UNIVERSITY OF NOVI SAD

FACULTY OF SCIENCE

DEPARTMENT OF MATHEMATICS
AND INFORMATICS

Zoltan Geler

Role of Similarity Measures in Time Series Analysis

- Doctoral Dissertation -

Mentor:

Prof. Dr. Mirjana Ivanović

Novi Sad, 2015

Preface

The main subject of this dissertation is a comprehensive examination of the role of similarity measures in time-series analysis through studying the influence of the Sakoe-Chiba band on accuracy of classification.

A time series represents a series of numerical data points in successive order, usually with uniform intervals between them. Time series are used for storage, analysis and visualization of data collected in many different domains, including science, medicine, economics and others. The increasing demand to work with large amounts of data has led to a growing interest in researching different tasks of time-series data mining. In recent years, there is an increasing interest in studying various aspects of time-series classification.

One of the most important issues of time-series classification is a thoughtful and adequate choice of the similarity measure. They enable comparison of time series by describing their similarity (or dissimilarity) with numerical values. In the field of time-series data mining a great number of different similarity measures has been proposed and used.

Many of these measures are implemented using dynamic programming. Unfortunately, the time requirements of this technique may adversely affect the possibilities of its application in practice. One way of dealing with this issue is to limit the search area by applying global constraints. However, these restrictions may affect the accuracy of classification, and therefore understanding their impact is of great importance.

The main task of this thesis is to discover and explain the role of these constrained similarity measures in the field of time-series analysis and data mining with a focus on classification accuracy.

This dissertation is divided into the following six chapters:

1. Introduction
2. Time Series and Similarity Measures
3. Time-Series Classification
4. Methods, Tools and Data Sets
5. Techniques for Improving Classification Accuracy
6. Framework for Time Series Analysis (FAP)
7. Conclusion

In the first chapter a concise overview of the related topics is given in conjunction with the objects of the dissertation.

The necessary basic concepts of time-series data mining, including the formal definition of time series, descriptions of the analyzed similarity measures and global constraints are given in chapter two.

Chapter three is devoted to time-series classification. A detailed overview of the nearest neighbor rule and its extensions along with several different weighting schemes are discussed in this chapter.

Chapter four describes the methods, tools and datasets used in our research.

A detailed description of the extensive experiments performed and discussion of results are presented in chapter five.

All experiments within this thesis were performed using our free Java library for time-series analysis and data mining (*Framework for Analysis and Prediction* developed at Department of Mathematics and Informatics, Faculty of Sciences, University of Novi Sad). The details of its structure and implementation are given in chapter six.

Chapter seven concludes the dissertation and lists some possible directions for future research.

The appendices of this dissertation contain detailed results of the experiments described in chapter five.

In the end, for an introduction to the extremely interesting field of time-series data mining, for valuable help, unreserved support and patient guidance through my research I am especially indebted to my supervisor Mirjana Ivanović. This thesis would not have been possible without her, and without encouragement she has given me over the last years.

I would also like to thank warmly all the members of the committee for their patience and valuable suggestions regarding this thesis. My special thanks and gratitude are addressed to Vladimir Kurbalija and Miloš Radovanović for their stimulating suggestions and advice during my research.

My special thanks are also goes to other colleagues and professors at the Department of Mathematics and Informatics, Faculty of Sciences and the Department of Media Studies, Faculty of Philosophy for a nice and pleasant studying and working environment.

I would like to thank Eamonn Keogh for collecting and making available the UCR time series datasets, as well as everyone who contributed data to the collection, without whom the experiments would not have been possible.

Finally, I thank my parents and brother for their endless support and patience during my studies.

Novi Sad, 2015 Zoltan Geler

Contents

Chapter 1

Introduction

Data mining can be defined as the process of discovering and extracting interesting and useful information from large databases [39, 63, 113, 126]. Temporal data mining is concerned with knowledge discovery from large datasets of temporal data [63, 77, 113]. This form of data represents sequential data ordered with respect to time [63, 117]. Time series are the most popular type of temporal data: they consist of real values usually sampled at regular time intervals [63, 77]. We can distinguish between labeled and unlabeled time series on the basis of whether or not there is a class label associated with them.

Each element of a time series describes the phenomenon under examination at a specific point in time. Depending on whether the observations were carried out continuously or at regular intervals we can differentiate continuous and discrete time series [13]. In this dissertation, we consider the specific form of discrete time series whose elements are uniformly spaced real numbers.

Time series are used for storage, display and analysis of data across a wide range of different domains, including various areas of science, medicine, economics, ecology, telecommunications and meteorology [25, 40, 63]. Esling and Agon [25] emphasize that in almost every scientific field, measurements are performed over time and that the collected data can be organized in the form of time series with the aim of extracting meaningful knowledge. For finding new and useful information from the sets of collected data we can rely on methods from statistical analysis and modeling, data mining and machine learning [63].

While research in statistical modeling techniques has a long history [63], the need to process increasing volumes of data has heightened the interest for studying different task types of temporal data mining: indexing, classification, clustering, prediction, segmentation, anomaly detection and others [21, 39]. The area of interest of this dissertation is primarily related to time-series classification.

In recent years, there is a growing interest for research in different aspects of time-series classification [34, 37, 45, 93, 111, 130, 131] - see Chapter 3. The possibility of applying many well-known machine learning techniques was investigated in the field of time-series classification. These techniques include: decision trees [102], neural networks [81], support vector machines [128], first order logic rules [101], Bayesian classifiers [85] and others. However, it was shown that the simple nearest-neighbor (1NN) approach often produces better results than the mentioned more complex classifiers for the time-series data [130].

The nearest neighbor algorithm (1NN) is probably one of the most esteemed algorithms in data mining [127]. It is based on the following very simple idea: unknown samples are placed into the class of their nearest neighbors [18]. The majority voting k-nearest neighbor (kNN) rule generalizes this concept by finding the k nearest neighbors and choosing the class that is most frequent among them [28]. The distance-weighted k-nearest neighbor rule proposed by Dudani [22] assigns weights to the neighbors according to their distance from the unclassified sample: greater significance is given to closer neighbors. This rule selects the class which produces the largest sum of the weights among the k nearest neighbors of the unclassified sample.

Since finding the nearest neighbors constitutes the core idea behind the kNN rule, one of the most essential questions of its implementation is related to the selection of an appropriate distance measure. In the domain of time series, several different similarity-based distance measures are applied for comparing data sequences [31, 68, 108]. The most commonly used and most frequently investigated time-series similarity measures are Euclidean distance [26], Dynamic Time Warping (DTW) [10], Longest Common Subsequence (LCS) [121], Edit Distance with Real Penalty (ERP) [16], and Edit Distance on Real sequence (EDR) [17].

Dynamic programming represents the basic technique of implementation for the vast majority of similarity measures, but because of its quadratic computational complexity it is often not suitable for larger real-world problems. To address this shortage, one can restrict the search area using global constraints such as the Sakoe-Chiba band [105] and the Itakura parallelogram [44]. In Section 5.2 we will show that this can significantly speed up the calculation (a part of these results is published in [59]).

Apart from speeding up the computation it was also suggested that the usage of global constraints can actually improve the accuracy of classification compared to unconstrained similarity measures [98, 130]. The accuracy of classification is commonly used as a qualitative assessment of a similarity measure [98]. Given all these positive effects of global constraints, it is important to carefully investigate their impact on various similarity measures.

The 1-nearest-neighbor graph (1NN) is a directed graph where each time series is connected with its nearest neighbor. Since the 1NN classifier assigns the class of the nearest neighbor to a yet unclassified time series, the changes in the 1NN graph directly affect classification accuracy. In Section 5.3, the results of which are published in [59] and [60], we will explore the change of the 1NN graph (with respect to the change of the constraint size) in order to better understand the influence of global constraints and to provide deeper insight into their advantages and limitations. Also, we will examine how these changes reflect on the nearest-neighbors' classes and investigate their impact on the accuracy of the 1NN classifier.

The choice of 1NN classifier was motivated by reports of its superiority [21, 93, 119, 130]. This method achieved among the best results compared to many other sophisticated classifiers for time-series data. In addition, the accuracy of 1NN directly reflects the quality of the underlying similarity measure [116], the investigation of which is one of our goals.

We expect that our results will aid researchers and practitioners in selecting and tuning appropriate time-series similarity measures for their respective tasks since, as we will show, constraints introduce qualitative differences in all considered measures. Furthermore, the insight into the behavior of similarity measures with respect to changing constraints can be beneficial to the design of efficient indexing strategies for fast computation of (approximate) nearest neighbors. This statement is supported by the report [98] that measures with the values of constraints around 5% produce the same or almost the same classification accuracies as unconstrained measures. In addition, as we will show in Section 5.2, the difference of computation times between an unconstrained measure and a measure with such a small constraints is two and somewhere three orders of magnitude.

All mentioned experiments for distance-measure assessment were conducted with the 1NN classifier as it was shown that it gives among the best results (compared to many classifiers, not only distance-based) with time-series data [21, 130]. This fact strongly indicates that the nearest neighbor has a particularly important meaning in time-series classification. In [93], the reasons and origins of this special behavior of the nearest neighbor are investigated, and related with the observed diversity of class labels in *k*-neighborhoods.

In Section 5.4, we will compare the accuracies of 1NN and kNN classifiers when using the two most representative time-series distance/similarity measures based on dynamic programming (DTW and LCS), in order to understand the special meaning of the first neighbor. Furthermore, we will attempt to improve the accuracy of kNN by favoring the first (few) neighbors.

Several different methods for assigning weights to nearest neighbors are proposed in the literature [22, 35, 36, 62, 69, 74, 84, 133] - all of these weighting schemes are presented in detail in Section 3.1. Generally, each paper that presents a new way of computing weights reports the superiority of the new method compared to some previous solutions. Several of these papers ([35, 36, 133]) compare various weighting schemes using a relatively small number of datasets (commonly 12) from the UCI machine learning repository [7]. The conclusions are usually based on comparing classification results using Euclidean distance.

The aim of the study in Section 5.5 is to compare different weighting schemes from the above mentioned papers in the domain of time series. Our research is motivated by the view that the simple 1NN rule gives better results than the majority voting kNN rule [21]. We will investigate whether the proposed weighting schemes can produce higher classification accuracies than the 1NN classifier. In order to achieve this objective our research encompasses the majority of all publicly available, labeled time-series datasets in the world which are provided by the UCR Time Series Repository [50]. Moreover, our examinations embrace the three most commonly used time-series similarity measures (Euclidean distance and the unconstrained DTW and LCS) and provide statistical support to the results obtained.

Usefulness of time series in the analysis of social, economic and natural phenomena has increased the importance of studying different fields of time-series data mining, which has led to the development of many new solutions. All these solutions are typically implemented separately and described in different publications. Development of a publicly

available free and open source library that contains the implementation of the most important algorithms for analysis and mining time series can support and facilitate researching new and comparing existing techniques in this domain. All of the experiments whose results are presented in this dissertation were carried out using our freely available *Framework for Analysis and Prediction* (FAP) which is presented in [58]. The basic features of this library are described in Section 4.1, and a detailed overview of its capabilities is given Chapter 6.

Chapter 2

Time Series and Similarity Measures

A time series represents the simplest form of temporal data: a series of numbers that describes the change of the observed phenomenon over time. Each number in a time series describes the phenomenon at one point in time [19]. Time series are suitable for representing social, economic and natural phenomena, medical observations, and results of scientific and engineering experiments.

They are used for prediction, anomaly detection, clustering and classification, which increased the importance of different research areas of temporal data mining and resulted in a large amount of work introducing new methodologies [19, 21, 39].

A d-dimensional time series Q can be defined as a sequence of ordered pairs as follows [6]:

$$\mathrm{Q} = \big((q_1, t_1), (q_2, t_2), \dots, (q_n, t_n)\big)$$

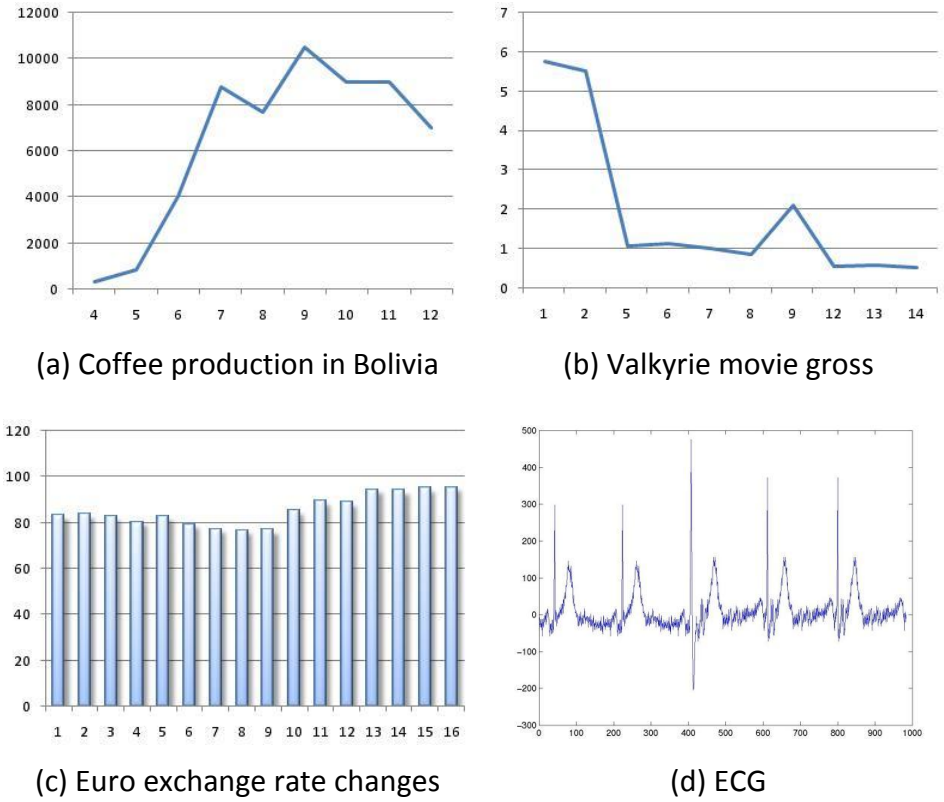
where $t_i \in T$ represents the time component, and $q_i \in R^d$ denotes the measured value of the observed phenomenon at time t_i. It is assumed that the time series is sorted by the time component, i.e. that

$$t_i < t_{i+1},\ \forall i \in \{1, \dots, n-1\}.$$

In this dissertation, we consider one-dimensional time series ($q_i \in R$) assuming that the time distance between the ordered pairs is constant, i.e. $t_{i+1} = t_i + c$, $\forall i \in \{1, \dots, n-1\}$, where c is a constant positive value. In this manner, the time components of Q can be omitted and it can be viewed as a sequence of real numbers: $Q = (q_1, q_2, \dots, q_n)$. As an example, Figure 2.1 (a) presents one possible graphical representation of time series $Q = (300, 800, 4000, 8800, 7700, 10500, 9000, 9000, 7000)$ which describes coffee production in Bolivia[1] for the period from April to December 2008. Figure 2.1 (b) depicts another example - the retained earnings (in millions of dollars) of the movie "Valkyrie" in cinemas[2] at the beginning of January 2009. Figure 2.1 (c) contains time series that illustrates Euro exchange rate changes[3] from January 2008 to April 2009 at the National Bank of Serbia

[1] http://www.ico.org
[2] http://www.movieweb.com
[3] www.nbs.rs

(NBS). In Figure 2.1 (d) we can see an ECG recording[4] that exemplifies the application of time series in the field of medicine.



(a) Coffee production in Bolivia (b) Valkyrie movie gross

(c) Euro exchange rate changes (d) ECG

Figure 2.1. Examples of time series

Slika 2.1. Primeri vremenskih serija

Time-series data arise from and are used in many different application fields including finance, medicine, and various domains of science. The increasing need for applying time series in order to solve miscellaneous problems led to an explosive growth of interest in researching different task types of time-series analysis. These tasks include [95]:

- **Indexing**: finding time series in a database that is most similar to a given time series Q based on some similarity/distance measure d.
- **Clustering**: finding a natural way of grouping the time series of a database based on some similarity/distance measure d.
- **Classification**: assigning unlabeled time series to one or more predefined classes.
- **Predicting**: predicting future values and behavior of time series - given a time series Q containing n values, predict the value at time $n + 1$.

[4] http://www.owlnet.rice.edu/~elec301/Projects02/empiricalMode/app.html

- **Summarization**: for a given time series Q which contains an extremely large number of points, create an approximation (reducing the number of points) that will keep the essential characteristics of Q.

- **Anomaly detection**: on the basis of a given time series Q which is considered to be normal, find all the parts of a non-interpreted time series S which represent anomalies or unexpected, interesting, surprising occurrences.

- **Segmentation**: we can distinguish between two sub-areas:
    - Given a time series Q containing n data points, on the basis of K piecewise segments ($K \ll n$) create a model $\bar{Q}$ which is a close approximation of Q.
    - Given a time series Q, partition it into K internally homogeneous sections.

2.1. Similarity Measures

One of the most important aspects of time-series analysis is the choice of appropriate similarity/distance measure – the measure which tells to what extent two time series are similar. However, unlike data types in traditional databases where the similarity/distance definition is straightforward, the distance between time series needs to be carefully defined in order to reflect the underlying (dis)similarity of these specific data, which is usually based on shapes and patterns.

The distance between two time series $Q = (q_q, q_2, \dots, q_n)$ and $S = (s_1, s_2, \dots, s_m)$ is defined using a proximity measure - a function that returns the nonnegative distance $d(Q,S)$ between them [25]. A *distance metric* is a proximity measure that for every time series Q, S and X satisfies the following conditions:

1. **Reflexivity**: $d(Q,S) = 0$ if and only if $Q = S$,
2. **Symmetry**: $d(Q,S) = d(S,Q)$,
3. **Triangle inequality:** $d(Q,S) \leq d(Q,X) + d(X,S)$,

The distance between two time series specifies their degree of (dis)similarity: greater distance denotes less similarity. On the other hand, higher values of a similarity measure between two time series indicates smaller distances between them. The similarity is usually taken between 0 and 1, where 0 indicates that the objects are unalike, and 1 denotes that they are identical. A distance function operates the opposite way: it returns 0 if the objects are the same [39]. Converting distance into similarity, and vice versa is possible by means of the following formula [54, 91]:

$$sim(Q,S) = \frac{1}{1 + dist(Q,S)}$$

As expected, there exists a large number of measures for expressing (dis)similarity of time-series data proposed in the literature. Overviews of different (dis)similarity measures can be found in [20], [25], [31] and [108]. In the rest of this section we will outline the measures which are the subject of this dissertation. In this dissertation, unless explicitly stated, we will not distinguish between similarity and distance measures, and will use the two terms interchangeably.

2.1.1. Euclidean Distance

The most common similarity measure in time-series data mining is probably the Euclidean distance [4, 26, 129]. Assuming that two time series, $Q = (q_q, q_2, \dots, q_n)$ and $S = (s_1, s_2, \dots, s_n)$, are of the same length n, we can think of them as points in n-dimensional space. In this manner we will be able to calculate their distance relying on the differences between the corresponding elements of the sequences as it is shown in Eq. (2.1).

$$d(Q,S) = \sqrt{\sum_{i=1}^{n}(q_i - s_i)^2} \tag{2.1}$$

The advantage of Euclidean distance is that it is very easy to compute and to understand. Furthermore, it fulfills the above mentioned conditions to be a distance metric and therefore it can be used for indexing time series in databases [26]. There are, however, some disadvantages, too: the sequences must have the same number of points (can be avoided by interpolation to equal length [98]), it is sensitive to shifting and scaling along the y-axis (can be precluded by normalizing the series [19, 32]), and it is also sensitive to distortions and shifting along the time axis [49].

Euclidean distance is a special case of the Minkowski distance (L_p norm), defined by Eq. (2.2):

$$L_p(Q,S) = \sqrt[p]{\sum_{i=1}^{n}|q_i - s_i|^p} \tag{2.2}$$

The most commonly used L_p norms are Euclidean distance ($p = 2$), Manhattan distance ($p = 1$) and Chebyshev distance ($p = \infty$) which is defined as follows:

$$L_\infty(Q,S) = \max_{i=1,\dots,n}|q_i - s_i|$$

2.1.2. Dynamic Time Warping (DTW)

Euclidean distance is based on linear aligning of related points of time series (Figure 2.2 (a)): the i-th point of the first series is paired with the i-th point of the second one. The assessment of the similarity can be improved by warping the time axis of one or both sequences (Figure 2.2 (b)). One of the most popular similarity measures based on non-linear aligning is the Dynamic Time Warping (DTW) [10, 49, 130].

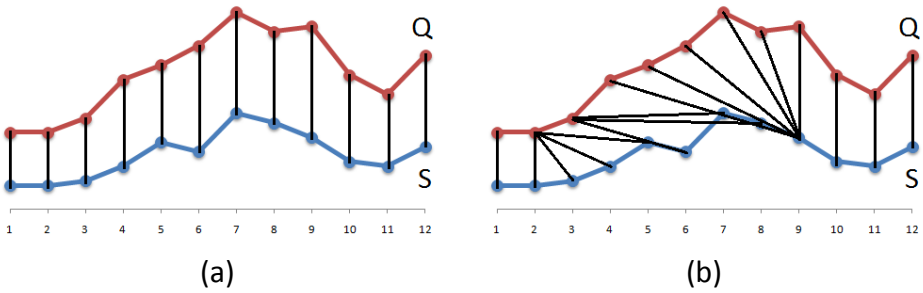


Figure 2.2. Linear (a) and non-linear (b) aligning of points

Slika 2.2. Linearno (a) i nelinearno (b) uparivanje tačaka

Let $Q = (q_q, q_2, \dots, q_n)$ and $S = (s_1, s_2, \dots, s_m)$ be two time series of lengths n and m. To align these two time series using DTW, we construct an n-by-m warping matrix $D = [d_{ij}]_{n,m}$ (Figure 2.3), where d_{ij} represents the squared distance between q_i and s_j, i.e. $d_{ij} = (q_i - s_j)^2$.

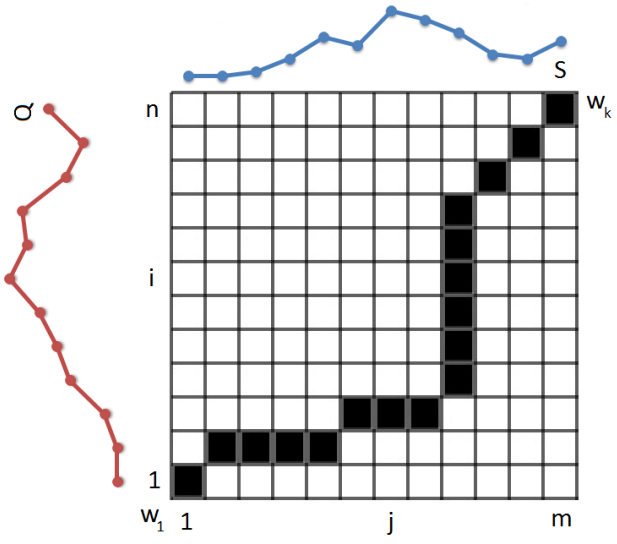


Figure 2.3. Optimal warping path inside the warping matrix

Slika 2.3. Optimalan put iskrivljenja unutar matrice iskrivljenja

DTW searches for the optimal warping path (the one that minimizes the total cumulative distance between Q and S) in the warping matrix D (shown with solid squares in Figure 2.3).

A warping path $W = (w_1, w_2, \dots, w_k)$ (where $max(n,m) \leq k \leq m+n-1$) represents a sequence of adjacent cells from the matrix. Each element of a warping path is of the form $w_i = (x_i, y_i)$ (where x_i denotes a row of the matrix and y_i denotes a column of the matrix) and the warping path must satisfy the following constraints:

- **Boundary condition** - the first and the last element of the warping path are in diagonally opposite corners of the matrix: $w_1 = (1,1)$, $w_k = (n,m)$.
- **Continuity condition** - cells of the matrix denoted by adjacent elements of the warping path must be neighbors: $x_{i+1} - x_i \leq 1$, $y_{i+1} - y_i \leq 1$.
- **Monotonity condition** - the warping path must be monotonically non-decreasing along the time axis: $x_{i+1} - x_i \geq 0$, $y_{i+1} - y_i \geq 0$.

From the set of all possible warping paths we are seeking for the optimal one, which minimizes the warping cost (the sum of the cells that constitute the warping path). This can be found using dynamic programming, as recursively defined by Eq. (2.3).

$$D(i,j) = \begin{cases} 0 & i = j = 0, \\ \infty & i = 0, j > 0 \text{ or } i > 0, j = 0, \\ d(q_i, s_j) + min \begin{cases} D(i-1, j-1) \\ D(i-1, j) \\ D(i, j-1) \end{cases} & i, j \geq 1. \end{cases} \tag{2.3}$$

where $d(q_i, s_j)$ is the distance between q_i and s_j. The distance between Q and S is then defined as $DTW(Q,S) = D(n,m)$. Euclidean distance can be seen as a special warping path which contains only the diagonal elements on the distance matrix, and which is defined for time series of the same length as $\sqrt{DTW(Q,S)}$.

DTW is used in a wide variety of different domains including: satellite image time series analysis [87], human action recognition [89], fault diagnosis of motor drives [139], detecting melodic motifs from audio [103], exploring dynamic mobility patterns of urban areas from mobile phone data [132] and voice command recognition [9].

2.1.3. Longest Common Subsequence (LCS)

The Longest Common Subsequence (LCS) approach calculates the distance between two sequences relying on a variation of the edit distance technique - a well known method in the field of string processing. The basic idea is to express the similarity of the time series based on the length of their longest common subsequence [121]. The length of the LCS can be computed using dynamic programming based on the recursive definition in Eq. (2.4). The condition $q_i = s_j$ is usually too strong for time series and it is often replaced with a parameterized condition $|q_i - s_j| \leq \epsilon$, where $0 < \epsilon < 1$. The dissimilarity $LCS(Q,S)$ between

two time series $Q = (q_q, q_2, \dots, q_n)$ and $S = (s_1, s_2, \dots, s_m)$ of length n and m is calculated according to Eq. (2.5) as presented in [95] and [19].

$$L(i,j) = \begin{cases} 0 & i = 0 \text{ or } j = 0, \\ 1 + L(i-1, j-1) & i,j > 0 \text{ and } q_i = s_{j,} \\ max\big(L(i-1,j), L(i,j-1)\big) & i,j > 0 \text{ and } q_i \neq s_j. \end{cases} \tag{2.4}$$

$$LCS(Q,S) = \frac{n + m - 2 \times L(n,m)}{n+m} \tag{2.5}$$

A framework based on LCS for action representation and recognition is presented in [122]. In [41] another framework based on LCS is proposed to solve the similarity search problem given user-defined instance-level constraints for tropical cyclone events, represented by arbitrary-length multidimensional spatiotemporal data sequences. The suitability of LCS for history-based travel-time predictions for vehicles traveling on known routes is explored in [118].

2.1.4. Edit Distance with Real Penalty (ERP)

Ding et al. [21] refer to distance measures that utilize linear aligning between the points of time series as *lock-step* measures (Euclidean distance and other forms of the L_p norm for $p \geq 1$). One of their main advantages is that they represent distance metrics and thus they can be easily applied for indexing in databases. However, the fixed mapping between the points makes them sensitive to *noise* (random error that occurs in the data mining process [77]) and time shifting. *Elastic* measures like DTW and LCS address these issues by allowing *one-to-many* (DTW) and *one-to-many*/*one-to-none* (LCS) mappings [21]. Since neither DTW nor LCS satisfy the triangle inequality [110], they are non-metric distance measures.

In [16] Chen and Ng propose the ERP distance function as a combination of the L_1 norm and elastic similarity measures. To handle local time shifting it calculates the real penalty between non-gap elements using L_1 distance. The distance for gaps is computed based on a constant value denoted by g (the default value is 0) in the definition of this measure (Eq. (2.6)). The distance between two time series $Q = (q_q, q_2, \dots, q_n)$ and $S = (s_1, s_2, \dots, s_m)$ of length n and m is then defined as $ERP(Q,S) = E(n,m)$. ERP is a distance metric, but it is sensitive to noise [17].

$$E(i,j) = \begin{cases} \sum_{k=1}^{j} |s_k - g| & i = 0, \\ \sum_{k=1}^{i} |q_k - g| & j = 0, \\ min\begin{cases} |q_i - s_j| + E(i-1, j-1) \\ |q_i - g| + E(i-1, j) \\ |s_j - g| + E(i, j-1) \end{cases} & otherwise \end{cases} \tag{2.6}$$

ERP was successfully used in solving various problems including classification of pulse waveforms [134], clustering trajectories of moving objects [86], querying time-series streams [33] and creating driving behavior for artificial agents [107].

2.1.5. Edit Distance on Real sequence (EDR)

EDR [17] is an elastic similarity measure based on edit distance which has been developed with the aim to improve the accuracy of LCS in the case when the time series contains similar sub-sequences with gaps of different sizes between them. EDR is robust to noise, outliers (instances that are in some sense different from the majority of others [91]) and local time shifting. In contrast to LCS, EDR assigns penalties according to the lengths of the gaps, but EDR also does not represent a distance metric.

Let $Q = (q_q, q_2, \dots, q_n)$ and $S = (s_1, s_2, \dots, s_m)$ be two time series of length n and m. The Edit Distance on Real Sequence (EDR) between Q and S is the number of insert, delete, or replace operations that are needed to change Q into S. The distance between Q and S is defined as $EDR(Q,S) = E(n,m)$ where the elements of matrix E are calculated recursively as follows:

$$E(i,j) = \begin{cases} j & i = 0, \\ i & j = 0, \\ min \begin{cases} subcost(i,j) + E(i-1,j-1) \\ 1 + E(i-1,j) \\ 1 + E(i,j-1) \end{cases} & otherwise \end{cases} \quad (2.7)$$

The *subcost* in Eq. (2.7) represents the cost of a replace, insert, or delete operation and it is calculated by the following formula:

$$subcost(i,j) = \begin{cases} 0 & |q_i - s_j| \leq \varepsilon, \\ 1 & |q_i - s_j| > \varepsilon. \end{cases}$$

where $0 < \epsilon < 1$.

A novel clustering method of trajectories based on EDR is presented in [2]. In [64] EDR is used for subtrajectory-based video indexing and retrieval. An adaptation of EDR and LCS for the shape similarity of fibers is presented in [70].

2.2. Global Constraints of Time-Series Similarity Measures

Each of the above-mentioned elastic similarity measures (DTW, LCS, ERP and EDR) relies on dynamic programming for finding the optimal path within the search matrix. Dynamic programming requires comparing each element of one time series with each element of the other one. This makes the calculation of the similarity measures quite slow and has some limitations when dealing with large datasets.

To improve the performance of these algorithms a number of techniques have been developed. The Sakoe-Chiba band [105] narrows the warping window around the diagonal of the matrix using a constant range r. The Itakura parallelogram [44] uses a similar approach: the range of the restriction is a function of i and j coordinates in the matrix. These restrictions of the search path are illustrated in Figure 2.4. Global constraints were originally introduced to prevent some bad alignments, where a relatively small part of one time series maps onto a large section of another time series.

Apart from speeding up the computation it was also suggested that the use of global constraints can actually improve the accuracy of classification compared to unconstrained similarity measures [98, 130]. In Chapter 5, we will explore a wide variety of r values and examine their effect on DTW, LCS, ERP and EDR distance measures constrained by the Sakoe-Chiba band. In Section 5.2 we will show that global constraints can significantly reduce the computation time of the elastic similarity measures. In Section 5.3 we will demonstrate that the constrained measures are qualitatively different from their unconstrained counterparts and explain how they can produce better classification accuracies compared to the unconstrained ones.

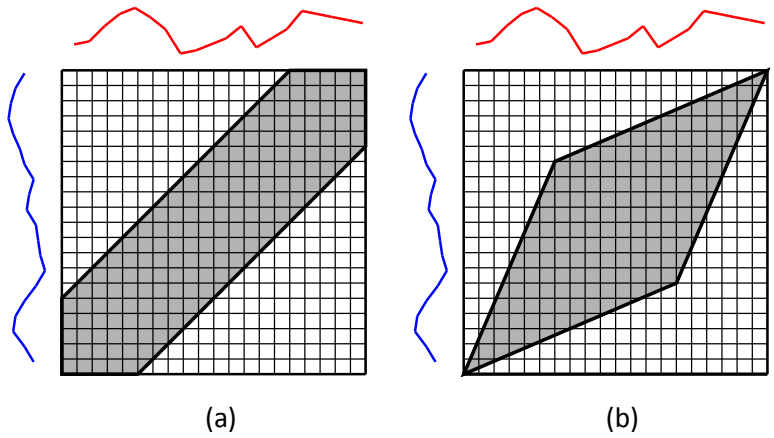


Figure 2.4. Sakoe-Chiba band (a) and Itakura parallelogram (b)

Slika 2.4. Sakoe-Chiba pojas (a) i Itakura paralelogram (b)

To illustrate the effects of constraining the warping window using the Sakoe-Chiba band, in Figure 2.5 we give a visual review of the warping windows and the associated optimal warping paths for different values of parameter r. These images were made by applying the DTW similarity measure (Section 2.1.2) on the following two time series (they are listed here with rounding to 3 decimal places) from the *ItalyPowerDemand* dataset:

$$Q = (-0.711, -1.183, -1.372, -1.593, -1.467, -1.372, -1.089, 0.046, 0.929, 1.086, 1.275, 0.960, 0.613, 0.014, -0.647, -0.269, -0.206, 0.613, 1.370, 1.464, 1.055, 0.582, 0.172, -0.269)$$

$$S = (-0.973, -1.183, -1.394, -1.464, -1.464, -1.464, -1.558, -0.668, 0.432, 1.064, 1.111, 1.041, 1.064, 0.877, 0.807, 1.017, 1.064, 0.736, 0.573, 0.268, 0.081, -0.013, 0.245, -0.200)$$

The *ItalyPowerDemand* dataset contains 1096 time series composed of 24 real numbers (see Table 4.1 in Section 4.2). Each time series of this dataset belongs to one of two classes. The above listed time series Q and S belong to different classes. Their graphical display is given in Figure 2.6.

Table 2.1 shows the examined different values of parameter r along with the corresponding warping window widths and the obtained distances between Q and S.

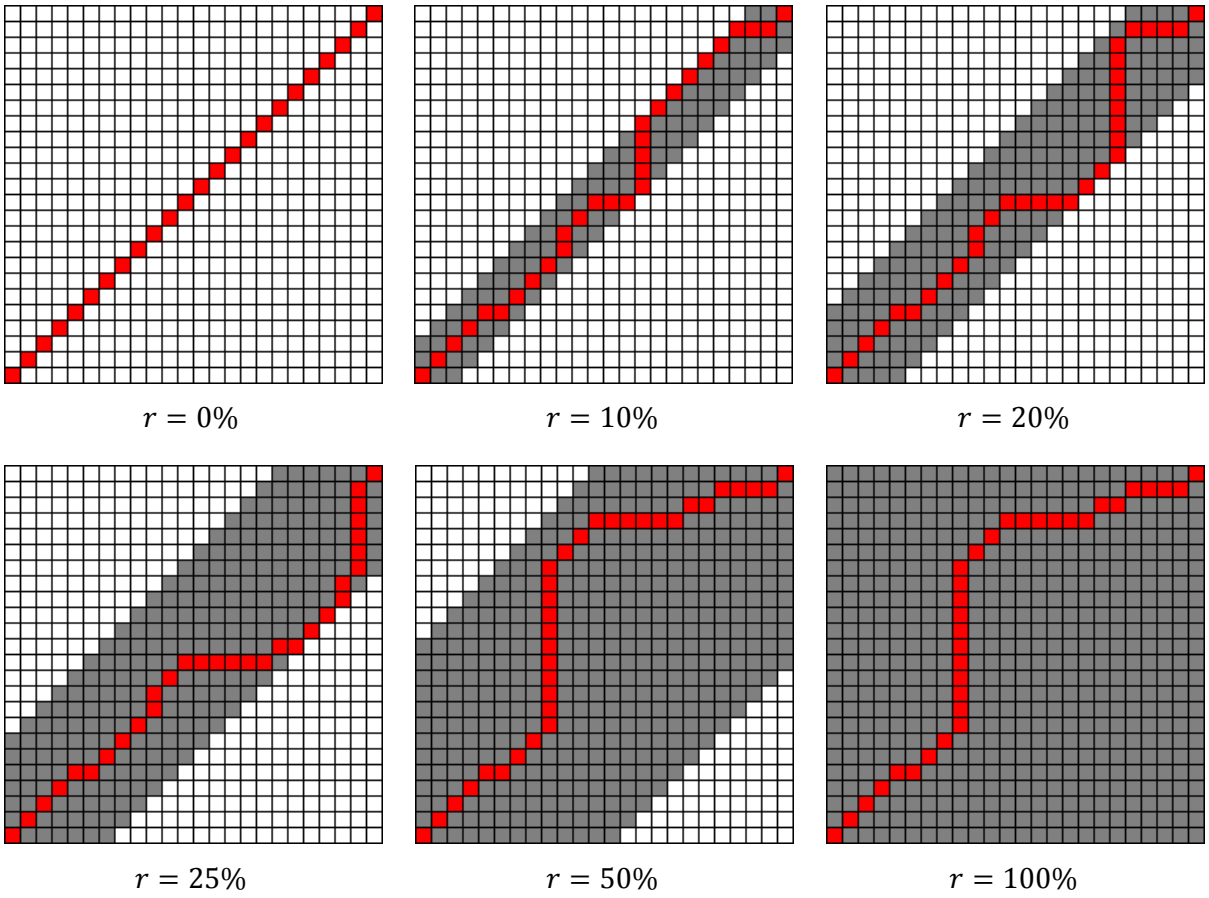


Figure 2.5. Warping windows and optimal warping paths for different values of parameter r

Slika 2.5. Prozori iskrivljenja i optimalni putevi iskrivljenja za različite vrednosti parametra r

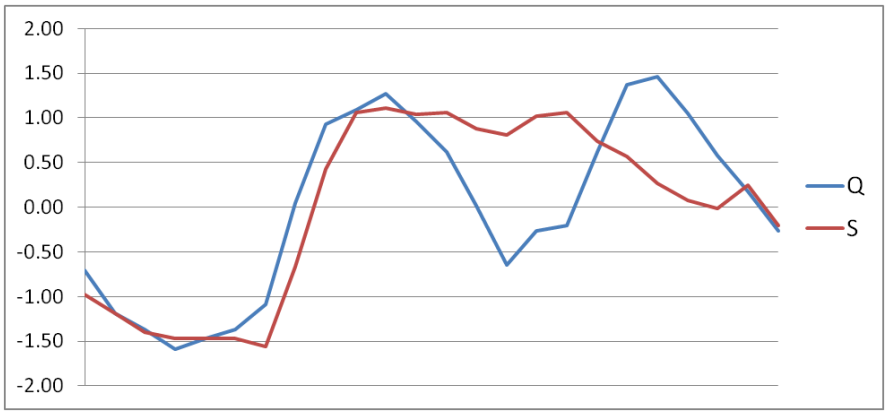


Figure 2.6. A graphical display of two time series from the *ItalyPowerDemand* dataset

Slika 2.6. Grafički prikaz dve vremenske serije iz skupa *ItalyPowerDemand*

r (%)	Length	Distance
0	0	10.829
10	2	6.649
20	4	5.798
25	6	4.937
50	12	4.814
100	23	4.814

Table 2.1. Different values of parameter r with the corresponding warping window widths and the obtained distances

Tabela 2.1. Različite vrednosti parametra r sa pripadajućim širinama prozora iskrivljenja i dobijena rastojanja

Chapter 3

Time-Series Classification

Classification is the process of grouping objects into predefined categories, classes. It is always done on the basis of a selected attribute (*class label*), which can have a finite number of different values. This way we always know the total number of different classes in advance. This is a key difference between classification and clustering - in case of clustering we do not know the number of groups in advance, groups are revealed during the process of clustering. Thus, classification can be considered as *supervised learning* and clustering as *unsupervised learning*. The purpose of a supervised learning algorithm is to create a prediction function that will make predictions for unseen objects based on a set of labeled examples of training data [46, 78]. On the other hand, an unsupervised learning algorithm is used to learn patterns exclusively from unlabeled objects [46, 78].

A typical task in data mining is to train a classifier based on available data. The aim of classification is to find rules that will ensure accurate mapping of objects (in our case, time series) into predefined classes. In general, the classification process consists of the following three main steps [1, 53]:

1. **Training the classifier**: we train the classifier based on a predetermined set of objects (the *training set*) using a learning algorithm. The class label must be known for each object of the training set. The training set is used for forming the classification rules, i.e. a function that will map every object of the set into a predefined class.

2. **Testing the classifier**: after training, the obtained classifier must be verified using a *test set*. The class label must be known also for each object form the test set. The *accuracy* of the classifier is checked by applying it to the objects of the test set and comparing the obtained class labels with the real labels of the objects. If the accuracy of the classifier is not at a satisfying level, we need to repeat the training process. In order to improve the accuracy of the classifier we must take into account several possibilities: maybe we need a larger training set, perhaps some important characteristics of the problem have not taken into consideration, maybe the selected algorithm needs parameter tuning, or maybe it is not appropriate for the current problem.

3. **Applying the classifier**: If the resulting classifier has an acceptable accuracy on the testing set, we can use it for classifying unclassified objects.

Time-series classification has attracted much attention recently in the time-series community [34, 37, 45, 93, 111, 130, 131]. Górecki and Luczak [34] introduced a new distance function for the nearest neighbor rule based on the general shape of time series rather than point-to-point comparison. In [37] after reducing the dimensionality of long time series using matrix factorization, Support Vector Machines are applied to classify them. As an attempt to improve classification accuracy, Jeong et al. [45] presented a novel generalization of the Euclidean distance and Dynamic Time Warping (DTW) by applying a modified logistic weight function. Radovanović et al. [93] investigated an aspect of the *dimensionality curse*[5] called *hubness* (the tendency of some elements of a dataset to appear among the k nearest neighbors of unexpectedly many other elements of the dataset) on time-series classification. To speed up the classification of time series, Spiegel et al. [111] proposed a variation of the DTW distance using a greedy approach that only evaluates matrix elements which most likely contribute to the actual *warping path* (see Section 2.1.2). In order to produce compact classifiers and preserve good classification accuracy Xi et al. [130] combined constraining the warping window (see Section 2.2) of DTW with *numerosity reduction* (discarding part of the training set to improve performance). Ye and Keogh [131] introduced a new time series primitive called *time series shapelets*[6] to address some limitations of the simple nearest neighbor algorithm.

Approaches to classification vary from purely statistical methods such as exponential smoothing [15] or ARIMA models [12], to those based on different data-mining techniques like neural networks [81], genetic algorithms [23], support vector machines [128] and fuzzy systems [88].

Statistical methods usually use autoregressive (AR) models where the current value of time series is generated as a linear combination of the previous values. These methods are more appropriate for forecasting, but some interesting works can be found in the area of classification. Kini and Sekhar [52] present the large margin autoregressive (LMAR) method that uses an AR model for each class and the large margin method for estimation of parameters of AR models. A system which builds groups of time series that share the same forecasting model applied to supply chains is presented by Turrado García et al. [120]. The similarity between two time series is defined in the following manner: two series will have the same associated ARIMA model if and only if the autocorrelation and partial autocorrelation functions give similar results in their N first positions. In the paper [43] by Huan and Palaniappan the neural-network classification of autoregressive features is applied on time series of electroencephalogram signals extracted during mental tasks.

An interesting approach for time-series classification is to transform the time series into standard feature vectors with a fixed dimensionality. In this case, many well-studied data-mining techniques can then be adopted for classification or clustering. Zhang et al. proposed an algorithm based on the orthogonal wavelet transform [136], in which the coefficients of the Haar wavelet were extracted as a feature vector for subsequent time-series clustering. Eruhimov et al. use DTW to transform the time axis of each signal in order to decrease the Euclidean distance between signals from the same class [24]. Afterwards, a range of

[5] The curse of dimensionality refers to problems associated with high dimensionality (large number of features) of data being processed [91].

[6] Time series subsequences which are in some sense maximally representative of a class [131].

attributes of both transformed time series and original time series are extracted to form a high-dimensional feature vector. Another interesting approach is the adaptation of segment-based representations to extract features from time series [30, 65, 73]. Recent activities in transformation of time series into feature vectors include utilization of segment-base features [138] and the extraction of meaningful patterns from original data [137].

The most widely used approach to time-series classification is to define a distance function between two time series and use some of the existing distance-based classifiers. In this approach, the key problem is how to define a robust distance or similarity measure that can reflect the overall shape of the time series. The most standard distance measures are described in the previous section. Recently, an alignment-based distance metric called Time Warp Edit Distance (TWED) was proposed by Marteau [71]. It has been proven that this metric satisfies the triangle inequality. In [34], Górecki and Luczak emphasize the importance of using derivatives in time-series distance functions. This approach considers the overall shape of a time series rather than individual point-to-point function comparison. A generalization of the DTW measure is proposed by Jeong et al. [45] as a novel distance measure, called Weighted DTW (WDTW). This measure penalizes the points according to the phase difference between a reference point of the first time series and test point of the second time series. The proposed approach can prevent some bad alignments where one point of the first time series maps onto a large part of the second time series.

In this dissertation we adopt the aforementioned widely used approach to time-series classification of using distance measures between time series in conjunction with two existing distance-based classifiers – 1NN and kNN. As was mentioned in the introduction, this choice was motivated by reports of the efficiencies of different classification techniques where the simple nearest-neighbor classifier achieving among the best results [21, 49, 93, 119, 130]. The additional upside of 1NN is that its accuracy directly reflects the quality of the underlying distance measure [116], which provides a basis for our qualitative analysis of the impact of constraints, and also provides a practical demonstration of our observations.

3.1. The Nearest Neighbor Rule

A time series Q of length n can be viewed as a sequence $(q_1, q_2, \dots, q_n)$ of real numbers which describes the change of the observed phenomenon at equidistant points of time [16, 25, 135]. The task of classification is to determine the class (label) of an unclassified time series Q based on a given training set of pre-classified time series $T = \{(T_1, T_1^C), (T_2, T_2^C), \dots, (T_m, T_m^C)\}$ [25, 39, 63]. In the remainder of this section we denote the set of different classes assigned to the elements of the training set T with C, i.e.

$$C = \{T_1^C, T_2^C, \dots, T_m^C\}$$

where T_i^C denotes the class of time series T_i.

According to Ratanamahatana et al. [95] the simple nearest neighbor rule (1NN) [18] is one of the most popular time-series classification methods. The class of a new sequence is

determined by the closest (most similar) member of the training set. In [130], Xi et al. have shown that the combination of the 1NN rule and the DTW similarity measure is a very competitive classifier compared to other more complex methods. Cover and Hart [18] proved that the asymptotic misclassification rate R of the 1NN rule satisfies the following condition:

$$R^* \leq R \leq R^* \times \left(2 - R^* \times \frac{|C|}{|C|-1}\right)$$

where R^* is the Bayesian probability of error.

One of the first formulations and analyses of the k-nearest neighbor rule originates from Fix and Hodges [28]. Let C denote a set of two classes, i.e. $C = \{c_1, c_2\}$, and let the training set T contain n_i representatives of class c_i ($i = 1,2$). A new time series Q is labeled with class c_1 if $k_1/n_1 > k_2/n_2$, where k_i denotes the number of training examples of class c_i which are among the $k = k_1 + k_2$ nearest neighbors of Q. The intuition behind such a formulation of the k-nearest neighbor rule is that the dominance of the selected class among the nearest neighbors of Q must be consistent with the total number of its representatives in the training set T.

The majority-voting k-nearest neighbor rule (kNN) algorithm is a natural extension of the 1NN rule: a new series Q is labeled with the class that is most frequent among the k nearest neighbors of Q inside the training set T. The choice of the class can be formally written as follows:

$$Q^C = \arg\max_{c \in C} \left\{ \sum_{i=1}^{k} E\left(c = T_i^C\right) \right\} \tag{3.1}$$

where T_i^C denotes the class of the i-th nearest neighbor, and $E(\cdot)$ is an indicator function that returns 1 if its argument is true and 0 otherwise. Ties are broken arbitrarily.

Wu et al. [127] highlight a number of issues related to the choice of k in the kNN classifier. If k is too small, the kNN rule can be sensitive to noise points. If k is too large, the closest neighbors can include many different classes. Another issue may arise from the equality of the neighbors in the process of majority voting regardless of their distance from the query object. This can be addressed by weighting the votes of the neighbors in accordance to their distance. If we denote the weight of the i-th nearest neighbor with w_i, Eq. (3.1) can be adjusted in the following way:

$$Q^C = \arg\max_{c \in C} \left\{ \sum_{i=1}^{k} w_i \times E\left(c = T_i^C\right) \right\}$$

In the remainder of this section we will review several different ways of calculating weights that are analyzed in the experiments whose results are presented in Sections 5.4 and 5.5.

The pseudocode of the algorithm that implements a distance-weighted kNN classifier is given in Section 5.1.

3.1.1. Dudani's weighting functions

The first distance-weighted voting method for kNN rule was proposed by Dudani in [22] (henceforth denoted Dudani). In this approach weights are taken from the interval [0, 1]. The closest neighbor is weighted with 1, the farthest with 0 and the others are scaled between by the linear mapping defined in Eq. (3.2), where d_i denotes the distance between the query sequence Q and the i-th of the nearest neighbors.

$$w_i = \begin{cases} \dfrac{d_k - d_i}{d_k - d_1} & , d_k \neq d_1 \\ 1 & , d_k = d_1 \end{cases} \tag{3.2}$$

Dudani has shown that for at least one arbitrarily chosen example of a small training set (considering a three-class problem with equal a priori class probabilities) the distance-weighted kNN rule produced lower probability of error than the majority voting kNN rule [22]. The probability of error was estimated using a Monte Carlo analysis and the distance between neighbors was calculated using Euclidean metric. The obtained results are shown in Figure 3.1 (from [22]). We can see that the probability of error for Dudani's weighting scheme is lower than that for the majority voting rule. According to Dudani, the high value of probability of error in case of the majority rule for k=2 is a consequence of many ties [22].

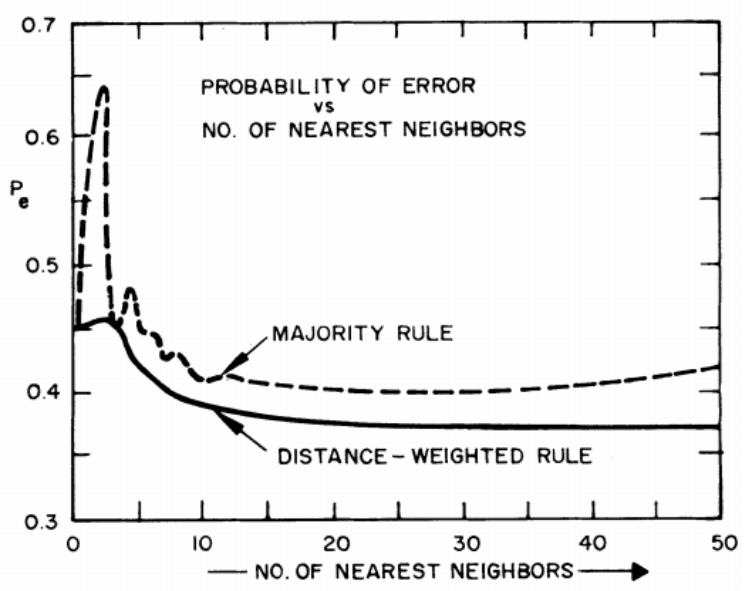


Figure 3.1. Plots of probability of error with respect to k for the distance-weighted and majority voting kNN rules

Slika 3.1. Verovatnoća greške u odnosu na parametar k u slučaju kNN klasifikatora sa i bez upotrebe težina

Dudani's belief is that the distance-weighted kNN rule is acceptable for small and moderate-sized training sets [22]. In his opinion, one of the reasons why the distance-weighted kNN rule achieves better results than the majority-voting k-nearest neighbor rule is that the probability of ties is lower in the first case.

Dudani [22] has also suggested two additional alternative weighting schemes: the inverse distance weight (Inverse, Eq. (3.3)) and the rank weight (Rank, Eq. (3.4)).

$$w_i = \frac{1}{d_i} \tag{3.3}$$

$$w_i = k - i + 1 \tag{3.4}$$

Instead of the inverse distance we may rely on the inverse of the squared distance [62, 74, 127] (ISquared). In both of these cases there is a possibility of division by zero. This is usually solved by adding a small constant ε (we have used 0.001 in our experiments, see Sections 5.4 and 5.5) to the denominator as in Eq. (3.5).

$$w_i = \frac{1}{d_i^2 + \varepsilon} \tag{3.5}$$

3.1.2. Macleod's weighting function

The weighting function in Eq. (3.2) excludes the k-th neighbor from the voting process in the situation when $d_k \neq d_1$ since $d_k - d_i = 0$ for $i = k$. Macleod et al. in [69] provide a generalization of Dudani's weighting function by introducing two new parameters: $s \geq k$ and $\alpha \geq 0$ (Macleod, Eq. (3.6)). Through them we can overcome this shortcoming. When $s = k$ and $\alpha = 0$, Eq. (3.6) becomes the original weighting scheme proposed by Dudani. From the several combinations of these parameters, which have been investigated in [69], we will use $s = k$ with $\alpha = 1$ (see Section 5.5).

$$w_i = \begin{cases} \dfrac{(d_s - d_i) + \alpha \times (d_s - d_1)}{(1 + \alpha) \times (d_s - d_1)} & , d_s \neq d_1 \\ 1 & , d_s = d_1 \end{cases} \tag{3.6}$$

3.1.3. The Fibonacci weighting function

In [84], Pao et al. have used the Fibonacci sequence as the weighting function (Fibonacci, Eq. (3.7)). They have compared this scheme with the three weighting methods defined by Dudani (linear mapping - Eq. (3.2), inverse distance - Eq. (3.3), and the rank weight - Eq. (3.4)) in the field of recognizing emotions from Mandarin speech. Beside the majority voting (kNN) and the distance-weighted k-nearest neighbor (WKNN) rule their study has also included two other variations of the weighted kNN classifier: *Categorical Average Patterns*

(WCAP) [115] and *Weighted Discrete* kNN (WDKNN) [83]. They have found that the Fibonacci weighting function outperforms the others in all of the examined classifiers.

$$\begin{gathered} w_i = w_{i+1} + w_{i+2} \\ w_k = w_{k-1} = 1 \end{gathered} \tag{3.7}$$

Pao et al. invited 18 males and 16 females to portray five different emotions (angry, bored, happy, neutral and sad). The obtained data were independently tagged by 10 human listeners and only those that had above 80% agreement of the taggers were chosen for the experiments. The resulting dataset included 151 angry, 83 bored, 96 happy, 116 neutral, and 124 sad recordings.

For the purposes of the experiments three acoustic features were extracted from the utterances: mel-frequency cepstral coefficients (MFCC), linear predictive coefficients (LPC) and linear predictive cepstral coefficients (LPCC). In the preprocessing stage, the data were normalized using min-max normalization [39, 62, 114].

In the first step of the experiments the best value of parameter k was selected in range from 1 to 15 using the Leave-One-Out (LOO) evaluation method (see Section 3.2) and the majority voting kNN classifier. The best accuracy (72.5%) was obtained for $k = 10$. Using this value of k, in the second phase of the examinations, the discussed weighting schemes in various classifiers were compared based on the results of the LOO evaluation method. According to the obtained accuracies in Table 3.1 (from [84]), the best results are achieved using the Fibonacci weighting function.

	Dudani	**Inverse**	**Rank**	**Fibonacci**
WKNN	75.6%	74.2%	73.8%	76.1%
WCAP	74.2%	73.6%	73.1%	74.5%
WDKNN	78.7%	79.5%	81.2%	81.4%

Table 3.1. Experimental results of different weighting functions in WKNN, WCAP, and WDKNN

Tabela 3.1. Rezultati eksperimenata sa različitim težinskim funkcijama za WKNN, WCAP i WDKNN

3.1.4. The uniform and dual-uniform weighting functions

Gou at al. [36] have introduced a weighting scheme calculated as the reciprocal value of the neighbors' rank (Uniform, Eq. (3.8)), and a weighting function (DualU, Eq. (3.9)) based on Dudani's linear mapping (Eq. (3.2)). They denote the weighted kNN classifiers that use these weighting schemes respectively with UWKNN, DWKNN and WKNN. In this paper they have compared the classification accuracies of the majority voting kNN classifier and these three weighted forms of the kNN classifier. Their conclusion is that the combined weighting (DualU) in Eq. (3.9) surpasses the other examined classifiers.

$$w_i = \frac{1}{i} \tag{3.8}$$

$$w_i = \begin{cases} \frac{d_k - d_i}{d_k - d_1} \times \frac{1}{i} & , d_k \neq d_1 \\ 1 & , d_k = d_1 \end{cases} \tag{3.9}$$

The experiments were conducted using artificially generated data and real data selected from the UCI machine learning repository [7] with numeric attributes only. The distances between the elements of the datasets were calculated using Euclidean distance. The optimal value of the neighborhood size k (which minimizes the error rate) was selected in range from 1 to 50 based on the Leave-One-Out (LOO) method (see Section 3.2).

In the first part of the examinations the experiments were conducted on artificial data. In the first step a dataset with 600 elements was used and the neighborhood size k was varied in range from 1 to 50. The influence of the neighborhood size k on the classification accuracies are presented in Figure 3.2 (from [36]).

In the second step the value of the parameter k was set to 15 and the size of the dataset was varied in range from 100 to 2000 in steps of 100. The influence of the sample size on the classification accuracies are depicted in Figure 3.3 (from [36]).

These results suggest that the DualU weighting function (DWKNN classifier) is better than the other discussed weighting schemes in both cases. The majority voting kNN classifier is inferior to the other examined classifiers within both experiments.

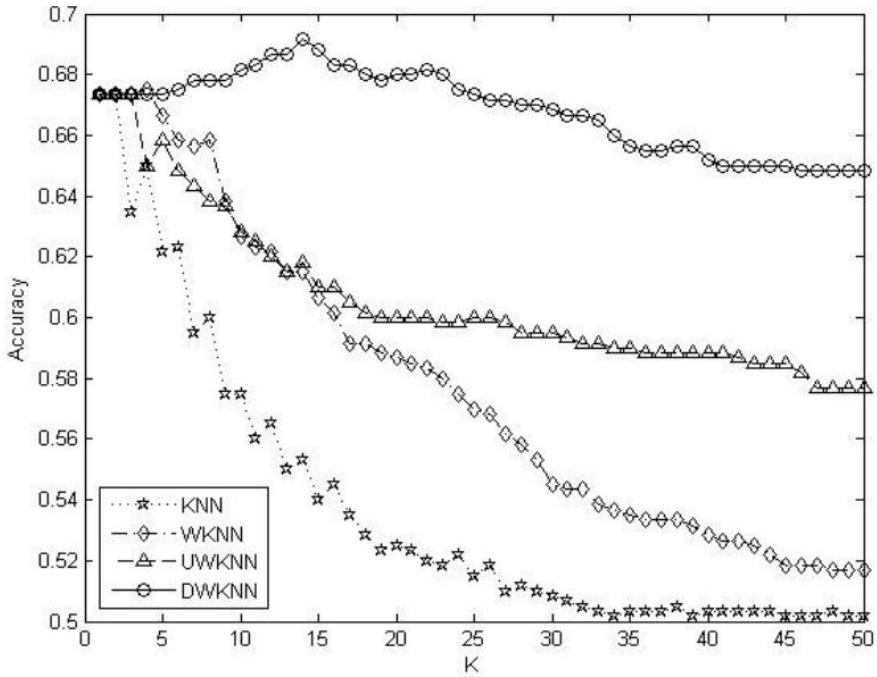


Figure 3.2. The influence of the neighborhood size k on the classification accuracies

Slika 3.2. Uticaj broja suseda k na tačnost klasifikacije

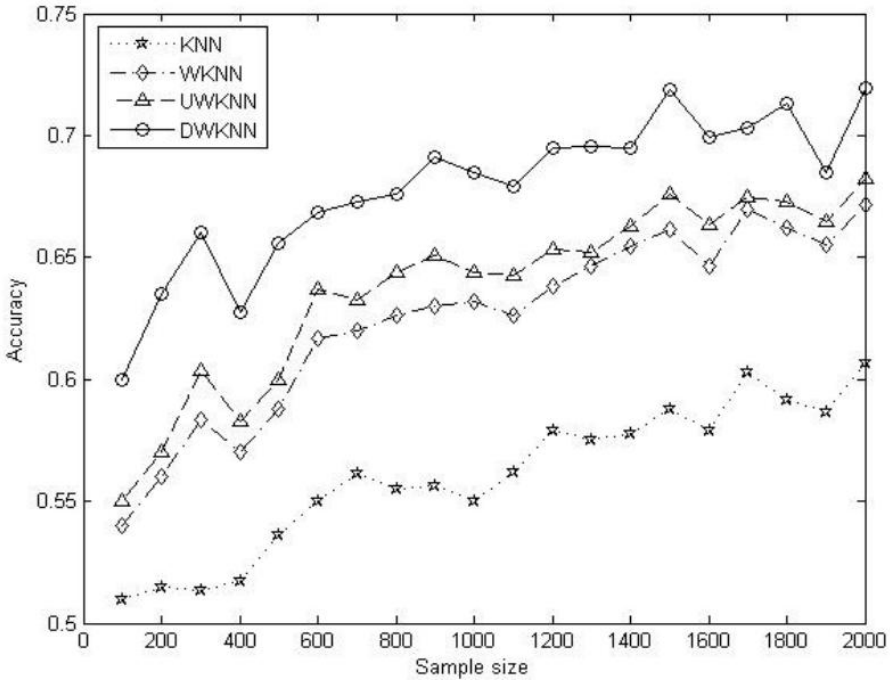


Figure 3.3. The influence of the sample size on the classification accuracies

Slika 3.3. Uticaj veličine skupa podataka na tačnost klasifikacije

In the second part of the examinations Gou et al. have compared the classification performance of the discussed classifiers on 12 real datasets from the UCI machine learning repository [7]. The basic characteristics of these datasets are presented in Table 3.2 (from [36]).

Dataset	Attributes	Instances	Classes
Pendigits	16	10992	10
Oppdigits	64	5620	10
Ionosphere	34	351	2
Glass	10	214	7
Landsat Satellite	36	6435	7
Libras Movement	90	360	15
Wine Quality-Red	11	1599	11
Zoo	17	101	7
Vehicle	18	946	4
Wine Quality-White	11	4898	11
Letter	16	20000	26
Image Segmentation	19	2310	7

Table 3.2. Some characteristics of the UCI datasets: the number of instances, attributes, and classes

Tabela 3.2. Neke osobine UCI skupova podataka: broj elemenata, atributa i klasa

The obtained lowest error rates with the corresponding optimal values of the parameter k are shown in Table 3.3 (from [36]). Of the 12 datasets in case of 7 datasets the smallest error rate was achieved by the DualU weighting function (i.e. the DWKNN classifier). Furthermore, the average error rate of DWKNN is the lowest among the examined classifiers.

Another important observation is that the optimal value of parameter k is low (less than 10) for every dataset in case of the kNN, WKNN and UWKNN classifiers (for the majority of the datasets this value is 1). On the other hand, the optimal value of k is much larger in case of the DWKNN classifier (larger than 5 - except for the Zoo dataset). Additionally, Gou at al. have shown that while the classification accuracy for kNN, WKNN and UWKNN drops quickly as the value of k grows, it remains stable for DWKNN or even increases for some datasets. In Sections 5.4 and 5.5 we have noticed some similar traits in the domain of time series.

Datasets	kNN	WKNN	UWKNN	DWKNN
Pendigits	0.58 (3)	**0.53 (6)**	0.55 (4)	0.55 (22)
Oppdigits	1.00 (4)	**0.96 (7)**	1.01 (8)	1.03 (32)
Ionosphere	13.39 (1)	13.39 (1)	13.39 (1)	**12.54 (14)**
Glass	26.64 (1)	26.64 (1)	26.64 (1)	**25.23 (8)**
Landsat Satellite	8.44 (4)	8.45 (7)	**8.34 (6)**	**8.34 (6)**
Libras Movement	12.78 (1)	12.78 (1)	12.78 (1)	**12.50 (6)**
Wine Quality-Red	38.46 (1)	38.15 (6)	38.46 (1)	**36.52 (44)**
Zoo	**1.98 (1)**	**1.98 (1)**	**1.98 (1)**	**1.98 (1)**
Vehicle	33.45 (5)	**32.74 (6)**	33.45 (7)	33.92 (30)
Wine Quality-White	38.38 (1)	38.36 (4)	38.38 (1)	**38.10 (17)**
Letter	3.64 (4)	**3.29 (8)**	3.39 (7)	3.32 (50)
Image Segmentation	3.33 (1)	**3.12 (6)**	3.33 (1)	3.29 (9)
Average Error	15.17	15.03	15.14	**14.78**

Table 3.3. The lowest error (%) of each method with the corresponding k in the parenthesis for all datasets

Tabela 3.3. Najmanje greške klasifikacije (%) zajedno sa odgovarajućim vrednostima parametra k u zagradi, za sve skupove podataka

3.1.5. Zavrel's weighting function

In [133] Zavrel has matched another weighting scheme against the linear mapping (Eq. (3.2)) and the inverse distance (Eq. (3.3)) weighting functions proposed by Dudani, as well as against the 1NN classifier and the majority voting rule. This scheme (Zavrel) is based on the exponential function as shown in Eq. (3.10) where α and β are constants determining the slope and the power of the exponential decay function respectively. In our inquiry in Section 5.5 we have selected $\alpha = \beta = 1$ as proposed in the Zavrel's paper.

$$w_i = e^{-\alpha d_i^{\beta}} \tag{3.10}$$

The survey covered 12 datasets from the UCI machine learning repository [7] and one additional linguistic dataset (*PP-attachment*) collected by Ratnaparkhi et al. [100]. The properties of the observed datasets are shown in Table 3.4 (from [133]). All the selected datasets have only numeric attributes.

In his experiments Zavrel relied on cosine distance [20]. Each experiment was repeated 10 times (10CV), unless a specific partition or separate dataset was used for testing in earlier work on the same dataset [133]. In case of the repeated experiments the datasets were randomly shuffled and the first 90% of the instances were used for training and the rest for testing. The optimal number of neighbors k was searched in the interval from 1 to 100.

Dataset	Attributes	Instances	Classes	Error estimation
PP-attachment	100	20801	2	test set, 3097 inst.
Glass	9	214	6	10CV
Wine	13	178	3	10CV
Sonar	60	208	2	10CV
Letter	16	16000 train	26	test set. 4000 inst.
Isolet	617	6238 train	26	test set, 1559 inst.
Vowel	10	517 train	11	test set, 473 inst.
Segmentation	19	210 train	7	test set, 2100 inst.
Ionosphere	34	351	2	10CV
Diabetes (pima)	8	768	2	10CV
Breast cancer (wpbc)	32	198	2	10CV
Breast cancer (wdbc)	30	569	2	10CV
Cleveland heart	13	303	5	10CV

Table 3.4. Characteristics of the UCI datasets used in Zavrel's experiments

Tabela 3.4. Osobine UCI skupova podataka korišćenim u Zavrelovim eksperimentima

Zavrel has found that the weighted voting can improve the kNN's accuracy and that Dudani's linear mapping (Eq. (3.2)) is superior to the other classifiers examined in his study. The detailed results are presented in Table 3.5. The optimal values of the parameter k are listed within the brackets. The dashes indicate that there is no improvement over 1NN classifier's accuracy.

Based on these results, we can see that the 1NN classifier is superior to the other classifiers only in case of one dataset (*Segmentation*). Furthermore, Zavrel has found that after reaching the optimal value of k, accuracy decreases almost monotonically with the increase of k. His conclusion is that a weighting function for the kNN classifier has the potential to make it more robust to the choice of k, and to improve its accuracy [133]. Based on the results of our extensive experiments, which are presented in Sections 5.4 and 5.5, we have discovered similar findings in the field of time-series data mining.

Dataset	1NN	kNN	Inverse	Dudani	Zavrel
PP-attachment	80.1	83.4 (13)	83.7 (13)	**84.2 (30)**	84.0 (35)
Glass	76.4	**77.3 (2)**	-	**77.3 (5)**	76.8 (3)
Wine	96.7	**97.8 (3)**	**97.8 (3)**	**97.8 (7)**	**97.8 (3)**
Sonar	82.5	-	83.1 (7)	**85.0 (9)**	-
Letter	95.6	-	-	**96.0 (5)**	-
Isolet	88.6	91.9 (13)	92.4 (13)	**92.9 (15)**	92.4 (3)
Vowel	52.6	-	55.6 (7)	**55.8 (15)**	55.0 (7)
Segmentation	**90.9**	-	-	-	-
Ionosphere	90.0	-	-	**90.6 (5)**	-
Diabetes (pima)	66.1	70.1 (3)	69.7 (80)	**70.3 (100)**	70.1 (3)
Breast cancer (wpbc)	69.0	79.5 (11)	**81.0 (9)**	79.5 (21)	79.5 (11)
Breast cancer (wdbc)	89.1	**93.3 (5)**	93.2 (5)	92.8 (30)	**93.3 (5)**
Cleveland heart	57.0	**62.7 (2)**	58.7 (11)	58.7 (9)	59.3 (100)

Table 3.5. Results of Zavrel's experiments

Tabela 3.5. Rezultati eksperimenata Zavrel-a

3.1.6. The dual distance-weighted function

The dual distance-weighted function (DualD) depicted with Eq. (3.11) was presented by Gou et al. in [35]. This novel weighted kNN rule extends Dudani's linear mapping (Eq. (3.2)): it weights the closest and the farthest neighbors the same way as the linear mapping, but assigns smaller values to those between them. This extension of the Dudani's scheme was proposed as an attempt to improve the performance of the distance-weighted kNN rule in case when outliers are present among the data and in case of datasets with imbalanced class distribution (when the number of representatives of some classes is significantly higher than the number of instances of other classes).

$$w_i = \begin{cases} \dfrac{d_k - d_i}{d_k - d_1} \times \dfrac{d_k + d_1}{d_k + d_i} & , d_k \neq d_1 \\ 1 & , d_k = d_1 \end{cases} \tag{3.11}$$

The authors have compared the classification accuracy of the newly proposed weighting scheme with the accuracies of the 1NN, the kNN and Dudani's linear-mapping based weighted kNN rule. The dual distance-weighted function has performed better with each of the 12 sets from the UCI machine learning repository [7] which were used in the experiments. The analyzed datasets are described in Table 3.6. Table 3.7 shows the results (average accuracies with the standard deviations) of the comparison. The optimal values of parameter k are given in parentheses. Both of these tables are taken from [35].

Dataset	Attributes	Instances	Classes	Training samples	Testing samples
Glass	10	214	7	140	74
Wine	13	178	3	100	78
Sonar	60	208	2	120	88
Parkin	22	195	2	120	75
Iono	34	351	2	200	151
Musk	166	476	2	276	200
Vehicle	18	846	4	500	346
Image	19	2310	7	1310	1000
Cardio	21	2126	10	1126	1000
Opt	64	5620	10	3000	2620
Landsat	36	6435	7	3435	3000
Letter	16	20000	26	10000	10000

Table 3.6. Characteristics of the UCI datasets used in the experiments with the DualD weighting scheme

Tabela 3.6. Osobine UCI skupova podataka korišćenih u eksperimentima sa DualD šemom težina

Dataset	1NN	kNN	Dudani	DualD
Glass	69.86 ±1.35	69.86 ±1.35(1)	69.86 ±1.56 (1)	**70.14 ±1.35 (5)**
Wine	71.15 ±1.81	71.15 ±1.81(1)	71.47 ±1.54 (4)	**71.99 ±1.38 (4)**
Sonar	80.62 ±1.62	80.62 ±1.62 (1)	81.59 ±1.81 (4)	**82.05 ±1.81 (5)**
Parkin	82.67 ±2.05	83.00 ±1.80 (4)	83.53 ±1.65 (8)	**83.93 ±1.86 (8)**
Iono	84.01 ±1.36	84.01 ±1.36 (1)	84.27 ±1.24 (7)	**84.44 ±1.24 (8)**
Musk	83.98 ±0.94	83.98 ±1.06 (1)	84.77 ±0.87 (6)	**85.10 ±0.94 (7)**
Vehicle	63.16 ±0.79	63.76 ±1.24 (3)	63.96 ±1.02 (8)	**64.34 ±1.17 (9)**
Image	95.19 ±0.31	95.19 ±0.31 (1)	95.19 ±0.33 (1)	**95.21 ±0.31 (4)**
Cardio	69.84 ±0.49	69.84 ±0.49 (1)	70.12 ±0.48 (5)	**70.30 ±0.45 (6)**
Opt	98.43 ±0.12	98.52 ±0.12 (4)	98.64 ±0.13 (7)	**98.65 ±0.14 (7)**
Landsat	89.89 ±0.21	90.35 ±0.27 (4)	90.63 ±0.25 (10)	**90.65 ±0.26 (11)**
Letter	94.34 ±0.086	94.38 ±0.10 (4)	94.89 ±0.092 (9)	**94.93 ±0.088 (9)**

Table 3.7. Results of the experiments with the DualD weighting scheme

Tabela 3.7. Rezultati eksperimenata sa DualD šemom težina

The experiments were conducted using 20-fold cross. The elements of the datasets were compared using Euclidean distance and the neighbors number (k) varied from 1 to 15.

Based on the obtained results Gou et al. concluded that the DualD weighting scheme is superior to 1NN, kNN and to Dudani's weighting function on all the examined datasets. In addition it can be seen that their novel weighting method needs more nearest neighbors to achieve the best classification performance compared to the other studied methods (Table 3.7). The results of our examinations in Section 5.5 showed that the DualD is one of

the best-performing weighting schemes when it comes to working with time series too, but it isn't absolutely superior (especially not in relation to Dudani's weighting scheme and the DualU weighting function).

Based on a detailed analysis of the results obtained for individual datasets Gou et al. showed that their weighting scheme along with Dudani's weighting function mostly outperforms kNN with the different values of k. This can be clearly seen in Figure 3.4 (from [35]) which depicts the average accuracies for different values of parameter k in case of three of the 12 examined datasets. In Section 5.4 we will show through extensive experiments that this applies to the field of time-series data mining, too: the introduction of weights improves classification accuracy of the kNN classifier over the 1NN and the majority voting kNN classifiers. Furthermore, we will demonstrate that the use of weights moderates the growth of the classification's error rate and the warping window's width as we increase the value of k.

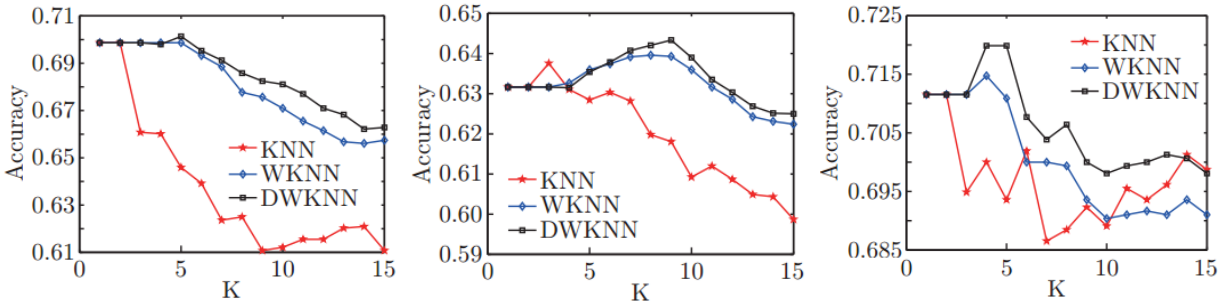


Figure 3.4. Average accuracies for different neighborhood size k

Slika 3.4. Prosečne tačnosti klasifikacije za različite vrednosti parametra k

3.2. Classifier Evaluation Methods

The performance of a classifier can be measured by counting the number of correctly and incorrectly classified test objects. The *accuracy* of a classifier is defined as the ratio of test objects that are correctly classified (Eq. (3.12)), and the *error rate* is defined as the ratio of misclassified test objects (Eq. (3.13)). The relationship between these two *performance metrics* is very simple: $error\ rate = 1 - accuracy$. More detailed descriptions of these and other performance metrics are given in [1], [5], [39], and [116].

$$accuracy = \frac{|correctly\ classified\ objects|}{|test\ set|} \tag{3.12}$$

$$error\ rate = \frac{|misclassified\ objects|}{|test\ set|} \tag{3.13}$$

In the remainder of this section we provide an overview of the most popular partitioning techniques used to divide the initial set of labeled objects into training and test sets. These methods were used in the experiments whose results are presented in Chapter 5 of this dissertation. Additional information about these techniques can be found in [1], [5], [39], and [116].

Holdout [1, 39, 116]. The idea behind the holdout method is to randomly partition the original set of labeled data into two disjoint subsets. One of the subsets is used for training and the other one for testing. The size of the training set and the test set is selected by the analysts (for example, 50%-50%, or two-thirds for training and one-third for testing). In order to improve the evaluation of the classifier's accuracy, the holdout method can be repeated several times. This expansion of the holdout method is called *random subsampling*. The accuracy is calculated as the average value of the iterations' accuracies.

Cross-Validation [1, 5, 39, 74, 78, 116]. In the first step of this approach, the original set of labeled objects is randomly divided into k disjoint sets of approximately equal size. Next, the testing is performed through k iterations: within each iteration, we choose one (not yet selected) subset for testing, and the union of all other subsets for training. In this way, each object is tested exactly once and is used $k-1$ times for training. The classification error is calculated as the average of the errors obtained from each iteration.

This evaluation method is called *k-fold cross-validation*. A technique called *n runs of k-fold cross-validation* is obtained by repeating the whole process n times. 10 runs of 10-fold cross-validation (SCV10x10) is commonly used in data-mining software - there are indications that this configuration gives a good estimate of the actual accuracy of the classifier [1, 39]. Other typical forms of cross-validation include: 5 runs of 2-fold cross-validation (SCV5x2) [5] and 9-fold cross-validation (SCV1x9) [91].

Depending on how is the initial dataset partitioned into disjoint subsets, we can distinguish between two types of cross-validation. If we do not take care of the equal distribution of different classes in the individual subsets, we get *random-split cross-validation*. On the other hand, if the different classes are more or less equally distributed in the subsets, approximately in the same way as in the original dataset, we get *stratified cross-validation*.

Ding et al. [21] applied the following variant of cross-validation: from the k disjoint subsets one is selected for training and the union of the other subsets is used for testing. We have used the same approach In the process of validating our implementation of similarity measures (Section 4.3).

Leave-One-Out [1, 5, 39, 78, 116]. The leave-one-out (LOO) method represents an extreme case of k-fold cross validation where the number of folds is equal to the number of objects in the initial dataset, i.e. $k = |D|$, where D denotes the original set. In this approach, each test set contains only one object. The advantage of this approach is that it utilizes as much data as possible for training. However, there are some drawbacks, too: it is computationally expensive and the testing sets are not representative (since each test set contains only one object) [1, 116].

Chapter 4

Methods, Tools and Datasets

The fourth chapter is devoted to the review of methods, tools and datasets used in the experiments whose results are described in Chapter 5. All of the examinations within this dissertation were performed by relying solely on our *Framework for Analysis and Prediction* (FAP). The basic features of this free and open source library are presented in Section 4.1, while the details of its structure and implementation are described in Chapter 6.

The characteristics of the datasets used in our studies are the subject of Section 4.2. This section provides also an explanation of how are the values of the matching threshold ε selected for the elastic similarity measures LCS (Section 2.1.3) and EDR (Section 2.1.5). Details on the validation of implemented similarity measures are given in Section 4.3.

The experiments within our studies were rather long-lasting and computationally demanding. To speed up their execution we have calculated the distances between time series in advance and saved the obtained values in form of distance matrices. This procedure is described in Section 4.4.

4.1. Framework for Analysis and Prediction (FAP)

Time-series data are generated and utilized in many domains of science, economics and medicine, and the amount of data from which we need to extract valuable information is continuously increasing. This has led to a growing interest of studying different fields of temporal data analysis [13, 19, 21, 25, 39, 95]. We can distinguish between two distinct fields of time series research [19, 63]: statistical analysis and modeling on one side, and the data mining and machine learning approach on the other side.

According to Das and Gunopulos [19] the statistical approach is mainly interested in identifying patterns, trend analysis, seasonality and forecasting. On the other hand, temporal data mining is focused on database management and on research tasks like indexing, classification, clustering, prediction, data representation and similarity measurement [21, 25, 63]. Laxman and Sastry [63] highlight two major differences between statistical analysis and data mining: data mining methods must be capable for efficient processing of a much larger quantity of temporal data, and their scope extends beyond standard time series analysis.

Ratanamahatana et al. [95] and Das and Gunopulos [19] emphasize that the methods studied by statisticians are of little furtherance for researchers in the field of time-series

data mining. As a consequence, it has become increasingly important to develop new methodologies for different task types of temporal data mining and to investigate and enhance the existing ones. However, the new solutions are usually separately implemented and described in different publications. In addition, as we have seen in Section 3.1, for many newly-introduced techniques dominance was claimed over some previous methods - often based only on a very limited number of case studies. This trend can be noticed in other areas of temporal data mining, too [21].

Motivated by these considerations we have developed a free and open source, multipurpose and multifunctional library for researchers and practitioners interested in time-series data mining. Our Framework for Analysis and Prediction is written in Java and it is designed to be a free and extensible software package which will cover all of the main tasks of temporal data mining and analysis. The intention behind our framework is to support and alleviate the investigation and comparison of different techniques utilized in this domain.

In its current version, beside the implementation of the most commonly used similarity measures described in Section 2.1, FAP contains several others including different forms of the L_p norm, Time Warp Edit Distance (TWED) [71], Spline [57], Swale [79] and the Canberra distance [53]. The elastic similarity measures (DTW, LCS, ERP, EDR, TWED) are implemented in three ways: without constraining the warping window, using the Sakoe-Chiba band [105], and employing the Itakura parallelogram [44]. Along with the nearest neighbor rule (1NN) and the majority voting NN rule (kNN), our library incorporates all of the different weighting schemes outlined in Section 3.1. Among the methods for testing the accuracy of classifiers, FAP currently supports stratified k-fold cross validation (SCV), leave-one-out (LOO) and the holdout method. There are also several classes implementing pre-processing transformations including scaling, shifting, min-max normalization [39, 62, 114], z-score normalization [39, 114], decimal scaling [39, 114], and linear equiscaling. Various time-series representations are also supported: Piecewise Linear Approximation (PLA) [51], Piecewise Aggregate Approximation (PAA) [47], Adaptive Piecewise Constant Approximation (APCA) [48], Symbolic Aggregate Approximation (SAX) [67] and Spline [57].

The Framework for Analysis and Prediction has already been successfully employed within various research domains including: developing a distributed distance matrix generator based on agents [75, 76], mining time series in the psychological domain [55, 56] and time-series analysis in the neurology domain [61]. FAP might be applied in other domains too, for example, signal processing [112] or image processing [109].

Details of the implementation and the general structure of the Framework for Analysis and Prediction are given in Chapter 6.

4.2. Datasets used in the Experimental Evaluation

The experiments performed within the research of this dissertation are executed on 46 datasets from University of California, Riverside (UCR) Time Series Repository [50]

(Table 4.1), which includes the majority of all publicly available, labeled time-series datasets in the world. This collection of datasets is most commonly used for validation of different time-series mining concepts. They originate from a plethora of different domains, including medicine, robotics, astronomy, biology, face recognition, handwriting recognition, etc. The length of time series varies from 24 to 1882 (column *L* in Table 4.1), depending of the dataset. The number of time series per dataset varies from 56 to 9236 (column S in Table 4.1) and the number of classes varies from 2 to 50 (column $|C|$ in Table 4.1). Column *ID* in Table 4.1 contains the labels assigned to datasets. These labels are used for dataset identification in tables with detailed results in appendices of this dissertation. In these tables there is not enough space for the names of the datasets and they are represented by these identification numbers.

ID	Dataset	S	L	|C|	LCS ε	EDR ε	ID	Dataset	S	L	|C|	LCS ε	EDR ε
1	**50words**	905	270	50	0.379296	0.299444	**24**	**mallat**	2400	1024	8	0.419795	0.419795
2	**adiac**	781	176	37	0.039886	0.039886	**25**	**medicalimages**	1141	99	10	0.15919	0.099494
3	**beef**	60	470	5	0.013361	0.018077	**26**	**motes**	1272	84	2	0.417493	0.298209
4	**car**	120	577	4	0.079931	0.079931	**27**	**noninvasivefatalecg_thorax1**	3765	750	42	0.23984	0.23984
5	**cbf**	930	128	3	0.099609	0.358591	**28**	**noninvasivefatalecg_thorax2**	3765	750	42	0.17988	0.17988
6	**chlorineconcentration**	4307	166	3	0.119638	0.099698	**29**	**oliveoil**	60	570	4	0.006686	0.006686
7	**cinc_ecg_torso**	1420	1639	4	0.779762	0.799756	**30**	**osuleaf**	442	427	6	0.099991	0.119989
8	**coffee**	56	286	2	2.292314	2.292314	**31**	**plane**	210	144	7	0.061007	0.040671
9	**cricket_x**	780	300	12	0.446341	0.492515	**32**	**sonyaiborobotsurface**	621	70	2	0.814122	0.456702
10	**cricket_y**	780	300	12	0.469044	0.549914	**33**	**sonyaiborobotsurfaceii**	980	65	2	0.853359	0.853359
11	**cricket_z**	780	300	12	0.430912	0.415522	**34**	**starlightcurves**	9236	1024	3	0.03998	0.159922
12	**diatomsizereduction**	322	345	4	0.039942	0.039942	**35**	**swedishleaf**	1125	128	15	0.239061	0.199217
13	**ecg200**	200	96	2	0.318091	0.377733	**36**	**symbols**	1020	398	6	0.03995	0.818969
14	**ecgfivedays**	884	136	2	0.876759	0.617716	**37**	**synthetic_control**	600	60	6	0.634644	0.932134
15	**faceall**	2250	131	14	0.597706	0.458241	**38**	**trace**	200	275	4	0.159709	0.179672
16	**facefour**	112	350	4	0.179743	0.059914	**39**	**twoleadecg**	1162	82	2	0.178899	0.159021
17	**fish**	350	463	7	0.060073	0.060073	**40**	**twopatterns**	5000	128	4	0.159374	0.219139
18	**gun_point**	200	150	2	0.119599	0.119599	**41**	**uwavegesturelibrary_x**	4478	315	8	0.619015	0.539142
19	**haptics**	463	1092	5	0.159927	0.159927	**42**	**uwavegesturelibrary_y**	4478	315	8	0.698888	0.299523
20	**inlineskate**	650	1882	7	0.219942	0.039989	**43**	**uwavegesturelibrary_z**	4478	315	8	0.359428	0.139778
21	**italypowerdemand**	1096	24	2	0.332841	0.35242	**44**	**wafer**	7164	152	2	0.259143	0.219275
22	**lighting2**	121	637	2	0.239812	0.439654	**45**	**wordssynonyms**	905	270	25	0.379296	0.618851
23	**lighting7**	143	319	7	0.299529	0.339467	**46**	**yoga**	3300	426	2	0.039953	0.05993

Table 4.1. Characteristics of the UCR datasets used in the experiments

Tabela 4.1. Osobine UCR skupova podataka korišćenih u eksperimentima

In the case of LCS and EDR the similarity also depends on the matching threshold ε (see Sections 2.1.3 and 2.1.5). Let $StDev$ denote the standard deviation of a particular dataset as in [21]. The value of parameter ε was determined by calculating the classification error of the 1NN classifier using the LOO evaluation method and the unconstrained similarity measures. We have selected the smallest value in range from $0.02 \times StDev$ to $StDev$ (with steps of $0.02 \times StDev$) which gave the best classification accuracy. For smaller datasets this

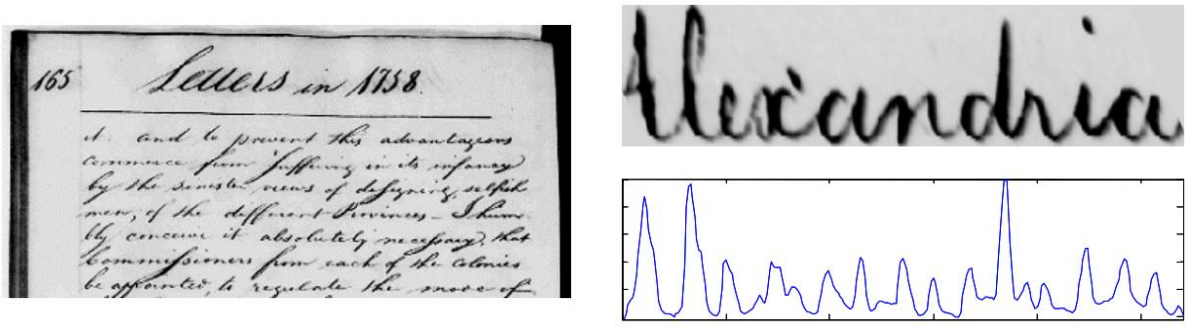
search was carried out on the entire dataset. In case of larger sets (which contain more than 1400 time series) it was performed on a stratified subset of the size not less than 400 elements. The obtained values of ε are presented in the corresponding columns of Table 4.1.

To illustrate the diversity of the datasets presented in Table 4.1 and used in this dissertation, in the remainder of this section we will give a brief overview of a representative selection.

50words. This is a subset of the *Word Spotting* dataset created by Rath and Manmatha [99] for word image matching. This dataset was extracted from the collections of handwritten manuscripts written by George Washington. The original dataset contains 2381 word images from 10 handwritten pages. From each word image four features are extracted and combined into a four-dimensional time series which describes the profile of the image [99]:

- **Projection Profile** - capture the distribution of ink along one of the two dimensions in a word image (*feature 1*).
- **Word Profiles** - the upper and lower word profiles capture part of the outlining shape of a word (*features 2 and 3*). A time series created based on the upper profile of the word "Alexandria" is shown in Figure 4.1 (from [124]).
- **Background/Ink Transitions** - capture the number of background to ink transitions (*feature 4*).

The *50words* dataset contains images of 50 common words from the original dataset (such as "the", "and", etc.) and is limited to the first dimension of each image [124].



(a) A sample of text written by George Washington.

(b) The word "Alexandria" and a time series created based on the upper profile of the word.

Figure 4.1. A sample of text written by George Washington and the upper profile of the word "Alexandria"
Slika 4.1. Uzorak teksta napisanog od strane Džordža Vašingtona i gornji profil reči "Alexandria"

Facefour. The *face* (four) dataset was introduced by Ratanamahatana and Keogh [96] within the context of a face classification problem based using head profiles. Four different persons (one female and three males) were photographed while they were making different expressions on their faces (talking, smiling, frowning, laughing, etc.). Starting from the neck

area, each head profile was converted into a time series by measuring the local angle of a trace of its perimeter. This procedure is depicted in Figure 4.2 (from [96]).

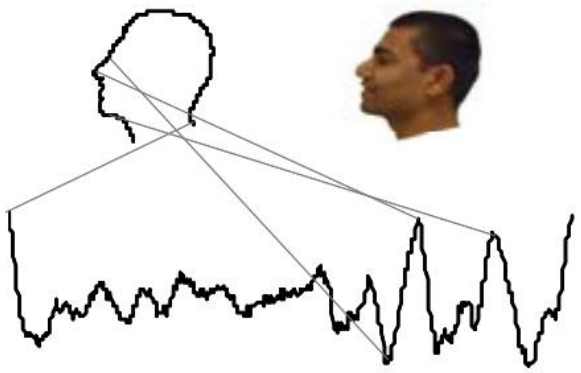


Figure 4.2. Converting a head profile into time series

Slika 4.2. Pretvaranje profila glave u vremensku seriju

Gun_Point. This dataset which originates from the video surveillance domain was presented by Ratanamahatana and Keogh in [96] and [97]. The time series of the Gun_Point dataset were created with the aid of one female and one male actor. It includes 100 instances of the following two classes:

- **Gun-Draw** - The hands of the actors are by their sides. They draw a gun from a holster mounted to their hips and point it at a target for about one second. After that, they return the gun to the holster, and their hands to their sides.
- **Point** - The hands of the actors are by their sides. After pointing at a target for about one second with their index finger, they return their hands to their sides.

The process of converting the above described movements into time series is illustrated in Figure 4.3 (from [96]).

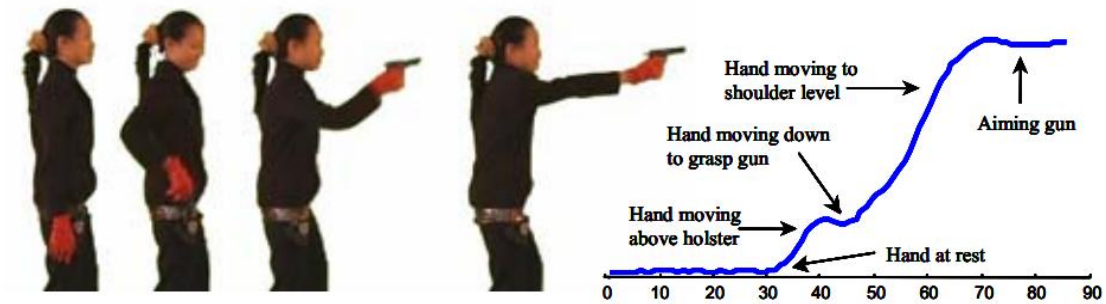


Figure 4.3. Representing a movement with a time series

Slika 4.3. Predstavljanje pokreta pomoću vremenske serije

Trace. The *trace* dataset is a simplified subset of the Transient Classification Benchmark presented by Davide Roverso [104]. This is an artificial dataset intended to simulate

instrumentation failures in a nuclear power plant. The original dataset contains 800 instances with 4 features (16 classes, 50 instances in each class). The trace dataset uses only the second feature of classes 2 and 6, and the third feature of classes 3 and 7 [96, 97].

Wafer. The *wafer* dataset was introduced by Robert T. Olszewski in his doctoral dissertation [82]. It represents a collection of measurements recorded from various sensors during the processing of silicon wafers for semiconductor fabrication. The wafers are grouped into two classes: normal and abnormal. The abnormal wafers embody different problems commonly encountered during semiconductor manufacturing [124].

4.3. FAP Validation

In the first stage of the development of our framework we have focused on implementing the most commonly used time-series similarity measures (Euclidean distance, DTW, LCS, ERP, EDR and others). Similarity measures embody a substantial element of many data mining tasks, including: classification, clustering, prediction, anomaly detection, and others. For this reason they constitute one of the most crucial components of our library. All measures are very carefully implemented with respect to efficiency and memory consumption.

In order to validate the correctness of the implementation of similarity measures within our Framework for Analysis and Prediction (FAP), we have measured the classification accuracy of 1NN for a number of datasets and we have compared the obtained results with the results presented in [21]. To evaluate the correctness of our implementation, we have conducted the cross-validation algorithm proposed in [21]: the one nearest neighbor (1NN) classifier is used on labeled data to evaluate the efficiency of the similarity/distance measures. Each time series in a dataset has a correct class label and the classifier tries to predict that label as the label of its nearest neighbor in the training set. There are several advantages with this approach:

- The underlying distance metric is crucial to the performance of the 1NN classifier; therefore, the accuracy of the 1NN classifier directly reflects the effectiveness of the similarity measure [116].

- 1NN is a very simple and parameter-free classifier. This reduces the possibility of appearance of errors in its implementation.

- Among many other classification techniques, such as decision trees, neural networks, Bayesian networks, support vector machines, etc., some of the best results in time-series classification are obtained using simple nearest neighbor methods [130].

For the sake of evaluating the effectiveness of each similarity measure, the following cross-validation algorithm has been applied [21]. First, a stratified split has been used to divide

the input dataset into k subsets (folds). The number of folds in the cross-validation algorithm, k, is shown in Table 4.2. These values are taken from [21].

The cross validation algorithm is applied as follows: using one subset at a time for the training set of the 1NN classifier, and the other $k-1$ subsets as the test set. If the similarity measure requires parameter tuning, the training set is divided into two equal-sized stratified subsets and one of them is used for parameter tuning. Finally, the average error rate of 1NN classification over the k-folds is reported in Table 4.2.

The experiments were conducted using the described methodology, on 20 diverse time series datasets. The data is provided by the UCR Time Series Repository [50], which includes the majority of all publicly available, labeled time series datasets in the world. The average error rates of the similarity measures on each dataset are shown in Table 4.2. We have compared our results with the results presented in [21] in order to verify the correctness of our implementation of similarity measures. The only differences appear at the second or third decimal place which is the consequence of randomization in the stratified random split of the cross-validation algorithm. These facts strongly support the correctness of our implementation, which has been our main goal.

Dataset	Number of folds	L_1	L_2	L_∞	$L_{1/2}$	DTW	CDTW	ERP	EDR	LCS	CLCS	Swale
50words	5	0,387	0,421	0,558	0,377	0,367	0,302	0,399	0,260	0,270	0,286	0,270
Adiac	5	0,486	0,462	0,429	0,510	0,457	0,438	0,439	0,437	0,435	0,431	0,435
Beef	2	0,550	0,517	0,517	0,533	0,550	0,517	0,550	0,600	0,500	0,600	0,500
CBF	16	0,046	0,070	0,535	0,046	0,002	0,005	0,003	0,032	0,030	0,034	0,030
Coffee	2	0,161	0,161	0,107	0,250	0,125	0,143	0,161	0,196	0,304	0,286	0,304
ECG200	5	0,154	0,151	0,184	0,151	0,221	0,175	0,214	0,173	0,206	0,166	0,206
FaceAll	11	0,185	0,221	0,399	0,177	0,090	0,071	0,078	0,032	0,037	0,040	0,037
FaceFour	5	0,116	0,190	0,448	0,083	0,145	0,109	0,060	0,027	0,054	0,056	0,054
fish	5	0,302	0,283	0,304	0,328	0,309	0,270	0,190	0,155	0,169	0,184	0,169
Gun_Point	5	0,089	0,116	0,173	0,095	0,129	0,041	0,057	0,049	0,056	0,067	0,056
Lighting2	5	0,217	0,300	0,388	0,219	0,180	0,199	0,360	0,260	0,294	0,267	0,294
Lighting7	2	0,412	0,406	0,595	0,413	0,287	0,308	0,776	0,447	0,405	0,441	0,405
OliveOil	2	0,150	0,117	0,183	0,200	0,133	0,117	0,150	0,167	0,183	0,200	0,183
OSULeaf	5	0,452	0,464	0,518	0,456	0,427	0,422	0,391	0,260	0,242	0,252	0,242
SwedishLeaf	5	0,289	0,296	0,358	0,282	0,249	0,205	0,164	0,136	0,151	0,146	0,151
synthetic_control	5	0,145	0,132	0,223	0,166	0,012	0,015	0,035	0,058	0,044	0,048	0,044
Trace	5	0,286	0,341	0,459	0,249	0,014	0,018	0,160	0,133	0,049	0,090	0,049
Two_Patterns	5	0,038	0,102	0,796	0,049	0,000	0,000	0,000	0,001	0,001	0,001	0,001
wafer	7	0,004	0,005	0,020	0,005	0,016	0,005	0,008	0,004	0,005	0,005	0,005
yoga	11	0,162	0,159	0,181	0,167	0,151	0,140	0,129	0,113	0,124	0,119	0,124

Table 4.2. Error rates of similarity measures

Tabela 4.2. Greške mera sličnosti

For the purpose of testing similarity measures a *Graphical User Interface* (SCVGUI) has been developed in Java at the Department of Mathematics and Informatics, Faculty of Sciences, University of Novi Sad. The input for this application is a specification of one or more cross-validation experiments. FAP will perform the cross-validations based on this specification.

4.4. Distance Matrices

One of the main goals of this dissertation is to reveal and clarify the influence of elastic similarity measures constrained with the Sakoe-Chiba band and the different weighting schemes on classification accuracy of the nearest neighbor classifier. In order to determine the class of an unclassified time series, the kNN classifier first must calculate the distances between that series and all other series in the training set.

In case of long time-series and/or complex similarity measures, it is very helpful to save the similarity values between time series, and thus speed up the experiments. Similarities between the time series of a dataset are kept in the form of a *distance matrix* where element (i, j) contains the distance between i-th and j-th time series from the dataset.

Table 4.3 shows part of the distance matrix obtained using the DTW similarity measure (Section 2.1.2) on the first 10 elements of the *beef* dataset (see Table 4.1 in Section 4.2). Since DTW is a symmetric distance (i.e. it satisfies the condition $d(Q, S) = d(S, Q)$ for every two time series Q and S) it is sufficient to compute a lower triangular matrix.

	1	2	3	4	5	6	7	8	9	10
1	0									
2	0.25158	0								
3	1.060174	1.374043	0							
4	0.666656	0.090452	1.026273	0						
5	0.073874	0.178184	0.319278	0.39273	0					
6	0.008442	0.226699	1.504747	0.634164	0.161802	0				
7	10.38343	11.51138	2.272932	10.42762	7.426631	11.79298	0			
8	2.950115	3.657049	0.406857	3.269754	1.251925	3.687409	0.988539	0		
9	0.984133	1.633875	0.209702	1.528592	0.236096	1.5402	3.707063	0.240569	0	
10	6.611101	7.693609	0.926652	6.936514	4.414624	7.760327	0.243946	0.342136	1.538323	0

Table 4.3. Part of the distance matrix obtained using DTW on the *beef* dataset

Tabela 4.3. Deo matrice rastojanja dobijene primenom DTW mere sličnosti na elemente skupa podataka *beef*

For the purposes of our experiments, we have developed a special *Graphical User Interface* (DMGUI) for generating distance matrices relying on the services of our FAP library (Section 4.1 and Chapter 6). DMGUI has been implemented in Java at the Department of Mathematics and Informatics, Faculty of Sciences, University of Novi Sad. Similarly as in the case of Graphical User Interface for validating FAP, the input for this application is a specification of one or more commands for generating distance matrices.

The process of generating distance matrices can be interrupted and resumed similarly as in the case of Graphical User Interface for validating FAP. Optionally, the serialized Java objects can even be migrated to another computer where the computation can be continued. Furthermore, based on the obtained distance matrices, this application provides another feature: it can produce matrices of nearest neighbors that are required for the experiments in Chapter 5. The *matrix of nearest neighbors* for one dataset is the matrix where element (i, j) contains the index of the j-th nearest neighbor of the i-th time series from the dataset.

Chapter 5

Techniques for Improving Classification Accuracy

A time series is composed of a series of numbers that describes the change of the observed phenomenon over time. Each element of a time series characterizes the phenomenon under investigation at one point in time [19]. Time series are used for storage, display and analysis of data across a wide range of different domains, including various areas of science, medicine, economics, ecology, telecommunications and meteorology [25, 40, 63].

The suitability of time series in a broad spectrum of different areas of research and application, and the need to process an ever-growing quantity of data, have intensified the study of different time-series data mining tasks: indexing, classification, clustering, prediction, segmentation, anomaly detection and others [21, 39].

In recent years, there is a growing interest for research in different aspects of time-series classification [34, 37, 45, 93, 111, 130, 131]. The possibility of applying many well-known machine learning techniques was investigated in this field. These techniques include: decision trees [102], neural networks [81], support vector machines [128], first order logic rules [101], Bayesian classifiers [85] and others.

In several significant research contributions in this domain [21, 93, 119, 130], the simple 1NN classifier (see Section 3.1) is selected as one of the most accurate classifiers for time-series data, demonstrating comparable and even superior performance than many more complex classification approaches.

The basic idea behind the 1NN and kNN classifiers is to find the nearest neighbors of a given unclassified time series among the time series of the training set (see Section 3.1). Finding the nearest neighbors requires comparing time series and calculating the distances between them (see Section 2.1). In the domain of time series, several different similarity-based distance measures are applied for comparing data sequences [31, 68, 108]. The most commonly used and most frequently investigated time-series similarity measures are Euclidean distance [26], Dynamic Time Warping (DTW) [10], Longest Common Subsequence (LCS) [121], Edit Distance with Real Penalty (ERP) [16], and Edit Distance on Real sequence (EDR) [17].

Most of these similarity measures are based on dynamic programming. It is well known that the computational complexity of these dynamic programming algorithms is quadratic, which is often not suitable for larger real-world problems. However, the usage of global constraints such as Sakoe-Chiba band [105] and Itakura parallelogram [44] can speed up the

calculation of similarities and it can prevent some bad alignments, where a relatively small part of one time series maps onto a large section of another time series (see Sections 2.2 and 5.1).

As an attempt to improve classification accuracy, several different methods for assigning weights to the nearest neighbors are proposed in the literature [22, 35, 36, 62, 69, 74, 84, 133] (see Section 3.1). Generally, each paper that presents a new way of computing weights reports the superiority of the newly introduced method compared to some previous solutions. Several of these papers [35, 36, 133] compare various weighting schemes using a relatively small number of datasets (commonly 12) from the UCI machine learning repository [7]. The conclusions are usually based on comparing classification results using only Euclidean distance and they are not investigated in the field of time-series data mining.

Based on the results of a series of extensive experiments, in this chapter we will investigate how the classification accuracy of 1NN and kNN classifiers can be further improved by constraining the similarity measures and assigning weights to the nearest neighbors. Furthermore, using these techniques we will verify the view that the simple 1NN is very hard to beat [130].

In the first step of our investigations, we will explore to what extent does the application of different sized Sakoe-Chiba bands speed up the calculation of similarities between time series (Section 5.2). We will report the calculation times for different sizes of Sakoe-Chiba bands in order to explore the speed-up gained from applying them on the two most representative similarity measures for time series based on dynamic programming: DTW and LCS.

In Section 5.3 we will present the results of examination of the influence of the Sakoe-Chiba band on classification accuracy of the simple nearest neighbor (1NN) classifier analyzing it through three stages. In the first stage we will explore the change of the 1NN graph with respect to the change of the constraint size (Section 5.3.1). In the second stage we will investigate how these changes impact on the 1NN classifier regarding the nearest-neighbors' classes (Section 5.3.2). Finally, in the third stage we will examine the constraint's impact on classification accuracy (Section 5.3.3).

We will show that there are notable differences in the effects of the constraints on different distance measures. DTW was found to be the most sensitive to the introduction of global constraints, while EDR is the least sensitive. The behavior of ERP and LCS measures was determined to be somewhere in between. Comparison of 1NN classifier performance showed that DTW generally has a slight edge over other distance measures, but is more sensitive to the choice of the constraint's size.

Section 5.4 is devoted to the analysis of the impact of the Sakoe-Chiba band on kNN classifier's accuracy. In this section we will observe the relationship between the parameter k (the number of neighbors used by the kNN classifier) and the average smallest error rate obtained for different values of parameter r (the size of the warping window expressed as the percentage of the time-series length). For each of the datasets and for each value of parameter k (in the range of 1 to 30) we will search for the value of parameter r which will

produce the smallest classification error. These examinations will be done for both the majority voting kNN and for the distance-weighted kNN classifier.

We will show that, on average, the kNN classifier gives the best results for $k = 1$ without a weighting scheme. On the other hand, when a weighting scheme is introduced, the situation is changed. The best results are obtained for the values around $k = 4$. When observing the value of the constraint parameter r, the introduction of the weighting scheme has an important impact. For unweighted kNN, the value of the constraint grows as k grows: we need wider and wider warping windows to get the best accuracy. On the other hand, using a weighting scheme the value of the constraint remains approximately the same for all values of k.

In Section 5.5 we will compare a large number of different weighting schemes in relation to the majority-voting kNN and the simple 1NN classifier in case of the three most commonly used time-series similarity measures (Euclidean distance, unconstrained DTW and unconstrained LCS). We will examine whether these weighting schemes can improve the classification accuracy. We will demonstrate that, among the discussed weighting schemes, the DualD (Eq. (3.11)) and the Dudani (Eq. (3.2)) give the best results with the considered similarity measures.

5.1. Constraining the Similarity Measures and Applying Weights

The choice of the similarity measure is one of the most significant aspects of time-series analysis - it should correctly reflect the resemblance between the data presented in the form of time series. Similarity measures represent a critical component of many tasks of mining time series, including: classification, clustering, prediction, anomaly detection, and others.

The advantages of Euclidean distance (it is easily implementable, fast to compute and it represents a distance metric - see Section 2.1.1) have made it one of the most commonly used similarity measure for time series research and applications [4, 14, 47, 48]. However, due to the linear aligning of the points of the time series it is sensitive to distortions and shifting along the time axis [49, 98]. To address this shortcoming, many different elastic similarity measures were proposed. Among them, some of the most widely used and studied are Dynamic Time Warping (DTW) [10], Longest Common Subsequence (LCS) [121] and their extensions, Edit Distance with Real Penalty (ERP) [16] and Edit Distance on Real sequence (EDR) [17].

The implementations of these elastic similarity measures are based on dynamic programming: in order to determine the similarity between two time series we need to compare each element of one time series with each element of the other one. This can lead to pathological non-linear aligning of the points (where a relatively small part of one time series maps onto a large section of the other time series) and slow down the computations.

One way to avoid these adverse effects is to constrain the warping path using the Sakoe-Chiba band [105] (see Section 2.2).

Algorithm 1 shows the application of the Sakoe-Chiba band in case of the DTW similarity measure (see Section 2.1.2). The input of the algorithm consists of two time series Q and S of the same length n, and the warping window width r specified as a percentage of n. In accordance with Eq. (2.3), we first initialize the warping matrix D (lines 3-5) and then calculate its elements constraining the warping window around its diagonal using the given value of parameter r (lines 6-16). The Sakoe-Chiba band is applied analogously also to the other examined elastic similarity measures (LCS, ERP and EDR), based on their recursive definitions given in Section 2.1.

Algorithm 1: Constraining DTW with the Sakoe-Chiba band

Input: two time series Q and S of the same length n, warping window width r in percentage of n

Output: distance between Q and S computed using DTW constrained by the Sakoe-Chiba band

1. Let D be an $(n+1) \times (n+1)$ matrix of real numbers
2. $len = floor(n * r/100)$
3. $D[0,0] = 0$
4. FOR i = 1 to len DO
5. $\quad D[0,i] = \infty$
6. FOR i = 1 TO n DO
7. $\quad start = MAX(i, i - len)$
8. $\quad end = MIN(n, i + len)$
9. $\quad D[i, start - 1] = \infty$
10. $\quad$ IF $(i + len \leq n)$
11. $\quad\quad D[i-1, end] = \infty$
12. $\quad$ FOR j = $start$ TO end DO
13. $\quad\quad delta = Q[i] - S[j]$
14. $\quad\quad D[i,j] = delta * delta + MIN(D[i-1,j], D[i-1,j-1], D[i,j-1])$
15. $\quad$ END
16. END
17. RETURN $D[n,n]$

The other technique that we will examine in order to improve the accuracy of the kNN classifier in the domain of time-series data mining refers to assigning weights to the neighbors of the unclassified time series according to their distance from it. We will show that using weighting schemes with kNN produces higher classification accuracy than the simple 1NN classifier for several data sets.

The process of applying weights with the kNN classifier is presented in Algorithm 2. The input of the algorithm consists of an unclassified time series Q, a training set T, the number of neighbors k, a distance measure d and the value of ε. As the result, the algorithm returns the label (from the set of labels of the k nearest neighbors of Q in the training set T) selected

for Q. In this example, weights are calculated as defined by Eq. (3.5). ε is an auxiliary constant which is necessary to avoid division by zero when the distance between the time series is zero (we have used 0.001 in our experiments in Sections 5.4 and 5.5).

Algorithm 2: Applying weights with the kNN classifier

Input: unclassified time series Q, training set T, number of neighbors k, distance measure d, constant ε

Output: the label selected for Q

1. Sort the elements of T in ascending order according to their distance from Q using distance measure d.
2. Let m denotes the number of different class labels among the first k elements of T
3. Initialize array $labels[1 \dots m]$ with the different class labels among the first k elements of T
4. Initialize array $weights[1 \dots m]$ with zeros
5. FOR i = 1 TO k DO
6. Find j so that $labels[j]$ = the label of $T[k]$
7. $weights[j] += 1/(d(Q, T[k])^2 + \epsilon)$
8. END
9. Find j so that $weights[j]$ is the largest value among $weights[1 \dots m]$
10. RETURN $labels[j]$

5.2. Performance of Constrained Similarity Measures

In this section we will investigate the influence of global constraints on the efficiency of the two most illustrative similarity measures based on dynamic programming: DTW and LCS. In order to verify to what extent will applying global constraints speed up the calculation of the similarities between time series we measured the time required to generate distance matrices of several datasets. The *distance matrix* for one dataset is the matrix where element (i, j) contains the distance between i-th and j-th time series from dataset. The calculation of the distance matrix is a time-consuming operation, which makes it suitable for measuring the efficiency of global constraints.

For every similarity measure, the experiments are performed with the unconstrained measure (100%) and a measure with the following constraints: 75%, 50%, 25%, 20%, 15%, 10%, 5%, 1% and 0% of the size of the time series. This distribution was chosen because it is expected that measures with larger constraints behave similarly to the unconstrained measure, while smaller constraints have more interesting behavior [98, 130].

A comprehensive set of experiments was conducted on 38 datasets from UCR Time Series Repository [50]. All experiments are performed on AMD Phenom II X4 945 with 3GB RAM. The calculation times for individual datasets (in milliseconds) are shown in Appendix A. The average calculation times for the observed warping window widths are presented in Table 5.1 and they are graphically displayed in Figure 5.1.

It is evident that the introduction of global constraints in both measures significantly speeds up the process of distance matrix computation. The difference of computation times between an unconstrained measure and a measure with a small constraint is two and somewhere three orders of magnitude.

Furthermore, it is known for DTW that smaller values of constraints can give more accurate classification [98]. The authors also reported that the average constraint size, which gives the best accuracy, for all datasets is 4% of the time-series length. In addition, the usage of lower bounding (such as LB_Keogh [49]) for constrained DTW can further speed up the process of indexing time-series data.

	100%	75%	50%	25%	20%	15%	10%	5%	1%	0%
DTW	45466332	41027080	32841382	19235818	15831060	12379785	8456105	4432871	1081910	202421.1
LCS	30807630	28694236	22935060	13408023	11049745	8609224	5909023	3138488	793754.3	254794.2

Table 5.1. Average calculation times of distance matrices in milliseconds

Tabela 5.1. Prosečna vremena izračunavanja matrica rastojanja u milisekundama

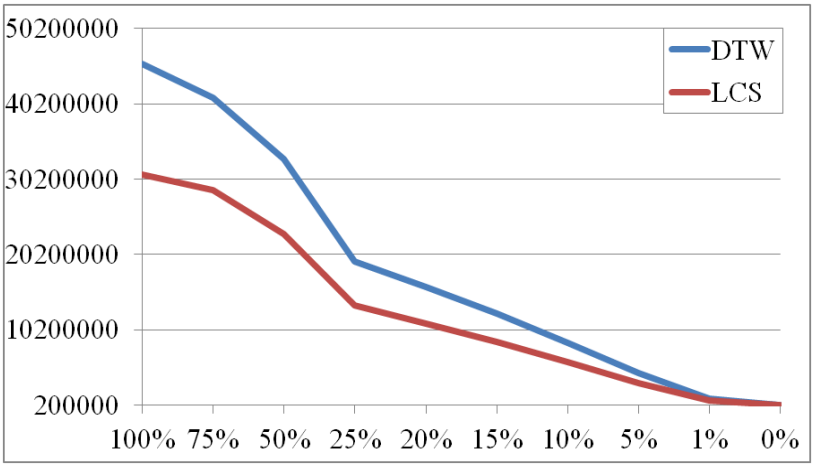


Figure 5.1. Graphical display of average calculation times of distance matrices for different warping window widths

Slika 5.1. Grafički prikaz prosečnih vremena izračunavanja matrica rastojanja za različite širine pojasa iskrivljenja

5.3. Improving the Accuracy of the 1NN Classifier Using Constrained Similarity Measures

In this section we will present the results of our expanded study of the influence of the Sakoe-Chiba band on the most widely used elastic similarity measures: DTW, LCS, ERP and EDR. To better understand the influence of global constraints we will explore the efficiency and behavior of the 1NN classifier for different values of constraints and investigate its accuracy.

Further, we will report the change of the 1NN graph with regard to the change of the global constraints. Our choice of 1NN was mainly motivated by reports that it achieves among the best results compared to many other sophisticated classifiers for time-series data [19, 85, 107, 117]. In addition, the accuracy of 1NN directly reflects the quality of the underlying similarity measure [116].

In the first phase of the experiments we will explore the change of the 1NN graph with respect to the change of the constraint size (Section 5.3.1).

In the second phase we will investigate how these changes impact on the 1NN classifier regarding nearest-neighbors' classes (Section 5.3.2).

The examination of the constraint's impact on classification accuracy is discussed in the third part of our study (Section 5.3.3).

For all four considered similarity measures (DTW, LCS, ERP and EDR) we have limited the warping window with the following constraint values for r: 90%, 80%, 70%, 60%, 50%, 45%, 40%, 35%, 30%, and all values from 25% to 0% in steps of 1% of time-series length. These values were chosen because it is expected that the measures with larger constraints behave similarly to the unconstrained measure, while smaller constraints have more interesting behavior [98, 130].

5.3.1. Change of the 1NN Graph with Narrowing Constraints

The 1-nearest-neighbor graph (1NN) is a directed graph where each time series is connected with its nearest neighbor. Since the 1NN classifier assigns the class of the nearest neighbor to a yet unclassified time series, the changes in the 1NN graph directly affect classification accuracy.

We calculated this graph for unconstrained measures and for measures with the following constraints: 90%, 80%, 70%, 60%, 50%, 45%, 40%, 35%, 30%, and all values from 25% to 0% in steps of 1% of time-series length. After that, we observed the change of the nearest neighbor graphs as the percentage of time series (nodes in the graph) that changed their nearest neighbor compared to the nearest neighbor in the unconstrained measure.

The graphical representation of results can be seen in Figures 5.2 through 5.6 for DTW, LCS, ERP and EDR, respectively. Each figure is represented by two charts for the sake of readability. The first chart (A) contains the behavior of 10 most representative datasets, illustrating the behavior of the majority of datasets. The second chart (B) shows the general statistics over all datasets: minimum values, maximum values, average values and the deviations from the average values. Appendix B contains detailed plots of the change of 1NN graph for all four examined elastic similarity measures.

Dynamic Time Warping (DTW). The 1NN graphs of the DTW measure (Figure 5.2) remain the same until the size of the constraint is narrowed to approximately 60%–50%, and after that the graphs start to change. As the width of the warping window becomes smaller, an

increasing number of datasets exhibit bigger changes. When the size of the Sakoe-Chiba band falls below 5% of the time-series length, changes are present in all of the datasets. For $r = 0\%$, changes higher than 50% have been registered for the majority of the datasets (the only exceptions are *beef*, *chlorineconcentration*, *coffee*, *italypowerdemand*, *mallat* and *oliveoil*) and for some of them the alteration levels even reach values above 90% (*50words*, *cbf*, *starlightcurves*, *synthetic_control*, *twopatterns*, *uwavegesturelibrary_x*, *wordssynonyms*).

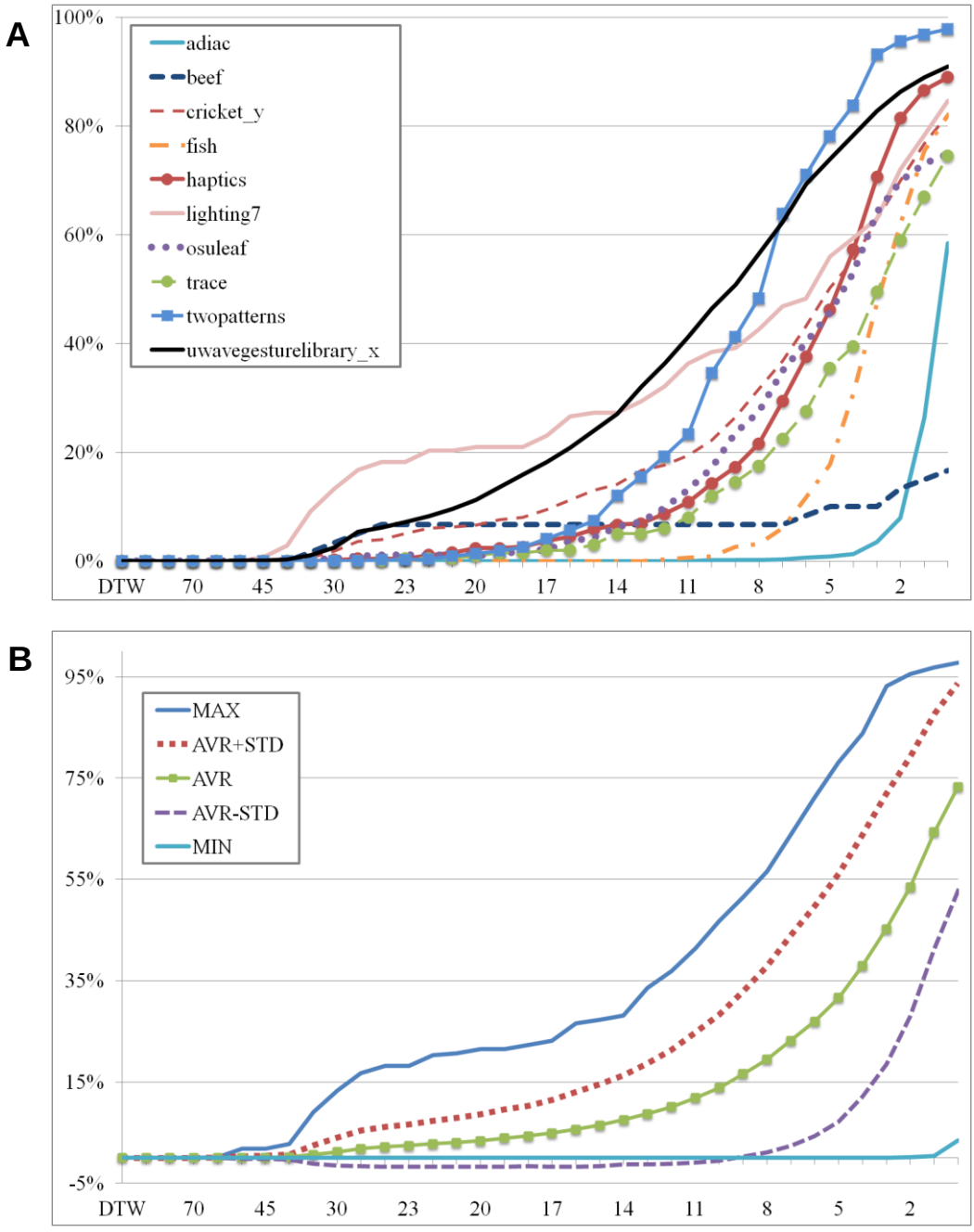


Figure 5.2. Change of 1NN graph for DTW

Slika 5.2. Promena 1NN grafa za DTW

Longest Common Subsequence (LCS). The situation with LCS (Figure 5.3) is even more drastic: the 1NN graphs remain the same to approximately 30%–25%, while for smaller constraints they change more quickly for most of the datasets. When r reaches 0%, changes greater than 90% occur in a much larger number of datasets than in the case of DTW (17, opposed to 7). However, there are some exceptions. A number of datasets (*beef*, *ecgfivedays*, *mallat*, *oliveoil, trace*) exhibits changes only for very small values of the constraint (less than 2%) and in the case of *chlorineconcentration* there are some oscillations.

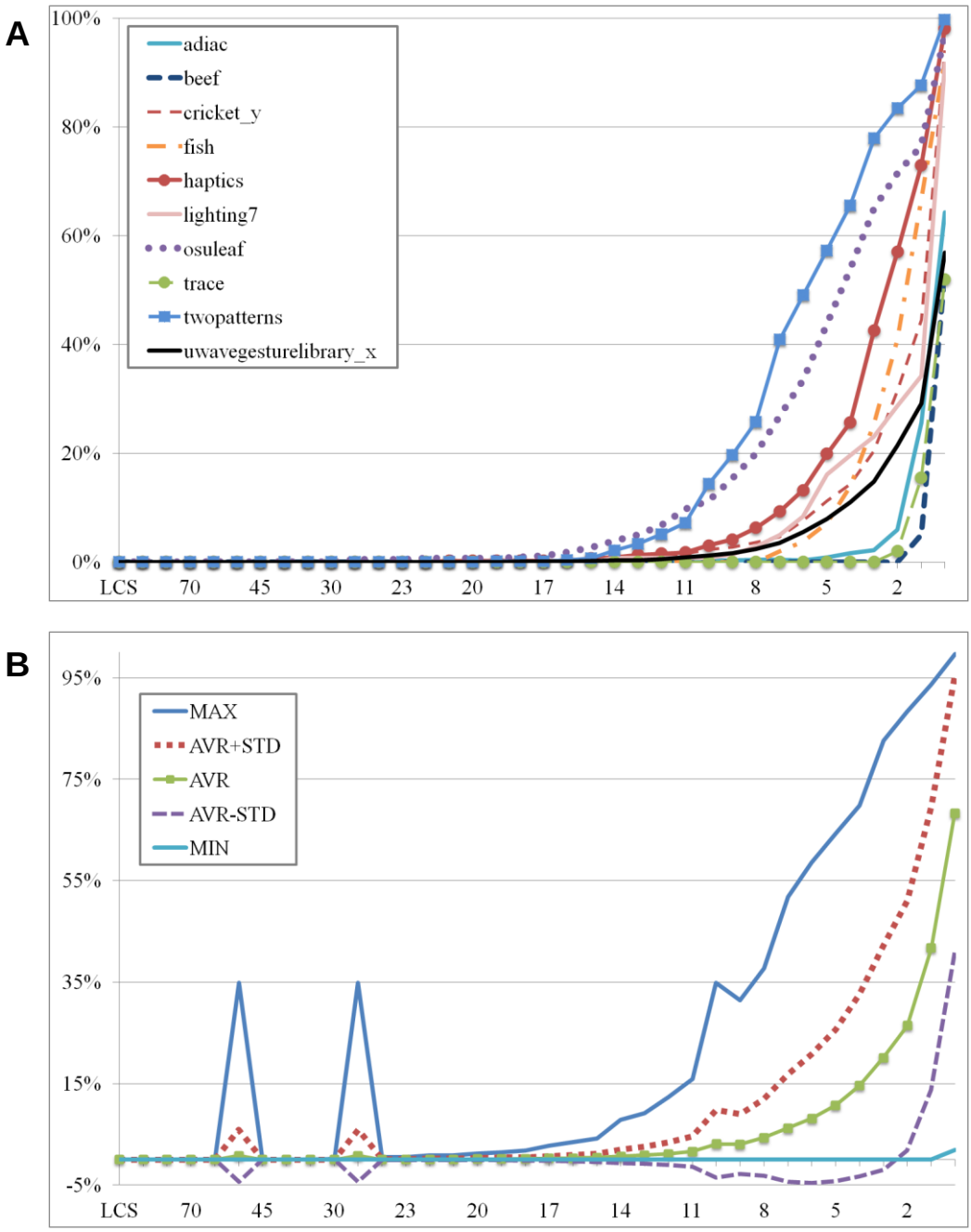


Figure 5.3. Change of 1NN graph for LCS

Slika 5.3. Promena 1NN grafa za LCS

The dataset *chlorineconcentration* produces exactly the same percentage of changed neighbors (34.873%) for several values of the constraint (50%, 25%, 10%, 1%), but in all other cases there are no changes at all - except for 0% (Figure 5.4). We have investigated the structure of this dataset and found that the time series are periodical, where all time series have approximately the same period. Since the LCS measure searches for the longest common subsequence, it turns out that for most constraint values the LCS algorithm finds the same sequence as the unconstrained LCS. Some values of the constraint break that sequence, which is then no more the longest, and as a consequence some other time series are found as nearest neighbors. This behavior is a consequence of the strict periodicity of this dataset.

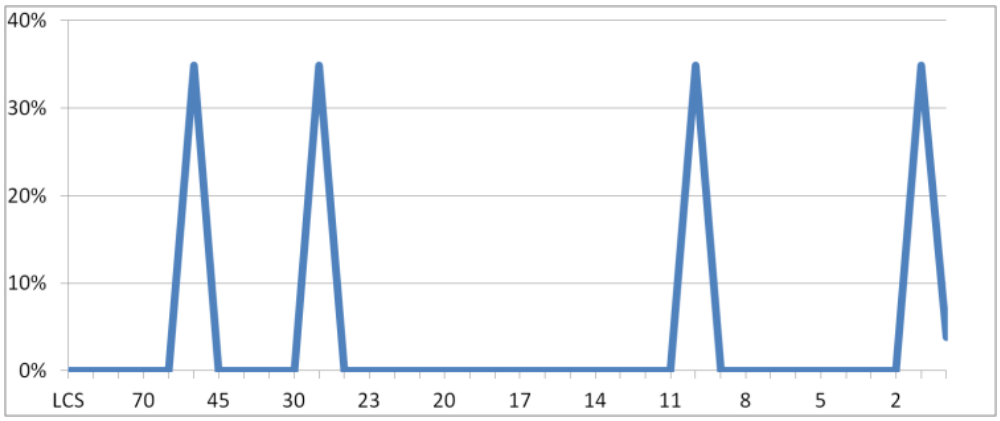


Figure 5.4. Change of 1NN graph for *chlorineconcentration* in case of LCS

Slika 5.4. Promena 1NN grafa za *chlorineconcentration* u slučaju LCS-a

Edit Distance with Real Penalty (ERP). The 1NN graph of the ERP measure (Figure 5.5) remains the same until the size of the constraint is narrowed to approximately 60%–50%, similarly as in the case of DTW. After that the graph starts to change more noticeably. For small values of the constraint (5%–0%) this change becomes significant for most of the datasets and in some cases even reaches values above 70%–90%. It is also evident that the use of the Sakoe-Chiba band does not affect the 1NN graph in the same way for each dataset: in case of a small number of datasets the changes are subtle or there are no changes at all (*adiac*, *chlorineconcentration*, *coffee*, gun_*point*, *mallat*, o*liveoil*, *trace*).

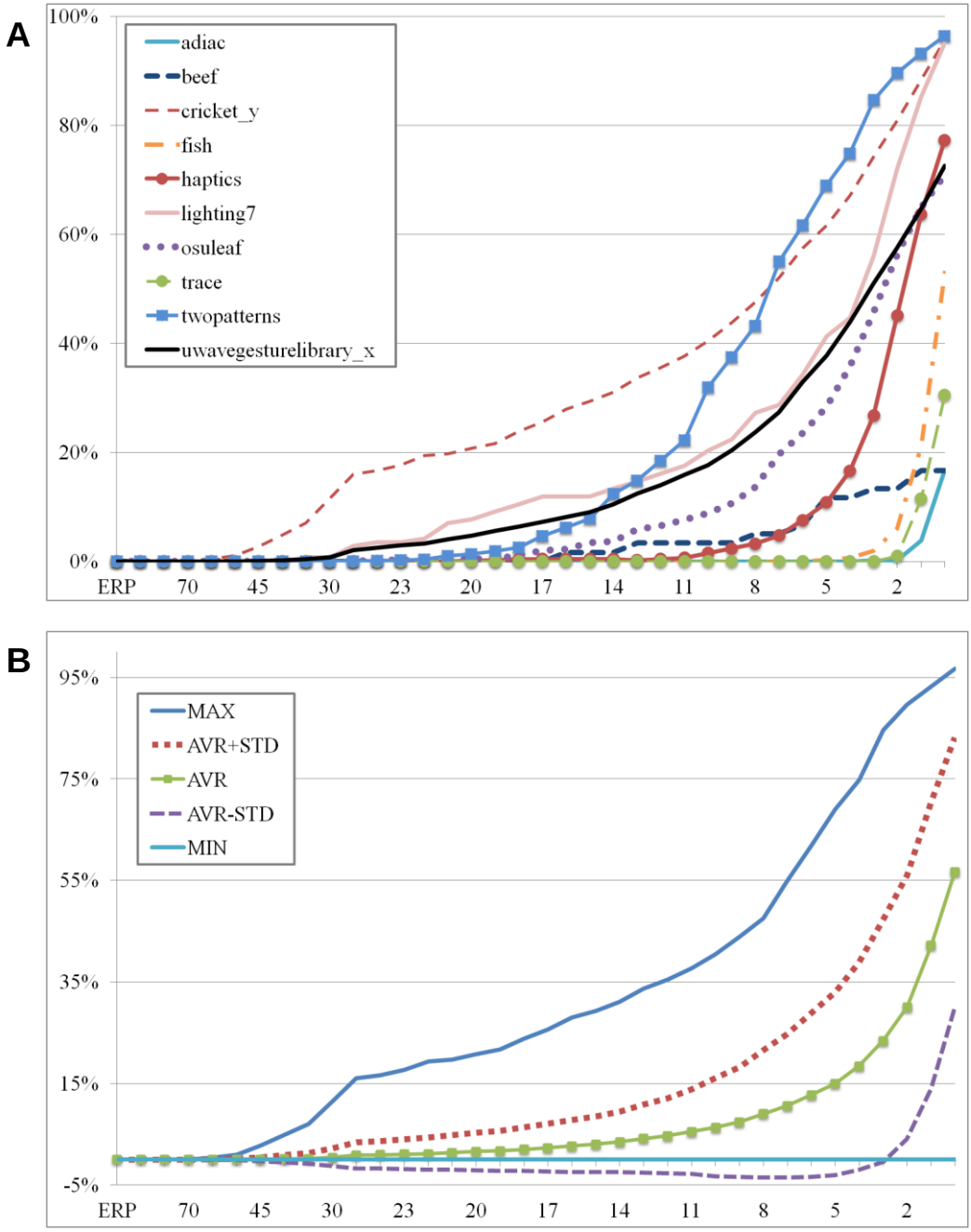


Figure 5.5. Change of 1NN graph for ERP

Slika 5.5. Promena 1NN grafa za ERP

Edit Distance on Real sequence (EDR). EDR (Figure 5.6) behaves in a similar manner to LCS, but there are three noticeable differences. The changes begin later, only when the value of the constraint drops below 20%. The changes do not reach such high values as in the case of LCS – the maximum values are between 60%–80%. Again, there is a small number of datasets where the changes are subtle (*beef*, *chlorineconcentration*, *ecgfivedays*, *mallat*, *oliveoil*, *trace*).

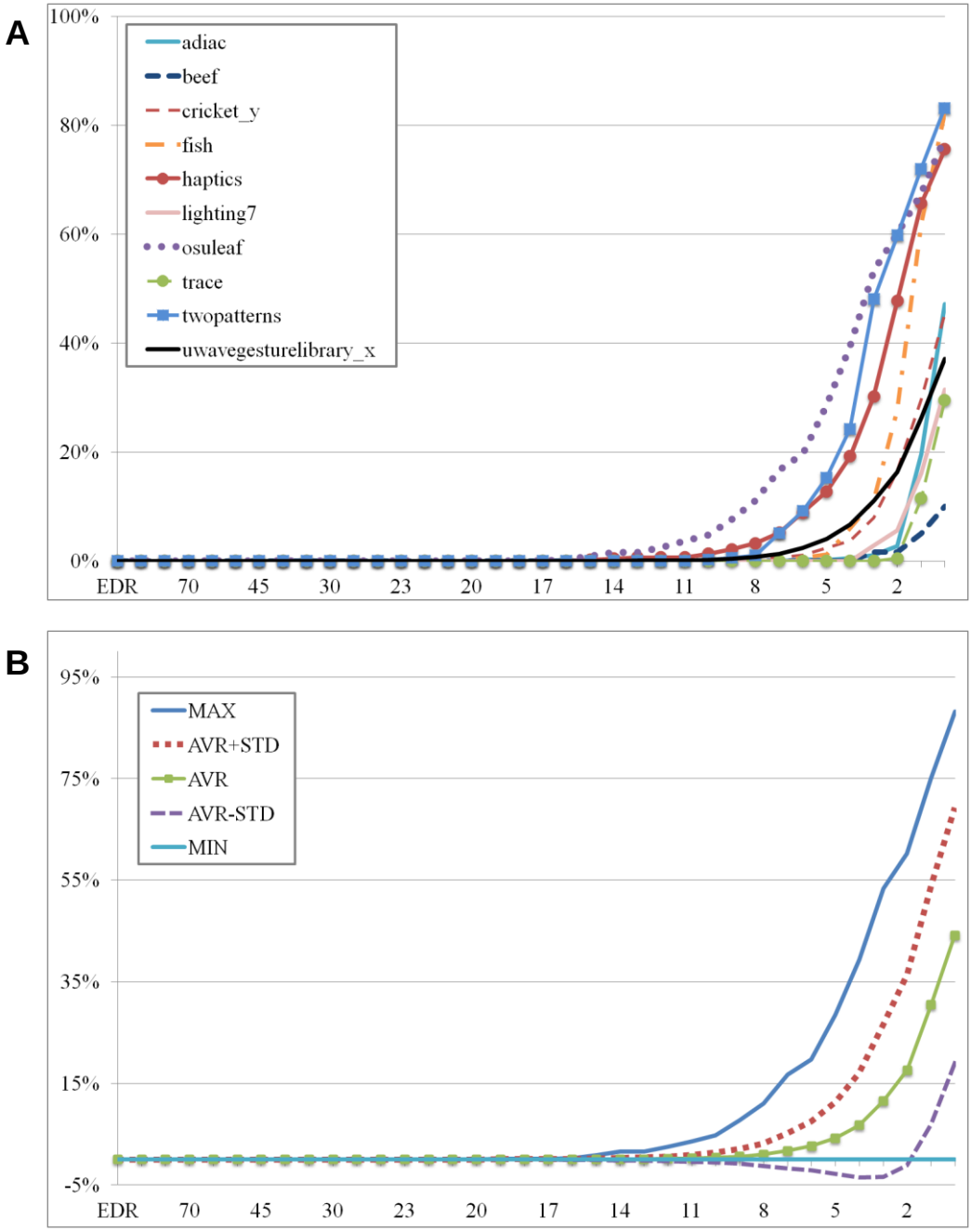


Figure 5.6. Change of 1NN graph for EDR

Slika 5.6. Promena 1NN grafa za EDR

From the obtained results we can clearly see that for low values of the constraint the change of the 1NN graph becomes significant for most datasets in the case of all four similarity measures. The results suggest that the constrained measures represent qualitatively and as well quantitatively different measures than the unconstrained ones.

It is also clear that the application of the Sakoe-Chiba band does not have the same effect on all datasets, and there are noticeable differences in the behavior of the similarity measures. By studying the results we can immediately see two quite conspicuous characteristics of the Sakoe-Chiba band. First, warping window width of 0% most drastically affects LCS: for a significant number of datasets there is a sudden increase in the percentage of changed neighbors compared to $r = 1\%$ (some examples are: *cinc_ecg_torso*, *coffee*, *diatomsizereduction*, *noninvasivefatalecg_thorax1*, *noninvasivefatalecg_thorax2* and *wafer*). Secondly, LCS, ERP and EDR for some datasets (for example, *chlorineconcentration*, *mallat*, *oliveoil*, *trace*) show only tenuous changes (or no changes at all except for the values of constraint $r = 0\%$), with appreciably larger changes in the case of DTW.

The general traits of differences between the similarity measures can easily be seen on the charts in Figure 5.7 and Figure 5.8. Figure 5.7 presents the average values of the changes in the 1NN graphs across all datasets, and Figure 5.8 shows the percentages of those datasets for which there are changes in the 1-nearest neighbor graph produced by the constrained similarity measures, compared to the 1NN graph of the unconstrained ones. It is obvious that the use of the Sakoe-Chiba band exhibits the greatest influence on DTW: changes in the 1NN graph arise as soon as the size of the constraint is narrowed to 60% and for very narrow warping windows they reach (on average) the highest values among the observed similarity measures.

The smallest influence occurs in the case of EDR: the 1NN graph begins to change only when the size of the warping window drops below 20% of the length of time series, and the average change for $r = 0\%$ is lowest here. For most datasets LCS and ERP behave very similarly: they are situated "between" DTW and EDR.

This relationship between the similarity measures can also be seen in Figure 5.9 which shows the highest width of the warping window required to change at least 10% of the nodes in the 1NN graph (the first chart again contains 10 most representative datasets for the sake of readability, while the second chart shows general statistics for all datasets - the detailed plots are presented in Appendix C). Changes of this magnitude appear earliest for DTW (with average warping window width about 10.18%), followed by ERP (6.33%), then by LCS (4.78%), and at the end by EDR (2.54%). Comparisons using the Wilcoxon sign-rank test [29] reveal statistical significance of pairwise differences, with p-values shown in Table 5.2 (where the difference between ERP and LCS may be considered the one borderline case).

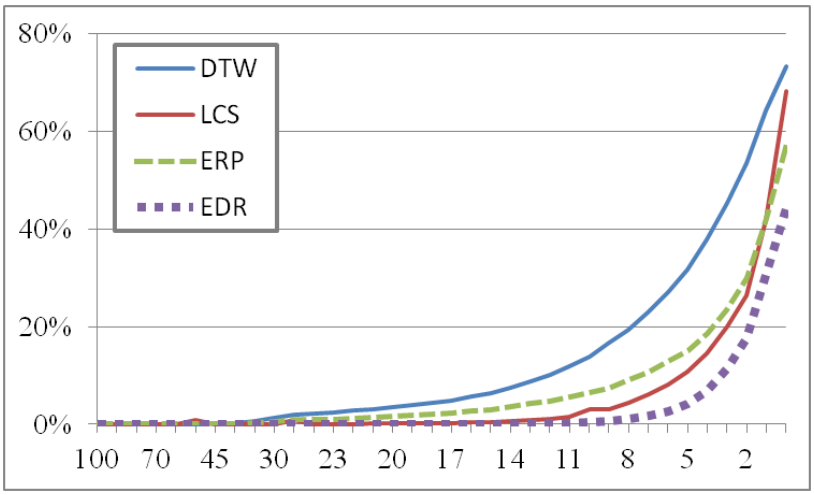


Figure 5.7. The average changes in the 1NN graph

Slika 5.7. Prosečne promene u 1NN grafu

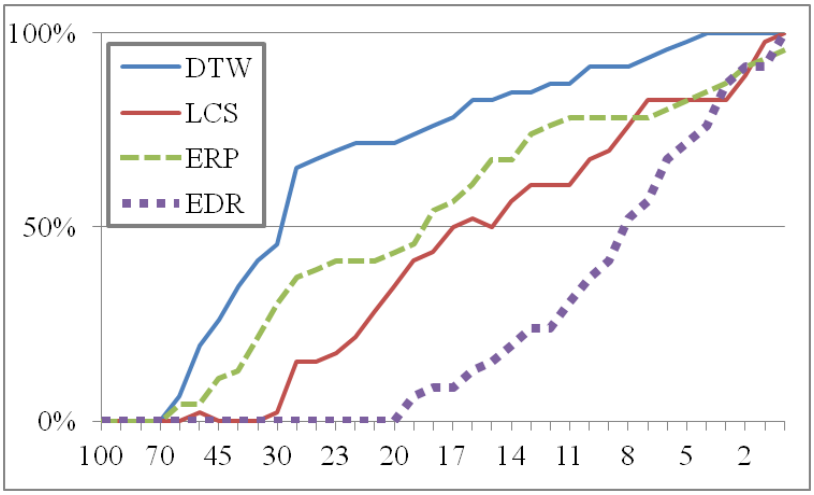


Figure 5.8. The percentage of the datasets with changed 1NN graphs

Slika 5.8. Procenat skupova podataka sa promenjenim 1NN grafovima

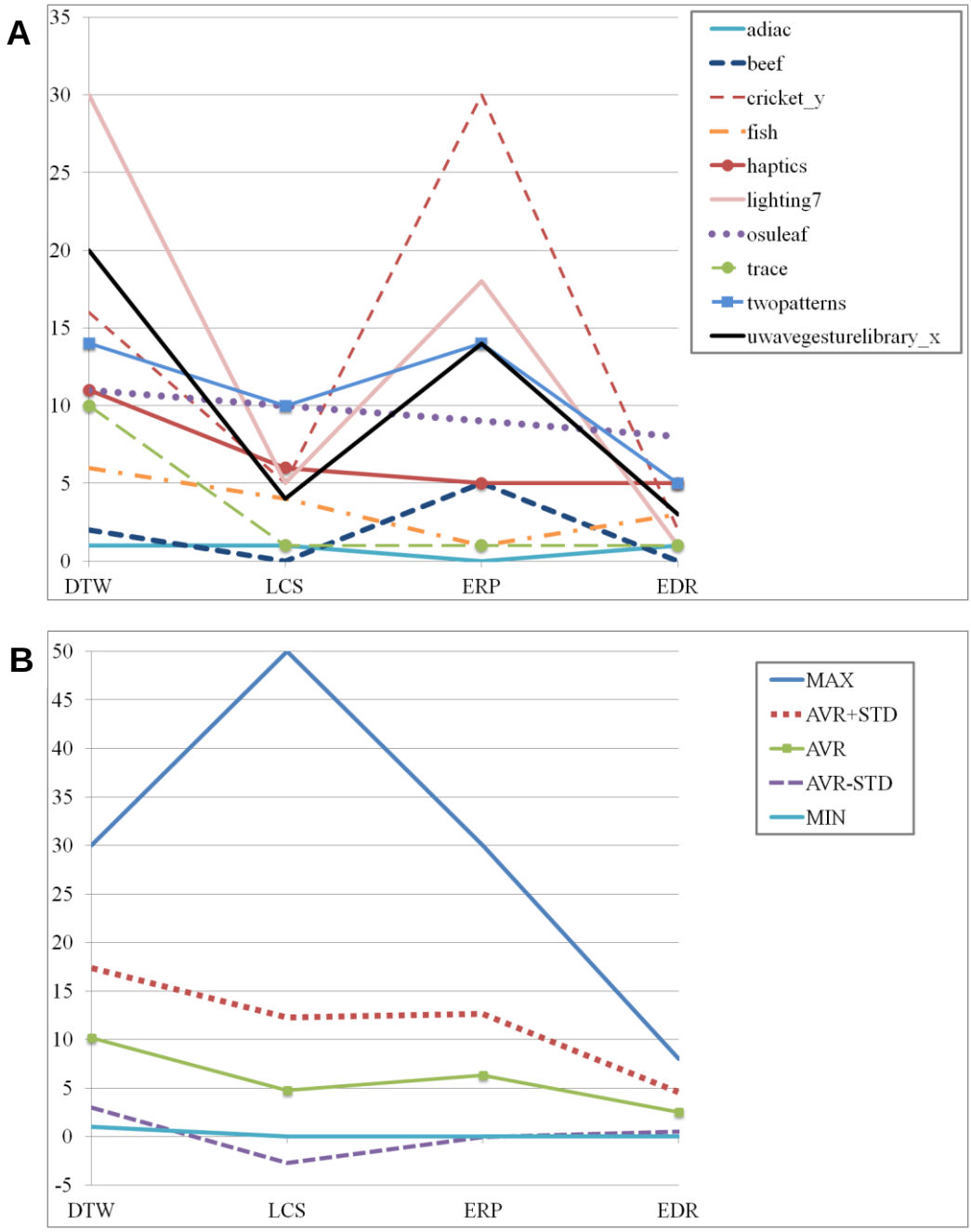


Figure 5.9. The highest warping windows needed to change at least 10% of the 1NN graph

Slika 5.9. Najveći pojasi iskrivljenja potrebni da se promeni najmanje 10% 1NN grafa

	LCS	ERP	EDR
DTW	4.63E-07	3.63E-06	9.25E-09
LCS		0.018192	0.001279
ERP			2.16E-05

Table 5.2. *p* values for the pairwise Wilcoxon sign-rank test of the differences in the highest width of the warping window required to change at least 10% of the nodes in the 1NN graph

Tabela 5.2. *p* vrednosti dobijene uparenim Vilkoksonovim testom rangova sa znakom nad razlikama najvećih širina pojasa iskrivljenja potrebnih da se promeni najmanje 10% čvorova u 1NN grafu

5.3.2. Change of Classes with Narrowing Constraints

The results described in Section 5.3.1 clearly indicate that the application of the Sakoe-Chiba band can significantly change the structure of the 1-nearest neighbor graph, especially for small warping-window widths. We have also seen that this influence is not manifested in the same way for different similarity measures. In this section we explore how these changes in the 1-nearest neighbor graph may affect the behavior of the 1NN classifier.

Classification denotes the process of grouping time series into predefined classes. The 1NN classifier represents a very simple form of classification: the class of the unclassified time series is determined as the class of its most similar time series. Despite its simplicity, the 1NN classifier often produces better results than other more complex classifiers [19, 85, 117].

Since the results of 1NN classification depend entirely on the class of the nearest neighbor, changes in the nearest-neighbor graph directly affect the classification. In this section we will examine the extent to which nearest neighbors change their classes under the influence of the Sakoe-Chiba band. Similarly to the first part of the experiments, we record the percentage of those nearest neighbors which have under the influence of global constraints changed their classes compared to the nearest neighbors in the unconstrained measure.

The graphical representation of results can be seen in Figures 5.10 through 5.13 for DTW, LCS, ERP and EDR, respectively. Each figure is represented by two charts for the sake of readability. The first chart (A) contains the behavior of 10 most representative datasets while the second chart (B) shows the general statistics over all datasets: minimum values, maximum values, average values and the deviation from the average values. Appendix C contains detailed plots of the change of classes for all four examined elastic similarity measures.

In Section 5.3.1 we have seen that major changes to the 1NN graph occur for $r < 5\%$. In this section we will examine that particular area in more detail. For easier description of the obtained results we rely on the following notation. Let N denote the set of nodes in the 1NN graph that changed their nearest neighbor compared to the nearest neighbor in the unconstrained measure, and let N_C denote the set of nodes in the 1NN graph whose nearest neighbor changed its class compared to the class of the nearest neighbor in the unconstrained measure. Obviously, N_C is a subset of N. Let δ denote the fraction of those modified nodes that have also changed their class:

$$\delta = \frac{|N_C|}{|N|}$$

The values of δ for $r < 5\%$ are given in Tables 5.3, 5.5, 5.7 and 5.9 for DTW, LCS, ERP and EDR. Within these tables, datasets with the highest δ values (greater than or equal to 50%) are marked with symbol ●, and those with the lowest values (smaller than or equal to 10%) with symbol ○. The dash sign among the results in these tables indicates that there are no changes in the nearest neighbor graph compared to the unconstrained similarity measure.

These datasets were excluded from consideration when we calculated the p values for the pairwise Wilcoxon sign-rank test of the differences in δ values across the datasets

Dynamic Time Warping (DTW). In case of DTW (Figure 5.10), nearest neighbors with changed classes are beginning to appear immediately with the first changes in the structure of the 1NN graph: when the width of the warping window drops to about 60% of the length of time series. The percentage of neighbors with changed class increases as the width of the Sakoe-Chiba band narrows, and for some datasets reaches values higher than 40% of the number of time series in the dataset (*haptics*, *inlineskate*).

Looking at Table 5.3 we can see that for $r < 5\%$ on average only about 22% of the changed nodes have modified their classes compared to the unconstrained measure. Changes greater than 50% are only present for three datasets: *adiac*, *haptics* and *inlineskate*. On the other hand, for one fourth of the observed datasets, δ is less than 10%. This is somewhat surprising since in the first part of the experiments we have found significant changes in the structure of the 1NN graph for $r = 0\%$ (about 25%–98%) for all datasets (except *chlorineconcentration* and *beef*).

Observing the p values obtained with the Wilcoxon sign-rank test (Table 5.4), we can notice statistically significant differences between the δ values computed for $r = 4\%$, $r = 3\%$ and $r = 2\%$.

	Dataset	4%	3%	2%	1%	0%		Dataset	4%	3%	2%	1%	0%
	50words	39.57	38.75	38.88	41.11	43.49	○	***mallat***	*3.64*	*3.66*	*3.68*	*3.34*	*3.09*
●	**adiac**	**60.00**	**74.07**	**58.06**	**54.85**	**50.44**		**medicalimages**	31.71	31.91	34.72	34.35	34.35
	beef	16.67	16.67	25.00	33.33	50.00		**motes**	15.52	15.51	13.87	13.77	13.77
	car	50.00	42.00	27.54	28.21	34.12		**noninvasivefatalecg_thorax1**	37.11	31.88	29.34	27.16	25.26
○	***cbf***	*0.00*	*0.13*	*0.00*	*0.11*	*1.02*		**noninvasivefatalecg_thorax2**	28.73	24.56	23.47	20.58	15.87
○	***chlorineconcentration***	*0.00*	*0.00*	*0.00*	*0.00*	*0.00*		**oliveoil**	0.00	33.33	28.57	25.00	11.76
○	***cinc_ecg_torso***	*9.85*	*7.58*	*4.68*	*2.66*	*1.62*		**osuleaf**	50.00	47.54	46.28	47.06	50.45
	coffee	25.00	42.86	41.67	46.15	42.86		**plane**	33.33	20.00	10.00	7.41	4.17
	cricket_x	32.45	31.62	30.80	31.43	41.69	○	***sonyaiborobotsurface***	*5.70*	*5.70*	*4.05*	*2.61*	*2.61*
	cricket_y	31.02	31.02	31.93	34.89	42.66		**sonyaiborobotsurfaceii**	11.06	8.01	8.01	5.96	5.96
	cricket_z	31.88	30.82	32.76	32.59	42.60		**starlightcurves**	9.22	9.19	10.13	11.10	12.49
	diatomsizereduction	33.33	25.00	8.33	1.52	0.40		**swedishleaf**	30.24	27.52	28.65	29.00	32.02
	ecg200	25.00	21.84	21.70	18.52	18.52	○	***symbols***	*1.45*	*1.50*	*1.59*	*2.56*	*3.33*
○	***ecgfivedays***	*2.04*	*1.27*	*0.90*	*1.28*	*1.36*	○	***synthetic_control***	*2.07*	*2.09*	*2.09*	*8.24*	*8.24*
	faceall	13.53	6.81	5.12	4.88	5.90		**trace**	0.00	0.00	2.54	5.22	14.77
	facefour	27.27	16.67	12.90	10.00	8.33	○	***twoleadecg***	*0.00*	*0.29*	*0.35*	*0.45*	*0.45*
	fish	41.67	36.75	29.95	28.03	29.62	○	***twopatterns***	*0.00*	*0.04*	*0.10*	*0.35*	*1.06*
	gun_point	22.54	17.65	16.00	12.95	12.00		**uwavegesturelibrary_x**	27.15	27.08	27.14	27.23	27.59
●	**haptics**	**69.06**	**65.75**	**67.11**	**67.33**	**68.93**		**uwavegesturelibrary_y**	37.83	37.21	36.86	37.06	37.38
●	**inlineskate**	**62.73**	**64.21**	**64.53**	**66.23**	**64.23**		**uwavegesturelibrary_z**	32.77	32.57	32.55	33.63	34.45
○	***italypowerdemand***	*5.45*	*5.45*	*5.45*	*5.45*	*5.45*	○	***wafer***	*3.10*	*2.57*	*2.23*	*1.13*	*0.70*
	lighting2	12.16	17.95	22.58	23.23	29.13		**wordssynonyms**	37.71	36.99	36.59	38.89	41.33
	lighting7	32.94	38.89	36.89	39.29	43.80		**yoga**	20.44	17.62	13.53	10.79	9.60

	4%	3%	2%	1%	0%
Average	23.11%	22.84%	21.29%	21.24%	22.37%

Table 5.3. δ values for DTW

Tabela 5.3. δ vrednosti za DTW

	3%	2%	1%	0%
4%	0.02497	0.03726	0.18922	0.86563
3%		0.02521	0.26009	0.90867
2%			0.84731	0.27520
1%				0.14337

Table 5.4. p values for the pairwise Wilcoxon sign-rank test of the differences in δ values across the datasets for DTW

Tabela 5.4. p vrednosti dobijene uparenim Vilkoksonovim testom rangova sa znakom nad δ vrednostima na svim skupovima podataka za DTW

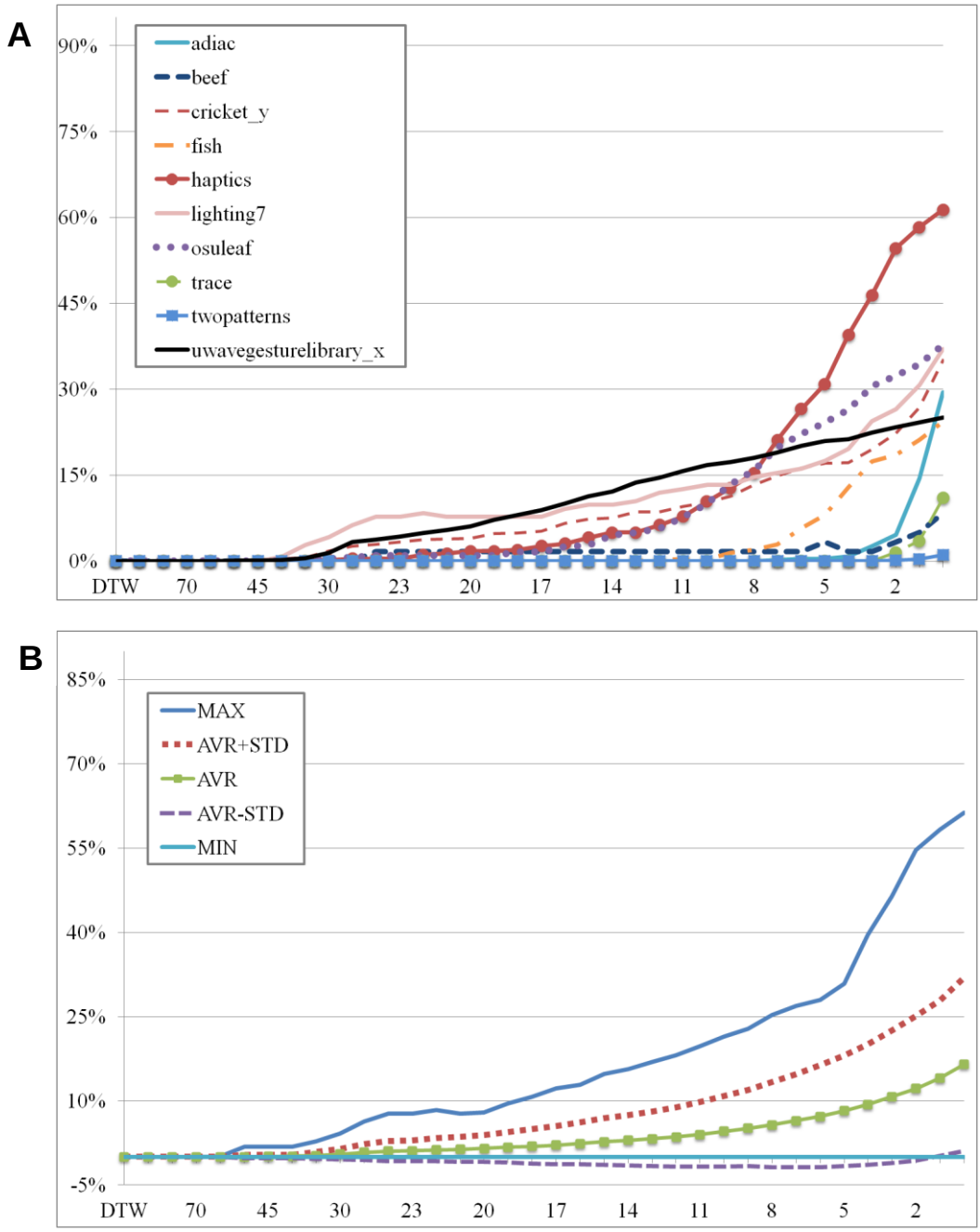


Figure 5.10. Change of classes for DTW

Slika 5.10. Promena klasa za DTW

Longest Common Subsequence (LCS). In accordance with the structure of the nearest neighbor graph for LCS, neighbors with altered classes begin to occur when the width of the warping window reaches 25%, but their number starts to grow significantly only when it drops below 10% (Figure 5.11).

For $r = 0\%$ the changes are more noticeable than with DTW, for a number of datasets they even reach values above 70% (*cricket_x*, *cricket_y*, *cricket_z*). For *lighting2*, *starlightcurves*, *symbols*, *synthetic*_control, and *yoga* major changes in the 1NN graph at $r = 0\%$ (greater than 90%) are accompanied by significantly smaller changes in terms of classes (smaller than 40%). This indicates that only a smaller part of the changed nodes also changed their class.

Comparing the results in Table 5.5 and Table 5.3 we can see that the average δ values for LCS are noticeably higher than for DTW, especially for $r = 0\%$, where it is almost twice as high (the Wilcoxon sign-rank test indicates significance for $r = 1\%$ and $r = 0\%$, with p values of 0.025467 and 3.34E-08, respectively). This means that changes in the structure of the LCS 1NN graph induced by applying the Sakoe-Chiba band more significantly alter the classes of nodes than for DTW.

Another noticeable difference between these two similarity measures refers to the presence of greater fluctuations of δ values for some datasets using LCS (*cbf*, *coffee*, *diatomsizereduction*, *faceall*, *facefour*, *synthetic_control,* and *yoga*). There is also some resemblance between LCS and DTW: *adiac*, *haptics* and *inlineskate* are again among the datasets with the highest δ values, and some of the datasets with the lowest δ values are common for these two measures.

According to the Wilcoxon sign-rank test (the corresponding p values are presented in Table 5.6), in case of LCS, the differences between the δ values across the datasets for $r < 5\%$ are significant. The only exception is between $r = 2\%$ and $r = 1\%$.

	Dataset	4%	3%	2%	1%	0%		Dataset	4%	3%	2%	1%	0%
	50words	43.55	39.66	37.74	35.83	58.88	○	***mallat***	-	-	-	*0.00*	*3.33*
●	**adiac**	**69.23**	**76.47**	**78.26**	**64.00**	**60.36**		**medicalimages**	41.44	38.07	37.50	55.18	55.18
	beef	-	-	-	66.67	84.38		**motes**	14.29	10.91	7.17	20.05	20.05
	car	42.86	28.57	23.81	21.25	74.11		**noninvasivefatalecg_thorax1**	87.50	100.00	72.00	50.50	39.83
	cbf	0.46	0.52	1.34	2.53	62.66		**noninvasivefatalecg_thorax2**	85.71	90.00	83.33	54.95	29.94
	chlorineconcentration	-	-	-	16.78	9.58		**oliveoil**	-	-	-	100.00	64.29
○	***cinc_ecg_torso***	-	-	*0.00*	*2.94*	*4.81*		**osuleaf**	35.71	36.11	35.13	39.64	64.72
	coffee	-	-	50.00	100.00	25.00		**plane**	11.11	15.38	4.55	2.50	8.64
	cricket_x	58.82	52.87	42.49	39.52	79.01		**sonyaiborobotsurface**	0.00	0.00	3.85	14.55	14.55
	cricket_y	41.96	35.85	31.43	31.99	81.42		**sonyaiborobotsurfaceii**	23.53	27.12	27.12	17.47	17.47
	cricket_z	55.08	50.00	45.56	39.83	79.42		**starlightcurves**	11.34	11.34	11.44	11.34	18.19
	diatomsizereduction	100.00	66.67	40.00	7.69	9.43		**swedishleaf**	12.32	11.74	14.16	17.11	35.84
	ecg200	30.30	26.67	19.74	22.29	22.29		**symbols**	3.55	3.80	4.24	4.86	39.46
	ecgfivedays	-	-	-	-	23.53		**synthetic_*control***	8.47	7.05	7.05	31.32	31.32
	faceall	18.31	5.65	4.28	4.05	28.50		**trace**	-	-	50.00	45.16	15.38
	facefour	40.00	12.50	7.41	2.78	51.43	○	***twoleadecg***	*0.00*	*0.00*	*0.00*	*3.82*	*3.82*
	fish	46.94	32.58	28.67	27.35	63.16		**twopatterns**	*0.12*	*0.13*	*0.26*	*0.78*	*70.22*
○	***gun_point***	*0.00*	*0.00*	*4.55*	*3.85*	*8.40*		**uwavegesturelibrary_x**	44.40	42.21	38.28	34.20	43.11
●	**haptics**	**77.31**	**66.50**	**65.53**	**65.09**	**71.37**		**uwavegesturelibrary_y**	42.24	37.26	35.55	34.16	53.97
●	**inlineskate**	**51.39**	**59.50**	**64.13**	**63.16**	**64.94**		**uwavegesturelibrary_z**	38.26	35.57	33.51	34.22	60.30
○	***italypowerdemand***	*10.00*	*10.00*	*10.00*	*10.00*	*10.00*		**wafer**	50.00	22.22	12.82	1.34	1.01
	lighting2	30.00	36.00	22.58	22.22	40.00		**wordssynonyms**	40.63	36.93	34.48	33.27	61.74
	lighting7	60.71	60.61	46.34	44.90	66.41		**yoga**	19.73	10.11	9.97	8.74	31.07

	4%	3%	2%	1%	0%
Average	35.46%	31.49%	27.96%	29.11%	40.49%

Table 5.5. δ values for LCS

Tabela 5.5 δ vrednosti za LCS

	3%	2%	1%	0%
4%	0.02197	0.00023	0.00255	0.00692
3%		0.00076	0.02272	0.00377
2%			0.14337	0.00058
1%				0.00004

Table 5.6. p values for the pairwise Wilcoxon sign-rank test of the differences in δ values across the datasets for LCS

Tabela 5.6. p vrednosti dobijene uparenim Vilkoksonovim testom rangova sa znakom nad δ vrednostima na svim skupovima podataka za LCS

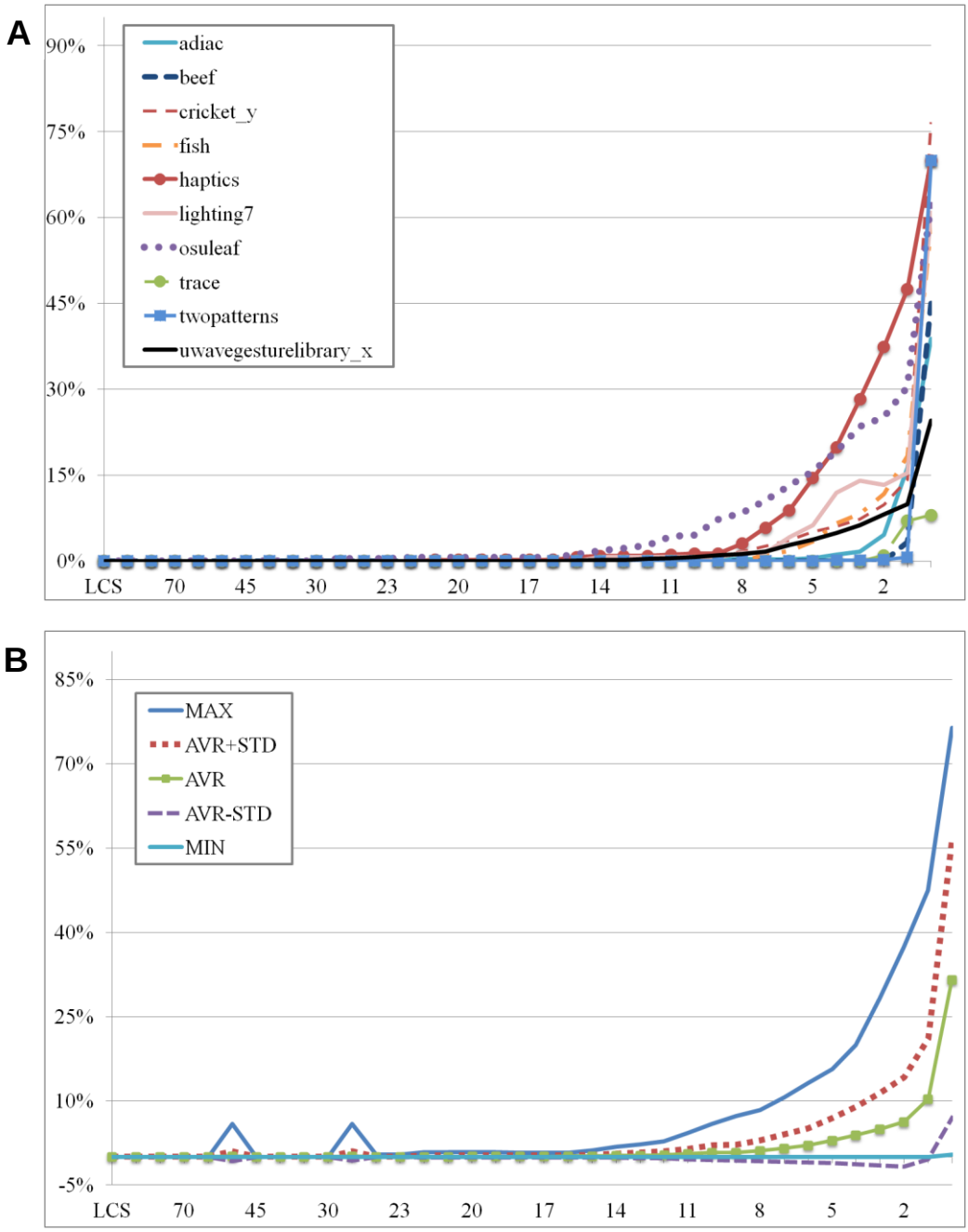


Figure 5.11. Change of classes for LCS

Slika 5.11. Promena klasa za LCS

Edit Distance with Real Penalty (ERP). Changes in the case of ERP (Figure 5.12) start from around $r = 60\%$ and are most visible for *cricket*_y in the same manner as for the 1NN graph (Figure 5.5).

Percentage of nodes in the nearest neighbor graph whose neighbors changed their classes compared to the unconstrained ERP reaches values higher than 50% for a smaller number of datasets (*cricket_x*, *cricket_y*, *cricket_z*, and *lighting7*).

There are datasets (*cbf*, *cinc_ecg_torso*, *ecgfivedays*, *facefour*, *starlightcurves*, *twoleadecg*, *twopatterns*, *yoga*) for which we noticed significant changes in the nearest neighbor graph (higher than 60% for $r = 0\%$) but that produce only minor changes of classes (less than 8%).

The average value of δ decreases as the warping window becomes smaller (Table 5.7). *Adiac*, *haptics* and *inlineskate* are still within the group of datasets which have the highest percentage of nodes with altered classes among the nodes changed by the Sakoe-Chiba band. We can see that several datasets with the lowest δ values reappear also with ERP.

In case of ERP, the differences in δ values are not statistically significant only between $r = 2\%, r = 1\%$ *and* $r = 0\%$. The corresponding p values of the Wilcoxon sign-rank test are shown in Table 5.8.

	Dataset	4%	3%	2%	1%	0%		Dataset	4%	3%	2%	1%	0%
	50words	52.96	48.21	44.95	42.12	43.88	○	***mallat***	-	-	-	*0.00*	*2.51*
●	**adiac**	**-**	**100.00**	**100.00**	**80.00**	**59.54**		**medicalimages**	37.93	37.90	33.67	35.94	35.94
●	**beef**	**57.14**	**62.50**	**62.50**	**60.00**	**60.00**		**motes**	13.33	13.25	12.37	12.96	12.96
	car	100.00	33.33	22.22	27.59	29.17		**noninvasivefatalecg_thorax1**	75.00	64.00	49.15	36.90	22.87
○	***cbf***	*0.00*	*0.00*	*0.00*	*0.00*	*2.60*		**noninvasivefatalecg_thorax2**	87.50	87.50	68.75	31.36	15.21
○	***chlorineconcentration***	-	-	-	-	*0.00*	○	***oliveoil***	-	-	-	-	-
	cinc_ecg_torso	28.26	28.00	9.20	4.18	7.58		**osuleaf**	58.23	53.20	51.00	45.64	48.88
○	***coffee***	-	-	-	-	-		**plane**	0.00	50.00	8.33	5.88	4.50
●	**cricket_x**	**72.92**	**69.85**	**68.64**	**66.80**	**74.34**	○	***sonyaiborobotsurface***	*4.88*	*4.88*	*3.55*	*3.31*	*3.31*
●	**cricket_y**	**89.89**	**88.43**	**87.96**	**88.39**	**91.02**	○	***sonyaiborobotsurfaceii***	*5.80*	*4.23*	*4.23*	*3.47*	*3.47*
●	**cricket_z**	**68.99**	**63.83**	**61.21**	**61.31**	**70.30**		**starlightcurves**	13.14	14.01	13.77	12.76	10.56
	diatomsizereduction	0.00	50.00	33.33	50.00	1.15		**swedishleaf**	16.28	13.37	13.56	20.60	25.33
	ecg200	59.26	45.95	29.85	17.65	17.65		**symbols**	10.89	8.57	5.63	5.01	5.43
○	***ecgfivedays***	*0.00*	*0.00*	*0.98*	*0.88*	*0.73*		**synthetic_control**	5.47	5.02	5.02	19.19	19.19
	faceall	8.59	5.13	3.64	5.62	19.87		**trace**	-	-	0.00	8.70	14.75
	facefour	50.00	20.00	5.88	4.17	4.23	○	***twoleadecg***	*1.33*	*1.37*	*0.30*	*0.93*	*0.93*
	fish	50.00	14.29	28.57	28.38	28.65	○	***twopatterns***	*0.00*	*0.00*	*0.16*	*0.49*	*2.72*
	gun_point	-	-	50.00	11.11	7.46		**uwavegesturelibrary_x**	33.42	30.96	29.41	28.93	29.25
●	**haptics**	**63.64**	**66.94**	**63.64**	**62.71**	**62.85**		**uwavegesturelibrary_y**	40.48	39.08	38.34	37.82	38.82
●	**inlineskate**	**57.36**	**54.40**	**53.79**	**55.65**	**59.77**		**uwavegesturelibrary_z**	36.51	36.36	34.44	33.93	34.35
○	***italypowerdemand***	*8.70*	*8.70*	*8.70*	*8.70*	*8.70*		**wafer**	63.16	40.48	26.09	4.97	1.60
	lighting2	50.65	51.76	48.89	45.54	41.03		**wordssynonyms**	45.29	42.06	37.65	36.02	37.93
●	**lighting7**	**90.63**	**90.00**	**88.35**	**90.16**	**91.91**		**yoga**	22.73	22.12	16.81	11.03	8.13

	4%	3%	2%	1%	0%
Average	37.96%	36.74%	31.54%	28.07%	26.39%

Table 5.7. δ values for ERP

Tabela 5.7. δ vrednosti za ERP

	3%	2%	1%	0%
4%	0.00697	0.00013	0.00069	0.01955
3%		0.00001	0.00053	0.04694
2%			0.10648	0.67407
1%				0.22493

Table 5.8. p values for the pairwise Wilcoxon sign-rank test of the differences in δ values across the datasets for ERP

Tabela 5.8. p vrednosti dobijene uparenim Vilkoksonovim testom rangova sa znakom nad δ vrednostima na svim skupovima podataka za ERP

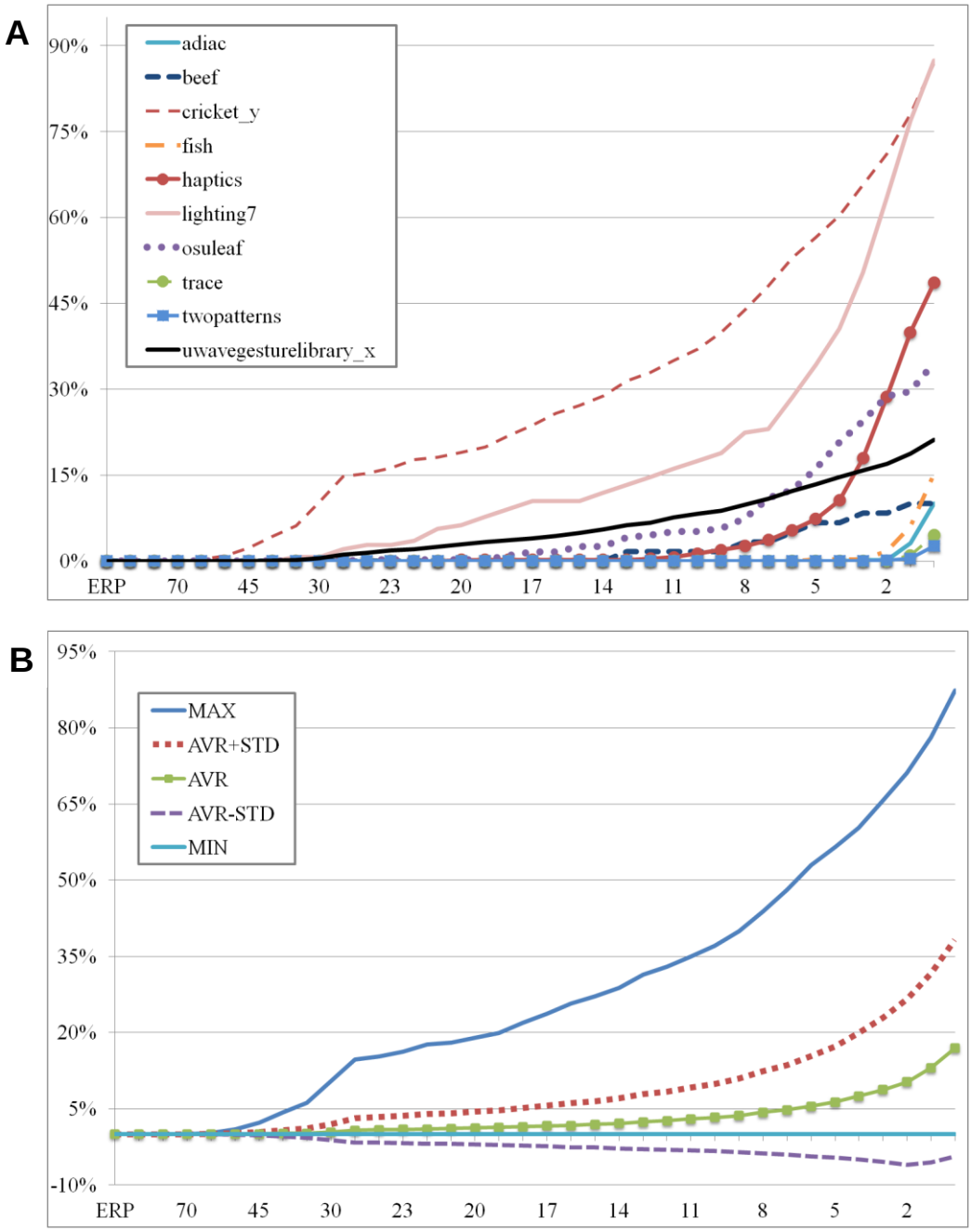


Figure 5.12. Change of classes for ERP

Slika 5.12. Promena klasa za ERP

Edit Distance on Real sequence (EDR). In compliance with the results for the EDR measure from Section 5.3.1, nodes with changed classes begin to appear when the width of the warping window is reduced below 20% of the length of the time series (Figure 5.13). Changes greater than one percent arise only for $r < 12\%$.

For constraint values close to zero, the highest number of nodes with changed classes emerges in case of *haptics* and *inlineskate*. They are the only two datasets that achieve changes larger than 40%.

In terms of results obtained for δ (Table 5.9), EDR most closely resembles ERP: they have very similar values and for both of them δ decreases as we reduce the width of the global constraint (the Wilcoxon sign-rank test reveals no significant difference).

Adiac, *haptics* and *inlineskate* retain their place among the datasets with the highest δ values, and a larger number of datasets with very small changes among the classes are repeated for EDR, too.

Interestingly, only EDR generates large δ values for the *symbols* dataset, all the other similarity measures yield values less than 11% (the only exception is LCS with $r = 0\%$, which gives almost 40%).

Based on the p values of the Wilcoxon sign-rank test (shown in Table 5.10), we can see, that statistically not significant differences in values are present between the following pairs of the constraint value: $r = 4\%$ and $r = 3\%$, $r = 2\%$ and $r = 0\%$ and $r = 1\%$ and $r = 0\%$.

	Dataset	4%	3%	2%	1%	0%		Dataset	4%	3%	2%	1%	0%
	50words	40.95	37.99	34.26	31.91	34.00	○	***mallat***	-	-	-	-	*0.00*
●	**adiac**	**75.00**	**62.50**	**85.71**	**66.67**	**53.26**		**medicalimages**	29.41	32.54	37.43	42.83	42.83
●	**beef**	**-**	**100.00**	**100.00**	**66.67**	**83.33**	○	***motes***	*9.84*	*9.62*	*7.77*	*7.38*	*7.38*
	car	71.43	31.58	20.51	19.74	31.18		**noninvasivefatalecg_thorax1**	80.00	85.71	76.92	65.75	28.11
○	***cbf***	*0.98*	*0.68*	*0.71*	*1.15*	*3.17*		**noninvasivefatalecg_thorax2**	50.00	66.67	80.00	54.76	17.63
○	***chlorineconcentration***	-	-	-	-	*0.00*		**oliveoil**	-	-	-	-	66.67
○	***cinc_ecg_torso***	-	-	*0.00*	*0.00*	*0.20*		**osuleaf**	31.61	31.36	33.08	37.92	48.69
	coffee	-	0.00	0.00	100.00	28.57	○	***plane***	*0.00*	*0.00*	*0.00*	*0.00*	*0.93*
	cricket_x	55.56	55.07	44.52	40.96	42.90		**sonyaiborobotsurface**	25.00	25.00	4.76	5.94	5.94
	cricket_y	44.83	50.00	41.09	39.22	38.70	○	***sonyaiborobotsurfaceii***	*0.00*	*7.14*	*7.14*	*7.64*	*7.64*
	cricket_z	57.50	53.57	44.79	40.82	41.33		**starlightcurves**	12.80	14.57	14.33	13.17	12.33
	diatomsizereduction	100.00	50.00	50.00	5.00	1.01		**swedishleaf**	8.75	9.55	13.25	17.25	26.04
	ecg200	33.33	37.50	25.00	10.87	10.87		**symbols**	66.67	42.86	42.86	32.00	28.33
○	**ecgfivedays**	-	-	-	-	*0.00*		**synthetic_control**	13.51	18.10	18.10	20.12	20.12
○	***faceall***	*7.84*	*5.97*	*3.60*	*4.24*	*5.02*		**trace**	-	-	0.00	47.83	33.90
	facefour	50.00	11.11	4.17	2.08	5.00	○	***twoleadecg***	-	*0.00*	*0.00*	*1.31*	*1.31*
	fish	47.62	28.21	27.84	24.88	35.07	○	***twopatterns***	*0.08*	*0.21*	*0.27*	*0.72*	*1.71*
○	***gun_point***	*0.00*	*0.00*	*10.00*	*4.92*	*5.05*		**uwavegesturelibrary_x**	46.82	40.94	34.79	30.56	29.22
●	**haptics**	**70.79**	**66.43**	**61.99**	**65.13**	**65.14**		**uwavegesturelibrary_y**	37.24	36.27	35.58	33.70	35.52
●	**inlineskate**	**59.76**	**58.36**	**56.74**	**58.89**	**66.16**		**uwavegesturelibrary_z**	33.84	32.56	30.84	32.08	35.34
○	***italypowerdemand***	*5.56*	*5.56*	*5.56*	*5.56*	*5.56*		**wafer**	50.00	50.00	28.57	4.85	0.88
	lighting2	-	100.00	66.67	50.00	27.03		**wordssynonyms**	48.05	48.25	45.29	43.28	41.31
	lighting7	-	50.00	50.00	30.43	40.00		**yoga**	18.95	17.55	12.27	9.85	9.75

	4%	3%	2%	1%	0%
Average	36.68%	34.33%	29.91%	28.05%	24.44%

Table 5.9. δ values for EDR

Tabela 5.9. δ values for EDR

	3%	2%	1%	0%
4%	0.14993	0.01437	0.00239	0.00430
3%		0.02741	0.00188	0.02742
2%			0.02019	0.40218
1%				0.53117

Table 5.10. p values for the pairwise Wilcoxon sign-rank test of the differences in δ values across the datasets for EDR

Tabela 5.10. p vrednosti dobijene uparenim Vilkoksonovim testom rangova sa znakom nad δ vrednostima na svim skupovima podataka za EDR

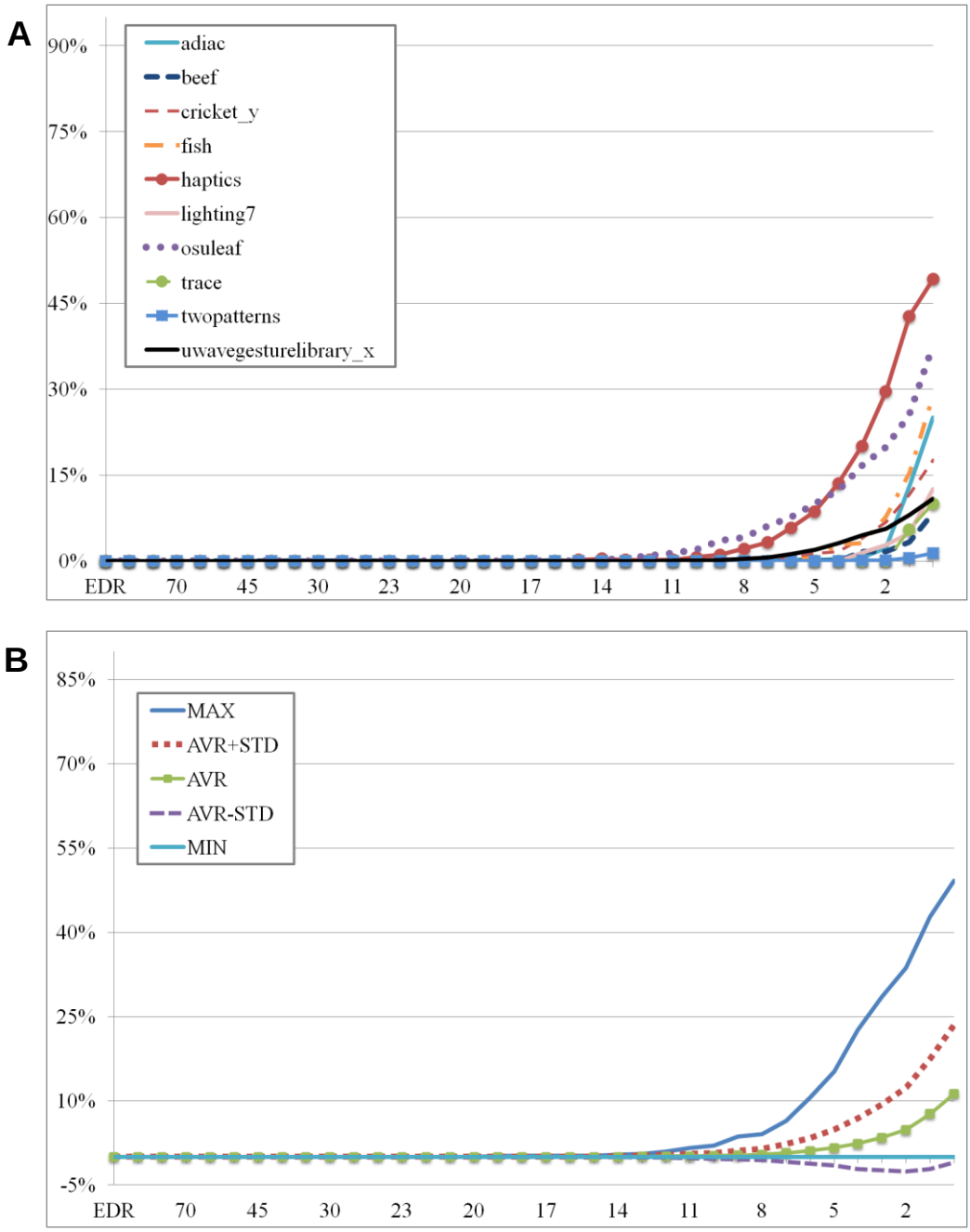


Figure 5.13. Change of classes for EDR

Slika 5.13. Promena klasa za EDR

In the second part of the experiments we have seen that the properties identified by analyzing the structure of the nearest neighbor graphs are, in general, replicated among the data describing the change of classes. The average numbers of changed classes in 1NN graphs are presented in Figure 5.14. We can notice resemblance to the graphs in Figure 5.7 which shows the average number of changed nodes in the nearest neighbor graphs: the biggest changes are induced for DTW, the least ones for EDR, while LCS and ERP are in between. However, there are some differences, too. LCS and ERP provide similar average number of changed neighbors, but the average number of altered classes is higher for ERP. In terms of classes, ERP is closer to DTW and LCS is closer to EDR. The other significant difference is found for small values of r ($< 2\%$). In this area the number of nodes for LCS that have changed their classes rapidly grows, and for $r = 0\%$ LCS overtakes DTW. This confirms that warping window of width $r = 0\%$ has a special impact on LCS.

Looking at the percentage of datasets for which there are nodes in the 1NN graph with changed classes, as shown in Figure 5.15, we can conclude that in case of all four measures the altered classes are beginning to appear immediately with the first changes in the structure of the nearest-neighbor graph. These graphs differ from the graphs shown in Figure 5.8 only slightly. This confirms that the Sakoe-Chiba band affects DTW the most, and that among the discussed similarity measures EDR is least affected.

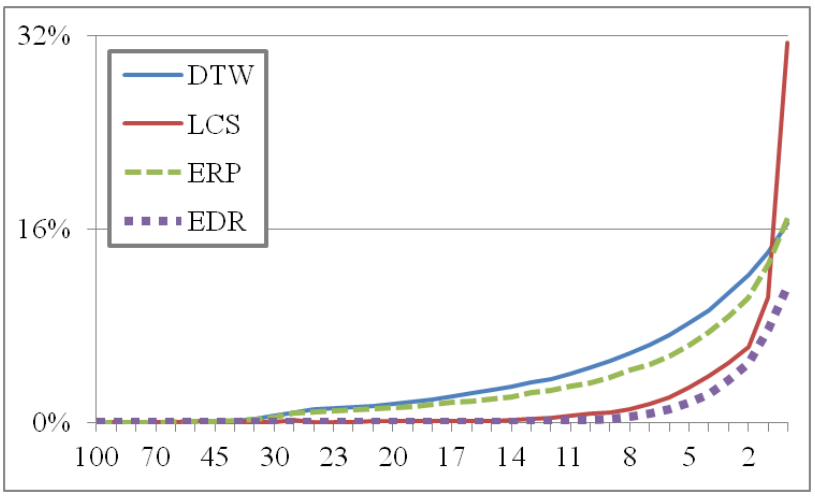


Figure 5.14. The average changes of classes

Slika 5.14. Prosečne promene klasa

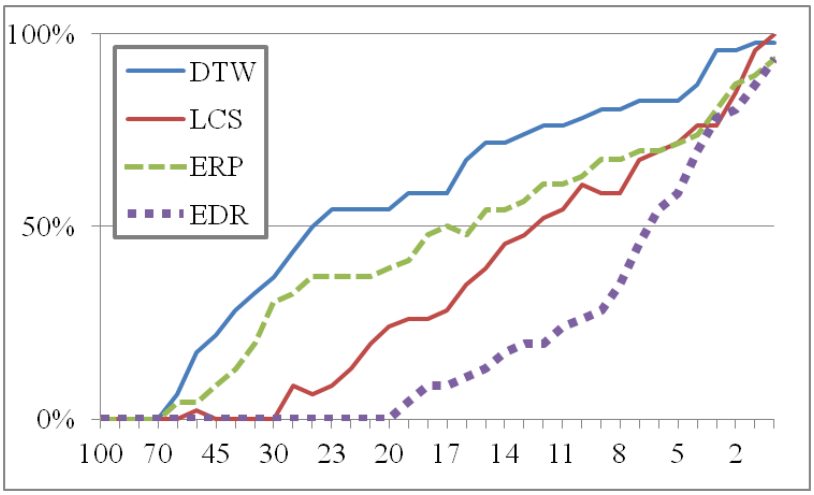


Figure 5.15. The percentage of the datasets with changed classes

Slika 5.15. Procenat skupova podataka sa promenjenim klasama

By analyzing the percentage of those nodes in the 1NN graph which have altered their classes compared to the unconstrained measures, in relation to the total number of changed nodes (regardless of the class), we have seen that (for $r < 5\%$) there are groups of datasets whose members exhibit similar characteristics under the influence of the Sakoe-Chiba band independent of the similarity measure. The first such group consists of the *adiac*, *haptics* and *inlineskate* datasets. Global constraints change their 1NN graphs in such way that a large part (over 50%) of the new neighbors has altered classes. The second group includes *cbf*, *italypowerdemand*, *twoleadecg* and *twopatterns*. These datasets give the lowest δ values (up to 10%) for each of the four measures (the only exception occurs in *cbf* and *twopatterns* with LCS for $r = 0\%$).

In Section 5.3.1 we have seen that there is a group of datasets (including *chlorineconcentration*, *mallat*, *oliveoil* and *trace*) for which LCS, ERP and EDR show only slight changes in the structure of the 1NN graph (or no changes at all) but DTW demonstrates more significant ones. For most of these datasets DTW has very low delta values, this could be the explanation why it has smaller average δ values than the other measures despite the fact that DTW produces the biggest changes in the nearest neighbor graph and among the classes, too.

5.3.3. Impact on Classification Accuracy

The results of Section 5.3.1 and Section 5.3.2 clearly confirmed that the use of the Sakoe-Chiba global constraint causes recognizable changes in the behavior of the four popular elastic similarity measures, especially for small values of the warping window width.

The last phase of our experiments is devoted to analyzing how these changes affect the accuracy of the 1NN classifier.

In the first step we will use stratified 9-fold cross-validation (SCV1x9) to find the smallest values of parameter r for which 1NN produces the smallest classification errors.

In the second step, we will discuss to what extent the 1NN graphs that correspond to the values of parameter r selected in the first step differ from the 1NN graphs of the unconstrained similarity measures.

In the third step, we will give a comparative review of the observed similarity measures based on the classification accuracies obtained by 10 runs of stratified 10-fold cross-validation (SCV10x10) using the values of parameter r from the first step.

We will conclude our analysis in the fourth step by observing some general differences between the studied similarity measures.

The lowest widths of the warping window for which SCV1x9 gives the smallest classification error are presented in Table 5.11 (the same set of r values was searched as in Section 5.3.1 and Section 5.3.2; in case of ties we report the lowest values of r).

On average, this value is lowest in the case of DTW (about 4% of the length of time series) and highest for ERP (almost 10% of the length of time series). Values greater than 10% were observed only for some of the 46 investigated datasets (they are highlighted in boldface in Table 5.11). We have found only two such datasets for DTW (*lighting2* and *motes*), six for EDR, nine for ERP, and ten for LCS – whereas in case of ERP several datasets have values greater than 30% (*cinc_ecg_torso*, *cricket_x*, *cricket_y*, *cricket_z*, *lighting2*, *lighting7*, *motes*).

Overall, in most cases the results obtained for different similarity measures are close to each other, but there are also datasets for which certain similarity measures differ substantially from the others in this respect (*cinc_ecg_torso*, *cricket_z*, *lighting2*, *lighting7*, *osuleaf*).

Dataset	DTW	LCS	ERP	EDR	Dataset	DTW	LCS	ERP	EDR
50words	6	8	4	**15**	**mallat**	4	1	1	0
adiac	1	1	1	1	**medicalimages**	5	10	7	6
beef	2	2	10	4	**motes**	**21**	**12**	**35**	**14**
car	1	8	5	7	**noninvasivefatalecg_thorax1**	0	1	2	6
cbf	2	8	1	5	**noninvasivefatalecg_thorax2**	0	1	3	0
chlorineconcentration	0	0	0	0	**oliveoil**	0	2	0	1
cinc_ecg_torso	2	2	**35**	1	**osuleaf**	5	**23**	**11**	**14**
coffee	3	3	0	2	**plane**	5	5	4	1
cricket_x	7	**12**	**40**	9	**sonyaiborobotsurface**	0	3	5	5
cricket_y	10	**14**	**70**	**12**	**sonyaiborobotsurfaceii**	2	2	2	2
cricket_z	7	**15**	**35**	9	**starlightcurves**	10	**14**	2	6
diatomsizereduction	0	1	0	1	**swedishleaf**	3	5	3	8
ecg200	0	2	0	4	**symbols**	7	6	5	6
ecgfivedays	1	1	0	0	**synthetic_control**	9	**17**	7	7
faceall	3	7	6	4	**trace**	4	4	2	2
facefour	3	1	9	4	**twoleadecg**	3	2	**11**	2
fish	1	5	3	6	**twopatterns**	4	5	3	5
gun_point	4	6	3	5	**uwavegesturelibrary_x**	6	9	7	**12**
haptics	5	6	4	6	**uwavegesturelibrary_y**	6	3	3	8
inlineskate	6	10	5	9	**uwavegesturelibrary_z**	3	**16**	4	**15**
italypowerdemand	0	5	0	0	**wafer**	0	4	3	0
lighting2	**13**	**15**	**45**	1	**wordssynonyms**	6	**12**	5	7
lighting7	5	5	**40**	4	**yoga**	2	7	5	7

	DTW	LCS	ERP	EDR
Average	4.07	6.54	9.70	5.28

Table 5.11. The values of parameter *r* for which SCV1x9 give the smallest error rate

Tabela 5.11. Vrednosti parametra *r* za koje SCV1x9 daje najmanju grešku

In Section 5.3.1 and Section 5.3.2 we have shown that application of the Sakoe-Chiba band exerts the greatest influence on DTW and the lowest influence on EDR, while the magnitude of its effect on LCS and ERP is somewhere between these two boundary cases. This difference can be perceived among the data in Table 5.12, too. Table 5.12 contains the percentages of those nodes of 1NN graphs which have changed their classes under the influence of the constraint parameter *r* (whose values are taken from Table 5.11), compared to the nodes of the 1NN graphs of the unconstrained measures. The number of datasets for which SCV1x9 gives lowest classification error without changes in the 1NN graph (regarding the classes of the nodes) is smallest for DTW (4) and highest for EDR (25). For LCS and ERP there are 16 such datasets. In order to achieve the best classification accuracy, changes over 10% are most frequently required for DTW (20 datasets), followed by ERP (7 datasets). In case of LCS and EDR, changes of this magnitude are needed only for the *adiac* dataset.

Dataset	DTW	LCS	ERP	EDR	Dataset	DTW	LCS	ERP	EDR
50words	*27.96*	5.75	*19.78*	**0.00**	**mallat**	0.67	**0.00**	**0.00**	**0.00**
adiac	*14.47*	*16.39*	3.07	*13.06*	**medicalimages**	*10.60*	2.10	3.59	1.49
beef	3.33	**0.00**	1.67	**0.00**	**motes**	1.89	0.08	**0.00**	**0.00**
car	*18.33*	**0.00**	**0.00**	**0.00**	**noninvasivefatalecg_thorax1**	*21.70*	1.35	0.77	0.03
cbf	**0.00**	**0.00**	**0.00**	**0.00**	**noninvasivefatalecg_thorax2**	*13.25*	1.33	0.19	7.92
chlorineconcentration	**0.00**	0.37	**0.00**	**0.00**	**oliveoil**	3.33	**0.00**	**0.00**	**0.00**
cinc_ecg_torso	1.41	**0.00**	**0.00**	**0.00**	**osuleaf**	*24.21*	0.45	5.20	0.23
coffee	5.36	**0.00**	**0.00**	**0.00**	**plane**	**0.00**	**0.00**	**0.00**	**0.00**
cricket_x	*13.21*	0.51	0.38	**0.00**	**sonyaiborobotsurface**	1.93	**0.00**	0.81	**0.00**
cricket_y	*10.26*	0.13	**0.00**	**0.00**	**sonyaiborobotsurfaceii**	2.76	1.63	1.43	0.10
cricket_z	*13.08*	0.13	0.26	**0.00**	**starlightcurves**	1.93	0.17	2.45	0.10
diatomsizereduction	0.31	0.62	0.31	0.31	**swedishleaf**	*15.29*	0.98	2.22	**0.00**
ecg200	*12.50*	7.50	*10.50*	1.50	**symbols**	0.29	0.78	0.39	**0.00**
ecgfivedays	0.68	**0.00**	0.57	**0.00**	**synthetic_control**	0.83	**0.00**	2.17	**0.00**
faceall	1.73	0.04	0.49	0.18	**trace**	**0.00**	**0.00**	**0.00**	**0.00**
facefour	2.68	1.79	0.89	0.89	**twoleadecg**	0.09	**0.00**	**0.00**	**0.00**
fish	*21.14*	3.43	0.29	0.29	**twopatterns**	**0.00**	**0.00**	**0.00**	**0.00**
gun_point	8.00	**0.00**	**0.00**	**0.00**	**uwavegesturelibrary_x**	*20.12*	0.96	*10.92*	**0.00**
haptics	*30.89*	8.86	*10.58*	5.83	**uwavegesturelibrary_y**	*26.66*	3.04	*16.66*	1.25
inlineskate	*22.46*	0.46	8.77	2.62	**uwavegesturelibrary_z**	*26.51*	0.76	*14.96*	0.18
italypowerdemand	2.65	0.18	1.09	0.09	**wafer**	0.57	0.06	0.24	0.11
lighting2	4.13	**0.00**	**0.00**	7.44	**wordssynonyms**	*26.30*	1.33	*14.48*	0.55
lighting7	*17.48*	6.29	**0.00**	**0.00**	**yoga**	5.06	0.39	0.21	0.03

	DTW	LCS	ERP	EDR
Average	9.48%	1.48%	2.94%	0.96%

Table 5.12. Percentage of nodes in 1NN graph with changed classes for the values of parameter *r* from Table 5.11, compared to unconstrained measures

Tabela 5.12. Procenat čvorova 1NN grafa sa promenjenim klasama za vrednosti parametra *r* iz tabele 5.11, u poređenju sa neograničenim merama

In order to compare the classification performance of the studied similarity measures we computed 1NN classification errors with the SCV10x10 evaluation method using the results from Table 5.11 as values for the global constraint parameter *r*. The lowest average error is produced by LCS (11.43%), the largest one by ERP (12.44%), and DTW (11.52%) and EDR (11.86%) are in between (Table 5.13). Looking at individual datasets the lowest classification error most often occurs with DTW (21 datasets), then with EDR (12 datasets) followed by LCS (11 datasets) and ERP (7 datasets). The mean value of the differences between the minimum and maximum errors is about 3.82. The biggest differences are with the following datasets: *symbols* (18.65), *osuleaf* (13.55), *car* (10.83), *lighting2* (8.79) and *trace* (8.35).

Dataset	DTW	LCS	ERP	EDR	Dataset	DTW	LCS	ERP	EDR
50words	18.63	16.03 ●	21.27 ○	**15.51** ●	**mallat**	1.20	7.02 ○	**0.63** ●	7.03 ○
adiac	**31.42**	33.18 ○	32.24	32.24	**medicalimages**	**18.71**	22.76 ○	18.72	21.81 ○
beef	47.67	**43.00**	51.17 ○	43.33	**motes**	4.51	**1.45** ●	2.95 ●	1.71 ●
car	18.17	11.42 ●	20.17	**9.33** ●	**noninvasivefatalecg_thorax1**	**15.65**	18.27 ○	16.49 ○	18.14 ○
cbf	0.02	0.02	**0.00**	0.05	**noninvasivefatalecg_thorax2**	**9.48**	11.37 ○	9.61	10.87 ○
chlorineconcentration	**0.27**	0.81 ○	0.60 ○	0.80 ○	**oliveoil**	**11.17**	12.00	12.00	13.00
cinc_ecg_torso	**0.01**	**0.01**	0.23 ○	0.07	**osuleaf**	26.49	**12.94** ●	24.89	12.97 ●
coffee	4.83	**4.47**	11.93 ○	4.57	**plane**	**0.00**	0.05	0.10	0.05
cricket_x	**15.68**	18.91 ○	20.29 ○	18.92 ○	**sonyaiborobotsurface**	1.29	1.55	**0.90**	1.66
cricket_y	**13.73**	15.72 ○	17.36 ○	16.88 ○	**sonyaiborobotsurfaceii**	1.19	1.82 ○	**1.10**	**1.10**
cricket_z	**14.95**	18.01 ○	20.55 ○	19.38 ○	**starlightcurves**	**6.28**	9.23 ○	10.03 ○	9.06 ○
diatomsizereduction	**0.06**	**0.06**	**0.06**	**0.06**	**swedishleaf**	11.32	8.54 ●	9.88 ●	**7.99** ●
ecg200	10.00	9.10	**7.35** ●	9.40	**symbols**	1.62	**1.36**	1.71	20.01 ○
ecgfivedays	0.17	**0.10**	0.25	0.15	**synthetic_control**	**0.45**	2.18 ○	1.02 ○	3.02 ○
faceall	1.45	0.87 ●	0.90 ●	**0.71** ●	**trace**	**0.10**	0.25	8.45 ○	1.80 ○
facefour	3.66	1.08 ●	0.98 ●	**0.89** ●	**twoleadecg**	**0.03**	0.12	0.19 ○	0.11
fish	13.46	9.97 ●	12.77	**7.77** ●	**twopatterns**	**0.00**	0.01	0.01	0.01
gun_point	1.90	**0.60** ●	1.75	**0.60** ●	**uwavegesturelibrary_x**	**19.16**	21.40 ○	20.68 ○	21.34 ○
haptics	53.50	51.34	54.14	**51.31**	**uwavegesturelibrary_y**	**25.66**	30.14 ○	27.97 ○	26.36 ○
inlineskate	44.18	40.80 ●	**36.45** ●	40.31 ●	**uwavegesturelibrary_z**	25.46	**24.72** ●	26.97 ○	26.63 ○
italypowerdemand	3.41	3.50	3.70	**2.79** ●	**wafer**	**0.11**	0.13	0.19 ○	0.12
lighting2	**9.50**	14.54 ○	12.57	18.29 ○	**wordssynonyms**	17.43	**15.51** ●	19.05 ○	18.51
lighting7	**21.17**	26.68 ○	28.37 ○	26.22 ○	**yoga**	4.66	2.88 ●	3.60 ●	**2.58** ●

	DTW	LCS	ERP	EDR
Average	11.52%	11.43%	12.44%	11.86%

Table 5.13. Classification errors obtained for SCV10x10 with the values of parameter *r* from Table 5.11

Tabela 5.13. Greške klasifikacije dobijene za SCV10x10 sa vrednostima parametra *r* iz tabele 5.11

Statistically significant differences in error rates are denoted by symbols ● and ○ in Table 5.13, with the former signifying improvement, and the latter degradation of classifier performance when comparing LCS, ERP and EDR measures with DTW.

For this we employed the corrected resampled t-test [80] which adjusts to the loss in degrees of freedom due to repeated runs of cross-validation, at significance level 0.001. We report DTW as the baseline method in Table 5.13 since it is the recommended best choice of distance measure [21].

In order to assess whether some distance measure can be said to be better than others in the average case, we counted the statistically significant wins and losses according to the corrected resampled t-test, for each distance measure, with the results summarized in Table 5.14. The counts suggest that DTW is generally better (despite having slightly higher

average error rate than LCS), with LCS and EDR tied second, and ERP exhibiting the worst performance.

	Wins	Losses	W–L
DTW	52	33	19
LCS	41	37	4
ERP	31	58	–27
EDR	40	36	4

Table 5.14. Statistically significant wins and losses counts for the 1NN classifier with different distance measures, across all datasets

Tabela 5.14. Broj statističko značajnih pobeda i poraza za 1NN klasifikator sa različitim merama rastojanja, na svim skupovima podataka

On the other hand, when we compare the average error rates across all datasets using the Wilcoxon sign-rank test (as in previous sections), the differences are not particularly strong, as shown in Table 5.15 which contains the corresponding p values. The one possibly significant difference when using this test is between DTW and ERP.

	LCS	ERP	EDR
DTW	0.79738	0.018043	0.4597
LCS		0.09515	0.93264
ERP			0.12063

Table 5.15. p values for the pairwise Wilcoxon sign-rank test of the differences in average error rates across the datasets

Tabela 5.15. p vrednosti dobijene uparenim Vilkoksonovim testom rangova sa znakom nad razlikama prosečnih grešaka na svim skupovima podataka

Overall, based on the statistical tests, we can conclude that there is some evidence to consider DTW as the generally best distance measure, and ERP as the generally worst, but the evidence is not overwhelming. Furthermore, when observing the statistical differences on individual datasets (corrected resampled t-test, 0.001 significance level), for every distance measure there are at least a couple of datasets where the measure is significantly superior to all others. Therefore, the choice of the best distance measure for a particular problem may be different from the generally best case.

Observing the graphs of average classification errors across different values of r (Figure 5.16), the most evident common characteristic of the four discussed similarity measures is that for small widths of the warping window ($< 6\%$) the average classification error steeply increases, and in all four cases reaches its maximum for $r = 0\%$ (DTW: 15.97%, LCS: 33.56%, ERP: 20.67%, EDR: 17.20%).

The largest increase occurs for LCS and the lowest one for DTW. Within the area from $r = 100\%$ to $r = 6\%$ the similarity measures exhibit different behaviors. While in case of

DTW the average classification error almost monotonously decreases from 14.04% to 12.38%, for ERP it almost monotonously increases from 13.03% to 14.63%. While a tendency of growth can also be noticed for LCS and EDR, the changes are very subtle: in case of LCS the average error ranges between 11.62% and 11.98%, and in case of EDR it ranges between 11.87% and 12.10%. This suggests that although DTW can be considered the best general choice according the previous analysis, LCS and EDR could be safer choices because of the less pronounced need for tuning the r parameter.

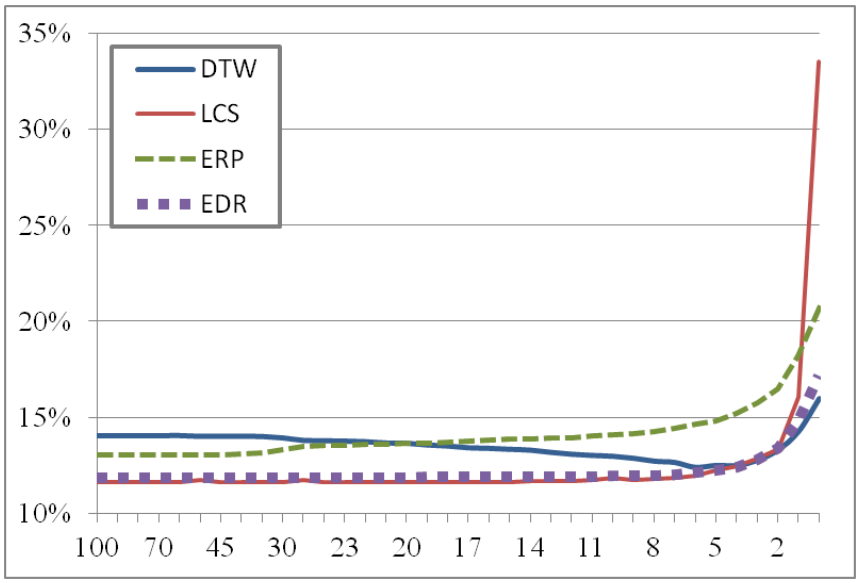


Figure 5.16. Average classification errors for SCV10x10
Slika 5.16. Prosečne greške klasifikacije za SCV10x10

5.4. Improving the Accuracy of the *k*NN Classifier Using Constrained Similarity Measures

Through extensive experiments in this section we will investigate the suggestions and findings regarding the influence of the Sakoe-Chiba band on the two most widely used elastic similarity measures (DTW and LCS) in combination with the 1NN and *k*NN classifiers. We will observe the following widths of the warping window: 100% (the unconstrained similarity measure), 90%, 80%, 70%, 60%, 50%, 45%, 40%, 35%, 30%, and all values from 25% to 0% in steps of 1%. These values were chosen based on our findings in Section 5.3 and on other reports that the measures with larger constraints behave similarly to the unconstrained ones, while the smaller constraints show more apparent discrepancies [21, 98, 130].

We are going to report the minimal value of the warping window that maximizes the classification accuracy of the k-nearest neighbor classifier for a large number of datasets. This classifier is chosen taking into account that among many classification methods (decision trees, neural networks, Bayesian networks, support vector machines, etc.) simple nearest-neighbor methods often give the best results when working with time series [21, 130]. In addition to that, the quality of distance/similarity measure directly influences the

accuracy of the kNN classifier, which makes it appropriate for distance/similarity measure assessment.

To obtain a better insight into the impact of constraining the warping window our experiments encompass five different evaluation methods of classification accuracy: leave-one-out (LOO), stratified 9-fold cross-validation (SCV1x9), 5 times repeated stratified 2-fold cross-validation (SCV5x2), 10 times repeated stratified 10-fold cross validation (SCV10x10) and 10 times repeated stratified holdout method (SHO10x) using two-thirds of available time series for training and one third for testing. The datasets are randomly shuffled in each run. Furthermore, we observe the unweighted and the weighted kNN classifier with the values of parameter k in range from 1 to 30. Weights are calculated by the formula in Eq. (3.5).

5.4.1. Preparing the Experiments

Finding the smallest warping window that provides the lowest error-rate for the kNN classifier for each individual value of parameter k by evaluating the accuracy in five different ways requires computing the similarity between the same time series many times. Calculating these similarities in advance and storing them in the form of distance matrices can significantly speed up the experiments. The distance matrix for one dataset is a matrix where element (i, j) contains the distance between the i-th and the j-th time series from the set.

As the kNN classifier is based on finding k series that are the most similar to a given time series, we can further speed up the experiments by preparing this information in advance. Using a *distance matrix* we can easily sort the time series in order to get the matrix of nearest neighbors. The i-th row of such a matrix contains all the time series of the given dataset (except the i-th series) sorted by their similarity with the i-th series: the first element of the row is the most similar and the last one is the least similar.

After the above preparatory computations we have calculated the classification errors using the above mentioned evaluation methods and the two considered similarity measures (DTW and LCS) relying on the kNN classifier with and without weights for each of the datasets. To get a general picture of the impact of the Sakoe-Chiba band we have searched for the smallest warping window that gives the best error rate and calculated the average values of these warping window widths and the corresponding error-rates for each value of the parameter k of the kNN classifier.

Table 5.16 shows the lowest error rates for the first and last few datasets and a subset of the observed values of parameter k obtained evaluating DTW by LOO and the unweighted kNN classifier (by varying the width of the warping window, i.e. parameter r). The average values of the error rates are given in the last row of the table. The widths of the smallest warping windows given as the percentage of time series length that provide these error rates are presented in Table 5.17.

k	1	2	3	4	…	27	28	29	30
50words	19.01%	19.01%	19.01%	19.78%		35.58%	36.69%	37.02%	37.46%
adiac	31.75%	31.75%	32.52%	32.65%		45.07%	45.84%	46.86%	47.12%
beef	48.33%	48.33%	48.33%	51.67%	…	56.67%	53.33%	55.00%	56.67%
car	17.50%	17.50%	20.00%	20.00%		35.83%	36.67%	36.67%	36.67%
cbf	0.00%	0.00%	0.00%	0.00%		0.00%	0.00%	0.00%	0.00%
⋮			⋮				⋮		
uwavegesturelibrary_y	24.94%	24.94%	23.36%	22.78%		25.77%	25.61%	25.86%	25.82%
uwavegesturelibrary_z	25.08%	25.08%	24.43%	24.30%		28.07%	28.36%	28.45%	28.45%
wafer	0.10%	0.10%	0.13%	0.13%	…	0.46%	0.46%	0.52%	0.52%
wordssynonyms	17.35%	17.35%	17.46%	18.34%		34.25%	35.03%	35.58%	36.46%
yoga	4.33%	4.33%	5.33%	5.33%		11.67%	11.67%	12.18%	12.18%
AVERAGE	**11.23%**	**11.23%**	**11.53%**	**11.73%**	**…**	**18.64%**	**18.75%**	**19.07%**	**19.32%**

Table 5.16. Lowest error rates for DTW obtained by LOO

Tabela 5.16. Najmanje greške klasifikacije za DTW dobijene sa LOO

k	1	2	3	4	…	27	28	29	30
50words	6	6	6	6		8	11	11	10
adiac	1	1	0	1		1	1	2	2
beef	2	2	0	0	…	0	2	0	8
car	1	1	1	1		6	6	6	6
cbf	2	2	1	1		3	3	4	4
⋮			⋮				⋮		
uwavegesturelibrary_y	3	3	9	9		5	7	5	5
uwavegesturelibrary_z	4	4	5	5		5	5	5	7
wafer	0	0	3	3	…	0	0	1	1
wordssynonyms	6	6	6	6		8	9	9	9
yoga	2	2	2	2		2	2	2	2
AVERAGE	**3.78**	**3.78**	**4.59**	**4.52**	**…**	**7.33**	**7.54**	**7.48**	**7.65**

Table 5.17. Smallest warping window widths for DTW obtained by LOO in percentage of the length of time series

Tabela 5.17. Najmanje širine pojasa iskrivljenja za DTW dobijene pomoću LOO, u procentima u odnosu na dužine vremenskih serija

In the rest of this section we will provide an overview of the obtained results. Section 5.4.2 is devoted to the unweighted kNN classifier and Section 5.4.3 describes the behavior of the weighted kNN classifier. Within both sections both elastic similarity measures (DTW and LCS) will be described with the following characteristics:

- The relation between the average value of the lowest classification errors and the different values of parameter k,

- The minimum and maximum average classification errors of the investigated evaluation methods.
- The change of the corresponding average smallest warping window width for the five observed evaluation methods with respect to the parameter k.
- The minimum and maximum average warping window widths along with the values of the parameter k for which these results have been obtained.

5.4.2. The Unweighted kNN Classifier

In this section we analyze the impact of the number of neighbors (parameter k) and the width of the warping window (parameter r) on the accuracy of the unweighted kNN classifier in the case of the constrained DTW and LCS similarity measures. We will show that with the increase of the value of k, the classification accuracy decreases. In addition, the width of the warping window must be constantly widened in order to achieve the best accuracy (as the value of k grows, we need wider and wider warping windows to get the best result).

Dynamic Time Warping (DTW). In Figure 5.17 we can clearly notice that the relationship between the parameter k and the average smallest error rate is almost linear – the growth of parameter k leads to the decline of classification accuracy. The highest average classification accuracy (88.772%) was achieved with the 1NN classifier and the LOO evaluation method and the lowest one (74.536%) with the 30NN classifier and the SCV5x2 evaluation method (Table 5.18).

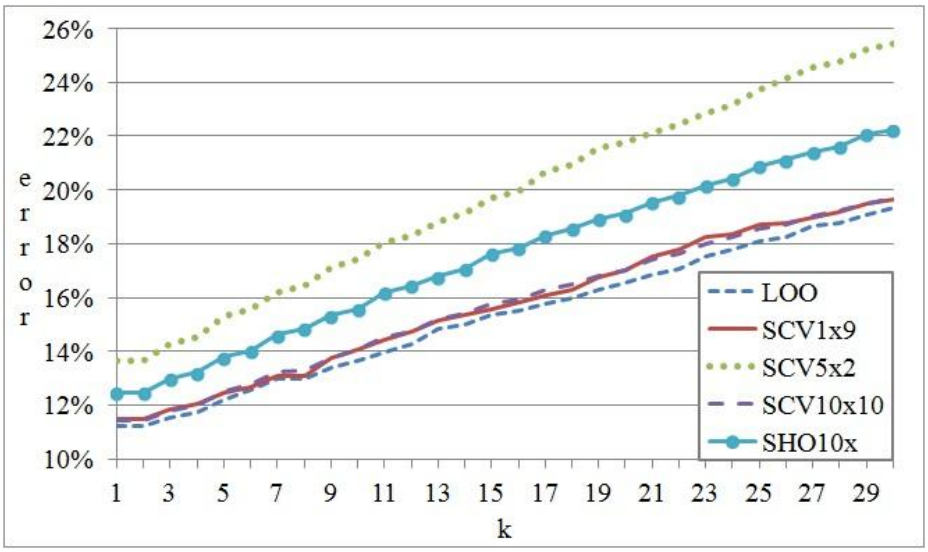


Figure 5.17. Average lowest error rates for DTW with the unweighted kNN classifier

Slika 5.17. Prosečne najmanje greške za DTW sa kNN klasifikatorom bez upotrebe težina

In case of the unweighted kNN classifier the average width of the smallest warping window which gives the lowest error rate for DTW varies in the range from 3.783% to 10.087%. We

can see that the increase of the parameter k implies the growth of the average warping window widths (Figure 5.18): we need wider and wider windows to get the best accuracy. The smallest average warping window (3.783%) was obtained using the LOO evaluation method and the 1NN classifier and the largest one (10.087%) with the SHO10x evaluation method and the 24NN classifier (Table 5.19).

	MIN		MAX		MAX-MIN
	error	k	error	k	
LOO	**11.228%**	**1**	19.317%	30	8.089
SCV1x9	11.494%	1	19.636%	30	8.142
SCV5x2	13.628%	1	**25.464%**	**30**	**11.836**
SCV10x10	11.410%	1	19.701%	30	8.291
SHO10x	12.471%	1	22.223%	30	9.752

Table 5.18. Minimum and maximum of the average lowest error rates for DTW with the unweighted kNN classifier

Tabela 5.18. Minimalne i maksimalne vrednosti proseka najmanjih grešaka klasifikacije za DTW sa kNN klasifikatorom bez upotrebe težina

Figure 5.18. Average smallest warping window widths (r) for DTW with the unweighted kNN classifier

Slika 5.18. Prosečne vrednosti najmanjih širina pojasa iskrivljenja (r) za DTW sa kNN klasifikatorom bez upotrebe težina

	MIN		MAX		MAX-MIN
	r (%)	k	r (%)	k	
LOO	**3.783**	**1**	7.652	30	3.870
SCV1x9	4.065	1	8.587	30	4.522
SCV5x2	4.913	1	9.935	24	5.022
SCV10x10	4.261	1	8.565	21	4.304
SHO10x	4.000	1	**10.087**	**24**	**6.087**

Table 5.19. Minimum and maximum of the average smallest warping window widths for DTW with the unweighted kNN classifier

Tabela 5.19. Minimalne i maksimalne vrednosti proseka najmanjih širina pojasa iskrivljenja za DTW sa kNN klasifikatorom bez upotrebe težina

Longest Common Subsequence (LCS). The relationship between the parameter k and the average classification accuracy is almost linear in the case of LCS, too. The average classification error grows as we increase the value of k (Figure 5.19). All five discussed methods for testing classification accuracy give the smallest average errors with 1NN (they are very close to the results of DTW) and the biggest ones with 30NN (Table 5.20) as in the case of DTW. Again, the best result was obtained by combining LOO with the 1NN classifier and the worst one by combining SCV5x2 with 30NN.

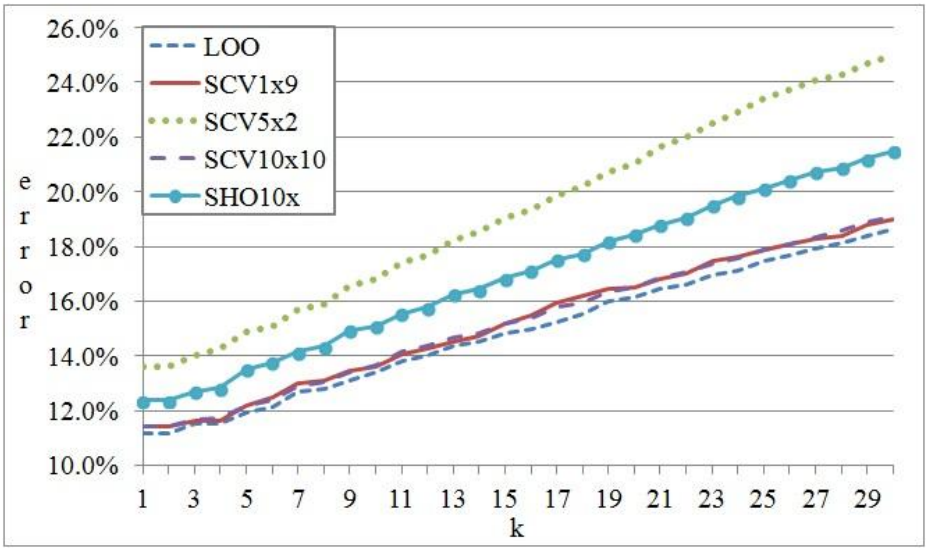


Figure 5.19. Average lowest error rates for LCS with the unweighted kNN classifier

Slika 5.19. Prosečne najmanje greške za LCS sa kNN klasifikatorom bez upotrebe težina

The minimum values of the averages of the smallest warping window widths are greater for LCS than for DTW by 2 or 3 (Table 5.21) and they are achieved for higher values of parameter k (by 3 to 7 — except for SCV10x10). The maximums of the r values are also greater than for DTW but they are achieved for smaller values of k (especially for SCV1x9). This means that while increasing the parameter k we need wider warping windows to get the best classification accuracy (Figure 5.20), the growth tendency is not as clear as in case

of DTW (Figure 5.18). The smallest average r value (5.761%) is produced again by the combination of LOO and 1NN, and the largest one (12.348%) is obtained using SCV5x2 and 21NN (Table 5.21).

	MIN		MAX		MAX-MIN
	error	k	error	k	
LOO	**11.180%**	**1**	18.622%	30	7.442
SCV1x9	11.425%	1	18.970%	30	7.546
SCV5x2	13.601%	1	**24.972%**	**30**	**11.372**
SCV10x10	11.396%	1	19.094%	30	7.698
SHO10x	12.369%	1	21.476%	30	9.106

Table 5.20. Minimum and maximum of the average lowest error rates for LCS with the unweighted kNN classifier

Tabela 5.20. Minimalne i maksimalne vrednosti proseka najmanjih grešaka klasifikacije za LCS sa kNN klasifikatorom bez upotrebe težina

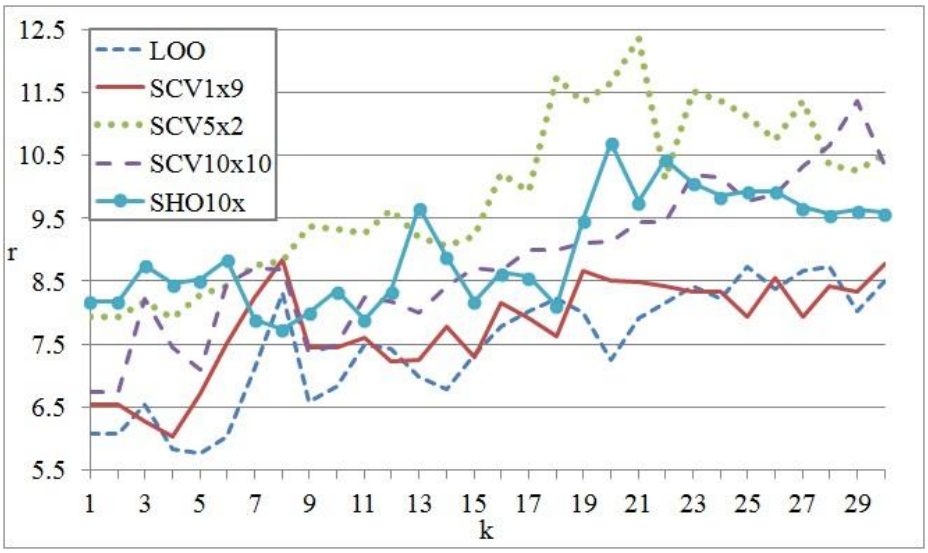


Figure 5.20. Average smallest warping window widths for LCS with the unweighted kNN classifier

Slika 5.20. Prosečne vrednosti najmanjih širina pojasa iskrivljenja za LCS sa kNN klasifikatorom bez upotrebe težina

	MIN		MAX		MAX-MIN
	r (%)	k	r (%)	k	
LOO	**5.761**	**5**	8.739	25	2.978
SCV1x9	6.022	4	8.848	8	2.826
SCV5x2	7.913	4	**12.348**	**21**	4.435
SCV10x10	6.739	1	11.370	29	**4.630**
SHO10x	7.739	8	10.696	20	2.957

Table 5.21. Minimum and maximum of the average smallest warping window widths for LCS with the unweighted kNN classifier

Tabela 5.21. Minimalne i maksimalne vrednosti proseka najmanjih širina pojasa iskrivljenja za LCS sa kNN klasifikatorom bez upotrebe težina

5.4.3. The Weighted kNN Classifier

The aim of this section is to show that, by applying weights in combination with global constraints, the kNN classifier can be made more accurate than the 1NN classifier. In these experiments weights are calculated by the formula in Eq. (3.5).

In case of both considered elastic similarity measures (DTW and LCS) the best accuracy is obtained with values of parameter k around 4. Furthermore, the value of the constraint r remains approximately the same for all values of k. In the remainder of this section we give a detailed overview of the obtained results.

Dynamic Time Warping (DTW). Looking at the chart in Figure 5.21 we can see that in the case of DTW the use of weights changes the influence of the parameter k on the accuracy of classification: instead of 1NN the smallest average error rates were achieved with 3NN (or 4NN in the case of SCV5x2 and SHO10x). After a brief decline and reaching the minimum value, the error rates begin to grow again, similarly as in the case of the unweighted kNN classifier but visibly slower. The attained maximum values of the classification errors are more than 1.5 times less than without weights (Table 5.22). The highest average classification accuracy was achieved by LOO and the lowest one by SCV5x2.

Figure 5.22 shows that the introduction of weights into the kNN classifier noticeably alleviates the growth of the average warping window widths. In this case the largest average warping window (6.848%) was achieved by the combination of the 8NN classifier and the SCV5x2 evaluation method (Table 5.23). The smallest average warping window (3.783%) was obtained using the 1NN classifier and the LOO evaluation method. The differences between the minimum and maximum average r values are about two times smaller than in the case of the unweighted kNN classifier.

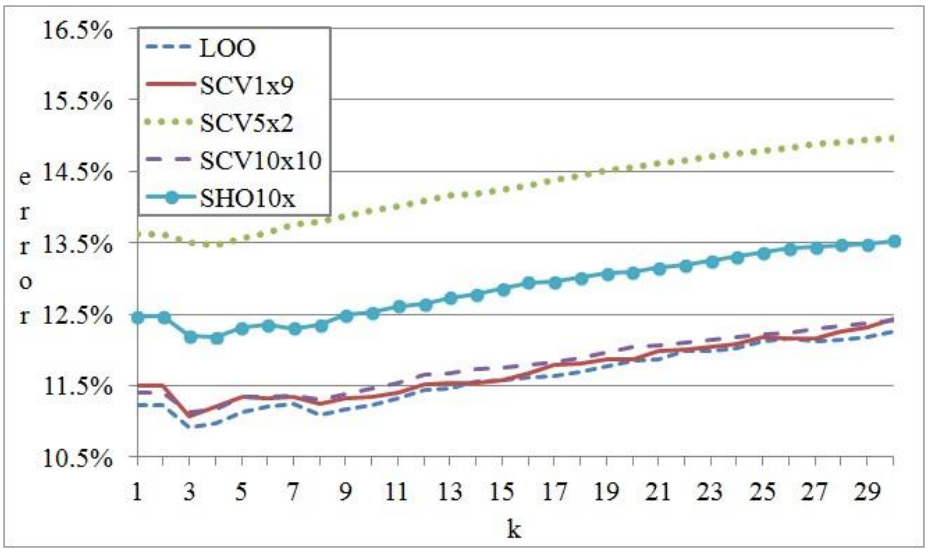


Figure 5.21. Average lowest error rates for DTW with the weighted *k*NN classifier

Slika 5.21. Prosečne najmanje greške za DTW sa *k*NN klasifikatorom sa upotrebom težina

	MIN error	*k*	MAX error	*k*	MAX-MIN
LOO	**10.923%**	**3**	12.256%	30	1.333
SCV1x9	11.072%	3	12.426%	30	1.354
SCV5x2	13.468%	4	**14.970%**	**30**	**1.502**
SCV10x10	11.134%	3	12.412%	30	1.278
SHO10x	12.177%	4	13.527%	30	1.350

Table 5.22. Minimum and maximum of the average lowest error rates for DTW with the weighted *k*NN classifier

Tabela 5.22. Minimalne i maksimalne vrednosti proseka najmanjih grešaka klasifikacije za DTW sa KNN klasifikatorom sa upotrebom težina

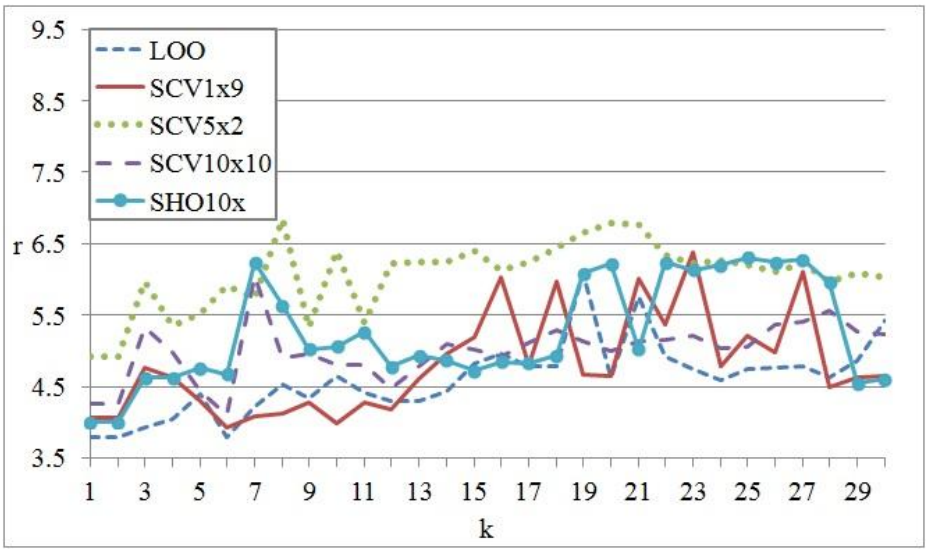


Figure 5.22. Average smallest warping window widths for DTW with the weighted kNN classifier

Slika 5.22. Prosečne vrednosti najmanjih širina pojasa iskrivljenja za DTW sa kNN klasifikatorom sa upotrebom težina

	MIN		MAX		MAX-MIN
	r (%)	k	r (%)	k	
LOO	**3.783**	**1**	6.087	19	2.304
SCV1x9	3.935	6	6.370	23	**2.435**
SCV5x2	4.913	1	**6.848**	**8**	1.935
SCV10x10	4.109	6	6.043	7	1.935
SHO10x	4.000	1	6.304	25	2.304

Table 5.23. Minimum and maximum of the average smallest warping window widths for DTW with the weighted kNN classifier

Tabela 5.23. Minimalne i maksimalne vrednosti proseka najmanjih širina pojasa iskrivljenja za DTW sa kNN klasifikatorom sa upotrebom težina

Longest Common Subsequence (LCS). In working with LCS, applying the weighted voting approach to the kNN classifier shifts the best average accuracy from $k = 1$ to around $k = 4$ (Figure 5.23). Upon reaching the minimum value, the average error rates elevate nearly linearly like in the case of the unweighted classifier. However, as with DTW, the differences between the smallest and largest average error rates are much slighter compared to the results obtained by the simple voting method. The minimum average classification errors are slightly lower than without the weights (Table 5.24). LOO gives the best average accuracy and SCV5x2 the lowest one.

Interestingly, in the case of LCS there is no apparent trend of increasing the average warping window width which is needed to achieve the best accuracy as we increase the value of k (Figure 5.24). We can even spot a minor narrowing period between approximately $k = 10$

and $k = 20$. The smallest average r value (5.761%) was generated by the combination of SCV1x9 and 27NN and the largest one (9.565%) by the combination SCV5x2 and 26NN (Table 5.25). The differences between the minimum and maximum average r values are smaller when using weights compared to the unweighted case.

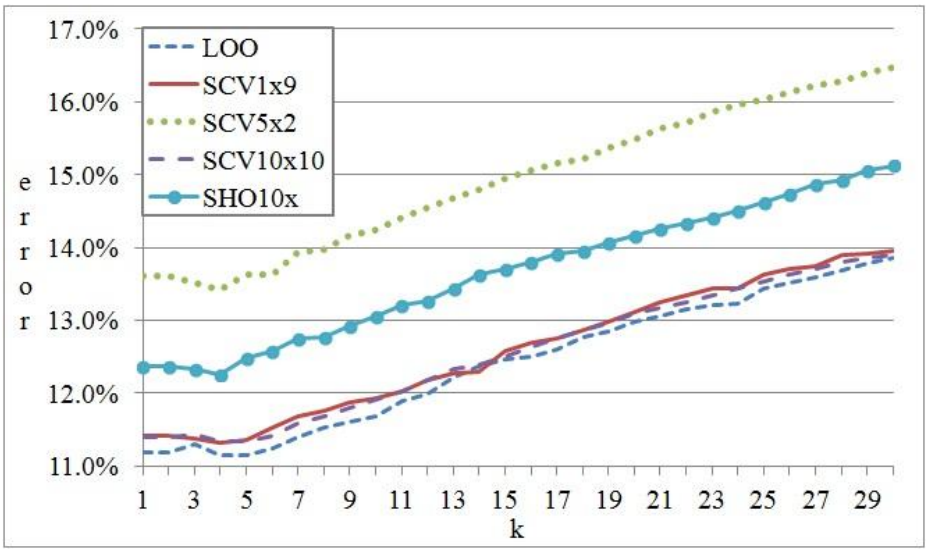


Figure 5.23. Average lowest error rates for LCS with the weighted kNN classifier

Slika 5.23. Prosečne najmanje greške za LCS sa kNN klasifikatorom sa upotrebom težina

	MIN		MAX		MAX-MIN
	error	k	error	k	
LOO	**11.145%**	**5**	13.866%	30	2.721
SCV1x9	11.327%	4	13.953%	30	2.626
SCV5x2	13.423%	4	**16.461%**	**30**	**3.038**
SCV10x10	11.335%	5	13.900%	30	2.565
SHO10x	12.261%	4	15.125%	30	2.864

Table 5.24. Minimum and maximum of the average lowest error rates for LCS with the weighted kNN classifier

Tabela 5.24. Minimalne i maksimalne vrednosti proseka najmanjih grešaka klasifikacije za LCS sa kNN klasifikatorom sa upotrebom težina

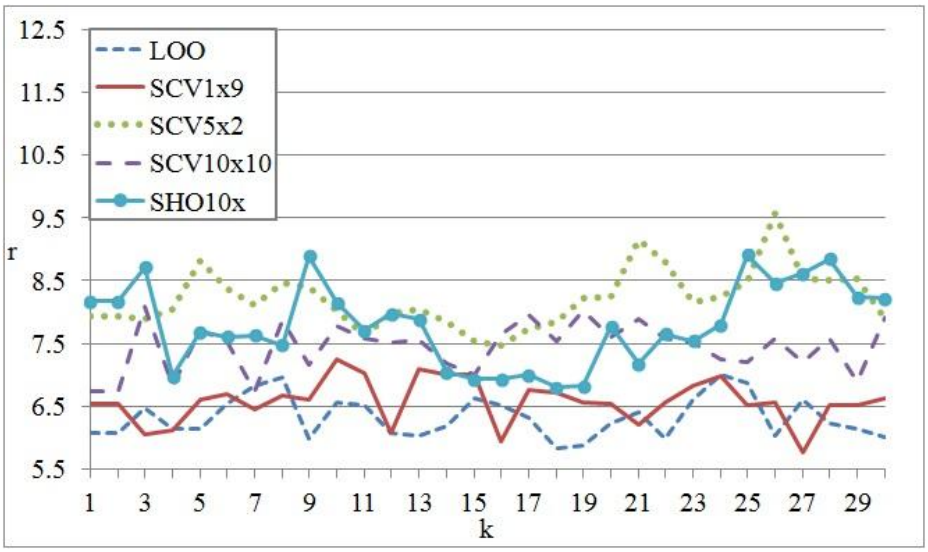


Figure 5.24. Average smallest warping window widths for LCS with the weighted *k*NN classifier

Slika 5.24. Prosečne vrednosti najmanjih širina pojasa iskrivljenja za LCS sa *k*NN klasifikatorom sa upotrebom težina

	MIN		MAX		MAX-MIN
	***r* (%)**	***k***	***r* (%)**	***k***	
LOO	5.826	18	7.000	24	1.174
SCV1x9	**5.761**	**27**	7.239	10	1.478
SCV5x2	7.478	16	**9.565**	**26**	2.087
SCV10x10	6.739	1	8.087	3	1.348
SHO10x	6.804	18	8.913	25	**2.109**

Table 5.25. Minimum and maximum of the average smallest warping window widths for LCS with the weighted *k*NN classifier

Tabela 5.25. Minimalne i maksimalne vrednosti proseka najmanjih širina pojasa iskrivljenja za LCS sa *k*NN klasifikatorom sa upotrebom težina

5.5. Improving the Accuracy of the *k*NN Classifier Using Weights

In Section 3.1 we have seen that a number of different extensions of the nearest-neighbor rule is proposed in the literature in order to improve the accuracy of classification. Assertions about the superiority of a new weighting scheme are often founded on comparing classification errors considering a small number of datasets (commonly from the UCI machine learning repository [7]). Moreover, in all of these experiments the distance between objects is determined solely by Euclidean distance. In the domain of time series, however, besides this *lock-step* measure several other *elastic* measures are also in use (the

most widely used are briefly outlined in Section 2.1). In this section we will report the results of our investigation of various NN classifiers on a large number of datasets from the UCR Time Series Repository [50] considering both the Euclidean and the unconstrained elastic similarity measures (DTW and LCS). These datasets encompass various different domains, including biology, astronomy, robotics, medicine, etc (see Section 4.2). In addition to classification accuracies, we will also provide statistical support to our findings.

In these experiments, the accuracy of classification is obtained by 10 runs of stratified 10-fold cross-validation (SCV10x10) using the best value of parameter k obtained in the range from 1 to 30 by stratified 9-fold cross-validation (SCV1x9) on the training set. Detailed results are shown in Appendix of the dissertation.

The average classification errors of the examined NN classifiers and the average values of the parameter k are presented in Table 5.26. The best results are marked with symbol ●, and the weakest ones with symbol ○. Column *Win* denotes the number of datasets for which the corresponding NN classifier gave the smallest classification error in comparison to the other classifiers. With respect to this category the biggest ones are constantly achieved by the dual distance-weighted kNN rule defined by Eq. (3.11) in Section 3.1 (DTW and LCS) and the DualU weighting scheme defined by Eq. (3.9) (L2 and LCS).

	Euclidean distance			DTW			LCS		
	Win	**Error**	***k***	**Win**	**Error**	***k***	**Win**	**Error**	***k***
1NN	9	0.1595		9 ○	0.1404		18 ○	0.1159	
***k*NN**	6 ○	0.1605	2.99	8	0.1394	2.80	6	0.1140	3.26
Inverse	○ 5	0.1593	3.55	9	0.1373	3.97	4	0.1125	4.19
ISquared	6	0.1586	4.11	10	0.1372	4.88	5	0.1122	4.27
Rank	6	0.1601	4.11	8	0.1399	3.97	8	0.1135	4.41
Fibonacci	6	0.1585	3.94	8	0.1378	4.27	10	0.1126	4.27
Dudani	11	0.1571	6.58	9	0.1369	5.90	13	0.1112	6.62
Macleod	6	0.1601	3.44	○ 7	0.1397	3.37	6	0.1130	3.85
DualD	10 ●	0.1567	6.87	● 19 ●	0.1359	6.56	● 21 ●	0.1103	7.18
Zavrel	10	0.1587	5.16	10	0.1380	5.10	○ 3	0.1133	3.56
Uniform	8	0.1570	6.25	13	0.1362	7.43	9	0.1124	7.22
DualU	● 13	0.1571	13.22	10	0.1369	13.98	● 21	0.1115	12.92

Table 5.26. Comparison of average accuracies of different NN classifiers for the three most commonly used time-series similarity measures

Tabela 5.26. Upoređivanje prosečnih tačnosti različitih NN klasifikatora za tri najčešće korišćenih mera sličnosti vremenskih serija

From Table 5.26 we can notice that in terms of average classification accuracy the simple nearest-neighbor classifier underperforms most of the other forms of the kNN classifier in case of all considered similarity measures. On the other hand, the best result is always obtained using the DualD distance-weighting scheme. It is also worth to notice that the differences between the best and worst average results are not particularly big: 0.0038 (Euclidean distance), 0.0045 (DTW), 0.0056 (LCS).

By comparing the average values of parameter k, it is evident that (in order to achieve the best accuracy) the largest number of neighbors is required by the DualU weighting scheme (Eq. (3.9)): the average value is greater than 12 for all investigated similarity measures. The required set of the closest objects is smallest in case of the majority voting NN rule - the mean value is less than 4 for all of the three measures.

In the rest of this section we will describe the results obtained for each similarity measure in more detail. We will provide a summary of comparison of the simple nearest-neighbor rule and the other versions of the kNN classifier in terms of the number of datasets for which they have achieved better and worse classification accuracies (Tables 5.27, 5.30 and 5.33). The average differences of the classification accuracies between the simple nearest-neighbor rule and the other considered classifiers are also listed in these tables.

In order to see whether there is a statistically significant difference between the analyzed classifiers, we counted the statistically significant wins and losses according to the corrected resampled t-test [11] at significance level 0.001, for each classifier. These results are presented in Tables 5.28, 5.31 and 5.34. In addition, we compared the average error rates across all datasets using the Wilcoxon sign-rank test [29]. The obtained p values are shown in Tables 5.29, 27.32 and 5.35. The rows and columns of these tables are sorted by overall average error rates (cf. Table 5.26), with the upper triangle of the symmetrical matrix of p values shown, thus enabling the reader to easily inspect the p values for methods to which a given method is supposedly superior to (reading by rows) or inferior to (reading by columns).

5.5.1. Euclidean distance

From Table 5.27 we can see that, compared to the simple nearest neighbor rule, among all of the observed classifiers, the Dudani, the DualD and the DualU weighting schemes gave better accuracies for the largest number of datasets (28, 28 and 25, respectively) and that the kNN classifier and the Rank weighting scheme gave the largest number of datasets with higher classification error than 1NN (26 and 25). The largest number of statistically significant wins (Table 5.28) and the smallest number of losses were produced by the Dudani weighting scheme. The DualD and the DualU schemes provide the second and the third best results in this comparison too. We can also notice that the 1NN rule has the smallest number of statistically significant wins and that the majority voting k-nearest neighbor rule gave the second worst result.

	***k*NN**	**Inverse**	**ISquared**	**Rank**	**Fibonacci**	**Dudani**	**Macleod**	**DualD**	**Zavrel**	**Uniform**	**DualU**
Better than 1NN	11	13	16	11	18	**28**	14	**28**	19	19	25
Average diff.	0.0092	0.0102	0.0100	**0.0118**	0.0063	0.0083	0.0094	0.0084	0.0078	0.0096	0.0054
Worse than 1NN	**26**	24	24	25	22	15	24	15	19	22	12
Average diff.	0.0056	0.0050	0.0049	0.0061	0.0030	**0.0082**	0.0067	0.0072	0.0056	0.0030	0.0017

Table 5.27. Comparison of kNN with 1NN in case of Euclidean distance

Tabela 5.27. Upoređivanje kNN-a i 1NN-a u slučaju Euklidskog rastojanja

It can be seen in Table 5.29 that according to the sign-rank test the DualD scheme exhibits significant superiority to all schemes except DualU and Dudani. On the other hand, the error rates of Dudani are judged to be significantly better than all other schemes (except DualD and DualU). The DualU scheme also performed strongly according to this test, exhibiting very low p values against most methods with lower average error rates, with the exceptions being Dudani and Zavrel. In all, it can be said that both statistical testing methodologies agree that DualD, Dudani and DualU are the best weighting schemes to use in combination with Euclidean distance, and that Uniform is the fourth best behind the mentioned three.

	Wins	Losses	W-L
NN	17	88	-71
***k*NN**	10	80	-70
Rank	21	57	-36
Macleod	20	47	-27
Inverse	20	41	-21
Fibonacci	35	50	-15
Zavrel	45	51	-6
ISquared	36	33	3
Uniform	53	26	27
DualU	72	30	42
DualD	98	12	86
Dudani	97	9	88

Table 5.28. Statistically significant wins and losses counts for different NN classifiers, across all datasets, in case of Euclidean distance

Tabela 5.28. Brojevi statističko značajnih pobeda i poraza različitih NN klasifikatora, na svim skupovima podataka, u slučaju Euklidskog rastojanja

	ERROR	0.1567	0.1570	0.1571	0.1571	0.1585	0.1586	0.1587	0.1593	0.1595	0.1601	0.1601	0.1605
ERROR		**DualD**	**Uniform**	**DualU**	**Dudani**	**Fibonacci**	**Isquared**	**Zavrel**	**Inverse**	**NN**	**Rank**	**Macleod**	***k*NN**
0.1567	**DualD**		0.06381	0.48380	0.38596	0.00413	0.00119	0.01538	1.00E-03	0.014	0.00188	0.00021	0.00054
0.1570	**Uniform**			0.18211	0.04543	0.06766	0.02760	0.91229	0.03147	0.35082	0.04191	0.02948	0.00794
0.1571	**DualU**				0.77659	0.00292	0.00558	0.29989	0.000349	0.004685	0.00026	0.00120	0.00018
0.1571	**Dudani**					0.00202	0.00298	0.05564	0.00054	0.02184	0.00269	0.00039	0.00036
0.1585	**Fibonacci**						0.13395	0.25889	0.052102	0.82448	0.03542	0.05326	0.11066
0.1586	**Isquared**							0.33566	0.567827	0.73618	0.36425	0.71153	0.11971
0.1587	**Zavrel**								0.16829	0.20967	0.17674	0.11121	0.02042
0.1593	**Inverse**									0.47733	0.283346	0.18803	1.07E-02
0.1595	**NN**										0.19973	0.40027	0.04322
0.1601	**Rank**											0.81227	0.21201
0.1601	**Macleod**												0.16094
0.1605	***k*NN**												

Table 5.29. p values for the pairwise Wilcoxon sign-rank test of the differences in average error rates across the datasets, in case of Euclidean distance

Tabela 5.29. p vrednosti dobijene uparenim Vilkoksonovim testom nad razlikama prosečnih grešaka na svim skupovima podataka, u slučaju Euklidskog rastojanja

5.5.2. Dynamic Time Warping (DTW)

The situation in the case of DTW is similar as in the case of Euclidean distance. From Table 5.30 we can see, that the Macleod weighting scheme has generated the smallest number of datasets (20) with better results than the 1NN classifier, and the DualU classifier the largest number (29). The second best result was obtained by the DualD method (28 datasets with better accuracies than the simple nearest-neighbor rule). This is reflected also in the number of statistically significant wins and losses (Table 5.31): after 1NN (which has the lowest number of statistically significant wins as in the case of Euclidean distance), *k*NN has the worst ratio of wins and losses (-60). The largest number of wins (108) was produced by the DualD scheme followed by the Dudani method (88 wins). Interestingly, according to the corrected resampled t-test the DualU weighting scheme did not rank among the best ones.

According to the results of the sign-rank test (Table 5.32), the DualD scheme outperforms all other classifiers (except DualU). Similarly as in the case of Euclidean distance, the Dudani method produced low p values against all of the weighting schemes with lower average error rates (except Zavrel and to a lesser extent 1NN). The Wilcoxon sign-rank test does not seems to confirm the wins-losses result of the Inverse scheme from Table 5.31: it exhibited higher p values against all other classifiers (among the ones with lower average error rates). Again, both of the statistical tests affirm that in combination with the DTW similarity measure the DualD and the Dudani weighting schemes are the best choices.

	*k*NN	Inverse	ISquared	Rank	Fibonacci	Dudani	Macleod	DualD	Zavrel	Uniform	DualU
Better than 1NN	21	24	24	22	22	26	20	28	26	24	**29**
Average diff.	0.0106	**0.0132**	0.0123	0.0106	0.0105	0.0126	0.0126	0.0128	0.0092	0.0123	0.0073
Worse than 1NN	22	19	19	21	21	17	**23**	15	17	19	13
Average diff.	0.0081	0.0092	0.0079	0.0101	0.0052	0.0098	0.0096	**0.0102**	0.0078	0.0055	0.0038

Table 5.30. Comparison of *k*NN with 1NN in case of DTW

Tabela 5.30. Upoređivanje *k*NN-a i 1NN-a u slučaju DTW-a

	Wins	Losses	W-L
NN	27	135	-108
***k*NN**	23	83	-60
Rank	29	58	-29
Fibonacci	30	57	-27
Macleod	29	52	-23
ISquared	51	51	0
DualU	54	46	8
Uniform	60	40	20
Zavrel	73	50	23
Inverse	60	31	29
Dudani	88	15	73
DualD	108	14	94

Table 5.31. Statistically significant wins and losses counts for different NN classifiers, across all datasets, in case of DTW

Tabela 5.31. Brojevi statističko značajnih pobeda i poraza različitih NN klasifikatora, na svim skupovima podataka, u slučaju DTW-a

ERROR		0.1359	0.1362	0.1369	0.1369	0.1372	0.1373	0.1378	0.1380	0.1394	0.1397	0.1399	0.1404
ERROR		**DualD**	**Uniform**	**DualU**	**Dudani**	**Isquared**	**Inverse**	**Fibonacci**	**Zavrel**	***k*NN**	**Macleod**	**Rank**	**NN**
0.1359	**DualD**		0.00316	0.13357	0.00042	0.02279	0.00291	0.00133	0.06053	0.000418	0.00001	0.00002	0.03108
0.1362	**Uniform**			5.25E-01	0.05157	0.40636	0.28572	0.37473	0.87716	0.14700	0.01955	0.03737	0.25786
0.1369	**DualU**				0.711526	0.48098	0.463833	0.02698	0.92218	1.22E-02	0.00859	0.00537	0.021118
0.1369	**Dudani**					0.06649	0.01065	0.03450	0.28526	0.01364	0.00005	0.00006	0.12307
0.1372	**Isquared**						0.59155	0.80707	0.51640	0.46378	0.28572	0.47234	0.15261
0.1373	**Inverse**							0.68058	0.91970	0.10434	0.31838	0.83965	0.29850
0.1378	**Fibonacci**								0.31019	0.19182	0.05198	0.04985	0.42395
0.1380	**Zavrel**									0.21874	0.17809	0.13747	0.01579
0.1394	***k*NN**										0.92787	0.86989	0.72472
0.1397	**Macleod**											0.18214	0.81651
0.1399	**Rank**												0.71198
0.1404	**NN**												

Table 5.32. p values for the pairwise Wilcoxon sign-rank test of the differences in average error rates across the datasets, in case of DTW

Tabela 5.32. p vrednosti dobijene uparenim Wilkoksonovim testom nad razlikama prosečnih grešaka na svim skupovima podataka, u slučaju DTW-a

5.5.3. Longest Common Subsequence (LCS)

In comparison with the nearest neighbor rule (Table 5.33), by the number of datasets with smaller classification error, the best ranked classifier is DualU (27). The second best result was produced by the DualD scheme (22) and the third one by the Dudani method (20). The largest number of datasets with larger classification error was found in the case of the Zavrel (28) and the Inverse schemes (27). The largest number of statistically significant wins (Table 5.34) and the smallest number of losses were produced by the DualD weighting scheme. Like in the case of DTW, it is followed by the Dudani method. Third place was taken by the DualU scheme. Again, the smallest number of statistically significant wins was produced by the simple one nearest neighbor classifier and the majority voting k-nearest neighbor rule.

The results of the Wilcoxon sign-rank test (Table 5.35) support the findings of the corrected resampled t-test (Table 5.34): the DualD weighting scheme surpasses all of the other analyzed methods (except DualU). Analogous to the results of the previous two similarity measures, in terms of the obtained p values, the Dudani method outmatched all of the weighting schemes with lower average error rates (except DualU and to a lesser extent Fibonacci and 1NN). As in the case of the Euclidean distance, the DualU method achieved low p values against the methods with lower average error rates. Based on the findings of the statistical tests we come to a similar conclusion as in the case of the Euclidean distance and the DTW similarity measure: the DualD and the Dudani are the best weighting schemes to use in combination with LCS. The DualU is the third best behind these two methods.

	*k*NN	Inverse	ISquared	Rank	Fibonacci	Dudani	Macleod	DualD	Zavrel	Uniform	DualU
Better than 1NN	13	16	16	14	16	20	14	22	14	16	**27**
Average diff.	0.0151	0.0185	0.0180	0.0180	0.0129	0.0169	**0.0188**	0.0159	0.0180	0.0153	0.0086
Worse than 1NN	26	27	25	24	21	17	26	14	**28**	20	7
Average diff.	0.0041	0.0052	0.0047	0.0058	0.0026	**0.0072**	0.0050	0.0065	0.0047	0.0042	0.0038

Table 5.33. Comparison of *k*NN with 1NN in case of LCS

Tabela 5.33. Upoređivanje *k*NN-a i 1NN-a u slučaju LCS-a

	Wins	Losses	W-L
NN	27	141	-114
***k*NN**	15	73	-58
Fibonacci	34	62	-28
Zavrel	25	40	-15
Macleod	31	42	-11
Rank	27	38	-11
ISquared	39	39	0
Uniform	41	37	4
Inverse	55	42	13
DualU	97	53	44
Dudani	91	9	82
DualD	100	6	94

Table 5.34. Statistically significant wins and losses counts for different NN classifiers, across all datasets, in case of LCS

Tabela 5.34. Brojevi statističko značajnih pobeda i poraza različitih NN klasifikatora, na svim skupovima podataka, u slučaju LCS-a

	ERROR	0.1103	0.1112	0.1115	0.1122	0.1124	0.1125	0.1126	0.1130	0.1133	0.1135	0.1140	0.1159
ERROR		**DualD**	**Dudani**	**DualU**	**Isquared**	**Uniform**	**Inverse**	**Fibonacci**	**Macleod**	**Zavrel**	**Rank**	***k*NN**	**NN**
0.1103	**DualD**		0.03735	0.81980	0.00009	0.00610	6.93E-05	0.02251	0.00030	0.00068	0.00080	0.00070	0.04812
0.1112	**Dudani**			0.97686	0.00018	0.02900	0.00185	0.10809	0.00333	0.00210	0.00269	0.00960	0.17364
0.1115	**DualU**				0.06025	0.07024	0.037258	0.01483	0.03783	0.00748	0.03147	0.00258	0.000719
0.1122	**Isquared**					0.35755	0.353404	0.50042	0.64363	0.66024	0.70707	0.21375	0.62242
0.1124	**Uniform**						0.41637	0.98574	0.35719	0.28526	0.30504	0.00551	0.27488
0.1125	**Inverse**							0.35482	0.76735	0.78553	0.640857	4.55E-01	0.88956
0.1126	**Fibonacci**								0.77259	0.63516	0.55212	0.23008	0.29790
0.1130	**Macleod**									0.98966	0.40844	0.19060	0.74192
0.1133	**Zavrel**										0.88008	0.07204	0.90049
0.1135	**Rank**											0.14431	0.70613
0.1140	***k*NN**												0.92773
0.1159	**NN**												

Table 5.35. p values for the pairwise Wilcoxon sign-rank test of the differences in average error rates across the datasets, in case of LCS

Tabela 5.35. p vrednosti dobijene uparenim Wilkoksonovim testom nad razlikama prosečnih grešaka na svim skupovima podataka, u slučaju LCS-a

5.6. Summary

In this chapter we have given an in-depth empirical investigation of the impact of the Sakoe-Chiba band (see Section 2.2) and various weighting schemes (see Section 3.1) on classification accuracy in the field of time-series data mining. Our analysis encompassed 46 datasets from the UCR Time Series Repository [50] and the most commonly used time-series similarity measures (Euclidean distance, DTW and LCS and their extensions ERP and EDR - see Section 2.1). Furthermore, we have provided statistical support to the results obtained. The pseudocodes of the studied algorithms are given in Section 5.1. In the rest of this section we will summarize our findings and give some directions about the use of the algorithms and their parameters.

In Section 5.3 we have, through extensive experiments, shown how can be the accuracy of the 1NN classifier improved by constraining the similarity measures using the Sakoe-Chiba band. The analysis of the changes in the 1NN graph induced by narrowing the warping window has revealed that the constrained similarity measures represent qualitatively and quantitatively different measures than the unconstrained ones.

On average, the smallest error rates were achieved with the values of parameter r (width of the warping window) smaller than 10% of the length of time series. On the other hand, when the value of r drops below 6%, the average classification errors begin to increase and reach their maximums for $r = 0\%$ (see Figure 5.16 in Section 5.3.3). It should also be noted that there are notable differences in the effects of the constraints on different similarity measures: the biggest changes were observed for DTW, the least ones for EDR, while LCS and ERP are in between.

The impact of constraining the warping window on the accuracy of both the unweighted and the weighted kNN classifier has been examined in Section 5.4. Weights were calculated by the formula in Eq. (3.5). With these experiments we have covered the two most representative time-series similarity measures: DTW and LCS.

In case of the unweighted kNN classifier the best results were obtained for $k = 1$. In addition, we have concluded that, as the value of parameter k (number of neighbors) grows, the classification accuracy decreases almost linearly and we need wider and wider warping windows to get the best result. For DTW the average value of parameter r ranges from about 4% to about 10% and for LCS from about 6% to about 13%.

The introduction of weights significantly improved the classification accuracy for all values of k and the lowest error rates were obtained for around $k = 4$. It is important to notice, that in this case the differences between the smallest and largest average error rates are much slighter. Furthermore, the average value of parameter r remains approximately the same for all values of k: for DTW it ranges from about 4% to about 7% and for LCS from about 6% to about 10%. All these findings indicate that the introduction of weights improves the quality and stability of kNN.

Based on the findings in Section 5.4, we have expanded our examinations on several other weighting schemes, too. Beside Euclidean distance these experiments have encompassed the unconstrained DTW and LCS. In Section 5.5 we have seen, that in the terms of average classification accuracy the simple 1NN classifiers has been outperformed by all the other variants of the kNN classifier (except for the Euclidean distance). However, the differences are not particularly big. We have also shown that both the corrected resampled t-test and the Wilcoxon sing-rank test support the DualD (see Section 3.1.6) and the Dudani (see Section 3.1.1) weighting functions as the best choices in combination with all three discussed similarity measures.

Chapter 6

Framework for Analysis and Prediction (FAP)

As a consequence of importance and usefulness of time series, there is a large number of applications which deal with time series, based on different approaches.

The most popular collection of data mining algorithms is implemented within the *WEKA* (*Waikato Environment for Knowledge Analysis*) tool [38]. *WEKA* supports a great number of data-mining and machine-learning techniques, including data pre-processing, classification, regression and visualization. However, *WEKA* is a general-purpose data-mining library, and it is not specialized for time series.

A similar system, *RapidMiner Studio*, is a product of company RapidMiner [94]. It is a collection of data-mining and machine-learning techniques. *RapidMiner* has a very sophisticated graphical user interface, and it is also extensible with the user's implementations. It supports some aspects of statistical time-series analysis, prediction and visualization. The *Starter Edition*, which is available free of charge, is limited to 1 GB of RAM, supports working only with CSV and XLS files and does not support databases.

ELKI (*Environment for DeveLoping KDD-Applications Supported by Index Structures*) [3] is an open source data mining software written in Java at *Ludwig Maximilian University of Munich*, Germany. This framework is free for scientific usage. *ELKI* incorporates several algorithms for clustering, outlier detection, classification, benchmarking and dataset statistics. It is designed to work with high dimensional real-valued feature-vectors - a special case of which are time series. As a support to work with time series, *ELKI* implements the four main similarity measures (DTW, LCS, ERP and EDR).

Also, there are several tools specialized for summarization and visualization of time series: *TimeSearcher* [42], *Calendar-Based Visualisation* [125], *Spiral* [123] and *VizTree* [66], but they are not specialized for real-world time-series analysis.

The above systems, which partially support time-series analysis, are mainly based on data mining methods. On the other hand, there is a large number of systems which are based on statistical and econometric modelling. Probably the most famous business system is *SAS* [106]. Among many business solutions including: Business Analytics, Business Intelligence, Data Integration, Fraud Prevention & Detection, Risk Management etc., SAS has an integrated subsystem for time series. This subsystem provides modelling of trend, cycles and seasonality of time series as well as time series-forecasting, financial and econometric analysis. However, *SAS* is a complex commercial system which is not freely available.

MATLAB [72] represents another well-known commercial software package for numerical computing which supports data analysis and visualization, programming and algorithm implementation and application development and deployment. Several *MATLAB* toolboxes support different aspects of time-series analysis including: Econometrics Toolbox, Financial Toolbox, Neural Network Toolbox, Signal Processing Toolbox and Wavelet Toolbox Toolbox.

R [90] is a programming language and a free software environment for statistical computing and graphics. It can be used on a wide vriety of platforms, including *UNIX*, *Windows* and *MacOS*. *R* supports statistical modeling, classical statistical tests, classification, clustering, regression, time series analysis and others. R supports several concepts of time-series analysis: linear filtering, decomposition, regression analysis, exponential smoothing, ARIMA models and others.

GRETL (*GNU Regression, Econometrics and Time-series Library*) is an open source, platform-independent library for econometric analysis [8]. It supports several least-square based statistical estimators, time-series models and several maximum-likelihood methods. *GRETL* also encloses a graphical user interface for the *X-12-ARIMA* environment. *X-12-ARIMA* is the Census Bureau's new seasonal adjustment program [27]. It supports several interesting concepts such as: alternative seasonal, trading-day and holiday effect adjustment; an alternative seasonal-trend-irregular decomposition; extensive time-series modeling and model-selection capabilities for linear regression models with ARIMA errors.

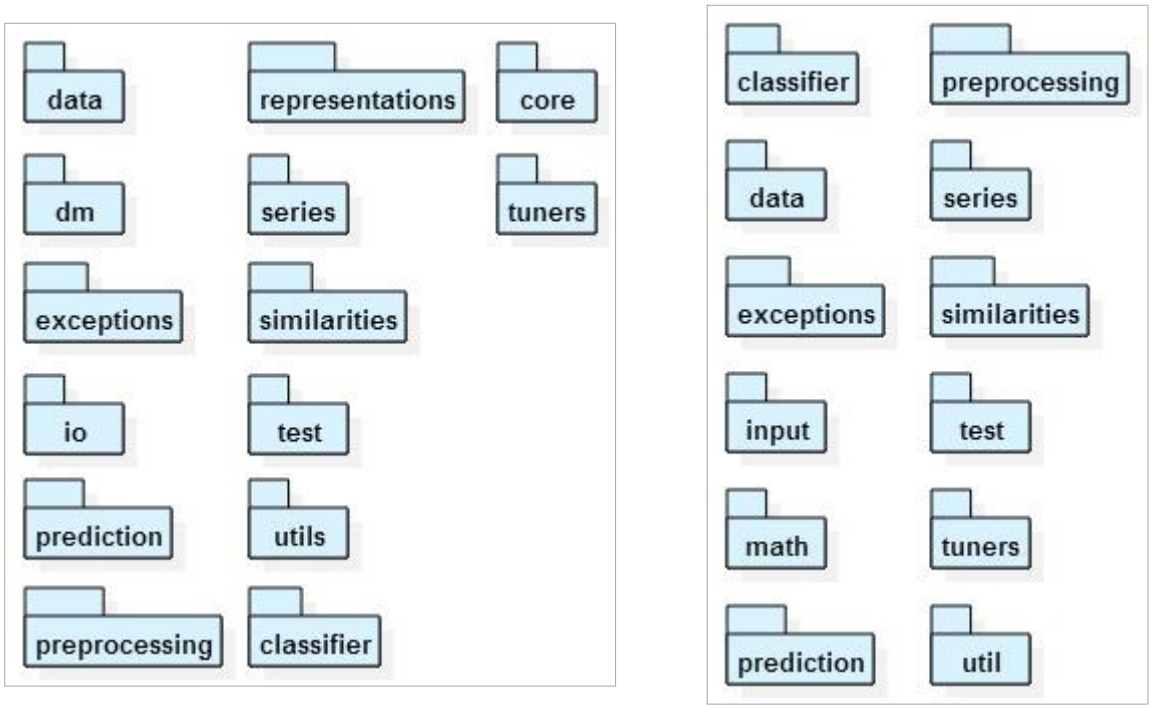
Clearly, two kinds of software applications can be distinguished: general purpose data-mining software packages which in some extent support time series, and software applications specialized for time series based on statistical and econometric models. So, it is evident that there is a huge gap between these two types of applications. There is no available system specialized for time-series mining (time-series analysis based on data-mining techniques) with the exception of the library presented in [21] which can be obtained on demand. This was our main motive for designing our system. *Framework for Analysis and Prediction* (FAP) will contain all main features and functionalities needed for time-series analysis (pre-processing tasks, similarity measures, time-series representations) and necessary for different data-mining tasks (indexing, classification, prediction, etc). We believe that such a system will significantly help researchers in comparing their own newly introduced and proposed concepts with the existing ones.

6.1. FAP Overall Structure

In the current state of development, all main similarity measures are implemented, as well as several parameter-tuning algorithms, classifiers, classifier evaluation methods, representations and pre-processing techniques. FAP is designed to incorporate all main aspects of time-series mining in one place and to combine easily some or all of them. Upon completion, the system will contain all important realizations of proposed concepts up to that point, as well as the possibility to add newly introduced ones with ease.

The overall structure of the framework is presented in Figure 6.1 (a). The essential part of the library is implemented in package fap.core (Figure 6.1 (b)). Its sub-packages contain the implementation of the basic structures, definitions of mutual interfaces and abstract classes that describe the fundamental functionality of the system and the common properties of objects. All the core classes support object serialization, have a default constructor and public getter and setter methods - thereby *JavaBeans* technology is also supported by FAP. Almost every package from the fap.core package has its outside counterpart, where the concrete implementations of desired concepts are stored.

In the rest of this section we will give a brief overview of the sub-packages which constitute the fap.core package. Details necessary to understand the functioning of the entire library are outlined in the following sections. A practical application of the FAP library is presented and explained in Section 6.8.



(a) FAP library (b) fap.core package

Figure 6.1. Architecture of the FAP library

Slika 6.1. Arhitektura biblioteke FAP

The auxiliary packages are fap.core.exceptions, fap.core.input, fap.core.math and fap.core.util.

The first contains the interface and basic classes which describe the exceptions thrown by FAP.

The fap.core.input package includes interfaces and classes which describe what classes responsible for loading data points and series of them should satisfy.

The third supporting package contains implementations of additional mathematical concepts needed in FAP (for example, the implementation of polynomials, needed for the Spline representation).

The interfaces in the fap.core.util package define common methods which must be implemented by classes that support resumable long-running operations and maintain monitoring of these operations through a callback mechanism.

Package fap.core.data contains basic implementations of data points and series of them. The raw representation of time series (as series of data points) and basic implementation of time-series datasets are given in package fap.core.series. This package also contains an interface which describes what properties other representations should satisfy.

One time series may have several representations. The interface that declares common methods for similarity computors is provided in the fap.core.similarities sub-package.

Some similarity measures have one or more parameters which need to be tuned based on a training dataset. The interface for tuning classes is stored in fap.core.tuners package.

Similarly, packages fap.core.prediction, fap.core.classifier and fap.core.preprocessing contain the interfaces which describe prediction techniques, classification algorithms and pre-processing tasks, respectively. The interface which needs to be implemented by the classes that are intended for evaluating the performance of classifiers is located in fap.core.test.

6.2. Basic Components

The basic components of FAP include time series consisting of data points, datasets of time series and structures which enable continuation of interrupted long-running operations and maintain tracking using a callback mechanism. In this section we will give an overview of the corresponding interfaces and classes.

6.2.1. Data Points and Time Series

A time series is defined as a sequence of ordered pairs of values (see Chapter 2). The first element of these pairs represents the time component and the second one the measured value of the observed phenomenon. This section provides an overview of the implementation of these core structures within the FAP library.

Time series objects (instances of the TimeSeries class - the corresponding UML diagram is shown in Figure 6.3) are implemented as series of data point objects (instances of the DataPoint class - Figure 6.2). Each data point has two coordinates of type double:

- X (the time component) and
- Y (the measured value).

Data points are serializable and comparable with regard to the time component.

Basic methods of data-point series are implemented through the abstract class DataPointSeries, including finding the

- minimum (getMinY),
- maximum (getMaxY) and
- the mean (getMeanY) value of the data points' Y coordinates (Figure 6.2).

Analogous methods exist for the X coordinate, but they are omitted from the diagram for clarity reasons. The fap.data package contains two particular serializable implementations: the first is based on array list (DataPointSeriesArrayList) and the second one is based on linked list (DataPointSeriesLinkedList).

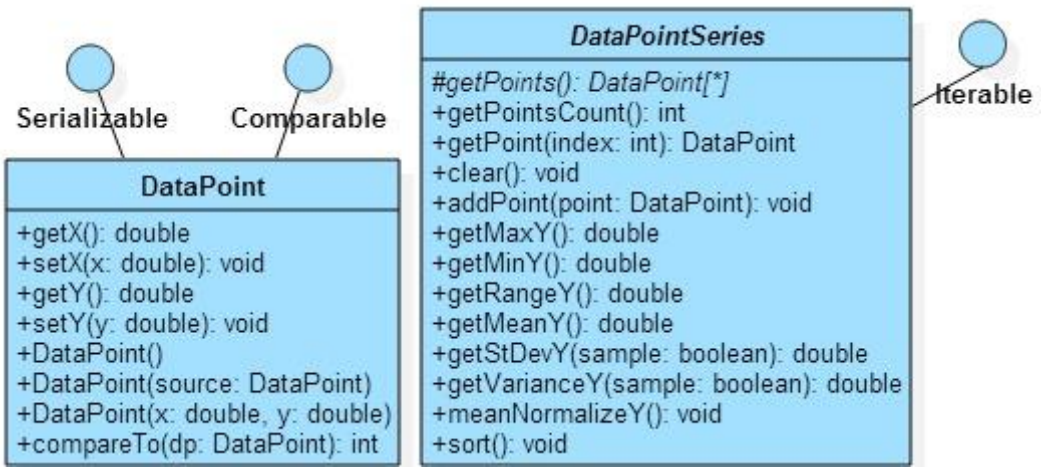


Figure 6.2. Data points and series of data points

Slika 6.2. Tačke podataka i niz tačaka podataka

Every TimeSeries object (Figure 6.3) contains:

- a series of data points,
- a double-typed field label (which represents the class of the time series),
- an int-typed supplementary property called index and
- may have several representations.

Most of the methods defined in class DataPointSeries are directly accessible through the TimeSeries class (they are not shown on the UML diagram for the sake of readability). TimeSeries objects can be serialized, too.

Time-series datasets are realized as objects of type TimeSeriesArrayList (Figure 6.3) which extends the generic ArrayList class. There are several auxiliary methods defined in this class, including:

- getLabels - retrieves the list of labels of the dataset's time series;
- getDistribution - retrieves the distribution of labels: for every label, the number of time series with that label;
- getSeriesByClasses - retrieves a list of datasets where each dataset contains time series with the same label;
- groupList - retrieves a list (dataset) of the time series grouped by their labels;
- getRandomSplit - randomly divides the dataset into k subsets of roughly equal size;
- getPercentageSplit - randomly divides the dataset into two subsets. Parameter percentage determines the percentage of time series that are placed in the first dataset;
- getStratifiedSplit - randomly divides the dataset into k stratified subsets of roughly equal size;
- getStratifiedPercentageSplit - randomly divides the dataset into two stratified subsets.

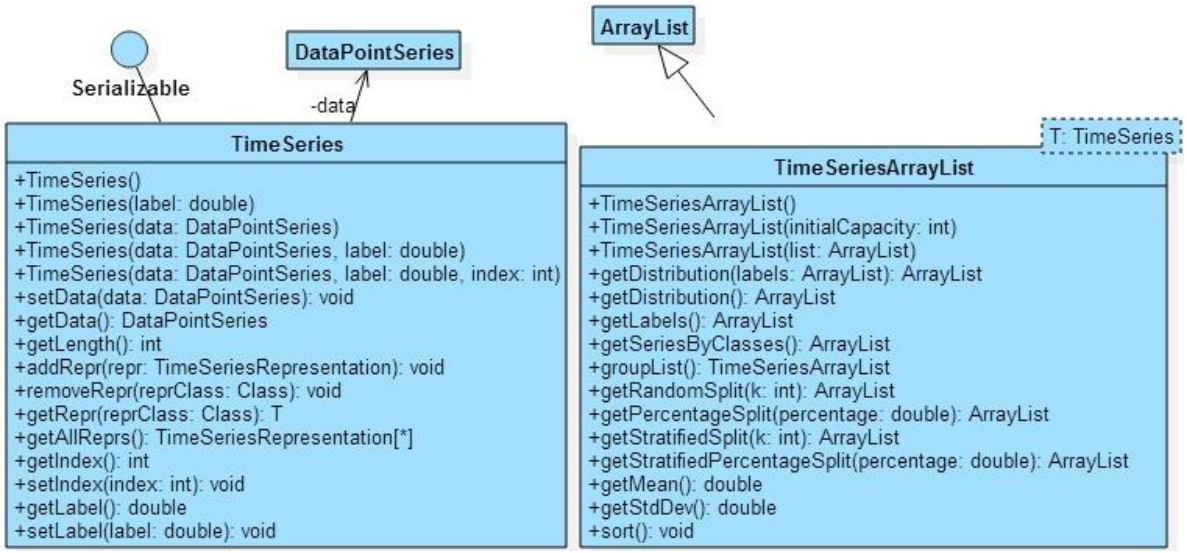


Figure 6.3. Time series and datasets

Slika 6.3. Vremenske serije i skupovi podataka

6.2.2. Resuming and Tracking

Preparing and performing the experiments whose results are described and analysed in Chapter 5 are both very time consuming processes. Department of Mathematics and Informatics, Faculty of Sciences, University of Novi Sad is equipped with several computer centers which can be used to perform these operations. However, they are often used for practical exercises within several computer science courses and also for carrying out experiments of other researches. Taking into account these restrictions, our FAP library is specifically designed to enable monitoring of long-time calculations through a callback mechanism, along with the possibility of their interruption without the loss of already obtained results. The incomplete computations can be resumed later. This section is devoted to the description of the components that enable these capabilities.

The callback mechanism and the Resumable interface (Figure 6.4) facilitate tracking the execution of long-running processes (for example, classification, evaluation of classifiers, tuning the parameters of similarity measures, etc.). Together with object serialization they represent the key elements in enabling the storage of partial results of time-consuming operations and the continuation of interrupted tasks.

Classes that perform long-running operations and should support interrupting and resuming their execution, need to implement the Resumable interface. The reset method should reset the state of the object and prepare it for reuse. In addition, these objects should indicate whether they have finished performing their task (isDone) and whether is the execution still in progress (isInProgress).

Methods that are necessary for the implementation of the callback mechanism are defined by the Callback interface. Classes that provide tracking of their activities should implement the CallbackEnabled interface (Figure 6.4) and should regularly call the callback method of the appropriate Callback object in accordance with the configuration set through the getCallbackCount and setCallbackCount methods. The first of these two methods indicates how many times it is expected that they call the callback method. However, they do not have to comply with this expectation. Instead, using the setCallbackCount method they can themselves determine the number of callbacks based on their own needs and capabilities.

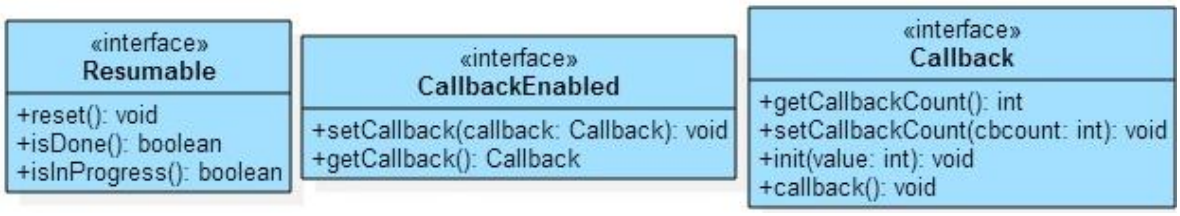


Figure 6.4. Interfaces for resuming and tracking long-running operations

Slika 6.4. Interfejsi za nastavljanje i praćenje dugotrajnih procesa

6.3. Similarity Measures and Tuners

Similarity measures represent essential ingredients of many time-series data mining tasks. Their role is to describe the similarity (or dissimilarity) between time series using numerical values (see Section 2.1). Many of them are dependent on various parameters. For achieving the best results, the values of these parameters have to be set prior to the use of the similarity measures. This is accomplished by relying on appropriate parameter tuners.

In this section we will show what needs to be satisfied by the classes that represent similarity measures and tuners in order to be in accordance with the FAP system. Furthermore, relying on the example of DTW we will explain how are the elastic similarity measures implemented within the FAP framework.

Classes that represent similarity measures need to implement the SimilarityComputor interface. This interface defines only one method that returns the similarity (distance) between the two time series submitted via its parameters (Figure 6.5). The AbstractSimilarityComputor abstract class defines only a default serial version ID value and provides a simple string representation of similarity measures through the toString method.

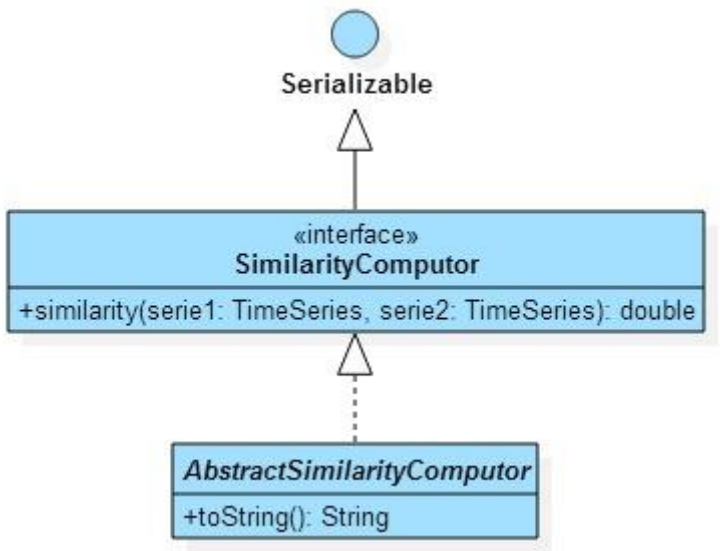


Figure 6.5. The SimilarityComputor interface and the AbstractSimilarityComputor class

Slika 6.5. SimilarityComputor interfejs i klasa AbstractSimilarityComputor

The fap.similarities package contains implementation of the following similarity measures: Minkowski distance (L_p norm), Euclidean distance (L_2), Manhattan distance (L_1), Chebyshev distance (L_∞), Canberra distance, and the Swale and Spline similarity measures.

FAP contains several elastic similarity measures, too: DTW, LCS, ERP, EDR and TWED. Their unconstrained versions are in the fap.similarities.unconstrained sub-package. Their constrained versions are implemented in two ways: applying the Sakoe-Chiba band and the

Itakura parallelogram - the corresponding classes are contained in the fap.similarities.constrained sub-package.

We will demonstrate the implementation of the elastic measures on the example of DTW. Suppose that we need to calculate the similarity between two time series of length n and m: $Q = (q_q, q_2, \dots, q_n)$ and $S = (s_1, s_2, \dots, s_m)$. We can get the required result based on the recursive definition of DTW (defined by Eq. (2.3) in Section 2.1.2) by relying on an auxiliary matrix D of format $(n+1) \times (m+1)$ as shown in Figure 6.6. The similarity between Q and S will appear in cell (n, m), i.e.: $DTW(Q, S) = D(n, m)$.

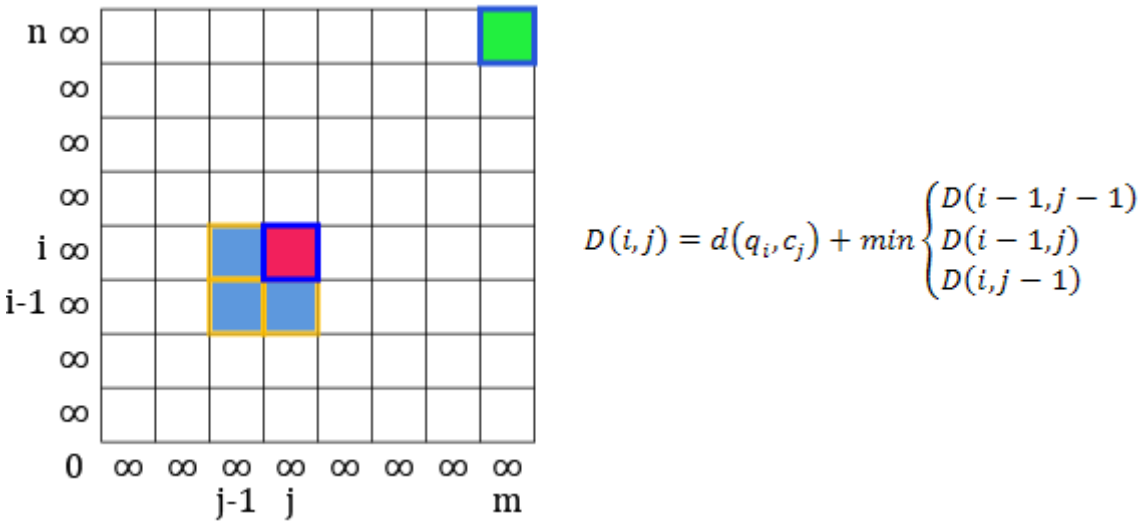


Figure 6.6. Computing DTW using dynamic programming

Slika 6.6. Računanje DTW-a pomoću dinamičkog programiranja

Memory and time requirements of the above algorithm can be significantly improved if we realize that it is not necessary to keep the whole matrix D in memory and to perform computations on the whole matrix. In each step, in fact, we use only two rows of D. To compute the value of cell (i, j), we need only the values of the neighboring cells $(i, j-1)$ and $(i-1, j)$. In this way, the matrix of format $(n+1) \times (m+1)$ can be reduced[7] to a matrix of format $2 \times (slen + 1)$ where $slen = min(n, m)$ as shown in Figure 6.7.

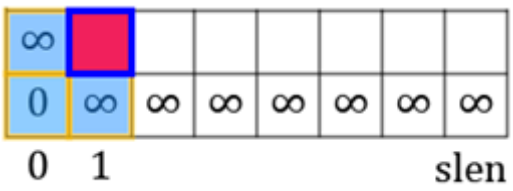


Figure 6.7. Improved implementation of dynamic programming

Slika 6.7. Poboljšana implementacija dinamičkog programiranja

[7] The assumption is that the similarity measure is symmetrical.

Classes intended for tuning the parameters of similarity measures need to implement the SimilarityTuner interface (or to inherit the AbstractSimilarityTuner class). It declares three methods: one for tuning the parameters (tune) using the given dataset, and two others for setting and getting the similarity measure object whose parameters need to be adjusted (Figure 6.8). The inheritance of the Serializable, Resumable and CallbackEnabled interfaces enables developing serializable and resumable tuners which activities can be traced. The AbstractSimilarityTuner is a convenience class that implements basic functionalities. Package fap.tuners.Keogh contains several tuners that are implemented based on the algorithms described in [21].

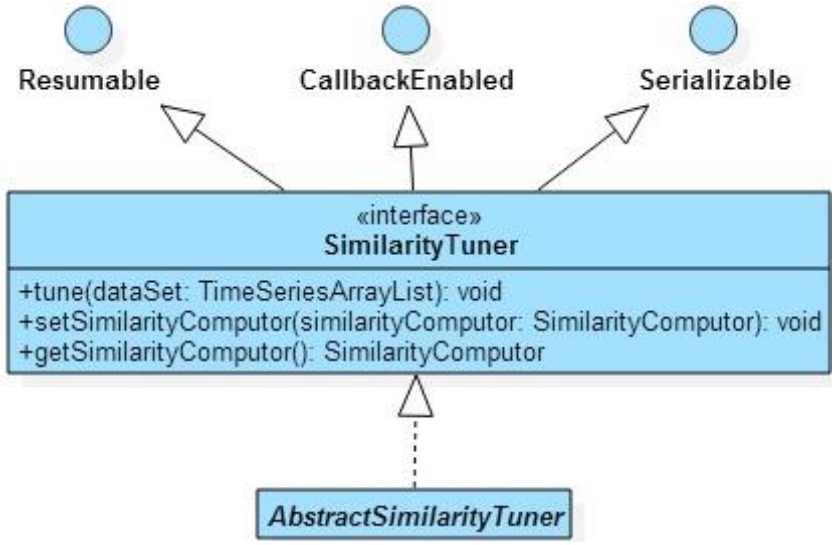


Figure 6.8. SimilarityTuner interface and AbstractSimilarityTuner class

Slika 6.8. SimilarityTuner interfejs i klasa AbstractSimilarityTuner

6.4. Classifiers

Classification denotes the process of grouping time series into predefined classes (see Chapter 3). This section will briefly introduce the interface of the FAP system that declares common methods for classifier classes.

The methods required for implementing classifiers are declared within the Classifier interface (Figure 6.9).

The build method conducts training the classifier based on the given dataset and similarity measure. Training can be a rather long-lasting process whose execution we may want to monitor and/or to pause at some moment. This can be ensured by an appropriate implementation of the methods declared within the Resumable and CallbackEnabled interfaces.

The classify method is responsible for classifying the given time series. It should return the label selected by the classifying algorithm.

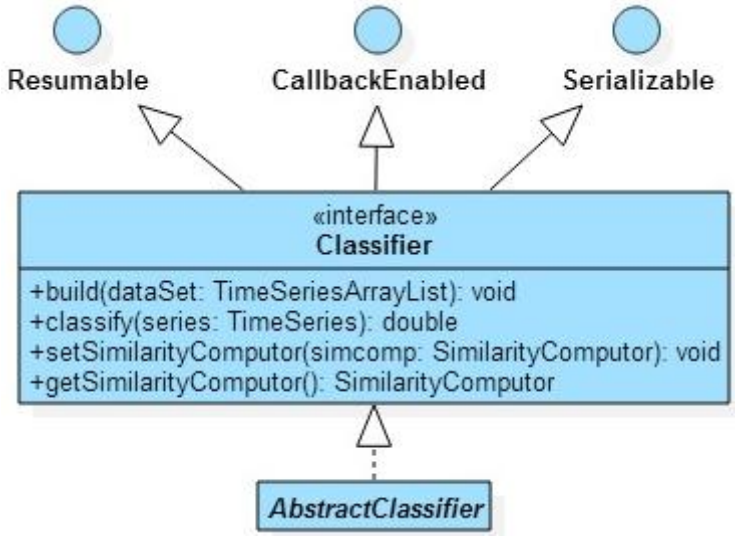


Figure 6.9. Classifier interface and AbstractClassifier class

Slika 6.9. Classifier interfejs i klasa AbstractClassifier

The AbstractClassifier is a convenience class that implements basic functionalities. The fap.classifier.NN package contains several classifiers: the simple nearest-neighbor classifier (1NN), the majority voting k-nearest neighbor classifier (kNN) and the distance-weighted k-nearest neighbor classifier in combination with every weighting scheme described in Section 3.1.

6.5. Classifier Evaluation

The aim of this section is to present the interface of the FAP system which specifies the common behavior of the classes intended for evaluating the performances of classifiers. Additionally, we will briefly explain the main properties of the implementations of the three most popular evaluation techniques described in Section 3.2.

The basic methods for evaluating the performance of classifiers are declared within the Test interface (Figure 6.10). The test method starts (or continues) the evaluation process using the given dataset and the classifier set by the setClassifier method.

The getErrorRatio method should return the average error ratio. The number of misclassified time series should be returned by the getMisclassified method.

Time-consuming evaluation techniques can be supported by implementing the inherited Resumable, CallbackEnabled and Serializable interfaces. The AbstractTest is a convenience class that implements basic functionalities.

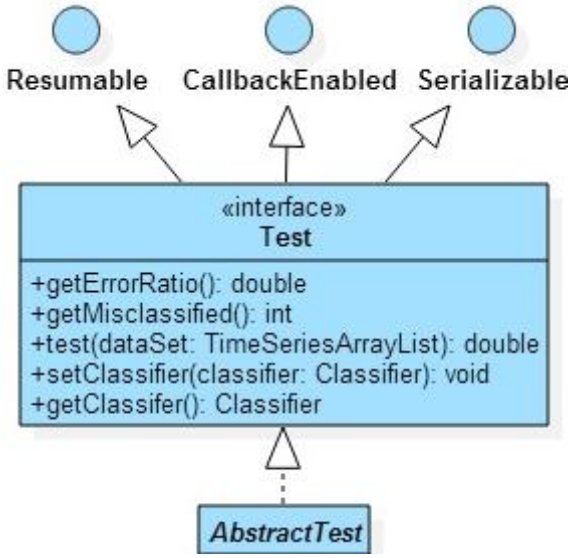


Figure 6.10. Test interface and AbstractTest class

Slika 6.10. Test interfejs i klasa AbstractTest

The fap.test package contains implementation of all three main evaluation algorithms presented in Section 3.2. In the rest of this section we will briefly overview the main characteristics of their implementations.

Holdout. The way of functioning of the Holdout class is determined through three parameters. The percentage of the dataset used for training is controlled by the percentage parameter (default value is 0.5). The rnd parameter is used to randomize the division of the initial dataset into training and test set (default value is null). The stratified property indicates whether we need stratified or random splitting (default value is true).

CrossValidation. The operating mode of the cross-validation algorithm implemented by the CrossValidation class is determined by the values of the following properties:

- foldsNumber - the number of folds. Default value is 10.
- stratified - indicates whether we want stratified or random splitting. Default value is true.
- rnd - random number generator used for splitting the initial dataset. Default value is null.
- special - indicates whether we want to use a variant of cross-validation defined by Ding et al. in [21]. Default value is false.

LeaveOneOut. This class does not define additional parameters.

6.6. Preprocessing

Often it is necessary to prepare and clean the available data from datasets prior to their use in different data mining tasks. This is achieved by applying different techniques of data preprocessing. The major steps in data preprocessing include: data cleaning, handling missing data, data integration, data reduction, data transformation and data discretization. A detailed overview of these techniques can be found in [39], [62], [114] and [116].

The PreprocessingTransformation interface (its UML diagram is shown in Figure 6.11) defines only one overloaded method. The task of the transform method is to transform the given time series (or an entire dataset).

The AbstractPreprocessingTransformation is a convenience class that implements basic functionalities.

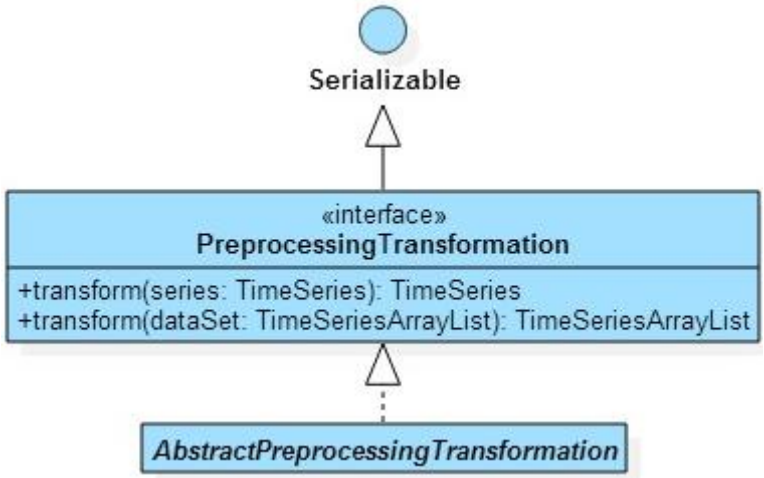


Figure 6.11. PreprocessingTransformation and AbstractPreprocessingTransformation

Slika 6.11. PreprocessingTransformation i AbstractPreprocessingTransformation

Suppose we have a time series $Q = \big((X_1, Y_1), (X_2, Y_2), \dots, (X_n, Y_n)\big)$. Package fap.preprocessing contains implementation of several preprocessing algorithms:

- **Scaling** the X and/or the Y coordinates of a time series by multiplying them with the given scaling factors scaleX and scaleY, e.g. $X_i = X_i \times scaleX$ and $Y_i = Y_i \times scaleY$, $\forall i \in \{1, \dots, n\}$.

- **Shifting** the X and/or the Y coordinates of a time series by adding to them the given shifting factors shiftX and shiftY, e.g. $X_i = X_i + shiftX$ and $Y_i = Y_i + shiftY$, $\forall i \in \{1, \dots, n\}$.

- **Z-score normalization**. Let μ denotes the mean value and δ the standard deviation of the Y coordinates of a given time series. This algorithm normalizes the time series by using the following formula [39, 114]:

$$Y_i = \frac{Y_i - \mu}{\delta}, \forall i \in \{1, \dots, n\}$$

- **Min-max normalization.** This algorithm maps the value of the Y coordinates into the interval $[newMin, newMax]$ using the following formula [39, 62, 114]:

$$Y_i = \frac{Y_i - minY}{maxY - minY} \times (newMax - newMin) + newMin, \forall i \in \{1, \dots, n\}$$

- **Decimal scaling**. Let k denotes the smallest integer such that $max_{i=1,\dots,n}\{|Y_i|\} < 10^k$. This algorithm transforms Q using the following formula [39, 114]:

$$Y_i = \frac{Y_i}{10^k}$$

- **Linear equiscaling**. Scales an equidistant time series to the desired length using linear interpolation.

6.7. Representations

Storing time series usually requires large amounts of space which makes performing different tasks of data mining more difficult. Furthermore, often we are not interested in the exact values of each time-series data point [95]. For these reasons, time-series databases generally contains only simplified representations of the series [25]. A formal definition of time-series representations is given by Esling and Agon in [25] as follows:

> Given a time series $T = (t_1, \dots, t_n)$ of length n, a representation of T is a model $\bar{T}$ of reduced dimnsionality $\bar{d}$ ($\bar{d} \ll n$) such that $\bar{T}$ closely approximates T.

This section describes how the representations of time series are realized within the FAP framework and provides an overview of the implemented representations.

Classes that constitute representations of time series need to implement the TimeSeriesRepresentation interface which UML diagram is depicted in Figure 6.12. The getValue method should retrieve the value of the corresponding time series at the given value of the time component. The getOutboundValue method should return the value of time series outside the range which is covered by current representation.

The following time-series representations are implemented inside the fap.representations package:

- **Piecewise Linear Approximation (PLA).** In this representation, a time series Q of length n is approximated with K straight lines (K is typically much smaller than n) [51]. The segments (the straight lines of the approximation) can be calculated either

using *linear interpolation* or *linear regression*. Their length can be determined using one of the following three approaches:

  - **Sliding Windows** - A segment is grown until the error of the approximation exceeds some predetermined threshold.
  - **Top-Down** - The time series is recursively partitioned until some stopping criteria is not satisfied.
  - **Bottom-Up** - Starting from the finest possible approximation, the segments are merged until some stopping criteria is not satisfied.

- **Piecewise Aggregate Approximation (PAA).** A time series Q of length n is approximated by dividing it into $N < n$ segments of equal length. The mean values of the Y coordinates of Q within these segments constitute the PAA representation of Q. In this way, we reduce the data from n dimensions to N dimensions [47].

- **Adaptive Piecewise Constant Approximation (APCA).** APCA is an extension of the PAA representation that allows arbitrary length segments [48].

- **Symbolic Aggregate Approximation (SAX).** The idea behind this representation is to reduce a time series Q of length n into a string of length N, where N is typically much smaller than n. The elements of the resulting string are taken from a predefined alphabet that contains k symbols. First, the initial time series is normalized and then it is approximated by the PAA representation using N segments. After that, the normal (Gaussian) distribution is divided into k intervals; so that each part has the same probability. Next, a symbol of the alphabet is assigned to each of the intervals. In the last step, the segments of the PAA representation are mapped into the symbols of the alphabet relying on these intervals [67].

- **Spline**. In this approach, time series are approximated by cubic splines through the set of their points [57].

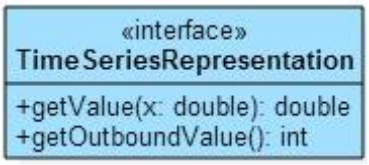


Figure 6.12. TimeSeriesRepresentation interface

Slika 6.12. TimeSeriesRepresentation interfejs

6.8. Using FAP

In this section we will demonstrate the usage of our FAP framework on the algorithm that we used to find the values of the parameter ε for the LCS (see Section 2.1.3) and EDR (see Section 2.1.5) similarity measures. In the process of comparing the elements of two time series, these two similarity measures rely on the value of this matching threshold.

Let $StDev$ denote the standard deviation of a particular dataset. The value of parameter ε was determined by calculating the classification error of the 1NN classifier using the LOO evaluation method. We have selected the value in range from $0.002 \times StDev$ to $StDev$ (with steps of $0.002 \times StDev$) which gave the best classification accuracy.

For smaller datasets this search was carried out on the entire dataset. In case of larger sets (which contain more than 1400 time series) it was performed on a stratified subset of the size not less than 400 elements. The obtained values of ε are presented in Table 4.1 in Section 4.2.

The procedure of calculating the value of parameter ε is presented in Algorithm 3.

As input, we need to specify the name of the dataset and a boolean value that indicates whether we want to compute with LCS or with EDR.

After the dataset is loaded (line 1), we check its number of time series: if it is greater than 1400 (line 2), we take a stratified subset of the dataset (line 3). For the sake of readability, this procedure is omitted from the algorithm.

In the next step, we instantiate the key objects:

- the similarity computor (line 4),
- the classifier (line 5) and
- the leave-one-out (LOO) evaluation method (line 6).

In the following five lines we initialize the auxiliary variables (lines 7-11).

The first step in each iteration of the while loop is to update the value of parameter ε of the similarity measure object (line 12).

Before performing the LOO evaluation we need to reset the inner state of the test object by calling its reset method (line 13). This is necessary because the implementations of the classifier evaluation methods support interrupting and resuming of their execution (see Section 6.2.2).

After the LOO evaluation process (line 14) is completed, we obtain the average error ratio (line 15) and if necessary, the core auxiliary variables are updated (lines 16-20). If the error is equal to zero further tests are not needed and we can get out of the loop (lines 19-20).

Increasing the value of parameter ε (line 19) closes the while loop.

The value of parameter ε that gave the best classification accuracy (smallest error ratio) is returned in the last line of the algorithm (line 22).

Algorithm 3: Calculating the value of the matching threshold for LCS and EDR

Input: the name of the dataset and boolean value which indicates whether we want to work with LCS or with EDR

Output: the value of parameter ε

```
    private static double getEpsilon(String dsname, boolean lcs) throws Exception {

1.          TimeSeriesArrayList<TimeSeries> dataset = DatasetUtils.loadDataSet(dsname);
2.          if (dataset.size()>1400)
3.                  dataset = getStratifiedSample(dataset,dsname);

4.          AbstractEpsilonSimilarityComputor similarityComputor =
                    lcs ? new LCSSimilarityComputor() : new EDRSimilarityComputor();

5.          Classifier classifier = new NNClassifier(similarityComputor);

6.          Test test = new LeaveOneOut(classifier);

7.          double stdev = dataset.getStdDev();
8.          double step = 0.02 * stdev;

9.          double minError = Double.POSITIVE_INFINITY;
10.         double bestEpsilon = 0;

11.         double epsilon = step;
            while (epsilon<=stdev) {

12.                 similarityComputor.setEpsilon(epsilon);

13.                 test.reset();
14.                 test.test(dataset);

15.                 double error = test.getErrorRatio();
16.                 if (error<minError) {
17.                         minError = error;
18.                         bestEpsilon = epsilon;
19.                         if (minError==0)
20.                                 break;

                    }
```

```
21.                     epsilon += step;
                }

22.             return bestEpsilon;
        }
```

Chapter 7

Conclusion

The subject of this dissertation encompasses a detailed overview and empirical analysis of the impact of Sakoe-Chiba global constraint [105] on the most commonly used elastic similarity measures in the field of time-series data mining with a focus on classification accuracy. The choice of similarity measure is one of the most significant aspects of time-series analysis - it should correctly reflect the resemblance between the data presented in the form of time series. Similarity measures represent a critical component of many tasks of mining time series, including: classification, clustering, prediction, anomaly detection, and others.

The research covered by this dissertation is oriented on several issues:

1. review of the effects of global constraints on the performance of computing similarity measures (Section 5.2),

2. a detailed analysis of the influence of constraining the elastic similarity measures on the accuracy of classical classification techniques (Sections 5.3 and 5.4),

3. an extensive study of the impact of different weighting schemes on the classification of time series (Section 5.5),

4. development of an open source library that integrates the main techniques and methods required for analysis and mining time series, and which is used for the realization of these experiments (Chapter 6).

A suitable choice of similarity measure between time series is an important part of similarity-based retrieval, and is in some form included in many tasks of time-series analysis. Since Euclidean distance [26] is a very simple measure which is calculated quickly and represents a distance metric, it has become one of the most commonly used measures of similarity between time series [4, 14, 47, 48]. However, it has two major disadvantages: it can only work with time series of the same length and is sensitive to distortions and shifting along the time axis [49, 98]. To overcome these weaknesses many elastic measures are proposed in the literature (DTW [10], LCS [121], ERP [16], EDR [17] etc.). These measures have better classification accuracy than Euclidean distance [21], but they are all based on dynamic programming, which means that their computation complexity is quadratic. To decrease the computation time of the elastic measures global constraints are introduced, narrowing the search path in the matrix.

In Section 5.2, we examined the influence of the Sakoe-Chiba global constraint on the performance of the two most representative elastic similarity measures for time series: DTW and LCS. Through an extensive set of experiments, we showed that the usage of global constraints can significantly reduce the computation time of these measures, which is their main weakness. Based on obtained results, we can conclude that the difference of computation times between an unconstrained measure and a measure with a small constraint is two and somewhere three orders of magnitude. In future work, it would be interesting to examine the impact of the Sakoe-Chiba band on the performance of other elastic similarity measures, too. Other possible directions for future work include expanding these examinations on the Itakura parallelogram.

It was suggested that, in the case of DTW, the use of global constraints can actually improve the accuracy of classification compared to unconstrained similarity measures [98, 130]. In Section 5.3, through an extensive set of experiments we have described in detail the impact of the Sakoe-Chiba band on the nearest-neighbor graph. We showed that the constrained measures are qualitatively different than the unconstrained ones. From the obtained results we can clearly see that for low values of the constraint (less than 15%–10%) the change of the 1NN graph becomes significant for all of the considered similarity measures. In addition to this, the results reveal that there are notable differences in the effects of the constraints on different distance measures. DTW was found to be the most sensitive to the introduction of global constraints regarding the 1NN graph, while EDR is the least sensitive. The behavior of ERP and LCS measures was determined to be somewhere in between.

Comparison of 1NN classifier performance showed that DTW generally has a slight edge over other distance measures (especially ERP), but is more sensitive to the choice of the r parameter. Statistical tests are somewhat confirmed that DTW can be considered as the commonly best distance measure, but the evidence is not particularly strong. For every distance measure there are at least a couple of datasets where the measure is significantly superior to all others. Therefore, the choice of the best distance measure for a particular problem may be different for the generally best case. Considering the average classification errors across different values of r, we can conclude that they grow conspicuously for small values ($r < 6\%$) and in all four cases reaches their maximums for $r = 0\%$. The largest increase occurs for LCS and the lowest one for DTW. In the interval from $r = 100\%$ to $r = 6\%$ the similarity measures behave differently: in case of DTW the average classification error almost monotonously decreases, while for ERP it almost monotonously increases. In case of LCS and EDR a slight tendency of growth can be spotted, but the changes are very subtle. This suggests that although DTW can be considered the best general choice, LCS and EDR could be safer choices because of the less pronounced need for tuning the r parameter.

The findings of our studies have clearly shown that all of the main elastic similarity measures (DTW, LCS, ERP and EDR) significantly change their behavior for small values of the global constraint. Thus, we expect our results to aid researchers and practitioners in selecting and tuning appropriate time-series similarity measures for their respective tasks, making the selection/tuning process simpler and faster, at the same time producing more accurate results. In addition, the insight into the behavior of similarity measures with respect to changing constraints can be beneficial to the design of efficient indexing strategies for fast computation of (approximate) nearest neighbors. In future work, we plan to expand the

investigation of the effects of these changes on the accuracy of a wider range of distance-based classifiers. It would also be interesting to explore the influence of the Itakura parallelogram compared to the influence of the Sakoe-Chiba band.

The results of experiments in Section 5.4 evidently confirmed the special importance of the first neighbor in time-series data. In the case of both studied elastic similarity measures (DTW and LCS), the error rate of the unweighted kNN classifier almost linearly grows as the number of neighbors k grows (Section 5.4.2). The kNN classifier actually gives the best results for the value $k = 1$ when consider k neighbors without a weighting scheme. On the other hand, when the weighting scheme defined by Eq. (3.5) is introduced (Section 5.4.3), the situation is changed to some extent. The best results are obtained for the value around $k = 4$. Overall, we can conclude that, the weighting scheme (which favors the first neighbor) significantly improved the classification accuracy for all values ok k.

When observing the value of constraint parameter r, the introduction of the weighting scheme has an important impact. For unweighted kNN, the value of the constraint grows as k grows. On the other hand, with the weighting scheme the value of the constraint remains approximately the same for all values of k. In addition, the difference between minimum and maximum values of constraints is about two times smaller with the weighting scheme.

All these observations indicate that favoring the first neighbor with a weighting scheme improves the quality and stability of kNN. The first neighbor has a special meaning in time-series data and taking this fact into consideration can significantly improve the quality of kNN for all values of k, by making it even more accurate than 1NN for some small values of k. In future work, it would be interesting to investigate the influence of other weighting schemes [22, 35, 36, 69] and to widen these studies to other similarity measures.

In the last decade classification has been intensively investigated in the field of time-series data mining [34, 45, 93, 130, 131]. Among the considerable number of proposed techniques, the simple nearest neighbor classifier (1NN) and the Dynamic Time Warping distance measure were shown to be one of the best combinations [130]. To improve the accuracy of classification, the majority voting k-nearest neighbor rule generalizes the idea of the 1NN rule by taking into account not one but k nearest neighbors. The next step in investigating the nearest neighbor rule is assigning different weights to the neighbors.

Several different weighting schemes are proposed in the literature [22, 35, 36, 62, 69, 74, 84, 133]. They have been examined exclusively in combination with Euclidean distance and they were either not tested in the domain of time series, or tested only in a very limited extent. Furthermore, the results were not supported by statistical tests. In Section 5.5, we have, through a detailed analysis, compared a wide variety of nearest-neighbor weighting schemes in combination with the three most commonly used time-series similarity measures based on the largest set of freely available labeled time-series datasets [50].

Observing the average classification accuracy, in the case of all of the considered similarity measures, the best results are obtained with the dual distance-weighting scheme defined by Gou at al. [35] (Eq. (3.11)). The worst average classification accuracy is produced by the simple nearest neighbor rule (except for Eucledean distance). It is worth noting that the

differences between the best and worst average results are not particularly big (< 0.01). With these detailed experiments we have confirmed the view that the simple 1NN is very hard to beat [130]. When observing the number of statistically significant wins and losses, the best results are achieved by the distance-weighted scheme defined by Dudani [22] (Eq. (3.2)) in case of Euclidean distance, and the dual distance-weighting scheme defined by Gou at al. [35] (Eq. (3.11)) in case of DTW and LCS. Both the corrected resampled t-test and the Wilcoxon sign-rank test results support the Dudani and the DualD weighting methods as the best choices in combination with the three discussed time-series similarity measures. The DualU (Eq. (3.9)) is the third best choice behind these two methods.

Since the elastic measures (DTW, LCS, ERP and EDR) generally provide better classification accuracies compared to non-elastic measures, it would be interesting to check the influence of different weighting schemes also on ERP and EDR (besides DTW and LCS) and on the constrained versions of all these measures. The major drawback of these measures is performance, since they are all based on quadratic complexity algorithms. In Chapter 5 we showed that the introduction of global constraints significantly speeds-up the computation process and in some cases even improves the classification accuracies. Furthermore, due to the high dimensionality of time-series data, it would be interesting to investigate the interaction of the hubness phenomenon [93] with different kNN weighting methods and the behavior of the hubs-based weighting scheme [92].

Time-series analysis and mining has been a very popular research area in the past decade. This resulted in a huge amount of proposed techniques and algorithms. A great majority of techniques and algorithms were sporadically introduced and sometimes not correctly compared with their counterparts. This is the consequence of a lack of a quality open-source system which supports different aspects of time-series mining. For all these reasons, we created a universal framework (Framework for Analysis and Prediction (FAP)) where all main concepts, like similarity measures, representation and pre-processing, will be incorporated. Such a framework would greatly help researchers in testing and comparing newly introduced concepts with the existing ones.

In the current state of development, all main similarity measures are implemented, as well as several classifiers, classifier evaluation methods, representations and pre-processing techniques. The implemented similarity measures include L_p, Swale, unconstrained and constrained DTW, LCS, ERP and EDR. The constrained measures are implemented using the Sakoe-Chiba band and the Itakura parallelogram. The system contains the implementation of the 1NN and kNN classifiers (including a great number of different weighting schemes), the Holdout, Cross-Validation and Leave-One-Out testing methods, the following time-series representations: PLA [51], PAA [47], APCA [48], SAX [67] and Spline [57] and several pre-processing transformations including scaling, shifting, min-max normalization, z-score normalization, decimal scaling and linear equiscaling. The details of its structure and implementation are described in Chapter 6. All of the examinations within this dissertation were performed by relying solely on this framework. Furthermore, it has been already successfully employed within various research domains including: developing a distributed distance matrix generator based on agents [75, 76], mining time series in the psychological domain [55, 56] and time-series analysis in the neurology domain [61].

We believe that the FAP system could be intensively used in future researches due to its numerous advantages. First, all important up to date concepts needed for time-series mining are integrated in one place. Second, modifications of existing concepts, as well as additions of newly proposed concepts could be obtained very easily (FAP is written in Java). Finally, FAP is free and open source, and everyone will be invited to contribute with newly proposed concepts. This will ensure that the system is always up to date and that all major techniques in time-series mining are supported.

The contributions of the results presented within this dissertation are manifold:

1. We have explicated the impact of the Sakoe-Chiba band on the performance of the constrained elastic similarity measures: for small constraint values, the difference of computation times between the unconstrained and constrained measures is two and somewhere three orders of magnitude.

2. Analyzing the 1-nearest neighbor graph with respect to the change of the constraint size, we have shown that for low values of the constraint (less than 15%–10%) the constrained measures become substantially different from the unconstrained ones. Furthermore, we have demonstrated that these changes are not the same for different similarity measures - DTW was found to be the most sensitive to the introduction of global constraints, while EDR is the least sensitive.

3. Through exhaustive experiments we have shown that, on average, the best classification accuracy is achieved for small values of parameter r. This value is lowest for DTW (about 4% of the length of time series) and highest for ERP (almost 10%). Changes in the 1-nearest neighbor graph generated by these values of r are the highest in case of DTW (on average, about 10% of the nodes), while the lowest in case of EDR (on average, about 1% of the nodes). Comparison of 1NN classifier performance showed that DTW generally has a slight edge over other distance measures, but the evidences are not particularly strong - the best choice depends on the particular problem.

4. Comparing the average classification errors of the four considered elastic similarity measures (DTW, LCS, ERP and EDR) we have pointed out their mutual property: the average classification error grows for small values of r ($< 6\%$) and it reaches its maximum at $r = 0\%$ in all four cases. We have seen that, although DTW can be considered the best general choice, LCS and EDR could be safer choices because of the less pronounced need for tuning the r parameter.

5. The detailed analysis of the kNN classifier (for DTW and LCS) has showed that, without a weighting scheme it gives the best results for $k = 1$. On the other hand, in case of the weighted kNN classifier, the best results are obtained round $k = 4$.

6. Observing the average classification accuracy we have found that the majority voting system and all of the examined weighting schemes produced better classification accuracies than the 1NN rule for a significant number of the

datasets. Both the corrected resampled t-test and the Wilcoxon sing-rank test results support the Dudani and the DualD weighting methods as the best choices.

7. To support these and future researches, we have developed a free and open-sourced library (FAP) that implements many of the most important algorithms in the field of time-series data mining and analysis.

As part of future research it would be interesting to extend all these studies also on other commonly used similarity measures, including ERP, EDR and TWED. Furthermore, it would be worthwhile to compare the impact of the Itakura parallelogram with the influence of the Sakoe-Chiba band.

Appendix A

ID	DTW unconstrained	75%	50%	25%	20%	15%	10%	5%	1%	0%
1	980156	897266	723485	429750	360391	281656	201703	112766	38672	20422
2	317047	292344	236593	144719	121578	96906	70125	41937	18172	13157
3	13672	12610	10031	5937	4968	3828	2688	1469	422	188
4	79844	73391	58656	34500	28562	22047	15141	8016	2016	672
5	258375	242703	198969	124719	105375	86031	62672	41766	23047	17203
6	8923375	8235500	6713578	4102078	3459047	2707140	1980485	1243328	557265	442625
7	88609875	79638047	63711094	36991468	30533875	23531718	16107266	8290062	1814203	146672
8	4766	4375	3547	2156	1843	1438	1031	641	266	140
12	203375	183782	147219	88172	73688	57172	40343	22625	7234	3172
13	7015	6500	5391	3562	3109	2578	1985	1422	953	875
14	256343	235609	191359	120281	102047	82000	60141	38203	21203	17032
15	1569844	1443265	1170437	728641	630266	502500	382782	245391	138532	108906
16	26594	23969	19328	11390	9562	7438	5265	2875	953	422
17	434297	392656	317093	185672	154672	119906	83672	45390	12031	3969
18	15922	14297	11859	7219	6234	4922	3719	2328	1234	906
19	4257391	3789922	3052828	1774359	1468031	1135609	781547	404907	88234	10844
20	25407250	22014203	17571907	10359203	8563921	6618313	4534843	2341546	491640	32109
21	32328	31719	29344	26891	24921	24578	23469	22062	20734	19797
22	101641	90781	72750	42828	35171	27391	19890	9875	2594	719
23	36156	32391	25968	15485	12765	10016	7141	3843	1360	625
24	97847485	88454641	70498062	41180453	34403500	26572297	18149719	9492531	2189141	284188
25	238094	216266	178250	115343	98062	81782	62672	43766	26516	24172
26	220938	202250	168765	114234	96188	82532	66093	51125	32015	28875
29	19828	17906	14391	8500	6969	5469	3844	2078	578	203
30	592328	536562	431062	254844	210047	164515	113000	61312	17672	5906
31	16344	14859	12172	7593	6375	5156	3828	2515	1281	953
32	39391	36516	30750	21172	18891	16187	13593	10343	7484	6953
33	87375	80937	68266	48547	43094	36687	30609	24828	18297	16672
34	1433139834	1293488860	1035132077	605505932	497113666	389422687	265040114	137944867	32319219	4230860
35	367797	338375	276969	175109	147719	121063	87922	60141	32735	24672
36	2708297	2462547	1984625	1174797	975469	754594	521344	278235	74953	29609
37	28734	27266	23047	16657	14938	12984	11141	9172	6891	6562
38	51828	47156	37625	22500	18906	14671	10406	5766	1969	1156
39	175250	164063	136407	91015	79610	67031	54750	41562	26765	24500
40	7317188	6744641	5514797	3489312	2929907	2405765	1736594	1169078	666938	580750
44	20128188	18483156	15043656	9417906	7821062	6203250	4657500	2884859	1442281	1208547
45	989203	910219	729062	435344	363875	281407	203390	112734	37422	19547
46	32217266	29147485	23421094	13682812	11531969	8860547	6179563	3373750	969672	357422

Table A.1. Calculation times of distance matrices for DTW

Tabela A.1. Vremena izracunavanja matrica rastojanja za DTW

ID	LCS unconstrained	75%	50%	25%	20%	15%	10%	5%	1%	0%
1	679921	646828	519922	312703	261859	204813	149641	85422	32203	21359
2	214640	203141	164703	102359	85969	68922	51141	32094	16047	13531
3	8875	8625	6859	4016	3360	2593	1859	1031	344	172
4	52282	48844	39187	23078	18984	14875	10281	5516	1531	609
5	170016	161250	134297	85671	72359	60656	45797	32406	20204	17547
6	5996172	5688891	4676984	2917032	2464000	1974609	1504000	978219	502875	449078
7	62996109	59056579	46699297	27158750	22369187	17262734	11882062	6155704	1402765	161766
8	3578	3438	2859	1672	1406	1109	828	484	235	109
12	136484	127375	102657	61296	51203	39781	28313	16110	5625	3062
13	4984	4671	3969	2719	2328	2000	1671	1234	860	828
14	184375	171718	139906	89625	76375	62219	47187	31281	18922	18219
15	1116625	1034640	845797	534875	463344	377438	294235	196766	119422	117875
16	17890	16766	13469	8015	6703	5204	3813	2125	781	469
17	285922	268282	214875	127484	105765	82329	58000	31906	9390	4156
18	10703	9969	8188	5188	4531	3719	2907	1984	1094	938
19	2793547	2593672	2074313	1218812	1008203	777266	539672	280437	64437	12359
20	16524687	15507859	12291281	7163172	5895296	4556593	3139500	1633141	351640	37016
21	27640	27438	26265	23718	24406	22953	21547	21532	20484	20016
22	71000	66578	53329	31531	26422	20484	14188	7547	2250	750
23	24219	22453	18125	10890	9015	7079	5031	2843	1093	641
24	67029406	62582938	50095359	29302062	24642968	18588750	12998219	6910859	1670969	307656
25	163812	154640	128781	85063	75265	63750	50578	36812	25422	25187
26	156000	147610	123953	86046	78313	66109	54578	43016	30641	30344
29	13313	12375	10016	5796	4781	3735	2656	1485	438	203
30	388375	371156	295813	174734	144938	114547	79140	43750	13984	6375
31	10907	10266	8484	5375	4484	3750	2922	2000	1203	1015
32	28796	27110	23140	16328	15047	13172	11469	9078	7172	7094
33	64734	60828	52547	37922	35125	30219	26265	22140	17703	17343
34	968111646	900357668	719464374	419590179	344832973	269150209	183543819	96485667	22870071	6240336
35	251297	236156	193234	124703	106187	89875	67359	47828	29672	25750
36	1779625	1707250	1362750	810141	667969	521125	366922	200188	61031	32125
37	21657	20875	17953	13250	12313	11000	9578	8188	6672	6578
38	35000	32406	26156	15672	13219	10422	7485	4297	1703	1125
39	124359	117235	99203	68094	61422	53375	44625	35610	25859	25235
40	5046594	4803813	3970578	2544031	2182422	1833125	1383812	1000500	662015	549859
44	14056203	13290391	10899766	6874594	5817140	4688906	3612797	2394250	1346766	1151922
45	679250	645062	518079	310281	260718	204047	148078	84640	31766	21172
46	21409297	20134156	16205828	9558015	7984328	6157015	4330890	2414438	787375	352359

Table A.2. Calculation times of distance matrices for LCS

Tabela A.2. Vremena izracunavanja matrica rastojanja za LCS

Appendix B

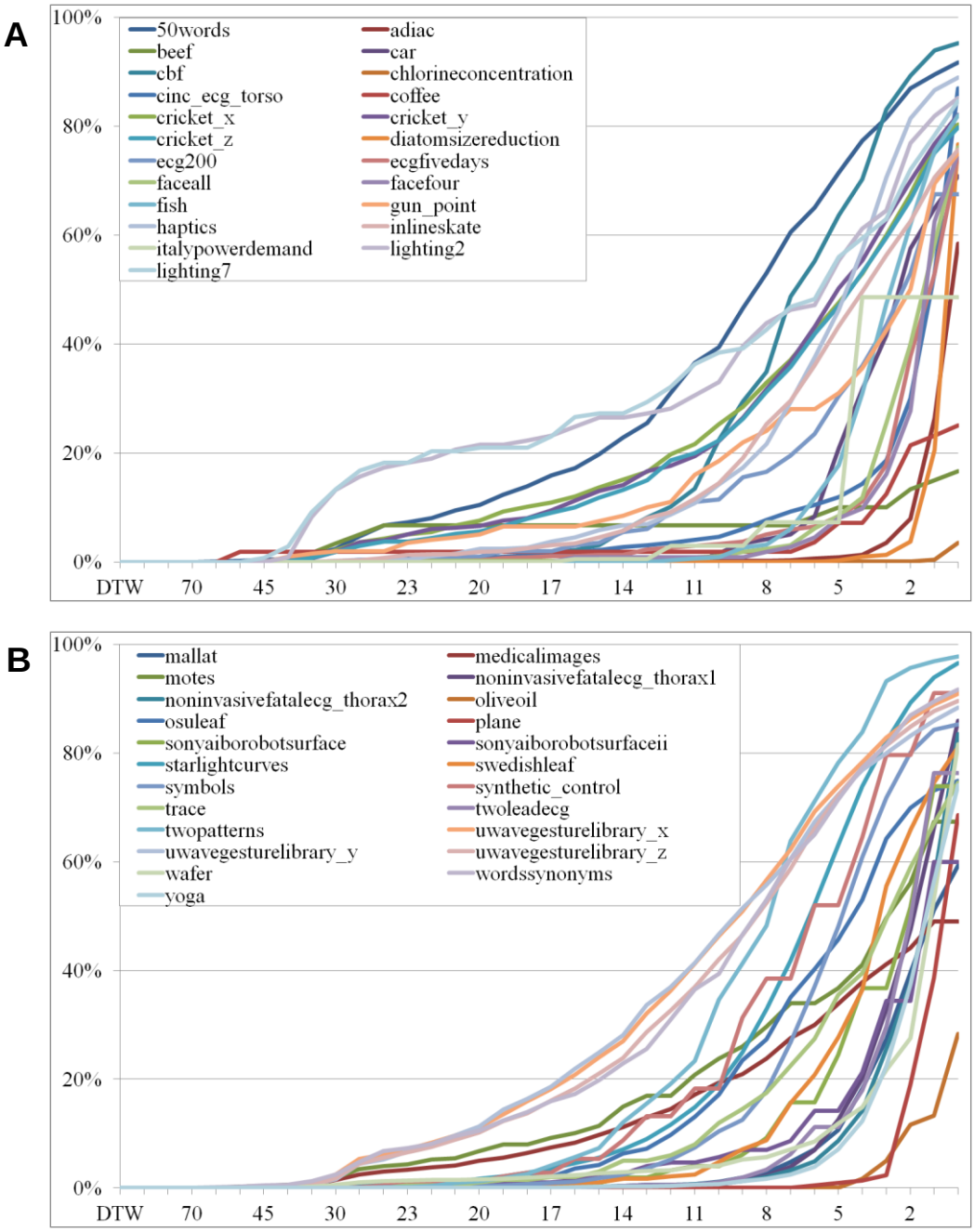


Figure B.1. Detailed plots of the change of 1NN graph for DTW

Slika B.1. Detaljni grafikoni promena 1NN grafa za DTW

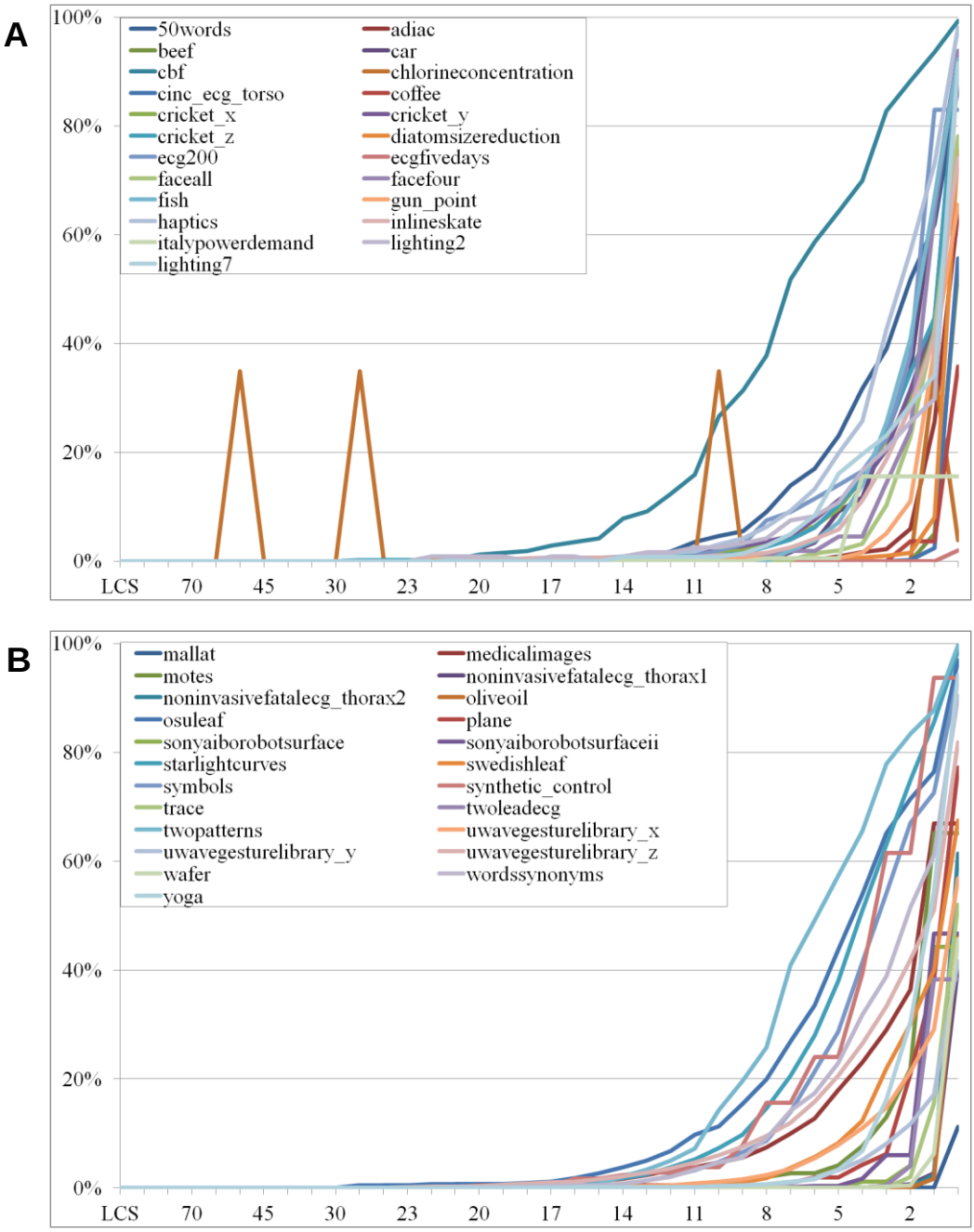


Figure B.2. Detailed plots of the change of 1NN graph for LCS

Slika B.2. Detaljni grafikoni promena 1NN grafa za LCS

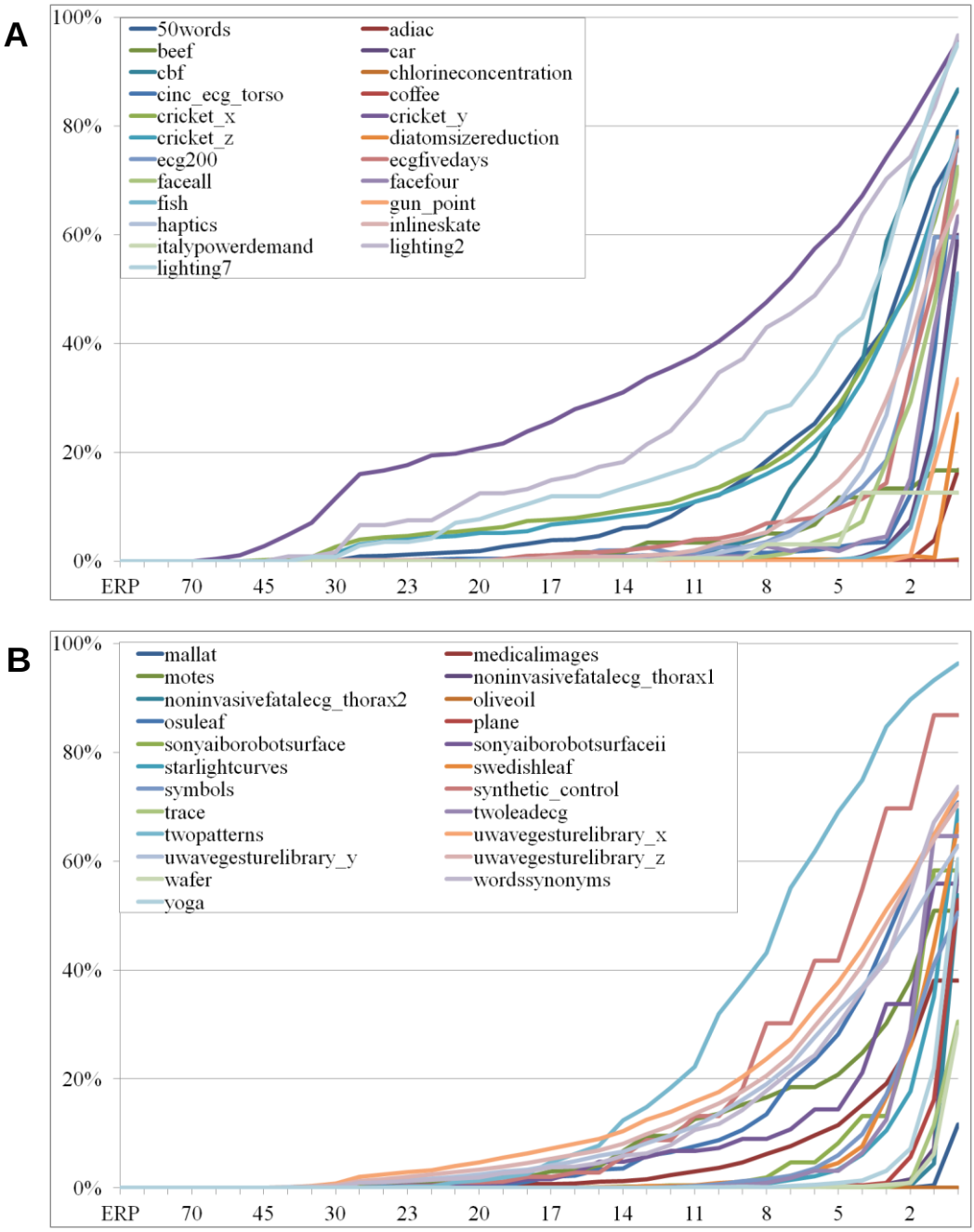


Figure B.3. Detailed plots of the change of 1NN graph for ERP

Slika B.3. Detaljni grafikoni promena 1NN grafa za ERP

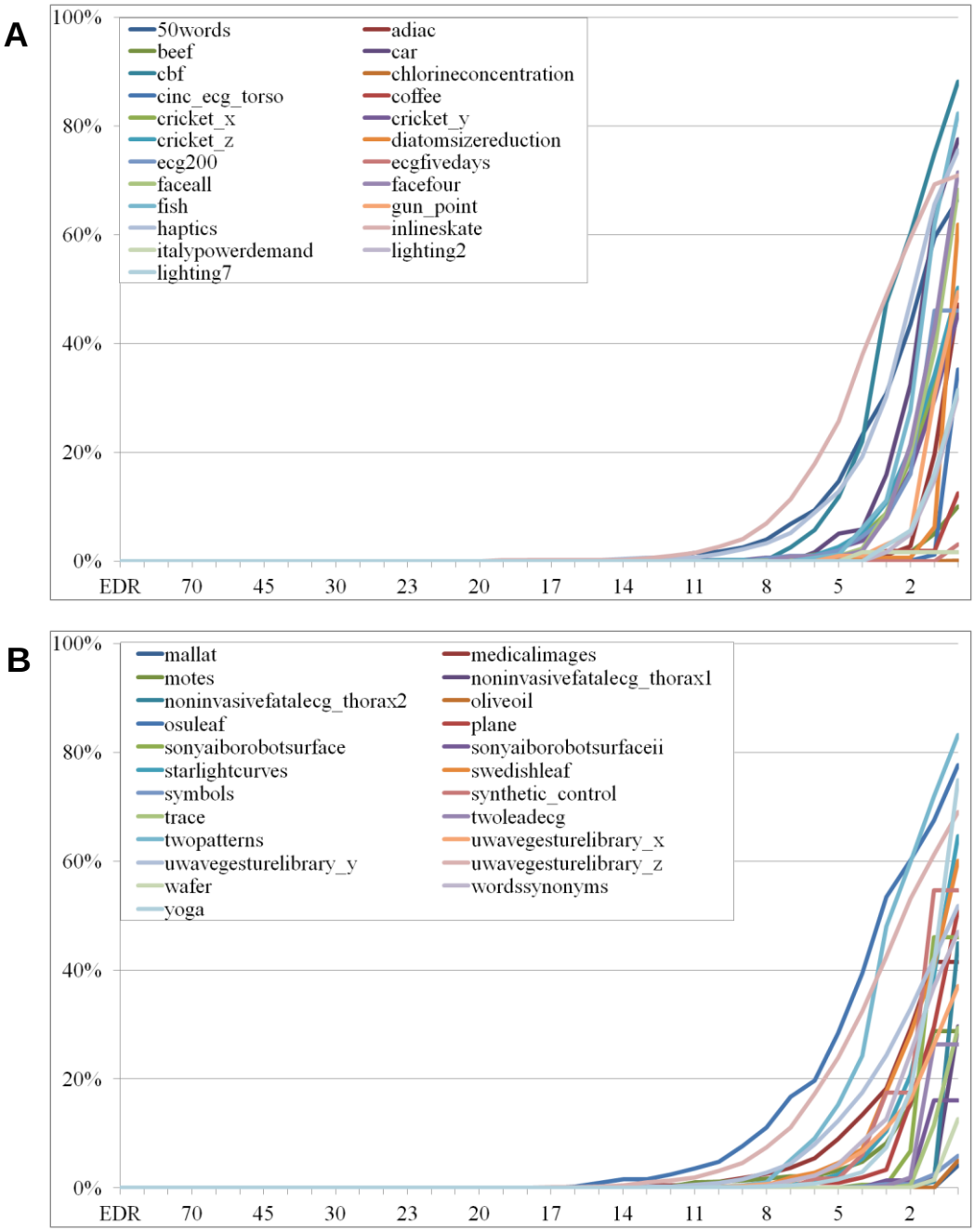


Figure B.4. Detailed plots of the change of 1NN graph for EDR

Slika B.4. Detaljni grafikoni promena 1NN grafa za EDR

Appendix C

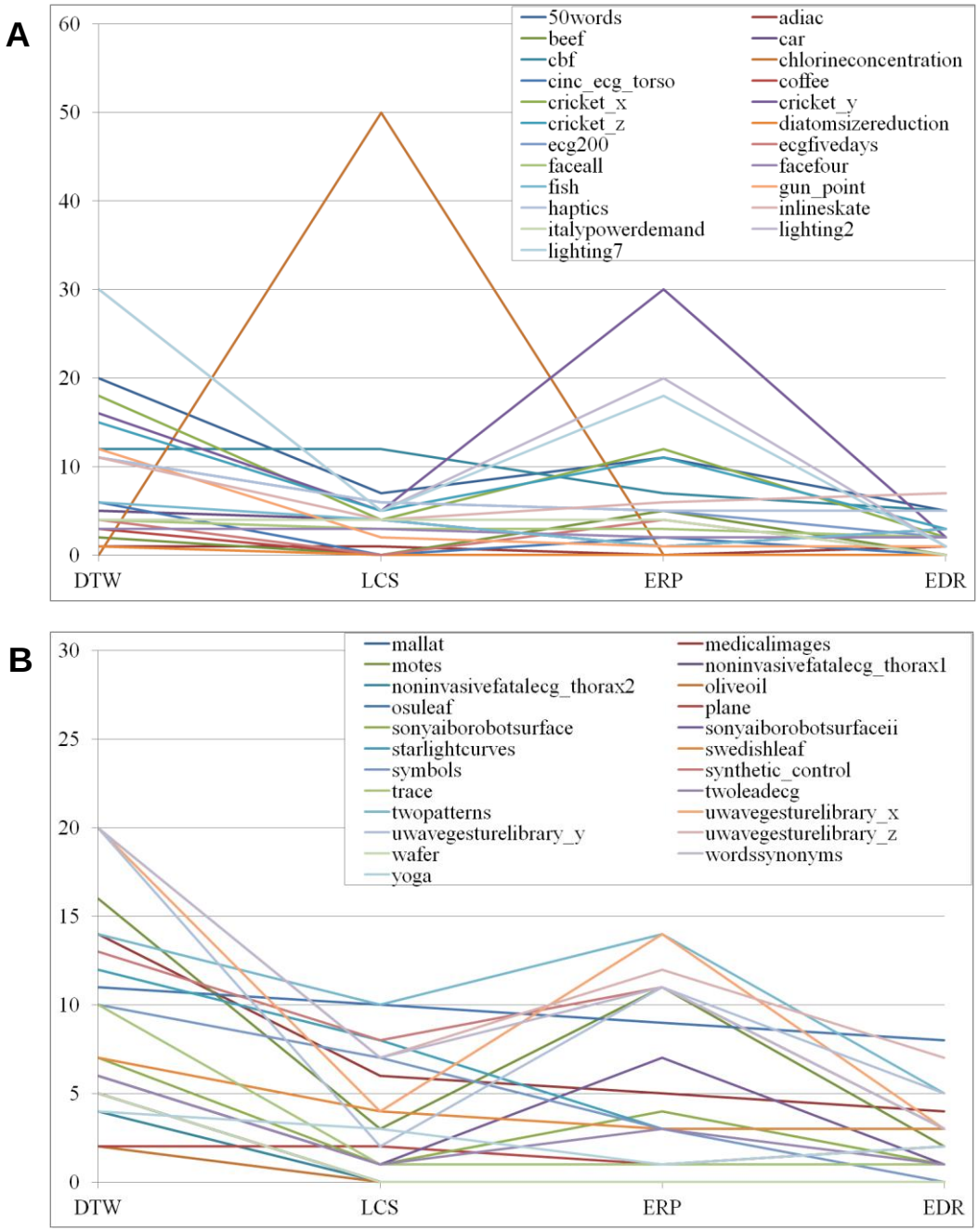


Figure C.1. Detailed plots of the highest warping windows needed to change at least 10% of the 1NN graph

Slika C.1. Detaljni grafikoni najvećih pojasa iskrivljenja potrebni da se promeni najmanje 10% 1NN grafa

Appendix D

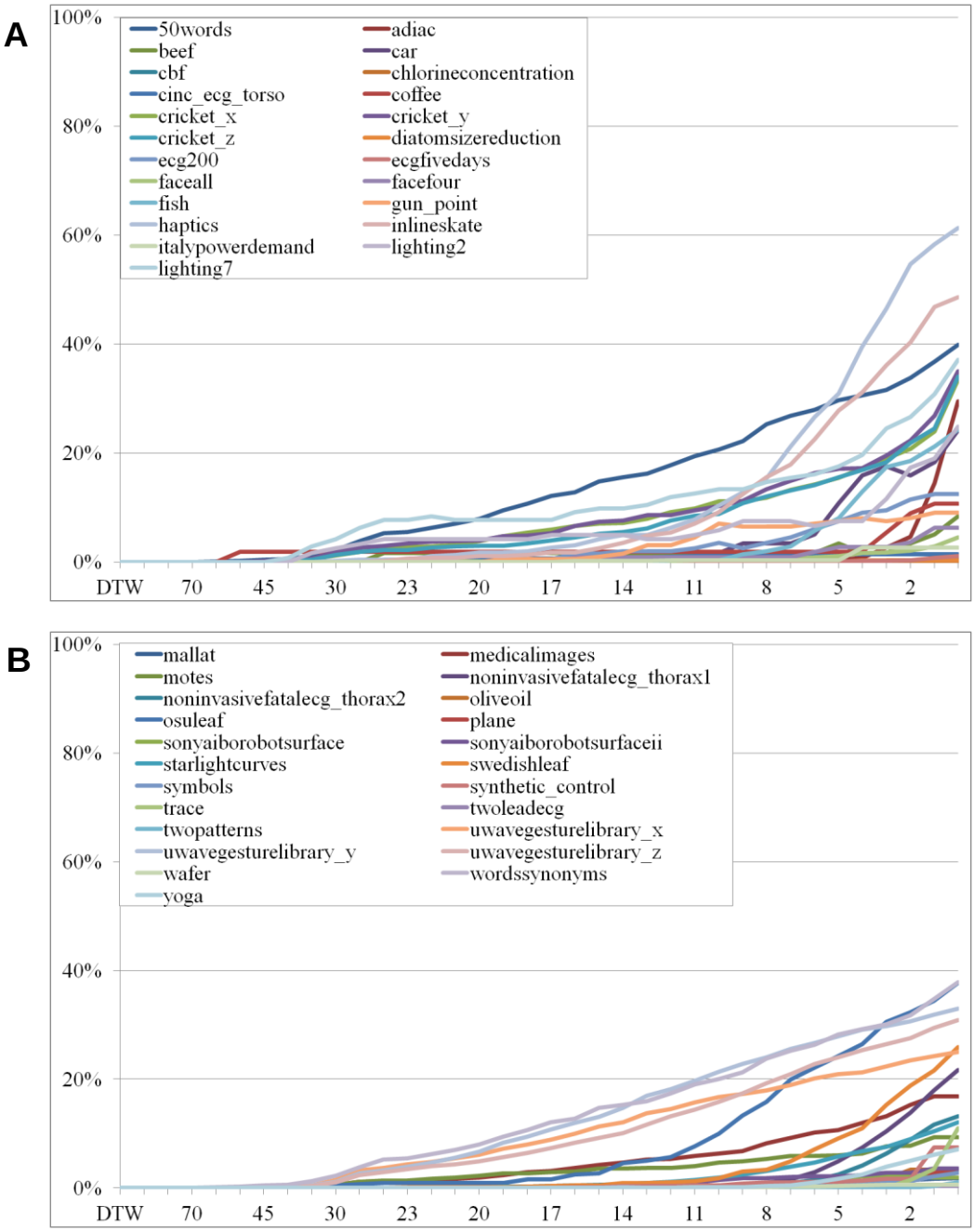


Figure D.1. Detailed plots of the change of classes for DTW

Slika D.1. Detaljni grafikoni promena klasa za DTW

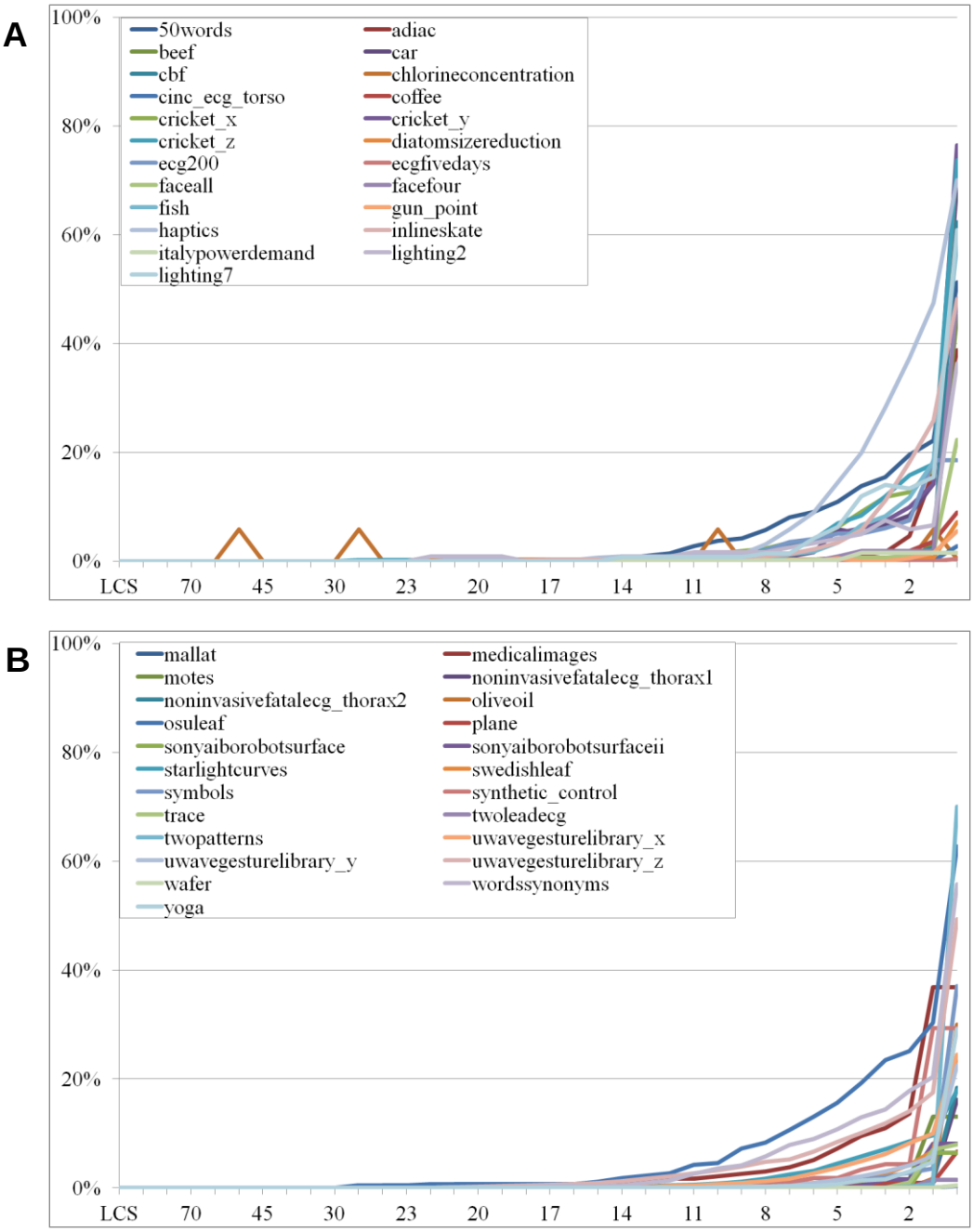


Figure D.2. Detailed plots of the change of classes for LCS

Slika D.2. Detaljni grafikoni promena klasa za LCS

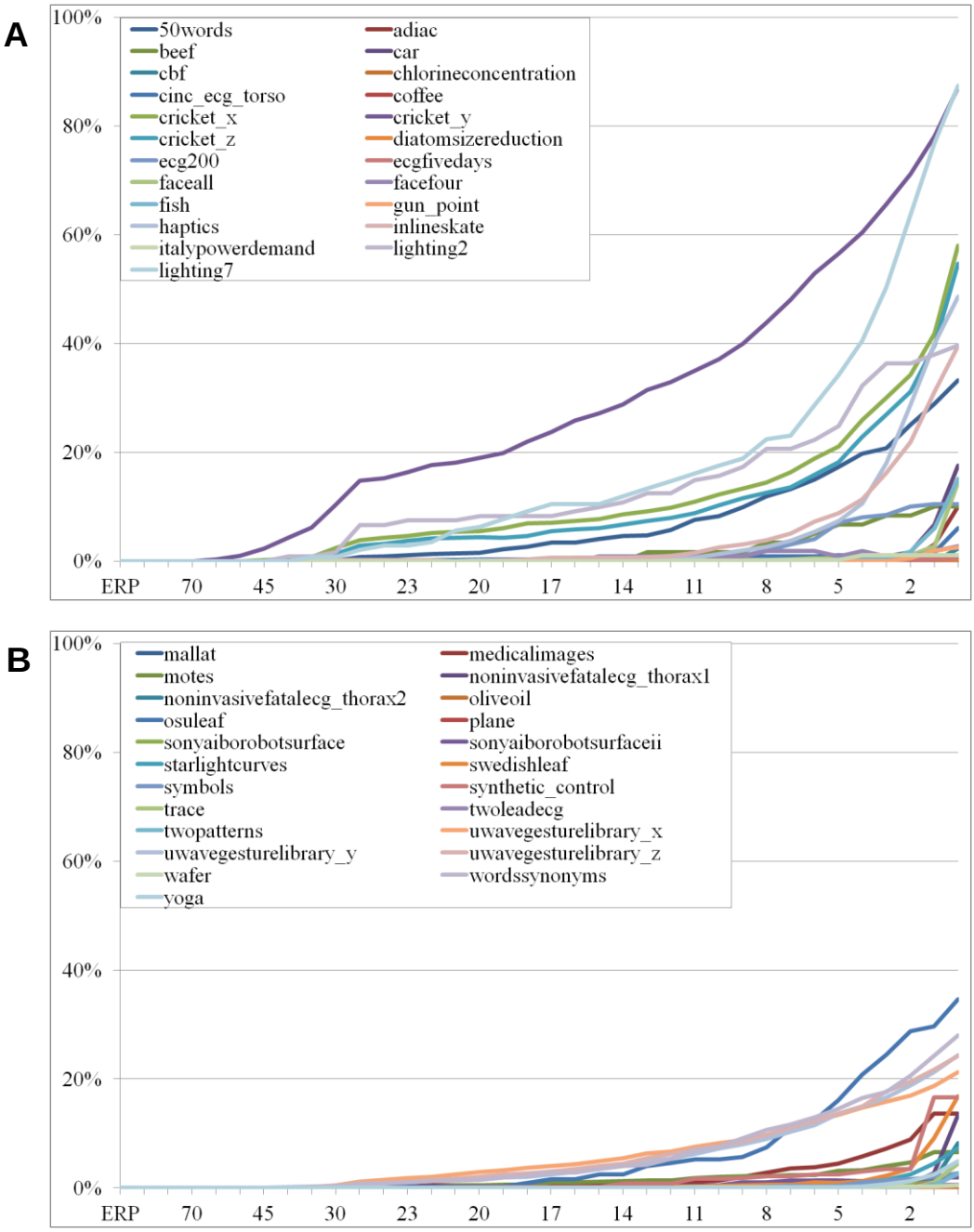


Figure D.3. Detailed plots of the change of classes for ERP

Slika D.3. Detaljni grafikoni promena klasa za ERP

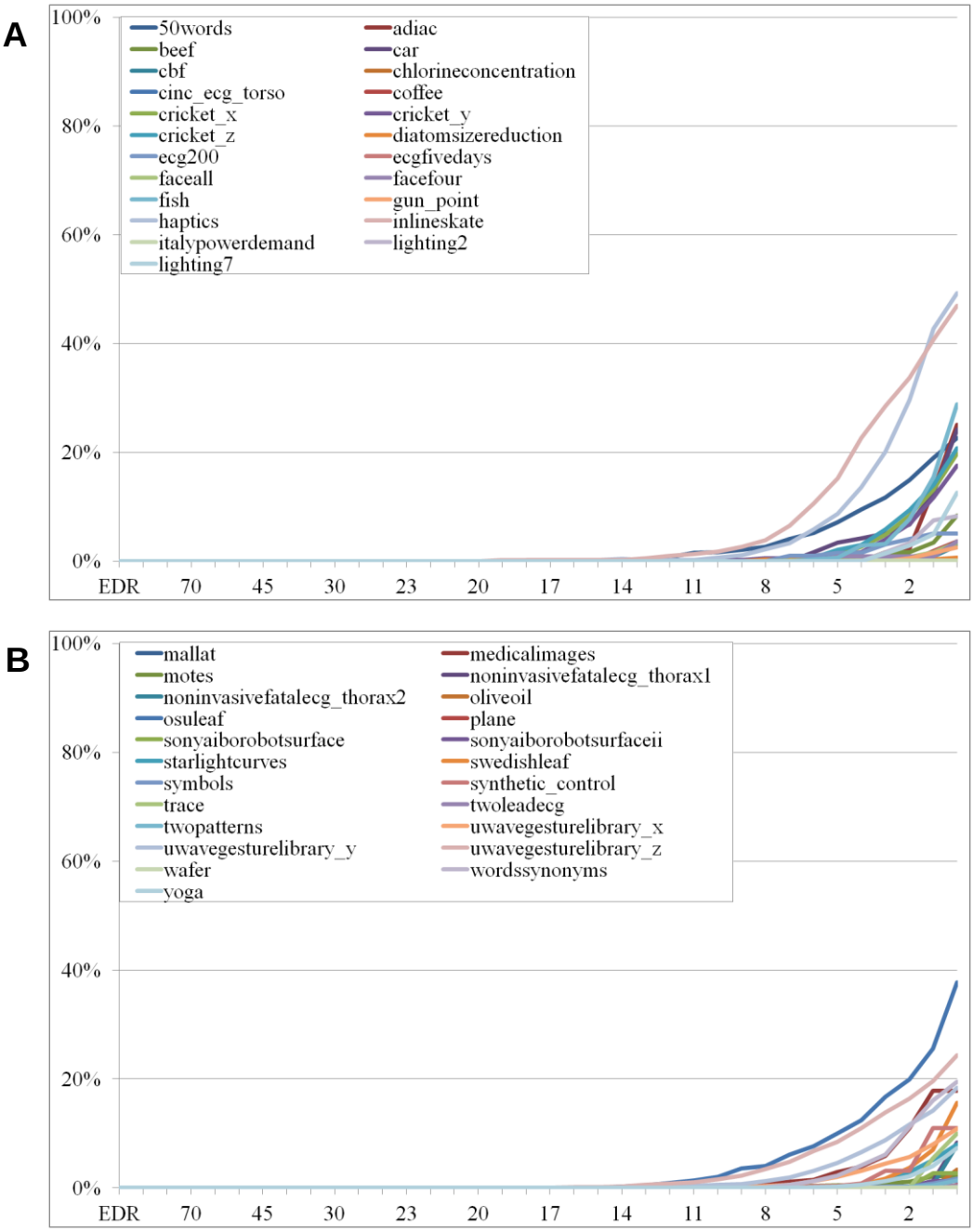


Figure D.4. Detailed plots of the change of classes for EDR

Slika D.4. Detaljni grafikoni promena klasa za EDR

Appendix E

ID	1NN	*k*NN		Inverse		ISquared		Rank		Fibonacci		Dudani		Macleod		DualD		Zavrel		Uniform		DualU	
	error	error	*k*	error	*k*	error	*k*	error	*k*	error	*k*	error	*k*	error	*k*	error	*k*	error	*k*	error	*k*	error	*k*
1	0.298	0.301	1.40	0.301	1.66	0.300	2.93	0.302	1.87	0.298	4.84	0.293	6.71	0.302	1.63	**0.292**	**7.24**	0.299	4.76	0.299	3.68	0.299	10.36
2	**0.315**	0.320	1.52	0.323	1.80	0.326	2.43	0.322	2.11	0.321	5.17	0.317	5.73	0.320	1.56	0.316	6.40	0.322	1.66	0.320	6.29	0.317	14.79
3	0.487	0.540	8.06	0.522	9.94	0.495	7.71	0.542	8.49	0.485	4.48	0.558	9.54	0.557	7.16	0.543	9.36	0.555	7.25	**0.483**	**10.38**	0.492	12.43
4	0.222	0.229	3.07	0.225	3.25	0.222	3.48	0.228	4.00	0.216	4.76	**0.204**	**6.13**	0.225	3.43	0.205	6.16	0.205	4.08	0.215	7.07	0.212	11.18
5	0.011	0.008	11.44	0.007	11.71	0.007	12.90	0.006	16.29	0.012	4.93	0.006	15.79	0.007	12.12	0.006	15.89	**0.006**	**13.54**	0.010	13.32	0.010	17.21
6	**0.002**	**0.002**	**1.00**	**0.002**	**1.00**	**0.002**	**1.00**	**0.002**	**1.00**	**0.002**	**1.00**	**0.002**	**1.00**	**0.002**	**1.00**	**0.002**	**1.00**	**0.002**	**1.00**	**0.002**	**1.00**	**0.002**	**2.11**
7	**0.001**	0.001	1.02	0.001	1.07	0.002	1.18	0.001	1.23	0.001	1.06	0.002	2.73	0.002	1.22	0.002	2.75	**0.001**	**1.04**	0.001	1.29	**0.001**	**1.98**
8	0.100	0.108	1.12	0.115	1.52	0.127	2.01	0.114	1.24	0.113	1.81	0.114	2.11	0.119	1.40	0.114	2.52	0.102	2.44	0.105	3.91	**0.099**	**6.41**
9	0.331	0.331	1.00	0.331	1.00	0.332	1.08	0.331	1.00	0.331	1.12	0.330	3.51	0.331	1.00	0.332	3.97	0.331	1.12	0.331	1.12	**0.328**	**13.32**
10	0.344	0.347	2.00	0.346	3.08	0.343	3.93	0.350	3.20	0.341	7.10	0.344	6.31	0.350	2.66	0.343	7.18	0.344	1.20	**0.338**	**8.10**	**0.338**	**18.17**
11	0.331	0.331	1.00	0.331	1.00	0.332	1.06	0.331	1.00	0.331	1.06	0.331	3.38	0.331	1.00	0.332	3.44	0.331	1.38	0.331	1.10	**0.328**	**14.61**
12	**0.001**	**0.001**	**1.00**	**0.001**	**1.00**	**0.001**	**1.00**	**0.001**	**1.00**	**0.001**	**1.00**	**0.001**	**1.00**	**0.001**	**1.00**	**0.001**	**1.00**	**0.001**	**1.00**	**0.001**	**1.00**	**0.001**	**1.00**
13	0.100	0.103	2.78	0.096	4.08	0.089	4.11	0.105	3.85	0.096	3.91	0.092	5.51	0.099	3.89	0.089	5.63	**0.085**	**4.18**	0.092	5.57	0.093	13.94
14	0.005	0.006	1.02	0.006	1.08	0.006	2.03	0.006	1.07	0.006	1.06	**0.005**	**2.92**	0.006	1.05	0.005	3.01	0.007	2.90	0.006	3.42	0.006	5.79
15	**0.040**	**0.040**	**1.00**	**0.040**	**1.00**	**0.040**	**1.00**	**0.040**	**1.00**	**0.040**	**1.00**	0.041	2.37	**0.040**	**1.00**	0.041	2.53	0.041	1.19	**0.040**	**1.00**	0.040	7.88
16	0.057	0.066	1.33	0.066	1.38	0.071	2.13	0.065	1.45	0.064	1.85	0.071	3.32	0.068	1.38	0.070	3.32	**0.055**	**3.33**	0.065	4.31	0.059	6.41
17	0.168	0.173	3.01	0.167	3.86	0.165	4.28	0.167	5.24	**0.153**	**6.35**	0.162	8.83	0.167	3.96	0.162	9.35	0.170	4.35	0.156	6.71	0.160	17.80
18	0.053	0.058	1.42	0.058	2.27	0.062	2.91	0.057	2.44	0.057	2.77	0.052	4.48	0.058	2.29	**0.050**	**4.50**	0.058	2.19	0.058	4.33	0.054	4.01
19	0.590	0.531	17.25	0.527	17.58	**0.522**	**18.73**	0.523	21.18	0.558	9.05	0.529	24.92	0.523	17.94	0.528	24.61	0.544	17.84	0.529	24.40	0.565	26.52
20	0.478	0.478	1.06	0.481	1.47	0.482	2.06	0.478	1.09	0.482	2.25	0.476	4.36	0.478	1.18	0.475	4.56	**0.474**	**6.62**	0.482	2.58	0.479	9.94
21	0.033	0.031	8.36	0.029	8.34	0.029	9.85	**0.028**	**10.29**	0.032	6.55	0.029	11.85	0.029	8.55	0.029	12.42	0.030	8.49	0.029	13.93	0.031	21.48
22	0.258	0.273	2.18	0.269	2.66	0.273	2.96	0.273	3.05	0.260	3.73	**0.241**	**5.84**	0.268	2.98	0.249	6.36	0.255	7.38	0.259	5.24	0.258	13.14
23	0.354	0.362	2.75	0.356	3.10	0.354	3.28	0.360	3.63	0.365	4.96	0.361	5.46	**0.352**	**3.06**	0.363	5.36	0.365	1.81	0.368	3.76	0.354	10.63
24	0.016	0.016	1.10	0.016	1.39	0.016	1.64	0.016	1.32	0.016	1.57	**0.015**	**5.04**	0.016	1.38	0.015	5.73	0.017	2.40	0.016	2.84	**0.015**	**14.71**
25	0.228	0.228	2.78	0.222	3.70	0.223	4.84	0.225	4.78	0.225	7.93	**0.211**	**8.50**	0.224	3.78	0.212	9.32	0.222	6.80	0.224	15.34	0.225	22.67
26	0.079	0.064	13.90	0.058	14.60	0.057	16.80	0.058	20.23	0.074	9.16	0.058	27.90	0.059	15.22	**0.057**	**28.47**	0.061	24.58	0.060	27.09	0.077	21.68
27	0.157	0.156	4.28	0.153	5.52	0.149	6.61	0.152	7.06	0.150	9.52	0.147	10.30	0.152	5.67	0.147	10.80	**0.145**	**15.87**	0.147	15.25	0.148	27.47
28	0.095	0.092	4.36	0.089	4.69	0.086	6.35	0.089	6.28	0.088	9.29	0.086	9.86	0.088	4.79	**0.086**	**10.89**	0.089	11.73	0.086	15.83	0.088	25.87
29	0.117	**0.112**	**2.02**	0.117	2.44	0.118	2.64	0.117	2.91	0.117	2.73	0.115	3.76	0.118	2.36	0.115	3.89	0.115	2.24	**0.112**	**5.84**	0.115	10.95
30	0.342	0.347	1.29	0.350	1.51	0.352	2.38	0.350	1.71	0.349	2.59	0.345	4.74	0.350	1.59	**0.339**	**5.11**	0.344	8.57	0.349	5.02	0.340	14.13
31	0.028	0.032	4.28	0.031	7.45	**0.026**	**9.39**	0.031	5.84	0.029	1.39	0.031	12.00	0.030	5.12	0.032	11.74	0.029	8.98	0.032	5.48	0.028	5.14
32	0.014	0.015	1.38	0.016	1.76	0.015	1.93	0.015	1.88	0.015	1.69	**0.014**	**3.84**	0.016	1.86	0.014	3.96	0.015	2.59	0.015	2.02	0.014	4.07
33	**0.014**	0.015	1.20	0.016	1.30	0.016	1.66	0.015	1.39	0.015	1.39	0.016	3.67	0.016	1.30	0.015	3.73	0.014	2.35	0.016	1.57	0.014	11.42
34	0.111	0.112	1.60	0.110	3.48	0.110	3.41	0.112	3.66	0.110	4.39	**0.108**	**5.54**	0.110	3.48	0.109	5.95	0.110	3.47	0.108	5.02	0.109	22.38
35	0.177	0.177	1.00	0.177	1.00	0.177	1.00	0.177	1.00	0.177	1.09	**0.174**	**4.08**	0.177	1.00	0.174	4.41	0.177	1.00	0.177	1.00	0.179	13.13
36	0.035	0.039	2.14	0.038	3.76	0.037	4.00	0.038	3.52	0.038	2.39	0.034	6.47	0.038	3.47	0.034	7.03	0.035	2.70	0.039	4.87	**0.031**	**18.27**
37	**0.077**	0.079	1.34	0.079	1.34	0.079	1.34	0.079	1.57	0.079	1.58	0.081	2.83	0.079	1.34	0.082	2.99	0.079	1.40	0.079	1.71	0.080	10.88
38	**0.119**	**0.119**	**1.00**	**0.119**	**1.00**	**0.119**	**1.00**	**0.119**	**1.00**	**0.119**	**1.00**	0.120	1.24	**0.119**	**1.00**	**0.119**	**1.39**	**0.119**	**1.00**	**0.119**	**1.00**	**0.119**	**1.15**
39	0.004	0.005	3.02	0.005	4.30	0.005	5.61	0.005	5.61	0.004	3.76	0.005	7.33	0.005	4.16	0.005	7.80	0.005	4.08	**0.004**	**4.47**	0.004	9.01
40	0.013	0.013	1.08	0.013	1.08	0.013	1.13	0.013	1.12	0.012	3.29	0.011	6.12	0.013	1.08	0.011	6.33	**0.011**	**4.05**	0.012	3.41	0.012	18.30
41	0.228	0.222	3.02	0.221	3.40	0.222	3.74	0.222	4.72	0.220	9.35	**0.218**	**6.39**	0.222	3.49	0.219	6.92	0.227	5.75	0.222	6.01	0.219	19.32
42	0.275	0.270	4.17	0.266	6.11	0.263	8.74	0.266	7.00	0.263	8.35	0.261	9.81	0.268	5.07	0.261	10.79	0.273	12.95	0.258	16.08	**0.257**	**28.98**
43	0.288	0.286	3.24	0.285	4.89	0.283	5.99	0.285	5.78	0.281	10.61	0.278	8.44	0.285	4.95	**0.277**	**8.74**	0.286	7.84	0.281	8.66	0.279	24.49
44	**0.001**	**0.001**	**1.00**	**0.001**	**1.00**	0.001	1.08	**0.001**	**1.00**	**0.001**	**1.00**	0.001	1.19	**0.001**	**1.00**	0.001	1.19	0.001	1.12	0.001	1.24	**0.001**	**1.21**
45	0.284	0.288	1.46	0.288	1.56	0.289	2.35	0.289	1.78	0.286	4.07	0.279	5.96	0.289	1.60	**0.278**	**6.19**	0.286	3.69	0.286	3.03	0.287	10.01
46	0.057	0.057	1.00	0.057	1.06	0.058	1.20	0.057	1.00	0.057	1.09	0.057	3.64	0.057	1.06	0.057	4.08	0.058	1.53	0.058	1.20	**0.056**	**11.92**

Table E.1. Classification errors and the values of parameter *k* obtained for Euclidean distance

Tabela E.1. Greške klasifikacije i vrednosti parametra *k* dobijenih za Euklidsko rastojanje

ID	1NN	*k*NN		Inverse		ISquared		Rank		Fibonacci		Dudani		Macleod		DualD		Zavrel		Uniform		DualU	
	error	error	*k*	error	*k*	error	*k*	error	*k*	error	*k*	error	*k*	error	*k*	error	*k*	error	*k*	error	*k*	error	*k*
1	0.279	0.266	3.00	0.261	3.95	0.264	5.18	0.267	4.31	0.263	8.74	0.262	6.13	0.267	3.27	**0.260**	**7.34**	0.268	6.17	0.266	6.92	0.265	21.77
2	0.336	0.337	1.33	0.343	2.12	0.343	2.20	0.342	2.47	0.337	4.30	0.338	6.31	0.341	1.81	0.336	6.82	0.339	1.53	0.341	5.39	**0.335**	**14.88**
3	0.477	0.542	5.42	0.513	7.90	0.512	4.58	0.542	5.78	0.485	3.13	0.543	6.44	0.542	5.34	0.540	7.18	0.545	5.46	**0.470**	**10.08**	0.475	11.59
4	**0.240**	0.253	1.27	0.269	3.01	0.261	5.53	0.263	1.93	0.259	2.98	0.258	4.29	0.265	2.06	0.256	5.02	0.264	2.48	0.260	4.09	0.251	9.62
5	0.000	**0.000**	**1.14**	**0.000**	**1.14**	**0.000**	**1.14**	**0.000**	**1.21**	**0.000**	**1.21**	**0.000**	**1.21**	**0.000**	**1.14**	**0.000**	**1.21**	**0.000**	**1.14**	**0.000**	**1.21**	**0.000**	**1.57**
6	0.003	**0.003**	**1.00**	**0.003**	**1.00**	**0.003**	**1.00**	**0.003**	**1.00**	**0.003**	**1.00**	**0.003**	**1.00**	**0.003**	**1.00**	**0.003**	**1.00**	**0.003**	**1.00**	**0.003**	**1.00**	**0.003**	**1.48**
7	0.015	**0.015**	**1.00**	**0.015**	**1.00**	0.015	1.34	**0.015**	**1.00**	**0.015**	**1.00**	0.016	1.89	**0.015**	**1.00**	0.015	2.43	0.015	2.56	**0.015**	**1.00**	0.015	2.73
8	**0.066**	0.076	1.02	0.087	1.24	0.088	1.68	0.085	1.15	0.085	1.24	0.082	1.16	0.083	1.11	0.087	1.35	0.074	1.00	0.087	1.89	0.076	1.70
9	0.176	0.181	1.55	0.181	1.76	0.181	2.65	0.180	1.80	0.179	4.05	0.177	4.12	0.181	1.66	0.174	4.49	**0.173**	**3.87**	0.178	6.91	0.175	15.99
10	0.168	0.172	1.43	0.164	3.08	0.167	6.59	0.171	2.43	0.170	4.98	0.163	6.05	0.172	2.06	**0.162**	**6.98**	0.168	5.94	0.174	10.08	0.169	16.77
11	**0.172**	0.175	1.28	0.176	1.60	0.179	3.28	0.176	1.65	0.177	2.77	0.175	4.05	0.177	1.40	0.173	4.97	0.173	4.84	0.180	4.59	0.176	7.70
12	0.004	**0.004**	**1.00**	**0.004**	**1.00**	**0.004**	**1.00**	**0.004**	**1.00**	**0.004**	**1.00**	**0.004**	**1.00**	**0.004**	**1.00**	**0.004**	**1.00**	**0.004**	**1.00**	**0.004**	**1.00**	**0.004**	**1.00**
13	0.168	0.183	3.04	0.165	4.02	0.157	4.99	0.183	4.07	0.176	3.37	0.156	5.30	0.182	3.86	**0.155**	**5.93**	0.164	4.80	0.174	3.86	0.168	12.70
14	**0.008**	0.009	1.28	0.010	2.01	0.010	2.20	0.009	1.61	0.009	1.87	0.009	2.81	0.009	1.68	0.009	2.82	0.008	1.24	0.009	2.23	0.009	7.02
15	0.022	0.022	1.00	0.023	1.19	0.024	1.85	0.022	1.04	0.022	1.37	0.021	4.07	0.022	1.06	0.021	4.60	**0.020**	**4.48**	0.022	1.47	0.021	15.33
16	0.054	0.073	3.05	0.072	3.62	0.071	3.65	0.070	4.28	0.064	1.91	0.070	5.85	0.070	3.58	0.071	7.18	0.055	1.00	**0.050**	**11.08**	0.057	12.51
17	0.204	0.205	3.01	0.188	3.82	**0.185**	**4.28**	0.204	4.82	0.202	4.48	0.193	6.07	0.191	3.54	0.190	6.98	0.189	3.59	0.201	9.58	0.196	13.46
18	0.085	0.092	2.30	0.092	4.67	**0.080**	**5.89**	0.090	4.67	0.091	2.41	0.084	6.43	0.091	3.91	0.085	7.01	0.094	3.92	0.091	7.75	0.088	9.96
19	0.579	0.534	16.72	0.532	16.63	0.535	16.18	0.530	18.61	0.552	9.88	0.526	17.92	0.533	16.64	**0.524**	**19.29**	0.538	20.64	0.531	22.63	0.561	26.71
20	0.456	0.455	2.46	**0.419**	**13.60**	0.428	15.81	0.466	6.60	0.459	5.72	0.460	10.29	0.469	4.92	0.449	12.35	0.447	9.16	0.457	11.23	0.455	20.14
21	0.046	0.040	5.58	0.039	6.17	**0.036**	**7.11**	0.037	8.28	0.038	6.28	0.039	12.15	0.038	6.68	0.038	12.54	0.040	6.27	0.039	13.28	0.039	23.92
22	**0.104**	0.110	1.20	0.116	1.65	0.115	1.59	0.111	1.43	0.113	1.78	0.114	2.68	0.118	1.63	0.111	2.76	0.106	1.90	0.116	2.66	0.106	6.98
23	0.276	0.243	4.58	0.239	5.61	0.257	7.56	0.250	6.59	0.238	6.80	0.232	9.19	0.242	5.55	**0.226**	**10.44**	0.267	4.10	0.228	13.57	0.253	21.96
24	0.014	0.013	2.66	0.011	3.80	0.011	4.66	0.012	4.75	0.012	5.53	0.011	7.79	0.011	3.94	0.011	8.39	0.011	4.04	0.012	8.09	**0.010**	**23.15**
25	0.197	0.207	2.72	0.190	8.33	0.187	12.48	0.199	6.04	0.195	7.68	0.195	9.30	0.199	5.08	0.193	11.98	0.187	18.02	**0.186**	**20.70**	0.192	25.97
26	0.046	**0.046**	**1.02**	0.047	1.70	0.049	4.91	0.046	1.20	0.047	1.87	0.047	2.23	0.046	1.22	0.047	3.02	0.047	3.55	0.048	3.29	0.047	9.07
27	0.192	0.179	5.32	0.177	5.93	0.177	6.85	0.176	8.83	0.177	10.49	0.171	12.35	0.175	6.67	**0.170**	**14.18**	0.175	6.34	0.175	18.21	0.182	28.84
28	0.121	0.123	4.37	0.119	6.75	0.116	10.03	0.117	8.55	0.115	8.67	0.113	11.52	0.117	7.02	**0.113**	**12.84**	0.118	6.20	0.114	21.28	0.116	25.95
29	0.112	0.118	1.94	0.118	2.17	0.120	1.99	0.122	2.56	0.105	3.28	0.123	3.60	0.120	2.16	0.123	3.60	0.120	1.99	**0.103**	**4.27**	0.118	10.11
30	0.290	0.293	1.13	0.307	4.32	0.293	6.58	0.303	2.52	0.296	3.54	0.301	6.46	0.303	2.22	0.297	7.21	**0.288**	**5.96**	0.297	7.08	0.294	13.61
31	**0.000**	**0.000**	**1.00**	**0.000**	**1.00**	**0.000**	**1.00**	**0.000**	**1.00**	**0.000**	**1.00**	**0.000**	**1.00**	**0.000**	**1.00**	**0.000**	**1.00**	**0.000**	**1.00**	**0.000**	**1.00**	**0.000**	**1.00**
32	**0.023**	0.026	1.08	0.026	1.11	0.026	1.11	0.026	1.16	0.026	1.15	0.027	2.79	0.026	1.22	0.027	2.77	0.027	1.38	0.026	1.30	0.025	3.23
33	0.027	0.029	1.28	0.029	2.01	0.029	2.16	0.029	1.61	0.029	1.87	0.028	3.68	0.029	1.74	0.028	3.76	**0.025**	**3.49**	0.029	2.49	0.028	10.43
34	0.065	0.061	4.01	0.059	4.81	0.060	4.63	0.060	5.77	0.060	8.16	**0.057**	**8.06**	0.059	4.81	0.057	8.52	0.061	4.69	0.059	11.90	0.059	27.50
35	0.184	0.183	2.96	0.184	3.11	0.184	3.03	0.183	4.23	0.179	7.75	0.175	6.40	0.183	3.29	**0.175**	**6.45**	0.185	3.17	0.182	5.84	0.179	19.08
36	0.018	0.019	1.36	**0.016**	**2.65**	0.017	3.01	0.019	1.54	0.019	1.81	0.017	4.71	0.019	1.70	0.017	5.26	0.018	1.32	0.019	2.08	0.017	11.01
37	0.008	0.005	4.06	0.005	4.20	0.005	4.46	0.005	5.30	0.004	3.99	0.006	7.92	0.005	4.33	0.006	7.74	0.007	4.44	**0.004**	**4.47**	0.006	24.52
38	**0.000**	**0.000**	**1.00**	**0.000**	**1.00**	**0.000**	**1.00**	**0.000**	**1.00**	**0.000**	**1.00**	**0.000**	**1.00**	**0.000**	**1.00**	**0.000**	**1.00**	**0.000**	**1.00**	**0.000**	**1.00**	**0.000**	**1.00**
39	0.001	0.002	2.46	0.001	3.96	**0.001**	**4.48**	**0.001**	**1.90**	**0.001**	**1.06**	**0.001**	**2.32**	0.001	2.90	**0.001**	**1.91**	0.002	2.56	**0.001**	**1.41**	**0.001**	**1.00**
40	**0.000**	**0.000**	**1.00**	**0.000**	**1.00**	**0.000**	**1.00**	**0.000**	**1.00**	**0.000**	**1.00**	**0.000**	**1.00**	**0.000**	**1.00**	**0.000**	**1.00**	**0.000**	**1.00**	**0.000**	**1.00**	**0.000**	**1.00**
41	0.248	0.224	6.55	0.225	7.19	0.228	8.68	0.223	9.05	0.231	9.96	**0.221**	**12.86**	0.223	7.14	0.222	14.25	0.230	16.77	0.227	18.86	0.232	28.85
42	0.340	0.293	6.78	0.294	11.48	0.298	15.75	0.291	11.73	0.304	13.36	0.289	15.40	0.292	8.41	**0.289**	**17.82**	0.301	17.64	0.291	24.84	0.302	29.65
43	0.301	0.283	6.50	0.279	7.45	0.279	10.57	0.279	8.88	0.281	10.48	0.276	11.68	0.281	6.98	**0.274**	**13.04**	0.278	22.99	0.276	18.74	0.280	28.99
44	0.006	0.006	1.00	0.006	1.02	0.006	1.13	0.006	1.00	0.006	1.00	0.006	2.55	0.006	1.00	0.006	2.53	0.006	1.33	0.006	1.00	**0.006**	**6.69**
45	0.267	0.255	2.83	0.251	3.84	0.252	4.92	0.257	4.24	0.249	8.41	0.250	6.16	0.257	3.13	**0.247**	**7.24**	0.253	6.11	0.254	7.70	0.250	22.71
46	0.060	0.060	1.00	0.061	2.15	0.060	2.82	0.060	1.00	0.061	1.24	0.059	4.02	0.060	1.09	**0.059**	**4.39**	0.060	1.38	0.061	1.53	0.059	12.30

Table E.2. Classification errors and the values of the parameter *k* obtained for DTW

Tabela E.2. Greške klasifikacije i vrednosti parametra *k* dobijenih za DTW

ID	1NN	*k*NN		Inverse		ISquared		Rank		Fibonacci		Dudani		Macleod		DualD		Zavrel		Uniform		DualU	
	error	error	*k*	error	*k*	error	*k*	error	*k*	error	*k*	error	*k*	error	*k*	error	*k*	error	*k*	error	*k*	error	*k*
1	0.163	0.165	1.25	0.167	1.98	0.168	2.95	0.167	2.41	0.166	3.13	0.167	5.54	0.168	1.71	0.166	5.68	0.166	1.53	0.167	3.90	**0.161**	**16.99**
2	0.363	0.350	3.34	0.337	4.29	0.345	4.09	0.348	5.10	0.347	8.00	0.336	7.07	0.346	4.04	**0.332**	**8.21**	0.349	3.77	0.350	10.51	0.344	19.13
3	**0.423**	0.458	2.53	0.463	3.19	0.428	1.12	0.473	4.12	0.430	4.32	0.472	3.66	0.460	3.25	0.450	4.07	0.473	3.49	0.440	7.06	0.428	8.27
4	**0.107**	**0.107**	**1.00**	0.108	1.03	0.111	1.25	**0.107**	**1.04**	**0.107**	**1.18**	0.108	1.70	0.109	1.05	**0.107**	**1.59**	0.108	1.03	**0.107**	**1.38**	**0.107**	**2.99**
5	**0.000**	**0.000**	**1.82**	**0.000**	**1.82**	**0.000**	**1.82**	**0.000**	**2.19**	**0.000**	**2.14**	**0.000**	**3.37**	**0.000**	**2.00**	**0.000**	**3.37**	**0.000**	**1.82**	**0.000**	**2.46**	**0.000**	**6.07**
6	0.011	0.011	1.00	0.009	2.52	0.011	1.00	0.011	1.00	0.011	1.00	**0.007**	**3.00**	0.011	1.00	**0.007**	**3.00**	0.011	1.00	0.011	1.00	0.008	3.70
7	**0.000**	**0.000**	**1.06**	**0.000**	**1.06**	**0.000**	**1.06**	**0.000**	**1.09**	**0.000**	**1.00**	**0.000**	**1.03**	**0.000**	**1.06**	**0.000**	**1.03**	**0.000**	**1.06**	**0.000**	**1.00**	**0.000**	**1.00**
8	**0.061**	0.063	1.02	0.068	2.01	0.066	1.08	0.063	1.03	0.064	1.09	0.062	1.09	0.063	1.02	0.064	1.18	0.063	1.02	**0.061**	**1.74**	**0.061**	**1.04**
9	0.187	0.192	2.33	0.187	2.70	0.187	3.31	0.191	3.25	0.183	6.25	0.184	5.68	0.190	2.57	0.184	6.49	0.189	2.56	0.184	7.51	**0.182**	**19.18**
10	0.158	0.161	1.23	0.162	1.64	0.168	3.26	0.163	2.01	0.162	3.09	0.164	5.66	0.162	1.65	0.161	6.52	0.160	1.40	0.162	8.71	**0.150**	**19.53**
11	0.181	0.184	1.34	0.186	1.69	0.190	2.45	0.185	1.77	0.185	4.26	0.186	5.69	0.187	1.68	0.187	6.74	0.185	1.47	0.184	4.96	**0.181**	**16.61**
12	**0.007**	0.007	1.42	0.007	1.42	0.007	1.42	0.007	1.63	**0.007**	**1.60**	**0.007**	**1.76**	**0.007**	**1.52**	**0.007**	**1.77**	0.007	1.42	**0.007**	**1.60**	**0.007**	**1.00**
13	**0.098**	**0.098**	**1.00**	0.101	1.26	0.102	1.37	**0.098**	**1.04**	0.102	1.48	**0.098**	**2.43**	0.100	1.21	0.099	2.85	0.100	1.24	0.103	1.97	0.101	6.25
14	**0.001**	**0.001**	**1.00**	0.001	1.26	0.001	1.03	**0.001**	**1.00**	0.001	1.03	0.001	1.66	0.001	1.16	0.001	1.66	0.001	1.03	0.001	1.03	0.001	1.97
15	0.009	0.010	1.70	0.010	2.55	0.010	2.78	0.010	2.40	**0.009**	**4.45**	0.009	5.23	0.010	2.36	0.009	5.24	0.010	2.01	0.009	5.17	0.009	7.94
16	**0.009**	0.022	4.90	0.022	5.65	0.016	8.44	0.027	5.77	**0.009**	**1.00**	0.010	1.27	0.024	5.13	**0.009**	**1.00**	0.023	5.30	**0.009**	**1.09**	**0.009**	**1.11**
17	0.098	0.092	5.46	0.076	6.14	0.077	7.67	0.083	8.59	0.094	6.39	**0.073**	**14.71**	0.082	6.84	**0.073**	**15.85**	0.083	5.62	0.081	12.83	0.100	15.15
18	0.008	0.010	1.30	0.010	1.57	0.007	2.49	0.010	1.59	0.010	1.54	0.007	3.17	0.011	1.43	**0.006**	**3.47**	0.010	1.33	0.010	1.97	0.007	6.61
19	0.531	0.514	10.32	0.496	9.91	0.497	11.23	0.502	12.47	0.510	7.91	0.492	17.94	0.501	10.08	**0.488**	**19.90**	0.499	9.45	0.507	18.87	0.521	21.68
20	0.401	0.398	3.76	0.372	9.42	0.376	7.65	0.392	6.00	0.391	7.67	**0.369**	**7.87**	0.386	4.35	0.371	10.28	0.390	4.26	0.394	10.22	0.390	23.95
21	0.036	0.033	3.88	0.031	4.17	**0.031**	**5.50**	0.032	5.53	0.033	5.02	0.032	8.58	0.031	5.12	0.032	8.48	0.032	4.47	0.033	5.89	0.033	6.50
22	**0.146**	0.152	1.42	0.166	2.01	0.179	3.45	0.163	2.17	0.154	1.48	0.173	4.35	0.169	2.32	0.173	4.56	0.167	1.92	0.173	4.71	0.161	8.53
23	**0.269**	0.278	4.98	0.273	5.52	0.272	6.22	0.277	7.34	0.272	6.58	0.285	9.25	0.273	5.44	0.280	9.52	0.274	5.24	0.273	11.73	0.270	17.42
24	0.069	0.032	22.92	0.031	25.64	0.031	23.29	0.032	27.36	0.053	8.35	0.031	25.27	0.031	25.40	**0.030**	**25.46**	0.032	23.38	0.041	27.94	0.031	24.64
25	0.229	0.232	3.37	0.225	5.51	0.217	6.92	0.222	6.08	**0.211**	**10.89**	0.212	12.59	0.224	5.92	0.212	14.22	0.222	4.36	0.215	14.76	0.221	26.15
26	0.015	0.016	1.36	0.017	2.70	0.017	4.12	0.017	2.45	0.016	3.07	0.016	4.28	0.017	2.45	0.017	5.49	0.017	2.32	0.016	7.74	**0.014**	**12.51**
27	0.184	0.155	6.33	**0.150**	**8.05**	0.153	6.70	0.154	9.24	0.157	10.06	0.151	13.48	0.152	7.60	0.151	17.26	0.153	6.50	0.153	21.12	0.157	28.57
28	0.114	0.096	7.75	0.092	14.90	0.095	8.45	0.094	12.54	0.097	10.60	0.092	23.00	0.095	9.50	**0.090**	**24.14**	0.095	8.10	0.094	22.78	0.100	28.98
29	**0.115**	0.117	2.36	0.123	3.31	0.117	3.11	0.120	3.46	**0.115**	**3.26**	0.120	4.15	0.118	2.81	0.120	4.37	0.117	2.75	**0.115**	**3.88**	**0.115**	**2.81**
30	0.131	0.138	1.94	0.138	2.94	0.141	3.23	0.137	2.84	0.136	4.92	0.132	7.27	0.137	3.16	0.130	7.95	0.137	2.80	0.134	7.63	**0.126**	**17.03**
31	**0.000**	0.001	1.14	0.001	1.24	0.001	1.24	**0.000**	**1.24**	**0.000**	**1.06**	**0.000**	**1.18**	**0.000**	**1.20**	**0.000**	**1.18**	0.001	1.24	**0.000**	**1.19**	**0.000**	**1.27**
32	0.016	0.019	1.92	0.020	2.84	0.019	2.95	0.018	2.51	0.017	2.41	0.018	4.81	0.017	2.78	0.018	5.15	0.019	2.46	0.018	4.67	**0.016**	**10.60**
33	**0.023**	0.024	1.16	0.025	1.50	0.025	1.81	0.024	1.29	0.024	1.45	0.025	3.53	0.025	1.49	0.025	3.49	0.024	1.34	0.024	1.68	0.024	9.00
34	0.092	0.087	3.20	0.086	4.32	0.086	4.38	0.085	5.61	0.086	6.58	0.085	9.07	0.086	4.43	**0.085**	**9.06**	0.086	4.31	0.086	6.80	0.086	28.28
35	0.090	0.093	1.59	0.093	1.94	0.093	2.09	0.093	2.11	0.093	3.10	**0.084**	**4.85**	0.092	2.31	0.086	5.00	0.093	1.91	0.092	5.28	0.087	14.42
36	**0.012**	0.014	1.92	0.014	3.91	0.014	4.24	0.013	1.80	0.013	1.51	**0.012**	**4.86**	0.014	2.92	**0.012**	**5.00**	0.015	3.18	0.014	5.86	**0.012**	**9.76**
37	0.022	0.020	7.63	0.018	7.97	0.018	8.18	0.019	9.33	0.022	5.59	0.021	11.99	**0.017**	**8.40**	0.021	12.25	0.018	7.91	0.019	10.01	0.021	15.64
38	**0.001**	0.003	1.12	0.002	1.29	0.002	1.13	0.003	1.30	0.002	1.18	**0.001**	**1.41**	0.004	1.27	**0.001**	**1.55**	0.003	1.26	0.002	1.32	**0.001**	**1.38**
39	**0.001**	0.002	3.22	0.002	3.40	**0.001**	**2.40**	0.002	3.39	0.001	2.11	**0.001**	**2.92**	0.001	2.68	**0.001**	**2.92**	0.002	3.16	0.001	2.32	**0.001**	**2.56**
40	**0.000**	**0.000**	**1.02**	**0.000**	**1.02**	**0.000**	**1.02**	**0.000**	**1.03**	**0.000**	**1.03**	**0.000**	**1.02**	**0.000**	**1.02**	**0.000**	**1.02**	**0.000**	**1.02**	**0.000**	**1.03**	**0.000**	**1.02**
41	0.213	0.193	5.55	0.193	6.15	0.193	6.29	0.190	7.57	0.193	10.57	0.189	10.33	0.191	5.77	**0.189**	**11.62**	0.191	5.59	0.189	14.33	0.192	27.14
42	0.299	0.267	8.24	0.263	10.14	0.260	10.12	0.262	11.26	0.271	12.00	0.260	13.76	0.261	8.83	**0.259**	**15.44**	0.262	8.36	0.263	24.65	0.270	27.94
43	0.251	0.240	3.74	0.240	4.66	0.239	5.05	**0.237**	**5.96**	0.237	9.16	0.238	9.81	0.238	4.57	0.237	11.02	0.239	4.31	0.238	11.96	0.239	27.44
44	0.002	0.002	1.10	0.002	1.38	0.002	2.62	0.002	1.16	0.002	1.21	0.002	3.29	0.002	1.52	0.002	3.66	0.002	1.14	0.002	1.69	**0.002**	**8.28**
45	0.154	0.156	1.29	0.157	1.88	0.159	2.52	0.157	1.82	0.156	3.23	0.156	4.84	0.157	1.80	0.153	5.20	0.156	1.56	0.157	4.15	**0.150**	**19.07**
46	0.030	0.031	1.02	0.031	1.23	0.032	1.70	0.031	1.03	0.031	1.69	0.030	5.13	0.031	1.32	**0.029**	**5.40**	0.031	1.29	0.032	2.07	0.030	17.39

Table E.3. Classification errors and the values of the parameter k obtained for LCS

Tabela E.3. Greške klasifikacije i vrednosti parametra k dobijenih za LCS

Sažetak

Vremenska serija predstavlja najjednostavniji oblik temporalnih podataka - niz brojeva koji opisuje promenu posmatrane pojave u toku vremena. Svaki broj vremenske serije opisuje dati fenomen u jednoj tački vremena [19]. Vremenske serije se koriste za skladištenje, prikaz i analizu podataka u širokom spektru različitih domena, uključujući razne oblasti nauke, medicine, ekonomije, ekologije, telekomunikacija i meteorologije [25, 40, 63]. U mnogim naučnim oblastima, merenja se vrše tokom vremena a prikupljeni podaci se mogu organizovati u obliku vremenskih serija radi pronalaženja korisnih informacija [25]. U pronalaženju novih i korisnih informacija iz prikupljenih podataka možemo se osloniti na metode statističke analize i modeliranja, odnosno na *data mining* i mašinsko učenje [63].

Istraživanje i analiza tehnika statističkog modeliranja ima dugu istoriju i upotrebu u raznim oblastima [63], ali rastuća potreba za obradom sve većih količina podataka doprinela je značajnijem proučavanju različitih zadataka *mining*-a vremenskih serija [21, 39, 95]:

- Indeksiranje (eng. *indexing*) - nalaženje vremenske serije u bazi podataka koja je najsličnija datoj vremenskoj seriji Q na osnovu date mere sličnosti/različitosti d.

- Klasifikacija (eng. *classification*) - svrstavanje date vremenske serije u jednu ili više predifinisanih klasa.

- Grupisanje (eng. *clustering*) - nalaženje prirodnog načina grupisanja vremenskih serija neke baze podataka na osnovu date mere sličnosti/različitosti d.

- Predviđanje (eng. *predicting*) - predviđanje budućih vrednosti vremenskih serija: na osnovu date vremenske serije Q koja sadrži n tačaka predvideti vrednost za $(n+1)$-tu tačku.

- Sumiranje (eng. *summarization*) - za datu vremensku seriju Q koja sadrži ekstremno veliki broj tačaka napraviti aproksimaciju (smanjenjem broja tačaka) koja će da zadrži suštinske osobine te serije.

- Otkrivanje anomalija (eng. *anomaly detection*) - na osnovu date vremenske serije Q za koju se smatra da je pravilna (normalna) naći sve delove još neprotumačene vremenske serije S koji predstavljaju anomalije ili neočekivane, interesantne pojave.

- Segmentacija (eng. *segmentation*) - razlikujemo dve podoblasti:

    - Za datu vremensku seriju Q koja sadrži n tačaka napraviti model $\bar{Q}$ na osnovu K pojedinačnih segmenata ($K \ll n$) tako da $\bar{Q}$ bude bliska aproksimaciju serije Q.

    - Datu vremensku seriju Q podeliti na K interno homogenih delova.

Među ovim zadacima, u poslednje vreme, sve više pažnje se posvećuje istraživanju različitih aspekata klasifikacije vremenskih serija [34, 37, 45, 93, 111, 130]. Klasifikacija predstavlja proces grupisanja vremenskih serija u unapred definisane kategorije, klase a vrši se na osnovu jednog izabranog atributa (oznake klase - eng. *class label*) koji može imati konačan broj različitih vrednosti - broj klasa je na ovaj način unapred poznat. To je ključna razlika između klasifikacije i grupisanja: u slučaju grupisanja ne znamo unapred koliko ćemo imati grupa - one se otkrivaju u toku samog procesa grupisanja. Na ovaj način, klasifikaciju možemo posmatrati kao nadgledano učenje (eng. *supervised learning*) a grupisanje kao nenadgledano učenje (eng. *unsupervised learning*). Zadatak algoritma nadgledanog učenja je da nađe funkciju koja će dati predviđanja za nove, nepoznate objekte na osnovu označenih, poznatih primera iz skupa obuka [46, 78]. S druge strane, zadatak algoritma nenadgledanog učenja je da nauči prepoznavanje obrasce isključivo pomoću neoznačenih objekata [46, 78].

Veliki broj radova je posvećen mogućnostima primene mnogih dobro poznatih tehnika mašinskog učenja kao što su: stabla odlučivanja [102], neuronske mreže [81], metoda potpornih vektora (eng. *support vector machines*) [128], pravila logike prvog reda (eng. *first order logic rules*) [101], Bejzov klasifikator [85]. Međutim, pokazano je da, u slučaju vremenskih serija, jedan od najjednostavnijih metoda tj. metoda najbližih suseda (eng. *nearest-neighbor rule*) često daje bolje rezultate od ovih složenijih metoda [130].

Metoda najbližeg suseda (skraćeno 1NN) je verovatno jedan od najcenjenijih *data mining* algoritama [127]. Ona se zasniva na veoma jednostavnoj ideji: nepoznati, još neklasifikovani uzorci stavljaju se u klasu njihovih najbližih suseda [18]. Metoda k najbližih suseda (eng. *k-nearest neighbor*) predstavlja uopštenje 1NN klasifikatora [28]: tražimo k najbližih suseda i, za još neklasifikovani uzorak, biramo onu klasu kojoj pripada najveći broj njegovih suseda. Jedan od mogućih nedostataka ovog pristupa jeste da su pri izboru klase svi susedi ravnopravni. Ovaj nedostatak možemo pokušati popraviti dodeljivanjem težina susedima u skladu sa njihovim rastojanjem od uzorka koji želimo da klasifikujemo [22]. U literaturi predložen je veći broj različitih načina računanja težina najbližih suseda [22, 35, 36, 62, 69, 74, 84, 133]. Svaki od ovih radova koji opisuje novi način računanja težina izveštava o superiornosti te nove metode u odnosu na neka prethodna rešenja, zaključci su obično zasnovani na upoređivanju tačnosti klasifikacije oslanjajući se isključivo na Euklidsko rastojanje, i na osnovu relativno malog broja skupova podataka koji nisu iz domena vremenskih serija. Metoda najbližih suseda i različiti načini računanja težina koji predstavljaju predmet istraživanja ove disertacije opisani su u odeljku 3.1.

Pošto nalaženje najbližeg suseda predstavlja osnovnu ideju 1NN metode, jedan od najbitnijih pitanja implementacije odnosi se na izbor odgovarajuće mere rastojanja - ona opisuje koliko su dve vremenske serije slične odnosno različite. Međutim, za razliku od tradicionalnih baza podataka kod kojih je definicija sličnosti/različitosti između podataka jednostavan, rastojanje između vremenskih serija treba pažljivo definisati tako da verno opisuje sličnost/različitost između pojava/objekata predstavljenih u obliku vremenskih serija. Rastojanje između dve vremenske serije definišemo pomoću nenegativne funkcije rastojanja koja opisuje stepen različitosti između njih [25]. Veće rastojanje označava manju sličnost između vremenskih serija i obrnuto.

U domenu vremenskih serija koristi se nekoliko različitih mera zasnovanih na sličnost serija [31, 68, 108]: Euklidsko rastojanje [26], dinamičko iskrivljenje vremena (eng. *Dynamic Time Warping* - DTW) [10], najduži zajednički podniz (eng. *Longest Common Subsequence* - LCS) [121], rastojanje uređivanja sa realnom kaznom (eng. *Edit Distance with Real Penalty* - ERP) [16], rastojanje uređivanja nad realnim serijama (eng. *Edit Distance on Real sequence* - EDR) [17] i druge. Pregled najčešće korišćenih mera sličnosti dat je u odeljku 2.1.

Euklidsko rastojanje se zasniva na linearno uparivanje tačaka vremenskih serija: *i*-ta tačka prve serije upoređuje se sa *i*-tom tačkom druge serije. Da bi poboljšali procenu sličnosti, elastične mere kao što su DTW, LCS, ERP I EDR omogućavaju i nelinearno uparivanje tačaka: jedna tačka prve serije može biti uparena sa nizom susednih tačaka druge serije i obrnuto.

Osnovnu tehniku implementacije većine navedenih mera sličnosti predstavlja dinamičko programiranje koja zbog složenosti izračunavanja često nije pogodna za primenu u većim problemima iz realnog sveta. Za razrešenje ovog problema možemo pokušati sa ograničavanjem oblasti pretrage uvođenjem globalnih ograničenja kao što su Sakoe-Chiba pojas [105] i Itakura paralelogram [44] - način primene ovih ograničenja prikazan je u odeljku 2.2. Pored ubrzanja računanja, sugerisano je i da upotreba globalnih ograničenja može poboljšati tačnost klasifikacije [98, 130].

Predmet istraživanja ove disertacije obuhvata detaljan pregled i analizu uticaja Sakoe-Chiba globalnog ograničenja [105] na najčešće korišćene elastične mere sličnosti u oblasti *data mining*-a vremenskih serija sa naglaskom na tačnost klasifikacije. Izbor mere sličnosti jedan je od najvažnijih aspekata analize vremenskih serija - ona treba verno reflektovati sličnost između podataka prikazanih u obliku vremenskih serija. Mera sličnosti predstavlja kritičnu komponentu mnogih zadataka *mining*-a vremenskih serija, uključujući klasifikaciju, grupisanje (eng. *clustering*), predviđanje, otkrivanje anomalija i drugih.

Istraživanje obuhvaćeno ovom disertacijom usmereno je na nekoliko pravaca:

1. pregled efekata globalnih ograničenja na performanse računanja mera sličnosti (odeljak 5.2),

2. detaljna analiza posledice ograničenja elastičnih mera sličnosti na tačnost klasifikacije klasičnih tehnika klasifikacije (odeljci 5.3 i 5.4),

3. opsežna studija uticaja različitih načina računanja težina (eng. *weighting scheme*) na klasifikaciju vremenskih serija (odeljak 5.5),

4. razvoj biblioteke otvorenog koda (*Framework for Analysis and Prediction* - FAP) koja će integrisati glavne tehnike i metode potrebne za analizu i *mining* vremenskih serija i koja je korišćena za realizaciju ovih eksperimenata (poglavlje 6).

Svi eksperimenti u okviru ove disertacije rađeni su sa skupovima podataka UCR (University of California, Riverside) repozitorijuma vremenskih serija [50], koji obuhvata većinu svih

javno dostupnih skupova označenih vremenskih serija u svetu. Ova kolekcija se najčešće koristi za validaciju različitih koncepata *mining*-a vremenskih serija. Obuhvaćeni skupovi podataka potiču iz obilja različitih domena, uključujući medicinu, robotiku, astronomiju, biologiju, prepoznavanje lica i rukopisa, itd. Pregled osobina korišćenih skupova dat je u odeljku 4.2.

Sva ispitivanja u okviru ove disertacije izvedena su oslanjajući se isključivo na FAP biblioteku (videti pogavlja 4 i 6) razvijenu na Departmanu za matematiku i informatiku Prirodno-matematičkog fakulteta u Novom Sadu. Za potrebe naših istraživanja razvili smo i dva posebna programa: SCVGUI za proveru tačnosti implementacije različitih mera sličnosti u okviru FAP biblioteke (odeljak 4.3) odnosno DMGUI za generisanje matrica rastojanja i susedstva radi ubrzavanja eksperimenata (odeljak 4.4). Oba ova programa podržavaju prekidanje započetih izračunavanja i njihovo nastavljanje od tačke prekida. Ova osobina je direkto podržana i od strane FAP biblioteke. Serijalizovani Java objekti koji opisuju računanja mogu se preneti i na druge računare na kojima se njihovo izvršavanje može biti nastavljeno. Mogućnost serijalizacije Java objekata iskoristili smo i za razvoj posebnog agentskog sistema za distribuirano generisanje matrice rastojanja [75, 76].

Na osnovu rezultata niza opsežnih eksperimenata prikazanih i tumačenih u poglavlju 5, ispitali smo mogućnosti poboljšanja tačnosti klasifikacije 1NN i kNN klasifikatora oslanjajući se na ograničavanje prozora iskrivljenja elastičnih mera sličnosti koristeći Sakoe-Chiba pojas, odnosno na dodeljivanje različitih težina najbližim susedima. Pored toga, koristeći ove tehnike proverili smo i stav da je tačnost klasifikacije jednostavanog 1NN klasifikatora teško nadmašiti [130]. Pseudokodovi korišćenih algoritama prikazani su i opisani u odeljku 5.1.

U odeljku 5.2 ispitali smo uticaj Sakoe-Chiba globalnog ograničenja na performanse dve najreprezentativnije elastične mere sličnosti vremenskih serija: DTW i LCS. Da bismo odredili u kojoj meri ubrzava primena globalnih ograničenja izračuvanje sličnosti između vremenskih serija, izmerili smo vreme potrebno za generisanje matrica rastojanja za veći broj skupova podataka - uz različitih vrednosti parametra r ograničenja: 100% (neograničena mera), 75%, 50%, 25%, 20%, 15%, 5%, 1% i 0%. Ove vrednosti označavaju širine Sako-Chiba pojasa izražene u procentima u odnosu na dužine vremenskih serija. Ovakva distribucija je izabrana na osnovu očekivanja da se mere sa većim pojasom ograničenja ponašaju slično kao i neograničene mere, dok manje vrednosti parametra r daju interesantnija odstupanja [98, 130].

Matrica rastojanja nekog skupa podataka je matrica u kojoj elemenat na poziciji (i, j) predstavlja rastojanje između i-te i j-te vremenske serije tog skupa. Izračunavanje matrice rastojanja je dugotrajan proces što ga čini pogodnim za merenje efikasnosti globalnih ograničenja. Preko opsežnih eksperimenata opisanih u odeljku 5.2 pokazali smo da upotreba globalnih ograničenja može značajno smanjiti vreme izračunavanja mera sličnosti. Na osnovu dobijenih rezultata možemo zaključiti da je razlika u dužini trajanja računanja između neograničenih i ograničenih verzija mera sličnosti (sa malim vrednostima parametra r) veličine reda dva a negde i tri. U okviru budućih istraživanja bilo bi interesantno ispitati uticaj Sakoe-Chiba ograničenja i na druge elastične mere sličnosti. Druge moguće teme budućih radova uključuju proširenje ovih ispitivanja i na Itakura paralelogramu.

U literaturi ([98, 130]) je sugerisano da, u slučaju DTW-a, upotreba globalnih ograničenja može poboljšati tačnost klasifikacije u odnosu na neograničenu meru sličnosti. Da bismo bolje razumeli uticaj Sakoe-Chiba ograničenja, u odeljku 5.3 istražili smo efikasnost i ponašanje 1NN klasifikatora pod različitim vrednostima parametra r i ispitali smo njegovu tačnost. Naši eksperimenti su pored DTW-a obuhvatili i ostale tri elastične mere sličnosti (LCS, ERP i EDR). U prvoj fazi eksperimenata (odeljak 5.3.1) analizirali smo promene u grafu susedstva u odnosu na smanjivanje veličine parametra r. U drugoj fazi eksperimenata (odeljak 5.3.2) istražili smo kako ove promene utiču na 1NN klasifikator u pogledu na oznake (klase) najbližih suseda. Ispitivanja smo zaokružili u trećoj fazi razmatranjem uticaja globalnih ograničenja na tačnost klasifikacije (odeljak 5.3.3).

Na osnovu opsežnih ispitivanja opisanih u odeljku 5.3 možemo zaključiti da se ograničene mere kvalitativno razlikuju od neograničenih. Iz dobijenih rezultata jasno možemo videti da za male vrednosti parametra r (manjih od 15%-10%) promene u 1NN grafu postaju značajne za sve razmatrane mere sličnosti. Pored toga, uočili smo i to da postoje značajne razlike u efektima ograničenja među različitim merama sličnosti. Posmatrajući 1NN graf utvrdili smo da je na primenu globalnih ograničenja najosetljivija DTW a najmanje osetljiva EDR. Promene u slučaju ERP i LCS mera su negde između prethodna dva granična slučaja. Posmatrajući najveće vrednosti parametra r potrebnih da najmanje 10% vremenskih serija promeni svoj najbliži sused (u odnosu na neograničene verzije) možemo zaključiti da promene ovog obima najranije se javljaju u slučaju DTW-a (oko $r = 10\%$), zatim u slučaju ERP-a (oko $r = 6\%$) i LCS-a (oko $r = 5\%$) i na kraju u slučaju EDR-a (oko $r = 3\%$). Uočene razlike potvrđene su i rezultatima uparenim Vilkoksonovim testom rangova sa znakom (eng. *pairwise Wilcoxon sign-rank test*).

Pošto rezultati klasifikacije u slučaju 1NN klasifikatora u potpunosti zavise od klase najbližeg suseda, promene u grafu susedstva direktno utiču na proces klasifikacije. U drugoj fazi eksperimenata istražili smo u kojoj meri menjaju najbliži susedi svoje klase pod uticajem Sakoe-Chiba ograničenja. Dobijeni rezultati potvrdili su saznanja do kojih smo došli u prvom koraku ispitivanja: najveće promene smo zabeležili za DTW, najmanje za EDR a LCS i ERP se nalaze između ove dve mere. Pored toga, analizirajući procenat onih vremenskih serija čiji su najbliži susedi promenili svoje klase (posmatrajući samo one serije koje su promenile svoje najbliže susede) u oblasti $r < 5\%$ utvrdili smo i to da između posmatranih elastičnih mera sličnosti postoji uočljiva razlika: rezultati uparenog Vilkoksonovog testa rangova sa znakom su različita za svaku ispitivanu meru (videti odeljak 5.3.2).

Upoređivanjem tačnosti klasifikacije došli smo do zaključka da DTW (u opštem slučaju) pokazuje blagu prednost u odnosu na ostale mere rastojanja (naročito u odnosu na ERP), ali je i najosetljivija na izbor vrednosti parametra r. Statistički testovi su donekle potvrdili da se DTW može smatrati kao generalno najbolja mera rastojanja, ali dokazi nisu naročito jaki. Za svaku meru sličnosti možemo naći barem nekoliko skupova podataka za koje je data mera superiornija u odnosu na ostale. Zbog toga, izbor najbolje mere rastojanja može varirati od problema do problema, bez obzira na ovaj opšti rezultat. Posmatrajući prosečne greške klasifikacije za različite vrednosti parametra r, nalazimo uočljiv rast za male vrednosti parametra r ($r < 6\%$). Maksimalne vrednosti se kod sve četiri mere dostižu za $r = 0\%$. Najveće povećanje je primećeno u slučaju LCS-a, a najmanje kod DTW-a. U intervalu od $r = 100\%$ do $r = 6\%$, razmatrane mere sličnosti se ponašaju različito: u slučaju DTW-a,

prosečna greška klasifikacije skoro monotono opada a kod ERP-a skoro monotono raste. U slučaju LCS-a i EDR-a možemo uočiti blagu tendenciju rasta, ali promene nisu velike. To sugeriše da, iako možda u opštem slučaju DTW izgleda kao najbolji izbor, ipak LCS i EDR mogu predstavljati sigurnije izbore zbog manje izražene potrebe za pažljivo podešavanje parametra r.

Nalazi naših istraživanja jasno su pokazali da sve glavne eleastične mere sličnosti (DTW, LCS, ERP i EDR) značajno menjaju svoje ponašanje za male vrednosti globalnog ograničenja. Očekujemo da će naši rezultati pomoći istraživačima i praktičarima u odabiru i podešavanju odgovarajuće mere sličnosti u skladu sa njihovim zadacima, što će ubrzati i olakšati i izbor i proces podešavanja, i obezbediti i tačnije rezultate. Pored toga, uvid u ponašanje mera sličnosti u odnosu na menjanje ograničenja može biti od koristi i za kreiranje efikasnih strategija indeksiranja radi nalaženja (približno) najbližih suseda. U okviru budućih istraživanja planiramo da proširimo eksperimente i na druge mere sličnosti, a takođe bi bilo interesantno uporediti uticaj Itakura paralelograme sa Sakoe-Chiba ograničenjem.

U odeljku 5.4 proširili smo naša ispitivanja i na kNN klasifikator (sa vrednostima parametra k u rasponu od 1 do 30). Da bismo dobili dublji uvid u uticaj ograničenja pojasa iskrivljenja, naši eksperimenti obuhvatili su pet različitih metoda evaluacije tačnosti klasifikacije (one su opisane u pogavlju 3.2): *leave-one-out* (LOO), ukrštena validacija sa slojevitom podelom na 9 podskupova (*stratified 9-fold cross-validation* - SCV1x9), 5 puta ponovljena ukrštena validacija sa slojevitom podelom na 2 podskupa (*5 times repeated stratified 2-fold cross-validation* - SCV5x2), 10 puta ponovljena ukrštena validacija sa slojevitom podelom na 10 podskupova (*10 times repeated stratified 10-fold cross validation* - SCV10x10) i 10 puta ponovljena slojevita *holdout* metoda (SHO10x). Pored toga, analizirali smo kNN klasifikator bez težina i sa primenom težina (težine su računate po formuli (3.5)): posmatrali smo uticaj parametra k na tačnost klasifikacije nalaženjem najmanjeg Sakoe-Chiba pojasa koji će dati najmanju grešku za kNN.

Rezultati eksperimenata odeljka 5.4 jasno su potvrdili poseban značaj prvog suseda kada je reč o vremenskim serijama. Bez primena težina najbolje rezultate dobili smo sa $k = 1$. U slučaju obe mere sličnosti (DTW i LCS), greška klasifikacije kNN klasifikatora bez težina raste skoro linearno kako povećavamo vrednost parametra k. S druge strane, sa uvođenjem težina situacija se menja u izvesnoj meri. Najbolje rezultate dobili smo za vrednosti oko $k = 4$. Generalno, možemo zaključiti da težinska šema (koja daje prednost najbližem susedu) značajno poboljšava tačnost klasifikacije za sve posmatrane vrednosti parametra k.

Rezultati su ukazali i na to da primena težina značajno utiče na parametar ograničenja r. U slučaju kNN klasifikatora bez težina, vrednost parametra r raste zajedno sa povećavanjem parametra k. S druge strane, primenom težinske šeme vrednost parametra r ostaje približno ista za sve vrednosti parametra k. Pored toga, razlika imeđu najmanjih i najvećih vrednosti parametra r je oko dva puta manja u odnosu na kNN klasifikator bez težina.

Sva ova zapažanja upućuju na to da primena težinske šeme sa favorizovanjem najbližeg suseda može poboljšati kvalitet i stabilnost kNN klasifikatora. Kada je reč o vremenskim serijama, najbliži sused ima posebno značenje. Uzimajući tu činjenicu u obzir možemo značajno poboljšati kvalitet kNN klasifikatora za sve vrednosti parametra k - za neke male

vrednosti može dati tačnije rezultate od 1NN. U budućim studijama bilo bi interesantno ispitati i uticaj drugih težinskih šema [22, 35, 36, 69] odnosno proširiti ispitivanja i na druge mere sličnosti, uključujući i verzije sa Itakura ograničenjem.

U poslednjoj deceniji klasifikacija je intenzivno ispitivana u oblasti *mining-a* vremenskih serija [34, 45, 93, 130, 131]. Između značajnog broja predloženih tehnika, metoda najbližeg suseda i dinamičko iskrivljenje vremena (DTW) pokazali su se kao jedna od najboljih kombinacija [130]. U pokušaju da se poboljša tačnost klasifikacije 1NN klasifikatora, prilikom izbora klase, kNN klasifikator uzima u obzir ne jedan, već k najbližih suseda. Sledeći korak u istraživanju mogućnosti poboljšanja jednostavne metode najbližih suseda jeste dodeljivanje različitih težina susedima. Kroz detaljnu analizu u okviru odeljka 5.5 uporedili smo širok spektar različitih težinskih funkcija u kombinaciji sa tri najčešće korišćenih mera sličnosti na osnovu najvećeg repozitorijuma besplatno dostupnih skupova označenih vremenskih serija [50].

U ovim eksperimentima posmatrali smo Euklidsko rastojanje i dve najreprezentativnije elastične mere sličnosti vremenskih serija (DTW i LCS) u neograničenom obliku. Tačnost klasifikacije računali smo pomoću 10 puta ponovljene ukrštene validacije sa slojevitom podelom na 10 podskupova (SCV10x10) koristeći najbolju vrednost parametra k iz oblasti od 1 do 30 dobijenu pomoću ukrštene validacije sa slojevitom podelom na 9 podskupova (SCV1x9) računate na skupu obuke.

Posmatrajući prosečnu tačnost klasifikacije, u slučaju svih razmatranih mera sličnosti, najbolji rezultati postignuti su pomoću funkcije težina definisane formulom (3.11) u radu [35]. Najlošiji rezultati dobijeni su sa metodom najbližeg suseda (1NN klasifikator) - osim u slučaju Euklidskog rastojanja. Važno je napomenuti da razlike između najboljih i najlošijih rezultata nisu naročito velike (< 0.01). Rezultati naših detaljnih ispitivanja potvrdili su mišljenje da je jednostavnu metodu najbližih suseda teško nadmašiti [130]. Posmatrajući broj statističko značajnih pobeda i poraza, u slučaju Euklidskog rastojanja najbolji rezultat postignut je pomoću funkcije težina definisane od strane Dudani-ja [22] (jednačina (3.2)), a u slučaju DTW i LCS mera pomoću težinske šeme definisane jednačinom (3.11) u radu [35]. Rezultati oba statistička testa (*corrected resampled t-test* i *Wilcoxon sing-rank test*) podržali su Dudani i DualD šeme težina kao najbolje izbore u kombinaciji sa svim analiziranim merama sličnosti. DualU funkcija težina definisana jednačinom (3.9) predstavlja treći najbolji izbor.

Pošto elastične mere (DTW, LCS, ERP i EDR) generalno daju precizniju tačnost klasifikacije u odnosu na ne-elastične mere, bilo bi interesantno proveriti uticaj različitih šema računanja težina i na ERP odnosno EDR (pored DTW i LCS) kao i na ograničene verzije svih ovih mera sličnosti. Glavni nedostatak ovih mera je brzina izračunavanja, pošto se zasnivaju na algoritmima kvadratne složenosti. U poglavlju 5 pokazali smo da primena globalnih ograničenja značajno ubrzava proces računanja, a u nekim slučajevima čak i poboljšava tačnost klasifikacije. Pored toga, zbog velike dimenzionalnosti podataka u obliku vremenskih serija, bilo bi zanimljivo istražiti interakciju fenomena *habovitosti* (eng. *hubness*) [93] sa različitim funkcijama računanja težina, odnosno ponašanje težinskih šema zasnovanih na *habovima* (eng. *hub*) [92].

Analiza i *mining* vremenskih serija predstavljali su veoma popularne oblasti istraživanja u protekloj deceniji. To je dovelo do pojave velikih količina predloženih tehnika i algoritama. Većina tih tehnika i algoritama predstavljene su sporadično a ponekad nisu pravilno upoređeni sa drugim, konkurentnim rešenjima iz datih oblasti. To je posledica nedostatka kvalitetnog sistema otvorenog koda koji podržava različite aspekte *mining-a* vremenskih serija. Motivisani svim ovim razlozima, razvili smo jednu univerzalnu biblioteku (*Framework for Analysis and Prediction* - FAP) u koju će biti uključeni svi glavni koncepti kao što su mere sličnosti, klasifikatori, reprezentacije i tehnike pretprocesiranja vremenskih serija. Takva biblioteka može u velikoj meri pomoći istraživačima u testiranju i upoređivanju novih koncepata sa već postojećima.

Postojeće softverske sisteme za analizu vremenskih serija možemo grupisati u dve velike kategorije: opšti sistemi *mining*-a podataka i mašinskog učenja (kao što su, na primer, *WEKA* i *RapidMiner Studio*) i statistički sistemi koji podržavaju statističke i ekonometričke modele vremenskih serija (na primer, *SAS*, *MATLAB* i *R*). Pored ove dve velike grupe aplikacija postoje i usko specijalizovani programi kao što su, na primer, *Spiral* i *VizzTree* koji se koriste za vizualizaciju vremenskih serija. FAP biblioteka je zamišljena kao besplatan, proširiv softverski paket otvorenog koda specijalizovan za potrebe analize i *mining*-a vremenskih serija. Razvija se u programskom jeziku Java što obezbeđuje platformsku nezavisnost, lakše održavanje i nadogradnju.

U trenutnom stanju razvoja implementirane su sve najvažnije mere sličnosti, nekoliko algoritama za podešavanje parametara mera sličnosti, tehnika klasifikacije, metoda evaluacije klasifikatora, reprezentacija vremenskih serija i tehnika pretprocesiranja. Implementirane mere sličnosti uključuju: L_p, Swale, neograničene i ograničene verzije elastičnih mera (DTW, LCS, ERP i EDR). Ograničene mere su implementirane pomoću Sakoe-Chiba i Itakura ograničenja. FAP sistem sadrži implementaciju 1NN i kNN klasifikatora (uključujući i sve težinske šeme opisane u odeljku 3.1 i korišćene u eksperimentima poglavlja 5) zajedno sa nekoliko metoda evaluacije tačnosti klasifikatora: *Holdout*, *Cross-Validation* i *Leave-One-Out*. Od reprezentacija biblioteka sadrži *Piecewise Linear Approximation* (PLA) [51], *Piecewise Aggregate Approximation* (PAA) [47], *Adaptive Piecewise Constant Approximation* (APCA) [48], *Symbolic Aggregate Approximation* (SAX) [67] i Spline [57], a od tehnika pretprocesiranja skaliranje (eng. *scaling*), pomeranje (eng. *shifting*), *min-max* i *z-score* normalizaciju, decimalno skaliranje (eng. *decimal scaling*) i *linear equiscaling*.

Važna osobina FAP biblioteke je postojanje mehanizama za praćenje, prekidanje i nastavljanje izvršavanja vremenski zahtevnih operacija kao što su klasifikacija, evaluacija klasifikatora, podešavanje parametara i druge (opisani su u odeljku 6.2.2). Ova rešenja zajedno sa serijalizacijom Java objekata omogućavaju čuvanje delimičnih rezultata i nastavljanje eksperimenata od tačke prekida.

Detalji strukture biblioteke i implementacije opisani su u poglavlju 6 a jedan primer korišćenja u odeljku 6.8. Sva ispitivanja u okviru ove disertacije realizovana su oslanjajući se isključivo na FAP biblioteku. Pored toga ona je već uspešno primenjena u okviru drugih istraživanja različitih naučnih domena, uključujući: razvoj distribuiranih generatora matrica rastojanja zasnovanih na agentima [75, 76], mining vremenskih serija u domenu psihologije [55, 56] i analiza vremenskih serija u domenu neurologije [61].

Ubeđeni smo da će se zbog svojih brojnih prednosti FAP moći uspešno koristiti i u okviru budućih istraživanja: svi važni koncepti koji su potrebni za mining vremenskih serija integrisani su u jednu celinu, modifikovanje postojećih i dodavanje novih rešenja može se lako realizovati (FAP je pisan u Javi). Pored toga, za zajednicu istraživača i praktičara može biti od važnosti i to da je FAP biblioteka besplatna i otvorenog koda, što će omogućiti da je svi zainteresovani proširuju i dopunjuju sa novim rešenjima odnosno da je prilagođavaju svojim potrebama. Sve ove osobine mogu doprineti da bude uvek ažurirana i da sve glavne tehnike *mining-a* vremenskih serija budu podržane.

Doprinosi i rezultati ostvareni i prikazani u ovoj disertaciji su višestruki:

1. Objasnili smo uticaj Sakoe-Chiba pojasa na performanse ograničenih elastičnih mera sličnosti: za male veličine ograničenja, razlika u dužini trajanja računanja između neograničenih i ograničenih verzija mera sličnosti je veličine reda dva a negde i tri.

2. Analiziranjem grafa susedstva u odnosu na promenu veličine ograničenja pokazali smo da za male vrednosti ograničenja (manje od 15%–10%) ograničene mere postaju značajno drugačiji od neograničenih. Pored toga, pokazali smo da ove promene nisu iste kod različitih mera sličnosti - DTW je najosetljivija na primenu globalnih ograničenja, a EDR je najmanje osetljiva.

3. Kroz niz iscrpnih eksperimenata pokazali smo da se, u proseku, najbolja tačnost klasifikacije postiže za male vrednosti parametra r. Ova vrednost je najmanja za DTW (oko 4% od dužine vremenskih serija) a najveća za ERP (skoro 10%). Promene u grafu susedstva generisane sa ovim vrednostima parametra r su najveće u slučaju DTW-a (promenjen je u proseku oko 10% čvorova) a najmanji u slučaju EDR-a (u proseku, oko 1% čvorova). Upoređivanje tačnosti klasifikacije 1NN klasifikatora pokazalo se da DTW generalno ima blagu prednost u odnosu na ostale mere sličnosti, ali dokazi nisu posebno jaki - najbolji izbor zavisi od konkretnog problema.

4. Upoređujući prosečne greške klasifikacije razmatranih elastičnih mera sličnosti (DTW, LCS, ERP i EDR) istakli smo njihovu zajedničku osobinu: prosečna greška klasifikacije raste za male vrednosti parametra r ($< 6\%$) i dostiže svoj maksimum za $r = 0\%$. Videli smo da, ioako se u opštem slučaju DTW može smatrati najboljim izborom, LCS i EDR mogu predstavljati sigurnije izbore zbog manje izražene potrebe za pažljivo podešavanje parametra r.

5. Detaljna analiza kNN klasifikatora (za DTW i LCS) je pokazala da se najbolji rezultati, bez korišćenja težina, dobijaju za $k = 1$. S druge strane, u slučaju težinske verzije kNN klasifikatora, najbolji rezultati se javljaju za vrednosti blizu $k = 4$.

6. Posmatrajući prosečnu tačnost klasifikacije otkrili smo da metoda k najbližih suseda (kNN klasifikator) - i verzija bez težina, i verzije sa težinama - daje bolje rezultate od 1NN klasifikatora u slučaju značajnog broja skupova. Rezultati

statističkih testova podržavaju Dudani i DualD šeme računanja težina kao najbolje izbore.

7. Za podršku ovih i budućih istraživanja razvili smo besplatnu biblioteku otvorenog koda (FAP) koja implementira mnoge od najvažnijih algoritama u oblasti *mining*-a i analize vremenskih serija.

U okviru budućih istraživanja bilo bi zanimljivo proširite sva ova ispitivanja i na druge često korišćene mere sličnosti, uključujući ERP, EDR i TWED. Pored toga, bilo bi poželjno i uporediti uticaj Itakura paralelograma sa uticajem Sakoe-Chiba pojasa.

References

[1] Abonyi, J. 2006. *Adatbányászat a hatékonyság eszköze*. ComputerBooks, Budapest.

[2] Abul, O., Bonchi, F. and Nanni, M. 2010. Anonymization of moving objects databases by clustering and perturbation. *Information Systems*. 35, 8 (Dec. 2010), 884–910.

[3] Achtert, E., Bernecker, T., Kriegel, H.-P., Schubert, E. and Zimek, A. 2009. ELKI in Time: ELKI 0.2 for the Performance Evaluation of Distance Measures for Time Series. *Advances in Spatial and Temporal Databases SE - 35*. N. Mamoulis, T. Seidl, T. Pedersen, K. Torp, and I. Assent, eds. Springer Berlin Heidelberg. 436–440.

[4] Agrawal, R., Faloutsos, C. and Swami, A. 1993. Efficient similarity search in sequence databases. *Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms*. Springer Berlin Heidelberg. 69–84.

[5] Alpaydin, E. 2004. *Introduction to Machine Learning*. MIT Press.

[6] Aßfalg, J. 2008. *Advanced Analysis on Temporal Data*. LMU München.

[7] Bache, K. and Lichman, M. 2013. UCI Machine Learning Repository.

[8] Baiocchi, G. and Distaso, W. 2003. GRETL: Econometric software for the GNU generation. *Journal of Applied Econometrics*. 18, 1 (2003), 105–110.

[9] Bala, A., Kumar, A. and Birla, N. 2010. Voice command recognition system based on MFCC and DTW. *International Journal of Engineering Science and Technology*. 2, 12 (2010), 7335–7342.

[10] Berndt, D. and Clifford, J. 1994. Using dynamic time warping to find patterns in time series. *KDD workshop* (1994), 359–370.

[11] Bouckaert, R. and Frank, E. 2004. Evaluating the Replicability of Significance Tests for Comparing Learning Algorithms. *Advances in Knowledge Discovery and Data Mining SE - 3*. H. Dai, R. Srikant, and C. Zhang, eds. Springer Berlin Heidelberg. 3–12.

[12] Box, G.E.P., Jenkins, G.M. and Reinsel, G.C. 2008. *Time Series Analysis: Forecasting and Control*. Wiley.

[13] Brockwell, P.J. and Davis, R.A. 2002. *Introduction to Time Series and Forecasting*. Springer.

[14] Chan, K.-P. and Fu, A.W.-C. 1999. Efficient time series matching by wavelets. *Data Engineering, 1999. Proceedings., 15th International Conference on* (1999), 126–133.

[15] Chatfield, C., Koehler, A.B., Ord, J.K. and Snyder, R.D. 2001. A New Look at Models For Exponential Smoothing. *Journal of the Royal Statistical Society: Series D (The Statistician)*. 50, 2 (2001), 147–159.

[16] Chen, L. and Ng, R. 2004. On The Marriage of Lp-norms and Edit Distance. *Proceedings of the Thirtieth international conference on Very large data bases* (2004), 792–803.

[17] Chen, L., Özsu, M.T. and Oria, V. 2005. Robust and fast similarity search for moving object trajectories. *Proceedings of the 2005 ACM SIGMOD international conference on Management of data* (New York, NY, USA, 2005), 491–502.

[18] Cover, T. and Hart, P. 1967. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*. 13, 1 (1967), 21–27.

[19] Das, G. and Gunopulos, D. 2003. Time Series Similarity and Indexing. *The Handbook of Data Mining*. N. Ye, ed. Lawrence Erlbaum Associates. 279–304.

[20] Deza, M.M. and Deza, E. 2012. *Encyclopedia of Distances*. Springer.

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

[22] Dudani, S.A. 1976. The Distance-Weighted k-Nearest-Neighbor Rule. *Systems, Man and Cybernetics, IEEE Transactions on*. 6, 4 (1976), 325–327.

[23] Eads, D.R., Hill, D., Davis, S., Perkins, S.J., Ma, J., Porter, R.B. and Theiler, J.P. 2002. Genetic Algorithms and Support Vector Machines for Time Series Classification. *Proc. SPIE*.

[24] Eruhimov, V., Martyanov, V. and Tuv, E. 2007. Constructing High Dimensional Feature Space for Time Series Classification. *Knowledge Discovery in Databases: PKDD 2007, 11th European Conference on Principles and Practice of Knowledge Discovery in Databases* (2007), 414–421.

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

[26] Faloutsos, C., Ranganathan, M. and Manolopoulos, Y. 1994. Fast subsequence matching in time-series databases. *ACM SIGMOD Record*. 23, 2 (Jun. 1994), 419–429.

[27] Findley, D.F., Monsell, B.C., Bell, W.R., Otto, M.C. and Chen, B.-C. 1998. New Capabilities and Methods of the X-12-ARIMA Seasonal-Adjustment Program. *Journal of Business & Economic Statistics*. 16, 2 (1998), 127–152.

[28] Fix, E. and Hodges, J.L. 1951. Discriminatory Analysis - Nonparametric Discrimination: Consistency Properties. USAF School of Aviation Medicine.

[29] García, S., Fernández, A., Luengo, J. and Herrera, F. 2009. A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability. *Soft Comput.* 13, 10 (Apr. 2009), 959–977.

[30] Geurts, P. and Wehenkel, L. 2005. Segment and combine approach for non-parametric time-series classification. *Proceedings of the 9th European conference on Principles and Practice of Knowledge Discovery in Databases* (Berlin, Heidelberg, 2005), 478–485.

[31] Giusti, R. and Batista, G.E.A.. 2013. An Empirical Comparison of Dissimilarity Measures for Time Series Classification. *Intelligent Systems (BRACIS), 2013 Brazilian Conference on* (Oct. 2013), 82–88.

[32] Goldin, D.Q. and Kanellakis, P.C. 1995. On similarity queries for time-series data: Constraint specification and implementation. *Principles and Practice of Constraint Programming — CP '95 SE - 9*. U. Montanari and F. Rossi, eds. Springer Berlin Heidelberg. 137–153.

[33] Gopalkrishnan, V. 2008. Querying time-series streams. *Proceedings of the 11th international conference on Extending database technology: Advances in database technology* (New York, NY, USA, 2008), 547–558.

[34] Górecki, T. and Luczak, M. 2013. Using derivatives in time series classification. *Data Mining and Knowledge Discovery*. 26, 2 (Mar. 2013), 310–331.

[35] Gou, J., Du, L., Zhang, Y. and Xiong, T. 2012. A New distance-weighted k-nearest neighbor classifier. *Journal of Information & Computational Science*. 9, 6 (2012), 1429–1436.

[36] Gou, J., Xiong, T. and Kuang, Y. 2011. A Novel Weighted Voting for K-Nearest Neighbor Rule. *Journal of Computers*. 6, 5 (2011), 833–840.

[37] Grabockal, J., Bedallig, E. and Schmidt-Thieme, L. 2013. Efficient Classification of Long Time-Series. *ICT Innovations 2012: Secure and Intelligent Systems*. 207, (2013), 47–57.

[38] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. and Witten, I.H. 2009. The WEKA data mining software: an update. *SIGKDD Explor. Newsl.* 11, 1 (Nov. 2009), 10–18.

[39] Han, J., Kamber, M. and Pei, J. 2011. *Data Mining: Concepts and Techniques*. Elsevier Science.

[40] Hand, D.J., Mannila, H. and Smyth, P. 2001. *Principles of Data Mining*. MIT Press.

[41] Ho, S.-S., Tang, W. and Liu, W.T. 2010. Tropical Cyclone Event Sequence Similarity Search via Dimensionality Reduction and Metric Learning. *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2010), 135–144.

[42] Hochheiser, H. and Shneiderman, B. 2001. Interactive Exploration of Time Series Data. *Proceedings of the 4th International Conference on Discovery Science* (London, UK, UK, 2001), 441–446.

[43] Huan, N.-J. and Palaniappan, R. 2004. Neural network classification of autoregressive features from electroencephalogram signals for brain–computer interface design. *Journal of Neural Engineering*. 1, 3 (2004), 142–150.

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

[45] Jeong, Y.-S., Jeong, M.K. and Omitaomu, O.A. 2011. Weighted dynamic time warping for time series classification. *Pattern Recognition*. 44, 9 (2011), 2231–2240.

[46] Jones, M.T. 2008. *Artificial Intelligence: A Systems Approach*. Infinity Science Press.

[47] Keogh, E., Chakrabarti, K., Pazzani, M. and Mehrotra, S. 2001. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and information Systems*. 3, 3 (2001), 263–286.

[48] Keogh, E., Chakrabarti, K., Pazzani, M. and Mehrotra, S. 2001. Locally adaptive dimensionality reduction for indexing large time series databases. *Proceedings of the 2001 ACM SIGMOD international conference on Management of data* (New York, NY, USA, 2001), 151–162.

[49] Keogh, E. and Ratanamahatana, C.A. 2005. Exact indexing of dynamic time warping. *Knowledge and Information Systems*. 7, 3 (May 2005), 358–386.

[50] Keogh, E., Zhu, Q., Hu, B., Y., H., Xi, X., Wei, L. and Ratanamahatana, C.A. 2011. The UCR Time Series Classification/Clustering Homepage: www.cs.ucr.edu/~eamonn/time_series_data/.

[51] Keogh, E.J., Chu, S., Hart, D. and Pazzani, M. 2004. Segmenting Time Series: A Survey and Novel Approach. *Data Mining In Time Series Databases*. M. Last, A. Kandel, and H. Bunke, eds. World Scientific Publishing Company. 1–22.

[52] Kini, V.B. and Sekhar, C.C. 2013. Large margin mixture of AR models for time series classification. *Appl. Soft Comput.* 13, 1 (2013), 361–371.

[53] Kotsiantis, S.B. 2007. Supervised Machine Learning: A Review of Classification Techniques. *Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies* (Amsterdam, The Netherlands, The Netherlands, 2007), 3–24.

[54] Kurbalija, V. 2009. *Time Series Analysis and Prediction Using Case Based Reasoning Technology*. University of Novi Sad, Serbia.

[55] Kurbalija, V., von Bernstorff, C., Burkhard, H.-D., Nachtwei, J., Ivanović, M. and Fodor, L. 2012. Time-series Mining in a Psychological Domain. *Proceedings of the Fifth Balkan Conference in Informatics* (New York, NY, USA, 2012), 58–63.

[56] Kurbalija, V., Ivanović, M., von Bernstorff, C., Nachtwei, J. and Burkhard, H.-D. 2014. Matching Observed with Empirical Reality–What you see is what you get? *Fundamenta Informaticae*. 129, 1 (2014), 133–147.

[57] Kurbalija, V., Ivanović, M. and Budimac, Z. 2009. Case-based curve behaviour prediction. *Software: Practice and Experience*. 39, 1 (2009), 81–103.

[58] Kurbalija, V., Radovanović, M., Geler, Z. and Ivanović, M. 2010. A Framework for Time-Series Analysis. *Artificial Intelligence: Methodology, Systems, and Applications SE - 5*. D. Dicheva and D. Dochev, eds. Springer Berlin Heidelberg. 42–51.

[59] Kurbalija, V., Radovanović, M., Geler, Z. and Ivanović, M. 2011. The Influence of Global Constraints on DTW and LCS Similarity Measures for Time-Series Databases. *Third International Conference on Software, Services and Semantic Technologies S3T 2011 SE - 10*. D. Dicheva, Z. Markov, and E. Stefanova, eds. Springer Berlin Heidelberg. 67–74.

[60] Kurbalija, V., Radovanović, M., Geler, Z. and Ivanović, M. 2014. The influence of global constraints on similarity measures for time-series databases. *Knowledge-Based Systems*. 56, (2014), 49–67.

[61] Kurbalija, V., Radovanović, M., Ivanović, M., Schmidt, D., Trzebiatowski, G.L. von, Burkhard, H. and Hinrichs, C. 2014. Time-Series Analysis in the Medical Domain: A Study of Tacrolimus

Administration and Influence on Kidney Graft Function. *Computers in Biology and Medicine*. 50, (2014), 19–31.

[62] Larose, D.T. 2005. *Discovering Knowledge in Data: An Introduction to Data Mining*. Wiley.

[63] Laxman, S. and Sastry, P.S. 2006. A survey of temporal data mining. *Sadhana*. 31, 2 (2006), 173–198.

[64] Le, T.-L., Boucher, A. and Thonnat, M. 2006. Subtrajectory-Based Video Indexing and Retrieval. *Advances in Multimedia Modeling SE - 41*. T.-J. Cham, J. Cai, C. Dorai, D. Rajan, T.-S. Chua, and L.-T. Chia, eds. Springer Berlin Heidelberg. 418–427.

[65] Lee, J.-G., Han, J., Li, X. and Gonzalez, H. 2008. TraClass: trajectory classification using hierarchical region-based and trajectory-based clustering. *Proc. VLDB Endow.* 1, 1 (Aug. 2008), 1081–1094.

[66] Lin, J., Keogh, E., Lonardi, S., Lankford, J.P. and Nystrom, D.M. 2004. Visually mining and monitoring massive time series. *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining* (New York, NY, USA, 2004), 460–469.

[67] Lin, J., Keogh, E., Wei, L. and Lonardi, S. 2007. Experiencing SAX: a novel symbolic representation of time series. *Data Mining and Knowledge Discovery*. 15, 2 (2007), 107–144.

[68] Lines, J. and Bagnall, A. 2014. Time series classification with ensembles of elastic distance measures. *Data Mining and Knowledge Discovery*. (2014), 1–28.

[69] Macleod, J.E.S., Luk, A. and Titterington, D.M. 1987. A Re-Examination of the Distance-Weighted k-Nearest Neighbor Classification Rule. *Systems, Man and Cybernetics, IEEE Transactions on*. 17, 4 (1987), 689–696.

[70] Mai, S.T., Goebl, S. and Plant, C. 2012. A Similarity Model and Segmentation Algorithm for White Matter Fiber Tracts. *Data Mining (ICDM), 2012 IEEE 12th International Conference on* (Dec. 2012), 1014–1019.

[71] Marteau, P.-F. 2009. Time Warp Edit Distance with Stiffness Adjustment for Time Series Matching. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*. 31, 2 (2009), 306–318.

[72] MATLAB homepage: *http://www.mathworks.com/products/matlab/*. Accessed: 2015-03-08.

[73] Megalooikonomou, V., Wang, Q., Li, G. and Faloutsos, C. 2005. A multiresolution symbolic representation of time series. *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on* (2005), 668–679.

[74] Mitchell, T.M. 1997. *Machine Learning*. McGraw-Hill, Inc.

[75] Mitrović, D., Geler, Z. and Ivanović, M. 2012. Distributed Distance Matrix Generator Based on Agents. *Proceedings of the 2Nd International Conference on Web Intelligence, Mining and Semantics* (New York, NY, USA, 2012), 40:1–40:6.

[76] Mitrovic, D., Ivanović, M. and Geler, Z. 2014. Agent-Based Distributed Computing for Dynamic Networks. *Information Technology And Control*. 43, 1 (2014), 88–97.

[77] Mitsa, T. 2010. *Temporal Data Mining*. Taylor & Francis.

[78] Mohri, M., Rostamizadeh, A. and Talwalkar, A. 2012. *Foundations of Machine Learning*. The MIT Press.

[79] Morse, M.D. and Patel, J.M. 2007. An efficient and accurate method for evaluating time series similarity. *Proceedings of the 2007 ACM SIGMOD international conference on Management of data* (New York, NY, USA, 2007), 569–580.

[80] Nadeau, C. and Bengio, Y. 2003. Inference for the Generalization Error. *Mach. Learn.* 52, 3 (Sep. 2003), 239–281.

[81] Nanopoulos, A., Alcock, R. and Manolopoulos, Y. 2001. Feature-based Classification of Time-series Data. *International Journal of Computer Research*. 10, (2001), 49–61.

[82] Olszewski, R.T. 2001. *Generalized Feature Extraction for Structural Pattern Recognition in Time-series Data*. Carnegie Mellon University.

[83] Pao, T.-L., Chen, Y.-T., Yeh, J.-H. and Chang, Y.-H. 2005. Emotion Recognition and Evaluation of Mandarin Speech Using Weighted D-KNN Classification. *Proceedings of the 17th Conference on Computational Linguistics and Speech Processing, ROCLING* (2005).

[84] Pao, T.-L., Chen, Y.-T., Yeh, J.-H., Cheng, Y.-M. and Lin, Y.-Y. 2007. A Comparative Study of Different Weighting Schemes on KNN-Based Emotion Recognition in Mandarin Speech. *Advanced Intelligent Computing Theories and Applications. With Aspects of Theoretical and Methodological Issues SE - 101*. D.-S. Huang, L. Heutte, and M. Loog, eds. Springer Berlin Heidelberg. 997–1005.

[85] Pavlovic, V., Frey, B.J. and Huang, T.S. 1999. Time-series classification using mixed-state dynamic Bayesian networks. *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.* (1999), 609–615.

[86] Pelekis, N., Kopanakis, I., Kotsifakos, E., Frentzos, E. and Theodoridis, Y. 2011. Clustering uncertain trajectories. *Knowledge and Information Systems*. 28, 1 (2011), 117–147.

[87] Petitjean, F., Inglada, J. and Gancarski, P. 2012. Satellite Image Time Series Analysis Under Time Warping. *Geoscience and Remote Sensing, IEEE Transactions on*. 50, 8 (Aug. 2012), 3081–3095.

[88] Petridis, V. and Kehagias, A. 1997. Predictive modular fuzzy systems for time-series classification. *Fuzzy Systems, IEEE Transactions on*. 5, 3 (Aug. 1997), 381–397.

[89] Pham, C.H., Le, Q.K. and Le, T.H. 2014. Human Action Recognition Using Dynamic Time Warping and Voting Algorithm. *VNU Journal of Science: Science and Communication Engineering*. 30, 3 (2014), 22–30.

[90] R Core Team 2014. R: A Language and Environment for Statistical Computing.

[91] Radovanović, M. 2010. *High-Dimensional Data Representations and Metrics for Machine Learning and Data Mining*. University of Novi Sad, Serbia.

[92] Radovanović, M., Nanopoulos, A. and Ivanović, M. 2010. Hubs in Space: Popular Nearest Neighbors in High-Dimensional Data. *J. Mach. Learn. Res.* 11, (Dec. 2010), 2487–2531.

[93] Radovanović, M., Nanopoulos, A. and Ivanović, M. 2010. Time-series classification in many intrinsic dimensions. *Proc. 10th SIAM Int. Conf. on Data Mining (SDM)* (2010), 677–688.

[94] RapidMiner Studio homepage: *http://rapidminer.com/*. Accessed: 2015-01-20.

[95] Ratanamahatana, C., Lin, J., Gunopulos, D., Keogh, E., Vlachos, M. and Das, G. 2005. Mining Time Series Data. *Data Mining and Knowledge Discovery Handbook SE - 51*. O. Maimon and L. Rokach, eds. Springer US. 1069–1103.

[96] Ratanamahatana, C.A. and Keogh, E. 2004. Everything you know about dynamic time warping is wrong. *3rd Workshop on Mining Temporal and Sequential Data, in conjunction with 10th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD-2004), Seattle, WA* (2004).

[97] Ratanamahatana, C.A. and Keogh, E. 2004. Making Time-series Classification More Accurate Using Learned Constraints. *Proceedings of the 2004 SIAM International Conference on Data Mining*. 11–22.

[98] Ratanamahatana, C.A. and Keogh, E. 2005. Three myths about dynamic time warping data mining. *Proceedings of SIAM International Conference on Data Mining (SDM'05)* (2005), 506–510.

[99] Rath, T.M. and Manmatha, R. 2003. Word image matching using dynamic time warping. *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on* (Jun. 2003), II–521–II–527 vol.2.

[100] Ratnaparkhi, A., Reynar, J. and Roukos, S. 1994. A Maximum Entropy Model for Prepositional Phrase Attachment. *Proceedings of the Workshop on Human Language Technology* (Stroudsburg, PA, USA, 1994), 250–255.

[101] Rodríguez, J., Alonso, C. and Boström, H. 2000. Learning First Order Logic Time Series Classifiers: Rules and Boosting. *Principles of Data Mining and Knowledge Discovery SE - 29*. D. Zighed, J. Komorowski, and J. Żytkow, eds. Springer Berlin Heidelberg. 299–308.

[102] Rodríguez, J.J. and Alonso, C.J. 2004. Interval and Dynamic Time Warping-based Decision Trees. *Proceedings of the 2004 ACM Symposium on Applied Computing* (New York, NY, USA, 2004), 548–552.

[103] Ross, J.C., P., V.T. and Rao, P. 2012. Detecting Melodic Motifs from Audio for Hindustani Classical Music. *ISMIR* (2012), 193–198.

[104] Roverso, D. 2000. Multivariate Temporal Classification By Windowed Wavelet Decomposition And Recurrent Neural Networks. *In 3 rd ANS International Topical Meeting on Nuclear Plant Instrumentation, Control and Human-Machine Interface* (2000).

[105] Sakoe, H. and Chiba, S. 1978. Dynamic programming algorithm optimization for spoken word recognition. *Acoustics, Speech and Signal Processing, IEEE Transactions on*. 26, 1 (1978), 43–49.

[106] SAS homepage: *http://www.sas.com/en_us.html*. Accessed: 2015-01-20.

[107] Schneider, M.C. and Burkhard, H.-D. 2012. Creating driving behavior for artificial agents in a social augmented micro-world. *Proceedings of the 21th International Workshop on*

Concurrency, Specification and Programming, Berlin, Germany, September 26-28, 2012 (2012), 336–342.

[108] Serrà, J. and Arcos, J.L. 2014. An empirical evaluation of similarity measures for time series classification. *Knowledge-Based Systems*. 67, 0 (2014), 305–314.

[109] Shi, T., Wang, P., Wang, J.-S. and Yue, S. 2012. Application of grid-based k-means clustering algorithm for optimal image processing. *Comput. Sci. Inf. Syst.* 9, 4 (2012), 1679–1696.

[110] Skopal, T. and Bustos, B. 2011. On Nonmetric Similarity Search Problems in Complex Domains. *ACM Comput. Surv.* 43, 4 (Oct. 2011), 34:1–34:50.

[111] Spiegel, S., Jain, B.-J. and Albayrak, S. 2014. Fast Time Series Classification Under Lucky Time Warping Distance. *Proceedings of the 29th Annual ACM Symposium on Applied Computing* (New York, NY, USA, 2014), 71–78.

[112] Stojanovic, R., Knezevic, S., Karadaglic, D. and Devedzic, G. 2013. Optimization and implementation of the wavelet based algorithms for embedded biomedical signal processing. *Comput. Sci. Inf. Syst.* 10, 1 (2013), 502–523.

[113] Sumathi, S. and Sivanandam, S.N. 2006. *Introduction to Data Mining and Its Applications*. Springer.

[114] Symeonidis, A.L. and Mitkas, P.A. 2005. *Agent Intelligence Through Data Mining (Multiagent Systems, Artificial Societies, and Simulated Organizations)*. Springer-Verlag New York, Inc.

[115] Takigawa, Y., Hott, S., Kiyasu, S. and Miyahara, S. 2005. Pattern Classification Using Weighted Average Patterns of Categorical k-Nearest Neighbors. *Proc. of the 1th International Workshop on Camera-Based Document Analysis and Recognition* (2005), 111–118.

[116] Tan, P.N., Steinbach, M. and Kumar, V. 2006. *Introduction to Data Mining*. Pearson Addison Wesley.

[117] Taniar, D. 2008. *Data Mining and Knowledge Discovery Technologies*. IGI Pub.

[118] Tiesyte, D. and Jensen, C.S. 2008. Similarity-based Prediction of Travel Times for Vehicles Traveling on Known Routes. *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* (New York, NY, USA, 2008), 14:1–14:10.

[119] Tomašev, N. and Mladenić, D. 2012. Nearest Neighbor Voting in High Dimensional Data: Learning from Past Occurrences. *Computer Science and Information Systems*. 9, 2 (2012), 691–712.

[120] Turrado García, F., García Villalba, L.J. and Portela, J. 2012. Intelligent system for time series classification using support vector machines applied to supply-chain. *Expert Systems with Applications*. 39, 12 (Sep. 2012), 10590–10599.

[121] Vlachos, M., Kollios, G. and Gunopulos, D. 2002. Discovering similar multidimensional trajectories. *Proceedings 18th International Conference on Data Engineering* (2002), 673–684.

[122] Vrigkas, M., Karavasilis, V., Nikou, C. and Kakadiaris, I.A. 2014. Matching mixtures of curves for human action recognition. *Computer Vision and Image Understanding*. 119, 0 (Feb. 2014), 27–40.

[123] Weber, M., Alexa, M. and Muller, W. 2001. Visualizing time-series on spirals. *Information Visualization, 2001. INFOVIS 2001. IEEE Symposium on* (2001), 7–13.

[124] Wei, L. and Keogh, E. 2006. Semi-supervised Time Series Classification. *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2006), 748–753.

[125] Van Wijk, J.J. and Van Selow, E.R. 1999. Cluster and Calendar Based Visualization of Time Series Data. *Proceedings of the 1999 IEEE Symposium on Information Visualization* (Washington, DC, USA, 1999), 4–9.

[126] Witten, I.H., Frank, E. and Hall, M.A. 2011. *Data Mining: Practical Machine Learning Tools and Techniques*. Elsevier Science.

[127] Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G., Ng, A., Liu, B., Yu, P., Zhou, Z.-H., Steinbach, M., Hand, D. and Steinberg, D. 2008. Top 10 algorithms in data mining. *Knowledge and Information Systems*. 14, 1 (2008), 1–37.

[128] Wu, Y. and Chang, E.Y. 2004. Distance-function Design and Fusion for Sequence Data. *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management* (New York, NY, USA, 2004), 324–333.

[129] Wu, Y.-L., Agrawal, D. and El Abbadi, A. 2000. A Comparison of DFT and DWT Based Similarity Search in Time-series Databases. *Proceedings of the Ninth International Conference on Information and Knowledge Management* (New York, NY, USA, 2000), 488–495.

[130] Xi, X., Keogh, E., Shelton, C., Wei, L. and Ratanamahatana, C.A. 2006. Fast time series classification using numerosity reduction. *Proceedings of the 23rd international conference on Machine learning* (New York, NY, USA, 2006), 1033–1040.

[131] Ye, L. and Keogh, E. 2009. Time Series Shapelets: A New Primitive for Data Mining. *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2009), 947–956.

[132] Yuan, Y. and Raubal, M. 2012. Extracting Dynamic Urban Mobility Patterns from Mobile Phone Data. *Geographic Information Science SE - 26*. N. Xiao, M.-P. Kwan, M. Goodchild, and S. Shekhar, eds. Springer Berlin Heidelberg. 354–367.

[133] Zavrel, J. 1997. An Empirical Re-Examination of Weighted Voting for k-NN. *Proceedings of the 7th Belgian-Dutch Conference on Machine Learning* (1997), 139–148.

[134] Zhang, D., Zuo, W., Zhang, D., Zhang, H. and Li, N. 2010. Classification of pulse waveforms using edit distance with real penalty. *EURASIP J. Adv. Signal Process*. 2010, (Feb. 2010), 28:1–28:8.

[135] Zhang, H., Ho, T. and Lin, M. 2004. A Non-parametric Wavelet Feature Extractor for Time Series Classification. *Advances in Knowledge Discovery and Data Mining SE - 71*. H. Dai, R. Srikant, and C. Zhang, eds. Springer Berlin Heidelberg. 595–603.

[136] Zhang, H., Ho, T.B., Zhang, Y. and Lin, M.S. 2006. Unsupervised Feature Extraction for Time Series Clustering Using Orthogonal Wavelet Transform. *Informatica (Slovenia)*. 30, 3 (2006), 305–319.

[137] Zhang, X., Wu, J., Yang, X., Ou, H. and Lv, T. 2009. A novel pattern extraction method for time series classification. *Optimization and Engineering*. 10, 2 (2009), 253–271.

[138] Zhang, Z., Cheng, J., Li, J., Bian, W. and Tao, D. 2012. Segment-Based Features for Time Series Classification. *Computer Journal*. 55, 9 (Sep. 2012), 1088–1102.

[139] Zhen, D., Wang, T., Gu, F. and Ball, A.D. 2013. Fault diagnosis of motor drives using stator current signal analysis based on dynamic time warping. *Mechanical Systems and Signal Processing*. 34, 1–2 (2013), 191–202.

List of Figures

List of Tables

Biography

Zoltan Geler was born on 21 October 1978 in Sombor. He enrolled in Veljko Petrović Gymnasium in 1993 in Sombor. He gained admission to the Faculty of Sciences in Novi Sad in 1997 to study Informatics - on the basis of the results achieved at the national and state competitions in programming - without taking the entrance exam. After defending his bachelor thesis titled “Heuristic implementation of a variation of the game Go-Moku”, for which he got a final mark of 10, he graduated from the university in 2006 with an average mark of 9.26.

He started his Master’s studies in Informatics at the Faculty of Sciences in Novi Sad in 2006. He passed the exams prescribed by the curriculum with an average mark of 10.0, and he defended his master's thesis titled “An example of using a non-relational database” in 2008 with a final mark of 10. In 2008 he started his Doctoral Studies in Informatics at the Faculty of Sciences of the University of Novi Sad. He passed all the exams with an average grade of 10.0.

From 2006 to 2008 he was hired as a postgraduate student responsible for Informatics at the Faculty of Philosophy in Novi Sad. Between 2008 and 2009 he was employed as a research fellow there. Since 2008 he has worked as an assistant lecturer on the field of Informatics at the same faculty. He is involved in conducting of exercises for students of the Faculty of Philosophy in two subjects: Computer literacy 1 and Computer literacy 2.

He is a co-author of six scientific papers in the field of mining time series and two scientific papers in the field of media studies.

Biografija

Zoltan Geler je rođen 21. oktobra 1978. godine u Somboru. Opšti smer gimnazije "Veljko Petrović" upisao je 1993. godine u Somboru. Na Prirodno-matematički Fakultet u Novom Sadu, smer Diplomirani informatičar primljen je 1997. godine - na osnovu postignutih rezultata na republičkim i saveznim takmičenjima iz programiranja - bez polaganja prijemnog ispita. Diplomirao je 2006. godine sa prosečnom ocenom 9.26 odbranivši diplomski rad pod nazivom "Heuristička implementacija jedne varijacije igre Go-moku" sa ocenom 10.

Master studije informatike na Prirodno-matematičkom fakultetu u Novom Sadu upisao je 2006. godine. Ispite predviđene planom i programom položio je sa prosečnom ocenom 10.0, a master rad pod nazivom "Primer korišćenja jedne nerelacione baze podataka" odbranio je 2008. godine sa ocenom 10. Doktorske studije informatike na Prirodno-matematičkom fakultetu u Novom Sadu upisao je 2008. godine. Položio je sve ispite sa prosečnom ocenom 10.0.

Od 2006. do 2008. godine bio je angažovan kao student postdiplomac za užu naučnu oblast Informatika na Filozofskom fakultetu u Novom Sadu. Od 2008. do 2009. godine bio je zaposlen na Filozofskom fakultetu u svojsvtu saradnika u nastavi za užu naučnu oblast Informatika. Od 2008. godine zaposlen je na istom fakultetu na poziciji asistenta za užu naučnu oblast Informatika. Uključen je u izvođenje vežbi za studente Filozofskog fakulteta iz dva predmeta: Informatička pismenost 1 i Informatička pismenost 2.

Koautor je 6 naučnih radova u oblasti *mining*-a vremenskih serija i 2 naučna rada u oblasti medijskih istraživanja.

UNIVERZITET U NOVOM SADU
PRIRODNO-MATEMATIČKI FAKULTET
KLJUČNA DOKUMENTACIJSKA INFORMACIJA

Redni broj: **RBR**	
Identifikacioni broj: **IBR**	
Tip dokumentacije: **TD**	Monografska dokumentacija
Tip zapisa: **TZ**	Tekstualni štampani materijal
Vrsta rada: **VR**	Doktorska disertacija
Autor: **AU**	Zoltan Geler
Mentor: **MN**	Mirjana Ivanović
Naslov rada: **MR**	Uloga mera sličnosti u analizi vremenskih serija
Jezik publikacije: **JP**	Engleski
Jezik izvoda: **JI**	Srpski / Engleski
Zemlja publikovanja: **ZP**	Srbija
Uže geografsko područje: **UGP**	Vojvodina
Godina: **GO**	2015
Izdavač: **IZ**	Autorski reprint
Mesto i adresa: **MA**	Novi Sad, Trg Dositeja Obradovića 4
Fizički opis rada: **FO**	(7, 171, 139, 51, 58, 0, 5) (broj poglavlja/ strana/ lit.citata/ tabela/ slika/ grafika/ priloga)
Naučna oblast: **NO**	Računarske nauke
Naučna disciplina: **ND**	Veštačka inteligencija
Predmetna odrednica / Ključne reči: **PO**	Analiza vremenskih serija, data mining, klasifikacija, mera sličnosti

UDK:
čuva se:
ČU

Važna napomena:
VN

Nema

Izvod:
IZ

Predmet istraživanja ove disertacije obuhvata detaljan pregled i analizu uticaja Sakoe-Chiba globalnog ograničenja na najčešće korišćene elastične mere sličnosti u oblasti *data mining*-a vremenskih serija sa naglaskom na tačnost klasifikacije. Izbor mere sličnosti jedan je od najvažnijih aspekata analize vremenskih serija - ona treba verno reflektovati sličnost između podataka prikazanih u obliku vremenskih serija. Mera sličnosti predstavlja kritičnu komponentu mnogih zadataka *mining*-a vremenskih serija, uključujući klasifikaciju, grupisanje (eng. *clustering*), predviđanje, otkrivanje anomalija i drugih.

Istraživanje obuhvaćeno ovom disertacijom usmereno je na nekoliko pravaca:

1. pregled efekata globalnih ograničenja na performanse računanja mera sličnosti,
2. detaljna analiza posledice ograničenja elastičnih mera sličnosti na tačnost klasifikacije klasičnih tehnika klasifikacije,
3. opsežna studija uticaj različitih načina računanja težina (eng. *weighting scheme*) na klasifikaciju vremenskih serija,
4. razvoj biblioteke otvorenog koda (*Framework for Analysis and Prediction* - FAP) koja će integrisati glavne tehnike i metode potrebne za analizu i *mining* vremenskih serija i koja je korišćena za realizaciju ovih eksperimenata.

Datum prihvatanja teme od strane NN veća:
DP

04. decembar 2014.

Datum odbrane:
DO

Članovi komisije:
KO

Predsednik:	dr Miloš Radovanović, docent, Prirodno-matematički fakultet, Novi Sad
Mentor:	dr Mirjana Ivanović, redovni profesor, Prirodno-matematički fakultet, Novi Sad
Član:	dr Zoran Budimac, redovni profesor, Prirodno-matematički fakultet, Novi Sad
Član:	dr Vladimir Kurbalija, vanredni profesor, Prirodno-matematički fakultet, Novi Sad
Član:	dr Zoran Bosnić, vanredni profesor, Fakultet za računarstvo i informatiku, Ljubljana, Slovenija

UNIVERSITY OF NOVI SAD
FACULTY OF SCIENCE
KEY WORDS DOCUMENTATION

Accession number: **ANO**	
Identification number: **INO**	
Document type: **DT**	Monograph type
Type of record: **TR**	Printed text
Contents Code: **CC**	PhD Thesis
Author: **AU**	Zoltan Geler
Mentor: **MN**	Mirjana Ivanović
Title: **TI**	Role of Similarity Measures in Time Series Analysis
Language of text: **LT**	English
Language of abstract: **LA**	Serbian / English
Country of publication: **CP**	Serbia
Locality of publication: **LP**	Vojvodina
Publication year: **PY**	2015
Publisher: **PU**	Author's reprint
Publ. place: **PP**	Novi Sad, Trg Dositeja Obradovića 4
Physical description: **PD**	(7, 171, 139, 51, 58, 0, 5) (no. of chapters/pages/bib.refs/tables/figures/graphs/appendices)
Scientific field: **SF**	Computer Science
Scientific discipline: **SD**	Artificial Intelligence
Subject / Key words: **SKW**	Time series analysis, data mining, classification, similarity measures

UC:
holding data:
HD

Note:
NO

None

Abstract:
AB

The subject of this dissertation encompasses a comprehensive overview and analysis of the impact of Sakoe-Chiba global constraint on the most commonly used elastic similarity measures in the field of time-series data mining with a focus on classification accuracy. The choice of similarity measure is one of the most significant aspects of time-series analysis - it should correctly reflect the resemblance between the data presented in the form of time series. Similarity measures represent a critical component of many tasks of mining time series, including: classification, clustering, prediction, anomaly detection, and others.

The research covered by this dissertation is oriented on several issues:

1. review of the effects of global constraints on the performance of computing similarity measures,
2. a detailed analysis of the influence of constraining the elastic similarity measures on the accuracy of classical classification techniques,
3. an extensive study of the impact of different weighting schemes on the classification of time series,
4. development of an open source library that integrates the main techniques and methods required for analysis and mining time series, and which is used for the realization of these experiments.

Accepted by the Scientific Board on:
ASB

December 04, 2014

Defended:
DE

Thesis defend board:
DB

President:	dr Miloš Radovanović, assistant professor, Faculty of Science, Novi Sad
Advisor:	dr Mirjana Ivanović, full professor, Faculty of Science, Novi Sad
Član:	dr Zoran Budimac, full professor, Faculty of Science, Novi Sad
Član:	dr Vladimir Kurbalija, associate professor, Faculty of Science, Novi Sad
Član:	dr Zoran Bosnić, associate professor, Faculty of Computer and Information Science, Ljubljana, Slovenia