarding a certain ϵ-threshold
$CR^{d,\epsilon}_{i,j}$	$(i,j)^{th}$ entry in cross recurrence plot
d	dimension(ality) of time series or phase space trajectory
$d(X, Y)$	time series distance function, such that $d : X \times Y \to \mathbb{R}^+$
$d_{SVD}(X, Y)$	similarity measure between two time series (segments) based on singular value decomposition (SVD)
D	distance matrix
$D_{i,j}$	$(i,j)^{th}$ entry in distance matrix
D_k	maximum cluster separation (for k groups)
DET	recurrence quantification measure: determinism
$DTW(X, Y)$	dynamic time warping distance

	between time series X and Y
ϵ	similarity threshold (or neighborhood)
e	index for time series segments
err	classification error
E	number of similarity operations or matrix elements involved in distance calculation, e.g. with cDTW
E_k	energy of k-largest singular values
$E(k)$	cluster validation index (for k groups) based on RRR distance measure
$ED(X, Y)$	Euclidean distance between time series X and Y
$ENTR$	recurrence quantification measure: entropy
f	constant factor of amortize computational complexity
F	(arbitrary) feature vector, such as used for cosine distance
$\gamma(i, j)$	cumulative distance of $(i, j)^{th}$ entry in warping matrix
h	threshold for error of time series segmentation
i	index, e.g. in time series or distance matrix
$I(k)$	cluster validation index (for k groups)
j	index, e.g. in time series or distance matrix
$JR^{d,\epsilon}$	joint recurrence plot of d-dimensional trajectory with regard to a certain ϵ-threshold
$JR^{d,\epsilon}_{i,j}$	$(i, j)^{th}$ entry in joint recurrence plot
$JCR^{d,\epsilon}$	joint cross recurrence plot of two d-dimensional trajectories with regard to a certain ϵ-threshold
$JCR(i, j)^{d,\epsilon}$	$(i, j)^{th}$ entry in joint cross recurrence plot
k	number of classes or clusters (prototypes/representatives)
k	index of warping path, e.g. w_k for $1 \leq k \leq K$
K	length of warping path, i.e. $\mathcal{W} = \{w_1, \ldots, w_k, \ldots, w_K\}$
l	length of time series subsequence or pattern
l	diagonal line length (in recurrence plots)
l_{min}	minimum diagonal line length (used for RQA measures)
$link(\cdot, \cdot)$	average linkage function for hierarchical clustering
L	lower bound sequence
m	length of time series
n	length of time series
N_l	number of diagonal lines with length l in a recurrence plot
$O(\cdot)$	computational complexity in O notation
ρ	power parameter for cluster validation index $I(k)$
p	L^p-space, L^p-norm, also used for Minkowski distance
p	percentage of expensive similarity computation for lower bounding
$p^\epsilon(l)$	probability that a diagonal line with length l occurs in RP

	regarding a certain ϵ-threshold
P	projection into p-dimensional subspace using SVD
$P^{\epsilon}(l)$	number of diagonal lines with length l
	in a recurrence plot with certain ϵ-threshold
Q	measure of homogeneity of time series segment
	mean squared deviation of projection P
r	size of warping window
	for dynamic time warping distance
r	rank of singular value decomposition
r	factor for dimensionality reduction with PAA
R	recurrence plot or matrix
$R^{d,\epsilon}$	recurrence plot of d-dimensional trajectory
	regarding certain ϵ-threshold
$R^{d,\epsilon}_{i,j}$	$(i,j)^{th}$ entry in recurrence plot
$RATIO$	recurrence quantification measure: ratio
RR	recurrence quantification measure: recurrence rate
RRR	recurrence plot-based distance measure
$RRR(X,Y)$	recurrence plot-based distance between time series X and Y
s	number of non-overlapping time intervals for segmentation
S	a subsequence, given a time series $X = \{x_1, \ldots, x_n\}$
	a subsequence of length l is defined as $S = \{x_i, \ldots, x_{i+l-1}\}$
	for $1 \leq i \leq n - l + 1$
$S(a,b)$	a time series segment (or subsequence)
	ranging from time point a to b, so that $S(a,b) = \{x_a, \ldots, x_b\}$
$S^{(i)}_j$	on/off state of appliance i at time point j
Σ	singular values of SVD
$\mathbb{S}$	column-wise segmentation
	e.g. $\mathbb{S} = \{S_1, S_2\} = \{\{x_1, \ldots, x_i\}, \{x_{i+1}, \ldots, x_n\}\}$
	for time series $X = \{x_1, \ldots, x_n\}$
t	size of time series set, e.g $\mathbb{X} = \{X_1, \ldots, X_t\}$
$T(S,C)$	a tuple containing a segment S and its corresponding class C
$\Theta(\cdot)$	Heaviside step function, $\Theta(z) = \{1 \vert z > 0; 0 \vert z \leq 0\}$
U	upper bound sequence
U	left-singular vectors of SVD
V	right-singular vectors of SVD
$\mathcal{W}$	warping path, e.g. $\mathcal{W} = \{w_1, \ldots, w_k, \ldots, w_K\}$
W	product of singular values and right-singular vectors
x	data point
x_i	time series element
X	time series, e.g. $X = \{x_1, \ldots, x_n\}$ with length n,

	where $x_i \in \mathbb{R}^d$ for $1 \leq i \leq n$
$\mathbb{X}$	set of time series, e.g. $\mathbb{X} = \{X_1, X_2, \ldots, X_t\}$ with $\lvert\mathbb{X}\rvert = t$
y	data point
y_j	time series element
$y_j^{(i)}$	power consumption of appliance i at time point j
$\bar{y}_j$	aggregated power consumption (sum of all appliances) at time point j
$\Delta y_j^{(i)}$	first-order difference of the power signal, so that $\Delta y_j^{(i)} = y_j^{(i)} - y_{j-1}^{(i)}$
Y	time series, e.g. $Y = \{y_1, \ldots, y_m\}$ with length m, where $y_j \in \mathbb{R}^d$ for $1 \leq j \leq m$
$\mathbb{Y}$	set of time series, e.g. $\mathbb{Y} = \{Y_1, \ldots, Y_t\}$ with $\lvert\mathbb{Y}\rvert = t$ elements
z	data point
Z	cluster centroid (or medoid)
$\mathbb{Z}$	set of cluster centroids, such that $Z \in \mathbb{Z}$

Chapter 1

Introduction

People like us, who believe in physics, know that
the distinction between past, present, and future
is only a stubbornly persistent illusion.

The only reason for time is
so that everything doesn't happen at once.

- Albert Einstein

From physics point of view, time is a dimension and measure in which events can be ordered from the past through the present into the future, and also the measure of durations of events and the intervals between them [35]. Modern philosophy and neuroscience widely agrees that the sequential ordering of events can be explained by the way of human perception [47, 58, 72, 108, 141]. The same chronological ordering can be found in man made computer models that are used for the analysis and processing of temporal data [16, 34, 50, 61].

In computer science, a time series is a sequence of data points, measured typically at successive points in time spaced at uniform time intervals [16, 32, 42, 61]. The collection of time series is usually performed by sensors that measure physical quantities and convert them into a signal, which can be interpreted by humans or machines. Since sensors are becoming increasingly inexpensive and pervasive [12, 26, 73, 102, 144], vast quantifies of temporal data can be found in domains as diverse as medicine [30, 106, 135], astronomy [158, 190], geophysics [18, 52, 78, 134], engineering [64, 70, 195], and quantitative finance [68, 197]. Furthermore, it has been shown that other data formats such as images [21, 71, 91, 215], videos [17, 75, 76], and audio signals [65, 55, 208] can be represented and interpreted as time series.

Time series analysis is mainly concerned with the extraction of meaningful statistics and other characteristics of data sequences [36, 45, 161]. Due to the natural temporal ordering of the data, time series analysis is distinct from other data analysis problems [9, 36, 69, 88, 157, 161]. In the context of data mining, pattern recognition and machine learning time series analysis is primarily used for classification [32], clustering [112], and segmentation [85]. Many algorithms for these tasks depend upon pairwise (dis)similarity comparisons of (sub)sequences by means of a distance measure [36, 42, 154, 161]. Hence, many scientific endeavors in time series mining aim at studying the properties of distance measures in consideration of their importance as essential subroutine [93, 154, 159, 160, 201, 221].

The distance between time series need to be carefully defined in order to reflect the underlying (dis)similarity of such data [8, 9]. To determine the (dis)similarity between time series a distance measure can compare either the raw data, extracted features, or model parameters [32, 36, 42, 112, 149, 161]. However, it usually requires domain knowledge to understand the characteristics which discriminate different classes of time series [24, 146, 149, 176].

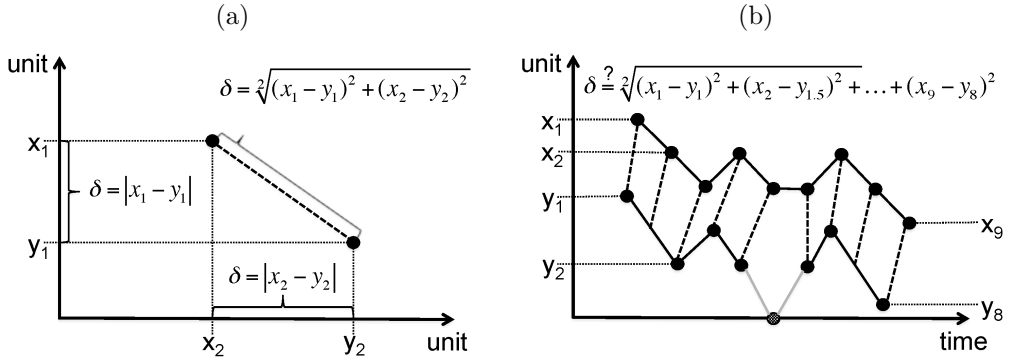


Figure 1.1: (a) Euclidean distance of data points in two-dimensional space. (b) Distance of high-dimensional time series data with multiple distortions.

Figure 1.1 illustrates (a) the Euclidean distance (ED) for data points in two-dimensional space and (b) the problems that arise when measuring the distance for high-dimensional time series with multiple distortions [49, 36, 42, 161]. Our human eye is able to recognize the shape-based similarity between time series $X = \{x_1, \ldots, x_9\}$ and $Y = \{y_1, \ldots, y_8\}$, even though time series Y has different offset, amplitude, length, scaling, phase, and exhibits missing values [119, 115]. In order to measure the 'true' underlying distance between time series X and Y we necessarily need to allow a non-linear alignment of the measurements, such as performed by the popular Dynamic Time Warping

(DTW) distance [93, 155, 160]. In general, the choice of distance measure depends on the signal distortions or rather the invariance required by the domain [8, 9].

1.1 Motivation

Although time series analysis has a long tradition, there still remain unsolved problems that spur further research. According to our literature survey the following issues are worth investigating:

1.1.1 Multivariate Data

In order to understand the properties and behavior of complex systems it is often necessary to measure and observe multiple physical quantities. These measurements can be described as multivariate time series, which usually require special handling [1, 2, 53, 70, 79, 195, 201]. We aim at developing time series distance measures, which are able to consider multiple parameters in a sensor fusion approach. More precisely, we intent to segment multivariate time series according to changes in correlation structure [182, 184] and classify/cluster time series regarding their co-occurring multivariate patterns or subsequences [51, 179, 185, 187].

1.1.2 Computational Complexity

With an ever more increasing size of data collections, the computational complexity of time series distance measures has gained particular importance [20, 83, 89, 91, 93, 98, 125, 154, 155, 166, 167, 211, 221]. Although lower bounding with pruning has been shown to reduce the number of expensive similarity computations, the efficiency gain depends strongly on the pruning power and varies with the data under study [183]. Instead of applying additional speed-up techniques, we aim at developing robust distance measures with low computational complexity that maintain or even improve classification accuracy [183].

1.1.3 Invariance to Distortions

The choice of time series distance measure depends on the invariance required by the domain [8, 9]. Recent work has introduced techniques to efficiently measure similarity between time series with invariance to (various combinations of) the distortions of warping, uniform scaling, offset, amplitude scaling,

phase, occlusion, uncertainty and wandering baseline [44, 84, 94, 201]. However, there are time series with other kind of invariance that have been missed by the community. In this work we aim to design distance measures which are able to determine the (dis)similarity of time series that exhibit similar subsequences in arbitrary order [179, 185, 187].

1.1.4 Superimposed Signals

The application of traditional shape-based distance measures on superimposed signals is generally without success, because the shape of such time series is an overlay of several individual patterns that are out of alignment [6, 48, 209, 220, 222]. Recent studies have shown that probabilistic models are well-suited to infer the individual components of a mixed signal [60, 97, 99, 100, 111, 114, 120, 162, 164, 192]. Therefore, we aim at evaluating models that are based on or adapted to a theory of probability and can be used to define time series distance measures [180, 189].

1.1.5 Nonlinear Systems

The output of a nonlinear system is not directly proportional to the input, meaning that it does not satisfy the superposition principle [128, 131, 132]. This phenomenon can be observed in complex systems such as the human heart or brain [106, 135]. When studying such complex systems we are unable to determine all the factors that influence the measured signal or time series. However, nonlinear analysis is able to explain dynamic effects like chaos, bifurcations, and harmonics [28, 45, 129, 130, 133, 134, 202, 204]. We aim to adopt nonlinear analysis to measure the distance between time series that exhibit distinct recurrent behavior [51, 179, 185, 187].

1.2 Outline of Thesis

Throughout this thesis, all our investigations are devoted the pairwise similarity comparison of time series. We propose time series distance measures that work either on raw data, extracted features, or model parameters. In the course of our work, we evaluate the proposed distance measures on various data mining tasks including, segmentation, classification, and clustering of time series. An outline of the thesis and a categorization of the individual chapters can be found in Figure 1.2(a). An overview of the research aspects that motivate the individual chapters of this thesis is shown in Figure 1.2(b).

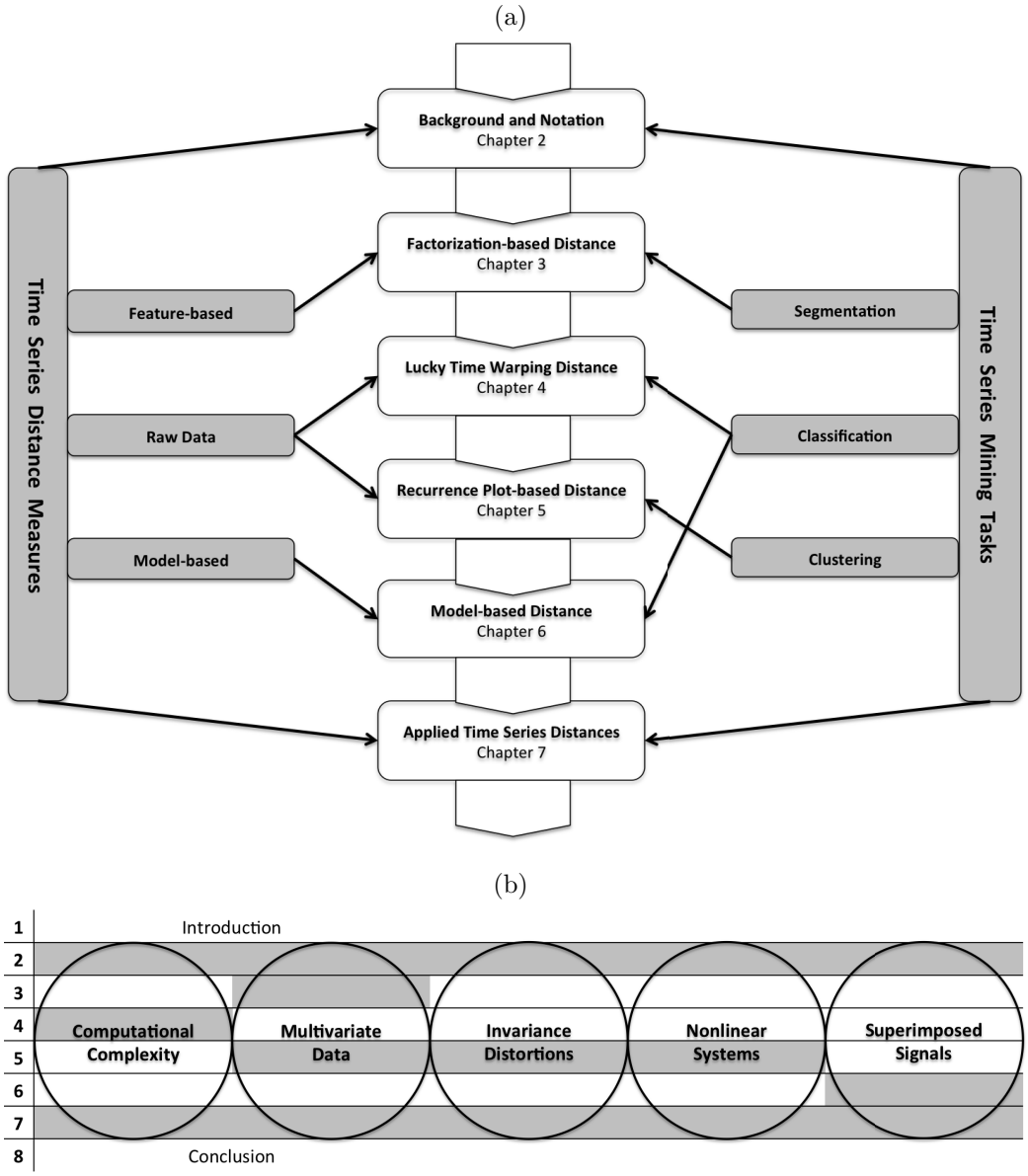


Figure 1.2: (a) Outline of thesis with a categorization of each individual chapter according to the performed time series mining task and employed time series distance measure. (b) Overview of research aspects that motivate the individual chapters of this thesis. Chapter 2 and 7 cover all considered aspects, since they introduce background/notation and present applied time series distances respectively. Apart from Chapter 5, which is motivated by several aspects, all main chapters touch on one subject only. Details on the motivation are presented in Section 1.2(b).

1.2.1 Background and Notation

Chapter 2 provides necessary background for following and understanding the main section (Chapter 3-6) of this thesis. We present a general notion of time series and discuss several state-of-the-art techniques that relate to time series mining. Furthermore, we illuminate the disadvantages and limitations of existing time series distance measures, which eventually motivated our own scientific endeavor.

1.2.2 Factorization-based Distance

In Chapter 3 we propose a generic three-step approach to the recognition of contextual patterns in multivariate time series which involves features extraction, segmentation, and clustering. The crux of our approach is a SVD-based bottom-up algorithm that identifies internally homogeneous time series segments, which are subsequently grouped and assigned to higher-level context. According to Figure 1.2(a), we categorize the subject matter of Chapter 3 as a time series segmentation task which uses a feature-based distance measure. The presented material summarizes our initial work on multivariate time series analysis [182, 184].

1.2.3 Lucky Time Warping Distance

Chapter 4 is concerned with the classification of time series by means of their shape drawn from the raw data, as shown in Figure 1.2(a). We propose a novel distance, with linear space and time complexity, which uses a greedy algorithm to accelerate distance calculations for nearest neighbor classification. Although our proposed time series distance is an approximation of the 'hard-to-beat' DTW distance, our approach is able to achieve higher classification accuracy on certain datasets. The presented material is based on our recent findings [183].

1.2.4 Recurrence Plot-based Distance

In Chapter 5 we propose a novel time series distance measure, based on the theoretical foundation of recurrence plots, which enables us to determine the (dis)similarity of multivariate time series that contain segments of similar trajectories at arbitrary positions. We use recurrence quantification analysis to measure the structures observed in recurrence plots and to investigate dynamical properties, such as determinism, which reflect the pairwise (dis)similarity of time series. As shown in Figure 1.2(a), Chapter 5 is mainly

concerned with clustering of time series by means of raw data from multiple sensors or data sources. This chapter has grown from our preliminary work on invariance [179] and our recent article about recurrences [185].

1.2.5 Model-based Distance

Chapter 6 gives attention to superimposed time series, in particular, aggregated energy signals such as coming from smart meters. We investigate energy disaggregation techniques to infer appliance states and consequently derive higher-level context about household occupancy. Since we identify the individual appliances with the help of probabilistic inference, Figure 1.2(a) categorizes the content of this chapter as a time series classification task which uses a model-based distance measure. Our comparison of energy disaggregation techniques has been described an earlier work [180].

1.2.6 Applied Time Series Distances

In Chapter 7 we present several of our software prototypes which were designed to solve real-world time series mining task by means of the previously introduced distance measures. Due to the broad spectrum of mining tasks and distance measures that are covered by this division of our thesis, we consider Chapter 7 as an overview about possible applications related to the above discussed problem settings. The presented software prototypes were already demonstrated to and appraised by an international audience [51, 187, 189] before.

1.2.7 Conclusion and Perspectives

Finally Chapter 8 concludes this thesis, discusses the relevance of our findings, and gives an outlook on future work. Our main contributions are summarized in the following section, described in part at the end of each chapter, and put into a global perspective in Chapter 8.

1.3 Main Contribution

Since we studied various time series mining tasks and proposed several novel distance measures, our contribution is manifold. The most important findings are summarized below.

1.3.1 Pattern Recognition

Although the sensory perception of humans is designed to recognizing patterns within our natural environment, we usually have difficulties to comprehend huge amounts of data as generated by sensors. In recent years, an increasing number of applications have employed data mining and machine learning techniques to find patterns in big amounts of data. Patterns may describe certain events, situations, or other high-level context, which is used by machines or humans for further decision making. In Chapter 3 we introduce a three-step approach for segmenting vehicular sensor data [182, 184]. In our case the resulting segments represent complex drive maneuvers, which are used for analysis of exhaust emission, but the proposed approach can also applied to other problem domains. In Chapter 6 we evaluate various probabilistic machine learning models which may be employed to recognize patterns in superimposed signals [180, 189]. Although we use these models to infer appliance states from aggregates energy signals, they can be used for other time series with similar properties as well. Therefore, we contribute to the time series community in that we present several possible ways how to recognize patterns in different kind of sensor data.

1.3.2 Sensor Fusion

In some domains we are confronted with measurements from multiple sensors, which observe various physical quantities of one and the same system. In order to understand the properties and behaviors of such complex system we need to process all measurements jointly in a sensor fusion approach. In Chapter 3 we present a sensor fusion approach for multivariate time series which is based on singular value decomposition [182, 184]. We consider each measurement or individual time series as a vector in the matrix that is subsequently factorized. In Chapter 5 we propose a novel formalization which allows us to analyze multivariate time series by joining multiple cross recurrence plots [51, 179, 185, 187]. Our formalization is very convenient, since it allows us to analyze different physical quantities simultaneously. Both sensor fusion approaches were applied to vehicular sensor data recording during car drives. However, both models are applicable to other domains with multivariate time series that were measured simultaneously and exhibit the same sample rate. We made a contribution by presenting various sensor fusion techniques and introducing novel ways for multivariate time series analysis.

1.3.3 Fast Classification

In time series classification, the combination of 1-Nearest-Neighbor (1NN) classifier with Dynamic Time Warping (DTW) distance has been shown to achieve high accuracy. However, if naively implemented, the 1NN-DTW approach is computationally demanding, because 1NN classification usually involves a great number of quadratic DTW calculations. In Chapter 4 we propose a novel Lucky Time Warping (LTW) distance, with linear time and space complexity, which uses a greedy algorithm to accelerate distance calculations [183]. The results show that, compared to constrained DTW with lower bounding, our LTW distance trades classification accuracy against computation time reasonably well, and therefore can be used as a fast alternative for nearest neighbor time series classification. We contribute to the time series community not only by providing an efficient and robust new distance, but also by correcting erroneous beliefs and opening new avenues regarding time series analysis.

1.3.4 Nonlinear Modeling

Recurrence plots aim to visualize and investigate recurrent states of nonlinear dynamical systems. Recurrence quantification analysis is used to quantify the structures observed in recurrence plots. In Chapter 5 we adopt recurrence plots and corresponding recurrence quantification analysis to measure the (dis)similarity of time series [51, 179, 185, 187]. In particular, we investigate dynamical properties like the determinism, which accounts for recurring subsequences of predetermined minimum length. Based on the formalization of cross recurrence plots and the denotation of determinism, we define a novel recurrence plot-based (RRR) distance. Given a set of time series, we employ the introduced RRR distance to find representatives which best comprehend the recurring temporal patterns contained in the data. Although Chapter 5 primarily focuses on clustering, our proposed RRR distance measure can also be used as a subroutine for other time series mining task. To the best of our knowledge, this is the first attempt to solve time series mining problems with nonlinear data analysis and modeling techniques commonly used by theoretical physicist. Our contribution bridges the gaps between time series analysis and nonlinear analysis by providing new nonlinear models for time series mining.

1.3.5 Order-Invariance

The choice of time series distance measure depends on the invariance required by the domain. Recent work has introduced techniques designed to efficiently measure similarity between time series with invariance to (various combinations of) the distortions of warping, uniform scaling, offset, amplitude scaling, phase, occlusions, uncertainty and wandering baseline. In Chapter 5 we propose a novel order-invariant distance measure which is able to determine the (dis)similarity of time series that exhibit similar subsequences at arbitrary positions [51, 179, 185, 187]. Order invariance is an important consideration for many real-life data mining applications, where sensors record contextual patterns in their naturally occurring order and the resulting time series are compared according to their co-occurring contextual patterns regardless of order or location. However, relevant literature is lacking a time series distance measure which is able to determine the (dis)similarity of signals that contain multiple similar events at arbitrary positions in time. Commonly used distance measures like *ED* and *DTW* are not designed to deal with order invariance, because they discriminate time series according to their shapes and fail to recognize cross-alignments between unordered subsequences. We made a contribution by introducing order invariance for time series, which has, to our knowledge, been missed by the community.

1.3.6 Source Separation

In certain applications, such as audio scene analysis and non-intrusive load monitoring, we are confronted with the problem that several signals have been mixed together into a combined signal and the objective is to recover the original component signals from the combined signal. The classical example of source separation is the 'cocktail party problem', where a number of people are talking simultaneously in a room, and a listener is trying to follow one of the discussions. The human brain can handle this sort of audio source separation problem, but it is a difficult problem in digital signal processing. In Chapter 6 we evaluate the performance of various machine learning models in the context of non-intrusive load monitoring, where an aggregated energy signal, such as coming from a smart meter, is analyzed to deduce what appliances are used in a household [180, 189]. Although the investigated models were merely used to solve the energy disaggregation problem, they can also be employed to separate the individual sources of mixed signals found in other domains. Our comprehensive comparison of different machine learning models contributes to the decision making process in similar source separation problems.

1.3.7 Applications

Although this doctoral thesis is primarily concerned with theoretical problems in time series analysis, our entire scientific endeavor is motivated by practical applications.

One important application area of our work is the optimization of vehicle engines with regard to exhaust emission [51, 179, 182, 184, 185, 187], which we address in Chapter 3, 5, and 7.2. Our work on engine optimization is a cooperation with researchers and engineers from one of the leading car manufacturers, who aim to run emission simulations based on operational profiles that characterize recurring driving behavior. To obtain real-life operational profiles the automotive engineers collect sensor data from test drives for various combinations of driver, vehicle and route. In Chapter 3 we propose a generic three-step approach to recognition of contextual patterns in multivariate time series which can be used to identify complex driving maneuvers in real-life vehicular sensor data [182, 184]. Chapter 5 presents another approach which identifies time series prototypes/representatives that best comprehend the recurring temporal patterns contained in a corresponding dataset [185]. We apply this approach to determine operational profiles that comprise frequently recurring driving behavior patterns, but our proposed model can also be used for time series dataset from other domains. Furthermore, Chapter 7.2 introduces BestTime, a platform-independent Matlab application with graphical user interface, which enables our collaborators to identify time series prototypes/representative in large datasets, allows for easy parameter setting, and provides visual results for straightforward analysis [187].

Another environmentally beneficial application that spurred our scientific endeavor is heating control [180, 189], since heating accounts for the biggest amount of total residential energy consumption. Smart heating strategies allow reducing such energy consumption by automatically turning off the heating when the occupants are sleeping or away from home. The present context or occupancy state of a household can be deduced from the appliances that are currently in use. In Chapter 6 we investigate energy disaggregation techniques to infer appliance states from an aggregated energy signal measured by smart meters, which are installed in a rapidly increasing number of households [180]. The main advantage of our proposed approach is its simplicity in that we refrain from implementing new sensor infrastructure in residential homes, which will eventually lead to higher acceptance rates among residents and provides alternative avenues for novel strategies in heating control and scheduling. In Chapter 7.1 we demonstrate SOE, a prototypical heating control system, which recommends optimized heating schedules that aim to

reduce the residential energy consumption [189]. SOE computes the optimized heating schedules based on manual adjustments of the residents and automatically determined occupancy states that were inferred by means of energy disaggregation techniques. In addition, SOE enables the residents to monitor and control their heating from remote using mobile devices. Our implementation of the SOE heating control system provides insights into practicality and usability, which are valuable for the intended deployment in real estates.

In this manner our scientific achievements contribute to important environmental issues, namely emission reduction [51, 179, 182, 184, 185, 187] and energy efficiency [180, 189].

1.4 Out of Scope

This thesis considers time series analysis in the context of data mining, machine learning, and pattern recognition. Our main purpose is to use time series analysis for the segmentation, classification, and clustering of temporal data. However, there are other motivations available for time series analysis, which are not covered by this thesis.

1.4.1 Forecasting and Prediction

In the context of statistics the primary goal of time series analysis is forecasting. Statistical models are able to predict the likely outcome of a time series in immediate future, given knowledge of the most recent observations. Regression analysis is one method to estimate future values of a signal based on its previous behavior, where predictions are usually based on statistical interpretation of time series properties in time domain. Other methods for the prediction of future values in time series are autoregressive (AR), fractional (F), integrated (I), and moving average (MA) models. Combinations of these ideas are known as ARMA, ARIMA, and ARFIMA models. Other types of nonlinear time series models represent the changes of variance or variability over time, also known as heteroskedasticity. These models comprise a wide variety of representations such as autoregressive conditional heteroskedasticity (ARCH) and related variations, including GARCH, TARCH, and others. Although our work does not focus on forecasting and prediction, there exists a huge body of literature on this topic [16, 57, 168].

1.4.2 Signal Estimation

In the context of signal processing the purpose of time series analysis is signal detection and estimation, which is based on harmonic analysis and filtering of signals in frequency domain. Most signal estimation techniques, such as the Fourier Transform and Kalman filter, aim at filtering signals from noise and predicting signal values at a certain points in time. Although there are difference in terminology between signal processing and time series analysis, both terms are often confused and used ambiguously. In general, signal processing is considered as the mathematical manipulation of an information signal to modify or improve it in some way, and, therefore it differs significantly from the research problems addressed in our work. This thesis gives no attention to traditional signal processing, but concentrates on pairwise similarity comparisons of time series by means of distance measures. However, interested readers which want to know more about signal detection and estimation may refer to the following publications [81, 152, 196].

1.5 In Summary

The research focus of this thesis is on time series distance measures, since the pairwise comparison of temporal (sub)sequences is an essential subroutine for many time series mining tasks. Our work introduces several novel time series measures that work either on raw data, extracted features, or model parameters. We employ the proposed distance measures to solve time series mining tasks as diverse as segmentation, classification, and clustering. In doing so we address important research issues such as the analysis of multivariate time series, the computational complexity of time series distance measures, the invariance required for temporal sequences with distortions, the separation of mixed signals, and the analysis of nonlinear systems. Our work contributes to the time series community by introducing novel approaches to pattern recognition in time series, presenting various sensor fusion techniques for multivariate time series, providing efficient and robust distance measures for fast time series classification, providing nonlinear models for time series mining, introducing previously disregarded invariance and proposing corresponding distance measures, comparing various machine learning model for signal separation, and relating our developments to real-world applications. In general, we consider time series analysis in the context of data mining, machine learning, and pattern recognition, wherefore time series forecasting/prediction and signal estimation are out of scope for this thesis.

Chapter 2

Background and Notation

At this point we introduce the mathematical notation of time series that is used throughout the following chapters. Furthermore, we present some theoretical background on time series mining and related concepts, which lay the foundation of our own work.

At first we give a general introduction to time series as well as distance measurements in high-dimensional real vector space, before explaining the principles of time series distance measures. Subsequently we discuss related problems such as the design of distance measures that are invariant to time series distortions and the computational complexity of pairwise similarity comparisons. Furthermore we briefly describe the most common time series mining tasks, including segmentation, classification, and clustering of temporal data sequences. In addition, we give some background on recurrence plots and corresponding recurrence quantification analysis, which we employ to differentiate the behavior and properties of time series.

2.1 Time Series

In general, a time series $X = \{x_1, \ldots, x_n\}$ is an ordered set of n real-valued numbers. The sequence of numbers (x_i for $1 \leq i \leq n$) is usually a temporal ordering, but other well-defined orderings, such as handwritten text or shapes [214], can also be fruitfully considered as time series [142].

Depending on the context we sometimes find alternative notations for time series in related literature. For instance, in the context of time series mining researchers typically speak about a query Q and candidate C time series [154], whereas in reference to nonlinear analysis theoretical physicists traditionally refer to a dynamical system x or y [132]. For the sake of consistency we use the symbols X and Y (written in capital letters) to denote

temporal data sequences within this thesis.

A lot of time series mining algorithms do not consider global properties, but evaluate local subsequences [142]. Given a time series $X = \{x_1, \ldots, x_n\}$, a subsequence of length l is defined as $S = \{x_i, \ldots, x_{i+l-1}\}$ for $1 \leq i \leq n - l + 1$. Accordingly, X contains $n - l + 1$ subsequences of length l. Depending on the discussed problem setting, we also refer to a subsequence as time series segment or temporal pattern.

In many time series applications we have to deal with multivariate time series, which consist of synchronously recorded (unambiguous) variables that where measured over the same time interval with the same sample rate. Such multivariate time series $X = \{x_i, \ldots, x_n\}$ are generally represented as a sequence $1 \leq i \leq n$ of d-dimensional measurements, so that $x_i \in \mathbb{R}^d$. Typical examples for multivariate time series are gyroscope or acceleration data, but also audio signals (e.g. represented as acoustic features) [70].

2.2 Distance Measures

Distance is a numerical description of how far apart objects are. In everyday usage, distance may refer to a physical length, e.g. a signpost saying that from your current location in Berlin its '500 meters to the Museum Island' and '2.5 kilometers to the New National Gallery'.

In mathematics, a distance function or metric is a generalization of the concept of physical distance. More formally, a metric is a function that behaves according to a specific set of conditions (non-negativity, identity of indiscernibles, symmetry, triangle inequality), and is a concrete way of describing what it means for elements of some space to be 'close to' or 'far away from' each other. These conditions express intuitive notions about the concept of distance. For example, that the distance between distinct points is positive and the distance from x to y is the same as the distance from y to x. The triangle inequality means that the distance from x to z via y is at least as great as from x to z directly. A semi-metric is a distance function that satisfies all conditions except for the triangle inequality.

Euclidean metric is the 'ordinary' distance between two points that one would measure with a ruler, and is given by the *Pythagorean* formula. However there exists a multitude of metrics, including the *Manhattan* distance that measures distance following only the axis-aligned directions and the *Chebyshev* distance that measures distance assuming only the most significant dimension is relevant.

The *Minkowski* distance is a generalization that unifies *Manhattan*, *Euclidean*, and *Chebyshev* distance. Assuming two n-dimensional data points,

x and y, the Minkowski distance is defined as:

$$\left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}} \tag{2.1}$$

The *Minkowski* distance typically uses p being 1 or 2, which corresponds to *Manhattan* distance or *Euclidean* distance respectively. In the limiting case of p reaching infinity, we obtain the *Chebyshev* distance:

$$\lim_{p \to \infty} \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}} \tag{2.2}$$

In everyday life, we usually measure distances in 3-dimensional space, which is how we perceive our world. However, the discussed metrics also allow distance measurements in high-dimensional space. This property is important for determining the distance between high-dimensional time series, where the dimensionality generally corresponds to the time series length. The representation of time series as real-valued vectors $X \in \mathbb{R}^n$ in n-dimensional *Euclidean* space allows the application of a wide range of data mining and machine learning algorithms that use gradient-based methods. However, as mentioned in Section 2.4, the Euclidean space is not suitable for time series with local scaling invariance or other distortions.

Similarity measures are in some sense the inverse of distance metrics, since they take large values for similar objects and either zero or negative value for very dissimilar objects. In information retrieval, cosine similarity is a commonly used similarity measure, defined on vectors arising from the bag of words model. In machine learning, common kernel functions such as the RBF kernel can be viewed as similarity functions [200].

2.3 Time Series Distance Measures

The distance between time series needs to be carefully defined in order to reflect the underlying (dis)similarity of such data [36]. This is particularly desirable for segmentation, classification, clustering, similarity-based retrieval and other mining procedures of time series [63].

Suppose we have two time series, $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_n\}$, of length n. To measure their similarity, we can use, for instance, the *Euclidean Distance (ED)*:

$$ED(X,Y) = \sqrt[2]{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2.3}$$

Although ED is a simple measure, it is competitive for many problems [36]. For this reason, ED has been suggested as a baseline measure for time series data mining benchmarks [87], which we considered for our empirical evaluation when introducing novel time series distances. Nevertheless, in many domains the time series data is distorted in some way, and either the distortion must be removed before using ED, or a more robust measure must be used instead [9].

Dynamic Time Warping (DTW) allows a more intuitive distance measure for time series that have similar shape, but are not aligned in time [93]. Given two time series $X = \{x_1, x_2, \ldots, x_i, \ldots, x_n\}$ and $Y = \{y_1, y_2, \ldots, y_j, \ldots, y_m\}$ of length n and m respectively, we can align these sequences using DTW by constructing an n-by-m matrix where element (i, j) contains the distance $d(x_i, y_j) = (x_i - y_j)^2$ between the two points x_i and y_j. Consequently, a warping path $\mathcal{W} = \{w_1 \ldots w_K\}$ is a contiguous set of matrix elements (as stated in Eqn. 2.4) that defines a mapping between X and Y under several constrains (such as boundary condition, continuity, monotonicity) [93]:

$$DTW(X,Y) = \min\left\{\sqrt[2]{\sum_{k=1}^{K} w_k}\right. \tag{2.4}$$

This path can be found using dynamic programming to evaluate the following recurrence which defines the cumulative distance $\gamma(i, j)$ as the distance $d(i, j)$ found in the current cell and the minimum of the cumulative distances of the adjacent elements [93]:

$$\begin{aligned}\gamma(i,j) = d(x_i, y_j) + \min\{&\gamma(i-1, j-1),\\ &\gamma(i-1, j),\\ &\gamma(i, j-1)\}\end{aligned} \tag{2.5}$$

The *Euclidean Distance* between two sequences can be seen as a special case of DTW where both time series have same length and the warping path complies with the main diagonal of the distance matrix. Although DTW exhibits quadratic time and space complexity $O(mn)$, it has been demonstrated to outperform ED in terms of accuracy for almost all time series mining task [36, 83]. To justify the introduction of novel distance measures, we always compare to DTW in our experiments.

Although there exists a plethora of time series similarity functions, we have already introduced the two most widely-used distance measures [95], namely ED and DTW. In general, time series distance measures can be categorized into L^p-norms and elastic measures that handle local time shifting [23]. ED, as stated in Eqn. 2.3, is associated with the L^2-norm and considers time series as points in n-dimensional space. The DTW distance, as formalized in Eqn. 2.4 and 2.5, belongs to the group of elastic measures that handle local scaling invariance. Following the guidelines for time series data mining benchmarks [87], we consider ED as a baseline measure and DTW distance as the state-of-the-art approach when evaluating our newly introduced distance measures.

2.4 Invariance to Distortions

The choice of time series distance measure depends on the invariance required by the domain [9]. Recent work has introduced techniques designed to efficiently measure similarity between time series with invariance to (various combinations of) the distortions of warping, uniform scaling, offset, amplitude scaling, phase, occlusions, uncertainty and wandering baseline [9]. Some of these time series distortions and their corresponding implications for distance measures are explained below.

Important considerations regarding time series distance measures include *amplitude invariance* and *offset invariance* [9]. If we try to compare two time series measured on different scales they will not match well, even if they have similar shapes. Similarly, even if two time series have identical amplitudes, they may have different offsets. To measure the true underlying similarity we must first center and scale the time series (by z-normalization).

Furthermore, *local scaling invariance* or rather *warping invariance* [9] should be taken into account when comparing time series. This invariance is necessary in almost all biological signals, which match only when one is locally warped to align with the other. Recent empirical evidence strongly suggests that *Dynamic Time Warping* is a robust distance measure which works exceptionally well [36].

In contrast to the *localized scaling* that DTW deals with, in many datasets we must account for *uniform scaling invariance* [9], where we try to match a shorter time series against the prefix of a longer one. The main difficulty in creating uniform scaling invariance is that we typically do not know the scaling factor in advance, and are thus condemned to testing all possibilities within a given range [83].

Phase invariance [9] is important when matching periodic time series

such as heart beats. By holding one time series fixed, and testing all circular shifts of the other, we can achieve phase invariance.

In domains where a small subsequence of a time series may be missing we must consider *occlusion invariance* [9]. This form of invariance can be achieved by selectively declining to match subsections of a time series. However, most real-life problems require *multiple invariance*.

2.5 Computational Complexity

The focus of computational complexity theory is to classify computational problems according their inherit difficulty, and relating those classes to each other. In time series mining, we are mainly concerned with the computational complexity of distance measures, which determine the pairwise similarity of temporal data sequences. The computational complexity of a distance measure generally depends on the length of the compared time series. For instance, using *Euclidean* distance a similarity comparison of two time series of length n requires linear $O(n)$ time and space [183]. In contrast, DTW constructs a similarity matrix of size $n \times n$ and subsequently traverses all entries to find the minimum cost warping path, resulting in quadratic $O(n^2)$ time and space complexity [93, 159, 167].

In general, we are aiming to reduce the computational complexity of time series distance measures without compromising classification accuracy. Chapter 4 discusses techniques like numerosity reduction, warping constraints, abstraction, lower bounding and approximation methods that were designed to accelerate distance calculations for nearest neighbor classification of time series with local scaling invariance [183]. However, most of these techniques are also applicable to other time series mining tasks.

2.6 Time Series Segmentation

Time series segmentation algorithms can be grouped into three different strategies, including sliding window, top-town, and bottom-up [85]. In the sliding window approach a segment is grown until it exceeds some error bound, repeating the process with the next data point not included in the newly approximated segment. Using the top-town or bottom-up strategy, a segmentation is accomplished either by recursively partitioning the given time series or gradually merging initially fine-grained sections until some stopping criteria is met. The stopping criteria for the time series segmentation can be formulated in such a way that the task is completed when a

certain number of segments is produced, the maximum error for any segment exceeds a user-specified threshold, or the combined error of all segments is higher than a predefined limit [86].

In Chapter 3 we solve the time series segmentation tasks with the bottom-up strategy using the combined error of segments as stopping criteria, which generally can be formalized as followed. Given a (multivariate) time series $X = \{x_1, \ldots, x_n\}$ with $x_i \in \mathbb{R}^d$ for $1 \leq i \leq n$, and defining a subsequence of X as a set of consecutive time points $S(a, b) = \{x_a, \ldots, x_b\}$ for $1 \leq a \leq b \leq n$, the segmentation of X into s non-overlapping subsequences can be formulated as $\mathbb{S} = \{S_e(a_e, b_e)|1 \leq e \leq s\}$, where $a_1 = 1, b_s = n$ and $a_e = b_{e-1}+1$. In order to evaluate a segmentation $\mathbb{S}$ we compute the combined error of all segments $\sum_{e=1}^{s} cost(S_e(a_e, b_e))$ by means of a cost function, which returns the approximation error for a corresponding subsequence using our underlying model. The employed bottom-up algorithm initially establishes a fine-grained segmentation, e.g. $\mathbb{S} = \{S(1, 2), S(3, 4), \ldots, S(n-1, n)\}$, of the time series $X = \{x_1, \ldots, x_n\}$ and iteratively merges the lowest cost pair of segments, e.g. $S(1, 4) = \min\{cost(S_e(a_e, b_{e+1}))|1 \leq e \leq s\}$, until the combined error of all segments exceeds a predefined threshold h, e.g. $h < cost(S(1, 4)) + \ldots + cost(S(n-1, n))$.

In general, the evaluation of a segment involves two steps. First of all we require a function, which takes in a time series and returns an approximation of it. Commonly used models include linear interpolation, linear regression, and piecewise linear approximation [85, 86]. In the second step we need a function, which takes in a time series and returns the approximation error of the utilized model. For instance, the approximation error in linear regression is the sum of the squares of all the vertical differences between the best-fit line and the actual data points. In Chapter 3 we adopt the SVD model, which is not only employed for approximating (multivariate) time series segments, but also for measuring the distance between established segments.

2.7 Time Series Classification

In the terminology of machine learning [5], classification is considered as an instance of supervised learning, that is learning where a training set of correctly identified observations is available. More precisely, given a training set of observations whose category is known, classification is the problem of identifying to which category a new observation belongs. An algorithm that implements classification, using a mathematical function that maps new input data to a category, is known as a classifier. The performance of a classifier is usually evaluated by its classification error, which is formalized

in the following paragraph.

Given a set of labeled observations $\mathbb{D}$ we can split the data into a test $\mathbb{X}$ and training set $\mathbb{Y}$, for instance, by using k-fold cross validation. For each round of cross-validation, we use the training set $\mathbb{Y}$ to predict the class labels of all instances $X \in \mathbb{X}$ in the test set. The classification error is consequently defined as $e = FP/|\mathbb{X}|$, the number of false predictions FP over the size of the test set $|\mathbb{X}|$, see Algorithm 2.1. The lower the average classification error over all rounds $\frac{1}{k}\sum_{i=1}^{k} e_k$ the better.

Algorithm 2.1 Classification Error

Input: $\mathbb{X}$ ... test set; $\mathbb{Y}$ ... training set; $\mathcal{C}$ class labels
Output: e ... classification error

1: $FP \leftarrow 0$ {number of false predictions}
2: **for all** $X \in \mathbb{X}$ **do**
3: $a \leftarrow \mathcal{C}(X)$ {actual class label}
4: $p \leftarrow classify(X, \mathbb{Y}, \mathcal{C})$ {predicted class label, e.g. by 1NN classifier}
5: **if** $a \neq p$ **then**
6: $FP \leftarrow FP + 1$
7: **end if**
8: **end for**
9: $e = FP/|\mathbb{X}|$

Algorithm 2.2 1NN – One-Nearest-Neighbor Classifier

Input: X ... test instance; $\mathbb{Y}$ ... training set; $\mathcal{C}$ class labels
Output: p ... predicted class

1: $bsf \leftarrow \infty$ {best so far}
2: **for all** $Y \in \mathbb{Y}$ **do**
3: $d \leftarrow distance(X, Y)$ {e.g. Euclidean distance: $\sqrt{\sum(X-Y)^2}$ }
4: **if** $d \leq bsf$ **then**
5: $p \leftarrow \mathcal{C}(Y)$
6: $bsf \leftarrow d$
7: **end if**
8: **end for**

There is a wide range of classification algorithms that can be applied to time series, but recent work suggests that simple nearest neighbor classification is difficult to beat [36, 95]. Since the one nearest neighbor (1NN) classifier is parameter free, its performance directly reflects the effectiveness of the underlying similarity metric, allowing a fair comparison of different

time series distance measures. In Chapter 4 we employ the 1NN classifier to compare the exceptionally difficult to beat DTW distance [93, 155] with our own LTW distance [183]. Algorithm 2.2 presents the basic procedure of 1NN classification, exemplifying the embedding of the Euclidean distance.

It has been proven that the error ratio of 1NN classifier is at most twice the Bayes error ratio [37]. Although there have been attempts to classify time series with decision trees, neural networks, Bayesian networks, support vector machines etc., the best published results (by a large margin) come from simple nearest neighbor methods [211].

2.8 Time Series Clustering

In machine learning, the problem of unsupervised learning is that of trying to find hidden structure in unlabeled data [37]. Since the examples given to the learner are unlabeled, there is no error or reward to evaluate a potential solution. Approaches to unsupervised learning include clustering (e.g. k-means, mixture models, and hierarchical clustering), hidden Markov models, as well as dimensionality reduction methods (e.g. principal component analysis and singular value decomposition).

Clustering is the task of grouping a set of objects in such a way that objects in the same group are more similar to each other than to those in other groups or clusters [37]. Due to the fact that clustering is an ill-defined problem, the notion of a cluster can not be precisely defined, which is one of the reasons why there are so many clustering algorithms [43]. Furthermore, it was noted that there is no objectively 'correct' clustering algorithm, because clustering is in the eye of the beholder [43].

However, there have been several suggestions for a measure, which can be used to compare how well different clustering algorithms perform on a dataset. Internal evaluation methods, like the Davies-Bouldin and Dunn index, assign the best score to the algorithm that produces clusters with high similarity within a cluster and low similarity between clusters [39, 137]. In contrast, external evaluation methods, such as the Rand measure and F-measure, use known class labels or external benchmarks to score clustering results [46, 126].

In the context of time series, many applications use k-means clustering with Euclidean distance (ED) as (dis)similarity function and arithmetic mean as averaging method [138]. However, k-means with ED is not suitable for time series where time shifting occurs among data sequences in the same class. Although the Dynamic Time Warping (DTW) distance has been proved to give more accurate results for data sequences with local scaling invariance,

k-means clustering with DTW is not practical, because the corresponding averaging function does not return cluster centers that reflect the time series characteristics [138]. The basic intuition behind k-means (and a more general class of clustering algorithms known as iterative refinement algorithms) is shown in Table 2.1.

Table 2.1: An outline of the k-means algorithm [88].

Algorithm k-means	
1.	Decide on a value for k.
2.	Initialize the k cluster centers (randomly, if necessary).
3.	Decide the class memberships of the objects by assigning them to the nearest cluster center.
4.	Re-estimate the k cluster centers by assuming the memberships found above are correct.
5.	If none of the objects changed membership in the last iteration, exit. Otherwise, to to 3.

Table 2.2: An outline of the hierarchical clustering algorithm [88].

Algorithm Hierarchical Clustering	
1.	Calculate the distance between all objects. Store results in a distance matrix.
2.	Search through the distance matrix and find the two most similar clusters/objects.
3.	Join the two clusters/objects to produce a cluster that has now at least two objects.
4.	Update the matrix by calculating the distances between this new cluster and all other clusters.
5.	Repeat step 2 until all objects are in one cluster.

In contrast, hierarchical clustering does not depend on an averaging function, but produces a nested hierarchy of similar groups of objects (e.g. time series) according to a pairwise distance matrix [88]. The main advantages of hierarchical clustering are its great visualization power (by means of a dendrogram) and its free choice of metric (e.g. ED or DTW), making it an useful tool for time series analysis [87]. Unlike k-means, most hierarchical algorithms are deterministic and do not require us to specify the number of clusters beforehand. However, its application is limited to only small datasets due to its quadratic computational complexity. Table 2.2 outlines the basic hierarchical clustering algorithm.

2.9 Recurrence Plots

Time series analysis is often concerned with temporal data sequences that exhibit distinct recurrent behavior, e.g. rhythmic activities that appear in electrocardiogram (ECG) and electroencephalography (EEG) measurements, but also recurring states that can be found in financial and climate data. The recurrence of states is a fundamental property of deterministic dynamical systems and is typical for nonlinear and chaotic systems [128].

The analysis of a dynamical system is usually performed using the phase space, which represents all possible states of the system. Given a sequence of observations of states of a dynamical system, the phase space can be reconstructed by delay embedding, e.g using to Takens' theorem [193].

Eckmann et al. [40] have introduced *Recurrence Plots* (*RPs*) to visualize and investigate high-dimensional phase space trajectories through a two-dimensional representation. Given a system $X = \{x_1, \ldots, x_n\}$ with n states that is embedded in d-dimensional phase space, a recurrence plot of X can be mathematically expressed as:

$$R_{i,j}^{d,\epsilon} = \Theta(\epsilon - ||x_i - x_j||), \quad x_i \in \mathbb{R}^d, \quad i, j = 1 \ldots n \tag{2.6}$$

where ϵ is a threshold distance, $||\cdot||$ is a norm, and $\Theta(\cdot)$ is the Heaviside step function. According to this definition, a recurrence of a state at time i at a different time j is pictured within a two-dimensional squared matrix with black and white dots, where black dots mark a recurrence [128].

Since $R_{i,i} = 1$ (for $i = 1 \ldots n$) by definition, the RP generally exhibits a black line at the main diagonal or *line of identify (LOI)*. Although the LOI does not contain any information about the recurrent states at times i and j, the totality of all recurrence points gives information about the dynamical properties of the system under study.

In practice it is not useful and largely impossible to find complete recurrences in the sense $x_i \equiv x_j$, because a state of a chaotic system would not recur exactly to the initial state but approaches the initial state arbitrarily close [128]. Therefore, a recurrence is defined as a state x_j that is sufficiently close to x_i in consideration of the ϵ-threshold. Related literature have suggested to choose the ϵ-threshold so that it does not exceed 10% of the maximum phase space diameter [218] and the recurrence point density is approximately 1% [219].

In order to compute an RP, a norm has to be chosen [128]. The most widely used norms are the L^1-norm (that corresponds to the *Manhattan* distance), the L^2-norm (*Euclidean* norm) and the L^∞-norm (*Supremum* norm). Assuming a fixed similarity threshold ϵ, the amount of recurring states grows

with increasing p (referring to L^p-norm) as the neighborhood of the phase space is getting bigger. The L^∞-norm is most commonly applied, because it is independent of the phase space dimension and easier to compute than any other norm [128].

The initial purpose of RPs was the visual inspection of higher dimensional phase space trajectories [128]. In general, RPs exhibit characteristic large scale and small scale patterns [40]. The large scale patterns *typology* offers a global impression which can be characterized as *homogeneous*, *periodic*, *drift* and *disrupted*. The closer inspection reveals small scale structures *texture* which are *single dots*, *diagonal lines* as well as *vertical* and *horizontal lines* [128, 132]. These small scale structures are the base of a quantitative analysis of the RPs, which is discussed at full length in the following section.

2.10 Recurrence Quantification Analysis

In order to go beyond the visual impressions yielded by *RPs*, several measures of complexity which quantify the above mentioned small scale structures have been introduced and are (collectively) known as *Recurrence Quantification Analysis (RQA)* measures [205, 218]. Commonly used *RQA* measures include *recurrence rate*, *determinism*, *divergence*, *entropy* as well as *trend*; each of which uses the recurrence point density and/or line structures found in a recurrence plot [128, 129, 132].

A computation of these RQA measures in small windows (sub-matrices) of the RP moving along the LOI yields the time-dependent behavior of these variables [128]. Recent studies have shown that these time-dependent variations of RQA measures can be used to investigate changing dynamics, such as chaos-order and chaos-chaos transitions [133].

According to the *RP* definition in Equation 2.6, the *recurrence rate (RR)* can be formalized as:

$$RR = \frac{1}{n} \sum_{i,j=1}^{n} R_{i,j}^{d,\epsilon}, \quad i,j = 1 \dots n \tag{2.7}$$

where $n \times n$ is the size of the recurrence matrix. The RR measures the density of the recurrence points by simply counting the non-zero entries (or black dots) in the RP [128]. In the limit this measure becomes the probability that a state will recur.

Other measures consider the diagonal lines. The frequency distribution (or histogram) of the lengths l of the diagonal structures in the RP for a certain ϵ-threshold is $P^\epsilon(l) = \{l_i; i = 1 \dots N_l\}$, where N_l is the number of

diagonal lines with length l (each line is counted only once) [128]. In general, processes with stochastic behavior cause none or short diagonals, whereas deterministic processes cause longer diagonals and less isolated recurrence points. Therefore, the ratio of recurrence points that form diagonal structures of minimum length l_{min} to all recurrence points is introduced as a measure for the *determinism* (or predictability) of the system under study [128].

$$DET = \frac{\sum_{l=l_{min}}^{n} l \cdot P^{\epsilon}(l)}{\sum_{i,j=1}^{n} R_{i,j}^{d,\epsilon}} \tag{2.8}$$

The threshold l_{min} allows to exclude the diagonal lines formed by tangential motion of the phase space trajectory, but too large l_{min} can worsen the reliability of the DET measure [128]. When computing the DET measure we usually specify a Theiler window [194], which excludes the recurrence points in a small corridor around the LOI that merely correspond to the tangential motion of the phase space trajectory.

Having defined the RR and DET measure, one can easily derive another RQA measure which expresses the relationship between both. The so-called $RATIO$ can be computed from the frequency distribution of the diagonal line lengths [128].

$$RATIO = n^2 \frac{\sum_{l=l_{min}}^{n} l \cdot P^{\epsilon}(l)}{\left(\sum_{l=1}^{n} l \cdot P^{\epsilon}(l)\right)^2} \tag{2.9}$$

Heuristic studies have revealed that the $RATIO$ can be used to discover transitions, where the RR decreases but the DET does not change [205].

In general, the (diagonal and vertical) line structures found in RPs are used to define various other RQA measures, including *divergence* (the inverse of the maximal length of diagonal lines), the *entropy* (complexity of deterministic structure in the system) and many more. Since RQA is a broad research area that simply goes beyond the scope of this thesis, we refer the interested reader to the relevant literature [128, 129, 130, 132, 133].

Chapter 3

Factorization-based Distance
for Time Series Segmentation

Nowadays researchers in many domains are faced with high dimensional, temporally variable data, such as coming from sensors, which is represented as a sequence of observations. A common problem in the data mining and machine learning community is the recognition of recurring patterns within such time series or data streams.

This work aims at developing and investigating efficient methods for the recognition of contextual patterns in multivariate time series from different application domains based on machine learning techniques. To this end, we propose a generic three-step approach that involves (1) feature extraction to build robust learning models based on significant time series characteristics, (2) segmentation to identify internally homogeneous time intervals and change points, as well as (3) clustering and/or classification to group the time series (segments) into the sub-population to which they belong to.

To support our proposed approach, we present and discuss experiments on real-life vehicular sensor data. Furthermore we describe a number of applications, where pattern recognition in multivariate time series is practical or rather necessary.

3.1 Introduction

Due to the fact that most electronic components have gotten substantially smaller and cheaper to produce, an increasing number of everyday electronic devices are equipped with smart sensors. Recent mobile phones have integrated location and acceleration sensors [22], modern vehicles are able to measure ambient temperature and average fuel consumption [182], and

twenty-first-century smart homes monitor user activities as well as consumption of resources [163]. There is a significant trend in using these integrated sensors to simplify human-computer-interaction by means of automatic adaptation of user interfaces or even machine behavior according to the recognized context [53, 66, 139].

According to the *Oxford Dictionary* the term "context" is defined as "the circumstances that form the setting for an event, statement, or idea, and in terms of which it can be fully understood and assessed". Although this term has been used in many ways in different areas of computer science [22, 53], we only focus on the context that can be discovered from sensor data, which are basically patterns that represent rare events (anomalies) or recurring situations.

Recently, there has been done a lot of research in the field of context-aware systems, ranging from adaptive applications for (mobile) electronic devices and ubiquitous applications in smart home environments to advanced driver assistance systems (ADAS) and event detection in video sequences. Context-aware applications take advantage of environmental characteristics to adapt to user needs without consuming user attention [22, 53, 139]. Smart environments make it possible to build ubiquitous applications that assist users during their everyday life, at any time, in any context [109, 163, 172]. Leading car manufacturers like the *Volkswagen AG* aim to identify characteristic drive(r) profiles from abnormal and recurring context (i.e. motor behavior and drive maneuvers) by analyzing vehicular sensor data recorded during car drives [182]. Media and entertainment companies like *Technicolor* push violent scene detection in videos [7, 55, 56] to help users to choose movies that are suitable for their children of different age; where corresponding context or rather events are discovered from audio and video features represented as changing signals equal to sensor data.

All these application examples involve large amounts of high dimensional data with significant between-sensor/signal correlations that are corrupted by non-uniform noise. To deal with these problems, we adopt a machine learning approach for detecting contextual patterns in multivariate time series [182]. With regard to sensor data, the term multivariate implies the exploration of multiple signals in parallel. We especially focus on multivariate time series analysis, because almost all real-life applications employ multiple sensors to retrieve contextual patterns, which describe environmental situations or system states.

Our research goal follows a dual agenda: First, we develop efficient methods for detecting contextual patterns in multivariate time series based on state-of-the-art and recent machine learning techniques. Second, we focus on developing methods that are not tailored to solutions of specific application

areas, but that will be applicable to other domains as well. To achieve this, we suggest a generic approach for contextual pattern detection in sensor data consisting of the following three steps: feature extraction, segmentation and clustering/classification.

Since multi-sensor data is often too complex and highly redundant [86], we transform the raw input data into a reduced representation in the feature extraction step. The challenge consists in extracting features in such a way that the reduced data contains all or most of the relevant information [121] for the underlying task of detecting contextual patterns. Based on the selected signal features we can partition a time series into segments or situations [1, 2], which might be classified or clustered to high-level context in a second step. This work introduces straightforward signal processing features as well as more sophisticated feature extraction algorithms to measure similarity between two time series (segments).

Given a feature representation of the input data, segmentation aims at partitioning the time series into internally homogeneous intervals of constant correlation. Our underlying assumptions are that contextual patterns arise from correlations of the given sensor data [182].

With regard to sensor signals, context or events are usually described as segments of multivariate time series, which are internally homogeneous. In the domain of time series analysis the term homogeneous refers to an unchanging correlation between the observed variables over a specified period of time. Due to the fact that most situations have different length and are often overlapping, it is a non-trivial task to determine clear borders between individual time series segments [1, 2].

Depending on the application and the examined dataset, we either want to group the retrieved time series segments into clusters of similar objects or we aim to classify the identified contextual patterns according to predefined labels. Usually, researchers apply machine learning techniques for automatic segmentation and clustering/classification of time series [1, 2, 7, 9, 55, 56, 86, 103, 159, 182], because it is not visible to the naked eye at which point in time a specific contextual pattern starts and ends, or even how similar two events are to each other; especially if we consider multiple signals at the same time (see Figure 3.1).

We applied the proposed three-step approach to the problem of recognizing recurring contextual patterns in vehicular sensor data [182] using a method proposed by Abonyi [2]. The results justify that PCA-based time series segmentation and hierarchical clustering according straight-forward signal processing features enables us to identify complex drive maneuvers, such as stop and go at traffic lights or overland drives, which consist of multiple consecutive time series segments.

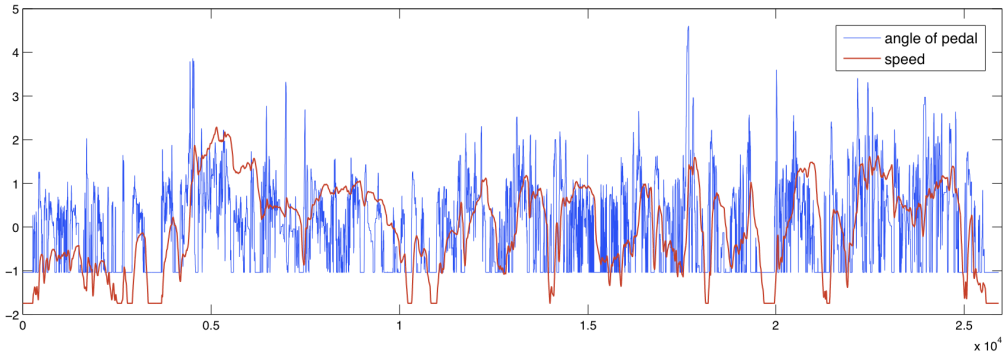


Figure 3.1: Progression of speed and accelerator signal during a car drive. In order to compare both signals against each other the time series data was z-normalized (the unnormalized speed signal ranges from 0 to 130 km/h, and the original accelerator measurements vary between 0 to 100 degrees).

We expect that machine learning methods may have a strong influence on detection of contextual patterns in multivariate time series analysis such that it may serve as a competitive alternative to existing state-of-the-art techniques [1, 2, 9, 103, 159]. Therefore, we want to gain insight in how selected machine learning methods cooperate in a multi-step approach to achieve a common goal, which in our case is restricted to detection of contextual patterns in multivariate time series. The expected result of this study will be a on-the-fly segmentation and classification algorithm that can be used to simultaneously identify and classify contextual patterns in multivariate time series.

Figure 3.1 shows the progression of speed and accelerator signal during a car drive and illustrates the necessity of machine learning techniques for automatic recognition and classification of time series segments or situations respectively. Although speed and accelerator signal are highly correlated, it is not visible to the naked eye at which point in time a specific situation starts or ends (e.g. stop and go at a traffic light). Due to the fact that we normally consider multiple signals at the same time, it is advisable to apply machine learning techniques for automatic recognition and classification of recurring patterns [86].

3.2 Problem Statement

“Intelligence is the ability to adapt to change” - *Stephen Hawking*. With regard to machine intelligence, one could argue that a machine needs to be aware of its operating context to adapt to change and to be considered as

intelligent. Most intelligent systems employ smart sensors to infer context, which can be used to adapt to the (environmental or system) changes at hand. Our goal is to recognize contextual patterns in multivariate time series, which can be used as an information source for intelligent systems to adapt to their ever-changing environment.

For further problem discussion we represent sensor data as a multivariate time series $X = \{x_1, \ldots, x_n\}$ with $x_i \in \mathbb{R}^d$ for $1 \leq i \leq n$, where d is the number of measurements and n is the number of discrete time points at which the measurements are taken. A contextual pattern (i.e. event or situation) of X is a tuple $T = (S, C)$ consisting of a segment $S = \{x_i, \ldots x_{i+l-1}\}$ of l consecutive columns of X together with a class label C from some set $\mathbb{C} = \{C_1, \ldots, C_k\}$. A segment S describes the characteristic features of a contextual pattern and determines its period. The class label C describes the nature of the contextual pattern. Detecting contextual patterns amounts in finding a column-wise segmentation $\mathbb{S} = \{S_1, \ldots, S_s\}$ of the matrix X together with a labeling function $c : \mathbb{S} \to \mathbb{C}, \quad S_i \mapsto c(S_i)$ that assigns a class label to each segment. Here, we demand that a segmentation of X consists of a nonempty set of consecutive columns that form a partition of the column set of X, e.g. $\mathbb{S} = \{S_1, S_2\} = \{\{x_1, \ldots, x_i\}, \{x_{i+1}, \ldots, x_n\}\}$.

It is important to note that $\mathbb{C}$ can be unknown and therefore constitutes a part of the learning problem. Thus, detecting contextual patterns in multivariate time series amounts in learning both, the data (segments) we want to classify as well as the class labels (contexts) we want to assign to the data. This is in contrast to traditional machine learning, where the class labels as well as the data we want to learn on are usually given.

3.3 Three-Step Approach

The proposed three-step approach for contextual pattern recognition in multivariate time series involves feature extraction, segmentation and clustering/classification. Each of the three sub-tasks can be fulfilled by different techniques, were some of them are discussed in the following sections. This survey also includes own contributions which we scrutinize in the subsequent case study (see Chapter 3.4).

3.3.1 Feature Extraction

A time series is a collection of observations made sequentially in time. People measure things, and things (with rare exception) change. In most cases, patterns, and not individual points, are of interest [86]. Time series collec-

tions are often huge and mining them requires a high-level representation of the data. Due to the fact that time series analysis aims to be time and space efficient, data mining researchers usually extract significant features or characteristics which given an approximation of the original data.

Feature extraction involves simplifying the amount of resources required to describe a large set of data accurately. When performing analysis of complex data one of the major problems stems from the number of variables involved. Feature extraction is a general term for methods of constructing combinations of the variables to get around these problems while still describing the data with sufficient accuracy [79].

In machine learning and statistics, feature selection is the technique of selecting a subset of relevant characteristics for building robust learning models. From a theoretical perspective, it can be shown that optimal feature selection for (un)-supervised learning problems requires an exhaustive search of all possible subsets of features [121]. For practical learning algorithms, the search is for a satisfactory set of features instead of an optimal set.

In the following we discuss several well-known signal processing features that are employed to describe time series. Furthermore we go into different similarity measures that are used to discriminate time series, which is necessary for clustering and classification.

Fourier Transform (FT) is on of the most popular signal processing features and a measure of how much of each frequency is existing in the signal [105]. If the time domain of a signal is given and the frequency amplitude is searched then the Fourier Transform is used. In general, FT is only valid for periodical signals. A special case of the continuous FT is the Discrete Fourier Transform (DFT), which requires a discrete periodical spectrum defined on a certain area. Therefore a sample rate needs to be defined.

Sometimes it is necessary to have both time and frequency information simultaneously, so a time localization of the spectral components is needed. Discrete Wavelet Transform (DWT) is a technique that decomposes a signal into several groups (vectors) of coefficients. Different coefficient vectors contain information about characteristics of the sequence at different scales [59].

Mel Frequency Cepstral Coefficients (MFCCs) are the dominant feature in speech recognition [122]. MFCCs are short-term spectral features that are calculated in several steps. First the signal is divided into frames. For each frame, the algorithm obtains the amplitude spectrum, which in turn is converted to Mel (a perceptually-based) spectrum, so called Mel scale. This transformation emphasizes lower frequencies, which are perceptually more meaningful for speech. It is possible however that the Mel scale may not be optimal for music as there may be more information in say higher frequencies.

Finally the algorithm takes the discrete cosine transform (DCT) to decorrelate the components of the feature vectors. MFCC's might be relevant for event detection in video sequences as well.

Principal Component Analysis (PCA) is a matrix factorization technique, which is used to reduce the dimensionality of the input space and/or to retrieve latent relations between variables of the observed dataset. PCA can be found in many applications, such as recommender systems [101, 169, 186], semantic analysis of textual information [38] and link prediction in complex networks [3, 113, 181]. Furthermore, PCA is a common feature extraction method in signal processing, which is employed for time series segmentation and clustering [1, 2, 182]. The sensor fusion algorithm developed by Abonyi et al. [1, 2] is able to segment a multivariate time series in a bottom-up manner. Adjacent segments are merged iteratively if their combined model does not exceed a predefined reconstruction error, which is computed by means of principal component analysis. One major advantage of the proposed segmentation algorithm is that it allows one to reuse the PCA model to define a distance measure to compare multivariate time series segments for clustering or classification [2, 175]. Note that PCA-based algorithms are able to detect changes in the mean, variance and correlation structure among multiple variables or signals [182].

In the context of pattern recognition in multivariate time series, feature extraction can be considered as a preprocessing step for further data mining and machine learning algorithms. However, for the clustering and classification task it is necessary to define a distance measure to compare time series (segments). Usually the established feature vectors are compared by means of cosine similarity or other metrics that conform to the euclidean space. Apart from that, there exist distance measures that work on the raw sensor data or time series measurements.

The Dynamic Time Warping (DTW) distance measure is a technique that has long been known in speech recognition community [92]. It allows a non-linear mapping of one signal to another by minimizing the distance between the two. A decade ago, DTW was introduced to the data mining community as a utility for various tasks for time series problems including clustering, classification, and anomaly detection [13]. Although, DTW is superior to Euclidean Distance (ED) for clustering and classification of time series, most research has utilized ED because it is more efficiently calculated [159]. Lower bounding (LB) that greatly mitigates DTWs demanding CPU time has sparked a flurry of research activity. However, LB and its applications still only allow DTW to be applied to moderate large datasets. In addition, almost all research on DTW has focused on speeding up calculations; there has been little work done on improving its accuracy.

3.3.2 Time Series Segmentation

Segmentation is a frequently used subroutine in clustering and classification of time series. It is used to locate stable periods of time or alternatively to identify change points [2]. There already exist different heuristic approaches to extract internally homogeneous segments. Some primitive algorithms search for inflection points to locate episodes [191]. Other algorithms determine segment borders by the Sliding Window technique, where a segment is grown until it exceeds some error bound [86].

In this work [182] we propose a bottom-up segmentation approach, which begins creating a fine approximation of the time series, and iteratively merges the lowest cost pair of segments until some stopping criteria is met. The cost of merging two adjacent segments is evaluated by the Singular Value Decomposition (SVD) model of the new segment. SVD-based algorithms are able to detect changes in the mean, variance and correlation structure among several variables [181, 182, 186]. The proposed approach can be considered as an extension of the sensor fusion algorithm developed by Abonyi [1, 2].

For the purpose of finding internally homogeneous segments from a given time series, we need to formalize a cost function for the individual time intervals. In most cases, the cost function is based on the distance between the actual values of the time series and a simple function (linear function, or polynomial of higher degree) fitted to the data of each segment [2]. Since our approach aims at detecting changes in the correlation structure among several variables, the cost function of the segmentation is based on the Singular Value Decomposition of the segment matrices, where each row is an observation, and each column is a variable. Before applying SVD, the observed variables need to be centered and scaled, in such a way as to make them comparable. Hence, we compute the z-scores using mean and standard deviation along each individual variable of the time series.

The SVD model projects the correlated high-dimensional data onto a lower-dimensional hyperplane, which is useful for the analysis of multivariate data. The distance of the original data from the hyperplane is a significant indicator for anomalies or unsteadiness in the progression of the observed variables. Hence, it is useful to analyze the reconstruction error of the SVD model to get information about the homogeneity of the factorized time series segments. In the proposed approach the reconstruction error is determined by the Q-measure [2, 182], which is commonly used for the monitoring of multivariate systems and for the exploration of the errors. The crux of the proposed time series segmentation is to use the Q-measure as an indicator for the homogeneity of individual or rather merged segments.

The sensor fusion algorithm developed by Abonyi [2] merges segments

bottom-up, whereas the initial fine-grain segmentation of the time series is determined manually. In case of a periodic signal and an odd initial segmentation it is highly probable that the bottom-up algorithm will not find recurring segments, because the time series was partitioned into internal inhomogeneous time intervals. Therefore, we propose to perform an initial fine-grain segmentation of the time series according to the critical points of the individual signals. Critical points of great interest are extrema as well as inflection and saddle points [191]. In order to determine the critical points of a curve or signal we need to calculate the first, second or third derivative respectively.

Due to the fact that sensors often produce extremely noisy and fluctuate signals, it is very likely that the examined time series exhibit dozens of critical points. Hence, we need to smooth the signal with a low-pass or base-band-pass filter that reduces the number of critical points, but does not substantially change the coordinates of the segmentation borders. We propose to employ the Savitzky-Golay filter [147], which is typically used to smooth out a noisy signal whose frequency span is large. Savitzky-Golay smoothing filters perform much better than standard averaging FIR (Finite Impulse Respond) filters, which tend to filter out a significant portion of the signal's high frequency content along with the noise. Although Savitzky-Golay filters are more effective at preserving the pertinent high frequency components of the signal, they are less successful than standard averaging FIR filters at rejecting noise. Savitzky-Golay filters are optimal in the sense that they minimize the least-squares error in fitting a higher-order polynomial to frames of noisy data [147]. The introduced critical point approach can be employed as a preprocessing step for the proposed bottom-up segmentation, but can also be considered as a segmentation technique by its own.

3.3.3 Clustering and Classification

Depending on the way of looking at the stated pattern recognition problem we either want to classify or cluster the time series (segments). Classification is a supervised learning technique, which employs labeled training data to gradually adjust model parameters until the classifier fits the presented data. The trained model can then be employed to predict labels for previously unseen input or time series (segments) respectively. In contrast, clustering is a method of unsupervised learning, which is used to group similar objects without training the mathematical model. Cluster analysis gives information about the underlying structure of the examined data and can be employed to identify recurring time series segments that fall into the same group and

unique time intervals that are represented as outliers.

Hierarchical clustering methods find successive groups using previously established ones. Agglomerative hierarchical clustering algorithms begin with each element as a separate cluster and merge them into successively larger clusters. Merges and splits are determined in a greedy manner. The results of hierarchical clustering are usually presented as a dendrogram. Hierarchical clustering has the distinct advantage that any valid measure of distance can be used. In fact, the observations themselves are not required; all that is used is a matrix of distances.

The choice of an appropriate metric does influence the shape of the clusters, as some elements might be close to each other according to one distance and farther away according to another. Some commonly used metrics for hierarchical clustering are Euclidean distance, cosine similarity, Mahalanobis distance (covariance matrix) and Pearson's correlation. For text or other non-numeric data, metrics such as the Hamming distance or Levenshtein distance are often used. Given an appropriate metric, an agglomerative hierarchical cluster tree can be created by several different methods, which differ from one another in how they measure the distance between the clusters. Available methods are single and complete linkage, also known as shortest and furthest distance, as well as average, centroid and inner-squared distance.

In the following we want to give a brief introduction to several popular classification techniques, which can be used to categorize time series (segments). The k-nearest neighbor (k-NN) algorithm [31] is amongst the simplest of all machine learning approaches. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common amongst its k-nearest neighbors. If $k = 1$, then the object is simply assigned to the class of its nearest neighbor (also known as 1-NN classifier).

Another popular classification technique in the machine learning community are decision trees, which are tree-like graphs of decisions and their possible consequences [153]. In data mining, trees can be described as the combination of mathematical and computational techniques to aid the description, categorization and generalization of a given set of data. They are used as a predictive model which maps observations about an item to conclusions about the item's target value. A tree can be learned by splitting the source set into subsets based on an attribute value test.

In addition, there also exist probabilistic models like Bayesian networks, which represent a set of random variables and their conditional dependencies via a directed acyclic graph [77]. For example, a Bayesian network could represent the probabilistic relationships between complex situations and corresponding sensor data. One advantage of Bayesian networks is that it is intuitively easier for a human to understand direct dependencies and local

distributions than complete joint distribution. Furthermore, Bayesian Networks can save considerable amount of memory compared to naive matrix based storing of conditional dependencies, if the dependencies in the joint distribution are sparse.

Recently, there has been done a lot of research in the field of kernel methods, especially Support Vector Machines (SVM) [82, 151]. Given a set of training samples, each labeled as belonging to one of two categories, a SVM training algorithm builds a model that predicts whether a new sample falls into one category or the other. Intuitively, a SVM model is a representation of the samples as points in space, mapped so that the samples of the separate categories are divided by a clear margin that is as wide as possible. New samples are then mapped into that same space and predicted to belong to a category based on which side of the margin they fall. More formally, a SVM constructs a hyperplane or set of hyperplanes in a high or infinite dimensional space, which can be used for classification, regression or other tasks. In general, the effectiveness of a SVM lies in the selection of the soft margin parameters and the kernel, which is a weighted function, used in non-parametric estimation techniques.

3.4 Case Study

The above survey presents several possible ways how to implement our proposed three-step approach for contextual pattern recognition in multivariate time series. In the following we describe how to apply the presented three-step approach in practice.

This case study exemplifies the recognition of contextual patterns, or rather situations, in vehicular sensor data recorded during car drive. To identify recurring situations we suggest to group the identified segments according their similarity. Although the proposed time series segmentation and segment clustering is discussed in terms of vehicular sensor data, it is also applicable to time series of other domains.

3.4.1 Bottom-Up Segmentation

The bottom-up time series segmentation, described below, is mainly inspired by the sensor fusion algorithm developed by Abonyi [2]. However, we extended the original algorithm by several features, such as automatic determination of model rank and merge threshold, as well as initial fine-grain segmentation according to critical points of individual signals [147]. Furthermore, we evaluate the extended segmentation algorithm (Chapter 3.4.3) on

both synthetic time series data and real-life multivariate sensor signals which comprise information about the progression of motor parameters during car drive. In case of motor signals, a time series segment may represent a critical situation that appeared during acceleration or loss of speed.

Our approach employs Singular Value Decomposition (SVD) [2, 181, 186] to find segment borders at those points in time where the correlation structure of the observed variables fulfill a significant change. We rely on the fact that SVD gives a close approximation of our time series data, whenever the observed variables exhibit a linear correlation. In other words, the SVD model is expected to give a high reconstruction error, if and only if the decomposed time series data fulfills a significant change in correlation structure [2]. We suggest a bottom-up segmentation algorithm that is based on the assumption that two adjacent time series segments can only be merged in case of a low reconstruction error, implying that the correlation structure among the observed variables does not considerably change within the two contiguous segments. Figure 3.2 exemplifies the segmentation of a multivariate time series via singular value decomposition.

The adopted bottom-up approach initially establishes a fine-grained segmentation of the time series, and iteratively merges the lowest cost pair of segments until some stopping criteria is met [86]. The cost of merging two adjacent segments is evaluated by the Singular Value Decomposition (SVD) model of the newly formed segment. SVD-based algorithms are able to detect changes in the mean, variance and correlation structure among several variables. In the following, we describe the mathematical foundations of time series segmentation in more detail.

For further problem discussion we represent a multivariate time series as $X = \{x_i, \ldots, x_n\}$ with $x_i \in \mathbb{R}^d$ for $1 \leq i \leq n$, where d is the number of measurements and n is the number of discrete time points. Consequently, a segment of X is defined as a set of consecutive time points $S(a, b) = \{x_a, x_{a+1}, \ldots, x_{b-1}, x_b\}$ or rather columns, with $1 \leq a < b \leq n$. The segmentation of time series X into s non-overlapping time intervals can thus be formulated as $\mathbb{S} = \{S_e(a_e, b_e) | 1 \leq e \leq s\}$, where $a_1 = 1$, $b_s = n$ and $a_e = b_{e-1} + 1$ for $1 \leq e \leq s$.

For the purpose of finding internally homogeneous segments from a given time series, we need to formalize the cost function for the individual time intervals. In most cases, the cost function $cost(S(a, b))$ is based on the distance between the actual values of the time series and a simple function (linear function, or polynomial of higher degree) fitted to the data of each segment [2]. Since our approach aims at detecting changes in the correlation structure among several variables, the cost function of the segmentation is based on the Singular Value Decomposition of the segment matrices, where

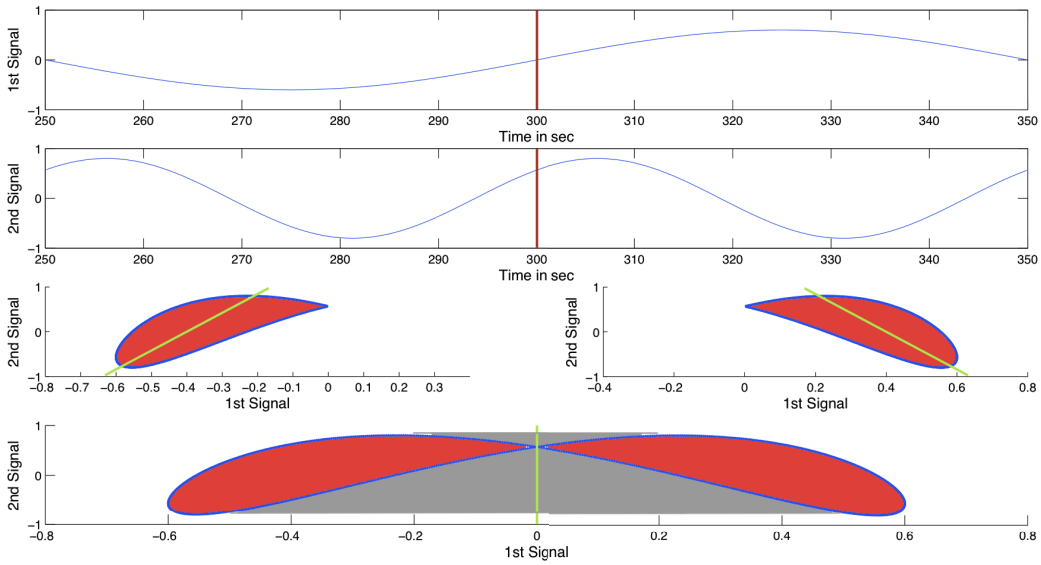


Figure 3.2: Segmentation of Multivariate Time Series via Singular Value Decomposition. The first two plots illustrate the progression of two synthetic signals, which are linearly independent. Assuming that the time series was initially segmented at predetermined inflection points of the first signal (segment borders are highlighted in red color), we are faced with the decision whether to merge the depicted segments or not. Our approach relies on the fact that we only merge contiguous segments, if the correlation structure among the signals does **not** fulfill a significant change. Meaning that the reconstruction error of the employed SVD model does not exceed a predefined threshold. In case of two observed parameters the correlation structure of a segment can be expressed as a position vector (two-dimensional hyperplane), which gives an approximation of the original segment data. The scatter plots of the individual segments show the distribution of the data points (marked in blue color) together with the largest singular-vector (highlighted in green color), which was determined via singular value decomposition and describes the correlation structure among the examined signals. Furthermore, the area highlighted in red color can be considered as the reconstruction error, which is the distance between the original segment data and the approximation given by singular value decomposition. Note that the scatter plots of the individual segments illustrate the data points situated in time interval [250,300] and [300,350] respectively. In the last plot, we can see that the reconstruction error grows if we merge the examined segments. Beside the error of the individual segments (red color), we get an additional error highlighted in gray color. The growth of the reconstruction error accounts for the fact that the SVD model gives a less precise approximation of the merged segments, implying a significant change in correlation structure among the examined signals. Hence our model would not merge the segments.

each row is an observation, and each column is a variable. Before applying SVD, the observed variables need to be centered and scaled, in such a way as to make them comparable. Hence, we compute the z-scores using mean and standard deviation along each individual variable of the time series.

The singular value decomposition of a segment matrix X can be described as an factorization $X_{d\times n} \approx U_{d\times n}\Sigma_{r\times r}V^T_{r\times n}$ into a matrix Σ which includes the singular values of X in its diagonal in decreasing order, and into the matrices U and V which include the corresponding left-singular and right-singular vectors. With the use of the first few r singular values and the corresponding singular vectors, the SVD model projects the correlated high-dimensional data onto a hyperplane which is useful for the analysis of multivariate data. When the SVD model has an adequate number of dimensions r, the distance of the original data from the hyperplane is a significant indicator for anomalies or unsteadiness in the progression of the observed variables. Hence, it is useful to analyze the reconstruction error of the SVD model to get information about the homogeneity of the factorized time series segments. In the proposed approach the reconstruction error is determined by the Q-measure [2], which is commonly used for the monitoring of multivariate systems and for the exploration of the errors. The Q-measure is defined as the mean squared deviation between the original segment data X and its projection P, which is an approximation given by the Singular Value Decomposition of rank r. Due to the fact that we are mainly interested in the correlation structure among the observed variables, the projection P just considers the singular-vectors given by factor matrix V and neglect the hidden structure among the observations held in factor matrix U. The crux of the proposed time series segmentation is to use the Q-measure as an indicator for the homogeneity of individual or rather merged segments. Equation 2.1 - 2.3 formulate the derivation of the cost function:

$$P_{d\times n} = X_{d\times n}V_{n\times r}V^T_{r\times n} \tag{3.1}$$

$$Q = \frac{1}{d\cdot n}\sum_{i=1}^{d}\sum_{j=1}^{n}(X_{i,j} - P_{i,j})^2 \tag{3.2}$$

$$cost_Q(S_e(a_e, b_e)) = \frac{1}{b_e - a_e + 1}\sum_{j=a_e}^{b_e} Q_j \tag{3.3}$$

The introduced bottom-up segmentation approach iteratively merges the cheapest pair of adjacent segments until a predetermined number of segments are reached or a specific threshold is exceeded. By the time the proposed

bottom-up algorithm terminates we expect that optimal time series segmentation was found. Figure 3.2 illustrates the growth of the reconstruction error, when two contiguous segments are merged. One possible way how to determine an appropriate merge threshold will be discussed in Chapter 3.4.3 (also refer to Figure 3.5).

The accuracy of the SVD model strongly depends on the rank of the decomposition. However, in most cases it is hard to determine the optimal rank in terms of model accuracy and computational complexity. Abonyi et al. [2] suggest to infer an appropriate rank from the scree-plot of the eigenvalues. Since this method is very imprecise and only works for variables with the same scale, we suggest to choose the model rank according the desired accuracy. In other words, we consider the r-largest singular values of matrix $\Sigma_{n\times n}$ that hold the required energy $[E_k = (sum_{i=1}^{r}\Sigma_{i,i})/(sum_{i=1}^{n}\Sigma_{i,i})]$. Note, that the calculation of the required model rank is based on the size of the final coarse segmentation. This derives from the fact that the approximation of large segments usually requires a high model rank, which most certainly also fits smaller segments.

The sensor fusion algorithm developed by Abonyi [2] merges segments bottom-up, whereas the initial fine-grain segmentation of the time series is determined manually. In case of a periodic signal and an odd initial segmentation it is highly probable that the bottom-up algorithm will not find recurring segments, because the time series was partitioned into internal inhomogeneous time intervals. Therefore, we propose to perform the initial fine-grain segmentation of a time series according to the critical points of the individual signals. Critical points of great interest are extrema as well as inflection and saddle points. In order to calculate the critical points of a curve or signal we need to calculate the first, second or third derivative respectively.

Due to the fact that sensors often produce extremely noisy and fluctuate signals, it is very likely that the examined time series exhibit dozens of critical points. Hence, we need to smooth the signal with a low-pass or base-band-pass filter that reduces the number of critical points, but does not substantially change the coordinates of the retrieved segmentation borders (refer to Figure 3.3.

In our critical point (CP) approach, we employ the Savitzky-Golay filter, which is typically used to smooth out a noisy signal whose frequency span is large. Savitzky-Golay smoothing filters perform much better than standard averaging FIR (Finite Impulse Respond) filters, which tend to filter out a significant portion of the signal's high frequency content along with the noise. Although Savitzky-Golay filters are more effective at preserving the pertinent high frequency components of the signal, they are less successful

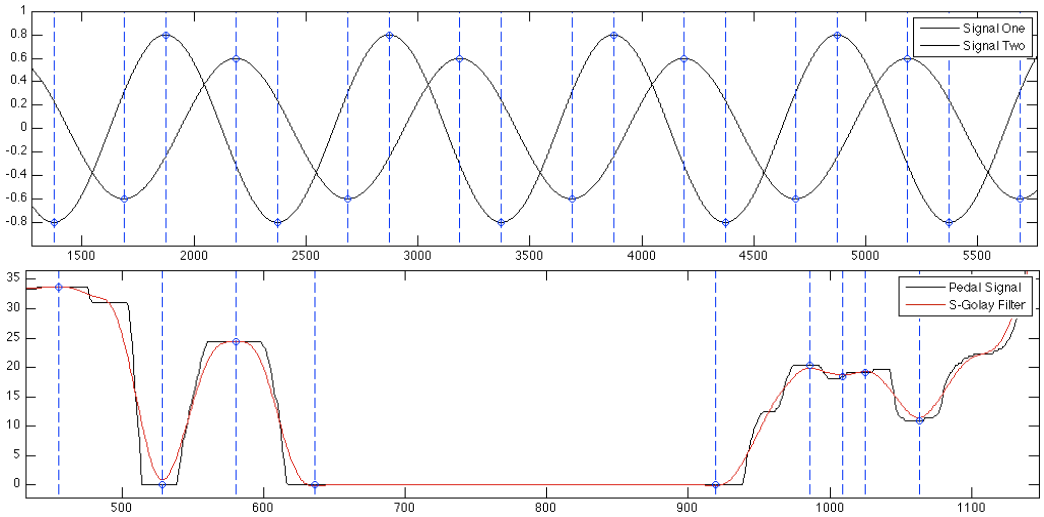


Figure 3.3: Time Series Segmentation based on Critical Points of Synthetic and Real-Life Signals. The first plot shows two synthetic signals, which are segmented according to their local extrema. Here the blue dots illustrate extrema and the blue dotted lines depict segment borders. In contrast to the common bottom-up segmentation approach, the critical point (CP) algorithm is able to find recurring time intervals for initial segmentation in periodic signals. Consequently all merged segments will also contain recurring subpatterns, which is absolutely necessary for segment classification or clustering. As illustrated in the second plot the critical point algorithm also gives satisfying results for real-life data. The plot shows the progression of the accelerator during a car drive as unvarnished and idealized signal (function plotted in black and red color respectively). For the smoothing of the accelerator signal we used the Savitzky-Golay filter with a second-order polynomial and a frame size of 30 data points. To eliminate residual noise we furthermore applied a moving average low-pass filter with the same frame size. For most signals the frame size has to be adjusted according the degree of noise and variance.

than standard averaging FIR filters at rejecting noise. Savitzky-Golay filters are optimal in the sense that they minimize the least-squares error in fitting a higher-order polynomial to frames of noisy data [147].

The introduced critical point (CP) approach can be employed as a preprocessing step for the proposed bottom-up segmentation (BU), but can also be considered as a segmentation technique by its own. In Section 4 we evaluate the mixed approach (BUCP) as well as both segmentation techniques separately.

3.4.2 Distance-based Segment Clustering

One major advantage of the proposed segmentation algorithm is that it allows one to reuse the SVD model to define a distance measure to compare multivariate time series segments [17,18]. Consider two segments S_i and S_j with data matrices X_i and X_j that hold the same d variables, and whose singular value decomposition transforms the original data into a r-dimensional subspace, denoted by $U_i\Sigma_i V_i^T$ and $U_j\Sigma_j V_j^T$ respectively. Then the similarity between these subspaces can be defined as the sum of the squares of the cosines of the angles between each vector of matrix W_i and W_j, which are composed by multiplying the respective singular values and right-singular vectors ($W_i = \Sigma_i V_i$ and $W_j = \Sigma_j V_j$). This similarity measure is formalized by the following equation:

$$d_{SVD}(X_i, X_j) = \frac{1}{r} trace((W_i^T W_j)(W_j^T W_i)) \tag{3.4}$$

Due to the fact that subspaces W_i and W_j contain the r most important singular vectors that account for most of the variance in their corresponding dataset, d_{SVD} is furthermore a measure of similarity between the segments S_i and S_j or their matrices X_i and X_j respectively.

Cluster analysis is the assignment of a set of observations into subsets (called clusters) so that observations in the same cluster are similar in some sense. It is a method of unsupervised learning, and a common technique for statistical data analysis used in fields like machine learning, data mining and information retrieval. Hierarchical methods find successive clusters using previously established ones. Agglomerative hierarchical clustering algorithms begin with each element as a separate cluster and merge them into successively larger clusters. The merges and splits are determined in a greedy manner. The results of hierarchical clustering are usually presented as a dendrogram. Figure 3.4 illustrates the dendrogram for a sample of time series segments.

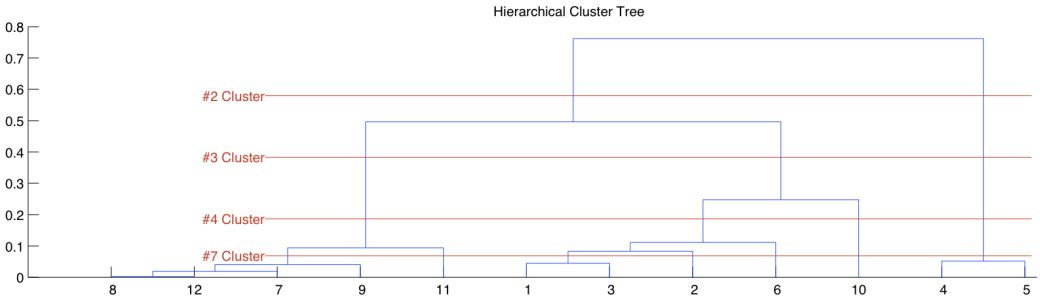


Figure 3.4: Dendrogram for Cluster Analysis of Time Series Segments. In general, the x-axis of a dendrogram identifies the indexes of the items (e.g. time series segments) that were grouped together, and the y-axis denotes the distance at which the single items were merged. Cutting a hierarchical cluster tree (dendrogram) at a specific height will give a clustering (highlighted in red color) for the selected precision or rather distance.

Hierarchical clustering has the distinct advantage that any valid distance measure can be used. In fact, the observations themselves are not required; all that is needed is a matrix of distances. The choice of an appropriate metric does influence the shape of the clusters, as some elements may be close to one another according to one distance and farther away according to another. Some commonly used metrics for hierarchical clustering are Euclidean distance, cosine similarity, Mahalanobis distance (covariance matrix) and Pearson's correlation. For text or other non-numeric data, metrics such as the Hamming distance or Levenshtein distance are often used. Given an appropriate metric, an agglomerative hierarchical cluster tree can be created by several different methods, which differ from one another in how they measure the distance between the clusters. Available methods are single and complete linkage, also known as shortest and furthest distance, as well as average, centroid and inner-squared distance. In our proposed approach we employ cosine distance (refer to Equation 3.6) to calculate the similarities between two feature vectors F_1 and F_2 as well as average linkage to compute average distance between all pairs of objects in any two clusters C_1 and C_2 (see Equation 3.5):

$$link(C_1, C_2) = \frac{1}{|C_1||C_2|} \sum_{c_1 \in C_1} \sum_{c_2 \in C_2} cos(c_1, c_2) \tag{3.5}$$

$$cos(F_1, F_2) = 1 - \frac{F_1 F_2^T}{\sqrt{(F_1 F_1^T)(F_2 F_2^T)}} \tag{3.6}$$

where all objects of a cluster are feature vectors, i.e. $c_1 \widehat{=} F_1$ and $F_1 \in C_1$.

Although distance metric and cluster measure directly influence the grouping of items or rather segments, meaningful clustering always depends on the selection of expressive features. In machine learning and statistics, feature selection is the technique of selecting a subset of relevant characteristics for building robust learning models. From a theoretical perspective, it can be shown that optimal feature selection for (un)-supervised learning problems requires an exhaustive search of all possible subsets of features. For practical learning algorithms, the search is for a satisfactory set of features instead of an optimal set. Beside the previously introduced feature extraction via SVD, we furthermore examine common signal processing features like standard deviation, local maxima and minima as well as the increase or decrease of the individual signals within a segment. The evaluation of the proposed time series segmentation and segment clustering is discussed in the following.

3.4.3 Evaluation on Vehicular Data

Having explained our three-step approach in detail we are now in the position to evaluate its performance on the recognition of complex drive maneuvers in real-life vehicular sensor data recorded during car drives. Although many different environmental and motor parameters were measured, we merely consider the progressing of speed, revolution, gear and accelerator signal, which are highly correlated and provide sufficient informative value to perform a preliminary evaluation.

In supervised learning, one usually has a training dataset of input and target values, which is used to adjust the parameters of the learning algorithm until it is able to model the underlying structure of the presented data. The trained model can then be used to predict target values for previously unseen input. However, in unsupervised learning one aims to discover the underlying structure of the presented data without having any target values for validation.

In case of our time series segmentation, we do not know what makes up a "good" segment and at which point of our bottom-up algorithm we should stop to merge adjacent segments. Nonetheless we are able to determine a merge threshold by means of the applied cost function [2, 182]. Figure 3.5 illustrates the growth of the Q-measure (sum over all segments) for each iteration of our bottom-up algorithm. A possible termination criterion for merging time series segments could be a steep ascent of the cost function. Besides automatically determining the number of segments (merge threshold), we furthermore compare different approaches of time series segmentation by evaluating the overall cost of the established segments. Figure 3.6 shows a comparison of our proposed bottom-up segmentation (BU), the introduced

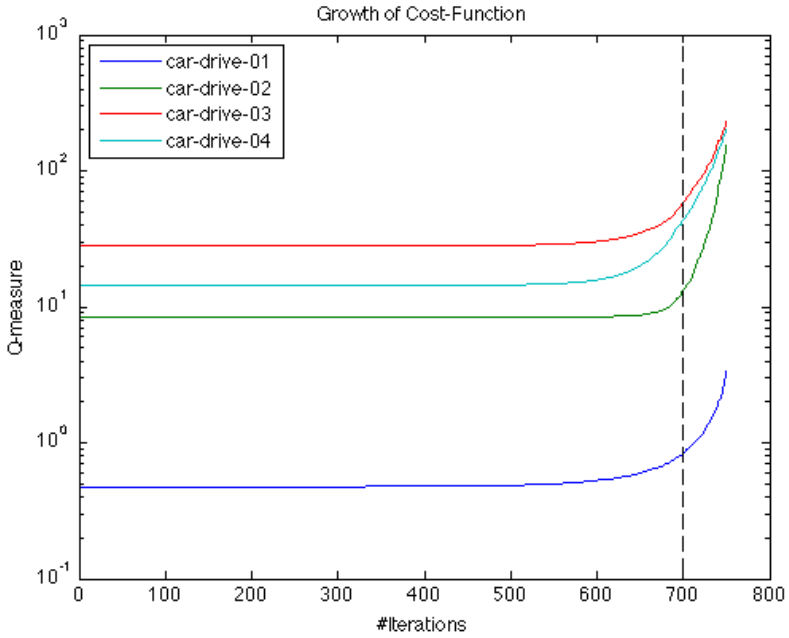


Figure 3.5: The Q-measure indicates how well our SVD-based model fits the segmentation of a time series. In the case that the segmentation cost (sum of Q-measures over all segments) exceeds a certain limit or the cost function exhibits a steep ascent, the bottom-up algorithm should stop to merge adjacent segments. The above plot shows the evolution of the segmentation cost for four different car drives of same length (8000 ms). All time series were initially partitioned into 800 segments of equal length, which then were iteratively merged to a final number of 50 segments by our bottom-up algorithm (doing 750 merge operations). The above plot illustrates that for all car drives the segmentation costs increases rapidly after about 700 iterations. Under these conditions, it would be advisable to terminate the bottom-up algorithm after a total number of 100 segments were established, which is furthermore used as a threshold for the following performance evaluation.

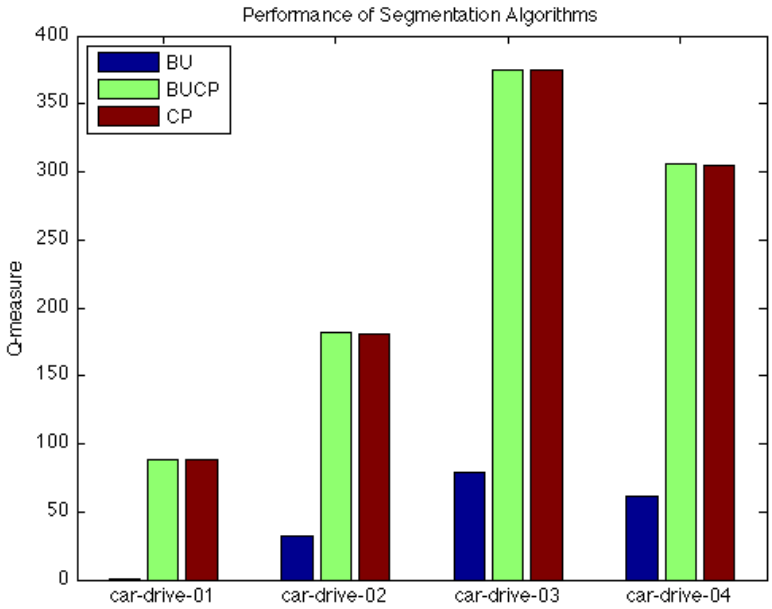


Figure 3.6: This plot shows the performance of three different segmentation algorithms in terms of the employed cost-function. For our evaluation we assume that the total cost of time series segmentation is the sum of Q-measures over all segments. Due to the fact that the Q-measure is an indicator for the homogeneity of individual segments, it also gives us an idea about how accurate our SVD-based model fits the underlying structure of the examined time series. As a general rule, the lower the sum of Q-measures for a given segmentation the better. In order to compare the bottom-up (BU), critical point (CP) and mixed BUCP approach, we evaluate all algorithms with the same parameter settings and on the same time series datasets. The comparison shows that for all examined car drives the straightforward bottom-up algorithm performed best. To our surprise, both CP algorithm and mixed BUCP approach achieved similar results. This can be explained by the fact that the CP algorithm segments a time series according local extrema of individual signals, which might not have a noticeable effect on the Q-measure of the overall model (considering all examined signals), even if the established segments are merged by means of the introduced BU approach in a second step. The performance of (BU-)CP might vary slightly for each individual signal.

critical point (CP) approach and a mixed BUCP segmentation for several different car drives.

Clustering falls into the category of unsupervised learning algorithm as well, and therefore exhibits the common problem of validation. Similar to the segmentation task we do not know for which threshold we should stop to merge single items or rather clusters. However, it is possible to validate the established clusters based on their intra and/or inter cluster similarity. In other words, similar to community detection in complex networks [110], we want to find groups of nodes or items that are more densely connected internally then with the rest of the network.

Due to the fact that we employ average linkage as cluster measure, both internal and external item distance grow monotonically with an increasing number of clusters. For the evaluation we take account of the external cluster distance, which can be retrieved from the hierarchical cluster tree generated during segment grouping (also refer to the dendrogram in Figure 3.4). The cost function for several different sample datasets and the determination of a reasonable number of clusters can be inferred from Figure 3.7. We found that the slope of the distance function is a useful indicator for determining a reasonable number of clusters with sufficient differentiation. However, cluster quantity always depends on the underlying structure of the time series and might vary from dataset to dataset.

For a better understanding of time series segmentation and clustering, we present an evaluated time series dataset in Figure 3.8. As mentioned previously, we are mainly interested in the recognition of complex drive maneuvers in sensor data recorded from vehicles. To identify complex drive maneuvers, we first of all employ time series segmentation and clustering as introduced before, and subsequently retrieve recurring patterns of segment sequences. In our case, sequences are regarded as sets of grouped segments, which exhibit a distinct order of cluster labels. If the determined time series segments are considered as individual situations in car driving, then segment sequences can be viewed as complex drive maneuvers, like stop and go at a traffic light. Figure 3.8 illustrates several recurring sequences, which denote complex drive maneuvers of a predefined length.

Drive maneuvers might differ from person to person, and also depend on the motorization of the vehicles. Therefore it is interesting to analyze the patterns that characterize individuals or vehicles respectively. In our future work, we plan to identify sequences that describe normal drives or rather drivers, as well as sequences that can be regarded as critical drive maneuvers or abnormalities. This knowledge is especially valuable for the motor vehicle industry, because it allows for performance optimization of an engine based on the understanding of frequent or abnormal situations.

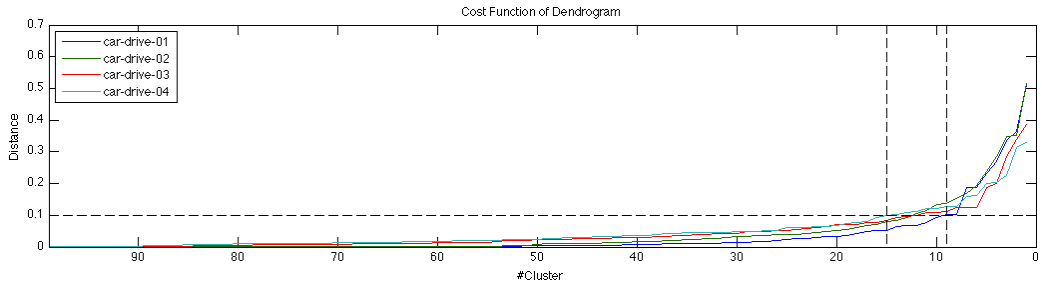


Figure 3.7: Cost Function of Hierarchical Cluster Tree. This plot illustrates the growth of distance between clusters for a decreasing number of groups. In other words, the graphs show at which cost the groups are merged by the agglomerative hierarchical clustering algorithm. Due to the fact that the cost function grows exponentially, the agglomerative hierarchical clustering should terminate at a certain point. If we assume a threshold of 0.1 distance (dashed black line), a number of 9 to 15 clusters would be appropriate to group the segments of the examined datasets. A threshold could be either predefined by an expert or learned by the clustering algorithm. For unsupervised learning the slope of the cost function is a useful indicator for determining a reasonable number of clusters with sufficient discrimination.

To sum up, our proposed pattern recognition approach extends the sensor fusion algorithm developed by Abonyi [2]. We use bottom-up segmentation to partition the time series data into homogeneous intervals that can be viewed as situations, and subsequently group the recognized segments by means of agglomerative clustering using straight-forward statistical features, such as standard deviation, local extrema as well as the slope of the individual signals within a segment. For the clustering task we employ cosine similarity to calculate the distances between feature vectors and average linkage to compute average distance between all pairs of objects (i.e. segments) in any two groups.

Although this paper discusses context recognition in terms of sensor data recorded from vehicles, the proposed time series analysis is also applicable to datasets from other domains. The following section gives some examples of related work.

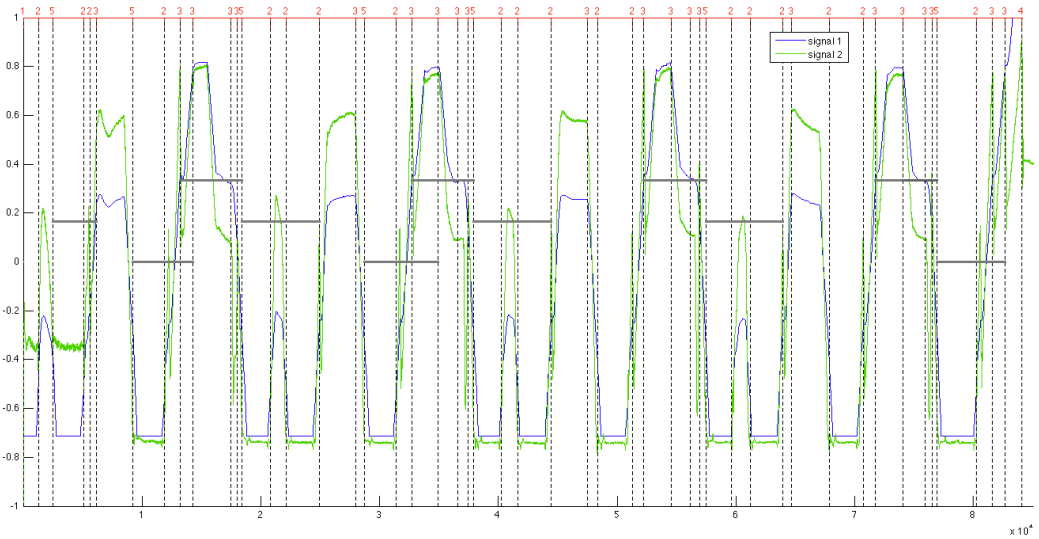


Figure 3.8: Time series segmentation and clustering of vehicular sensor data, in our case speed and accelerator signal. This plot combines the results of time series segmentation, segment clustering and sequence analysis. In the first step we employed the bottom-up (BU) segmentation approach to identify internally homogeneous time intervals. Whereas the merge threshold was determined according the growth of the segmentation cost (refer to Figure 3.5). The determined segments are unequally distributed and represent individual situations that occur during a car drive (illustrated by vertical broken lines). In a second step we grouped the identified segments by means of agglomerative hierarchical clustering. To measure the similarity of the individual segments we extracted different signal features like standard deviation, minimum and maximum value, as well as ascent. The established feature vectors were compared by means of cosine similarity. For the examined time series data we decided to group the individual segments into 5 clusters (according to the cost function of the cluster tree; also refer to Figure 3.7). The cluster labels (highlighted in red color) are shown at the top of the respective segment borders. In the final step we retrieved cluster sequences that occur in regular time intervals. The cluster sequences are illustrated by horizontal bars, whereas identical sequences are plotted at the same height (highlighted in dark gray). In case of vehicular sensor data, sequences of segments can be regarded as complex drive maneuvers, such as stop and go at traffic lights or overland drives.

3.5 Related Work

The sensor fusion algorithm developed by Abonyi et al. [2] is able to segment a multivariate time series in a bottom-up manner. Adjacent segments are merged iteratively if their combined model does not exceed a predefined reconstruction error, which is computed by means of principal component analysis. The PCA model is used to measure the homogeneity of the segment according the change of correlation among the variables. Since the PCA model defines linear hyperplanes, the proposed segmentation algorithm can be considered as the multivariate extension of piecewise linear approximation. Our own work can be considered as an extension of the sensor fusion algorithm and has been evaluated on several real-life datasets.

In a following paper [1] Abonyi introduces a modified time series segmentation approach, which is able to group overlapping and vague segments by means of fuzzy clustering. The proposed algorithm favors contiguous clusters in time and is capable to detect changes in the hidden structure of multivariate time-series, where local probabilistic PCA models are used to represent the segments. The evaluation results suggest that the proposed approach can be applied to extract useful information from temporal databases. In our future work we plan to employ fuzzy clustering to model overlapping situation in car driving.

A group of researchers from the University of California (Keogh et al. [86]) reviewed the three major segmentation approaches in the literature (*Sliding Windows, Top-Down & Bottom-Up*) and provided an extensive empirical evaluation on a heterogeneous collection of datasets. In order to evaluate the segmentation algorithms, all datasets are represented as piecewise linear approximation (PLA), which refers to an approximation of a time series. Among all segmentation algorithms the bottom-up approach performed best, which corresponds to our own evaluation. Furthermore, they introduce a novel on-line algorithm for segmenting time series that scales linearly to the size of the data and produces high quality approximations.

In a survey about time series abstraction methods, Hoeppner [67] discusses data representations that have been used in literature. The author states that an abstracted data representation is necessary to account for the human way of perceiving time series. Moreover, the paper suggests to use multi-scale methods or rather multi-resolution analysis, which only considers features that persist over a broad range of scale and compensate dislocation effects. We aim to extend our work to multi-scale methods to capture features from segments or situations of different scales.

Most segmentation algorithms belong to the supervised learning family, where a labeled corpus is available to the algorithm in the learning phase.

An unsupervised learning algorithm for segmenting sequences of symbols or categorical events is presented by Shani et al. [173], where the algorithm never sees examples of successful segmentation, but still discovers meaningful segments. The proposed algorithm computes a maximum likelihood segmentation, which is most appropriate to hierarchical sequences, where smaller segments are grouped into larger segments. One advantage of this probabilistic approach is that it allows one to suggest conditional entropy as quality measurement of segmentation in absence of labeled data. The idea of measuring the quality of segmentation inspired our own performance evaluation on the basis of the employed cost function.

In many cases time series segmentation is used to recognize context from sensor data. For instance, recognizing the context of use is important in making mobile devices as simple to use as possible. The awareness of a user's situation can help the device and underlying services in providing an adaptive and personalized user interface. Himberg et al. [66] present a randomized variation of dynamic programming that minimizes the intra-segment variances and gives approximately optimal results. Although their approach is interesting, we are more fascinated by the idea of integrating our own segmentation and clustering algorithm into a real-life context-aware application.

The use of context in mobile devices is receiving increasing attention in mobile and ubiquitous computing research. Gellersen et al. [53] have investigated multi-sensor context-awareness in a series of projects and report experience from development of a number of device prototypes. In order to design context-aware systems, raw sensor data is transformed into 'cues' that incorporate various features based on simple statistics (e.g., standard deviation, quartile distance, etc.). The mapping from cues to context may be explicit, for instance when certain cues are known to be relevant indicators of a specific context, or implicit in the result of a supervised or unsupervised learning technique. Based on several software prototypes, Gellersen et al. [53] have shown that the integration of diverse sensors is a practical approach to obtain context representing real-life situations. This paper describes an interesting use case of multi-sensor context-awareness and approves the relevance and topicality of our own foundational research.

Another considerable context-aware application is *CenceMe* [139], which infers the presence of individuals using sensor-enabled mobile phones and shares this information through social network applications (such as *Facebook* and *MySpace*). The *CenceMe* framework contributes to the design of light-weight classifiers, running on mobile phones, which realize a split-level classification paradigm. To infer the presence of individuals, audio data is classified using the unsupervised learning technique of discriminant analysis,

where the feature vectors are composed of the mean and standard deviation of the DFT (Discrete Fourier Transform) power. In case of context-inference from accelerometer readings the mean, standard deviation, and the number of peaks per unit time are accurate feature vector components, providing high classification accuracy. Classification of accelerometer data is based on a decision tree technique (J48 algorithm of *WEKA* data mining workbench[1]).

3.6 Conclusion

This chapter is concerned with time series segmentation, segment clustering, and segment sequence discovery. We have proposed a SVD-based bottom-up algorithm that identifies internally homogeneous time series segments. To recognize recurring patterns, the established time series segments were grouped via agglomerative hierarchical clustering. Subsequently we retrieved recurring sequences of grouped segments, which can be considered as complex situations or higher-level context.

The proposed bottom-up segmentation extends the introduced sensor fusion algorithm [2] by several features, such as automatic determination of model rank and merge threshold. Furthermore, we evaluated the extended segmentation algorithm on several real-life datasets, which comprise the progression of motor parameters during a car drive.

Our evaluation demonstrated that the proposed bottom-up (BU) approach performs better than the straightforward critical point (CP) segmentation. Additionally, we suggested to employ the segmentation cost to determine the merge threshold. Based on further experiments we have showed how to choose a suitable number of clusters for grouping established time series segments.

Although this paper discusses time series segmentation and clustering in terms of multivariate sensor data recorded from vehicles, we have explained that our approach is suitable for other domains, such as sensor data collected from smart phones. In Appendix A 3.8 we describe different real-life use cases and applications, where time series analysis may be applied to recognize higher-level context or complex situations.

3.7 Future Work

In our future work, we plan further analysis of vehicular sensor data to categorize drives and drivers. Usually, drive maneuvers differ from person to

[1] `http://www.cs.waikato.ac.nz/~ml/weka/`

person, and also depend on the motorization of the car. Therefore, it is interesting to analyze the patterns that characterize individuals or vehicles respectively. This knowledge is especially valuable for car manufacturers like the *Volkswagen AG*, because it allows more efficient testing of recently developed engines [182].

For the classification of contextual patterns in car driving we plan to employ the Dynamic Time Warping (DTW) distance measure [92, 159], because it is able to handle local scaling (warping) invariance [9]. Kroschel et al. [103] describe an advanced DTW segmentation and classification approach, which detects change points and categorizes time series segments simultaneously. We aim to extend this approach to cope with the task of multivariate time series analysis.

Since our goal is to design solutions for diverse application scenarios, our long-term perspective consists in exploring feature extraction, segmentation and clustering/classification methods for contextual pattern recognition in sensor data from other domains, such as smart home environments and mobile electronic devices. We expect to gain insight in how selected machine learning methods cooperate in multi-step approaches for solving the problem of detecting contextual patterns in multivariate time series.

3.8 Appendix A - Applications

This section describes some practical use-cases for our proposed time series analysis [182]. In the following we introduce various domains, where segmentation and/or classification of multivariate time series is needed or necessary.

3.8.1 Recurring Situations in Car Driving

Our proposed pattern recognition and classification for multivariate time series [182] was developed in cooperation with the *Volkswagen AG*, Wolfsburg. In order to optimize their engines and to comply with emission regulations, *VW* aims to identify abnormal and recurring situations that take place during car drives. In general, significant situations are acceleration or loss of speed, and complex drive maneuvers that are a sequence of both.

The developed algorithm first separates the motor signals or rather time series into segments that can be considered as situations, and then clusters the recognized segments into groups of similar context. The time series segmentation is established in a bottom-up manner according the correlation of the individual signals. Recognized segments are grouped in terms of statistical features using agglomerative hierarchical clustering. Subsequently we

retrieved sequences of grouped segments, which exhibit a distinct order of cluster labels and can be considered as complex situations or higher-level context. If the determined time series segments are regarded as individual situations in car driving, then segment sequences can be viewed as complex drive maneuvers, like stop and go at a traffic light. Drive maneuvers might differ from person to person, and also depend on the motorization of the vehicles. Therefore it would be interesting to analyze the patterns that characterize individuals or vehicles respectively.

Our proposed segmentation and classification approach was evaluated on the basis of real-life sensor data from different vehicles recorded during test drives. According to our evaluation it is feasible to recognize recurring patterns in time series by means of bottom-up segmentation and hierarchical clustering. Although the proposed pattern recognition and classification for multivariate time series is discussed in terms of vehicular sensors, it is also applicable to time series data of other domains. In the following we describe further real-life applications for time series analysis that might be of interest for our future research.

3.8.2 Event Detection in Video Sequences

Event detection in video sequences is one of the active research areas nowadays. It is related to many application domains, ranging from semantic indexing and retrieval to intelligent video surveillance. Various approaches to detect video events have been proposed in the literature, most of which rely on supervised learning methods [7, 55, 56]. For these supervised learning approaches, large-scale training data is required to model video events and to train classifiers adapted to detect them. However, the training data is poorly available.

Semi-supervised machine learning approaches seem appropriate for video domains in which labeled data is limited, whereas unlabeled data abounds. Unlabeled data may be relatively easy to collect, for example by using video search engines (e.g. Google Videos, You Tube). Labeled data, however, is often expensive or time consuming to obtain, as it requires efforts of expert human annotators. Additionally, in security, surveillance and monitoring contexts (labeled and/or unlabeled) data is difficult to collect because of privacy issues. Semi-supervised learning addresses building improved classifiers by using large amounts of unlabeled data (which does not necessarily need to be collected beforehand), together with limited amounts of labeled one.

Aiming to avoid such issues, we propose a generic multi-modal video event detection system using a semi-supervised learning approach [4]. In our future work, we plan to investigate different semi-supervised machine

learning techniques and intend to build a framework for event detection in videos. The content of the video data will be described by combined visual and audio features.

As a matter of course, both audio and visual data will be evaluated by means of time series analysis. In case of event detection in video sequences, time series can be used to analyze temporal trends, data variation, as well as cyclical irregularity. A robust distance or similarity measure for time series is Dynamic Time Warping (DTW), which allows non-linear alignments [9, 92, 159]. As a result, the probability of the existence of a pre-defined event will be provided by the multi-modal classifier.

3.8.3 Context-Awareness of Mobile Devices

Context awareness of mobile devices is a paradigm in which applications can discover and take advantage of user activity and device status. Recently an increasing number of smart phones and tablet computers integrate sensors, which enable researchers to study user behavior and corresponding environmental conditions [22].

The vision of context-aware computing was influenced by ubiquitous computing, where computers provide value-added services throughout the physical environment without being visible to the user. Likewise, context-aware applications take advantage of environmental characteristics to adapt to user needs without consuming user attention [22].

A general motivation for context-awareness in mobile applications are fast changing environments and increasingly complex situations [53]. Context at system level can be exploited for resource and power management. At application level, context-awareness enables both adaptive applications and explicit context-based services. And at the user interface level, the use of context facilitates a shift from explicit to implicit human-computer interaction.

Gellersen et al. [53] have investigated multi-sensor context-awareness in a series of projects and report experience from development of a number of device prototypes. In order to design context-aware systems, raw sensor data is transformed into *cues* that incorporate various statistical features. The mapping from *cues* to context may be explicit, for instance when certain *cues* are known to be relevant indicators of a specific context, or implicit in the result of a supervised or unsupervised learning technique. Based on several software prototypes, Gellerson et al. [53] have shown that the integration of diverse sensors is a practical approach to obtain context representing real-life situations. Their paper describes an interesting use-case of multi-sensor context-awareness and approves the relevance and topicality of our

own fundamental research.

Another considerable context-aware application is *CenceMe* [139], which infers the presence of individuals using sensor-enabled mobile phones and shares this information through social network applications (such as *Facebook* and *MySpace*). The *CenceMe* framework contributes to the design of light-weight classifiers, running on mobile phones, which realize a split-level classification paradigm.

We believe that our proposed pattern recognition approach for multivariate time series [182] can be used to enable context-awareness of mobile devices, where sensor signals or rather time series are segmented into complex situations, which in turn are classified according domain knowledge.

3.8.4 Ambient Assisted Living

Smart environments make it possible to build ubiquitous applications that assist users during their everyday life, at any time, in any context [163]. This requires the applications to adapt themselves to the current context and provide a usable and suitable interaction at all times. Lehmann et al. [109, 163, 172] propose a model-based approach, which allows adapting the user interface at runtime to numerous contexts-of-use (user, platform and environment). Their multi-access service platform (*MASP*) provides automatic distribution and layout algorithms for adapting the applications also to contexts unforeseen at design time.

Developers of applications for smart environments must cope with a multitude of sensors, devices, users and thus contexts [109]. Applications in smart environments are required to monitor themselves and their environment and provide appropriate reactions to monitored changes before they lead to a disruption of operation. The required context information must be well defined and properly modeled.

Context or rather situations are stored in the context-model, which aggregates environmental informations measured by integrated device sensors as well as system-states given by interactions between users and interface [172]. Our proposed pattern recognition approach for multivariate time series [182] could be employed to identify recurring and/or abnormal situations in smart environments, which in turn could be used to automatically adapt user interfaces.

Chapter 4

Lucky Time Warping Distance
for Time Series Classification

In time series mining, the Dynamic Time Warping (DTW) distance is a commonly and widely used similarity measure. Since the computational complexity of the DTW distance is quadratic, various kinds of warping constraints, lower bounds and abstractions have been developed to speed up time series mining under DTW distance.

In this chapter, we propose a novel Lucky Time Warping (LTW) distance, with linear time and space complexity, which uses a greedy algorithm to accelerate distance calculations for pairwise similarity comparisons of time series. Our evaluation on 1-Nearest-Neighbor (1NN) time series classification shows that, compared to constrained DTW with lower bounding, our LTW distance trades classification accuracy against computational cost reasonably well, and therefore can also be considered as a fast alternative for other time series mining tasks.

4.1 Introduction

The Dynamic Time Warping (DTW) distance was first introduced to the data mining community almost two decades ago [13], and since then has been used as a utility for various time series mining tasks; including classification, clustering, and indexing of (sub)sequences [36]. Its current popularity and widespread use is owing to the fact that, in contrast to the Euclidean distance, the DTW distance works well for time series with various distortions and multiple invariance [9, 179]. But the 'brute-force' dynamic programming approach of DTW leads to quadratic space and time complexity [14].

In time series classification, the combination of 1-Nearest-Neighbor (1NN) classifier with DTW has been proven exceptionally difficult to beat [211].

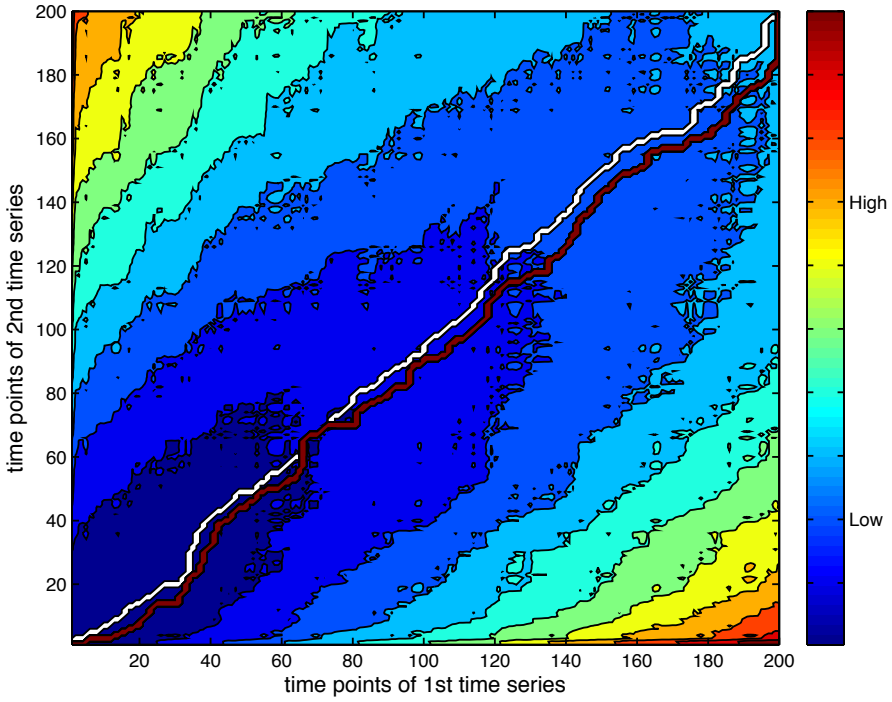


Figure 4.1: Warping matrix of two randomly generated time series, exemplifying the spread of the cumulative warping cost. The optimal (minimum cost) warping path computed by the dynamic programming approach of the DTW distance is shown in red color, and the lucky warping path found by the greedy approach of the LTW distance is shown in white color. Both paths pass through low cost regions (ranging from dark to light blue) and avoid high cost regions (displayed in shades of yellow and red). Note that a warping path is required to start and finish in diagonal opposite corners of the matrix and has to be continuous as well as monotonous, also see 4.2.1.

Nevertheless, if naively implemented, the 1NN-DTW approach is computationally demanding, because 1NN classification usually involves a great number of quadratic DTW calculations [160].

Hitherto existing approaches that aim to reduce the computational complexity of 1NN-DTW time series classification fall into the categories of algorithms [166, 167] which use (1) numerosity reduction to discard a large fraction of training data [20, 150, 211], (2) constraints to limit the number of cells that are evaluated in the cost/similarity matrix [74, 159, 160, 165], (3) abstraction to perform DTW on a reduced representation of the data [29, 90] or (4) lower bounding to reduce the number of times DTW must run [93, 98, 154, 216].

We propose a novel Lucky Time Warping (LTW) distance, with linear time and space complexity, which uses a greedy algorithm to accelerate distance calculations for pairwise similarity comparisons of time series. Our evaluation on 1NN time series classification shows that, compared to state-of-the-art distance measures like constrained DTW with lower bounding, our LTW distance trades classification accuracy against computation time reasonably well, and therefore can also be considered as a fast alternative for other time series mining tasks. Figure 4.1 allows a geometric intuition for the optimal/minimum cost and lucky warping path returned by the DTW and LTW distance function respectively.

Our work was inspired by other alternative procedures for implementing a DTW algorithm with substantially reduced computation, including early techniques for a directed graph search through a grid to find the best warping path [19] and recently proposed greedy approximate solutions [125]. But, to the best of our knowledge, there is no study that thoroughly compares the performance of the traditional DTW distance (with/out constraint and lower bound) to an alternative greedy approximate solution, like our proposed LTW distance, in respect of classification accuracy and computation complexity, on an extensive time series archive [95], as presented in this work.

The rest of the paper is organized as followed: Section 4.2 introduces notation and background to allow a formal definition of our LTW distance in Section 4.3. Experimental results on classification accuracy and computational cost of our proposed LTW distance are presented in Section 4.4. We conclude with a discussion of future work in Section 4.5.

4.2 Notation and Background

The 1NN-DTW approach for time series classification has been shown to achieve high classification accuracy, and its computational demand has been mitigated by warping constraints and pruning techniques. Hence, we want to give some background on DTW and related speedup techniques, before we introduce and compare our own LTW distance.

4.2.1 Dynamic Time Warping

The DTW algorithm aims to find the optimal warping path with minimum cost to describe the similarity of two time series. In case that the optimal warping path amounts to a relatively low cost the examined time series are considered to be similar, whereas high cost or distance implies dissimilarity [13]. To determine the (dis)similarity of two time series or rather to find the

optimal warping path the DTW algorithm uses a brute-force dynamic programming approach which tests all possible paths within the defined warping window, leading to quadratic complexity [14].

Since many of the subsequent explanations are based on DTW, we present a formal definition of the DTW distance as introduced by Keogh et al. [90, 92]. Suppose we have two time series X and Y, of length n and m respectively, where:

$$\begin{aligned} X &= \{x_1, x_2, \ldots, x_i, \ldots, x_n\} \\ Y &= \{y_1, y_2, \ldots, y_j, \ldots, y_m\} \end{aligned} \tag{4.1}$$

To align these two sequences, the DTW algorithm first constructs a n-by-m matrix, where the $(i,j)^{th}$ element of the matrix corresponds to the squared distance, $d(x_i, y_j) = (x_i - y_j)^2$, which is the alignment between points x_i and y_j. To find the best match between these two sequences, the DTW algorithm retrieves a path through the matrix that minimizes the total cumulative distance between them. In particular, the optimal path is the path that minimizes the warping cost:

$$DTW(X,Y) = \min\left\{\sqrt{\sum_{k=1}^{K} w_k}\right\} \tag{4.2}$$

where w_k is the matrix element $(i,j)_k$ that also belongs to k^{th} element of a warping path $\mathcal{W}$, a contiguous set of matrix elements that represent a mapping between Q and C. This warping path can be found using dynamic programming to evaluate the following recurrence.

$$\gamma(i,j) = d(x_i, y_j) + \min\{\gamma(i\text{-}1, j\text{-}1), \gamma(i\text{-}1, j), \gamma(i, j\text{-}1)\} \tag{4.3}$$

where $d(x_i, y_j)$ is the distance found in the current cell, and $\gamma(i,j)$ is the cumulative distance of $d(x_i, y_j)$ and the minimum cumulative distances from the three adjacent cells. Moreover, the warping path is typically subject to several constraints.

- The **Boundary conditions** require the warping path start and finish in diagonally opposite corner cells of the matrix, with $w_1 = (1,1)$ and $w_K = (m,n)$.
- The **Continuity conditions** restrict the allowed steps in the warping path to adjacent cells, including diagonally adjacent cells. Given $w_k = (i,j)$ then $w_{k-1} = (i',j')$ where $i - i' \leq 1$ and $j - j' \leq 1$.
- The **Monotonicity conditions** force the points in $\mathcal{W}$ to be monotonically spaced in time. Given $w_k = (i,j)$ then $w_{k-1} = (i',j')$ where $i - i' \geq 0$ and $j - j' \geq 0$.

4.2.2 Numerosity Reduction

One way to reduce the classification time of the 1NN-DTW approach is to discard a large fraction of the training data by means of numerosity reduction algorithms [20, 211], which choose the objects that maintain or even improve the accuracy. Other work on dissimilarity-based classifiers even suggests that by using a few, but well chosen prototypes, it is possible to achieve a better classification performance in both speed and accuracy than by using all the training samples [150].

4.2.3 Warping Constraints

Constraints work well in domains where time series have only a small variance, but perform poorly if time series are of events that start and stop at radically different times [167]. The two most commonly used constraints are the Sakoe-Chiba Band [165] and the Itakura Parallelogram [74], which both speed up DTW by a constant factor, but still lead to quadratic complexity if the window size is a linear function of the time series length [160, 167]. Given the formal definition of DTW in Section 4.2.1, the Sakoe-Chiba Band [165] is an:

> **Adjustment Window condition** which corresponds to the fact that time-axis fluctuation in usual cases never causes a too excessive timing difference. Given $w_k = (i, j)$ then $|i - j| \leq r$ where r is the size of the warping window.

Recent work has shown that learned constraints are able to boost classification accuracy and outperform any uniform band [159]. Although optimal band shape allows tighter lower bounds and thus more effective pruning, the Euclidean distance is still orders of magnitude faster than DTW with learned constraints [159].

4.2.4 Abstraction

Early work on time series representation introduces a modification of DTW which operates on a higher level abstraction, and produces one to two orders of magnitude speedup with no decrease in accuracy [29, 90].

Although abstraction speeds up DTW by operating on a reduced presentation of the data, the calculated warp path becomes increasingly inaccurate as the level of abstraction increases [167], and projecting the low resolution warp path to the full resolution usually gives a result far from optimal, because significant local variations are ignored.

4.2.5 Lower Bounding

Another way to mitigate the high computational complexity of 1NN-DTW time series classification is to prune similarity calculations by lower bounding the DTW distance measure. Recent studies on large-scale time series data have shown that DTW in combination with certain lower bounds lead to quasi-linear complexity for indexing problems [154, 160], leading to the widespread belief that there is no need for further improvements in speed. However, pruning does not effect the intrinsic quadratic complexity of the actual DTW calculation [167].

There are two desirable properties of a lower bounding measure; it must (1) be fast to compute and (2) return a relatively close approximation of the true distance[93, 159, 160]. Since the aim of lower bounding is to prune computational expensive similarity calculations, an approximation is required to be faster than the original measure and should be at most linear in the length of the sequences.

Known approaches to lower-bound the DTW distance include approximations that involve the difference between minimum and maximum points [98, 216] and more sophisticated techniques that employ global constraints to create warping-envelopes [93].

For further explanations and experiments we consider the lower bound introduced by Keogh and Ratanamahatana [93], denoted as LB_Keogh, which compares each candidate time series Y to the upper U and lower L sequence of the bounding envelope that encloses the time series query X. In case of the Sakoe-Chiba band with an window size r the two new sequences can be defined as:

$$\begin{aligned} U_i &= \max(x_{i-r} : x_{i+r}) \\ L_i &= \min(x_{i-r} : x_{i+r}) \end{aligned} \tag{4.4}$$

where $\forall\ U_i \geq x_i \geq L_i$. Having defined U and L, the lower bounding measure for DTW can be formulated as:

$$LB\_Keogh(X,Y) = \sqrt{\sum_{i=1}^{n} \begin{cases} (y_i - U_i)^2 & \text{if } y_i > U_i \\ (y_i - L_i)^2 & \text{if } y_i < L_i \\ 0 & \text{otherwise} \end{cases}} \tag{4.5}$$

For any two sequences X and Y of the same length n, for any global constraint on the warping path of the form $j - r \leq i \leq j + r$, the following inequality holds:

$$LB\_Keogh(X,Y) \leq DTW(X,Y) \tag{4.6}$$

4.2.6 Amortized Computational Cost

Since execution time is a poor choice for evaluating the performance gain achieved by lower bounds, recent work [92, 93] suggest to compute the average or amortized percentage of warping matrix that need to be accessed for each individual DTW distance calculation.

In order to compare the computational efficiency of 1NN time series classification in relation to arbitrary distance measures and pruning techniques, we propose to compute the amortized computational cost (ACC) by accounting the basic similarity operations of the classification task with regard to the time series length. We define a basic similarity operation as the squared distance between a pair of time points, which corresponds to the calculation of exactly one entry in the time warping matrix:

$$d(x_i, y_j) = (x_i - y_j)^2 \tag{4.7}$$

Assuming two time series of length n and a warping constraint of r time points (Sakoe-Chiba band), we can compute the number of similarity operations or rather matrix elements E that are involved in a single cDTW (constrained Dynamic Time Warping) distance calculation by the following equation:

$$E_{cDTW}(n, r) = (2r + 1)n - r(r + 1) \tag{4.8}$$

In case that cDTW is combined with lower bounding (LB) techniques, we can reduce the number of expensive cDTW similarity calculations at the ratio of all 1NN comparisons to p percent. Due to the fact that a lower bound approximation is computed for every similarity comparison, no matter if an expensive similarity calculation follows or not, one may assume an additional computational cost that is constant. The computational cost for LB_Keogh [93] equates to:

$$E_{\text{LB-Keogh}}(n) = 2n \tag{4.9}$$

since each candidate time series is compared to the upper and lower sequence of the bounding envelope that encloses the time series query. The comparison between candidate time series and bounding envelope is based on the Euclidean distance which involves $2n$ squared distance computations or basic similarity operations. Consequently, one can compute the amortized computational cost of cDTW with LB_Keogh as follows:

$$E_{\text{cDTW-LB}}(n,r) = E_{\text{LB-Keogh}}(n) + p * E_{cDTW}(n,r)$$
$$= 2n + p * \Big((2r+1)n - r(r+1)\Big) \tag{4.10}$$

In Section 4.4.4 we evaluate the ACC of 1NN classification in combination with DTW and LB_Keogh (1NN-DTW-LB approach) for a variety of time series datasets.

4.3 Lucky Time Warping Distance

In this section we introduce a novel Lucky Time Warping (LTW) distance, explain its underlying algorithm to find a warping path, and analyze its theoretical minimum and maximum computational cost.

The LTW distance finds a warping path that is constrained by the same (Boundary, Continuity, and Monotonicity) conditions that apply to the DTW distance. But, unlike the DTW distance, which computes all elements within the warping matrix or rather warping window to find the minimum cost warping path, the LTW distance uses a greedy approach that only evaluates matrix elements which most likely contribute to the actual warping path. According to the formalization in Section 4.2.1, the LTW distance between two time series X and Y is defined as:

$$LTW(X,Y) = \sqrt{\sum_{k=1}^{K} w_k} \tag{4.11}$$

where w_k is the k^{th} element of a warping path $\mathcal{W}$. Given w_k with $k = (i,j)$ then the warping path can be found using the following recurrence.

$$w_{k+1} = \min\{d(x_{i+1}, y_{j+1}), d(x_{i+1}, y_j), d(x_i, y_{j+1})\} \tag{4.12}$$

where ties are resolved in the order as given in the list of arguments. Imposing an order on picking the next warping w_{k+1} ensures that the LTW distance is well defined. Since

$$LTW(X,Y) = LTW(Y,X) \tag{4.13}$$

only holds if there are no ties, i.e. $d_{dia} \neq d_{up} \neq d_{right}$ (see Algorithm 4.1), we find that the LTW distance is not symmetric, and consequently not a metric. From

$$LTW(X,Y), LTW(Y,X) \geq DTW(X,Y) \tag{4.14}$$

follows that the LTW distance is an upper bound of the DTW distance. Since global constraints such as the Sakoe-Chiba band and the Itakura parallelogram can be imposed on the LTW distance in the usual way, the upper bounding property holds true in the constrained case. We deliberately refrain of discussing the tightness of LTW in respect to DTW, as will be explained in Section 4.4.5.

4.3.1 Lucky Warping Path

In contrast to DTW's brute-force approach, our LTW distance determines the warping path in a greedy manner, performing a variation of best-first search within the defined warping window. Consequently, the warping path computed by LTW is contained in the set of paths examined by DTW, just like the Euclidean distance is a special case of DTW where no warping takes place. Since DTW always finds the optimal warping path with minimum cost, our proposed LTW distance is supposed to return a suboptimal path with higher or equal cost/distance.

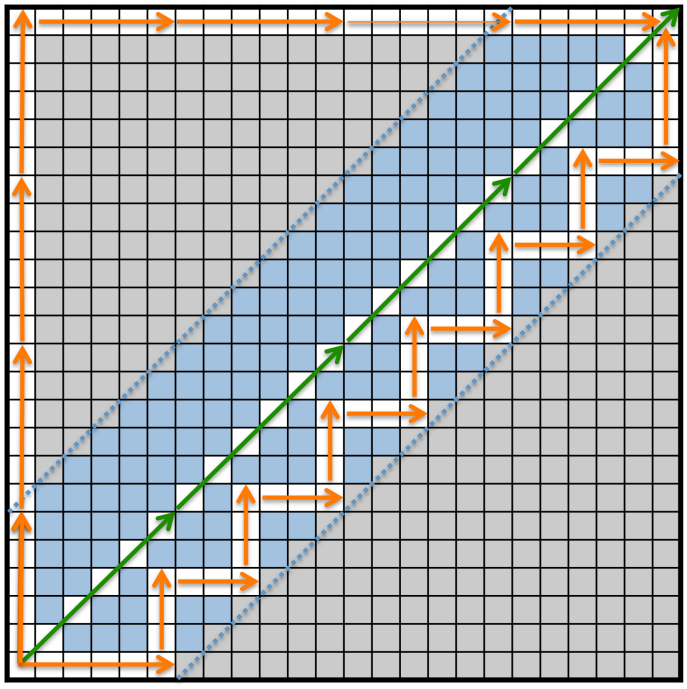


Figure 4.2: The shortest and longest possible warping path, constructed by the greedy algorithm of our LTW distance, highlighted in green and orange color respectively. The warping window is shown in blue.

Figure 4.1 illustrates the optimal and lucky warping path corresponding to the DTW and LTW distance of two randomly generated time series. Furthermore, Figure 4.1 exemplifies the spread of the cumulative warping cost, and shows that both optimal and lucky warping path only pass through low cost regions and avoid high cost regions. The observation of Figure 4.1, and other experiments on random data which we omit for sake of brevity, give reason to expect the lucky warping path to be close to the optimal warping path, or even to overlap with the minimum cost path in some time intervals. Nevertheless, as will be explained in Section 4.4.5, we refrain of discussing the tightness of LTW in respect to DTW.

4.3.2 Greedy Algorithm

In contrast to the brute-force dynamic programming approach of DTW, our LTW distance determines the warping path in a greedy manner, performing a kind of best-first search through the similarity matrix.

Algorithm 4.1 LTW Distance Measure

Input: $X, Y \ldots$ time series; $r \ldots$ warping window
Output: $d \ldots$ lucky distance

1: $i, j \leftarrow 1$
2: $d \leftarrow (x_i - y_j)^2$ $\{x_i, y_j$ equals $X(i), Y(j)\}$
3: $n \leftarrow$ length of X
4: $m \leftarrow$ length of Y
5: **while** $(i \leq n)$ and $(j \leq m)$ **do**
6: **if** $(i + 1 \leq n)$ and $(j + 1 \leq m)$ **then**
7: $d_{dia} \leftarrow (x_{i+1} - y_{j+1})^2$
8: **end if**
9: **if** $(i + 1 \leq n)$ and $(|i + 1 - j| \leq r)$ **then**
10: $d_{up} \leftarrow (x_{i+1} - y_j)^2$
11: **end if**
12: **if** $(j + 1 \leq m)$ and $(|j + 1 - i| \leq r)$ **then**
13: $d_{right} \leftarrow (x_i - y_{j+1})^2$
14: **end if**
15: $d_{\min} = \min(d_{dia}, d_{up}, d_{right})$
16: $d \leftarrow d + d_{\min}$
17: $i, j \leftarrow \text{index}(d_{\min})$ {update position}
18: **end while**

As described in Algorithm 4.1, the LTW approach constructs a warping path by iteratively moving from one corner to the opposite corner of the sim-

ilarity matrix (*Line 5*), repeatedly evaluating the cost of adjacent cells (*Line 7,10, and 13*) - going one step diagonal, up and right (*Line 6, 9, and 12*). Since a warping path must be monotonically spaced in time, the direction of the iterative moves through the similarity matrix is restricted, preventing backward movement. In fact, our LTW distance meets the Boundary, Continuity, and Monotonicity condition as formalized in Section 4.2.6.

4.3.3 Computational Cost

In the following section, we examine the computational cost of our proposed LTW distance measure from a theoretical perspective.

The introduced greedy algorithm constructs a warping path in an iterative manner, where each iteration involves the cost evaluation of three adjacent cells in the similarity matrix, one for each direction of the next possible move. Algorithm 4.1 demonstrates that the evaluation of an individual cell is associated with one call of the squared distance function (*Line 7, 10, and 13*), which represents our basic similarity operation (see Equation 4.7).

Consequently, the theoretical computational cost of one LTW distance calculation is three times the length of the corresponding warping path. But it is important to understand that with zero warping the cost degrades to one times the length of the warping path, because our LTW distance only needs to evaluate the cells on the main diagonal, essentially returning the Euclidean distance.

Figure 4.2 gives an intuition for the minimum and maximum computational cost with and without warping constraint. Assuming that we want to measure the distance between two time series of length n, and knowing that the shortest possible warping path corresponds to the main diagonal of the similarity matrix, we expect LTW's greedy algorithm to take at least n iterative steps to compute the lucky warping path, leading to a minimum computational cost of $E_{LTW}(n) = 3n$ with warping window and $E_{LTW}(n) = n$ with zero warping.

Furthermore, Figure 4.2 illustrates that the longest possible warping path (that starts and finishes in diagonally opposite corner cells of the similarity matrix) equals the sum of the time series lengths, meaning the LTW algorithm exhibits a maximum computation cost of $E_{LTW}(n) = 3(n + n) = 6n$, which is essentially linear. Nevertheless, the worst case is likely to happen with and without warping window, since $E_{LTW}(n) = E_{cLTW}(n)$ as illustrated in Figure 4.2.

4.4 Empirical Evaluation

The goal of our comparative evaluation is twofold: (i) we investigate how well the LTW distance is suited for 1NN classification; and (ii) we assess the average amortized computational cost of the greedy algorithm for computing the LTW distance.

4.4.1 UCR Time Series

For our empirical evaluation we consider 43 datasets of UCR time series [95] with different length and number of classes, as well as different properties of noise, auto-correlation and stationarity. Each dataset is split into a training and test set, which is used for distance/parameter learning and evaluation respectively. A list of the examined UCR time series datasets and the corresponding classification results can be found in Figure 4.5.

4.4.2 Experimental Protocol

For the sake of comparison, all our experiments on classification performance involved the 1NN classifier. We compared the classification accuracy of the 1NN approach applying the following distance measures:

- cLTW - Lucky Time Warping distance with warping constraint (Sakoe-Chiba band)
- ED - Euclidean distance - baseline measure [87]
- DTW - Dynamic Time Warping distance without warping constraint (100% warping window)
- cDTW - Dynamic Time Warping distance with warping constraint (Sakoe-Chiba band) - state-of-the-art approach [154]

To compute the distances, we applied the greedy algorithm for cLTW and dynamic programming for DTW and cDTW. We learned the bandwidth of the Sakoe-Chiba band for cLTW and cDTW by leave-one-out cross validation on the training set of the respective dataset. We chose the warping window size that gives the highest classification accuracy.

Moreover, we compared the ACC of cLTW and cDTW_LB - constrained Dynamic Time Warping with lower bounding, in particular LB_Keogh [93] as formulated Section in 4.2.5. We considered ACCs instead of clock times to avoid implementation bias, caused by the disparity of algorithmic design of the competing approaches.

4.4.3 Classification Accuracy

Figure 4.3 illustrates the classification accuracy of cLTW in comparison to ED, DTW and cDTW. Each scatter plot in Figure 4.3 summarizes the classification results of two contrasted measures, where each data point demonstrates the achieved accuracy for one individual time series dataset. Data points that lie on the upper side of the main diagonal (highlighted in red) illustrate a superior classification accuracy of the LTW algorithm, whereas data points on the lower side of the main diagonal indicate better performance of the compared distance measure. In case that a data point is plotted on or very close to the main diagonal, we can infer that both measures achieved the same or similar results on the corresponding dataset.

As we can see in the first scatter plot of Figure 4.3, the majority of data points (37/43) lie on the upper side of the main diagonal, meaning cLTW outperforms ED in terms of classification accuracy. From the second plot in Figure 4.3 we can infer that cLTW performs slightly better than DTW without warping constraint, because almost all data points are arranged close to the main diagonal, some lie on the diagonal, and half of them (21/43) lie on the upper side. That cLTW is superior to DTW can be confirmed by the summary given in Table 4.1, which shows that cLTW has a lower average classification error. From the third plot in Figure 4.3 we can observe that more data points indicate superior performance of cDTW, although cLTW achieves better classification results on a considerable number (16/43 excluding ties) of the examined datasets.

The detailed classification results of the considered distance measures on all examined datasets can be found in Figure 4.5. A summary is presented in the following Table 4.1.

	min	max	avg	std
ED	0.0050	0.6580	0.2541	0.1510
DTW	0.0000	0.6230	0.2113	0.1550
cDTW	0.0015	0.6130	0.1911	0.1453
cLTW	0.0020	0.6299	0.2033	0.1550

Table 4.1: Minimum, maximum, average, and standard deviation of classification errors for all four considered measures.

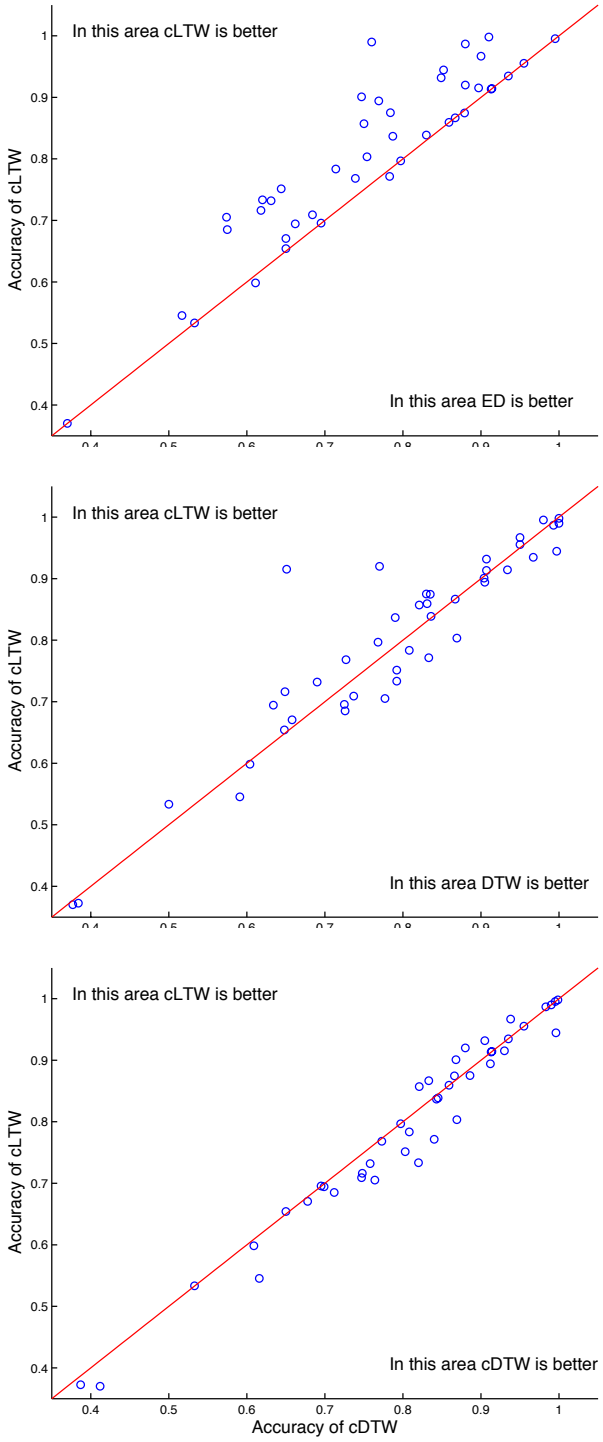


Figure 4.3: Classification accuracy of our introduced cLTW distance measure on all 43 considered time series datasets, listed in Figure 4.5, compared to ED, DTW and cDTW. In summary, cLTW achieved higher accuracy in 37/43, 21/43, and 16/43 cases, excluding ties that lie on the diagonal.

4.4.4 Amortized Computation Cost

Figure 4.4 shows the amortized computational cost (ACC) ratio of cDTW_LB to cLTW, where each data point compares the efficiency on one individual time series dataset. An ACC ratio of 1, indicated by the black solid line in Figure 4.4, means that both distance measures achieved the same efficiency or average amortized computational cost. Data points that lie above this baseline denote a higher efficiency or rather lower amortized computational cost of our cLTW distance. Figure 4.4 reveals that in some cases cLTW achieves an efficiency gain by one order of magnitude. On average cLTW is about 3.2 times faster than cDTW_LB, indicated by the black dashed line in Figure 4.4.

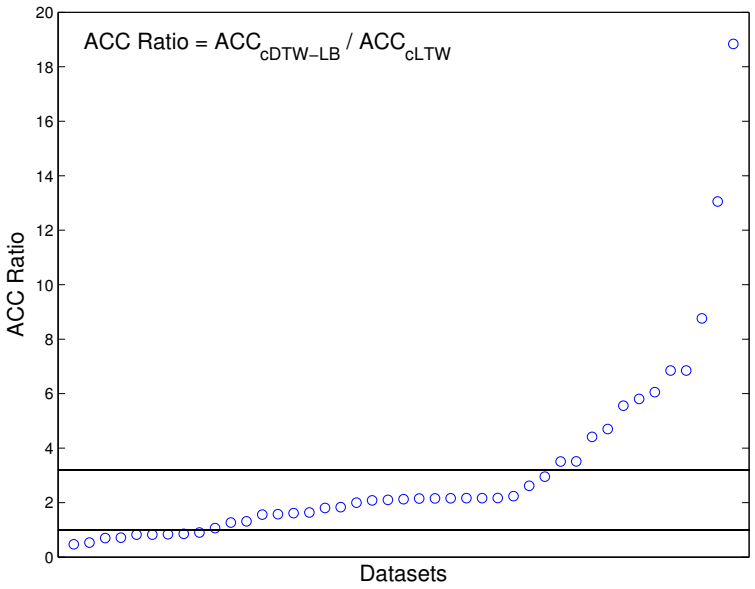


Figure 4.4: ACC ratio of cDTW_LB to LTW. Each data point demonstrates the contrasted efficiency on one time series dataset. The vertical axis shows how much LTW is faster than cDTW_LB. Results are arranged in ascending order along the horizontal axis. The black solid line indicates an ACC ratio of 1 or rather same efficiency. The average ACC ratio or speed-up of 3.2 is shown by the black dashed line.

Detailed results on the average ACC of cLTW and cDTW_LB on all examined time series datasets are listed in Figure 4.5. A summary of the ACC results is presented in Table 4.2.

	min	max	avg	std
cDTW_LB	$2.02n$	$92.75n$	$10.89n$	$15.69n$
cLTW	$0.96n$	$4.94n$	$3.29n$	$1.39n$

Table 4.2: Minimum, maximum, average, and standard deviation of ACC results for cDTW_LB and cLTW in $E(n) = f \times n$ notation, where n is the time series length and f is the respective constant factor.

The ACC results for cLTW are within the range of $n \leq E_{cLTW}(n) \leq 5n$, which corresponds to our theoretical minimum and maximum computational cost derived in Section 4.3.3. In contrast, the ACC results of cDTW_LB are within the range of $2n \leq E_{\text{cDTW-LB}}(n) \leq 93n$, depending on the size of the warping window and the amount of pruned DTW calculations (as stated in Equation 4.10).

Based on the presented ACC results we can infer that the efficiency of cDTW_LB varies strongly with the data under study, whereas cLTW is more robust in terms of required similarity operations. Due to the fact that the computational cost of cLTW does not depend on warping constraints or pruning power, it is an excellent choice for 1NN classification of arbitrary time series.

UCR DATASET	Classification Error in Percent of Misclassified Time Series 1NN-ED	1NN-DTW	1NN-cDTW (x%)	1NN-cLTW (x%)	ACC in E(n)=f*n Notation 1NN-cDTW-LB	1NN-cLTW
50words	0,3690	0,3100	**0,2420** (6)	0,2681 (3)	4,85 n	**3,82** n
Adiac	**0,3890**	0,3960	0,3910 (3)	0,4015 (3)	7,27 n	**3,46** n
Beef	**0,4670**	0,5000	**0,4670** (0)	**0,4667** (0)	2,15 n	**1,00** n
CBF	0,1480	**0,0030**	0,0040 (11)	0,0556 (3)	19,98 n	**4,53** n
ChlorineConcentration	0,3500	0,3520	0,3500 (0)	**0,3458** (2)	**2,02** n	4,28 n
CinC_ECG_torso	0,1030	0,3490	**0,0700** (1)	0,0848 (2)	10,12 n	**4,71** n
Coffee	0,2500	0,1790	0,1790 (3)	**0,1429** (2)	7,22 n	**4,01** n
Cricket_X	0,4260	**0,2230**	0,2360 (7)	0,2949 (3)	7,60 n	**4,87** n
Cricket_Y	0,3560	0,2080	**0,1970** (17)	0,2487 (2)	26,19 n	**4,71** n
Cricket_Z	0,3800	0,2080	**0,1800** (7)	0,2667 (4)	7,78 n	**4,94** n
DiatomSizeReduction	0,0650	**0,0330**	0,0650 (0)	0,0654 (0)	2,23 n	**1,00** n
ECG200	0,1200	0,2300	0,1200 (0)	**0,0800** (2)	**2,05** n	3,84 n
ECGFiveDays	**0,2030**	0,2320	**0,2030** (0)	**0,2033** (0)	2,15 n	**0,99** n
FaceAll	0,2860	**0,1920**	**0,1920** (3)	0,2166 (1)	5,88 n	**3,65** n
FaceFour	0,2160	0,1700	**0,1140** (2)	0,1250 (1)	11,05 n	**3,74** n
FacesUCR	0,2310	0,0951	**0,0880** (12)	0,1059 (3)	24,45 n	**4,04** n
FISH	0,2170	0,1670	**0,1600** (4)	0,2286 (2)	12,00 n	**3,41** n
Gun_Point	**0,0870**	0,0930	**0,0870** (0)	**0,0867** (0)	2,11 n	**0,99** n
Haptics	0,6300	0,6230	**0,5880** (2)	0,6299 (0)	5,80 n	**1,00** n
InlineSkate	0,6580	0,6160	**0,6130** (14)	0,6273 (1)	92,75 n	**4,92** n
ItalyPowerDemand	**0,0450**	0,0500	**0,0450** (0)	**0,0447** (0)	2,07 n	**0,96** n
Lighting2	0,2460	**0,1310**	**0,1310** (6)	0,1967 (4)	31,39 n	**4,58** n
Lighting7	0,4250	**0,2740**	0,2880 (5)	0,3151 (4)	11,75 n	**4,49** n
MALLAT	0,0860	**0,0660**	0,0860 (0)	0,0857 (0)	2,08 n	**1,00** n
MedicalImages	0,3160	0,2630	**0,2530** 20	0,2908 (2)	16,05 n	**3,41** n
MoteStrain	**0,1210**	0,1650	0,1340 (1)	0,1254 (1)	3,19 n	**2,99** n
OliveOil	**0,1330**	**0,1330**	0,1670 (1)	**0,1333** (0)	13,03 n	**1,00** n
OSULeaf	0,4830	0,4090	**0,3840** (7)	0,4545 (3)	14,54 n	**4,14** n
SonyAIBORobotSurface	0,3050	**0,2750**	0,3050 (0)	0,3045 (0)	2,14 n	**0,99** n
SonyAIBORobotSurfaceII	**0,1410**	0,1690	**0,1410** (0)	**0,1406** (0)	2,13 n	**0,98** n
StarLightCurves	0,1510	0,0930	0,0950 (16)	**0,0681** (23)	41,42 n	**4,73** n
SwedishLeaf	0,2130	0,2100	**0,1570** (2)	0,1632 (1)	**2,81** n	3,28 n
Symbols	0,1000	0,0500	0,0620 (8)	**0,0332** (9)	26,35 n	**3,85** n
synthetic_control	0,1200	**0,0070**	0,0170 (6)	0,0133 (11)	7,26 n	**4,43** n
Trace	0,2400	**0,0000**	0,0100 (3)	0,0100 (3)	**3,98** n	4,82 n
Two_Patterns	0,0900	**0,0000**	0,0015 (4)	0,0020 (7)	**2,93** n	4,11 n
TwoLeadECG	0,2530	**0,0960**	0,1320 (5)	0,0992 (7)	7,37 n	**4,02** n
uWaveGestureLibrary_X	0,2610	0,2730	**0,2270** (4)	0,2317 (3)	**3,07** n	3,40 n
uWaveGestureLibrary_Y	0,3380	0,3660	**0,3010** (4)	0,3057 (3)	**2,80** n	3,39 n
uWaveGestureLibrary_Z	0,3500	0,3420	**0,3220** (6)	0,3294 (1)	3,85 n	**2,93** n
wafer	0,0050	0,0200	0,0050 (1)	**0,0047** (1)	**2,08** n	2,97 n
WordsSynonyms	0,3820	0,3510	**0,2520** (8)	0,2837 (4)	7,60 n	**3,81** n
Yoga	0,1700	0,1640	**0,1550** (2)	0,1613 (1)	**2,68** n	3,20 n

Figure 4.5: Detailed experimental results on classification error and amortized computational cost (ACC) for our proposed LTW distance on all examined datasets. For the sake of completeness, we list the classification errors of ED, DTW and cDTW published on the UCR Time Series Classification/Clustering Page [95]. Still, all prior results were verified and reproduced by our own implementation. For each dataset the lowest achieved classification error and amortized computational cost are printed in bold font. Rows with gray background indicate the datasets were LTW achieved both lower classification error and lower average ACC. Please note that the respective best/learned warping window size for each dataset is listed in brackets behind the classification accuracies. The ACC results assume the same window size for each approach respectively.

4.4.5 Discussion

Our empirical evaluation reveals that in some cases LTW achieved higher classification accuracy than DTW and/or cDTW, meaning that an optimal warping path strategy may lead to suboptimal classification results. In other words, an optimal warping path, as returned by DTW, does not necessarily lead to a low classification error, and an approximate suboptimal warping path, as returned by LTW, may contribute to overall higher classification accuracy. Intuitively this statement can be justified by the fact that an optimal warping path does not necessarily minimize the classification error. Practically this statement can be confirmed by the finding [95] that ED is able to achieve higher classification accuracy than DTW and/or cDTW (see results for Adiac and MoteStrain dataset in Figure 4.5). Due to this statement, there is no point in computing the tightness of LTW in respect to DTW.

In addition to classification accuracy, computational complexity is a crucial decision criterion for choosing an appropriate time series distance measure. Our theoretical and empirical evaluation demonstrate that LTW is linear in respect to time series length, and furthermore performs pairwise distance calculations orders of magnitudes faster than DTW and cDTW, which exhibit quadratic complexity. Although cDTW in combination with lower bounding leads to quasi-linear complexity for querying large time series databases [154, 160], we have shown that this approach causes comparatively high ACCs for 1NN classification, where a multitude of unlabeled time series is compared against a set of unordered prototypes. Our empirical results on average amortized computational cost (ACC) demonstrate that our straightforward LTW approach without pruning is advantageous to 1NN classification for almost all examined time series datasets.

4.5 Conclusion and Future Work

To speed up 1NN classification of time series, we proposed the Lucky Time Warping (LTW) distance, which determines the warping path in a greedy manner. In this work we showed that LTW:

- calculates the similarity between a pair of time series in linear time and space,
- trades classification accuracy against computational cost reasonably well,
- is able to cope with time series that exhibit warping invariance,

- is straightforward and easy to implement,
- is unaffected by warping constraints,
- does not depend on the pruning power of lower bounding techniques;

making LTW not only a good choice for 1NN classification of arbitrary time series, but also an appropriate distance for other mining tasks.

Since this work only discusses a naive implementation of the LTW algorithm, there is potential for further improvements in speed, for instance by adopting early abandoning and online normalization techniques [154]. For instance, we can early abandon the computation of the lucky time warping distance if the current sum of the squared differences between each pair of data points that contribute the warping path exceeds the distance to the best-so-far neighbor.

Further improvements in classification accuracy might be achieved by learning a warping path that minimizes the classification problem, and not the time series distance function.

Chapter 5

Recurrence Plot-based Distance
for Time Series Clustering

Given a set of time series, our goal is to identify prototypes that cover the maximum possible amount of occurring subsequences regardless of their order. This scenario appears in the context of the automotive industry, where the objective is to determine operational profiles that comprise frequently recurring driving behavior patterns. This problem can be solved by clustering, however, standard distance measures such as the dynamic time warping distance might not be suitable for this task, because they aim at capturing the cost of aligning two time series rather than rewarding pairwise occurring patterns.

In this chapter, we propose a novel time series distance measure, based on the theoretical foundation of recurrence plots, which enables us to determine the (dis)similarity of multivariate time series that contain segments of similar trajectories at arbitrary positions. We use recurrence quantification analysis to measure the structures observed in recurrence plots and to investigate dynamical properties, such as determinism, which reflect the pairwise (dis)similarity of time series. In experiments on real-life test drives from Volkswagen, we demonstrate that clustering multivariate time series using the proposed recurrence plot-based distance measure results in prototypical test drives that cover significantly more recurring patterns than using the same clustering algorithm with dynamic time warping distance.

5.1 Introduction

Clustering of times series data is of pivotal importance in various applications [95] such as, for example, seasonality patterns in retail [104], electricity

usage profiles [120], DNA microarrays [140], and fMRI brain activity mappings [210]. A crucial design decision of any clustering algorithm is the choice of (dis)similarity function [9, 112]. In many clustering applications, the underlying (dis)similarity function measures the cost of aligning time series to one another. Typical examples of such functions include the DTW and the Euclidean distance [36, 87, 154].

Alignment-based (dis)similarity functions, however, seem not to be justified for applications, where two time series are considered to be similar, if they share common or similar subsequences of variable length at arbitrary positions [27, 117, 156, 217]. A real-life example for such an application comes from the automotive industry, where test drives of vehicles are considered to be similar, if they share similar driving behavior patterns, i.e. engine behavior or drive maneuvers, which are described by the progression of multiple vehicle parameters over a certain period of time [182, 184]. In this scenario, the order of the driving behavior patterns does not matter [179], but the frequency with which the patterns occur in the contrasted time series.

Recent work [69] on time series distance measures suggests to neglect irrelevant and redundant time series segments, and to retrieve subsequences that best characterize the real-life data. Although subsequence clustering is a tricky endeavor [88], several studies [27, 89, 117, 156, 217] have demonstrated that in certain circumstances ignoring sections of extraneous data and keeping intervals with high discriminative power contributes to cluster centers that preserve the characteristics of the data sequences. Related concepts that have been shown to improve clustering results include time series motifs [27, 117], shapelets [156, 217], and discords [89].

In this contribution, we propose to adopt recurrence plots (RPs) [128, 129, 132] and related recurrence quantification analysis (RQA) [130, 131, 133] to measure the similarity between multivariate time series that contain segments of similar trajectories at arbitrary positions in time [179]. We introduce the concept of joint cross recurrence plots (JCRPs), an extension of traditional RPs, to visualize and investigate multivariate patterns that (re)occur in pairwise compared time series. In dependence on JCRPs and known RQA measures, such as determinism, we define a **R**ecu**RR**ence plot-based (RRR) distance measure, which reflects the proportion of time series segments with similar trajectories or recurring patterns respectively.

In order to demonstrate the practicability of our proposed recurrence plot-based distance measure, we conduct experiments on both synthetic time series and real-life vehicular sensor data [179, 182, 184]. The results show that, unlike commonly used (dis)similarity functions, our proposed distance measure is able to (i) determine cluster centers that preserve the characteristics of the data sequences and, furthermore, (ii) identify prototypical time

series that cover a high amount of recurring patterns.

The rest of the paper is organized as follows. In Section 5.2 we state the general problem being investigated. Subsequently we introduce traditional recurrence plots as well as various extensions in Section 5.4. Recurrence quantification analysis and corresponding measures are discussed in Section 5.5. Our proposed recurrence plot-based distance measure and respective evaluation criteria are introduced in Section 5.6. The experiments results are presented and discussed in Section 5.8. Finally we conclude with future work in Section 5.9.

5.2 Problem Statement

Car manufacturers aim to optimize the performance of newly developed engines according to operational profiles that characterize recurring driving behavior. To obtain real-life operational profiles for exhaust simulations, Volkswagen (VW) collects data from test drives for various combinations of driver, vehicle and route.

Given a set $\mathbb{X} = \{X_1, X_2, \ldots, X_t\}$ of t test drives, the challenge is to find a subset of k prototypical time series $\mathbb{Y} = \{Y_1, \ldots, Y_k\} \in \mathbb{X}$ that best comprehend the recurring (driving behavior) patterns found in set $\mathbb{X}$. Test drives are represented as multivariate time series $X = \{x_1, \ldots, x_n\}$ of varying length n, where $x_i \in \mathbb{R}^d$ is a d-dimensional feature vector summarizing the observed measurements at time i. A *pattern* $S = \{x_i, \ldots, x_{i+l-1}\}$ of $X = \{x_1, \ldots, x_n\}$ is a subsequence of l consecutive time points from X, where $l \leq n$ and $1 \leq i < i + l - 1 \leq n$. Assuming two time series $X = \{x_1, ..., x_n\}$ and $Y = \{y_1, \ldots, y_m\}$ with patterns $S_x = \{x_i, \ldots, x_{i+l-1}\}$ and $S_y = \{y_j, \ldots, y_{j+l-1}\}$ of length l, we say that S_x and S_y are *recurring patterns* of X and Y if $d(S_x, S_y) \leq \epsilon$, where and $d : S \times S \rightarrow \mathbb{R}^+$ is a (dis)similarity function and ϵ is a certain similarity threshold. Note that recurring patterns of X and Y may occur at arbitrary positions and in different order.

Since we aim to identify k prototypical time series that (i) best represent the set $\mathbb{X}$ and (ii) are members of the set $\mathbb{X}$, one can employ the k-medoid clustering algorithm.

5.3 Related Work

The main goal of clustering is to organize unlabeled data into homogeneous groups that are clearly separated from each other. In general, clustering

involves the clustering algorithm, the similarity or rather distance measure, and the evaluation criterion. Clustering algorithms are categorized into partitioning, hierarchical, density-based, grid-based, and model-based methods. All of these clustering algorithms can be applied for static and temporal data [112]. In the following we discuss important considerations, common pitfalls, successful applications, and recent developments in time series clustering.

Time Series Clustering: Unlike static data, temporal data evolves over time and therefore requires special handling. One could either modify the existing clustering algorithms to handle time series data or convert the time series into a form that can be directly clustered. The former approach works with the raw time series, and the major modification lies in replacing the distance/similarity measure. The latter approach converts the raw time series either into feature vectors or model parameters, and then applies conventional clustering algorithms. Thus time series clustering approaches can be categorized into raw-data-based, feature-based, and model-based methods [112].

Time Series Representation: In this study we mainly focus on clustering methods that work with raw data, in particular multivariate time series with same sample rate. Clustering time series only differs from conventional clustering in how to compute the similarity between data objects [112]. Therefore, the key is to understand the unique characteristics of the time series and then to design an appropriate similarity measure accordingly. For instance, Meesrikamolkul et al. [138] have proposed a novel method which combines the widely used k-means clustering algorithm with the Dynamic Time Warping distance measure, instead of the traditional Euclidean distance, to study sequences with time shifts. Unlike before, the new method determines cluster centers that preserve the characteristics of the data sequences.

Distance/Similarity Measures: Besides Euclidean distance and Dynamic Time Warping distance, commonly used similarity measures include Minkowski distance, Levenshtein distance, Short Time Series distance, Pearson correlation coefficient, cross-correlation-based distances, probability-based distance functions, and many others. The choice of similarity measure depends on whether the time series is discrete-valued or real-valued, uniform or non-uniform sampled, univariate or multivariate, and whether the data sequences are of equal or unequal length [112].

Distortions and Invariance: Furthermore, the choice of the time series distance measure depends on the invariance required by the domain. The literature [9] has introduced techniques designed to efficiently measure similarity between time series with invariance to (various combinations of) the distortions of warping, uniform scaling, offset, amplitude scaling, phase, occlusions, uncertainty and wandering baseline. Recent work [179] has proposed an order-invariant distance which is able to determine the (dis)similarity of time series that exhibit similar subsequences at arbitrary positions. The authors demonstrate that order invariance is an important consideration for domains such as automotive engineering and smart home environments [182, 184], where multiple sensors observe contextual patterns in their naturally occurring order, and time series are compared according the occurrence of these multivariate patterns.

Evaluation Criterion: Evaluation criteria for clustering are distinguished between known ground truth and unknown ground truth [112]. In case of known ground truth, the similarity between known clusters and obtained clusters can be measured. The most commonly used clustering quality measure for known ground truth is the Rand Index or minor variants of it [217]. In contrast, without prior knowledge the clusters are usually evaluated according their within-cluster similarity and between-cluster dissimilarity [112]. Various validity indices have been proposed to determine the number of clusters and their goodness. For instance, the index I has been found to be consistent and reliable, irrespective of the underlying clustering technique and data dimensionality, and furthermore has been shown to outperform the Dunn and David-Bouldin index [137].

Realistic Assumptions: The majority of publicly available time series datasets were preprocessed and cleaned before publishing. For instance, the UCR archive [95] contains only time series with equal length, which are mostly snippets of the original data that were retrieved manually. The publication of perfectly aligned pattens of equal length has lead to huge amount of time series classification and clustering algorithms that are not able to deal with real-world data, which contains irrelevant sections. Hu et al. [69] suggest to automatically build a data dictionary, which contains only a small subset of the training data and neglects irrelevant sections and redundancies. The evaluations shows that using a data dictionary with a set of retrieved subsequences for each class leads to higher classification accuracy and is several time faster than the compared strawman algorithms. However, one needs to be careful about how to retrieve subsequences, for reasons explained in the following.

Subsequence Clustering: Keogh and Lin [88] state that the clustering of time series subsequences is meaningless, referring to the finding that the output does not depend on input, and the resulting cluster centers are close to random ones. In almost all cases the subsequences are extracted with a sliding window, which is assumed to the quirk in clustering. To produce meaningful results the authors suggest to adopt time series motifs, a concept highly related to clusters. Their experiments demonstrate that motif-based clustering is able to preserve the patterns found in the original time series data [88].

Time Series Motifs: Motifs are previously unknown, frequently occurring patterns, which are useful for various time series mining tasks: such as summarization, visualization, clustering and classification of time series [27, 117]. According to the definition [117] a time series motif is a subsequence that comprises all non-trivial matches within a given range. Since the naive (brute-force) approach to motif discovery has quadratic complexity, Lin et al. [117] introduce a new motif discovery algorithm that provides fast exact answers, and faster approximate answers, achieving a speedup of one to two orders of magnitude. In order to reduce the number of possible candidates of motifs, Chiu et al. [27] propose to omit consecutive subsequences that resemble each other. Furthermore, the set of subsequences in each motif should be mutually exclusive, because otherwise the motifs would be essentially the same. Although normalization techniques are commonly applied to compare time series with different offset and amplitude, Chiu et al. [27] state that these are important characteristics that might proof to be useful to distinguish motifs, because after normalization most subsequences correspond to almost the same upward or downward trend and become indistinguishable.

Time Series Shapelets: Most existing methods for time series clustering rely on distances calculated on the shape of the signals. However, time series usually contain a great amount of measurements that do not contribute to the differentiation task or even decrease cluster accuracy. Hence, to cluster time series, we are generally better off ignoring large sections of extraneous data and keeping intervals with high discriminative power. Recent work [156, 217] proposes to use local patterns, so called shapelets, to cluster time series databases. According to the definition [217], a shapelet is a time series snippet that can separate and remove a subset of the data from the rest of the database while maximizing the separation gap or rather information gain. Although the experiments demonstrate that shapelet-based clustering gives better results than statistical-based clustering of the entire time series,

finding optimal shapelets is a nontrivial task, and almost certainly harder than the clustering itself [217]. However, the results underline the importance of ignoring some data to cluster time series in real world applications under realistic settings.

Time Series Discords: Different from motifs or shapelets, time series discords are subsequences of longer time series that are most unusual or rather maximally different to all the rest of the time series subsequences. Keogh et al. [89] have shown that time series discords are particularly attractive as anomaly detectors because they only require one intuitive parameter, namely the length of the subsequences. Furthermore, discords have implications for the time series clustering, cleaning and summarization.

Time Series Prototypes: To sum up, the concepts that may possibly be adapted to identify time series prototypes (as described in our problem statement in Section 5.2) include motifs [27, 117] and shapelets [156, 217]. However, in both cases this would require major modifications of the existing algorithm. A straightforward approach to solve the stated problem is presented in the following sections.

5.4 Recurrence Plots

Recurrence plots (RPs) are used to visualize and investigate recurrent states of dynamical systems or rather time series [130, 170]. Even though RPs give very vivid and impressive images of dynamical system trajectories, their implicit mathematical foundation is deceptively simple [128]:

$$R_{i,j}^{d,\epsilon} = \Theta(\epsilon - ||x_i - x_j||) \qquad x_i \in \mathbb{R}^d, \quad i,j = 1 \dots n \tag{5.1}$$

where $X = \{x_1, \dots, x_n\}$ is a d-dimensional time series of length n, $||\cdot||$ a norm and Θ the Heaviside function. One of the most crucial parameters of RPs is the recurrence threshold ϵ, which influences the formation of line structures [132]. In general, the recurrence threshold should be chosen in a way that noise corrupted observations are filtered out, but at the same time a sufficient number of recurrence structures are preserved. As a rule of thumb, the recurrence rate should be approximately one percent with respect to the size of the plot. For quasi-periodic processes, it has been suggested to use the diagonal line structures to find the optimal recurrence threshold. However, changing the threshold does not preserve the important distribution of recurrence structures [130].

A general problem with standard thresholding methods is that an inappropriate threshold or laminar states cause thick diagonal lines, which basically corresponds to redundant information. Schultz et al. [170] have proposed a local minima-based thresholding approach, which can be performed without choosing any particular threshold and yields in clean RPs of minimized line thickness. But this approach comes with some side effects, e.g., bowed lines instead of straight diagonal lines.

Furthermore, it is important discuss the definition of recurrences, because distances can be calculated using different norms [128]. Although the L^2-norm is used in most cases, the L^∞-norm is sometimes preferred for relatively large time series with high computational demand [130].

Although traditional RPs only regard one trajectory, we can extend the concept in a way that allows us to study the dynamics of two trajectories in parallel [129]. A cross recurrence plot (CRP) shows all those times at which a state in one dynamical system occurs in a second dynamical system. In other words, the CRP reveals all the times when the trajectories of the first and second time series, X and Y, visits roughly the same area in the phase space. The data length, n and m, of both systems can differ, leading to a non-square CRP matrix [131, 132].

$$CR_{i,j}^{d,\epsilon} = \Theta(\epsilon - ||x_i - y_j||) \qquad x_i, y_j \in \mathbb{R}^d, \quad i = 1 \dots n, \quad j = 1 \dots m \tag{5.2}$$

For the creation of a CRP both trajectories, $X = \{x_1, \dots, x_n\}$ and $Y = \{y_1, \dots, y_m\}$, have to present the same dynamical system with equal state variables because they are in the same phase space. The application of CRPs to absolutely different measurements, which are not observations of the same dynamical system, is rather problematic and requires some data preprocessing with utmost carefulness [132].

In order to test for simultaneously occurring recurrences in different systems, another multivariate extension of RPs was introduced [129]. A joint recurrence plot (JRP) shows all those times at which a recurrence in one dynamical system occurs simultaneously with a recurrence in a second dynamical system. With other words, the JRP is the Hadamard product of the RP of the first system and the RP of the second system. JRPs can be computed from more than two systems. The data length of the considered systems has to be the same. [131, 132].

$$\begin{aligned} JR_{i,j}^{d,\epsilon} &= \Theta(\epsilon^x - ||x_i - x_j||) \cdot \Theta(\epsilon^y - ||y_i - y_j||) \\ \epsilon &= \{\epsilon^x, \epsilon^y\}, \quad x_i \in \mathbb{R}^{d1}, \quad y_j \in \mathbb{R}^{d2}, \quad i, j = 1 \dots n \end{aligned} \tag{5.3}$$

Such joint recurrence plots have the advantage, that the individual measurements can present different observables with different magnitudes or range. They are often used for the detection of phase synchronization [131, 132].

Since this work aims at clustering test drives, which involves pairwise (dis)similarity comparisons of multivariate time series, we propose a combination of joint and cross recurrence plot, namely (JCRP) joint cross recurrence plot. A JCRP shows all those times at which a multivariate state in one dynamical system occurs simultaneously in a second dynamical system.

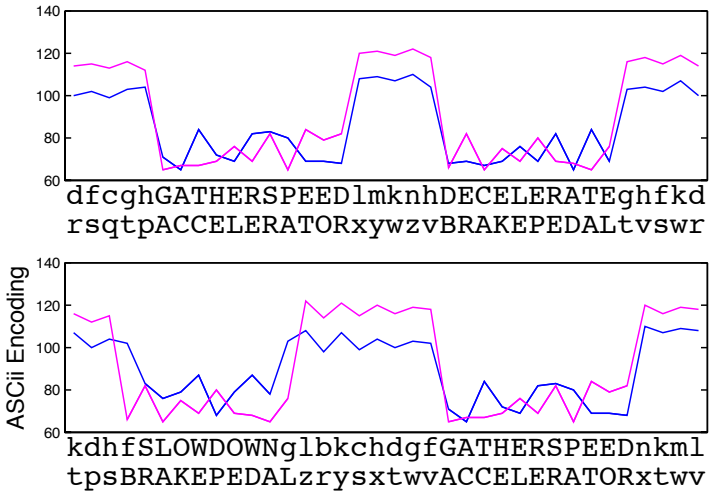
$$JCR_{i,j}^{d,\epsilon} = \Theta(\epsilon^1 - ||x_i^1 - y_j^1||) \cdot \ldots \cdot \Theta(\epsilon^d - ||x_i^d - y_j^d||) \tag{5.4}$$
$$\epsilon, x_i, y_j \in \mathbb{R}^d, \quad i = 1 \ldots n, \quad j = 1 \ldots m$$

For the creation of a JRCP both trajectories, $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_m\}$, need to have the same dimensionality or number of parameters d, but can have different length, n and m. We shall see that JCRPs are very useful, because they enable us to compare two multivariate systems with the same set of variables, each having different amplitude. In other words, the introduced JCR notation allows us to determine an ϵ-threshold for each individual parameter, which is advantageous if we want to analyze different physical quantities within there natural unit or scope. A toy example for JCRPs is given in the following:

$$X = \begin{cases} \texttt{dfcghGATHERSPEEDlmknhDECELERATEghfkd} \\ \texttt{rsqtpACCELERATORxywzvBRAKEPEDALtvswr} \end{cases}$$
$$Y = \begin{cases} \texttt{kdhfSLOWDOWNglbkchdgfGATHERSPEEDnkml} \\ \texttt{tpsBRAKEPEDALzrysxtwvACCELERATORxtwv} \end{cases}$$

Assume two multivariate time series X and Y which comprise the speed and accelerator signal recorded during different car drives. Both time series contain multivariate states or rather string sequences that occur in both systems, as demonstrated in Figure 5.1(a). The corresponding JCRP of X and Y, as illustrated in Figure 5.1(b), shows the times at which a multivariate state occurs simultaneously in both systems. Furthermore, the diagonal line structure in Figure 5.1(b) reveals that both trajectories run through a similar region in phase space for a certain time interval. With other words, both systems contain the same multivariate pattern, which represents that the driver hits the 'ACCELERATOR' pedal and the vehicle simultaneously 'GATHERSPEED'. In Section 5.5 we discuss how to interpret single recurrence points and diagonal line structures, and explain how to use them to define a distance measure for time series with certain distortions or invariance.

(a)



(b)

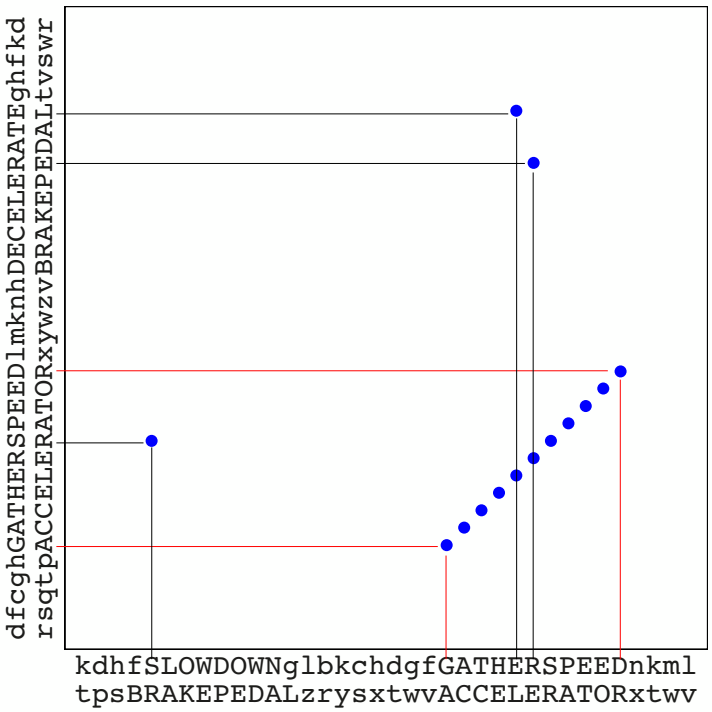


Figure 5.1: (a) ASCii decimal encoding of two multivariate time series X and Y which contain the same pattern or string sequence at different positions in time. (b) Joint cross recurrence plot (JCRP) of time series X and Y, introduced in Figure 5.1(a), with $\epsilon = 0$. The diagonal line structure in the recurrence plot indicates the existence and position of a co-occurring multivariate pattern. The single recurrence points can be considered as noise.

5.5 Recurrence Quantification

Recurrence quantification analysis (RQA) is used to quantify the structures observed in recurrence plots [132]. RQA is grounded in theory, but possesses statistical utility in dissecting and diagnosing nonlinear dynamic systems across multiple fields of science [204]. The explicit mathematical definition to distinct features in recurrence plots enables us to analyze signals that are multivariate, nonlinear, non-stationary and noisy.

The global (large-scale) appearance of a RP can give hints on stationarity and regularity, whereas local (small-scale) patterns are related to dynamical properties, such as determinism [204]. Recent studies have shown that determinism, the percentage of recurrence points that form lines parallel to the main diagonal, reflects the predictability of a dynamical system [132].

Given a recurrence matrix $R^{d,\epsilon}$ with $n \times n$ entries generated by any of the introduced recurrence plot variations, such as our proposed JCRP, we can compute the determinism DET for a predefined ϵ-threshold and a minimum diagonal line length l_{min} as followed [131, 132]:

$$DET = \frac{\sum_{l=l_{min}}^{n} l \cdot P^{\epsilon}(l)}{\sum_{i,j=1}^{n} R_{i,j}^{d,\epsilon}} \tag{5.5}$$

where $P^{\epsilon}(l)$ is the histogram of diagonal lines of length l with respect to a certain ϵ neighborhood.

$$\begin{aligned} P^{\epsilon}(l) = \sum_{i,j=1}^{n} \Big\{ & \left(1 - R_{i-1,j-1}^{d,\epsilon}\right) \\ & \cdot \left(1 - R_{i+l,j+l}^{d,\epsilon}\right) \\ & \cdot \prod_{k=0}^{l-1} R_{i+k,j+k}^{d,\epsilon} \Big\} \end{aligned} \tag{5.6}$$

In general, processes with chaotic behavior cause none or short diagonals, whereas deterministic processes cause relatively long diagonals and less single, isolated recurrence points [132, 202]. In respect to JCRPs, diagonal lines usually occur when the trajectory of two multivariate time series segments is similar according to a certain threshold. Since we aim to measure the similarity between time series that contain segments of similar trajectories at arbitrary positions, which in turn cause diagonal line structures, we propose to use determinism as a similarity measure. According to the introduced JCRP approach, a high DET value indicates high similarity or rather a high

percentage of multivariate segments with similar trajectory, whereas a relatively low DET value suggests dissimilarity or rather the absence of similar multivariate patterns.

However, data preprocessing like smoothing can introduce spurious line structures in a recurrence plot that cause high determinism value. In this case, further criteria like the directionality of the trajectory should be considered to determine the determinism of a dynamic system, e.g. by using iso-directional and perpendicular RPs [130, 131, 132]. In contrast to traditional recurrence plots, perpendicular recurrence plots (PRPs) consider the dynamical evolution of only the neighborhoods in the perpendicular direction to each phase flow, resulting in plots with lines of the similar width without spreading out in various directions. Removing spurious widths makes it more reasonable to define line-based quantification measures, such as divergence and determinism [28]. Another solution is to estimate the entropy by looking at the distribution of the diagonal lines [130]. The entropy is based on the probability $p^{\epsilon}(l)$ that diagonal lines structures with certain length l and ϵ-similarity occur in the recurrence matrix [131, 132], and can be computed as follows:

$$ENTR = -\sum_{l=l_{min}}^{n} p^{\epsilon}(l)\ \ln\ p^{\epsilon}(l) \tag{5.7}$$

Recurrence plots (RPs) and corresponding recurrence quantification analysis (RQA) measures have been used to detect transitions and temporal deviations in the dynamics of time series. Since detected variations in RQA measures can easily be misinterpreted, Marwan et al. [133] have proposed to calculate a confidence level to study significant changes. They formulated the hypothesis that the dynamics of a system do not change over times, and therefore the RQA measures obtained by the sliding window technique will be normally distributed. Consequently, if the RQA measures are out of a predefined interquantile range, an observation can be considered significantly. Detecting changes in dynamics by means of RQA measures obtained from a sliding window have been proven to be useful in real-life applications such as comparing traffic flow time series under fine and adverse weather conditions [202].

Since recurrence plot based techniques are still a rather young field in nonlinear time series analysis, systematic research is necessary to define reliable criteria for the selection of parameters, and the estimation of RQA measures [130].

5.6 Recurrence Plot-based Distance

According to our formalization of joint cross recurrence (JCR) in Equation 5.4 and the denotation of the determinism (DET) in Equation 5.5, we can define our RecuRRence Plot-based (RRR) distance measure as follows:

$$RRR = 1 - DET \tag{5.8}$$

In theory the DET value ranges from 0 to 1, depending on the proportion of diagonal line structures found in the corresponding JCR plot. However, in practice a JCR plot typically exhibits noise or individual recurrence points that do not contribute the diagonal lines. Consequently, the RRR distance is only close to 0 or 1, if the trajectories are similar or dissimilar respectively, and does not satisfy the conditions of a metric.

Although our proposed RRR distance measure can be used as a subroutine for various time series mining tasks, this work primarily focuses on clustering. Our aim is to group a set of t unlabeled time series $\mathbb{X} = \{X_1, \ldots, X_t\}$ into k clusters $\mathbb{C} = \{C_1, \ldots, C_k\}$ with corresponding centroids $\mathbb{Z} = \{Z_1, \ldots, Z_k\}$. In order to evaluate the performance of the time series clustering with respect to our RRR distance, we suggest to quantify the number of similar patterns that recur within the established clusters. Therefore, we define our cost function as the following cluster validation index:

$$E(k) = \frac{1}{t-k} \sum_{Z \in \mathbb{Z}} \sum_{X \in \{C_z \setminus Z\}} RRR(X, Z) \tag{5.9}$$

According to our problem setting, the more patterns occur jointly when comparing each centroid $Z \in \mathbb{Z}$ with all objects $X \in \{C_z \setminus Z\}$ of the corresponding cluster, the lower E, the better our clustering, and the more characteristic are the corresponding prototypes.

Furthermore we are going to evaluate the clustering of time series according to the index I [137], whose value is maximized for the optimal number of clusters:

$$I(k) = (\frac{1}{k} \cdot \frac{E(1)}{E(k)} \cdot D_k)^{\rho} \tag{5.10}$$

The index I is a composition of three factors [137], namely $1/k$, $E(1)/E(k)$, and D_k. The first factor will try to reduce index I as the number of clusters k increases. The second factor consists of the ratio of $E(1)$, which is constant for a given dataset, and $E(k)$, which decreases with increase in k. Consequently, index I increases as $E(k)$ decreases, encouraging more clusters that are compact in nature. Finally, the third factor, D_k (which measures the

maximum separation between two clusters over all possible pairs of clusters), will increase with the value of k, but is bounded by the maximum separation between two points in the dataset.

$$D_k = \max_{i,j=1}^{k} ||Z_i - Z_j|| \tag{5.11}$$

Thus, the three factors are found to compete with and balance each other critically. The power ρ is used to control the contrast between the different cluster configurations. Previous work [137] suggests to choose $\rho = 2$.

The index I has been found to be consistent and reliable, irrespective of the underlying clustering technique and data dimensionality, and furthermore has been shown to outperform the Dunn and David-Bouldin index [137].

5.7 Dimensionality Reduction

As with most problems in computer science, the suitable choice of representation greatly affects the ease and efficiency of time series data mining [116]. Piecewise Aggregate Approximation (PAA), a popular windowed averaging technique, reduces a time series X of length n to length n/r by dividing the data into r equal sized frames. The mean value of the data falling within a frame is calculated and a vector or time series Y of these values becomes the data-reduced representation.

$$Y_j = \frac{r}{n} \sum_{i=\frac{n}{r}(j-1)+1}^{\frac{n}{r}j} X_i \tag{5.12}$$

$$i = 1 \ldots n, \quad j = 1 \ldots r$$

The PAA dimensionality reduction is intuitive and simple, yet has been shown to rival more sophisticated dimensionality reduction techniques like Fourier transforms and wavelets [116]. Having transformed a time series database into PAA, we can apply our proposed recurrence plot-based time series distance measure on the reduced representation. Since the computational complexity of our RRR distance measure is quadratic in the length n of the time series, reducing the original time series to r dimensions leads to a performance improvement of factor $(n/r)^2$. In our experiments on the real-life vehicular data we use a compression rate of $n/r = 10$, which correspond to a speedup of two orders of magnitude or rather 100 times less matrix entries to compute. However, this approach comes with the cost of missing recurrences [130].

Another approach to reduce the computational complexity of our proposed recurrence plot-based (RRR) time series distance measure is to constrain the number of cells that are evaluated in the distance matrix [167]. Constraints have been successfully applied to the Dynamic Time Warping (DTW) distance to create tight lower bounds which allow to prune similarity calculations [92, 93]. The two most commonly used constraints are the Itakura Parallelogram [74] and the Sakoe-Chiba Band [165], which both speed up calculations by a constant factor, but still lead to quadratic complexity if the window size r is a linear function of the time series.

Given the formal definition of (joint) cross recurrence (see Equation 5.2 and 5.4), the Sakoe-Chiba Band is an adjustment window condition which corresponds to the fact that time-axis fluctuations in usual cases never causes a too excessive timing difference [165]:

$$|i - j| \leq r \tag{5.13}$$
$$\forall\, x_i, y_j \in \mathbb{R}^d, \quad i = 1 \ldots n, \quad j = 1 \ldots m$$

In general, constraints work well in domains where time series have only a small variance, but perform poorly if time series are of events that start and stop at radically different times [167]. Since this study considers time series that exhibit recurring patterns at arbitrary positions, we refrain from applying constraints for the data under study.

5.8 Evaluation

The goal of our evaluation is to assess how well the RRR distance is suited for: (i) calculating the similarity between time series with order-invariance (in Section 5.8.1), (ii) clustering time series that contain similar trajectories at arbitrary positions (in Section 5.8.2), and (ii) identifying prototypical time series that cover as much as possible patterns which co-occur in other sequences of the dataset (in Section 5.8.3).

5.8.1 Order-Invariance

In this section we demonstrate the practicality of our proposed *RRR* distance on a sample dataset of synthetic time series. As illustrated in Figure 5.2(a), we consider four different normally distributed pseudo-random time series with artificially implanted sinus patterns. The first two time series comprise the same subsequences in reverse order, whereas the last two time series contain a subset of the artificially implanted signals.

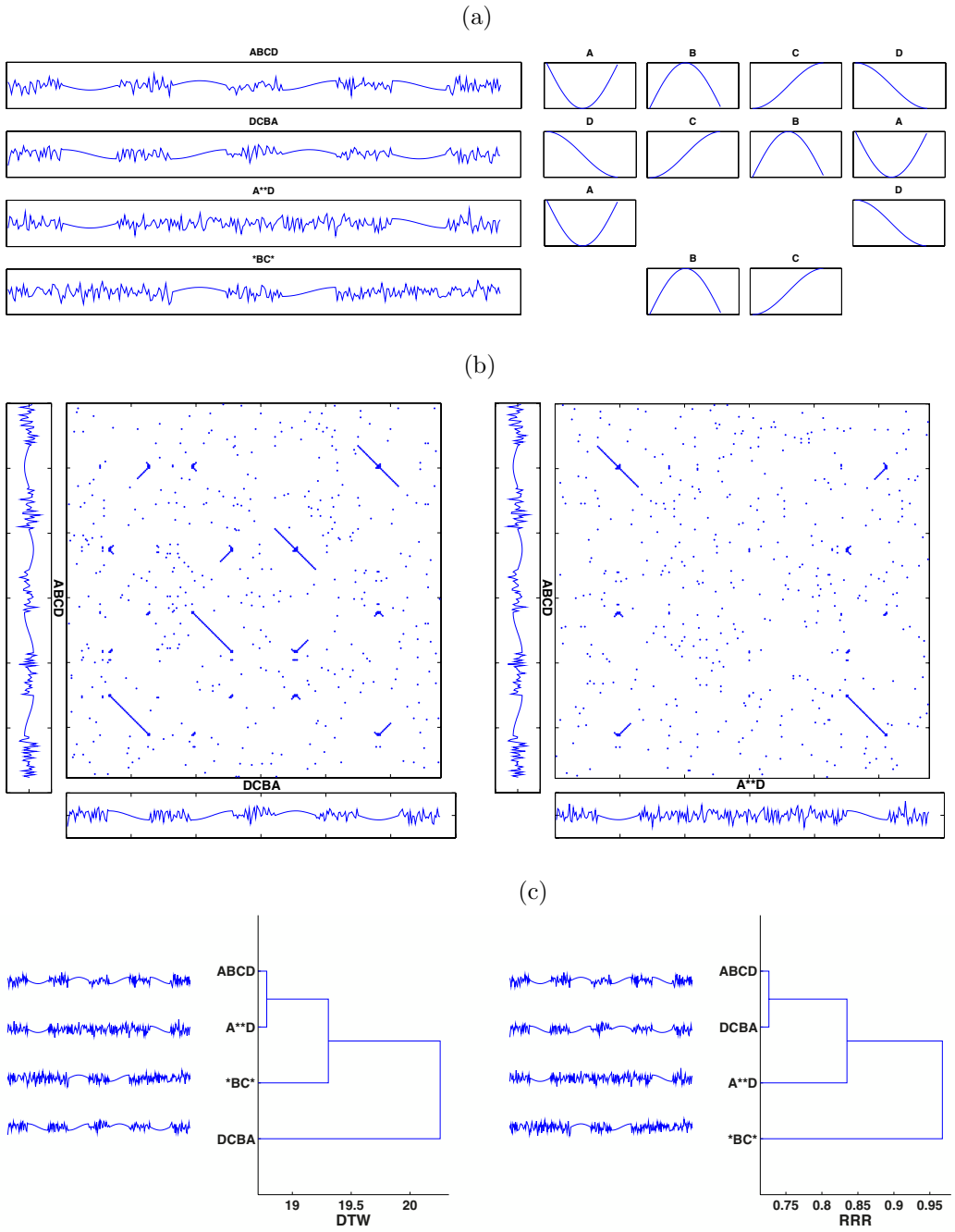
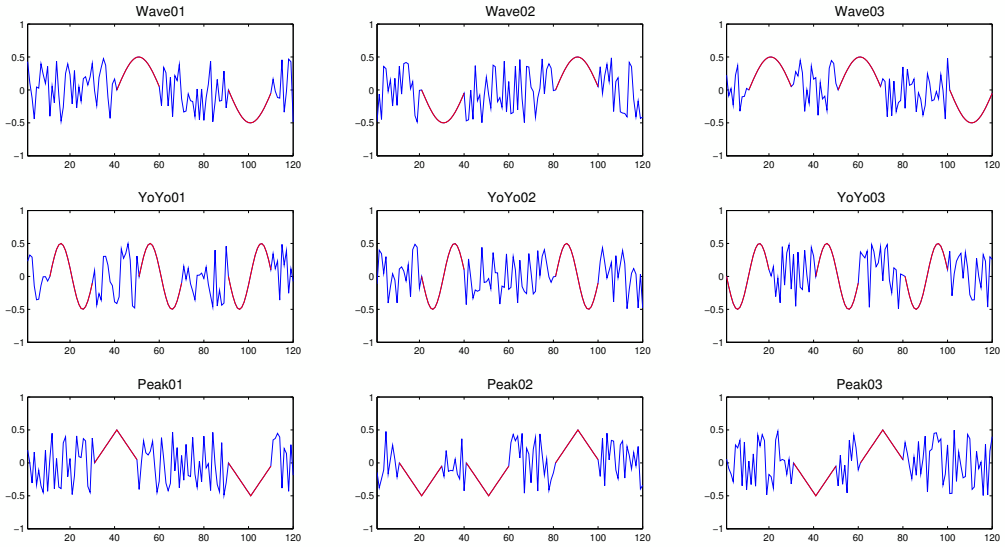


Figure 5.2: (a) Sample dataset of normally distributed pseudo-random time series (named as ABCD, DCBA, A**D and *BC*, illustrated left) with artificially implanted sinus patterns (labeled as A to D, presented in their occurring order on the right). (b) Cross Recurrence Plot (CRP) of synthetic time series ABCD and DCBA (left) as well as ABCD and A**D (right) introduced in Fig. 5.2(a). Note that the main diagonal runs from upper left to bottom right. (c) Agglomerative hierarchical cluster tree (dendrogram) of synthetic time series data (introduced in Fig. 5.2(a)) according to the DTW distance (left) and our proposed RRR distance (right), where the x-axis reveals the distance between the time series being merged and the y-axis illustrates the corresponding name and shape of the signal.

(a)



(b)

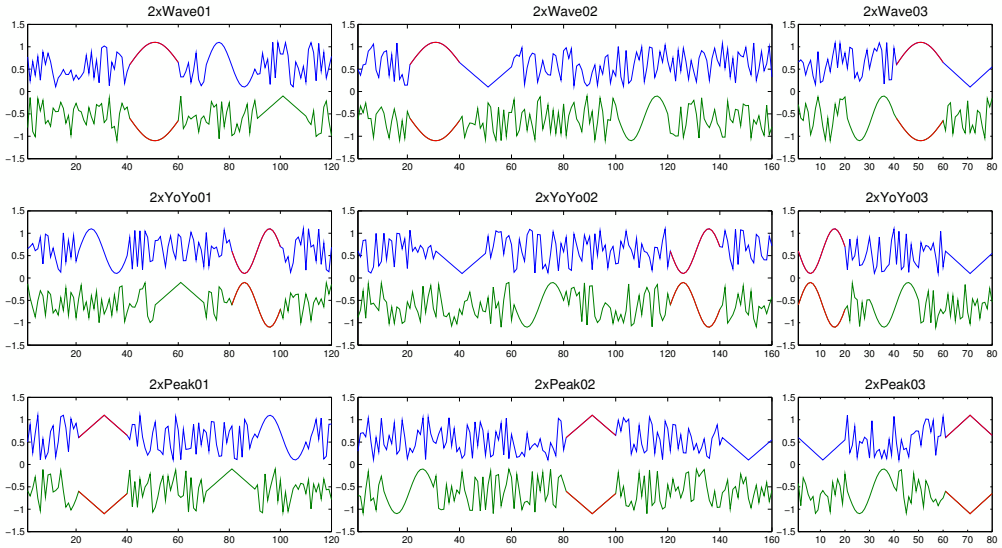


Figure 5.3: Univariate (a) and multivariate (b) synthetic time series with artificially implanted patterns (red color) at arbitrary positions, where each time series belongs to one of three groups (Wave, YoYo, and Peak).

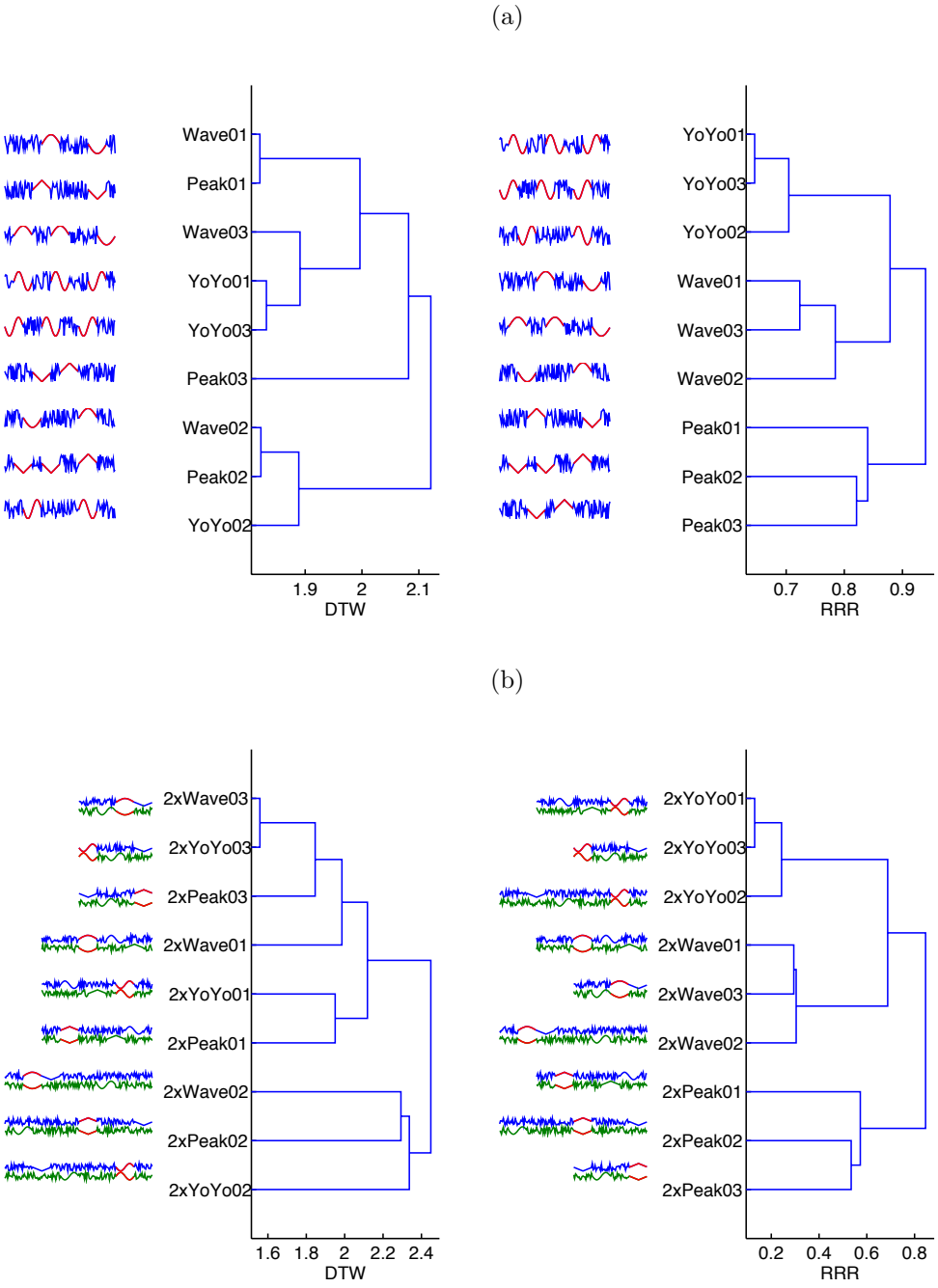


Figure 5.4: Cluster tree (dendrogram) of univariate (a) and multivariate (b) synthetic time series (introduced in Figure 5.3) according to the DTW and RRR distance. The x-axis reveals the distance between the time series being merged and the y-axis illustrates the corresponding name and shape of the time series.

Figure 5.2(b) illustrates the cross recurrence plot (CRP) of time series ABCD and DCBA as well as ABCD and A**D introduced in Figure 5.2(a). Lines parallel to the main diagonal (from upper left to bottom right corner) indicate similar subsequences in both time series. The percentage of recurrence points that form diagonal lines is much higher in the CRP of the time series ABCD and DCBA than in the CRP of the pair ABCD and A**D. As discussed in Section 5.6, we quantify the local small-scale structures in the recurrence plots by means of the determinism DET (refer to Equation 5.5).

Figure 5.2(c) shows a direct comparison of *Dynamic Time Warping* and our introduced RRR distance measure. As expected, the hierarchical cluster tree generated by means of DTW indicates a relatively small distance between the time series ABCD, A**D and *BC*, because they exhibit similar subsequences at the same positions. However, DTW treats the time series DCBA as an outlier, because the artificially implanted patterns occur in reverse order and cross-alignment is prevented. In contrast, the RRR measure considers the time series ABCD and DCBA as most similar, as the order of the matched patterns is disregarded. Furthermore, the dendrogram generated by means of RRR reveals that the time series A**D and *BC* are dissimilar to ABCD and DCBA, which is due to the fact that the overlap of same or similar subsequences is relatively small ($\leq 50\%$).

The results presented in Figure 5.2 serve to demonstrate that the proposed RRR distance measure is able to handle time series with order-invariance. In the following we investigate the capability of our RRR measure to cluster time series which exhibit same or similar subsequences at arbitrary positions in time.

5.8.2 Synthetic Data

This controlled experiment aims at visualizing the clustering results of the proposed RRR distance measure compared to the DTW distance.

We generated a labeled dataset, which consists of nine time series from three different categories, called Wave, YoYo and Peak. Each category comprises three time series characterized by multiple occurrence of the same artificial patterns at arbitrary positions. The dataset consists of univariate time series of equal length, as shown in Figure 5.3. To visualize the clustering results of the RRR and DTW distance, we applied agglomerative hierarchical clustering with complete linkage on the synthetic dataset.

Figure 5.4 illustrates the generated hierarchical cluster trees for both examined distance measures on the synthetic time series. The first observation to be made is that RRR perfectly recovers the cluster structure provided by the ground truth, given our knowledge that there are three categories. In

contrast, the DTW distance fails and assigns time series of different categories to the same cluster at an early stage. The second observation to be made is that RRR is able to recover the ground truth even if a large portion of the time series is noisy. The DTW distance, however, groups time series into the same clusters, if they have globally a similar shape. Therefore, the noisy parts of the time series supersede or superimpose the relevant recurring patterns.

5.8.3 Real-Life Data

This experiment aims at assessing the time series prototypes identified by the proposed RRR distance measure compared to the DTW distance.

For our evaluation we consider the VW DRIVE dataset, which consists of 124 real-life test drives recorded by one vehicle operated by seven different individuals. Test drives are represented as multivariate time series of varying length and comprise vehicular sensor data of the same observed measurements. Since we aim to identify operations profiles that characterize recurring driving behavior, we exclusively consider accelerator, speed, and revolution measurements, which are more or less directly influenced by the driver. The complete VW DRIVE dataset contains various other measurements, such as airflow and engine temperature, and can be obtained by mailing the first author of this paper.

To measure the (dis)similarity of the VW DRIVE time series using our proposed RRR distance, we first need to determine the optimal similarity threshold ϵ and pattern length l_{min} for each of the considered measurements, such that a considerable amount of the recurring patterns is preserved.

Figure 5.5 shows the determinism value for the accelerator, speed, and revolution signal, in regard to different parameters settings. We can observe that for all considered signals the DET value decreases with increasing pattern length l_{min} and decreasing similarity threshold ϵ. Furthermore, Figure 5.5 reveals that the speed signal is highly deterministic, meaning that the same patterns occur frequently, whereas the acceleration and revolution signal are less predictable and show more chaotic behavior.

Since we aim to analyze all signals jointly by means of the proposed joint cross recurrence plot (JCRP) approach, we have to choose a pattern length or rather minimum diagonal line length l_{min} that is suitable for all signals. In general, we are looking for relatively long patterns with high similarity. In other words, we aim to find a parameter setting with preferably large l_{min} and small ϵ which results in a DET value that is above a certain threshold. To preserve the underlying characteristics or rather recurring patterns contained in examined data, at least 20% of the recurrence points should form diagonal

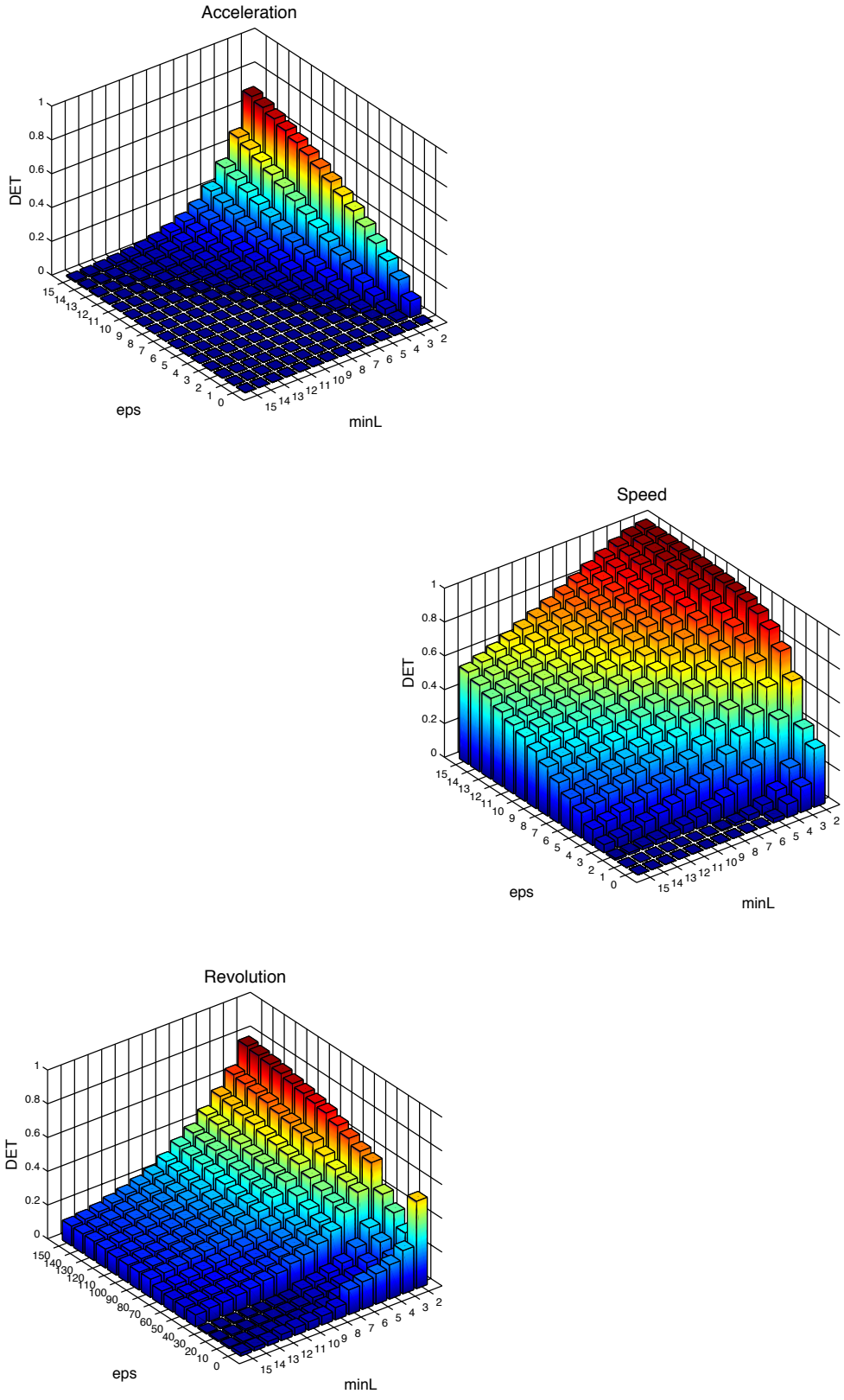


Figure 5.5: Determinism (DET) value for changing similarity threshold ϵ and minimum diagonal line length l_{min} for accelerator, speed and revolution signal; based on the cross recurrence plots (CRPs) of 10 randomly selected pairs of tours from our DRIVE dataset. Note that the DET was averaged.

line structures, which corresponds to $DET \geq 0.2$. Based on this criterion we choose $l_{min} = 5$ and $\epsilon = 14/2/40$ for the accelerator, speed, and revolution signal respectively. Note that the individual signals were not normalized, wherefore the ϵ-threshold represents the accelerator pedal angle, kilometers per hour, and rotations per minute.

To identify prototypical time series using RRR and DTW distance respectively, we applied k-medoids clustering with random initialization. For evaluation purpose we computed index I and E for a varying number of k prototypes. The results of index I were normalized in a way that the highest value, which indicates the optimal number of clusters, equals one. Since index E is a sum of RRR values (see Equation 5.9) and $RRR = 1 - DET$, the lower E, the higher the average DET value, and the more recurring (driving behavior) patterns are comprised of the prototypes identified by the respective distance measure.

Figure 5.6 shows the empirical results for clustering univariate and multivariate time series of the VW DRIVE dataset using RRR and DTW distance respectively. Since the VW DRIVE dataset consists of 'only' 124 test drives recorded by one and the same vehicle, the optimal number of clusters for both RRR and DTW distance is rather small. However, the proposed RRR distance is able to find cluster configurations with lower index E values or rather prototypes with higher amount of recurring patterns than the DTW distance. In case of univariate time series (a), in particular speed measurements, RRR and DTW achieved an index E value of around 0.52 and 0.65 for the optimal number of clusters, which corresponds to a determinism value of 0.48 and 0.35 respectively. In the multivariate case (b), RRR and DTW reached an index E value of around 0.74 and 0.84 for the optimal number of clusters, which corresponds to determinism value of 0.26 and 0.16 respectively. As might be expected, the results for the univariate time series are better than for the multivariate case, because the search space expands and the probability of recurring patterns decreases with an increasing number of dimensions or measurements respectively. In both cases, however, our RRR distance performs about 10% better than the compared DTW distance, meaning that the identified prototypes contain 10% more recurring (driving behavior) patterns.

Figure 5.7 shows the prototype or rather medoid time series of the biggest cluster found by the k-medoids algorithm (for $k = 2$) in combination with our RRR distance measure. In the univariate case (a) the medoid contains a high amount of patterns that recur in the time series objects of the corresponding cluster, making it an excellent prototype. As expected, in the multivariate case (b) the medoid time series contains less and shorter intervals of recurring patterns.

	(a) Speed		(a) Speed		(b) Acceleration, Speed, and Revolution		(b) Acceleration, Speed, and Revolution		
k	I_RRR	E_RRR	I_DTW	E_DTW	I_RRR	E_RRR	I_DTW	E_DTW	k
1	-	0.5441	-	0.7041	-	0.7959	-	0.8737	1
2	**1.0000**	**0.5168**	0.1162	0.6794	**1.0000**	**0.7393**	0.7775	0.8622	2
3	0.8778	0.5034	0.6904	0.6602	0.7820	0.7203	0.9088	0.8405	3
4	0.6431	0.4952	0.7548	0.6474	0.5558	0.7064	0.8585	0.8413	4
5	0.4647	0.4924	0.4438	0.6474	0.3883	0.6992	**1.0000**	**0.8407**	5
6	0.3479	0.4909	**1.0000**	**0.6480**	0.2821	0.6934	0.9746	0.8420	6
7	0.2687	0.4888	0.2993	0.6479	0.2141	0.6910	0.2529	0.8452	7
8	0.2151	0.4892	0.1894	0.6493	0.1679	0.6897	0.3100	0.8482	8
9	0.1751	0.4866	0.1189	0.6507	0.1362	0.6855	0.3955	0.8478	9
10	0.1469	0.4862	0.1271	0.6524	0.1131	0.6837	0.2119	0.8534	10
11	0.1254	0.4838	0.3730	0.6530	0.0960	0.6818	0.2624	0.8545	11
12	0.1078	0.4823	0.1184	0.6544	0.0825	0.6784	0.4089	0.8528	12
13	0.0947	0.4817	0.1616	0.6518	0.0717	0.6781	0.2517	0.8576	13
14	0.0838	0.4804	0.2449	0.6531	0.0635	0.6755	0.2453	0.8574	14
15	0.0745	0.4805	0.2988	0.6598	0.0565	0.6746	0.2941	0.8603	15
16	0.0672	0.4803	0.2365	0.6570	0.0508	0.6718	0.2753	0.8588	16
17	0.0609	0.4780	0.1862	0.6507	0.0462	0.6674	0.1106	0.8535	17
18	0.0557	0.4774	0.1761	0.6569	0.0422	0.6687	0.2091	0.8622	18
19	0.0514	0.4751	0.3307	0.6603	0.0387	0.6687	0.1336	0.8596	19
20	0.0473	0.4756	0.0899	0.6579	0.0358	0.6667	0.1036	0.8563	20

Figure 5.6: Evaluation of RRR and DTW distance for clustering a) univariate and b) multivariate time series of our DRIVE dataset. We compare the index E for the number of clusters k where the (normalized) index I reaches its maximum. The results are based on 1000 runs of k-medoids clustering with random initialization.

5.9 Conclusion and Future Work

This work is a first attempt to solve time series clustering with nonlinear data analysis and modeling techniques commonly used by theoretical physicists. We adopted recurrence plots (RPs) and recurrence quantification analysis (RQA) to measure the (dis)similarity of multivariate time series that contain segments of similar trajectories at arbitrary positions and in different order.

Strictly speaking, we introduced the concept of joint cross recurrence plots (JCRPs), a multivariate extension of traditional RPs, to visualize and investigate recurring patterns in pairwise compared time series. Furthermore, we defined a recurrence plot-based (RRR) distance measure to cluster time series with order invariance.

The proposed RRR distance was evaluated on both synthetic and real-life time series, and compared with the DTW distance. Our evaluation on syn-

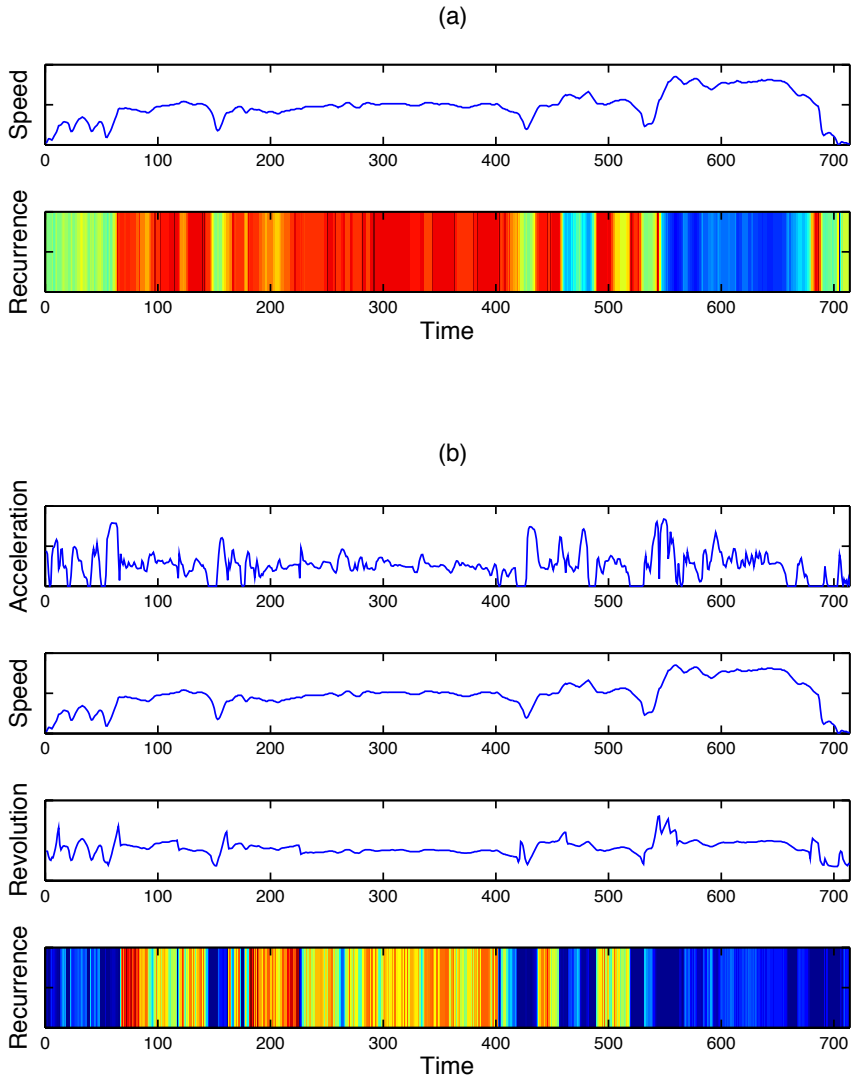


Figure 5.7: Medoid time series of biggest cluster (with $k = 2$) found by our RRR distance measure for a) univariate and b) multivariate case. The intervals highlighted in red color indicate patterns that frequently recur in the time series objects of the corresponding cluster, whereas intervals in blue indicate low recurrence.

thetic data demonstrates that the RRR distance is able to establish cluster centers that preserve the characteristics of the time series. The results on real-life vehicular data show that, in terms of our cost function, RRR performs about 10% better than DTW, meaning that the determined prototypes contain 10% more recurring driving behavior patterns.

Worthwhile future work includes 1) the investigation of RQA measures which quantify recurring patterns with uniform scaling, 2) the application of speed-up techniques for RP computations, and 3) the formalization/analysis of a RP-based distance metric.

Chapter 6

Model-based Distance

for Superimposed Time Series

Superimposed time series pose a problem for traditional shape-based distance measures such as dynamic time warping. For instance, the superposition of several known time series patterns usually creates unfamiliar shapes, which are hard to classify. Model-based approaches that rely on statistical inference, such as the Hidden Markov Model (HMM), have been shown to perform well on aggregated or mixed signals. In general, a model-based time series distance can either be understood in terms of residuals (that are obtained by comparing individual time series against a trained model) or in terms of parameters (estimated by training a model for each individual time series). This works demonstrates a practical application of model-based distances for superimposed time series, namely heating control.

Heating control is of particular importance, since heating accounts for the biggest amount of total residential energy consumption. Smart heating strategies allow to reduce such energy consumption by automatically turning off the heating when the occupants are sleeping or away from home. The present context or occupancy state of a household can be deduced from the appliances that are currently in use.

In this chapter we investigate energy disaggregation techniques to infer appliance states from an aggregated energy signal measured by a smart meter. Since most household devices have predictable energy consumption, we propose to use the changes in aggregated energy consumption as features for the appliance/occupancy state classification task. We evaluate our approach on real-life energy consumption data from several households, compare the classification accuracy of various machine learning techniques, and explain how to use the inferred appliance states to optimize heating schedules.

6.1 Introduction

The main goal of our study is to provide a framework for heating control and scheduling which considers the occupancy states of residential homes. Since most solutions for occupancy state identification involve complex sensor infrastructure and costly hardware which cause high usage barrier [15, 99, 123, 148], we aim to use given information from available electricity smart meters. We propose to employ energy disaggregation to infer appliance usage, which is, as we will show, beneficial to occupancy state identification. In the following we briefly introduce the value of appliance usage information, before we explain how we use this information for the purpose of heating control.

In the context of domestic environments, consumers vastly underestimate the energy used for heating and overestimate the energy used for appliances that replace manual labor tasks [48]. Numerous studies have identified that consumers get a better understanding of their energy use by clear, concise, and direct feedback about appliance-specific consumption information [120, 164, 222].

In regard to power grid operators and power suppliers, knowledge about the energy consumption on appliance level is critical to the development of power system planning, load forecasting, billing procedures, and pricing models [48, 164]. In addition, the identification of electric appliances in domestic environments is important, because the increasing number of renewable energy sources in the power grid requires electric utilities to be able to quickly react to changes in supply and demand [162].

The growing need for accurate and specific information about domestic energy consumption on device level has led to numerous studies on appliance load monitoring [6, 48, 100, 220, 222]. Existing solutions for appliance load monitoring can be classified into two primary techniques [48, 209]: distributed direct sensing and single-point sensing.

Distributed direct sensing typically requires a current sensor to be installed in-line with every device and is therefore often referred to as intrusive load monitoring. Although intrusive load monitoring easily achieves a consumption breakdown, deploying a large number of sensors in the residential environment quickly leads to high cost and discouraging high usage barrier [209].

Single-point sensor systems are easier to deploy and are typically subsumed under the concept of non-intrusive load monitoring (NILM) [209]. Energy disaggregation is the task of using an aggregated energy signal, such as that coming from a single-point sensor or rather whole-home power monitor, to make inferences about the different loads of individual appliances [100]. However, single-point sensor systems require knowledge about the

household devices and their electrical characteristics [209]. The challenges in energy disaggregation are mainly due to appliances with similar energy consumption, appliances with multiple settings, parallel appliance activity, and environmental noise [164]. Recent studies [97, 100, 111, 114, 120, 192] have shown that machine learning techniques represent a suitable solution to recognize appliances in such dynamic and unpredictable environments.

In this work we consider energy disaggregation techniques to derive occupancy states from appliance usage data in order to use this information in smart heating control strategies [99]. Heating control is of particular importance, since heating accounts for the biggest amount of total residential energy consumption and recent studies have shown that up to 30% of the total energy can be saved by turning the heating off when the occupants are asleep or away [123]. Existing work on the inference of occupancy states in residential environments includes statistical classification of aggregated energy data [99], hot water usage [15] as well as human motion and activity [148]. Our own approach to infer occupancy states differs in that we consider appliance usage, which gives more detailed information about the present context in a household and the devices that suggest user activity. Furthermore, our proposed framework does not require any additional infrastructure, and, therefore, is more likely to be accepted by residents.

For the evaluation of our approach we consider the REDD dataset [100], which consists of whole-home and device specific electricity consumption for a number of real houses over the period of several month. In our experiments we compare the performance of different models for the appliance/occupancy state classification task. We use cross-validation (training on all houses and leave-one-out for testing) to evaluate how well the different models generalize. Our results suggest that the Naive Bayes classifier is suitable for the prediction of occupancy/appliance states and fits the problem of real-time heating control.

The rest of the paper is structured as follows. In Section 6.2 we give some background on recent advances in energy disaggregation. Section 6.3 introduces the formal notation of our appliance state classification task. Our proposed framework for heating control and scheduling by means of energy disaggregation techniques is described in Section 6.4. The experimental design and results on our approach are presented in Section 6.5. Eventually, we conclude our study and give an outlook on future work in Section 6.6.

6.2 Background

Energy disaggregation, also referred as non-intrusive load monitoring, is the task of using an aggregated energy signal, such as that coming from a whole-home power monitor, to make inferences about the different individual loads of the system [100]. This approach is seen as an intermediate between existing electricity meters (which merely record whole-home power usage) and fully energy-aware home appliance networks, where each individual device reports its own consumption [162].

For a thorough evaluation of various energy disaggregation mechanisms under real-world conditions, a comprehensive collection of power consumption data is needed [162]. Most approaches to energy disaggregation have been supervised, in that the model is trained on individual device power signals [222]. The vast majority of supervised disaggregation approaches have evaluated the trained models on the same devices but in new conditions [6].

Research on energy disaggregation has been encouraged by publicly available datasets such as REDD [100], which contains information about the power consumption of several different homes on device level, and, therefore, allows cross-validation for individual appliances. Experiments on the REDD dataset have shown that the Factorial Hidden Markov Model (FHMM) is able to disaggregate the power data reasonably well [100]. In that case, the disaggregation task is framed as an inference problem and the performance of energy disaggregation is evaluated considering the percentage of energy correctly classified.

Although FHMMs have shown to be a powerful tool [54] for learning probabilistic models of multivariate time series, the combinatorial nature of distributed state representation makes an exact algorithm for inferring the posterior probabilities of the hidden state variables intractable. Approximate inference can be carried out using Gibbs sampling or variational methods [54]. Recent work [97] on energy disaggregation presents different FHMM variants which incorporate additional features and better fit the probability distribution of the state occupancy durations of the appliances.

Another work [164] proposes Artificial Neural Networks (ANNs) for appliance recognition, because they (i) do not require prior understanding of appliance behavior, (ii) are capable of handling multiple states, and (iii) are able to learn while running. The results show that after training the ANN with generated appliance signatures, the proposed system is able to recognize the previously learned appliances with relatively high accuracy, even in demanding scenarios. To tune the ANN, the authors suggest to use the generated signatures to create a training dataset with all possible combinations of appliance activity. Comparing the disaggregation performance for different

ANN algorithms, additional work [111] suggests to employ back-propagation rather than the radial-base-function.

In another study [209] the authors propose a disaggregation algorithm that consists of several consecutive steps including normalization, edge detection via thresholding and smoothing techniques, extraction of power level and delta level consumption, matching of known appliances from a signature database with extracted delta vectors, and labeling of recognized devices. The proposed system does not require setup or training, because the user is able to label appliance signatures via her smart phone. In that case, the appliance signatures are based on apparent, reactive, real, and distortion power measured by the smart meter.

The classification of household items based on their electricity usage profile over a fixed time interval is discussed in yet another study [120]. The authors consider the time series classification problem of identifying device types through daily or weekly demand profiles. The proposed approach concentrates on bespoken features such as mean, variance, kurtosis, skewness, slope, and run measures. The experiments show that classification using the bespoken features performs better than classification using the raw data. However, the nature of similarity captured strongly depends on the features extracted.

In a similar work [162] the authors present an appliance identification approach based on characteristic features of traces collected during the 24 hours of a day. The extracted features include temporal appliance behavior, power consumption levels, shape of the consumption, active phase statistics, and noise level characteristics. Each resulting feature vector is annotated by the actual device class and used to train the underlying model of the selected classifier. Among various tested classifiers the Random Committee algorithm perform best in categorizing new and yet unseen feature vectors into one of the previously trained device types. Additional work [111] demonstrates that the solution from any single-feature, single-algorithm disaggregation approach could be combined under a committee decision mechanism to render the best solution.

Yet another work [192] presents a non-intrusive appliance load monitoring technique based on integer programming. Since the overall load current is expressed as a superposition of each current of the operating appliance, the monitoring problem can be formulated as an integer quadratic programming problem by expressing the operating conditions as integer variables. Besides that the proposed method does not require relearning when a new appliance is installed in the house, it is furthermore able to distinguish between different device modes and some-type appliances that operate simultaneously.

To monitor the states of multiple appliances via electricity consump-

tion measurements, another work [114] introduces the Bayes filter approach, which computes the posterior distribution over the current state given all observations to date. Since the state transition of an appliance is a continuous process, the authors employ a sliding window to take the temporal factor into consideration and extract the past records of data to be features. The estimated states are represented as binary strings, where each bit denotes the on/off state of one individual appliance. According to the results, the Bayes filter outperforms the KNN, Naive Bayes, and SVM classifier.

Leveraging recent advances in device and appliance power supplies, another series of studies [48, 60] extends the energy disaggregation approach by using high-frequency sampling of voltage noise, which provides an additional feature vector that can be used to distinguish more accurately between energy usage signatures. Appliances conduct a variety of noise voltage back onto the home's power wiring, yielding measurable noise signatures that are easily detectable using appropriate hardware. An important advantage of voltage noise signatures is that any electrical outlet inside the home can be used as a single installation point.

6.3 Notation

Since different devices tend to draw different amounts of power, which are consistent over time, total power is a reasonable feature to use for classification [48]. Most devices have predictable current consumption and can be categorized according to the magnitude of real/reactive power. Assuming a household with d devices, the power consumption of an individual appliance $i = 1, \ldots, d$ over a period of m time points can be expressed as: $Y^{(i)} = \{y_1^{(i)}, \ldots, y_j^{(i)}, \ldots, y_m^{(i)}\}$. Usually we only observe the sum of all power outputs at each time: $\bar{y_j} = \sum_{i=1}^{d} y_j^{(i)}$, with $j = 1, \ldots, m$.

Given the aggregated power signal most research on energy disaggregation [6, 220, 222] aims at inferring the individual device consumption. Since we aim to infer the context or rather occupancy states in residential environments in order to optimize heating control, we are mainly interested in the ON/OFF states of individual appliances $S^{(i)} = \{s_1^{(i)}, \ldots, s_j^{(i)}, \ldots, s_m^{(i)}\}$, where $s_j^{(i)} = 1$ if device i is turned 'on' at time point j, and $s_j^{(i)} = 0$ otherwise. The appliance state identification task can be framed as an inference problem. Given an aggregated power signal $\bar{Y} = \bar{y}_1, \ldots, \bar{y}_m$, we intend to compute the posterior probability $p(s_j^{(i)}|\bar{y}_j)$ of individual appliance states $s_j^{(i)}$ for each device $i = 1, ..., d$ and each time point $j = 1, \ldots, m$.

Due to the fact that the aggregated power signal is super-imposed and unnormalized, and, therefore, unsuitable for the appliance state identifica-

tion, we consider the changes in power consumption as features, which can be derived by the first-order difference of the power signal: $\Delta y_j^{(i)} = y_j^{(i)} - y_{j-1}^{(i)}$ for $j = 2, \ldots, m$. Thus the appliance state identification task could also be formulated as a classification problem, where a certain change in power consumption categorizes a device into either 'ON' or ' OFF' state.

6.4 Framework and Algorithms

Figure 6.1 shows a flowchart of our proposed framework for heating control and scheduling by means of energy disaggregation. The input for our heating control framework is an aggregated energy signal, such as that coming from a smart meter in a residential home. In the first step (i) we extract features from the energy signal, i.e. changes in consumption, which can be used to categorize the individual electrical devices. Subsequently (ii) we use the extracted features as input for the appliance state classification. For the sake of simplicity, Figure 6.1 assumes that the individual appliance models were trained on other households prior to the classification task. Given the classified ON/OFF states for each appliance, we can eventually (iii) infer the occupancy state of the respective household and recommend optimized heating schedules.

In the following subsections we describe the (i) feature extraction, (ii) appliance state classification and (iii) inference of occupancy in more detail.

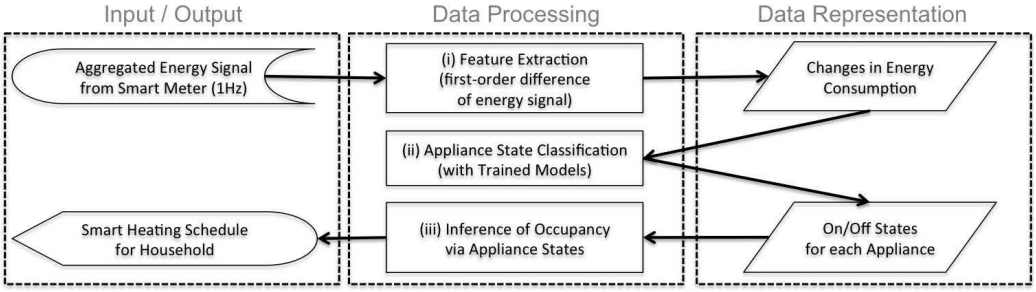


Figure 6.1: Framework for Heating Control and Scheduling by means of Energy Disaggregation Techniques.

6.4.1 Feature Extraction

Given the overall energy consumption of a household and the energy consumption of the individual appliances in this household, we aim to build a

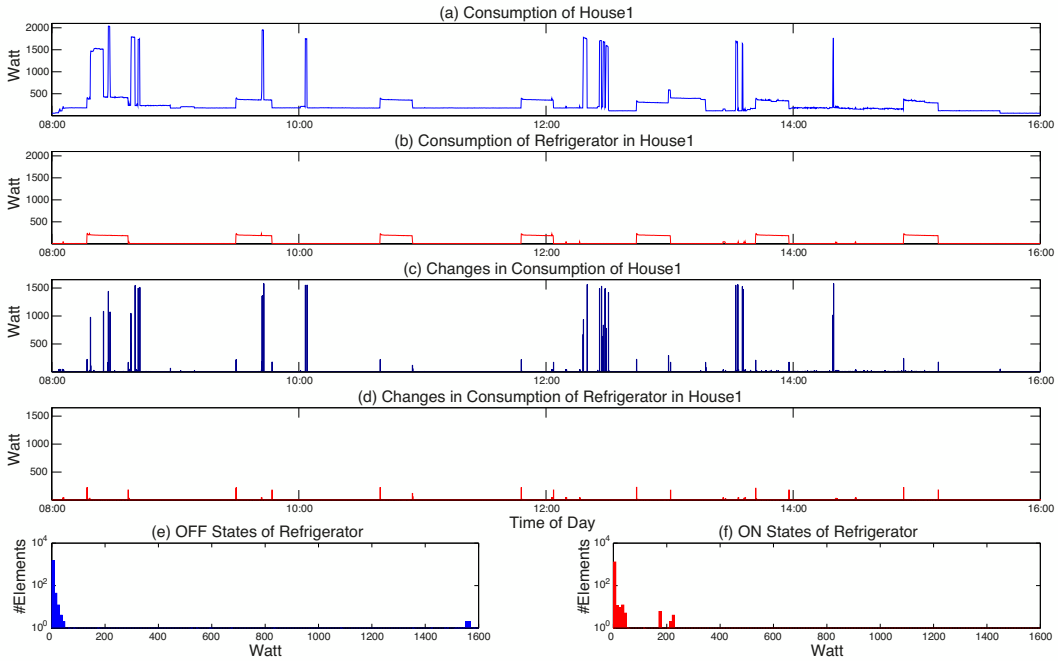


Figure 6.2: Energy consumption of (a) House1 and (b) its Refrigerator over an interval of 8 hours (on May 21st, 2011). Plot (c) and (d) show the changes in power consumption for House1 and its Refrigerator. The distribution of power changes that classify the Refrigerator's ON/OFF states are illustrated in Plot (e) and (f).

model for each appliance in order to estimate its ON/OFF states in a previously unknown environment or household. Since an appliance can be either turned ON or OFF, the device state identification can be formalized as a two class problem. For the training of an individual appliance model we consider the changes in power consumption that classify the respective device states. In our approach the input for the classification model are two distributions of power changes, which represent the features that characterize one or the other class.

Figure 6.2 illustrates the feature extraction process on the basis of real-life measurements from the REDD dataset, in particular the energy consumption of (a) House1 and (b) its Refrigerator for a sample time frame of 8 hours. We can see that (a) the overall energy consumption is the sum of (b) the Refrigerator's energy consumption and the energy consumption of other appliances. Given this information, we can derive the changes in energy consumption by the first-order difference of the power signals. This step is often referred to as edge detection, since the stable periods in the signal are filtered

out. The edges or changes in power consumption of the overall energy signal and the Refrigerator signal are shown in Figure 6.2 (c) and (d) respectively. Knowing which edges specify (d) the activity of the Refrigerator, we can easily separate the changes in energy consumption that categorize other devices by considering all the edges in (c) the overall energy signal which do not belong to the Refrigerator. The distribution of the edges that classify the Refrigerator's ON/OFF states are illustrated in Figure 6.2 (e) and (f). These distributions can serve as training input for most probabilistic models.

6.4.2 Appliance State Classification

In this study we aim at evaluating the appliance state classification task by means of various machine learning techniques, including Naive Bayes (NB) classifier, Factorial Hidden Markov Model (FHMM), Classification Tree (CT), and One-Nearest-Neighbor (1NN) classifier.

We selected these models based on their complementary characteristics and degree of popularity regarding the energy disaggregation task. Table 6.1 shows typical characteristics of the considered machine learning techniques [136], although the characteristics strongly depend on the underlying algorithm and the problem. Therefore, Table 6.1 should be considered as a guide for an initial choice of models.

	NB	**FHMM**	**CT**	**1NN**
Fitting Speed	Fast	Fast	Fast	Fast
Prediction Speed	Fast	Fast	Fast	Medium
Memory Usage	Low	Low	Low	High
Easy to Interpret	Yes	No	Yes	No

Table 6.1: Characteristics of Algorithms.

The NB classifier is a simple probabilistic model based on applying Bayes' theorem with strong independence assumptions, which has been applied for appliance and occupancy recognition in various studies [99, 114, 120, 162]. Speed and memory usage of the NB classifier are good for simple distributions, but can be poor for large datasets [136].

The FHMM is a statistical model in which the system under study is assumed to be a Markov process with unobserved or hidden states. FHMMs have been successfully applied to the energy disaggregation problem [97, 100, 222], however their complexity increases with the number of states and the length of the Markov chain [54, 97].

CTs map observations about an item to conclusions about the item's target value, meaning the predicted outcome is the class to which the data

belongs. Decision tree learning has been proven to be applicable to appliance identification on metering data in a couple of recent studies [6, 162].

The 1NN classifier is often regarded as the simplest straw man or baseline approach [87], and has been considered for the energy disaggregation task in several studies [114, 120, 222]. 1NN usually has good performance in low dimensions, but can have poor predictions in high dimensions. For linear search, 1NN does not perform any fitting [136].

6.4.3 Inference of Occupancy

We assume that there exists a direct relationship between appliance usage and occupancy states in residential homes. For instance, if the lighting is turned ON we usually know that the residents are at home, unless someone forgot to turn OFF the lighting. Hence, lighting may be a straightforward indicator for occupancy states, enabling us to verify manually adjusted heating schemes and recommend optimized heating schedules.

However, heating control is much more complex, because the usage of certain appliance actually requires to decrease the temperature. For example, when residents turn ON the oven or stove the temperature in the kitchen rises automatically, and we can reduce heating to save energy, instead of just opening the window. In case the heating control system would have knowledge about the installation points of all devices, one could even use the appliance states to control the temperature in individual rooms.

The knowledge of individual appliance states furthermore allows us to infer devices that are unrelated to occupancy. For instance, the refrigerator automatically switches between ON and OFF state every few minutes, no matter if the residents are at home or not. The same is true for devices in standby mode or appliances such as the smoke alarm or electronic panels which are constantly drawing power. Therefore, by just looking at the overall energy consumption of a household it is impossible to distinguish between occupancy states.

The accuracy of the appliance state classification and the implications for heating control will be scrutinized in the following section.

6.5 Empirical Evaluation

The goal of our evaluation is twofold: (i) we investigate which of the considered machine learning models is most accurate for the the appliance state classification task; and (ii) we assess the use of the identified appliance or rather occupancy states for heating control.

6.5.1 Energy Data

We consider the REDD dataset [100], which comprises electricity consumption measurements from six household at the granularity level of individual devices, and represents to date one of the largest and richest publicly available collections of power consumption data [10]. There are approximately 20 consecutive days of measurements available for each house, providing data from the two main phases and each individual circuit at 1Hz frequency rate. Measured appliances include main consumers such as Air Conditioning, Dishwasher, Disposal, Electrical Heating, Microwave, Oven, Refrigerator, Stove, Washer/Dryer as well as other miscellaneous electronics and outlets (see Table 6.2).

6.5.2 Experimental Design

In our empirical evaluation we compare the classification accuracy of the introduced machine learning models (see Table 6.1) on the REDD dataset. Strictly speaking, we assess the appliance state classification accuracy for all considered models on a granularity level of individual devices. The training of the respective models is done on appliance-specific consumption measurements of one particular device for all households but one. The aggregated electricity consumption signal of the left-out household is then used for testing the performance of the trained models for each individual device. This evaluation principle is also commonly known as cross-validation with leave-one-out.

6.5.3 Classification Accuracy

Table 6.2 illustrates the classification accuracy per (a) household and (b) appliance for all examined models, including Naive Bayes (NB), Factorial Hidden Markov Model (FHMM), Classification Trees (CT), and One-Nearest-Neighbor (1NN) classifier. The classification results present the performance of the trained models in an unknown environment or rather before unseen household.

The results in Table 6.2 (a) show the classification accuracy of device states per household averaged over all appliances. For instance, the NB model achieved an accuracy of 0.8429 for House1, meaning that the model was trained on House2-6 and tested on the previously unknown House1, where 84.29% of all device states were classified correctly. However, as illustrated in Table 6.2 (a) the classification accuracy of each model varies with the household, which is due to the fact that the examined households use

(a) Classification Accuracy of Device States per Household averaged over all Appliances

	NB	**FHMM**	**CT**	**1NN**
House 1	0.8429	0.8414	0.8319	0.7615
House 2	0.9310	0.9300	0.9224	0.8062
House 3	0.9275	0.9200	0.8908	0.7213
House 4	0.8645	0.8616	0.8746	0.7038
House 5	0.9864	0.9854	0.9839	0.7638
House 6	0.8131	0.7873	0.7752	0.6050
MEAN	**0.8942**	**0.8876**	**0.8798**	**0.7269**

(b) Classification Accuracy of Device States per Appliance averaged over all Households

	NB	**FHMM**	**CT**	**1NN**
Air Conditioning	0.9315	0.9248	0.9300	0.9138
Bathroom GFI	0.9328	0.9275	0.9324	0.9134
Dishwasher	0.9541	0.9493	0.9551	0.9134
Disposal	0.9955	0.9818	0.9970	0.9918
Electrical Heating	0.8863	0.8856	0.8620	0.8895
Electronics	0.8875	0.7970	0.7404	0.0991
Furnace	0.8216	0.8211	0.7294	0.5216
Kitchen Outlets	0.7902	0.7915	0.7070	0.1775
Lighting	0.7751	0.7737	0.8006	0.7611
Microwave	0.9516	0.9473	0.9526	0.9279
Miscellaneous	0.9242	0.9295	0.9296	0.7237
Outdoor Outlets	0.9982	0.9995	0.9997	0.9996
Oven	0.9754	0.9804	0.9815	0.9811
Refrigerator	0.7834	0.7872	0.7952	0.7898
Smoke Alarm	0.9729	0.9629	0.9738	0.6234
Stove	0.9346	0.9288	0.9363	0.8330
Subpanel	0.9808	0.9807	0.9815	0.9811
Unknown Outlets	0.9578	0.9558	0.9555	0.3432
Washer Dryer	0.9287	0.9256	0.9297	0.8763
MEAN	**0.9148**	**0.9079**	**0.8994**	**0.7505**

Table 6.2: Cross-validation of trained models.

appliances of different manufacturers with dissimilar energy profiles.

Table 6.2 (b) presents the classification accuracy of device states per appliance averaged over all households. For example, the results show that the NB model is able to classify the device states of the Air-Conditioning with an average accuracy of 93.15%, taking the mean of House1-6. In general, all models achieved a relatively high classification accuracy for appliances with distinctive energy profiles, such as the Dishwasher or Oven, but performed less well on appliances with changes in consumption that can be easily be confused with other devices, like the Refrigerator or Lighting.

By taking the mean over all results for (a) each household and (b) each appliance per model, shown in the bottom row of Table 6.2 (a) and (b) respectively, we can easily see that on average the NB model achieved the highest classification accuracy, closely followed by FHMM and CT. Although the 1NN classifier shows relatively high classification accuracy for several individual appliances, it is unable to correctly classify the device states of others, and, therefore, achieved the lowest average classification performance.

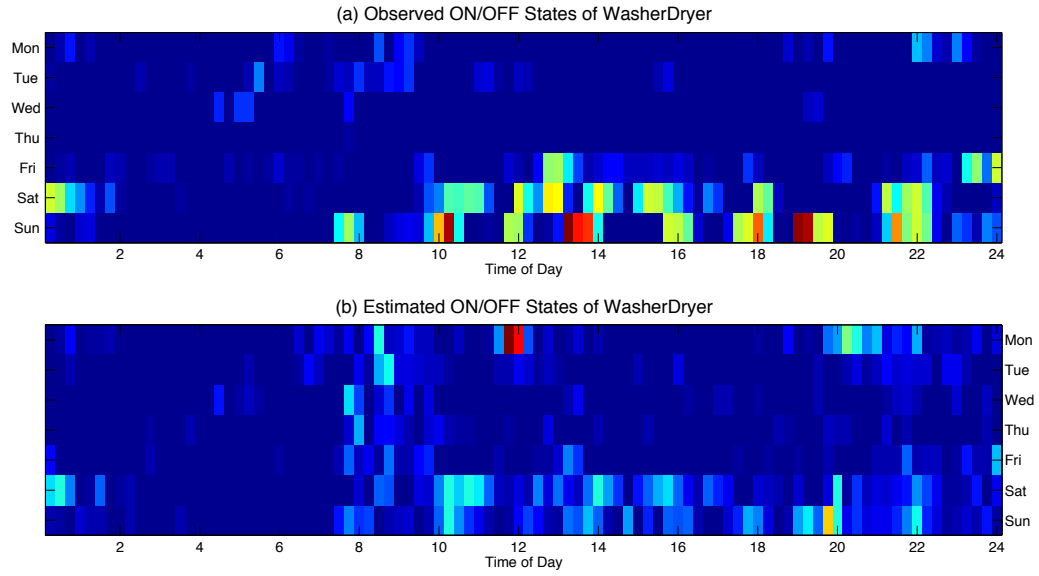


Figure 6.3: Observed and estimated ON/OFF states for the Washer/Dryer in House1 averaged over the period of 4 weeks, illustrating the actual and predicted device activities in the time from April 25th to May 15th 2011, where every quarter of an hour aggregates the activities that occurred during the same weekday and time of day. The transition from blue, to yellow, to red colored areas illustrates low, moderate, and high device activity.

6.5.4 Heating Control

In this subsection we discuss how the classified ON/OFF device states can be used for heating control and scheduling. Since the Naive Bayes (NB) model achieved the highest average accuracy on classifying device states of appliances in an unknown household (see Table 6.2), we will consider the NB approach in our following exemplification.

Figure 6.3 shows the (a) observed and (b) estimated ON/OFF states for the Washer/Dryer in House1 over a period of four weeks, where every quarter of an hour aggregates the device activities that occurred during the same weekday and time of day. By illustrating the (a) observed activity of the Washer/Dryer, which constitutes our ground truth, we see that this appliance is mostly used on Fridays and weekends. The (b) estimated activity of the Washer/Dryer, inferred from the overall energy consumption of House1 by the trained NB model, shows similar behavior patterns for weekends, but predicts false ON states for Mondays.

By taking a closer look at the confusion matrix of observed and estimated ON/OFF device states for the Washer/Dryer in House1, shown in Table 6.3, we are able to gain a better understanding of the estimated appliance activity. Table 6.3 reveals the percentage of true positives (TP) or true ON states, true negatives (TN) or true OFF states, false positives (FP) or false ON states, and false negatives (FN) or false OFF states. Although the NB model achieves a high classification accuracy [(TP+TN)/(TP+TN+FP+FN)=96.83%], the percentage of falsely classified states [FP+FN=3.17%] in not negligible, explaining the mistaken Washer/Dryer activity estimated for Mondays (see Figure 6.3(b)). The FP and FN estimates imply heating during absence and cooling during occupancy respectively.

	Observed ON	Observed OFF
Estimated ON	True Positive (TP) = 0.59%	False Positive (FP) = 1.14%
Estimated OFF	False Negative (FN) = 2.03%	True Negative (TN) = 96.24%

Table 6.3: Confusion matrix of observed and estimated ON/OFF device states for the Washer/Dryer in House1, where Accuracy=TP+TN=96.83%.

The cause of falsely classified states can also be explained with help of Figure 6.2. By examining the distribution of ON and OFF states of the Refrigerator in House1, shown in Figure 6.2 (e) and (f) respectively, we can see there is a significant overlap of changes in power consumption that are caused by both the Refrigerator and other devices. According to Figure 6.2

(e) and (f), changes in power consumption that range from around 1 to 50 Watt occur at times when the Refrigerator is turned ON as well as when its is turned OFF, leading to an inaccurate appliance model.

In order to decrease the number of FP and FN device states one could orchestrate the trained appliance models or consider additional features that distinguish the appliances more accurate. However, this goes beyond the scope of this study, but could be part of future work.

A more thorough evaluation of the heating schedules would require datasets that comprise information about actual occupancy states in the residential homes and preferences of the residents in regard of temperature settings.

6.6 Conclusion and Future Work

In this work we reviewed recent advances in energy disaggregation and adopted established appliance identification strategies to infer occupancy states for smart heating control and scheduling. Our proposed approach to appliances state identification considers the changes in power consumption as characteristic to classify the individual devices. In our evaluation we have shown that the Naive Bayes classifier is able to achieve relatively high accuracy on the appliance state identification task, even in unknown environments or households. Furthermore, we explained how to use the information about identified appliances to infer occupancy states in residential homes. We exemplified the idea of occupancy-based heating schedules and discussed the problem of falsely identified appliance states.

The main advantage of our proposed framework for heating control and scheduling is its simplicity in that we refrain from implementing new infrastructure in residential homes, but use given information from available electricity smart meters. This approach will eventually lead to higher acceptance rates among residents and provides alternative avenues for novel heating control strategies.

Since our appliance state identification strategy can replace sensing infrastructure that is used to identify occupancy states in residential homes, it would also be interesting to compare the energy savings provided by our approach with the performance of existing frameworks, such as the smart thermostat [123]. However, this would require datasets that comprise information about actual occupancy states in the residential homes and preferences of the residents in regard of temperature settings.

Our proposed approach to appliance state identification can furthermore be beneficial for other applications. Recent studies [10] have shown that the availability of smart meter data alone is often not sufficient to achieve high

load disaggregation accuracies. Future work could combine the knowledge of total energy consumption with additional information about sequences of events, such as ON/OFF states for each individual appliance, to improve the accuracy of certain disaggregation algorithms [10] that use such events along with smart meter data.

Chapter 7

Applied Time Series Distances

In the previous chapters (3-6) we introduced several different time series distance measures, discussed their theoretical properties, and evaluated their actual performance on datasets from various domains. Certainly all proposed time series distance measures were designed with a particular purpose in mind. The development of the introduced distance measures was mainly funded by industrial partners (see acknowledgement in front matter), which have a general interest in using the distance measure to solve real-world time series mining tasks. In general, the industry partners require an easy-to-use software prototype with an intuitive graphical user interface, which enables them to load data, select parameters, and visualize results in a straightforward manner.

In this chapter we present several software prototypes, which were designed on behalf of our industry partners for the purpose of mining time series datasets. In the following we introduce three time series analysis tools, including SOE, BestTime, and MatArcs.

SOE analyzes aggregated energy consumption signals, such as coming from smart meters, in order to generate optimized heating schedules. The mathematical foundation and applied algorithm of SOE are discussed in Chapter 5. The corresponding software prototype and application context are explained in Chapter 7.1.

BestTime is a tool that allows to identify time series representatives, which best comprehend the recurring temporal patterns in a given dataset. The underlying problem and our proposed solution are discussed in Chapter 4. The graphical user interface and functionality of BestTime are presented in Chapter 7.2.

MatArcs visualizes recurring patterns in time series and is, therefore, somehow related to the BestTime tool. The software architecture and applied mathematical concepts of MatArcs are explained in Chapter 7.3.

7.1 SOE
Heating Control via Energy Disaggregation

A large part of the residential energy consumption attributes to heating. We propose to reduce this consumption by automatically generating optimized heating schedules based on actual presence and preferences of the residents using aggregated energy signals. To demonstrate this, this work introduces SOE - a single-agent smart heating control system. SOE integrates energy disaggregation algorithms using a Naive Bayes classifier to infer appliances states that indicate presence.

7.1.1 Introduction

Smart heating strategies enable us to reduce residential energy consumption by a significant amount [123]. Heating control systems can be optimized according to various parameters, including outside temperature, current energy price, and actual presence of the residents. This work demonstrates how to use an aggregated energy signal, measured by a smart meter, to identify the habits of the residents in terms of presence. We propose to employ energy disaggregation techniques to make inference about the use of certain household appliances, which indicate physical presence of the occupants. This approach is also referred as non-intrusive load monitoring [100, 222] and has the benefit that, unlike other systems [123, 148], it does not require additional sensor infrastructure.

In a previous study [180], we evaluated various machine learning algorithms on the appliance state identification task using the REDD dataset [100], which provides both device-specific and aggregated energy consumption for a number of houses over a period of several weeks. Our empirical evaluation has shown that the Naive Bayes (NB) classifier was able to achieve a relative high accuracy on estimating individual appliance states [180].

This demonstration presents SOE, a single-agent heating control system, that proposes optimized heating schedules that aim to reduce the residential energy consumption. SOE computes the optimized heating schedules based on manual adjustments of the residents and automatically determined occupancy states. In addition, SOE enables the residents to monitor and control their heating from remote using mobile devices.

7.1.2 Main Purpose

Our approach is based on the assumption that the present occupancy state of a household can be inferred from the appliances that are currently in use.

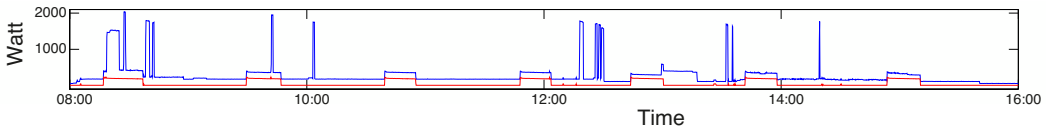


Figure 7.1: Aggregated energy consumption of REDD House1 (blue line) and its fridge (red line) [100]. Although the fridge is not suited to determine physical presence in a household, this plot illustrates the superposition of the individual appliance signals and the changes in consumption that we use as features for the devices state identification.

Since an appliance can be either turned ON or OFF, we formalized the appliance state identification as a two class classification problem. The aggregated energy signal derived from smart meters is usually superimposed and unnormalized and, therefore, unsuitable for the appliance state identification task. However, the signal can be used to derive the appliance state by considering changes in power consumption as features for the appliance state identification. The changes in power consumption can be derived by the first-order difference of the aggregated energy signal.

Given these features we aim at building an appliance model that considers the distribution of the ON and OFF appliances states. Our approach comprises three steps, including (1) feature extraction from the aggregated energy signal, (2) identification of individual appliance states based on the extracted features, and (3) inference of household occupancy from the identified appliance use [180].

Most appliances have predictable energy consumption, which is consistent over time, and can be categorized according to the magnitude of real/reactive power [222]. Figure 7.1 illustrates the aggregated energy consumption of an individual household and its refrigerator.

Taking this observation into account, we used the REDD dataset to compare the classification accuracy of four different machine learning models, Naive Bayes (NB) classifier, Factorial Hidden Markov Model (FHMM), Classification Tree (CT), and One-Nearest-Neighbor (1NN) classifier. The considered models were cross-validated over all six houses of the REDD dataset for each of the measured appliances, such as air conditioning, microwave oven, washing machine, and others. Our evaluation [180] shows that averaged over all six household the NB classifier is able to achieve comparatively high accuracy of about 91.5% on the appliance state identification task, even in unknown environments or households. To deduce the presence in a household, the inferred appliance states are subsequently weighted according their significance or information value.

In order to build a practical application we embedded the implementation of our trained appliance model in our SOE agent using the Matlab to Java compiler.[1] The SOE agent [124] is responsible for the heating control in a home and has access to the aggregated energy signal using SML (Smart Message Language)[2] and the MUC (Multi Utility Communication)[2] interface. For further information about the machine learning part of the heating control system the interested reader is referred to *Spiegel* and *Albayrak* [180].

7.1.3 Demonstration

The SOE heating control system aims at integrating the user as an essential part of the heating control process. At this point we want to address questions concerning various aspects of human computer interaction. This includes the usability and acceptance of the developed system with regard to different user groups and/or environments. Users are able to access the system using mobile devices and control the heating process in a fine-grained manner (refer to Figure 7.3).

Figure 7.2 illustrates the overall architecture of a SOE agent [124]. The residents are enabled to adjust the thermostat and create heating schedules for each individual room (refer to Figure 7.3). Given the aggregated power consumption of the household our implemented energy disaggregation component is able to classify individual appliance states. The inferred appliance use is subsequently employed to infer presence and to propose optimized heating schedules to the residents.

In case that the manually created and automatically optimized heating schedules differ from each other, the SOE agent will provide recommendations for possible adaptations. These suggestions are shown as notifications, whereas the user can either accept the recommended adaptations or reject the automatically generated heating schedule to manually conduct changes. This is of utter importance, because the number of falsely classified appliance states is not negligible. False estimates imply heating during absence or cooling during presence, and are, therefore, undesired.

In our future work, we intend to reduce the number of false estimates and in consequence to improve the appliance classification by using acceptance/rejection as reward/punishment signal for reinforcement learning strategies. In order to demonstrate the system outside of our showroom, we use a common notebook to simulate the smart home and an iPad to show the SOE application.

[1] `http://www.mathworks.de/products/javabuilder/`

[2] `http://www.vde.com/en/fnn/extras/sym2/Infomaterial/Pages/default.aspx`

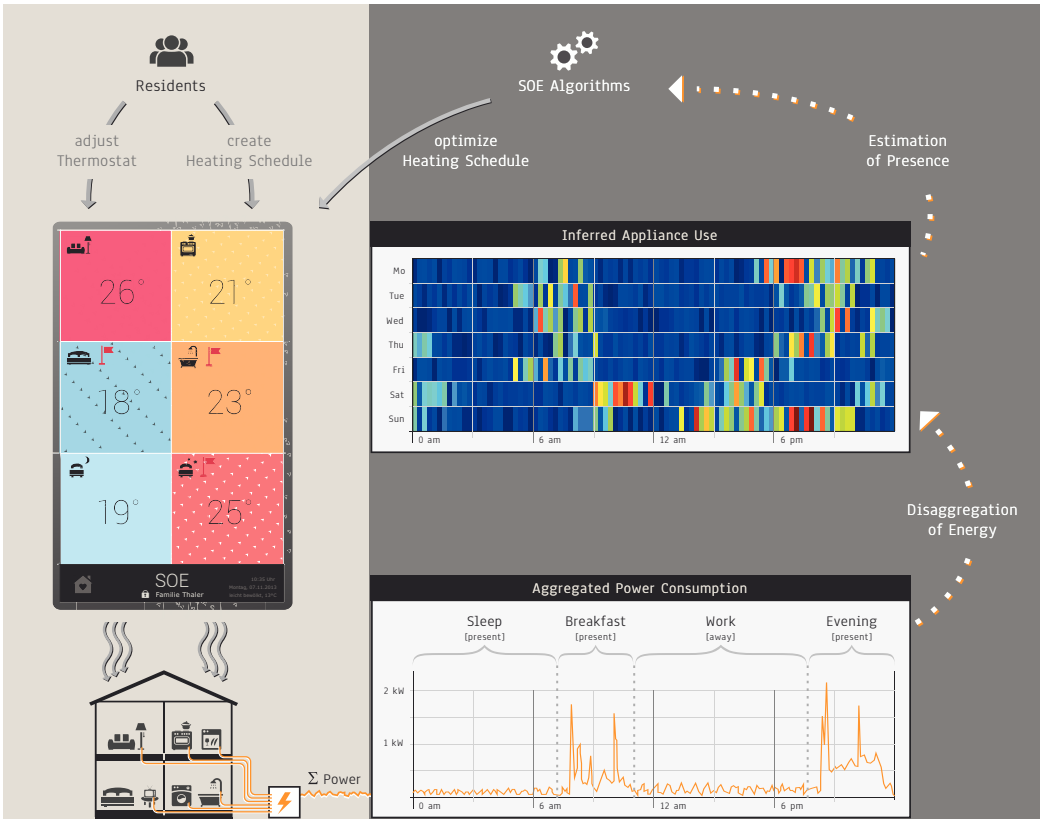


Figure 7.2: Overall architecture of SOE heating control system. The aggregated energy signal is disaggregated using a Naive Bayes classifier to infer appliance usages. From such usage the occupants' presence is estimated and used to optimize the heating schedule. The whole system can be controlled by the residents using tablets and smart-phones.

7.1.4 Conclusion

This work demonstrates the machine learning algorithms employed for the SOE (System for Optimization of in-house Energy demand) project. We explained how to derive the present occupancy state of a household by inferring the use of individual appliances. The main advantage of our approach is that it merely requires an aggregated energy signal measured by a commercial smart meters, without the need for additional sensor infrastructure. Our implementation of the SOE heating control system provides insights into practicality and usability, which are valuable for the intended deployment in real estates.

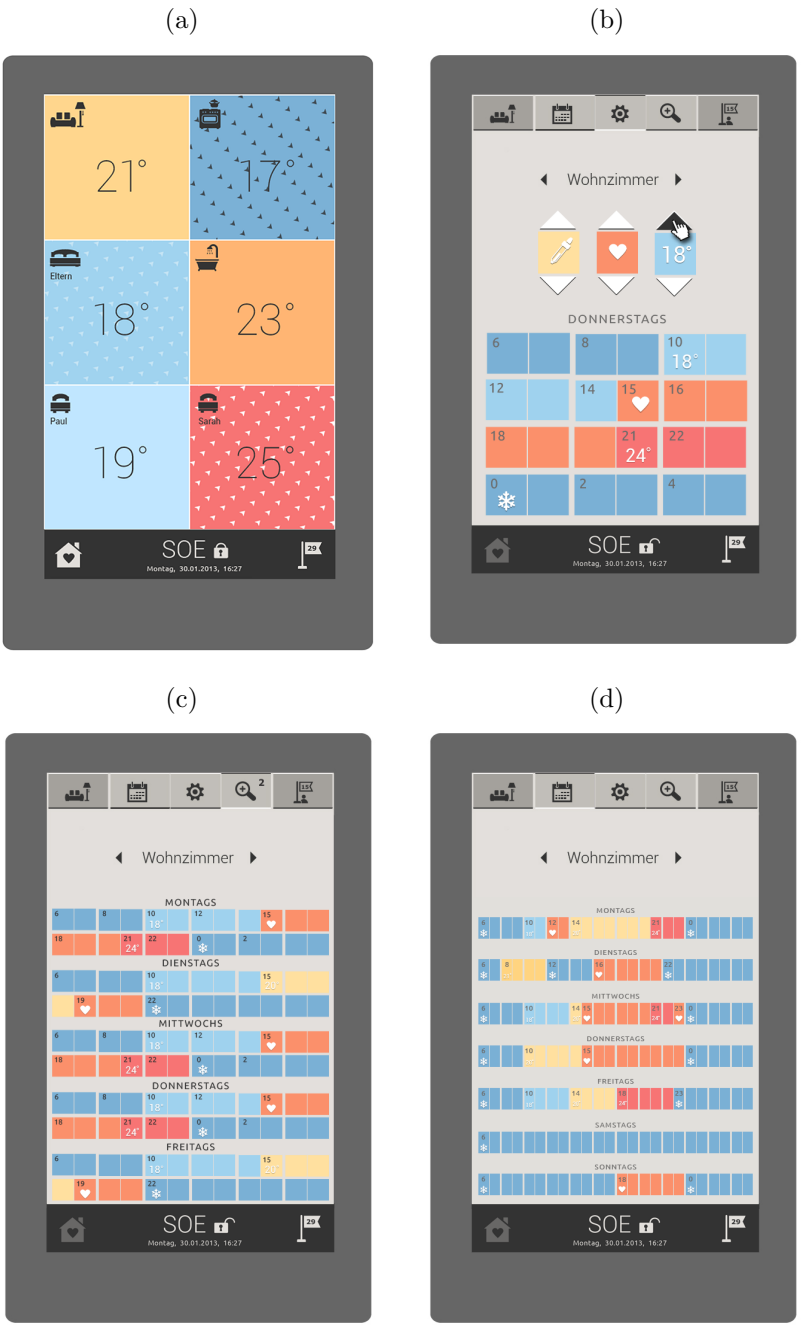


Figure 7.3: Graphical user interface (GUI) of SOE heating control system, showing the temperature settings for (a) all rooms at current time, (b) one individual room for a specific weekday, (c) a single room for all workdays, and (d) one individual room for the entire week. The manually created heating schedules are compared against the automatically optimized scheme in order to give recommendations for possible energy savings at idle time intervals.

7.2 BestTime
Finding Representatives in Time Series Datasets

Given a set of time series, we aim at finding representatives which best comprehend the recurring temporal patterns contained in the data. We demonstrate BestTime, a Matlab application that uses recurrence quantification analysis to find time series representatives.

7.2.1 Introduction

This work presents BestTime, a platform-independent Matlab application with graphical user interface, which enables us to find representatives that best comprehend the recurring temporal patterns contained in a certain time series dataset. Although BestTime was originally designed to analyze vehicular sensor data and identify characteristic operational profiles that comprise frequent behavior patterns [179], our extended version [188] can be used to find representatives in arbitrary sets of single- or multi-dimensional time series of variable length.

Our approach to find representatives in time series datasets is based on agglomerative hierarchical clustering [112]. We define a representative as the time series that is closest to the corresponding cluster center of gravity [138]. Since we want a representative to comprehend the recurring temporal patterns contained in the time series of the respective cluster, we need a distance measure that accounts for similar subsequences regardless of their position in time [179].

However, traditional time series distance measures, such as the Euclidean distance (ED) and Dynamic Time Warping (DTW), are not suitable to match similar subsequences that occur in arbitrary order [9, 36]. Hence, we propose to employ Recurrence Plots (RPs) and corresponding Recurrence Quantification Analysis (RQA) [132, 204] to measure the pairwise (dis)similarity of time series with similar patterns at arbitrary positions. In earlier work [185] we introduced a novel Recurrence Plot-based distance measure, which is used by our BestTime tool to cluster time series and find representatives.

We formalize the introduced problem in Section 7.2.2. Our proposed solution is presented in Section 7.2.3. The operation of our BestTime tool is described in Section 7.2.4 and illustrated in Figure 7.4 - 7.5. Supplementary material, including the executable Matlab code of BestTime, real-life data for testing, and a video demonstration, can be found online [188]. We conclude with future work in Section 7.2.5.

7.2.2 Problem Statement

Given a time series $X = \{x_1, ..., x_n\}$ of length n with $x_i \in \mathbb{R}^d$, a pattern S of X is a subsequence $S = \{x_i, \ldots, x_{i+l-1}\}$ of $l < n$ consecutive values, for $1 \leq i \leq n - l + 1$. Assuming a set $\mathbb{X} = \{X_1, \ldots, X_t\}$ of d-dimensional time series of variable length, the goal is to find $k << t$ representatives $\mathbb{Z} = \{Z_1, \ldots, Z_k\} \subseteq \mathbb{X}$ that comprise as much patterns from $\mathbb{X} \backslash \mathbb{Z}$ as possible.

7.2.3 Finding Representatives

Our approach consists of three consecutive steps including 1.) the calculation of a distance matrix that stores the pairwise distances between all time series objects in the considered database, 2.) the clustering of all time series objects according the precomputed distances, and 3.) the selection of representatives that are in close proximity to the found cluster centers.

1. Calculating Distance Matrix.

Let $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_m\}$ be two d-dimensional time series of length n and m respectively, then for the similarity thresholds $\epsilon^1, \ldots, \epsilon^d \geq 0$, we define a recurrence matrix $R^{d,\epsilon}$ as follows:

$$R^{d,\epsilon}_{i,j} := \prod_{k=1}^{d} \Theta(\epsilon^k - |x^k_i - y^k_j|) \tag{7.1}$$

$$i = 1 \ldots n, \quad j = 1 \ldots m, \quad \Theta(\cdot) \text{ Heaviside function.}$$

Note that $R^{d,\epsilon}_{i,j} = 1$, if $|x^k_i - y^k_j| \leq \epsilon^k$ holds for all $k = 1 \ldots d$ dimensions, and $R^{d,\epsilon}_{i,j} = 0$ otherwise. That means $R^{d,\epsilon}_{i,j} = 1$, if X at time i is similar to Y at time j. Furthermore, note that diagonal lines of 'ones' in $R^{d,\epsilon}$ indicate that X and Y are similar over a certain time interval, and, thus, reveal patterns that co-occur in X and Y [132].

Since we aim at finding representatives which comprehend patterns that co-occur in other time series, we propose to quantify the diagonal line structures in $R^{d,\epsilon}$ by means of the determinism DET [132, 204]:

$$DET = \frac{\sum_{l=l_{min}}^{\min(n,m)} l \cdot P^{\epsilon}(l)}{\sum_{\substack{i=1,\ldots,n \\ j=1,\ldots,m}} R^{d,\epsilon}_{i,j}} \tag{7.2}$$

where P^ϵ is the histogram of the lengths of diagonal lines in $R^{d,\epsilon}$, i.e. $P^\epsilon(l)$ is the number of diagonal lines of length l. Consequently, the determinism (DET) reflects the percentage of recurrence points or entries in $R^{d,\epsilon}$ that form diagonal lines of minimum length l_{min} [132].

Having quantified the minimum length patterns, we are finally in the position to define our time series distance measure as:

$$d(X, Y) := 1 - DET \tag{7.3}$$

Note that $d(\cdot, \cdot) \in [0, 1]$, where high values correspond to low similarity, and vice versa. Given a set $\mathbb{X} = \{X_1, \ldots, X_t\}$ of t time series, we store the pairwise distances in a matrix D of size $t \times t$, where $D_{i,j} = d(X_i, X_j)$ for $i, j = 1, \ldots, t$.

2. Clustering Time Series.

Given a desired number of representatives k and the predetermined distance matrix D, we employ hierarchical clustering with complete linkage to map each time series of our dataset $\mathbb{X} = \{X_1, \ldots, X_t\}$ to a corresponding class $\mathbb{C} = \{C_1, \ldots, C_k\}$, so that $c : \mathbb{X} \to \mathbb{C}$. The linkage criterion determines the distance between sets of observations as a function of pairwise distances between observations:

$$\max\{D_{i,j} = d(X_i, X_j) : c(X_i) \neq c(X_j)\} \tag{7.4}$$

Although we propose to use hierarchical clustering on account of its deterministic behavior, one could also employ any other cluster algorithm for this step.

3. Selecting Representatives.

Having split our dataset $\mathbb{X} = \{\mathbb{X}_1, \ldots, \mathbb{X}_k\}$ into k disjoint subsets, so that $c(\mathbb{X}_i) = i$ for $1 \leq i \leq k$, we are eventually able to select a time series representative $Z_i \in \mathbb{X}_i$ for each cluster in the following manner:

$$Z_i := \underset{X \in \mathbb{X}_i}{\arg\min} \sum_{Y \in \mathbb{X}_i} d(X, Y) \tag{7.5}$$

According to our definition, a representative is the object most similar to all other objects in the respective cluster. In regard to our proposed distance measure, a representative is therefore the time series that comprehends as many as possible patterns contained in the considered dataset or subset.

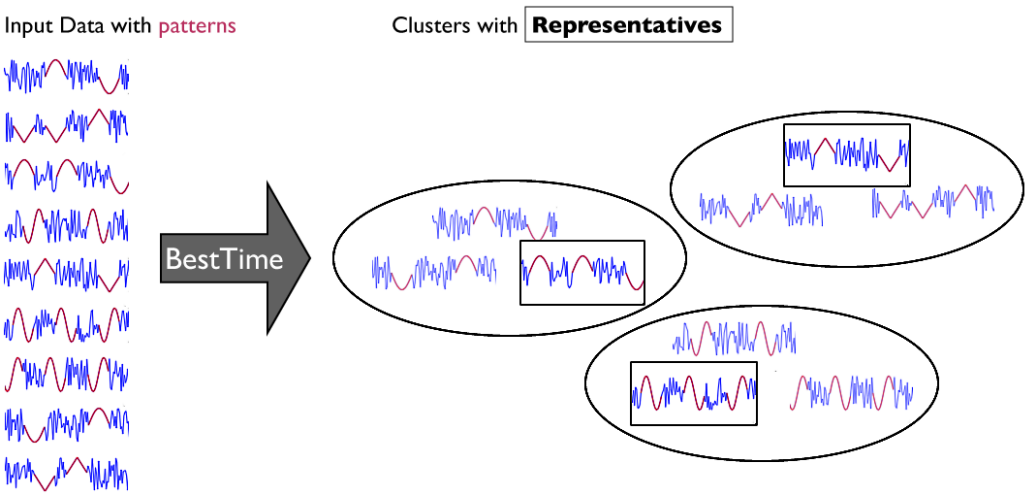


Figure 7.4: Given a set of time series with previously unknown patterns, we aim to cluster the data and find a representative (highlighted) for each group.

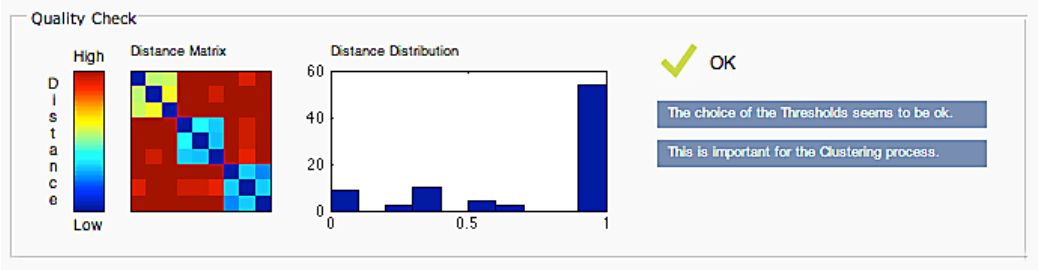
7.2.4 BestTime

BestTime is a platform-independent Matlab application, which provides an user-friendly interface. It enables a user to find representatives in arbitrary time series datasets by clustering the data sequences according to co-occurring patterns. In the following we briefly describe the operation of our BestTime application and illustrate the data processing for a small set of sample time series in Figure 7.4 and 7.5. Please feel free to download our BestTime tool [188] to follow the stepwise operating instructions given below.

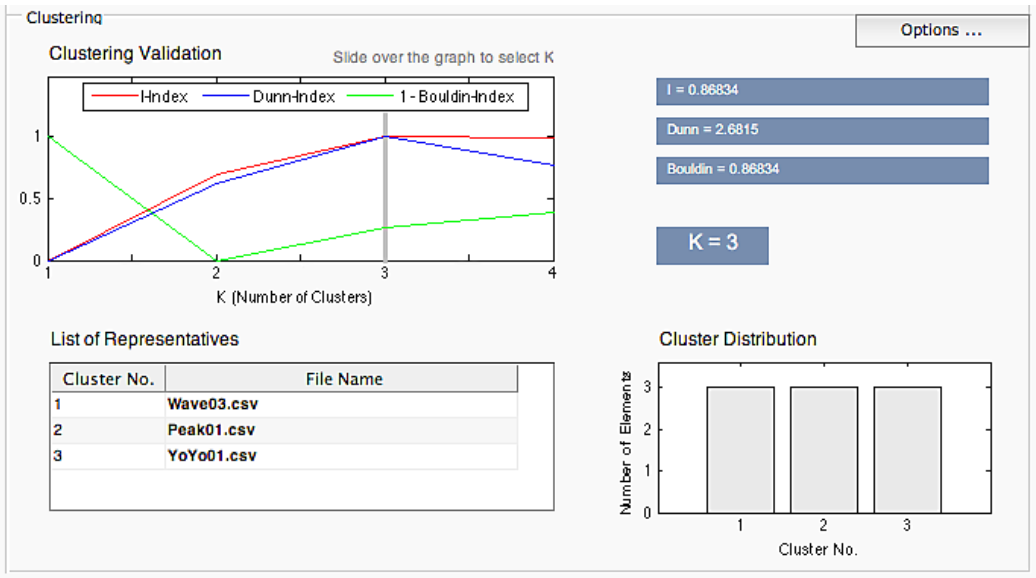
Input Data. BestTime is able to analyze multivariate time series with same dimensionality and of variable length. Each individual time series needs to be stored in an independent csv (comma separated values) file, where rows correspond to observations and columns correspond to variables. Optionally, the first row may specify the names of the variables. The user selects an input folder that should contain all time series in specified csv format. A small set of sample time series that we use as input is illustrated in Figure 7.4.

Minimum Number of Observations. Depending on the application, the user can optionally reduce the size of the dataset by specifying the minimum length of the time series which should be consider for further processing.

(a)



(b)



(c)

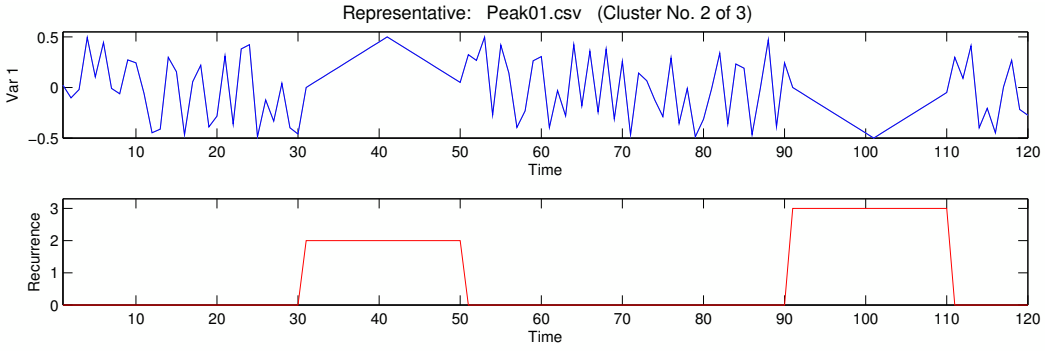


Figure 7.5: BestTime operation and data processing for finding representatives in time series datasets, exemplified on sample time series introduced in Figure 7.4. (a) Visualization of computed distance matrix and distance distribution, which are used to ensure both appropriate parameter settings and clusters that preserve the time series characteristics. (b) Clustering results, which show various validation indexes for a changing number of clusters, the list of identified representatives for a selected number of clusters, and the cardinality of the individual clusters. (c) Detailed view of a representative and its corresponding pattern frequency with regard to the selected cluster.

Data Reduction Rate. Since the cost of our pairwise distance calculations is quadratic in the length of the time series, we offer the possibility to reduce the length via piecewise aggregate approximation [36]. Given a time series of length n and a reduction rate r, the approximate time series is of length n/r.

Minimum Pattern Length. As described in Sec. 7.2.3, the predetermined minimum pattern length l_{min} directly influences the time series similarity. This parameter strongly depends on the application and needs to be chosen by a domain expert.

Variable Selection. In case of time series datasets with multiple dimensions, the user interface of our tool offers the possibility to select the variables that should be considered for further analysis.

Similarity Threshold. This parameter is usually very sensitive and directly influences the clustering result. Since it may be challenging to determine an appropriate similarity threshold ϵ for each variable, our tool can alternatively recommend (estimated) thresholds.

Parallel Computing. Calculating the distance matrix is costly for large datasets. However, this step is fully parallelized and runs almost n_{CPU}-times faster than serial processing. Up to twelve parallel workers are supported.

Quality Control. Our tool presents a colored plot of the computed distance matrix and a histogram of the distance distribution in order to ensure appropriate parameter settings as well as clusters that preserve the time series characteristics. Since both plots are updated iteratively during distance calculations, we can abort computation anytime the preview suggests undesired results. For the distance matrix, a high variance in the distances/colors indicates an appropriate parameter setting, and a low variance in the distances/colors may result in poor clustering. In general, good clustering results can be achieved when the distances do not accumulate at either end of the interval (all close to zero or one). Figure 7.5(a) shows the quality control for our sample dataset.

Clustering Validation. To support the user in choosing an optimal number of k clusters or representatives, our tool validates the cluster goodness for changing k according to three cluster validation indexes. Figure 7.5(b) shows the cluster validation for our sample dataset.

Cluster Distribution. The clustering may result in groups of different size. Our tools illustrates the cluster distribution to identify outliers and emphasize prominent groups with expressive representatives. For our sample dataset all clusters have the same size, see Figure 7.5(b).

List of Representatives. Since we aim at finding representatives, our tool does not only show a list of identified candidates as illustrated in Figure 7.5(b), but also allows to visualize the time intervals or patterns that co-occur in other time series of the same cluster, see Figure 7.5(c).

7.2.5 Conclusion and Future Work

We have introduced BestTime, a Matlab tool, which implements a recurrence-plot based approach to find time series representatives that best comprehend the recurring temporal patterns in a corresponding dataset. Furthermore, we provide supplementary online material [188], which includes our BestTime tool, real-life testing data, a video demonstration, and a technical report. In future work we plan to reduce the computational complexity of pairwise (dis)similarity comparisons by means of an approximate distance measure.

7.3 MatArcs
Visualization of Recurring Patterns in Time Series

In statistics, exploratory data analysis (EDA) is an approach to analyzing datasets to summarize their main characteristics, often with visual methods. A statistical model can be used or not, but primarily EDA is for seeing what the data can tell us beyond the formal modeling or hypothesis testing task.

We developed *MatArcs*, a novel EDA technique which allows to identify recurring patterns in time series that merit further study. The main advantages of *MatArcs* are the compact visualization of multivariate time series and the localization of recurring subsequences with variable length. Our proposed algorithm is able to fulfill the multi-layered task of exploratory data analysis, which includes i) the processing of multivariate and high-dimensional time series (even with noise and distortions), ii) the identification of recurring subsequences or patterns that describe the underlying structure of the data, and iii) the visualization of the extracted structure in a comprehensive way.

7.3.1 Introduction

Data visualization techniques are very important for data analysis, since the human eye has been frequently advocated as the ultimate data mining tool [115]. However, there has been surprisingly little work on visualizing multivariate and high-dimensional time series in an intuitive way, so that the human eye can easily recognize the underlying structure of the data.

In case of time series the underlying structure can be described by recurring subsequences or patterns that represent certain events or behavior (e.g. periodicities and irregular cyclicities in climate data). Identified patterns cannot only be used to understand the structure of an individual time series, but also to compare the structure of different temporal measurements.

Our proposed *MatArcs* approach is able to visualize the patterns of multivariate time series (that consist of multiple temporal data sequences of same length with measurements taken at identical time points) in an two-dimensional plot. The basic idea of our novel visualization approach is exemplified in Figure 7.6, where the recurring patterns of an univariate time series are illustrated. In general, we use arcs to visualize the connections between recurring time series patterns and semicircles of different size to indicate the relative frequency of the identified patterns. The main purpose of this visualization technique is to make complex time series data accessible to the human eye, depicting recurring patterns that describe the time series structure. We employ Euclidean distance to find similar time series subsequences, although any other distance measure could be used to identify recurring patterns.

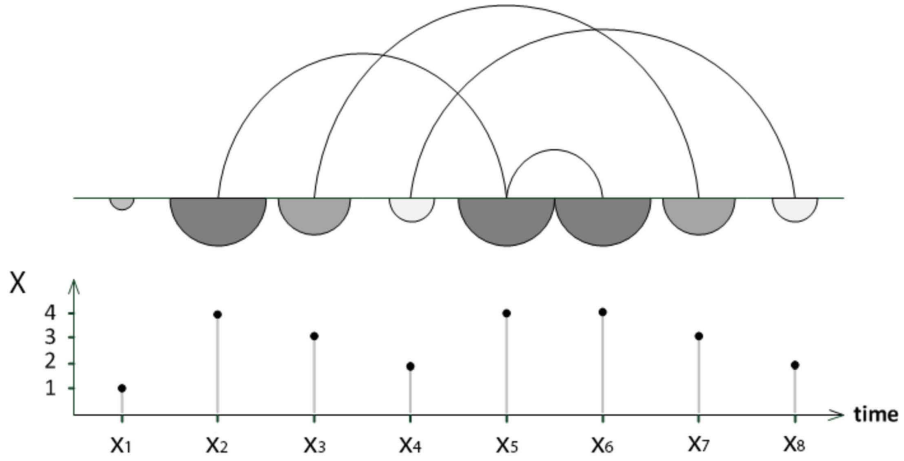


Figure 7.6: The proposed MatArcs tool uses arcs to visualize the connections between recurring time series patterns and semicircles of different size to indicate the relative frequency of the identified patterns. If we consider individual measurements as subsequences of length one, the most frequent pattern in the illustrated sample time series is value 4, which occurs at time point 2, 5 and 6 respectively.

The introduced *MatArcs* visualization technique meets the objectives of exploratory data analysis (EDA) [198], due to the fact that it is able to: i) suggest hypothesis about the causes and of observed phenomena, ii) assess assumptions on which statistical inference will be based, iii) support the selection of appropriate statistical tools and techniques, and iv) provides a basis for further data collection. Moreover *MatArcs* is novel in that it i) uses arc diagrams to visualize recurring time series patterns, ii) differentiates patterns by semicircles of different size and color, iii) allows to illustrate multivariate states with help of joint recurrence plots, which are explain in detail in Section 7.3.3.

The rest of this work is structured as follows. Section 7.3.2 provides background on related work. In Section 7.3.3 we introduce our *MatArcs* approach and visualization framework. A case study on Google Trends data is presented in Section 7.3.4. We conclude with future work in Section 7.3.5.

7.3.2 Related Work

In exploratory data analysis (EDA) literature there exist a wide range of visualization approaches, including graphical techniques such as histograms and scatter plots. However, most traditional EDA techniques do not consider the chronological ordering of observations or produce ambiguous time

series visualizations [62, 203]. Although the performance of existing EDA approaches is hard to measure, common sense tells us that an applicable visualization techniques should contribute to the understanding of the data and illustrate underlying patterns in their natural occurring order. Given the limitations of the human eye in regard to spatial perception, it is often challenging to comprehend the immense information of large datasets in an individual visualization.

Scatter plots, for instance, are a type of mathematical diagram using Cartesian coordinates to display values for two variables for a set of data. However, they are not suitable to visualize multiple variables and temporal correlations such as existent in multivariate time series. This limitation not only based on the fact that the scatter plots are restricted to three space dimensions and the color spectrum, but also arise from the loss of temporal dependence.

Arc diagrams [145], in contrast, address the before mentioned shortcomings of illustrating recurring temporal patterns. They were first introduced in the field of music theory as a technique to visualize complex coherent structures, which promote deeper understanding.

For example in tonal analysis, arc diagrams were successfully used to examine repeating structures [203]. The input data (of the implemented tonal analysis algorithm) is represented as strings, which are subsequently interpreted as univariate time series. Although this approach keeps the identified temporal patterns in a suffix tree, it does not interpret the significance of the repeating sequences and merely reflects the complexity of a dataset. Moreover, the described tonal analysis algorithm was not designed to identify multivariate patterns such as found in time series with multiple variables.

Another use case of arc diagrams is the visualization of threads found in email, where *thread arcs* combine the chronology of messages [96]. Although *thread arcs* have been shown to produce compact visualizations that reveal relevant messages, the approach does not meet the requirements of recognizing recurring multivariate patterns of arbitrary length.

Since traditional EDA techniques generally neglect the visualization of temporal aspects and existing arc diagrams are unable to illustrate multivariate patterns, there is a practical necessity for new visualization techniques that allow to depict recurring temporal patterns in multivariate time series. The main challenge is how to represent the multivariate pattern in a two-dimensional visualization in order to make the data easily accessible to the human eye. Furthermore, an applicable time series visualization technique should not only except strings (sequences of symbols) as an input[62, 203], but also other types of temporal measurements with/out time reference.

7.3.3 MatArcs Visualization

This section presents our proposed MatArcs visualization framework, which is implemented in Matlab and based on Arc diagrams. In the following we introduce the mathematical notion of the problem and describe the system architecture of the framework, as shown in Figure 7.7.

Notation

Assume a time series $X = \{x_1, \ldots, x_n\}$ with $x_j \in \mathbb{R}^d$ for $1 \leq j \leq n$. We can also consider X as a matrix of size $d \times n$, where x_{ij} corresponds to the element of row i and column j, for $1 \leq i \leq d$ and $1 \leq j \leq n$ respectively. Given X, we can define a pattern $S(a, b) = \{x_a, \ldots, x_b\}$ of length $l = b - a + 1$ as a subsequence of l consecutive multivariate states $x_j = \{x_{1j}, x_{2j} \ldots, x_{dj}\} \in \mathbb{R}^d$ for $1 \leq a \leq j \leq b \leq n$.

Furthermore, suppose that $d : S(a_1, b_1) \times S(a_2, b_2) \rightarrow \mathbb{R}_+$ is a distance function that determines the pairwise (dis)similarity between subsequences $S(a_1, b_1), S(a_2, b_2) \in X$ of same length $b_1 - a_1 = b_2 - a_2$. We regard two subsequences $S(a_1, b_1)$ and $S(a_2, b_2)$ as a recurrent pattern in X if they are (i) non-overlapping, i.e. $1 \leq a_1 \leq b_1 < a_2 \leq b_2 \leq n$, and (ii) similar, i.e. $d(S(a_1, b_1), S(a_2, b_2)) \leq \epsilon$, where $\epsilon \geq 0$ is a threshold.

For the pairwise (dis)similarity comparison of (sub)sequences we use the L^2-norm ($\|X - Y\|_2$), although any other (dis)similarity function can be chosen. In case of multivariate time series with many variables (large d) the L^∞-norm might be more computational efficient.

In order to visualize recurring patterns or time series subsequences we employ recurrence plots [132]. A recurrence plot of a univariate time series $X = \{x_1, \ldots, x_n\}$ (with $d = 1$) can be formalized by the following equation:

$$R_{i,j}^{d,\epsilon} = \Theta(\epsilon - ||x_i - x_j||) \tag{7.6}$$

where $x_i, x_j \in \mathbb{R}^1$ are univariate states at times i and j (for $i, j = 1 \ldots n$), and $\Theta(\cdot)$ is the Heaviside step function ($\Theta(z) = \{1|z > 0; 0|z \leq 0\}$).

The recurring patterns of a multivariate time series $X = \{x_1, \ldots, x_n\}$ (with $x_j \in \mathbb{R}^d$ for $1 \leq j \leq n$) can be visualized by a joint recurrence plot, which is formalized in the following equation:

$$JR_{i,j}^{d,\epsilon} = \prod_{k=1}^{d} \Theta(\epsilon^k - ||x_i^k - x_j^k||), \tag{7.7}$$

where $\epsilon = \{\epsilon^1, \ldots, \epsilon^d\}$ is chosen for each variable, and x_j^i can also be expressed as x_{ij}. The joint recurrence plot is an extension of the traditional

recurrence plot, in that it simultaneously computes d recurrence plots which are subsequently joint by logical conjunction. Note, that the individual dimensions or variables are allowed to describe different physical quantities, but are required to have the same number of measurements. For example, we can compute a joint recurrence plot for weather data containing humidity and temperature measurements of each minute for one and the same day.

In the following section we describe the individual step of processing multivariate time series for the proposed visualization.

Architecture

The proposed MatArcs framework consists of three main components, namely i) input, ii) solver, and iii) visualization. These components and the data processing pipeline are illustrated in Figure 7.7. The functionality of the individual components is discussed in the following paragraphs.

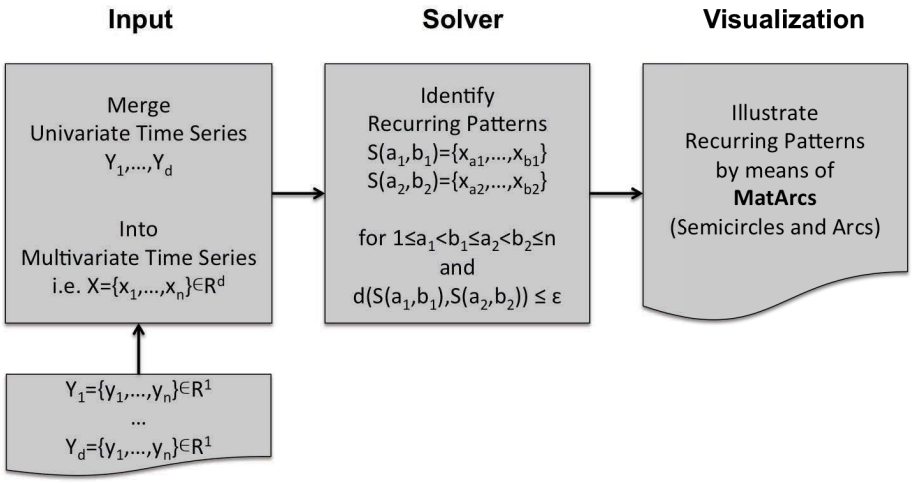


Figure 7.7: MatArcs architecture, illustrating the data processing pipeline.

Input: As with most data mining problems, our time series visualization requires a specific data format, which typically involves data preprocessing. For instance, if one aims to provide a visualization of an input that consists of multiple univariate time series with heterogeneous time scales. Our MatArcs framework addresses this issue by combining the input set of univariate time series to one multivariate time series. In case of heterogeneous time scales or unequal length, the univariate time series need to be aligned prior to combination. The combination of d univariate time series $Y_1 \ldots Y_d$ of length n to one multivariate time series $X \in \mathbb{R}^{d \times n}$ is carried out by the input component, as illustrated in Figure 7.7.

The filtering or smoothing of the univariate time series is unnecessary and even undesired, because rectified outliers should not contribute to recurring patterns. Consequently NaN-values (which might possibly correspond to measurements errors) are simply ignored.

Solver: The identification of recurring patterns in multivariate time series requires the pairwise similarity comparison of all non-overlapping subsequences of same length. Our MatArcs visualization just requires to compare all multivariate patterns of length one, since patterns of higher length are illustrated by several successive arcs (see following paragraph on visualization component). In general, two multivariate patterns of same length l are considered to be similar if each multivariate state $1, \ldots, l$ is similar. In turn, a multivariate state is similar if the distance (i.e. L^2-norm) of the individual variables does not exceed the respective threshold. In case that we aim to identify recurring patterns that are absolute identical ($\epsilon = 0$ for all variables), Equation 7.7 can be simplified as followed:

$$JR_{i,j}^{d,\epsilon} = \prod_{k=1}^{d} \Theta(0 - ||x_i^k - x_j^k||_2) = \Theta(-||x_i - x_j||_\infty) \qquad x_i, x_j \in \mathbb{R}^d$$

According to the above formalization, the joint recurrence can be either understood in terms of the L^∞ norm or interpreted as a logical conjunction of individual recurrence plots (one for each variable). In case of an univariate time series the recurrence plot can be directly derived from the distance matrix. A toy example for the sample time series $X = \{2, 4, 3, 5, 2, 4\}$ and the similarity threshold $\epsilon = 0$ is shown in Figure 7.8.

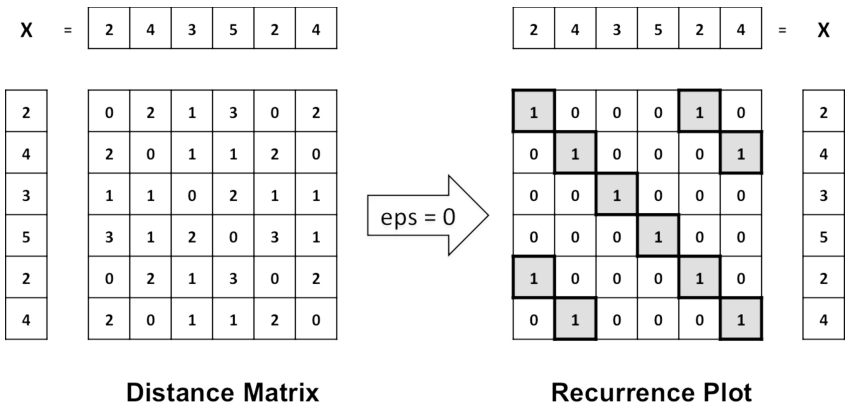


Figure 7.8: Distance Matrix and Recurrence Plot of Sample Time Series.

As illustrated in Figure 7.8 a distance matrix can generally be regarded as an unthresholded recurrence plot. Applying a similarity threshold to the distance matrix, as formalized in Equation 7.7, reveals structures that indicate recurring patterns (i.e. non-zero entries within the recurrence plot).

Note that recurrence plots are usually symmetric and their main diagonal always exhibits non-zero entries, since it reflects the line of identity. Therefore, we merely consider the upper triangle of the recurrence plot for further processing, thus cutting the computational effort in half. For example, the recurrence plot in Figure 7.8 reveals that the measurements taken at time point one and five ($x_1 = 2 = x_5$) as well as two and six ($x_2 = 4 = x_6$) are similar to each other.

In general, a relatively small similarity threshold might lead to few or no recurrences, whereas a comparatively high similarity threshold often causes noisy recurrence plots. As a rule of thumb, the similarity threshold should not exceed 10% of the maximum phase space diameter and the recurrence point density should be approximately 1%. However, the selection of this parameter strongly depends on the application and examined data. Although it is possible to clean the recurrence plot before visualizing the identified patterns, for instance by removing redundant or spurious information, this is out of scope for our use case.

Visualization:

Given a (multivariate) time series, our MatArc visualization aims at illustrating the recurring patterns identified in the previous step. The MatArcs visualization consists of three steps, including the presentation of i) arcs that connect recurring patterns, ii) semicircles that indicate the frequency of patterns, and iii) colors that differentiate between patterns.

In our visualization approach the size of the arcs corresponds to the temporal distance between recurring patterns, where the height of an arc is half the distance. For example, the recurring pattern $x_1 = 2 = x_5$ in Figure 7.8 has a temporal distance of $|1 - 5| = 4$ time units.

The radius of the semicircles is proportional to the frequency of the patterns and is calculated by the ratio between the number of non-zero elements in the corresponding row and the entire recurrence plot: $r_i = \frac{\sum_{j=1}^{n} JR_{i,j}}{\sum_{i,j=1}^{n} JR_{i,j}}$.

Finally all arcs and semicircles that belong to one recurring pattern are highlighted in the same color, which makes our MatArcs visualization more accessible to the human eye and simplifies the interpretation.

7.3.4 Case Study

Although the introduced MatArcs visualization is applicable to time series from arbitrary domains, we want to demonstrate the practicality of our approach for some real-life data. In the following we present a case study on *Google Trends* data, which is publicly available on the internet. A *Google Trend* generally contains the number of searches for a specific search request over a certain period of time. The data is available as .csv-file and normalized according the average number of requests for the corresponding search term[3]. In particular, we consider the search term *renewable energy* for two different countries, namely Germany and the United States. We choose 01/Jan/2004 to 26/Mar/2012 as time interval, which consists of around 400 measurements (one for each week).

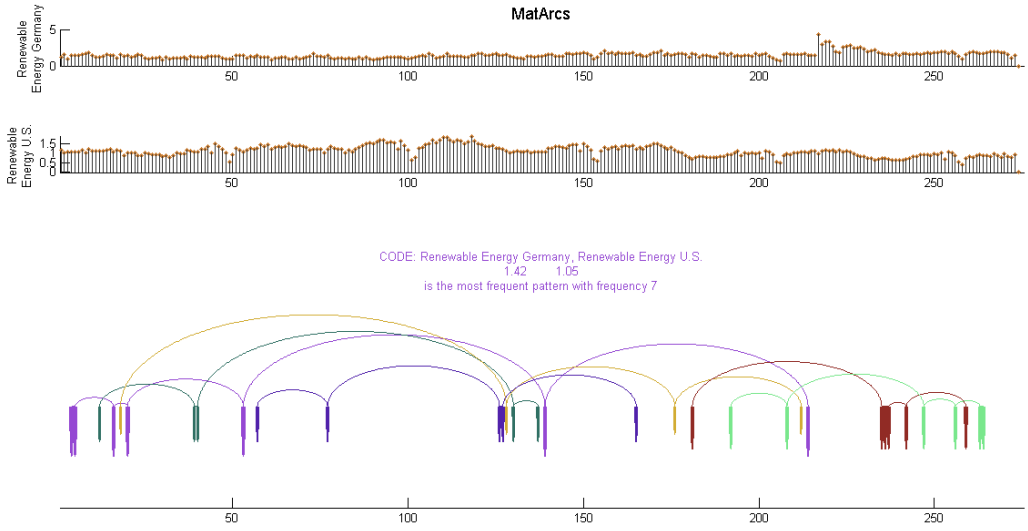
Given the described trend data for *renewable energy*, we want to identify recurring patterns (e.g interest or disinterest in that topic) which occur jointly in both countries. In the first step, the input component of our visualization framework merges the two univariate time series into one multivariate time series, since we are interested in multivariate patterns. Secondly, the solver component computes the joint recurrence plot of the multivariate trend time series and thereby identifies multivariate patterns that recur within the specified time interval. Eventually, the visualization component is able to plot the identified patterns, using the proposed arc and semicircles representation. The input time series and the visualization result is illustrate in Figure 7.9(a) and 7.9(b) in detail.

In our case study we determine the similarity threshold for both time series to be 5% of the standard deviation. Although we can affect the number of identified patterns by changing the similarity threshold, we should bear in mind that the visualization needs to be clear and concise. Furthermore, it is important to mention that the similarity threshold directly influences the computation effort needed to plot the patterns. Figure 7.9(b) shows an excellent example of a visualization that is neither to crowded with information nor misses any details. Since an inappropriate setting of the similarity threshold can easily cause noisy recurrence plots, we only take patterns into consideration that recur at least three times. This constraint ensures concise results and prevents computational expensive visualizations.

Figure 7.9(a) and 7.9(b) illustrate the described *Google Trends* data that is used as input as well as the MatArcs visualization result. The trend data indicates the number of search requests for the *Renewable Energy* term, where the x-axis represents the time domain and the y-axis depicts the frequency of the search term. Note that the frequency values are continuous, but the

[3] `www.google.com/intl/en/trends/about.html`, Chapter 7. How is the data scaled

(a)



(b)

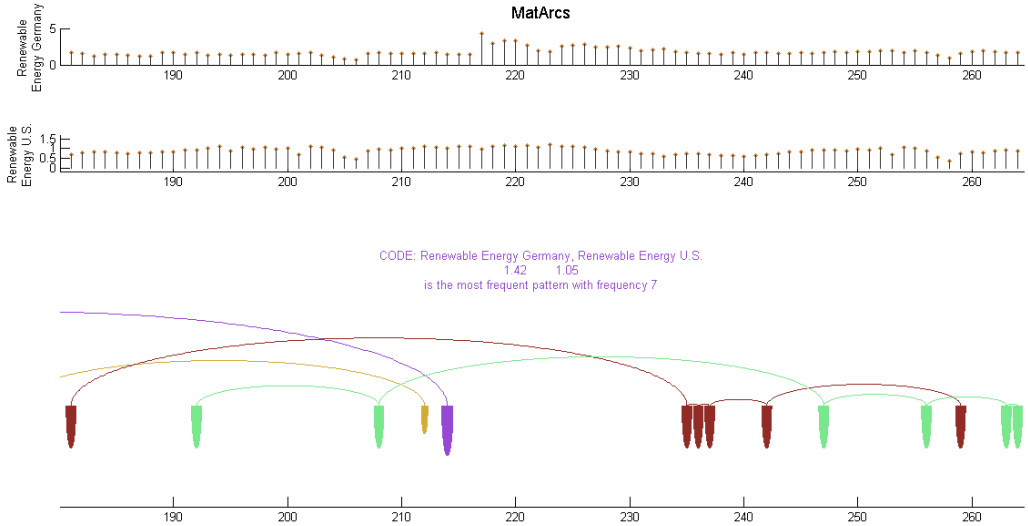


Figure 7.9: (a) MatArcs visualization of *Google Trends* data, which consists of two time series that describe the frequency of the search term *Renewable Energy* for two countries (Germany and the United States) over a period of more than 8 years (01/Jan/2004 to 26/Mar/2012) on a weekly basis. (b) Detailed view of above plot, showing the exact positions of the recurring patterns. Note that the color of the most frequent pattern coincides with the color of the text which states the number of recurrences. In this example the multivariate state with the values 1.42 and 1.05 occurs 7 times and is highlighted in purple color.

time points are discrete. For instance in our sample trend data for Germany, at time point 217 the search frequency for *Renewable Energy* is 4.36 times higher than the average recording of all search requests handled by *Google* during the considered time interval.

The MatArcs visualization in Figure 7.9(a) reveals that the multivariate state with values 1.42 and 1.05 (for Germany and US respectively) occurs 7 times within the examined time interval. Semicircles and arcs (that indicate the positions and connect the instances) of the most frequent pattern are highlighted in purple color. Figure 7.9(b) shows only small part of the data presented in Figure 7.9(a), which allows us a more detailed view on the identified patterns. For instance, the magnified visualization emphasizes a frequently recurring pattern (highlighted in light green color) that occurs for the first time in 2011 and then recurs in irregular time intervals.

In case of the described trend data, the multivariate patterns can be interpreted as events in which the international interest (represented by Germany and the US) in the *Renewable Energy* issue is repeatedly the same. The frequency and time span of the patterns generally give information about the significance and life cycle of an event. For example in Figure 7.9(a), the purple and yellow pattern indicate a long-term trends, whereas the green and brown pattern specify short-term trends. However, the purple pattern occurs more frequently than the yellow one, making it a more promising candidate for a significant motif or topic. Furthermore, an arc with a big radius tells us that a pattern reemerges after a long period of time, whereas several consecutive arcs with small radii are evidence of temporal patterns. Although the recurrence of multivariate patterns indicates an interrelation between the individual variables, it should not be confused with correlation or dependence as known from statistics.

As mentioned in Section 7.3.2, scatter plots are typically employed in exploratory data analysis in order to visualize the correlation or dependence of variables. Figure 7.10 shows the scatter plot of our two variables (or time series) for the search term *Renewable Energy*, as discussed above. In comparison to our MatArcs visualization in Figure 7.9(a) and 7.9(b), we can see that the scatter plot fails to reveal any structure. This is due to the fact that scatter plots lack the ability to identify temporal relationships in time series. Beyond that, scatter plots are incapable to visualize multivariate data, which contains more than three variables. Since our MatArcs visualization allows the visual examination of multivariate time series, it contributes to the exploratory data analysis community in providing new visual insights.

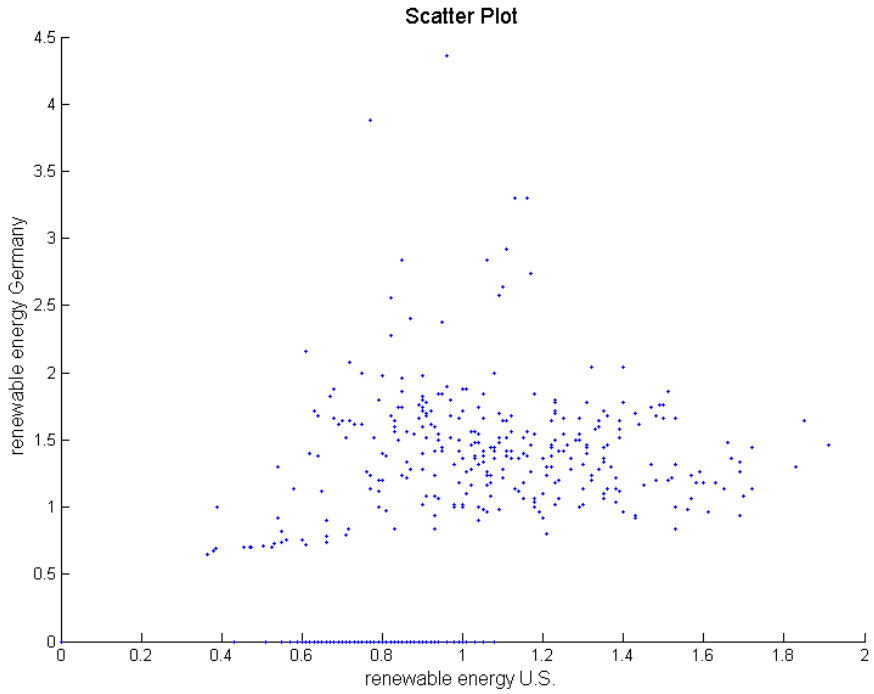


Figure 7.10: Scatter plot of *Google Trends* data for the search term *Renewable Energy*, comparing Germany and the United States. An alignment of the data points, which might indicate a correlation or dependence, is not observable.

7.3.5 Conclusion and Future Work

In this work we have proposed MatArcs, a novel visualization techniques, which enables us to identify recurring patterns in multivariate time series. Depending on the examined data, recurring pattern can represent events or situation that closely resemble each other and occur at several positions in time. Usually the recurring patterns are hidden in the data and are previously unknown, but can be uncovered by means of our MatArcs approach.

Unlike traditional techniques in exploratory data analysis, such as scatter plots, our MatArcs approach is able to visualize temporal data with multiple variables. In particular, MatArcs is designed to identify multivariate patterns in time series from arbitrary domains. To identify multivariate patterns we employ joint recurrence plots, which can be imagined as a superposition of several thresholded distance matrices - one for each variable. The introduced mathematical notation of joint recurrence allows us to visualize multivariate patterns in a clear and concise manner. In particular our MatArcs approach

illustrates the information contained in the recurrence plots by means of arcs and semicircles, which connect recurring patterns and indicate their relative frequency. We explained how to clean noisy recurrence plots and determine an appropriate similarity threshold prior visualization.

In our case study, we discussed an example of our MatArcs visualization using *Google Trends* data. In particular, we illustrated the multivariate patterns that recur jointly in the trend data for the search term *Renewable Energy* for both Germany and United States. Our case study showed that our MatArcs approach is able to identify long-term and short-term behavior with statistical significance.

Future work includes the visual examination of time series from other domains, such as vehicular sensor data, motion data, and gene sequences. Furthermore, we want to enable our MatArcs visualization to identify recurring patterns of variable length. In addition, we plan to implement an extension that allows interactive data exploration, so that changes made by the user are incorporated into the visualization in real-time.

7.3.6 Appendix: Explorative Data Analysis

In order to encourage the data mining community to use our proposed MatArcs visualization, we have developed a web framework that implements the introduced approach. Our *MatArcs* web tool is publicly available at `http://www.anytime.dai-labor.de` and can be used to visually explore multivariate time series from arbitrary domains. Some screen shots of the web interface are shown in the following:

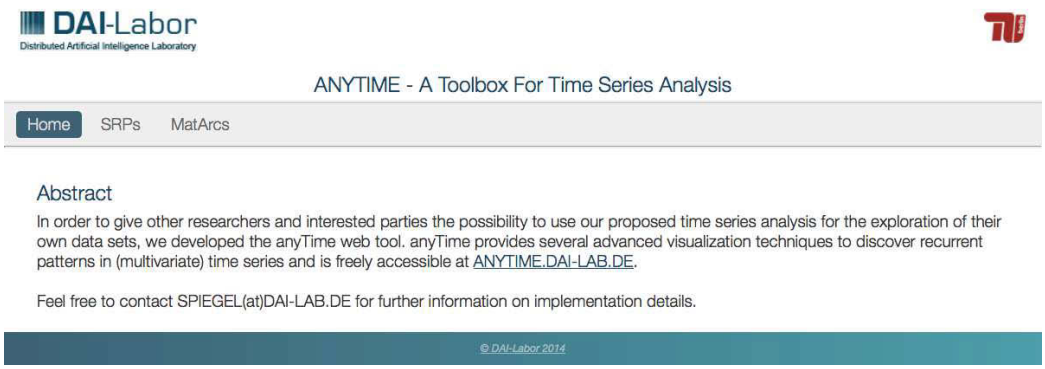


Figure 7.11: The *ANYTIME* web framework includes our *MatArcs* approach and other time series visualization techniques, which aim at identifying and illustrating recurring patterns in multivariate temporal data sequences.

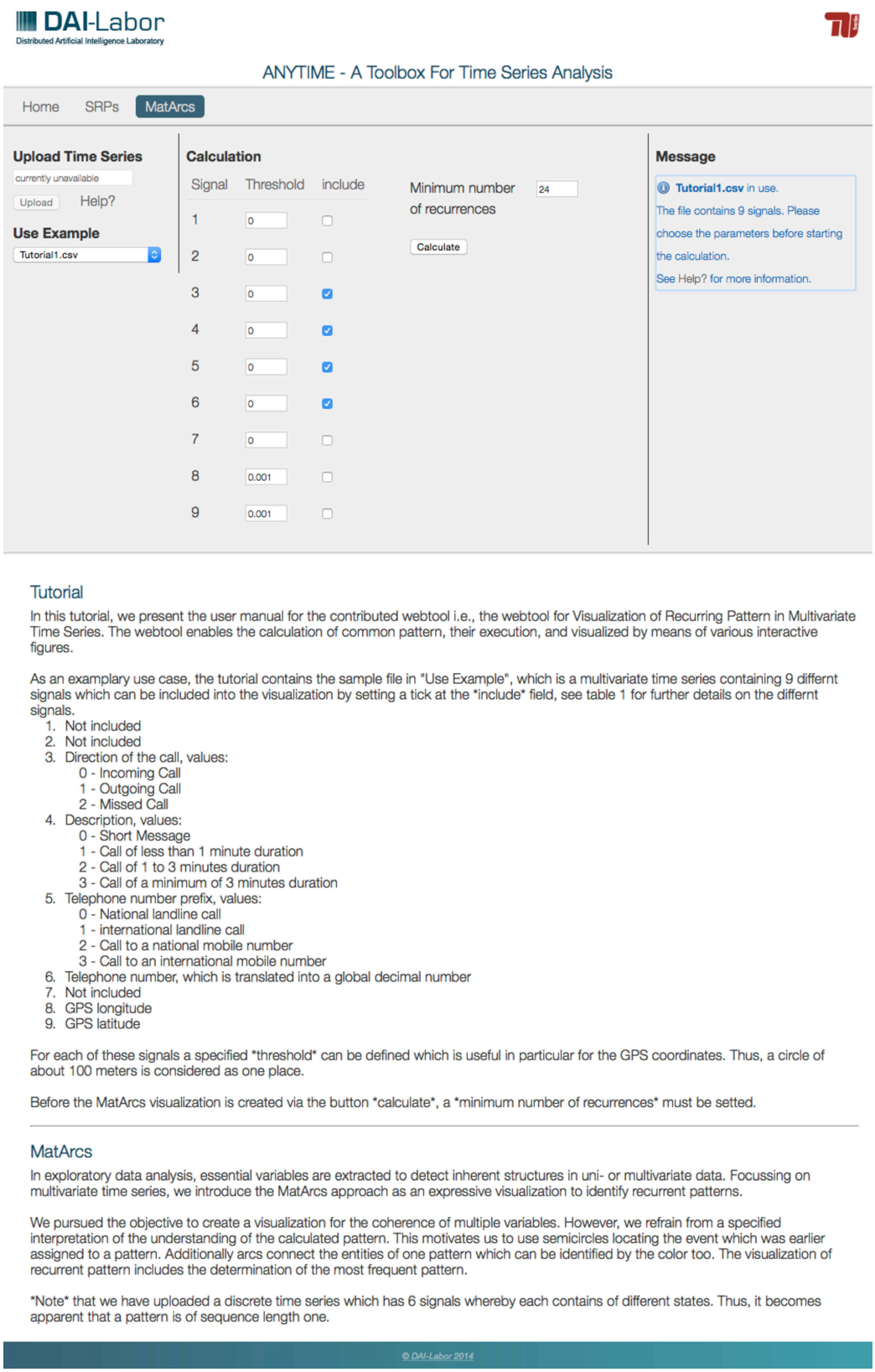


Figure 7.12: Web interface of *MatArcs* visualization tool, explaining data input, parameter selection, and result interpretation in an intuitive manner.

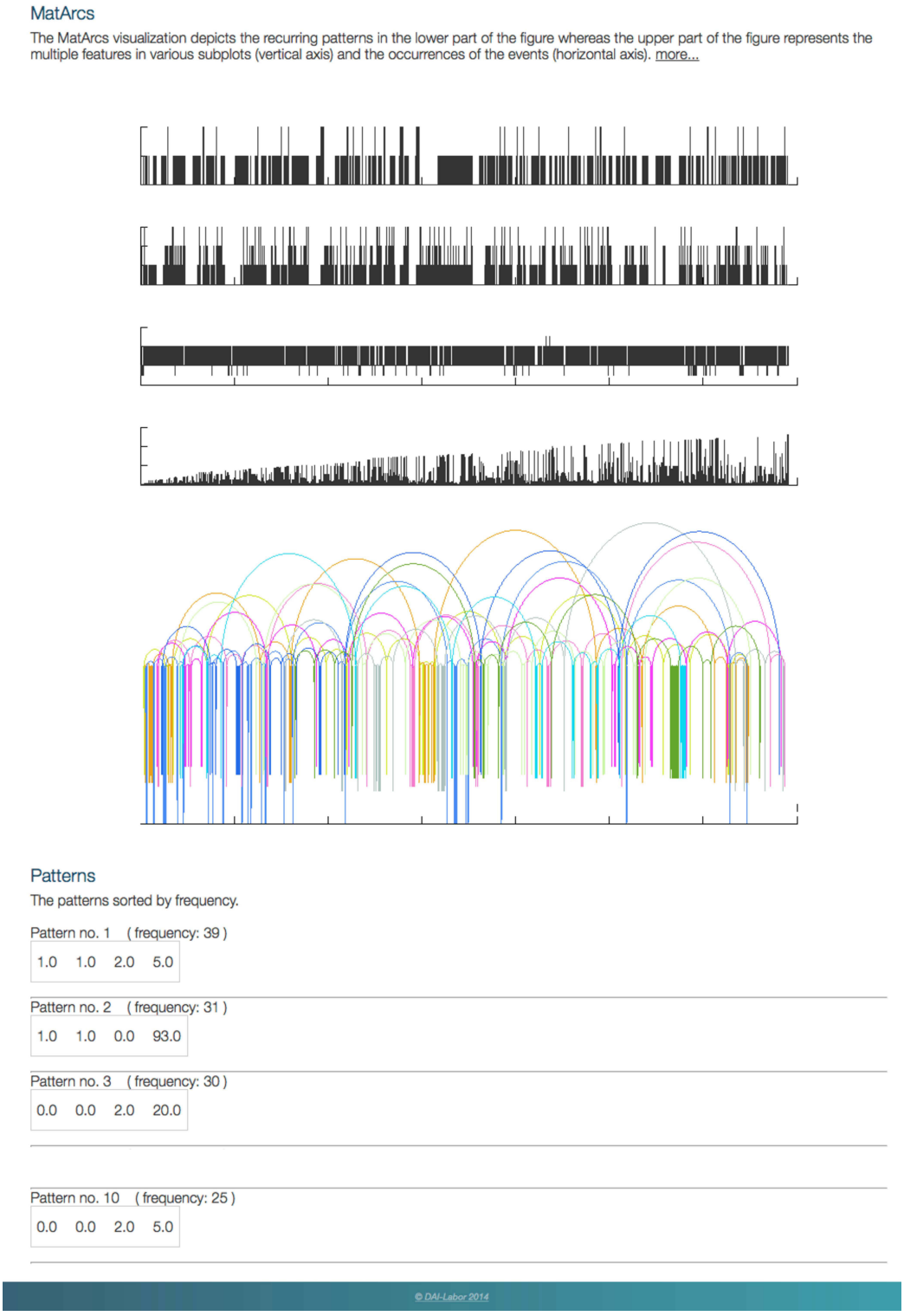


Figure 7.13: Web interface of *MatArcs* tool, showing multivariate input data, arc representation of underlying structure, and top most frequent patterns.

Chapter 8

Conclusion

This thesis considered time series analysis in the context of data mining, machine learning, and pattern recognition. The emphasis was on time series distance measures, since the pairwise (dis)similarity comparison of temporal data sequences is an essential subroutine in all time series mining tasks. We discussed the weak points of existing methods and proposed several novel distance measures for time series segmentation, classification, and clustering. Our newly introduced distance measures addressed several time series mining problems such as the distortion of temporal measurements, the processing of superimposed signals, the computational complexity of pairwise (dis)similarity comparisons, the handling of multivariate time series, and the analysis of nonlinear systems. Furthermore we demonstrated that our proposed time series distance measures are beneficial in real-world applications, including the optimization of vehicle engines in regard to exhaust emission and the optimization of heating control in terms of energy efficiency.

In Chapter 3 we proposed a generic three-step approach to the recognition of contextual patterns in multivariate time series. The crux of our approach is a factorization-based distance measure, which allows for the identification of internally homogeneous time series segments. Although we illustrated the practicability of our approach on vehicular sensor data, recognition of patterns in (multivariate) time series is an important issue in many other context-aware applications.

Chapter 4 focused on the fast classification of time series that exhibit local scaling. We introduced a novel lucky time warping distance with linear space/time complexity, which uses a greedy algorithm to accelerate distance calculations for nearest neighbor classification. Our experiments on a large variety of time series with different properties showed that the proposed lucky time warping distance trades classification accuracy against computational cost reasonably well.

In Chapter 5 we introduced another novel time series distance measure, based on the theoretical foundations of recurrence plots, which allows to determine the (dis)similarity of (multivariate) time series that contain segments of similar trajectories at arbitrary positions. We employed recurrence quantification analysis to measure the structures observed in cross recurrence plots and to investigate dynamical properties, which reflect the pairwise (dis)similarity of time series. This work was the first attempt to solve time series clustering with nonlinear data analysis and modeling techniques commonly used by theoretical physicists.

Chapter 6 gave attention to superimposed time series, in particular, aggregated energy signals such as coming from smart meters. We investigated energy disaggregation techniques to infer appliances states and consequently derive higher-level context about household occupancy. Although this chapter merely investigated energy consumption data, our literature survey and empirical evaluation of different disaggregation techniques provides a solid basis for decision making in other applications where mixed signals need to be separated into their individual sources.

In Chapter 7 we presented several of our software prototypes which were designed to solve real-world time series mining tasks. The demonstrated time series mining tools implement our introduced distance measures and provide graphical user interfaces for straightforward parameter setting and exploratory time series analysis. All of our software prototypes were developed in cooperation with industry partners and address problems that arise in domains such as car manufacturing and home automatization.

8.1 Contributions

Our work contributed to the time series community by introducing several novel distance measures, which have been shown to be useful for various time series mining tasks, such as segmentation, classification, and clustering of temporal data. The proposed time series distance measures addressed various important research problems regarding the analysis of distorted data, multivariate measurements, superimposed signals, and nonlinear systems. We not only made a contribution by improving classification accuracy and/or reducing computational complexity of existing distance measures, but we also explored and created new avenues for pattern recognition, sensor fusion, fast classification, nonlinear modeling, and other scientific aspects of time series analysis. In the following paragraphs we comment on the particular relevance of our most important findings.

8.1.1 Pattern Recognition

Pattern recognition is used in many fields, including medicine, astronomy, geophysics, engineering, and quantitative finance. In these fields many of today's applications utilize sensors to measure and observe various physical quantities. The measurements are typically represented as sequences of data points, and the recognition of certain patterns in such time series is important to retrieve higher-level information and consequently derive knowledge that is necessary for further decision-making. In this work we presented several novel strategies to model time series and find relevant patterns, which describe certain events, situations, or other higher-level context (refer to Chapter 3, 5 and 6). We demonstrated the practicability of our pattern recognition strategies on sensor data recorded by vehicles and smart meters, but our generic models also allow for the recognition of patterns in time series from other domains.

8.1.2 Sensor Fusion

Sensor fusion is the combining of sensory data from disparate sources such that the resulting information is in some sense better (more accurate, more complete, or more dependable) than would be possible when these sources would be used individually [33, 41, 212]. In conjunction with time series analysis the concept of sensor fusion involves the processing and analysis of multivariate measurements. In this work we proposed two different models that allow for combined processing and analysis of multivariate time series. Our first approach (see Chapter 3) is based on singular value decomposition, where each measurement or individual time series is represented as one vector in a matrix which is subsequently factorized to determine significant changes in correlation and identify internally homogeneous time series segments. Our second approach (see Chapter 5) is based on joint cross recurrence plots, an extension of traditional recurrence plots, which enables us to pairwise compare multivariate time series according to co-occurring patterns or segments of similar trajectory. In the context of sensor fusion both of our models are very convenient, because they allow us to process and analyze multivariate sensory data of different physical quantities simultaneously.

8.1.3 Fast Classification

Fast time series classification, which involves efficient and effective pairwise similarity comparisons as an essential subroutine, is of paramount importance when processing ever increasing amounts of temporal data. In this work we

introduced a novel lucky time warping distance, with linear space and time complexity, which uses a greedy algorithm to accelerate distance calculations (refer to Chapter 4). Our empirical results have proven that, compared to state-of-the-art techniques such as constrained dynamic time warping with lower bounding, the proposed lucky time warping distance trades classification accuracy against computational cost reasonably well. Furthermore, we have shown that our lucky time warping distance is straightforward and easy to implement, can cope with time series that exhibit warping invariance, and, therefore, is an excellent choice for one-nearest-neighbor classification of arbitrary time series. Moreover, our work on fast classification contributes to the time series community by correction erroneous believes and opening new avenues regarding optimal warping path strategies.

8.1.4 Nonlinear Modeling

In some time series applications we are confronted with complex systems, such as the human heart and brain or even vehicle engines, where independent variables affecting the system can show complex synergistic nonlinear effects and phenomena are not well expressed in traditional mathematical terms. Contrary to traditional (phenomenological) modeling, which describes a system in terms of laws of nature, nonlinear modeling can be utilized in processes or systems where there is a lack of fundamental understanding on the root causes of most crucial factors. In this work we employed recurrence plots and corresponding recurrence quantification analysis to model complex situations where relationships of different variables or measurements are not known (see Chapter 5 and 7.2). More precisely, we investigated dynamical properties like the determinism, which accounts for recurring subsequences of predetermined minimum length, to model complex drive maneuvers in vehicular sensor data. Based on the formalization of cross recurrence plots and the denotation of determinism, we defined a novel distance measure which allows us to determine the pairwise (dis)similarity of multivariate time series that contain segments of similar trajectory at arbitrary positions. Furthermore, we have demonstrated that given a set of arbitrary time series, our proposed recurrence plot-based distance measure is able to identify representatives which best comprehend the recurring temporal patterns contained in the data, which is regarded as a common time series mining task with high practical relevance.

8.1.5 Invariance

In the majority of real-life time series applications we have to deal with sensor data that exhibits different kind of distortions as well as measuring errors or inaccuracies. The choice of distance measure for solving a particular time series mining tasks usually depends on the invariance required by the domain. Recent work has introduced techniques designed to efficiently measure similarity between time series with invariance to the distortions of warping, uniform scaling, offset, amplitude scaling, phase, occlusions, uncertainty and wandering baseline. In this work we proposed a novel order-invariant distance measure which is able to determine the (dis)similarity of time series that exhibit similar subsequences at arbitrary positions (refer to Chapter 5). Order invariance is an important consideration for many real-life data mining applications, where sensors record contextual patterns in their naturally occurring order and the resulting time series are compared according to co-occurring contextual patterns regardless of their order or location (see Chapter 7.2). To the best of our knowledge, order invariance has so far been missed by the community, and relevant literature is lacking of an order-invariant time series distance measure.

8.1.6 Signal Separation

In other time series applications we are confronted with the problem that several individual sources have been mixed together into a combined signal and the objective is to recover the original components or sources. A typical example of source separation is energy disaggregation, where an aggregated energy signal coming from a smart meter is disaggregated to deduce what appliances are used in a household and/or how much energy the individual appliances consume. In this work we evaluated various machine learning models with regard to their performance in classifying appliance states and inferring household occupancy (see Chapter 6). Our goal was to reduce the residential energy consumption by automatically optimizing heating schedules base on the deduced physical presence and learned behavioral patterns (see Chapter 7.1). Although the investigated models were merely used to solve the energy disaggregation problem, they can also be employed for to separate the sources of mixed signals found in other domains. Our literature survey and empirical evaluation of different machine learning techniques, which fit the energy disaggregation task, contributes to the decision making process in similar source separation problems.

8.1.7 Applications

All of the above stated contributions are of rather theoretical nature, but we also demonstrated the practical relevance of our newly developed time series distance measures. One main application area of our work is the optimization of vehicle engines with regard to exhaust emission, which is an important issue for car manufacturers. On behalf of the Volkswagen AG we developed several different time series distance measures, which were used to identify characteristic operational profiles from large amounts of sensor data recorded during test drives. Our factorization-based distance measure introduced in Chapter 3 was designed to discover complex drive maneuvers in vehicular sensor data, or more generally speaking, to identify internally homogeneous segments in multivariate time series. Furthermore, our recurrence plot-based distance measure introduced in Chapter 5 was devised to identify time series representatives that best comprehend the recurring temporal patterns contained in a corresponding dataset, or strictly speaking, to determine operational profiles that comprise frequently recurring driving behavior patterns. In addition, we have developed a software tool (see Chapter 7.2) which enables our clients to use the proposed recurrence plot-based time series distance measure in an easy and straightforward manner by providing an user interface for loading data, setting parameters, and analyzing results.

Another important application that motivated our scientific endeavor is heating control, since heating accounts for the biggest amount of total residential energy consumption. Therefore, we have introduced a new heating strategy, where the physical presence or activity of residents is deduced from certain appliance usage patterns and the heating is consequently turned off when the residents are definitely sleeping or away from home. In Chapter 6 we investigated various energy disaggregation techniques to infer appliance/occupancy states from aggregated energy signals measured by smart meters, which are installed in an increasing number of households and obviate additional sensor infrastructure. Furthermore, we have developed a prototypical heating control system (refer to Chapter 7.1) that recommends optimized heating schedules based on automatically determined occupancy states, which were inferred by means of energy disaggregation techniques.

8.1.8 Summary

Summing up, our introduced distance measure and developed time series mining tools have contributed to important environmental issues, such as emission reduction and energy efficiency, but also find application in various other domains. In addition, our work explored and created new avenues for

time series analysis in issues as diverse as pattern recognition, sensor fusion, fast classification, nonlinear modeling, source separation, and invariance.

8.2 Perspectives

In this section we illuminate current trends, new frontiers, and resulting perspectives in time series mining with regard to our own research.

8.2.1 Online Processing

This thesis considers time series analysis for different scenarios where the data is processed offline, meaning the whole input data is given from the beginning and an output is computed based on the problem data at hand. However, an increasing number of context-aware applications make decisions based on higher-level information that is derived from sensors data in real-time. Therefore, recent work has introduced a wide variety of online algorithms for real-time analysis of sensor or time series streams [25, 64, 70, 80, 86, 116, 118, 143, 157, 174, 199, 207]. Online algorithms generally process the input data piece-by-piece in a serial fashion, without having the entire input available from the start. In the context of time series streams a serialization of the input data is usually achieved by the sliding window technique, where the current window contains a certain amount of the most recent sensor measurements.

The online analysis of data streams brings about certain challenges, including the learning of new (or drifting) concepts [64], the development of algorithms that operate in time sensitive and computationally constrained environments [174], the handling of high-dimensional time series coming from multiple sensors [70], the implementation of efficient models which can be updated in constant time and space in the face of very high data arrival rates [80, 86], the introduction of new representations that allow dimensionality/numerosity reduction and lower bounding [116, 207], the reformulation of traditional time series mining problems for streaming data [118], the introduction of data-editing techniques for very fast streams and resource limited sensors [143], the development of models that ignore time intervals that do not represent a discrete underlying behavior [157], and the formulation of anytime algorithms that achieve high accuracy even if interrupted after seeing only a small fraction of the data [199].

In future work we plan to implement our proposed lucky time warping distance (see Chapter 4) in an online algorithm. We believe that the linear time and space complexity of our introduced lucky time warping distance

makes it an excellent choice for the real-time processing of time series streams with local scaling. Furthermore, we aim to reduce the computational cost of our proposed recurrence plot-based distance measure (see Chapter 5) in order to allow fast (dis)similarity comparisons for the online processing of sensors streams which are affected by unknown variables that cause nonlinear effects. In particular, we intend to eliminate the quadratic complexity of recurrence plot calculations by directly approximating recurrence quantification analysis measures without computing the actual recurrence matrix.

8.2.2 Semi-supervised Learning

The time series mining tasks examined in this work can be categorized into supervised and unsupervised learning problems or classification and clustering respectively. Contrary to supervised and unsupervised learning, which either requires labeled data that is difficult to obtain or utilizes unlabeled examples that lead to an ill-defined problem, semi-supervised learning algorithms are useful in application domains in which labeled data is limited and unlabeled data is plentiful. Hence, recent work on time series mining has introduced various semi-supervised learning techniques, including algorithms which use a surprisingly small set of human annotated examples to perform accurate classification [11], gradually learn new positive examples from the unlabeled data by using only as few as one labeled example [24], build accurate time series classifiers when only a handful of labeled examples is available [127, 206], exploit the commonality among labeled examples to allow monitoring at high bandwidths [207].

Since human experts may only review or label a small portion of the increasing amount of temporal data collected by widespread sensors, we plan to investigate semi-supervised learning algorithms in our future work. We intend to adopt our proposed lucky time warping and recurrence plot-based distance (see Chapter 4 and 5) to online algorithms, where a small set of labeled examples are sufficient to perform accurate classification of contextual patterns and new positive examples are learned from a continuous data streams to gradually build more robust models.

8.2.3 Distance and Feature Learning

All of the time series distance measures proposed in this work were designed for specific domains. However, the main problem with specifically designed distances is that they might not perform as well in other domains, since other datasets may require the selection/extraction of different features that are necessary to distinguish the individual classes. Recent work has introduced

techniques that could possibly be used to learn a distance metric for the input space from a given collection of (dis)similar points that preserves the distance relations [213], although the distance metric learning problem has not been well addressed so far. Other work has proposed deep learning algorithms to automatically learn feature representations (often from unlabeled data) thus avoiding a lot of time-consuming engineering [107]. But deep learning methods are often looked at as a black box, with most confirmations done empirically, rather than theoretically. Nonetheless, we believe that distance and feature learning will eventually eliminate the need for hand-designing time series features.

In future work we plan to investigate machine learning techniques, such as distance learning and deep learning, to automatically learn time series distance measure without selecting features manually. One promising idea might be to learn weights for Euclidean distance in order to minimize the error in time series classification. In case of time series with local scaling, one could possibly learn a warping path that maximizes the overall classification accuracy, as briefly mentioned in Chapter 4.

8.3 Closing Remarks

Our work on distance measures contributes to the time series community in that we presented new strategies for pattern recognition, proposed advanced sensor fusion methods for multivariate measurements, developed approximate algorithms for fast classification, built advanced models for nonlinear analysis, introduced novel analysis techniques for temporal data with order invariance, evaluated various machine learning models on their performance to separate mixed signals, and discussed real-life time series mining problems.

In addition to our own contribution we emphasized current trends and new perspectives in time series mining. For instance, we pointed out that a multitude of online algorithms has emerged, because an increasing number of time series mining tasks involve the processing of continuous sensor streams in real-time. Furthermore, we asserted that semi-supervised learning gains more importance, since human experts may only review or label a very small portion of the growing amount of temporal data collected by widespread sensors. Moreover, we underlined the great potential of distance learning and deep learning, which might be employed to automatically learn time series distance measures in near future.

All in all, we strongly believe in the future of time series mining, as temporal data occurs in an ever increasing number of practical applications and many scientific challenges still remain unsolved.

Bibliography

[1] J. Abonyi, B. Feil, S. Nemeth, and P. Arva. Modified gath–geva clustering for fuzzy segmentation of multivariate time-series. *Fuzzy Sets Syst.*, 149:39–56, January 2005.

[2] J. Abonyi, B. Feil, S. Nemeth, and P. Avra. Principal component analysis based time series segmentation: A new sensor fusion algorithm. *Preprint*, 2004.

[3] E. Acar, D. M. Dunlavy, and T. G. Kolda. Link prediction on evolving data using matrix and tensor factorizations. In *ICDMW '09: Proceedings of the 2009 IEEE International Conference on Data Mining Workshops*, pages 262–269, Washington, DC, USA, 2009. IEEE Computer Society.

[4] E. Acar, S. Spiegel, and S. Albayrak. Mediaeval 2011 affect task: Violent scene detection combining audio and visual features with svm. In *MediaEval*, volume 807 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2011.

[5] E. Alpaydin. *Introduction to Machine Learning*. The MIT Press, 2nd edition, 2010.

[6] K. C. Armel, A. Gupta, G. Shrimali, and A. Albert. Is disaggregation the holy grail of energy efficiency? the case of electricity. *Energy Policy*, 52(C):213–234, 2013.

[7] L. Ballan, M. Bertini, A. Bimbo, L. Seidenari, and G. Serra. Event detection and recognition for semantic annotation of video. *Multimedia Tools Appl.*, 51:279–302, January 2011.

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

[9] G. E. A. P. A. Batista, X. Wang, and E. J. Keogh. A complexity-invariant distance measure for time series. In *SDM*, pages 699–710, 2011.

[10] C. Beckel, W. Kleiminger, T. Staake, and S. Santini. Improving device-level electricity consumption breakdowns in private households using on/off events. *SIGBED Rev.*, 9(3):32–38, 2012.

[11] N. Begum, B. Hu, T. Rakthanmanon, and E. J. Keogh. A minimum description length technique for semi-supervised time series classification. In *IRI (best papers)*, pages 171–192, 2013.

[12] M. Beigl, A. Krohn, T. Zimmer, and C. Decker. Typical sensors needed in ubiquitous and pervasive computing. In *In Proceedings of the First International Workshop on Networked Sensing Systems (INSS '04*, pages 153–158, 2004.

[13] D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *KDD*, pages 359–370, 1994.

[14] D. J. Berndt and J. Clifford. Finding patterns in time series: a dynamic programming approach. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in knowledge discovery and data mining*, pages 229–248. American Association for Artificial Intelligence, Menlo Park, CA, USA, 1996.

[15] P. Boait and R. Rylatt. A method for fully automatic operation of domestic heating. *Energy and Buildings*, 42(1):11–16, 2010.

[16] G. E. Box, G. M. Jenkins, and G. C. Reinsel. *Time Series Analysis: Forecasting and Control, 4th Edition*. Wiley, 2008.

[17] W. Brendel and S. Todorovic. Activities as time series of human postures. In *ECCV (2)*, pages 721–734, 2010.

[18] D. Brillinger, E. A. Robinson, and F. P. Schoenberg, editors. *Time Series Analysis and Applications to Geophysical Systems*. Springer, 2004.

[19] M. Brown and L. Rabiner. An adaptive, ordered, graph search technique for dynamic time warping for isolated word recognition. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 30(4):535 – 544, aug 1982.

[20] K. Buza, A. Nanopoulos, and L. Schmidt-Thieme. Insight: Efficient and effective instance selection for time-series classification. In *PAKDD*, pages 149–160, 2011.

[21] B. J. L. Campana and E. J. Keogh. A compression-based distance measure for texture. *Statistical Analysis and Data Mining*, 3(6):381–398, 2010.

[22] G. Chen and D. Kotz. A survey of context-aware mobile computing research. Technical report, Dartmouth College, Hanover, NH, USA, 2000.

[23] L. Chen and R. Ng. On the marriage of lp-norms and edit distance. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, VLDB '04, pages 792–803. VLDB Endowment, 2004.

[24] Y. Chen, B. Hu, E. J. Keogh, and G. E. A. P. A. Batista. Dtw-d: time series semi-supervised learning from a single example. In *KDD*, pages 383–391, 2013.

[25] Y. Chen, M. A. Nascimento, B. C. Ooi, and A. K. H. Tung. Spade: On shape-based pattern detection in streaming time series. In *ICDE*, pages 786–795, 2007.

[26] Y. Chen, A. Why, G. E. A. P. A. Batista, A. Mafra-Neto, and E. J. Keogh. Flying insect classification with inexpensive sensors. *CoRR*, abs/1403.2654, 2014.

[27] B. Chiu, E. J. Keogh, and S. Lonardi. Probabilistic discovery of time series motifs. In *KDD*, pages 493–498, 2003.

[28] J. Choi, B. Bae, and S. Kim. Divergence in perpendicular recurrence plot; quantification of dynamical divergence from short chaotic time series. *Physics Letters A*, 263(4-6):299–306, 1999.

[29] S. Chu, E. J. Keogh, D. Hart, and M. J. Pazzani. Iterative deepening dynamic time warping for time series. In *SDM*, 2002.

[30] M. Chuah and F. Fu. Ecg anomaly detection via time series analysis. In P. Thulasiraman, X. He, T. Xu, M. Denko, R. Thulasiram, and L. Yang, editors, *Frontiers of High Performance Computing and Networking ISPA 2007 Workshops*, volume 4743 of *Lecture Notes in Computer Science*, pages 123–135. Springer Berlin Heidelberg, 2007.

[31] J. G. Cleary and L. E. Trigg. K*: An instance-based learner using and entropic distance measure. In *International Conference on Machine Learning*, pages 108–114, 1995.

[32] M. Corduas. Mining time series data: A selective survey. In F. Palumbo, C. N. Lauro, and M. J. Greenacre, editors, *Data Analysis and Classification*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 355–362. Springer Berlin Heidelberg, 2010.

[33] J. L. Crowley and Y. Demazeau. Principles and techniques for sensor data fusion. *Signal Processing*, 32(12):5 – 27, 1993. Intelligent Systems for Signal and Image Understanding.

[34] J. D. Cryer and K.-S. Chan. *Time Series Analysis with Applications in R*, volume 14 of *Springer Texts in Statistics*. Springer, 2008.

[35] P. Davies. *About Time: Einstein's Unfinished Revolution*. Penguin Science, UK, 1996.

[36] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. J. Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *PVLDB*, 1(2):1542–1552, 2008.

[37] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[38] S. T. Dumais, G. W. Furnas, T. K. Landauer, S. Deerwester, and R. Harshman. Using latent semantic analysis to improve access to textual information. In *CHI '88: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 281–285, New York, NY, USA, 1988. ACM.

[39] J. C. Dunn. Well separated clusters and optimal fuzzy-partitions. *Journal of Cybernetics*, 4:95–104, 1974.

[40] J.-P. Eckmann, S. O. Kamphorst, and D. Ruelle. Recurrence plots of dynamical systems. *Europhysics Letters*, 4(9), 1987.

[41] W. Elmenreich. *Sensor Fusion in Time-Triggered Systems*. PhD thesis, Technische Universität Wien, Institut für Technische Informatik, 2002.

[42] P. Esling and C. Agon. Time-series data mining. *ACM Comput. Surv.*, 45(1):12:1–12:34, Dec. 2012.

[43] V. Estivill-Castro. Why so many clustering algorithms: A position paper. *SIGKDD Explor. Newsl.*, 4(1):65–75, June 2002.

[44] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. *SIGMOD Rec.*, 23(2):419–429, May 1994.

[45] J. Fan and Q. Yao. Characteristics of time series. In *Nonlinear Time Series: Nonparametric and Parametric Methods*, Springer Series in Statistics, pages 29–88. Springer New York, 2003.

[46] I. Färber, S. Günnemann, H. Kriegel, P. Kröger, E. Müller, E. Schubert, T. Seidl, and A. Zimek. On using class-labels in evaluation of clusterings. In *MultiClust: 1st International Workshop on Discovering, Summarizing and Using Multiple Clusterings Held in Conjunction with KDD 2010, Washington, DC*, 2010.

[47] J. Fischer and D. Whitney. Serial dependence in visual perception. *Nature Neuroscience*, 17:738–743, 2014.

[48] J. Froehlich, E. Larson, S. Gupta, G. Cohn, M. Reynolds, and S. Patel. Disaggregated end-use energy sensing for the smart grid. *IEEE Pervasive Computing*, 10(1):28–39, 2011.

[49] T.-C. Fu. A review on time series data mining. *Eng. Appl. Artif. Intell.*, 24(1):164–181, Feb. 2011.

[50] C. Furia, D. Mandrioli, A. Morzenti, and M. Rossi. *Modeling Time in Computing*, volume 16 of *Monographs in Theoretical Computer Science*. Springer, 2012.

[51] J. Gaebler, S. Spiegel, and S. Albayrak. Matarcs: An exploratory data analysis of recurring patterns in multivariate time series. In *Proceedings of Workshop on New Frontiers in Mining Complex Patterns (NFMCP)*, ECML-PKDD'12. Springer, 2012.

[52] A. R. Ganguly, J. Gama, O. A. Omitaomu, M. M. Gaber, and R. R. Vatsavai. *Knowledge Discovery from Sensor Data*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 2008.

[53] H. W. Gellersen, A. Schmidt, and M. Beigl. Multi-sensor context-awareness in mobile devices and smart artifacts. *Mob. Netw. Appl.*, 7:341–351, Oct 2002.

[54] Z. Ghahramani and M. I. Jordan. Factorial hidden markov models. *Machine Learning*, 29(2-3):245–273, 1997.

[55] T. Giannakopoulos, A. Makris, D. I. Kosmopoulos, S. J. Perantonis, and S. Theodoridis. Audio-visual fusion for detecting violent scenes in videos. In *Hellenic Conference on Artificial Intelligence*, pages 91–100, 2010.

[56] Y. Gong, W. Wang, S. Jiang, Q. Huang, and W. Gao. Detecting violent scenes in movies by auditory and visual cues. In *Proceedings of the 9th Pacific Rim Conference on Multimedia: Advances in Multimedia Information Processing*, PCM '08, pages 317–326, Berlin, Heidelberg, 2008. Springer-Verlag.

[57] J. G. D. Gooijer and R. J. Hyndman. 25 years of time series forecasting. *International Journal of Forecasting*, 2006.

[58] S. Grondin. Timing and time perception: A review of recent behavioral and neuroscience findings and theoretical directions. *Attention, Perception, and Psychophysics*, 72(3):561–582, 2010.

[59] R. Gupta, A. Mittal, and K. Singh. A time-series-based feature extraction approach for prediction of protein structural class. *EURASIP J. Bioinformatics Syst. Biol.*, 2008:2:1–2:7, January 2008.

[60] S. Gupta, M. S. Reynolds, and S. N. Patel. Electrisense: single-point sensing using emi for electrical event detection and classification in the home. In *Proceedings of the 12th ACM international conference on Ubiquitous computing*, pages 139–148, 2010.

[61] J. D. Hamilton. *Time Series Analysis*. Princeton Univ. Press, 1994.

[62] E. Hamori and J. Ruskin. H curves, a novel method of representation of nucleotide series especially suited for long dna sequences. *The Journal of Biological Chemistry*, 258(2):1318–1327, 1983.

[63] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 3rd edition, 2011.

[64] Y. Hao, Y. Chen, J. Zakaria, B. Hu, T. Rakthanmanon, and E. J. Keogh. Towards never-ending learning from time series streams. In *KDD*, pages 874–882, 2013.

[65] Y. Hao, M. Shokoohi-Yekta, G. Papageorgiou, and E. J. Keogh. Parameter-free audio motif discovery in large data archives. In *ICDM*, pages 261–270, 2013.

[66] J. Himberg, K. Korpiaho, H. Mannila, J. Tikanmaeki, and H. Toivonen. Time series segmentation for context recognition in mobile devices. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, ICDM '01, pages 203–210, Washington, DC, USA, 2001. IEEE Computer Society.

[67] F. Hoeppner. Time series abstraction methods - a survey. In *Informatik bewegt: Informatik 2002 - 32. Jahrestagung der Gesellschaft fuer Informatik E.V. (GI)*, pages 777–786. GI, 2002.

[68] W. Hopwood and P. Newbold. Time series analysis in accouting: A survey and analysis of recent issues. *Journal of Time Series Analysis*, 1(2):135–144, 2008.

[69] B. Hu, Y. Chen, and E. J. Keogh. Time series classification under more realistic assumptions. In *SDM*, 2013.

[70] B. Hu, Y. Chen, J. Zakaria, L. Ulanova, and E. J. Keogh. Classification of multi-dimensional streaming time series by weighting each classifier's track record. In *ICDM*, pages 281–290, 2013.

[71] B. Hu, T. Rakthanmanon, B. J. L. Campana, A. Mueen, and E. J. Keogh. Image mining of historical manuscripts to establish provenance. In *SDM*, pages 804–815, 2012.

[72] D. L. H. Hugo Merchant and W. H. Meck. Neural basis of the perception and estimation of time. *Annual Review of Neuroscience*, 36, 2013.

[73] J. Indulska, D. J. Patterson, T. Rodden, and M. Ott, editors. *Pervasive Computing, 6th International Conference, Pervasive 2008, Sydney, Australia, May 19-22, 2008, Proceedings*, volume 5013 of *Lecture Notes in Computer Science*. Springer, 2008.

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

[75] W. Jiang and A. C. Loui. Audio-visual grouplet: temporal audio-visual interactions for general video concept classification. In *ACM Multimedia*, pages 123–132, 2011.

[76] W. Jiang and A. C. Loui. Grouplet-based distance metric learning for video concept detection. In *ICME*, pages 753–758, 2012.

[77] G. H. John and P. Langley. Estimating continuous distributions in bayesian classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 338–345, San Mateo, 1995. Morgan Kaufmann.

[78] E. R. Kanasewich. *Time Sequence Analysis in Geophysics*. The University of Alberta Press, third edition, 1981.

[79] A. Karahoca and M. Nurullahoglu. Human motion analysis and action recognition. In *Proceedings of the 1st WSEAS International Conference on Multivariate Analysis and its Application in Science and Engineering*, pages 156–161, Stevens Point, Wisconsin, USA, 2008. World Scientific and Engineering Academy and Society (WSEAS).

[80] S. Kasetty, C. Stafford, G. P. Walker, X. Wang, and E. J. Keogh. Real-time classification of streaming sensor data. In *ICTAI (1)*, pages 149–156, 2008.

[81] S. M. Kay. *Fundamentals of Statistical Signal Processing: Estimation Theory*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1998.

[82] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. Improvements to platt's smo algorithm for svm classifier design. *Neural Comput.*, 13:637–649, March 2001.

[83] E. J. Keogh. Efficiently finding arbitrarily scaled patterns in massive time series databases. In N. Lavrac, D. Gamberger, L. Todorovski, and H. Blockeel, editors, *Lecture Notes in Computer Science*, volume 2838, pages 253–265. Springer Berlin / Heidelberg, 2003.

[84] E. J. Keogh. Efficiently finding arbitrarily scaled patterns in massive time series databases. In *PKDD*, pages 253–265, 2003.

[85] E. J. Keogh, S. Chu, D. Hart, and M. Pazzani. Segmenting time series: A survey and novel approach. In M. Last, A. Kandel, and H. Bunke, editors, *Data mining in time series databases*, volume 57 of *Series in Machine Perception and Artificial Intelligence*, pages 1–22. World Scientific Publishing Company, 2004.

[86] E. J. Keogh, S. Chu, D. Hart, and M. J. Pazzani. An online algorithm for segmenting time series. In *Proceedings of the 2001 IEEE*

International Conference on Data Mining, ICDM '01, pages 289–296, Washington, DC, USA, 2001. IEEE.

[87] E. J. Keogh and S. Kasetty. On the need for time series data mining benchmarks: A survey and empirical demonstration. *Data Min. Knowl. Discov.*, 7(4):349–371, 2003.

[88] E. J. Keogh and J. Lin. Clustering of time-series subsequences is meaningless: implications for previous and future research. *Knowl. Inf. Syst.*, 8(2):154–177, 2005.

[89] E. J. Keogh, J. Lin, and A. W.-C. Fu. Hot sax: Efficiently finding the most unusual time series subsequence. In *ICDM*, pages 226–233, 2005.

[90] E. J. Keogh and M. J. Pazzani. Scaling up dynamic time warping for datamining applications. In *KDD*, pages 285–289, 2000.

[91] E. J. Keogh and T. Rakthanmanon. Fast shapelets: A scalable algorithm for discovering time series shapelets. In *SDM*, pages 668–676, 2013.

[92] E. J. Keogh and C. A. Ratanamahatana. Everything you know about dynamic time warping is wrong. In *KDD*, 2004.

[93] E. J. Keogh and C. A. Ratanamahatana. Exact indexing of dynamic time warping. *Knowl. Inf. Syst.*, 7(3):358–386, 2005.

[94] E. J. Keogh, L. Wei, X. Xi, M. Vlachos, S.-H. Lee, and P. Protopapas. Supporting exact indexing of arbitrarily rotated shapes and periodic time series under euclidean and warping distance measures. *VLDB J.*, 18(3):611–630, 2009.

[95] E. J. Keogh, Q. Zhu, B. Hu, Y. Hao, X. Xi, L. Wei, and C. A. Ratanamahatana. The UCR time series classification/clustering homepage. `www.cs.ucr.edu/~eamonn/time_series_data/`, 2011.

[96] B. Kerr. Thread arcs: an email thread visualization. In *Proceedings of the Ninth annual IEEE conference on Information visualization*, INFOVIS'03, pages 211–218, Washington, DC, USA, 2003. IEEE Computer Society.

[97] H. Kim, M. Marwah, M. F. Arlitt, G. Lyon, and J. Han. Unsupervised disaggregation of low frequency power measurements. In *SDM*, pages 747–758, 2011.

[98] S.-W. Kim, S. Park, and W. W. Chu. Efficient processing of similarity search under time warping in sequence databases: an index-based approach. *Inf. Syst.*, 29(5):405–420, 2004.

[99] W. Kleiminger, C. Beckel, and S. Santini. Opportunistic sensing for efficient energy usage in private households. In *Proceedings of the Smart Energy Strategies Conference 2011*, 2011.

[100] J. Z. Kolter and M. J. Johnson. Redd: A public data set for energy disaggregation research. In *Proc. of SustKDD Workshop on Data Mining Applications in Sustainability*, 2011.

[101] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434, New York, NY, USA, 2008. ACM.

[102] P. E. Kourouthanassis and G. M. Giaglis, editors. *Pervasive Information Systems*, volume 10 of *Advances in Management Information Systems*. M.E. Sharp, 2008.

[103] K. Kroschel, G. Rigoll, and B. Schuller, editors. *Statistische Informationstechnik: Signal-und Mustererkennung, Parameter- und Signalschtzung*, volume 5. Springer Berlin / Heidelberg, 2011.

[104] M. Kumar, N. R. Patel, and J. Woo. Clustering seasonality patterns in the presence of errors. In *KDD*, 2002.

[105] O.-W. Kwon and T.-W. Lee. Phoneme recognition using ica-based feature extraction and transformation. *Signal Process.*, 84, June 2004.

[106] C. Lainscsek, M. Hernandez, J. Weyhenmeyer, T. Sejnowski, and H. Poizner. Non-linear dynamical analysis of eeg time series distinguishes patients with parkinson's disease from healthy individuals. *Front. Neurol.*, 4:200, 2013.

[107] Q. V. Le, M. Ranzato, R. Monga, M. Devin, G. Corrado, K. Chen, J. Dean, and A. Y. Ng. Building high-level features using large scale unsupervised learning. In *ICML*, 2012.

[108] R. Le Poidevin. The experience and perception of time. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Stanford Press, fall 2011 edition, 2011.

[109] G. Lehmann, A. Rieger, M. Blumendorf, and S. Albayrak. A 3-layer architecture for smart environment models. In *Proceedings of PERCOM : the 8th annual IEEE international conference on Pervasive Computing and Communications*, pages 636–641. IEEE Computer Society, IEEE Computer Society Publications Office, 2010.

[110] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. Statistical properties of community structure in large social and information networks. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 695–704, New York, NY, USA, 2008. ACM.

[111] J. Liang, S. Ng, G. Kendall, and J. Cheng. Load signature study - part i: Basic concept, structure, and methodology. *IEEE Trans. on Power Delivery*, 25(2):551–560, 2010.

[112] T. W. Liao. Clustering of time series data - a survey. *Journal on Pattern Recognition*, 38(11):1857–1874, 2005.

[113] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management*, pages 556–559, New York, NY, USA, 2003. ACM.

[114] G.-Y. Lin, S.-C. Lee, and J. Y.-J. Hsu. Sensing from the panel: Applying the power meters for appliance recognition. In *Proceedings of the 14th Conference on Artificial Intelligence and Applications*, 2009.

[115] J. Lin, E. J. Keogh, and S. Lonardi. Visualizing and discovering non-trivial patterns in large time series databases. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume 4, page 61. SAGE Publications, 2004.

[116] J. Lin, E. J. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *SIGMOD*, pages 2–11, 2003.

[117] J. Lin, E. J. Keogh, S. Lonardi, and P. Patel. Finding motifs in time series. In *KDD*, 2002.

[118] J. Lin, E. J. Keogh, and W. Truppel. Clustering of streaming time series is meaningless. In *DMKD*, pages 56–65, 2003.

[119] J. Lin, E. J. Keogh, L. Wei, and S. Lonardi. Experiencing sax: A novel symbolic representation of time series. *Data Min. Knowl. Discov.*, 15(2):107–144, Oct. 2007.

[120] J. Lines, A. Bagnall, P. Caiger-Smith, and S. Anderson. Classification of household devices by electricity usage profiles. In *IDEAL*, pages 403–412, 2011.

[121] H. Liu and M. Hiroshi. *Feature Selection for Knowledge Discovery and Data Mining*, volume 454 of *Springer International Series in Engineering and Computer Science*. Springer, Berlin / Heidelberg, 1998.

[122] B. Logan. Mel frequency cepstral coefficients for music modeling. In *Int. Symposium on Music Information Retrieval*, 2000.

[123] J. Lu, T. Sookoor, V. Srinivasan, G. Gao, B. Holben, J. Stankovic, E. Field, and K. Whitehouse. The smart thermostat: using occupancy sensors to save energy in homes. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, pages 211–224, 2010.

[124] M. Luetzenberger, T. Kuester, T. Konnerth, A. Thiele, N. Masuch, A. Hessler, J. Keiser, M. Burkhardt, S. Kaiser, and S. Albayrak. JIAC V — a MAS framework for industrial applications (extended abstract). In T. Ito, C. Jonker, M. Gini, and O. Shehory, editors, *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems*, pages 1189–1190, 2013.

[125] S. MacLean and G. Labahn. Elastic matching in linear time and constant space. In *DAS*, 2010.

[126] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[127] K. Marussy and K. Buza. Success: A new approach for semi-supervised classification of time-series. In *ICAISC (1)*, pages 437–447, 2013.

[128] N. Marwan. *Encounters with Neighbours: Current Developments of Concepts based on Recurrence Plots and their Applications*. PhD thesis, University of Potsdam, 2003.

[129] N. Marwan. A historical review of recurrence plots. *European Physical Journal Special Topics*, 164(1):3–12, 2008.

[130] N. Marwan. How to avoid potential pitfalls in recurrence plot based data analysis. *I. J. on Bifurcation and Chaos*, 21(4):1003–1017, 2011.

[131] N. Marwan, M. Romano, and M. Thiel. Recurrence plots and cross recurrence plots. `www.recurrence-plot.tk`.

[132] N. Marwan, M. Romano, M. Thiel, and J. Kurths. Recurrence plots for the analysis of complex systems. *Physics Reports*, 438(5-6):237–329, 2007.

[133] N. Marwan, S. Schinkel, and J. Kurths. Recurrence plots 25 years later - gaining confidence in dynamical transitions. *Europhysics Letters*, 101(2), 2013.

[134] N. Marwan, M. Thiel, and N. Nowaczyk. Cross recurrence plot based synchronization of time series. *Nonlinear Processes in Geophysics*, 9(3/4):325–331, 2002.

[135] N. Marwan, N. Wessel, U. Meyerfeldt, A. Schirdewan, and J. Kurths. Recurrence-plot-based measures of complexity and their application to heart-rate-variability data. *Phys. Rev. E*, 66:026702, Aug 2002.

[136] *MathWorks (TM), (R2013a). Statistics Toolbox: Supervised Learning Workflow and Algorithms.*

[137] U. Maulik and S. Bandyopadhyay. Performance evaluation of some clustering algorithms and validity indices. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(12):1650–1654, 2002.

[138] W. Meesrikamolkul, V. Niennattrakul, and C. A. Ratanamahatana. Shape-based clustering for time series data. In *PAKDD*, pages 530–541, 2012.

[139] E. Miluzzo, N. D. Lane, K. Fodor, R. Peterson, H. Lu, M. Musolesi, S. B. Eisenman, X. Zheng, and A. T. Campbell. Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, SenSys '08, pages 337–350, New York, NY, USA, 2008. ACM.

[140] C. S. Moeller-Levet, F. Klawonn, K.-H. Cho, and O. Wolkenhauer. Fuzzy clustering of short time-series and unevenly distributed sampling points. In *LNCS, Proceedings of the IDA2003*, pages 28–30, 2003.

[141] C. Montemayor. *Minding Time: A Philosophical and Theoretical Approach to the Psychology of Time*. Supplements to the Study of Time. Brill, 2012.

[142] A. Mueen, E. J. Keogh, and N. B. Shamlo. Finding time series motifs in disk-resident data. In *ICDM*, pages 367–376, 2009.

[143] V. Niennattrakul, E. J. Keogh, and C. A. Ratanamahatana. Data editing techniques to allow the application of distance-based outlier detection to streams. In *ICDM*, pages 947–952, 2010.

[144] M. S. Obaidat, M. Denko, and I. Woungang, editors. *Pervasive Information Systems*. John Wiley and Sons, 2011.

[145] M. T. S. of New York State. *Theory and Practice: Journal of the Music Theory Society of New York State*, volume 13. The Society, 1988.

[146] R. T. Olszewski. *Generalized Feature Extraction for Structural Pattern Recognition in Time-Series Data Thesis Committee*. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, 2001.

[147] S. J. Orfanidis. Savitzky-golay filtering. In *Introduction to Signal Processing*, Englewood Cliffs, NJ, 1996. Prentice-Hall.

[148] V. Pallotta, P. Bruegger, and B. Hirsbrunner. Smart heating systems: Optimizing heating systems by kinetic-awareness. In *Proc. of 3rd International Conference on Digital Information Management*, 2008.

[149] E. Pekalska and R. P. W. Duin. *The Dissimilarity Representation for Pattern Recognition: Foundations And Applications (Machine Perception and Artificial Intelligence)*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2005.

[150] E. Pkalska, R. P. W. Duin, and P. Paclík. Prototype selection for dissimilarity-based classifiers. *Pattern Recogn.*, 39(2):189–208, Feb. 2006.

[151] J. Platt. *Machines using Sequential Minimal Optimization*. MIT Press, Cambridge, MA, USA, 1999.

[152] H. V. Poor. *An Introduction to Signal Detection and Estimation*. Springer Texts in Electrical Engineering. Springer, 2nd edition, 1994.

[153] J. R. Quinlan. *C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning)*. Morgan Kaufmann, 1992.

[154] T. Rakthanmanon, B. J. L. Campana, A. Mueen, G. Batista, M. B. Westover, Q. Zhu, J. Zakaria, and E. J. Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *KDD*, pages 262–270, 2012.

[155] T. Rakthanmanon, B. J. L. Campana, A. Mueen, G. E. A. P. A. Batista, M. B. Westover, Q. Zhu, J. Zakaria, and E. J. Keogh. Addressing big data time series: Mining trillions of time series subsequences under dynamic time warping. *TKDD*, 7(3):10, 2013.

[156] T. Rakthanmanon and E. J. Keogh. Fast-shapelets: A scalable algorithm for discovering time series shapelets. In *SDM*, 2013.

[157] T. Rakthanmanon, E. J. Keogh, S. Lonardi, and S. Evans. Time series epenthesis: Clustering time series streams requires ignoring some data. In *Proceedings of the 2011 IEEE 11th International Conference on Data Mining*, ICDM '11, pages 547–556. IEEE Computer Society, 2011.

[158] T. S. Rao, M. Priestly, and O. Lessi. *Applications of Time Series Analysis in Astronomy and Meteorology*. Chapman and Hall, 2013.

[159] C. A. Ratanamahatana and E. J. Keogh. Making time-series classification more accurate using learned constraints. In *SDM*, 2004.

[160] C. A. Ratanamahatana and E. J. Keogh. Three myths about dynamic time warping data mining. In *SDM*, 2005.

[161] C. A. Ratanamahatana, J. Lin, D. Gunopulos, E. J. Keogh, M. Vlachos, and G. Das. Mining time series data. In O. Maimon and L. Rokach, editors, *Data Mining and Knowledge Discovery Handbook*, pages 1049–1077. Springer US, 2010.

[162] A. Reinhardt, P. Baumann, D. Burgstahler, M. Hollick, H. Chonov, M. Werner, and R. Steinmetz. On the accuracy of appliance identification based on distributed load metering data. In *Proc. of 2nd IFIP Conference on Sustainable Internet and ICT for Sustainability*, 2012.

[163] D. Roscher, G. Lehmann, V. Schwartze, M. Blumendorf, and S. Albayrak. Dynamic distribution and layouting of model-based user interfaces in smart environments. In H. Hussmann, G. Meixner, and D. Zuehlke, editors, *Model-Driven Development of Advanced User Interfaces*, volume 340 of *Studies in Computational Intelligence*, pages 171–197. Springer Berlin / Heidelberg, 2011.

[164] A. G. Ruzzelli, C. Nicolas, A. Schoofs, and G. M. P. O'Hare. Real-time recognition and profiling of appliances through a single electricity sensor. In *Proceedings of the 7th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, pages 279–287, 2010.

[165] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *Transactions on Acoustics, Speech and Signal Processing*, 26(1), 1978.

[166] Y. Sakurai, M. Yoshikawa, and C. Faloutsos. Ftw: fast similarity search under the time warping distance. In *PODS*, pages 326–337, 2005.

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

[168] N. I. Sapankevych and R. Sankar. Time series prediction using support vector machines: A survey. *Comp. Intell. Mag.*, 4(2):24–38, May 2009.

[169] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl. Application of dimensionality reduction in recommender system - a case study. In *In ACM WebKDD Workshop*, 2000.

[170] A. P. Schultz, Y. Zou, N. Marwan, and M. T. Turvey. Local minima-based recurrence plots for continuous dynamical systems. *I. J. Bifurcation and Chaos*, 21(4):1065–1075, 2011.

[171] D. Schultz, S. Spiegel, N. Marwan, and S. Albayrak. Approximation of diagonal line based measures in recurrence quantification analysis. In *Physics Letters A*. Elsevier, 2015.

[172] V. Schwartze, S. Feuerstack, and S. Albayrak. Behavior-sensitive user interfaces for smart environments. In *ICDHM '09: Proceedings of the 2nd International Conference on Digital Human Modeling*, pages 305–314, Berlin, Heidelberg, 2009. Springer-Verlag.

[173] G. Shani, C. Meek, and A. Gunawardana. Hierarchical probabilistic segmentation of discrete events. In *ICDM'09*, pages 974–979, Washington, DC, USA, 2009.

[174] J. Shieh and E. J. Keogh. Polishing the right apple: Anytime classification also benefits data streams with constant arrival times. In *ICDM*, pages 461–470, 2010.

[175] A. Singhal and D. E. Seborg. Matching patterns from historical data using pca and distance similarity factors. In *Proceedings of the American Control Conference*, Arlington, VA, 2001. IEEE.

[176] A. A. Smith. *Classification and Alignment of Gene-Exression Time-Series Data*. PhD thesis, University of Wisconsin-Madison, 2009.

[177] S. Spiegel. Discovery of driving behavior patterns. In *Smart Information Services - Computational Intelligence for Real-Life Applications*, Advances in Computer Vision and Pattern Recognition, pages 315–343. Springer International Publishing Switzerland, March 2015.

[178] S. Spiegel. Optimization of in-house energy demand. In *Smart Information Services - Computational Intelligence for Real-Life Applications*, Advances in Computer Vision and Pattern Recognition, pages 271–289. Springer International Publishing Switzerland, March 2015.

[179] S. Spiegel and S. Albayrak. An order-invariant time series distance measure - position on recent developments in time series analysis. In *Proceedings of 4th International Conference on Knowledge Discovery and Information Retrieval (KDIR)*, pages 264–268. SciTePress, 2012.

[180] S. Spiegel and S. Albayrak. Energy disaggregation meets heating control. In *Proceedings of 29th Symposium on Applied Computing (SAC)*. ACM, 2014.

[181] S. Spiegel, J. Clausen, S. Albayrak, and J. Kunegis. Link prediction on evolving data using tensor factorization. In *Proceedings of the 15th International Conference on New Frontiers in Applied Data Mining*, PAKDD'11, pages 100–110, Berlin, Heidelberg, 2012. Springer-Verlag.

[182] S. Spiegel, J. Gaebler, A. Lommatzsch, E. D. Luca, and S. Albayrak. Pattern recognition and classification for multivariate time series. In *Proceedings of the 5th International Workshop on Knowledge Discovery from Sensor Data*, SensorKDD '11, pages 34–42, New York, NY, USA, 2011. ACM.

[183] S. Spiegel, B.-J. Jain, and S. Albayrak. Fast time series classification under lucky time warping distance. In *Proceedings of 29th Symposium on Applied Computing (SAC)*. ACM, 2014.

[184] S. Spiegel, B.-J. Jain, E. D. Luca, and S. Albayrak. Pattern recognition in multivariate time series - dissertation proposal. In *Proceedings of 4th Workshop for Ph.D. Students in Information and Knowledge Management (PIKM)*, CIKM'11. ACM, 2011.

[185] S. Spiegel, J.-B. Jain, and S. Albayrak. A recurrence plot-based distance measure. In *Translational Recurrences*, volume 103 of *Springer Proceedings in Mathematics and Statistics*, pages 1–15. Springer International Publishing, 2014.

[186] S. Spiegel, J. Kunegis, and F. Li. Hydra: A hybrid recommender system [cross-linked rating and content information]. In *Proceedings of the 1st ACM International Workshop on Complex Networks Meet Information and Knowledge Management*, CNIKM '09, pages 75–80, New York, NY, USA, 2009. ACM.

[187] S. Spiegel, D. Schultz, and S. Albayrak. Besttime: Finding representatives in time series datasets. In T. Calders, F. Esposito, E. Hllermeier, and R. Meo, editors, *Proceedings 7th European machine learning and data mining conference (ECML/PKDD), Machine Learning and Knowledge Discovery in Databases*, volume 8726 of *Lecture Notes in Computer Science (LNCS)*, pages 477–480. Springer Berlin Heidelberg, 2014.

[188] S. Spiegel, D. Schultz, M. Schacht, and S. Albayrak. Supplementary online material - besttime app, test data, video demonstration, technical report:. `www.dai-lab.de/~spiegel/besttime.html`, 2013.

[189] S. Spiegel, V. Schwartze, M. Schacht, S. Ahrndt, and S. Albayrak. Heating control via energy disaggregation: A practical demonstration. In *15th International Conference on Human-Computer Interaction (INTERACT)*, Springer (LNCS) Series. Springer, 2015. IN PREPARATION.

[190] J.-L. Starck and F. Murtagh. *Handbook of Astronomical Data Analysis*. Springer-Verlag, 2002.

[191] G. Stephanopoulos and C. Han. Intelligent systems in process engineering: A review. In *Computational Chemical Engineering*, volume 20, pages 743–791, Miamisburg, OH, USA, 1996. Elsevier.

[192] K. Suzuki, S. Inagaki, T. Suzuki, H. Nakamura, and K. Ito. Nonintrusive appliance load monitoring based on integer programming. In *Proc. on SICE Annual Conference*, 2008.

[193] F. Takens. Detecting strange attractors in turbulence. In D. A. Rand and L. S. Young, editors, *Dynamical systems and turbulence*, pages 366–381. Spinger-Verlag, 1981.

[194] J. Theiler. Spurious dimension from correlation algorithms applied to limited time-series data. *Phys. Rev. A*, 34:2427–2432, Sep 1986.

[195] A. Theissler. *Detecting anomalies in multivariate time series from automotive systems*. PhD thesis, Brunel University London, 2013.

[196] H. L. V. Trees. *Detection, Estimation, and Modulation Theory: Part I, II and III*. Wiley, 2001.

[197] R. S. Tsay, editor. *Analysis of Financial Time Series*. Probability and Statistics. Wiley, third edition, 2010.

[198] J. W. Tukey. *Exploratory Data Analysis*. Behavioral Science. Addison-Wesley, first edition, 1977.

[199] K. Ueno, X. Xi, E. J. Keogh, and D.-J. Lee. Anytime classification using the nearest neighbor algorithm with applications to stream mining. In *ICDM*, pages 623–632, 2006.

[200] J. P. Vert, K. Tsuda, and B. Schoelkopf. A primer on kernel methods. *Kernel Methods in Computational Biology*, pages 35–70, 2004.

[201] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, and E. J. Keogh. Indexing multi-dimensional time-series with support for multiple distance measures. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 216–225, New York, NY, USA, 2003. ACM.

[202] E. I. Vlahogianni and M. G. Karlaftis. Comparing traffic flow time-series under fine and adverse weather conditions using recurrence-based complexity measures. *Journal on Nonlinear Dynamics*, 69(4):1949–1963, 2012.

[203] M. Wattenberg. Arc diagrams: Visualizing structure in strings. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 110–116, 2002.

[204] C. L. Webber, N. Marwan, A. Facchini, and A. Giuliani. Simpler methods do it better: Success of recurrence quantification analysis as a general purpose data analysis tool. *Physics Letters A*, 373(41):3753–3756, 2009.

[205] C. L. Webber and J. P. Zbilut. Dynamical assessment of physiological systems and states using recurrence plot stategies. *Journal of Applied Physiology*, 76(2):965–973, 1994.

[206] L. Wei and E. J. Keogh. Semi-supervised time series classification. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 748–753, New York, NY, USA, 2006. ACM.

[207] L. Wei, E. J. Keogh, H. V. Herle, A. Mafra-Neto, and R. J. Abbott. Efficient query filtering for streaming time series with applications to semisupervised learning of time series classifiers. *Knowl. Inf. Syst.*, 11(3):313–344, 2007.

[208] C. Weihs, U. Ligges, and K. Sommer. Analysis of music time series. In A. Rizzi and M. Vichi, editors, *Compstat 2006 - Proceedings in Computational Statistics*, pages 147–159. Physica-Verlag HD, 2006.

[209] M. Weiss, A. Helfenstein, F. Mattern, and T. Staake. Leveraging smart meter data to recognize home appliances. In *Proc. of IEEE International Conference on Pervasive Computing and Communications*, pages 190–197, 2012.

[210] A. Wismueller, O. Lange, D. R. Dersch, G. L. Leinsinger, K. Hahn, B. Puetz, and D. Auer. Cluster analysis of biomedical image time-series. *Int. J. Comput. Vision*, 46(2):103–128, 2002.

[211] X. Xi, E. J. Keogh, C. R. Shelton, L. Wei, and C. A. Ratanamahatana. Fast time series classification using numerosity reduction. In *ICML*, pages 1033–1040, 2006.

[212] N. Xiong and P. Svensson. Multi-sensor management for information fusion: issues and approaches. *Information Fusion*, 3(2):163–186, 2002.

[213] L. Yang and R. Jin. Distance metric learning: A comprehensive survey. Technical report, Department of Computer Science and Engineering, Michigan State University, 2006.

[214] D. Yankov, E. J. Keogh, J. Medina, B. Chiu, and V. Zordan. Detecting time series motifs under uniform scaling. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pages 844–853, New York, NY, USA, 2007. ACM.

[215] L. Ye and E. J. Keogh. Time series shapelets: a new primitive for data mining. In *KDD*, pages 947–956, 2009.

[216] B.-K. Yi, H. V. Jagadish, and C. Faloutsos. Efficient retrieval of similar time sequences under time warping. In *ICDE*, pages 201–208, 1998.

[217] J. Zakaria, A. Mueen, and E. J. Keogh. Clustering time series using unsupervised-shapelets. In *ICDM*, pages 785–794, 2012.

[218] J. P. Zbilut and C. L. Webber. Embeddings and delays as derived from quantification of recurrence plots. *Physics Letters A*, 171(3-4):199–203, 1992.

[219] J. P. Zbilut, J.-M. Zaldivar-Comenges, and F. Strozzi. Recurrence quantification based liapunov exponents for monitoring divergence in experimental data. *Physics Letters A*, 297(3-4):173–181, 2002.

[220] M. Zeifman and K. Roth. Nonintrusive appliance load monitoring: Review and outlook. *IEEE Trans. Consumer Electronics*, 57(1):76–84, 2011.

[221] Q. Zhu, G. E. A. P. A. Batista, T. Rakthanmanon, and E. J. Keogh. A novel approximation to dynamic time warping allows anytime clustering of massive time series datasets. In *SDM*, pages 999–1010, 2012.

[222] A. Zoha, A. Gluhak, M. A. Imran, and S. Rajasegarar. Non-intrusive load monitoring approaches for disaggregated energy sensing: A survey. *Sensors*, 12(12):16838–16866, 2012.