A Time Series Representation Model for Accurate and Fast Similarity Detection

Francesco Gullo, Giovanni Ponti,
Andrea Tagarelli *, Sergio Greco

Dept. of Electronics, Computer and Systems Sciences, University of Calabria

Abstract

Similarity search and detection is a central problem in time series data processing and management. Most approaches to this problem have been developed around the notion of dynamic time warping, whereas several dimensionality reduction techniques have been proposed to improve the efficiency of similarity searches. Since the continuous increasing of sources of time series data and the cruciality of real-world applications that use such data, we believe there is a challenging demand for supporting similarity detection in time series in a both accurate and fast way. Our proposal is to define a concise yet feature-rich representation of time series, on which the dynamic time warping can be applied for effective and efficient similarity detection of time series. We present the *Derivative time series Segment Approximation (DSA)* representation model, which originally features derivative estimation, segmentation and segment approximation to provide both high sensitivity in capturing the main trends of time series and data compression. We extensively compare DSA with state-of-the-art similarity methods and dimensionality reduction techniques in clustering and classification frameworks. Experimental evidence from effectiveness and efficiency tests on various datasets shows that DSA is well-suited to support both accurate and fast similarity detection.

Key words: Time series data, representation models, similarity detection, dimensionality reduction, clustering, classification

* Corresponding author. Address: via P. Bucci, 41C, I87036 Arcavacata di Rende (CS), Italy. Tel.: +39 0984 494751. Fax: +39 0984 494713.

Email addresses: fgullo@deis.unical.it (Francesco Gullo), gponti@deis.unical.it (Giovanni Ponti), tagarelli@deis.unical.it (Andrea Tagarelli), greco@deis.unical.it (Sergio Greco).

1 Introduction

A *time series* is a sequence of (real) numeric values upon which a total order based on timestamps is defined. Time series are generally used to represent the temporal evolution of objects, hence enormous amounts of such data are naturally available from several sources of different domains, including speech recognition, medicine and biology measurement, financial and market data analysis, telecommunication and telemetry, sensor networking, motion tracking, meteorology, and so on.

Most research on time series data management and knowledge discovery has been devoted to the similarity search and detection problem, which arises in many tasks such as indexing and query processing, change detection, frequent pattern mining, classification, and clustering. In this work we refer to clustering and classification as evaluation frameworks for similarity detection. In particular, we focus on the clustering task as it is necessary when the data being organized are not associated with predefined categories, which is a very frequent context in real-world application domains. Clustering of time series data has been attracting a growing interest in several scenarios. For instance, in the biomedical domain, frequently posed problems include finding groups of genes with similar expression profiles across a number of experiments, organizing patients according to different healthy/disease conditions, and finding groups of similar functional activities of the human brain in response to a given stimulus. In the socio-economics domain, clustering energy/power consumption patterns can support applications of fraud detection. Other challenging scenarios involve, for instance, seasonality patterns of retail data, personal income data, models of ecological dynamics, multimedia data streams. A more exhaustive list of applications which demand for time series clustering can be found in [23].

The common way to compare two time series is "warping" the time axis in order to achieve an alignment between the data points of the series. The Dynamic Time Warping (DTW) algorithm has long been known in speech recognition [30], and shown to be an effective solution for measuring the distance between time series [1]. Indeed, unlike the Euclidean distance, DTW allows elastic shifting of a sequence to provide a better match with another sequence, thus it can handle time series with local time shifting and different lengths.

Besides the similarity problem in time series, another issue concerns the high dimensionality that characterizes time series data in many application domains. To address this issue, various dimensionality reduction techniques have been proposed, following two main approaches in which a (continuous) time series is approximated with either a piece-wise discontinuous function or a low-order continuous function.

Dimensionality reduction methods are useful for modeling time series into a more compact form. However, while this can help to compare time series efficiently, dimensionality reduction methods may lose significant information about the main trends in a time series, which are essential to effective similarity

detection. Indeed, in many real-world applications there is a growing interest to develop methods that are able to fit an emerging demand for both *accurate and fast similarity detection*. In this respect, we believe there is a number of special requirements that should be satisfied by any representation model to support accurate and fast similarity detection in time series, which are summarized as follows:

- *Time warping-aware.* Time series should be modeled into a form that can be naturally mapped to the time domain. This will make it feasible to benefit from using dynamic time warping for similarity detection.
- *Low complexity.* Since the high dimensionality of time series data, modeling time series should be performed maintaining a reasonably low complexity, which is possibly linear with the series length.
- *Sensitivity to features.* It is clearly desirable that time series approximation is able to preserve as much information in the original series as possible. For this purpose, approximating a time series should be accomplished in such a way that it tailors itself to the local features of the series, in order to capture the important trends of the series.
- *Parameter-free.* Most representation models and dimensionality reduction methods require the user to specify some input parameters, such as, e.g., the number of coefficients or symbols. However, prior domain knowledge is often unavailable, and the sensitivity to input parameters can seriously affect the accuracy of the representation model or dimensionality reduction method.

In this paper, we present a time series representation model which is conceived to support accurate and fast similarity detection. This model is called *Derivative time series Segment Approximation (DSA)*, as it achieves a concise yet feature-rich time series representation by combining the notions of *derivative estimation*, *segmentation* and *segment approximation*.

Our DSA involves a segmentation scheme that employs the paradigm based on a piecewise discontinuous function. However, in contrast to any other technique of dimensionality reduction, the segmentation step is performed on the derivative version of the original time series, rather than directly on the raw time series. The derivative estimates represent a new feature space that enables the identification of the trends of the original series. Moreover, the final segment modeling step allows for concisely fitting the detected trends in a low-dimensional, time warping-aware representation of the original time series. As we proved experimentally, the intuition underlying the DSA model works out very advantageously in supporting accurate and fast similarity detection, indeed DSA is able to fulfill all of the desiderata mentioned above:

- DSA sequences can be compared by using DTW directly;
- the derivative-based feature generation allows for representing a time series

by focusing on the trends that are characteristic of the series;
- the segmentation step in DSA has a computational complexity which is linear with the series length, and it is adaptive with respect to the identified trends of the series;
- the absence of mandatory input parameters in DSA addresses the unavailability of prior domain knowledge.

We conducted an extensive experimental evaluation of DSA within clustering and classification frameworks, by considering aspects of effectiveness as well as efficiency. This evaluation necessarily involved the prominent state-of-the-art methods for time series representation and dimensionality reduction. Experimental evidence has shown that DSA supports accurate and fast similarity detection, in terms of a number of results that are summarized in Section 5.4.

The rest of the paper is organized as follows. Section 2 discusses the state-of-the-art for similarity search/detection and dimensionality reduction, and provides a first comparison between our proposal and the competing methods. Section 3 presents our DSA model in detail. Section 4 and Section 5 describe the experimental methodology and relating results to assess DSA and the competing methods on benchmark datasets. Section 6 presents an application of DSA on a real case study. Finally, Section 7 provides concluding remarks and some pointers to future research.

2 Related work

As we mentioned in the Introduction, DTW is widely used to perform similarity search and detection in time series. Given two sequences T_1 and T_2, DTW performs a non-linear mapping of one sequence to another by minimizing the total distance between them. For doing this, a $(|T_1| \times |T_2|)$-matrix storing the squared Euclidean distances between the two sequences is used to find an optimal warping path (i.e., a sequence of matrix elements) via a dynamic programming algorithm. Moreover, a number of pruning techniques and computationally cheap lower bounds (e.g., [15,21,42]) have been proposed to make DTW able to support fast and tight indexing and query processing. However, a major weakness of DTW is that it tends to produce "singularities", i.e., alignments of a single point of a series with multiple points of another series. This phenomenon becomes undesirable when unexpected singularities are produced. An effective variant of DTW, called Derivative Dynamic Time Warping (DDTW) [19], has been proposed to reduce the phenomenon of singularities. Basically, DDTW considers new features in the sequences while maintaining a computational complexity equal to DTW. The novelty of DDTW is that local derivatives of the data points are estimated to capture information on the trends in the sequences and to find a warping more robust to singularities. For instance, two data points having identical values, one with a negative slope (i.e., part of a falling trend) and the other one with a positive slope (i.e., part of a rising trend), are correctly not mapped each other when DDTW is

used. In a sense, DDTW can be seen as DTW equipped with a preliminary preprocessing step, in which the original data points are replaced with their derivatives.
An alternative, although not computationally more convenient approach to similarity search and detection in time series is based on edit distance-like string matching measures. The Longest Common SubSequence (LCSS) algorithm [37] is a variant of the edit distance that uses the length of the longest common subsequence of two sequences to define the distance between them. LCSS can deal with noisy time series by performing approximate matching rather than exact matching of time series, although it suffers from large-grained similarity. Edit Distance with Real sequences (EDR) [8] performs the same distance quantization of LCSS (which is parametric with respect to a tolerance threshold) in order to remove noisy effects. Unlike LCSS and EDR, Edit distance with Real Penalty (ERP) [7] is a metric and still supports local time shifting. ERP can be seen as a variant of EDR and DTW, although it does not require a noise-tolerance threshold like EDR, and does not replicate previous data points to add a gap like DTW.
To address the high dimensionality issue in time series, there are mainly two basic approaches as we mentioned above: approximating a time series by a piecewise discontinuous function or applying a low-order continuous function to a time series.
The first approach includes Discrete Wavelet Transform (DWT) [6,39], Swinging Door (SD) [2], Piecewise Linear Approximation (PLA) [28,17], Piecewise Aggregate Approximation (PAA) [16,18,41], Adaptive Piecewise Constant Approximation (APCA) [5], and Symbolic Aggregate approXimation (SAX) [24]. Using DWT, a time series is represented in terms of a finite length, fast decaying, oscillating and discretely sampled waveform (mother wavelet), which is scaled and translated in order to create an orthonormal wavelet basis. Each function in the wavelet basis is related to a real coefficient: the original series is reconstructed by computing the weighted sum of all the functions in the basis, using the corresponding coefficient as weight. The Haar basis [3] is the most widely used in wavelet transformation. The DWT representation of a time series of length n consists in identifying n wavelet coefficients, whereas a dimensionality reduction is achieved by maintaining only the first p coefficients (with $p \ll n$).

SD is a data compression technique that belongs to the family of piecewise linear trending functions. Recently, SD has been adopted in several PI data analysis scenarios (e.g., [34]). Also, in [12], SD has been compared to wavelet compression. The SD algorithm employs a heuristic to decide whether a value is to be stored within the segment being grown or it is to be the beginning of a new segment. Given a pivot point, which indicates the beginning of a segment, two lines (the "doors") are drawn from it to envelop all the points up to the next one to be considered. The envelop has the form of a triangle according to a parameter that specifies the initial amplitude of the lines. The setup of this parameter has impact on the data compression level.

In the PLA method, a time series is represented by a piecewise linear function, i.e., a set of line segments. Several approaches have been proposed to recognize PLA segments (e.g., [28,17]); among these methods, the most efficient ones are able to produce a PLA representation with computational complexity linear with the length of the time series.

PAA transforms a time series of n points in a new one composed by p segments (with $p \ll n$), each of which is of size equal to n/p and is represented by the mean value of the data points falling within the segment. Like PAA, APCA approximates a time series by a sequence of segments, each one represented by the mean value of its data points. A major difference from PAA is that APCA can identify segments of variable length. Also, the APCA algorithm is able to produce high-quality approximations of a time series by resorting to solutions adopted in the wavelet domain.

The SAX representation of a time series involves three steps. Initially, the PAA version of a time series is computed, then the PAA coefficients are quantized, and finally each quantization level is represented by a single character, called SAX symbol.

Note that representing a time series of n points according to DWT, SD, (the fastest versions of) PLA, PAA and SAX can be performed in $\mathcal{O}(n)$, whereas the complexity of APCA is $\mathcal{O}(n \log(n))$.

Dimensionality reduction techniques based on piecewise discontinuous approximations can be combined with existing similarity measures, in order to improve the computational cost in similarity searches. In particular, the use of DTW on the coefficients obtained by segmenting a time series has been investigated in the literature (e.g., [18]), and several lower bounding measures operating on segmented versions of a time series have been defined. Among these methods, the Fast search method for dynamic Time Warping (FTW) [33] has been recently proposed as one of the most effective methods that use the time warping distance on a coarse version of the original sequences.

The other approach to dimensionality reduction, which approximates a time series with a continuous polynomial, includes Singular Value Decomposition (SVD) [22,14], Discrete Fourier Transforms (DFT) [32,31], splines, non-linear regression and Chebyshev polynomials [4,25]. SVD consists of space rotation and truncation applied on a data matrix and is computationally more expensive than all the other discussed methods for dimensionality reduction. DFT and Chebyshev approaches are quite close to DWT: they are based on the use of a set of orthonormal functions, whose contributions to the whole representation are given by the relating coefficients. Major differences among these representations regard the functions that compose the orthonormal basis (i.e., sine waves for DFT, and Chebyshev polynomials for Chebyshev) and the computational cost (i.e., $\mathcal{O}(n \log n)$ for DFT, and $\mathcal{O}(n)$ for Chebyshev). Also, Chebyshev approximation is very close to the optimal minimax polynomial, which represents an approximation able to minimize the maximum deviation

from the original data points.

It has been recently observed from an empirical viewpoint that there is no absolute winner among the dimensionality reduction methods in every application domain. [1] Nevertheless, it is interesting to compare the above methods with respect to the desiderata discussed in the Introduction. The requirement for time warping-aware representation is not satisfied by methods based on orthonormal functions such as DFT and Chebyshev, while the requirement for low complexity is satisfied by a few methods (e.g., DWT, Chebyshev, PAA, SAX and our DSA). The sensitivity to features can be considered according to three main sub-requirements (which are satisfied by our DSA) for the segments detected in an individual time series: *i)* segments may have different lengths, *ii)* any segment represents different slopes of a subsequence of data points, *iii)* segments capture the series trends. Point *i)* is addressed by APCA and SD, but not by SAX, PAA and PLA; SD and PLA satisfy point *ii)*, unlike PAA, SAX, and APCA; finally, no one of such methods satisfies point *iii)*. Also, most dimensionality reduction methods are not parameter-free, which is instead an advantageous feature of DSA—indeed, the threshold used to control the segmentation step is automatically determined in DSA, consequently the user is not required to specify it.

3 Derivative time series Segment Approximation

In this section we describe our *Derivative time series Segment Approximation (DSA)* model to represent time series into a concise form which is designed to capture the significant variations in the time series profile. [2] More precisely, a DSA sequence is the result of a transformation that applies to a time series and yields a shorter sequence of values approximating the segments identified in the derivative version of the original series. DSA entails derivative estimation, segmentation and segment modeling to map a time series into a different value domain which allows for maintaining information on the significant features of the original series in a dense and concise way.

We hereinafter denote a time series with $T = [(x_1, z_1), \ldots, (x_n, z_n)]$, where each couple (x_h, z_h) is composed by a real numeric value (x_h) and a timestamp (z_h); as is often the case by assuming a fixed sampling period, T can be simply rewritten as $T = [x_1, \ldots, x_n]$. Also, we can assume that the timestamp associated with the first point x_1 is set to be zero.
Given a time series T of length n, DSA computes a new sequence τ of p values, with $p \ll n$, by three main steps:

(1) Derivative estimation — the original time series is transformed into a new one in which each point is replaced with its first derivative estimate.

[1] http://www.cs.ubc.ca/∼rng/psdepository/chebyReport2.pdf
[2] A preliminary version of DSA was originally presented in [10]

(2) Segmentation — the derivative time series is decomposed into variable-length segments, each of which is comprised of a subsequence of points having close slopes.
(3) Segment approximation — the individual segments are substantially mapped to angular values, which represent synthetic information on the average slopes within the segments.

3.1 Derivative estimation

Given a time series $T = [x_1, \ldots, x_n]$, the derivative estimation step yields a sequence $\dot{T} = [\dot{x}_1, \ldots, \dot{x}_n]$, whose elements are first derivative estimates of the points in T.

A simple yet effective derivative estimation model is that exploited in [19]—we hereinafter refer to it as DDTW estimation model—which computes, for each point (except the first and the last one in the series), the mean value between the slope of the line from the left neighbor to the point and the slope of the line from the left neighbor to the right neighbor. Formally:

$$\dot{x}_h = \begin{cases} \dot{x}_{h+1} & \text{if } h = 1 \\ \frac{1}{2}[(x_h - x_{h-1}) + \frac{1}{2}(x_{h+1} - x_{h-1})] & \text{if } h \in [2..n\text{-}1] \\ \dot{x}_{h-1} & \text{if } h = n \end{cases} \tag{1}$$

We slightly modify the DDTW estimation model by also considering the slope of the line from the point to the right neighbor; actually, this modification leads to an algebraic simplification producing an expression that is equivalent to consider only the slope of the line from the left neighbor to the right neighbor. The derivatives of the first and the last point in the series are computed by taking into account their respective neighbors as well. Formally:

$$\dot{x}_h = \begin{cases} x_{h+1} - x_h & \text{if } h = 1 \\ \frac{1}{2}(x_{h+1} - x_{h-1}) & \text{if } h \in [2..n\text{-}1] \\ x_h - x_{h-1} & \text{if } h = n \end{cases} \tag{2}$$

We investigated how the performances of DSA and DDTW may vary depending on the derivative estimation model. As we describe in Appendix A, the DSA derivative estimation model (Eq. 2) leads to a better derivative-based feature space than the DDTW derivative estimation model (Eq. 1).

3.2 Segmentation

Segmenting a time series of length n consists in identifying $p-1$ delimiter points ($p \ll n$) to partition the series into p contiguous subsequences of points (segments) having similar features.

In our approach, segmentation is computed on the derivative version of a time series. Precisely, a derivative time series $\dot{T} = [\dot{x}_1, \ldots, \dot{x}_n]$ is transformed into a sequence $S_{\dot{T}} = [s_1, \ldots, s_p]$ of variable-length segments of the form $s_j = [s_{j,1}, \ldots, s_{j,k_j}] = [\dot{x}_{j_1}, \ldots, \dot{x}_{j_{k_j}}]$, such that:

- $s_{1,1} = \dot{x}_1$ and $s_{p,k_p} = \dot{x}_n$, and
- for each $j \in [1..p\text{-}1]$, s_{j,k_j} immediately precedes $s_{j+1,1}$ in the time axis.

A critical aspect in segmentation is how to determine the segment delimiters. For this purpose, we follow a sliding windows approach, i.e., a segment is grown until it exceeds an error threshold, and the process continues from the next point not yet considered. Although more refined segmentation schemes could be devised (e.g., top-down or bottom-up schemes), in this work we chose to pursue the above idea for the sake of its simplicity.

The key idea in our segmentation method is to break a series according to the first point such that the absolute difference between it and the mean of the previous points is above a certain threshold ϵ; this point becomes the anchor for the next segment to be identified in the rest of the series. Formally, let $\mu(s_j)$ denote the average of the points in a sequence s_j of $S_{\dot{T}}$, i.e., $\mu(s_j) = (\sum_{h=1}^{k_j} \dot{x}_{j_h})/k_j$, for each $j \in [1..p\text{-}1]$. The sequence s_j is identified as a segment if and only if

$$|\mu([s_{j,1}, \ldots, s_{j,h}]) - s_{j,h+1}| \leq \epsilon, \ \forall h \in [1..k_j - 1]$$

and

$$|\mu([s_{j,1}, \ldots, s_{j,k_j}]) - s_{j+1,1}| > \epsilon$$

Intuitively, this condition allows for aggregating subsequent data points having very close derivatives; in such a way, the growth segment s_j represents a subsequence of points with a specific trend.

To estimate the threshold ϵ, we resort to an index of spread of the (derivative) data points within a sequence. The objective is to produce a number of segments that is large enough to capture the "characteristic" trends in the original series (i.e., subsequences of points having close derivative estimates), but small enough to guarantee a reasonably good degree of compression. Precisely, we devise three definitions of ϵ, namely *dataset-oriented*, *series-oriented*, or *segment-oriented*.

The dataset-oriented definition of ϵ aims to express this threshold by globally referring to a given collection of time series. Given a dataset $\mathcal{D}$ of N time

series, ϵ can be defined as:

$$\epsilon(\mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N}\frac{|\dot{T}_i|}{\max\{|\dot{T}| \mid T \in \mathcal{D}\}}\sigma(\dot{T}_i)$$

where $\sigma(\dot{T}_i)$ denotes the standard deviation over the points in the i-th derivative series, and normalization of the series lengths is provided to deal with variable-length series.

The above definition is reasonably adequate when most time series in the collection show similar shapes; however, this may not necessarily hold in several real domains (e.g., sensor network measurements). Therefore, an estimation of ϵ might be provided by a series-oriented definition, i.e., globally to each individual time series T:

$$\epsilon(\dot{T}) = \sigma(\dot{T})$$

Another definition of ϵ may involve the individual segments being identified in each series. We can hence define a segment-oriented ϵ for each segment s_j as

$$\epsilon(s_j) = \sigma(s_j)$$

It is easy to observe that, regardless the definition of ϵ, the segmentation step on a dataset of N series can be performed in $\mathcal{O}(N \times n_{max})$, where n_{max} is the maximum of the series lengths. Since the segment-oriented definition is tailored to the local features of an individual series, we chose to adopt it in the segmentation step of our DSA model.

It is worth noting that the segmentation step in DSA does not require any user-specified parameter, since the threshold ϵ is automatically chosen by analyzing the information of each series. By contrast, this step is not automatic for other methods of dimensionality reduction (i.e., APCA, SAX, SD, PAA, PLA, Chebyshev, DWT and DFT), where the user is required to specify an input, such as the number of segments or coefficients being computed. We believe this represents an important advantage of our method.

3.3 Segment approximation

The individual segments of a derivative time series are represented with a synthetic information capturing their respective main features. More precisely, each segment s_j is mapped to a pair formed by the value z_j+1, where z_j is the timestamp of the last point ($\dot{x}_{j_{k_j}}$) in s_j, and an angle that explains the average slope of the portion of time series bounded by s_j. This is mathematically expressed by the notion of arctangent applied to the mean of the (derivative) points in each segment.

Given a segmented derivative time series $S_{\dot{T}} = [s_1, \ldots, s_p]$, the final step of segment approximation yields a sequence $\tau = [(\alpha_1, t_1), \ldots, (\alpha_p, t_p)]$ such that

$$\alpha_j = \arctan(\mu(s_j)), \quad j \in [1..p]$$
$$t_j = t_{j-1} + k_j, \qquad j \in [1..p]$$

where we assume $t_0 = 0$ for any DSA sequence.

Modeling a given time series by means of the DSA representation hence leads to a new sequence whose elements (pairs angle-timestamp) still maintain a direct association to the original time domain, while concisely representing the features of original points. This makes the DSA model able to fully support dynamic time warping, i.e., (dis)similarities between DSA sequences can be computed by using DTW-based measures.

As a final remark, it is easy to observe that the time complexity of computing a DSA sequence from a time series of length n has a total cost $\mathcal{O}(n)$, since the three steps, namely derivation, segmentation and segment approximation, cost $\mathcal{O}(n)$, $\mathcal{O}(n)$, and $\mathcal{O}(p)$ ($p \ll n$), respectively.

4 Experimental methodology

We devised an experimental evaluation to assess the ability of our DSA in supporting effective and efficient similarity detection within clustering and classification frameworks. We compared DSA against state-of-the-art methods for modeling and comparing time series data, which include LCSS, EDR, ERP, DTW, DDTW and FTW as distance measures, and APCA, SAX, PAA, PLA, SD, Chebyshev, DWT and DFT as dimensionality reduction methods. Since our DSA and the competing dimensionality reduction methods are not similarity/distance measures, we chose to apply DTW over the segments/coefficients computed by each particular representation scheme in the time domain (i.e., APCA, SAX, PAA, PLA, SD and DSA), whereas we used the Euclidean distance (L_2) to compare the sequences obtained by Chebyshev, DWT and DFT, as suggested in their respective works.

Before going into the details of the experimental results, in this section we introduce the clustering and classification algorithms and the validity criteria used in the experimental evaluation. We also discuss the preliminary task of preprocessing of the raw time series and the setup of the various methods that compete with our DSA.

4.1 Algorithms

Finding the best strategy of time series clustering or classification is not an objective of this work; rather, we are interested in assessing the impact of

the proposed time series representation model in similarity detection, and hence we conceived standard clustering and classification frameworks for time series data. Specifically, we resorted to well-known paradigms, namely partitional clustering, agglomerative hierarchical clustering [13] and nearest neighbor classification [26]. As stated in [23], partitional and hierarchical clustering methods have been extensively used in the context of time series clustering. Analogously, using nearest neighbor learning has been widely encouraged for time series classification (e.g., [20]).

4.1.1 Partitional clustering

The partitional clustering paradigm is characterized by simplicity, and low computational and memory requirements. In this work, we mainly use the popular K-Means algorithm [13], which is a centroid-based partitional clustering method; in Appendix C we also discuss a more general partitional clustering method. It is worth noting that choosing the number of output clusters does not represent a drawback in our evaluation context, since we selected datasets for which reference classifications are available, and hence we were able to fix the number of clusters equal to the actual number of classes in each clustering experiment. Also, we address the random selection of the initial cluster centroids by performing multiple runs of the K-Means algorithm to avoid that the quality results were due to random chance. In order to define the cluster centroids, we adopt two strategies depending on whether or not the representation model produces variable-length segments. Concerning SAX, PAA, PLA, SD, Chebyshev, DWT and DFT, we compute the cluster representatives by simply averaging the corresponding coefficients over the time series in any specific cluster. In the following, we present a method for computing cluster representatives of DSA sequences; we remark that although this method has been originally conceived for DSA clusters, it can be easily adapted to any representation model that is able to produce variable-length segments (coefficients), such as APCA.

Computing cluster representatives in K-Means. Let us denote with $\mathcal{T}_C = \{\tau_1, \ldots, \tau_M\}$ a cluster of DSA sequences, where each τ_i has the form $[(\alpha_{i_1}, t_{i_1}), \ldots, (\alpha_{i_{p_i}}, t_{i_{p_i}})]$, and with $C = \{T_1, \ldots, T_M\}$ the cluster of original time series such that each $T_i \in C$ is associated with a unique $\tau_i \in \mathcal{T}_C$. The objective is to compute a DSA sequence prototype $rep(\mathcal{T}_C)$ as the representative of cluster C.

We identify a fixed number v of segments over which the average series is defined. We can reasonably define the number of segments v as the closest integer to the mean $(\sum_{i=1}^{M} p_i)/M$ over all the series $\tau_i \in \mathcal{T}_C$. The timestamps associated to the new v segments are defined as $\hat{t}_j = t_{max} \times j/v$, for each $j \in [1..v]$, where $t_{max} = \max\{t_{u_{p_u}} \mid \tau_u \in \mathcal{T}_C\}$ $(i \in [1..M])$ and $\hat{t}_0 = 0$. For each τ_i, the angle α'_{i_j} corresponding to the timestamp $\hat{t}_j$ (with $\hat{t}_j \leq t_{i_{p_i}}$) is computed

to be equal to the angle α_{i_u} of the u-th segment including the point sampled at time $\hat{t}_j$. Formally, α'_{i_j} is equal to the angle α_{i_u} such that $(\alpha_{i_u}, t_{i_u}) \in \tau_i$ and $t_{i_{u-1}} < \hat{t}_j \leq t_{i_u}$, for all $i \in [1..M], j \in [1..v]$. Note that any pair $(\alpha'_{i_j}, \hat{t}_j)$ is introduced only if the condition $\hat{t}_j \leq t_{i_{p_i}}$ holds, i.e., if the i-th time series is defined in the timestamp $\hat{t}_j$.

For each τ_i the new v_i pairs $(\alpha'_{i_j}, \hat{t}_j)$ are then included in the rewritten DSA sequence $\tau''_i = [(\alpha''_{i_1}, t''_{i_1}), \ldots, (\alpha''_{i_{q_i}}, t''_{i_{q_i}})]$ which is computed as

$$\textit{time-sort}\{(\alpha_{i_1}, t_{i_1}), \ldots, (\alpha_{i_{p_i}}, t_{i_{p_i}}), (\alpha'_{i_1}, \hat{t}_1), \ldots, (\alpha'_{i_{v_i}}, \hat{t}_{v_i})\}$$

where $\hat{t}_{v_i}$ is the new timestamp with maximum value defined over the i-th sequence.
Formally, for each $\tau_i \in \mathcal{T}_C$, the sequence $\hat{\tau}_i = [(\hat{\alpha}_{i_1}, \hat{t}_1), \ldots, (\hat{\alpha}_{i_v}, \hat{t}_{v_i})]$ is computed, where

$$\hat{\alpha}_{i_j} = \frac{\sum_{(\alpha''_{i_u}, t''_{i_u}) \in \tau''_i \,\wedge\, t''_{i_u} \in (\hat{t}_{j-1}, \hat{t}_j]} [\ \alpha''_{i_u} \times (t''_{i_u} - t''_{i_{u-1}})\ ]}{\hat{t}_j - \hat{t}_{j-1}}, \quad j \in [1..v_i]$$

The DSA representative $rep(\mathcal{T}_C)$ is finally computed as:

$$rep(\mathcal{T}_C) = [(\overline{\alpha}_1, \hat{t}_1), \ldots, (\overline{\alpha}_v, \hat{t}_v)], \quad \text{where} \quad \overline{\alpha}_j = \frac{\sum_{\hat{t}_j \leq \hat{t}_{v_i} \wedge i \in [1..M]} \hat{\alpha}_{i_j}}{|\{\hat{t}_j | \hat{t}_j \leq \hat{t}_{v_i}\}|}$$

for each $j \in [1..v]$. Note that $\Delta t = \hat{t}_j - \hat{t}_{j-1}$ is a constant for each $j \in [1..v]$.

4.1.2 Hierarchical clustering

The agglomerative hierarchical clustering paradigm allows us to test the competing methods in a clustering framework which does not rely on a notion of cluster prototype and on the cluster initialization. For this purpose, we use the UPGMA algorithm (Unweighted Pair Group Method using arithmetic Averages), which is based on group-average-linkage to compute the distance between any two clusters [13].

4.1.3 K-nearest neighbor classification

As the most basic instance-based learning method, the *K-Nearest-Neighbor (K-NN)* algorithm [26] is a straightforward approach to assess the various representation and similarity detection methods in our context. Indeed, according to the K-NN algorithm, the classification of an instance (time series) will be most similar to the classification of instances that are similar to each other.

To assess the effectiveness of classification algorithms, we directly compare the result of the automatic assignment with a reference (expected) organization of the data. Analogously, in clustering frameworks, we assess how well a clustering solution fits a given scheme of known classes, thanks to the availability of reference classifications for all the test datasets.

F-measure (F) [36] is the most commonly used external criterion, and is defined as the harmonic mean between the Information Retrieval notions of precision (P) and recall (R):

$$F = \frac{2 \times P \times R}{P + R}$$

Given a set $\mathcal{D}$ of series, let $\Gamma = \{\Gamma_1, \ldots, \Gamma_H\}$ be the expected organization of the series in $\mathcal{D}$, and $\mathcal{C} = \{C_1, \ldots, C_K\}$ be the output of a classification or clustering algorithm. Precision of C_j with respect to Γ_i is the fraction of the series in C_j that has been correctly classified, i.e., $P_{ij} = |C_j \cap \Gamma_i|/|C_j|$. Recall of C_j with respect to Γ_i is the fraction of the series in Γ_i that has been correctly classified, i.e., $R_{ij} = |C_j \cap \Gamma_i|/|\Gamma_i|$. In the case of classification, the condition $H = K$ holds, and the overall precision and recall are defined as

$$P = \frac{1}{H}\sum_{i=1}^{H} P_{ii} \qquad R = \frac{1}{H}\sum_{i=1}^{H} R_{ii}$$

whereas, in the case of clustering, the definitions are as follows:

$$P = \frac{1}{H}\sum_{i=1}^{H} P_{i} \qquad R = \frac{1}{H}\sum_{i=1}^{H} R_{i}$$

where each P_i and R_i are equal to P_{ij^*} and R_{ij^*}, respectively, such that $j^* \in argmax_{j=1..K}\{P_{ij}, R_{ij}\}$.

Another popular measure used to compare two classifiers is the *error rate* (E), which takes into account both errors of commission and errors of omission:

$$E = \frac{1}{2 \times |\mathcal{D}|}\sum_{i=1}^{H}(|C_i \setminus \Gamma_i| + |\Gamma_i \setminus C_i|)$$

The *accuracy* of a classifier is defined in terms of the error rate as $A = 1 - E$. Note that all the measures above range within zero and one; in particular, higher values of F-measure and accuracy indicate better quality.

4.3 Preprocessing time series

Raw time series are usually preprocessed by *smoothing* data points in order to reduce the noise in the data. Moving average represents the simplest family of smoothing models, as it is a compromise between the mean and the random walk model. Given a raw time series $T = [x_1, \ldots, x_n]$ and a smoothing degree δ (i.e., the maximum width of the moving average), the *centered δ-point moving average* recomputes the data points by considering both the previous and next observations around a center:

$$x_h^{smoothed} = \begin{cases} \mu([x_1, \ldots, x_{h+r}]) & \text{if } h-r \leq 0 \\ \mu([x_{h-r}, \ldots, x_{h+r}]) & \text{if } h-r > 0 \text{ and } h+r \leq n \\ \mu([x_{h-r}, \ldots, x_n]) & \text{if } h+r > n \end{cases}$$

where $r = (\delta - 1)/2$ denotes the maximum number of back and forward points that are taken into account for smoothing the h-th point.

More refined models, such as exponential smoothing models, compute the weighted average of past observations on the basis of previously smoothed observations. Given a smoothing factor $\omega \in [0..1]$, the simple *exponential smoothing* is computed as:

$$x_h^{smoothed} = \begin{cases} x_h & \text{if } h = 1 \\ \omega x_h + (1-\omega) x_{h-1}^{smoothed} & \text{if } h > 1 \end{cases}$$

It should be emphasized that denoising is essential to make the data amenable to the further analysis tasks, regardless of the specific representation method or distance measure used. In particular, in derivative-based feature spaces, denoising time series data (e.g., via a smoothing function) before differentiating them is necessary to avoid that the approximation of derivatives by finite differences will amplify the noise present in the data. In this respect, the combination of smoothing prior to the step of derivative estimation in our DSA approach (as well as in DDTW) can be seen as somehow similar to the regularization of a differentiation process [35], although potentially less accurate and general.

4.4 Setup of the competing methods

Unlike our DSA, most of the competing methods require one or more parameters to be set. In some cases, which include LCSS, EDR, ERP and Chebyshev,

typical settings are suggested in their respective works; specifically, the matching thresholds for LCSS and EDR are assumed to be equal to $(\max \sigma(T_i))/4$ and $\min \sigma(T_i)$ respectively (for all the series T_i in a dataset), the constant gap for ERP is set to 0, and the number of coefficients for Chebyshev is set to 20. Such parameter settings revealed to be good enough to enable the respective methods to achieve their best performances in accuracy. In particular, we took care in monitoring the behavior of the Chebyshev method, and finally found no significant improvement in accuracy by increasing the number of Chebyshev coefficients.

In other cases, to make a comparative evaluation possible in terms of accuracy and efficiency, we aimed to prepare the various methods to perform at levels of data compression which were as close as possible. We tried several values for the parameters of APCA, SAX, PAA, PLA, DWT, DFT and FTW. More precisely, for each dataset and algorithm, we varied the setting of each of these methods in such a way that it achieved the same compression (i.e., number of segments) obtained by DSA, and $\pm 5\%$, $\pm 10\%$, and $\pm 20\%$ of the DSA compression; then, we measured the relative clustering/classification quality (F-measure) scores and finally chosen the setting corresponding to the best score. Analogously, the alphabet length W (i.e., the number of symbols) required by SAX was chosen, for each dataset and clustering/classification algorithm, as the value that led to the best trade-off between clustering/classification quality and time performance.

A final remark concerns SD, which requires a deviation threshold (i.e., the "doors" amplitude); however, setting this parameter is even more difficult, since the compression factor (i.e., the number of segments) cannot be specified directly in swinging door compression. In [34,40], many calibration trials are conducted to find the deviation thresholds corresponding to a given compression factor, for each dataset. We followed this approach and set the deviation threshold in such a way that the number of segments produced by SD was as near as possible to the number of segments produced by DSA, finally choosing the value that led to best clustering/classification quality.

5 Results

5.1 Data description

We selected seven datasets coming from various application domains, and characterized by different series profiles and dimensionality. Table 1(a) summarizes the characteristics of the main datasets used for the experiments, and Fig. 1 shows the shapes of sample representative instances in each dataset.

GunX comes from the video surveillance domain, whereas Tracedata simulates signals representing instrumentation failures. In CBF (Cylinder-Bell-Funnel), each class is characterized by a specific pattern, namely a plateau (C), an in-

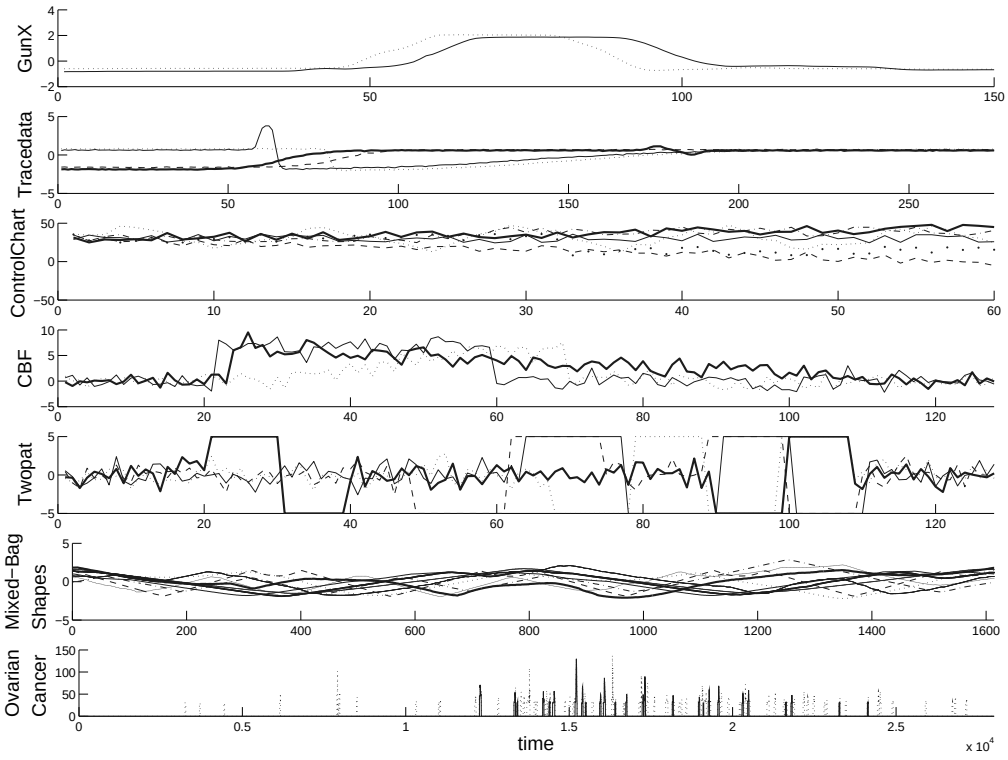


Fig. 1. Sample instances from the test datasets. One time series from each class is displayed for each dataset.

creasing ramp followed by a sharp decrease (B), a sharp increase followed by a decreasing ramp (F). ControlChart contains synthetically generated control charts which are classified into one of the following: normal, cyclic, increasing trend, decreasing trend, upward shift, and downward shift. In Twopat, two different patterns (upward step and downward step) are used to define the classes down-down, up-down, down-up, and up-up. Mixed-BagShapes contains time series derived from shapes belonging to nine classes (bone, cup, device, fork, glass, hand, pencil, rabbit and tool). The first five datasets are available at http://www.cis.temple.edu/∼latecki/TestData/TS_Koegh/, whereas Mixed-BagShapes can be found at http://www.cs.ucr. edu/∼eamonn/shape/.

Besides the benchmark datasets above, we also used OvarianCancer [29], which contains proteomic spectra generated by Surface-Enhanced Laser Desorption and Ionization - Time Of Flight Mass Spectrometry (SELDI-TOF MS). The spectra (i.e., MS data) are derived from an analysis of serum samples of a female population belonging to two classes (ovarian cancer diseased and healthy). It should be emphasized that OvarianCancer data, like most of MS datasets, are huge-dimensional and largely affected by noisy factors. Noise is typically due to a number of reasons, such as sample preparation, insertion of the samples into the mass spectrometer, and instrumental and measurement errors. For this purpose, OvarianCancer spectra were subject to a preliminary preprocessing phase specific for MS data [27,38]. MS preprocessing has been

Table 1
Datasets used in the experiments: (a) information on the original series, and (b) segmentation and compression using DSA

(a)

dataset	*size*	*classes*	*time steps*
GunX	200	2	150
Tracedata	200	4	275
ControlChart	600	6	60
CBF	300	3	128
Twopat	800	4	128
Mixed-BagSh.	160	9	1,614
OvarianCancer	49	2	28,000

(b)

dataset	*avg. no. DSA segments*	*compr.*
GunX	68	55%
Tracedata	118	57%
ControlChart	35	42%
CBF	77	40%
Twopat	38	70%
Mixed-BagSh.	816	49%
OvarianCancer	943	97%

recognized as a crucial step for tasks of MS data management and knowledge discovery, and mainly consists of operations such as noise reduction, baseline subtraction and peak detection. The interested reader can find details about the preprocessing steps carried out in [11].

It is interesting to have a look at the impact on the time series dimensionality by using DSA. Table 1(b) shows that DSA achieves a 59% compression of the original series lengths on average, with a maximum compression percentage of 97% in the OvarianCancer dataset. As we shall discuss later in this section, the reasonably good rate of compression achieved by DSA does not have a negative impact on the accuracy in detecting similarities.

5.2 Effectiveness evaluation

We measured the ability of DSA and the competing methods in supporting time series clustering and classification effectively. We investigated how clustering/classification results can be influenced by choosing different alternatives for data preprocessing and setting the parameters therein involved; then, we assessed the performance of DSA and the other methods according to their respective best settings.

5.2.1 Tuning preprocessing parameters

We used the smoothing functions previously described in Section 4.3 to preprocess the time series in each dataset. In the case of centered moving average, the smoothing degree $\delta(= 2r + 1)$ was varied within a typical range, namely [5..9], whereas ω in exponential smoothing settings was varied from 0 to 1 by a 0.1 step. Moreover, we tried to perform zero, one or more iterations of smoothing (up to 5), on the various datasets and for each preprocessing scheme; the

rationale here is that smoothing should be avoided to prevent loss of information for low-noise series and, conversely, excessive noise might be treated with multiple smoothing. It should be noted that, in the case of K-Means evaluation, we performed multiple runs (100) of the K-Means algorithm and finally averaged the quality results over the runs to obtain a single value of F-measure.

We observed that smoothing helped to improve the performance of the various methods on all the datasets, except OvarianCancer; OvarianCancer represented an exception since, in this case, data was preliminarily subject to domain-specific preprocessing steps (Section 5.1), hence further preprocessing via smoothing would have tended to cause loss of information on potentially significant data features.

The exponential smoothing revealed to be more effective than moving average, as it was selected 74 out of 90 times as the best preprocessing way. The parameter ω was set to low values in most cases, thus suggesting the need for a greater smoothing effect (which is indeed achieved by values of ω closer to zero). Also, the number of smoothing iterations appeared to be not relevant in practice; three iterations of smoothing were enough in most cases, except for SD which always required four or five iterations. In Appendix B, we report further details about the preprocessing stage, including the best settings and an evaluation of the impact of smoothing on the various datasets.

5.2.2 Accuracy in time series clustering

We evaluated DSA and the other methods in two clustering frameworks, namely K-Means and UPGMA. In relation to the selected clustering algorithm, each method was used with the best preprocessing setup for the specific dataset. In any case, the quality of the obtained clustering solutions was calculated in terms of F-measure.

Table 2 refers to K-Means clustering and shows the quality results averaged over 100 runs of this algorithm. Looking at the table we can observe that DTW on DSA sequences (for short, DTW on DSA) was the first ranked method in all the datasets except CBF; however, in this dataset, DTW on DSA was only 1% below the performance of DDTW, which turned out to be the best method among the competing ones. Also, DTW on DSA always led to better results than DTW alone. It should be emphasized that the comparison with DDTW is particularly important in order to gain an insight into the role of derivative-based features in time series representation and the impact of combining derivative estimation and segmentation in the accuracy of similarity detection.

DTW on DSA performed as good as or better than the remaining methods, and in some cases the performance difference was quite evident. In particular, DTW on DSA led to quality improvements up to about 59% with respect to

Table 2
Summary of average quality results (F-measure) for K-Means clustering

	GunX	Trace data	Control Chart	CBF	Twopat	Mixed BagSh.	Ovarian Cancer
LCSS	0.59	0.30	0.50	0.79	0.36	0.32	0.34
EDR	0.54	0.74	0.88	0.86	0.42	0.70	0.58
ERP	0.72	0.62	0.76	0.58	0.39	0.48	0.34
DTW	0.66	0.78	0.87	0.89	*0.95*	*0.77*	0.60
DDTW	*0.89*	*1*	*0.89*	*0.97*	*0.95*	0.76	*0.62*
FTW	0.74	0.90	0.81	0.67	0.55	0.73	0.58
L_2 on DFT	0.63	0.77	0.78	0.67	0.39	0.70	0.36
L_2 on DWT	0.61	0.67	0.76	0.74	0.36	0.68	0.36
L_2 on CHEBY	0.57	0.72	0.70	0.69	0.38	0.72	0.34
DTW on SD	0.67	0.95	0.85	0.87	0.89	0.76	0.69
DTW on PLA	0.73	0.77	*0.89*	0.87	0.75	0.74	0.60
DTW on PAA	0.68	0.78	0.87	0.86	0.73	0.75	0.59
DTW on SAX	0.73	0.77	0.83	0.87	0.69	0.71	0.58
DTW on APCA	0.77	0.81	*0.89*	0.83	0.91	0.74	0.55
DTW on DSA	**0.92**	**1**	**0.90**	**0.96**	**0.97**	**0.78**	**0.75**

DWT, DFT and Chebyshev, and up to 25% with respect to SAX, PAA, PLA, SD, and APCA. Among these competing methods, it can be noted that DTW on APCA and SD obtained the best results; in general, DTW on APCA, SD, SAX, PAA and PLA achieved higher quality clustering than Chebyshev, DWT and DFT.

Table 3 reports on the quality results obtained by the UPGMA algorithm. A first remark on these results is that the F-measure scores for the various methods were generally much lower than the corresponding results obtained by the K-Means algorithm on all the datasets, except OvarianCancer; in particular, for DTW on DSA, there was a quality decrease from 19% (in GunX) to 36%

Table 3
Summary of quality results (F-measure) for UPGMA clustering

	GunX	Trace data	Control Chart	CBF	Twopat	Mixed BagSh.	Ovarian Cancer
LCSS	0.47	0.23	0.38	0.39	0.24	0.24	0.33
EDR	0.49	0.38	0.41	0.47	0.28	0.28	0.56
ERP	0.49	0.37	0.40	0.40	0.33	0.28	0.54
DTW	0.61	0.48	0.48	0.51	0.61	0.38	0.61
DDTW	*0.72*	*0.76*	*0.54*	0.49	*0.64*	0.41	*0.63*
FTW	0.60	0.52	0.44	0.44	0.50	0.40	*0.63*
L_2 on DFT	0.60	0.40	0.29	0.47	0.39	0.28	0.62
L_2 on DWT	0.60	0.40	0.29	0.47	0.39	0.28	0.62
L_2 on CHEBY	0.60	0.40	0.29	0.47	0.39	0.28	0.62
DTW on SD	0.51	0.55	0.40	0.49	0.43	*0.42*	0.60
DTW on PLA	0.61	0.63	0.40	0.51	0.43	0.29	0.61
DTW on PAA	0.61	0.61	0.36	0.51	0.56	0.41	0.61
DTW on SAX	0.61	0.60	0.48	*0.56*	0.44	0.31	0.61
DTW on APCA	0.61	0.63	0.40	0.51	0.43	0.29	0.61
DTW on DSA	**0.73**	**0.82**	**0.54**	**0.60**	**0.67**	**0.51**	**0.73**

(in CBF and ControlChart). However, like in K-Means clustering, DTW on DSA revealed to behave as good as or better than the other methods and, in some cases (i.e., CBF and Mixed-BagShapes) we observed increased advantages (quality improvements) with respect to the corresponding results by K-Means.

5.2.3 Accuracy in time series classification

Analogously to the evaluation by clustering frameworks, we assessed the performance of DSA and the competing methods using the K-NN classification algorithm. Table 4 shows the best results obtained by the various methods,

where each triple of values refers to the parameter K, F-measure and accuracy, in that order. We split each dataset 70% for training, the remainder for testing. Concerning the choice of K (the neighborhood size), it should be emphasized that [20] recommends the use of the "simple yet very competitive" 1-NN algorithm (i.e., K-NN with K equal to 1); we followed the lead of this authoritative reference, however we also considered more values of K, up to $K = 5$.

Using DTW on DSA led to both F-measure and accuracy equal to 1 for all the selected K on GunX, Tracedata, CBF and Twopat. Moreover, DTW on DSA performed better than the other methods on Mixed-BagShapes and GunX (in this dataset, DDTW achieved optimal accuracy like DSA, although requiring a neighborhood size greater than 1). It should also be noted that DDTW confirmed to be the best method among the competing ones.

We observed that both F-measure and accuracy tended to decrease for higher values of K. In particular, the use of $K = 1$ revealed to be the best choice, in terms of F-measure or accuracy, 87 out of 105 times over all the methods and datasets—which advocates the aforementioned recommendation by [20]. [3]

5.3 Efficiency evaluation

We measured the time performances of DSA and the other methods in accomplishing the tasks of modeling and clustering time series. For each dataset, we randomly selected samples of sizes equal to 25%, 50%, 75% and 100% of the size of the entire dataset and, for each sample and method, we used the respective best preprocessing setup. [4] We left the string matching based approaches (i.e., LCSS, EDR and ERP) out of this presentation since they revealed to be drastically slower than all the other methods. [5]

5.3.1 Performances in time series modeling

Table 5 summarizes the best performances (in milliseconds) in modeling time series on the various datasets. PAA, PLA, SAX, SD and DWT performed as the fastest methods; actually, this result was not surprising since simpler

[3] The complete list of K-NN results for all the methods and datasets is available at http://www2.deis.unical.it/tagarelli/dsa/

[4] For each of the smaller samples of the datasets, we performed a preprocessing stage for the various methods, as we did for the entire datasets (see Section 5.2.1); however, we cannot present here details on such preprocessing tests due to the space limits of this paper.

[5] Experiments were conducted on a platform Intel Pentium IV 3GHz with 2GB memory and running Microsoft WinXP Pro.

Table 4
Summary of quality results (F-measure, accuracy, and the corresponding K in parentheses) for K-NN classification

	GunX	Trace data	Control Chart	CBF	Twopat	Mixed-BagSh.	Ovarian Cancer
LCSS	F=.63 (*3*) A=.63 (*3*)	F=.42 (*3*) A=.75 (*1*)	F=.65 (*1*) A=.85 (*1*)	F=.36 (*5*) A=.37 (*1*)	F=.51 (*1*) A=.68 (*1*)	F=.80 (*1*) A=.92 (*1*)	F=.29 (*1*) A=.63 (*3*)
EDR	F=.83 (*5*) A=.80 (*5*)	F=.95 (*1*) A=.96 (*1*)	F=1 (*1*) A=1 (*1*)	F=1 (*1*) A=1 (*1*)	F=1 (*1*) A=1 (*1*)	F=.76 (*3*) A=.94 (*2*)	F=.62 (*1*) A=.80 (*1*)
ERP	F=.99 (*1*) A=.99 (*1*)	F=1 (*1*) A=1 (*1*)	F=1 (*1*) A=1 (*1*)	F=.79 (*5*) A=.81 (*5*)	F=.64 (*1*) A=.75 (*1*)	F=.78 (*1*) A=.96 (*1*)	F=.58 (*1*) A=.63 (*1*)
DTW	F=.95 (*3*) A=.95 (*3*)	F=1 (*1*) A=1 (*1*)	F=1 (*1*) A=1 (*1*)	F=1 (*1*) A=1 (*1*)	F=1 (*1*) A=1 (*1*)	F=.84 (*1*) A=.97 (*1*)	F=.68 (*5*) A=.80 (*2*)
DDTW	F=1 (*2*) A=1 (*2*)	F=1 (*1*) A=1 (*1*)	F=1 (*1*) A=1 (*1*)	F=1 (*1*) A=1 (*1*)	F=1 (*1*) A=1 (*1*)	F=.92 (*1*) A=.98 (*1*)	F=.70 (*1*) A=.87 (*2*)
FTW	F=.89 (*4*) A=.88 (*4*)	F=1 (*1*) A=1 (*1*)	F=1 (*1*) A=1 (*1*)	F=.93 (*1*) A=.93 (*1*)	F=.88 (*1*) A=.90 (*1*)	F=.83 (*2*) A=.96 (*2*)	F=.68 (*3*) A=.80 (*1*)
L_2 on DFT	F=.96 (*1*) A=.96 (*1*)	F=1 (*1*) A=1 (*1*)	F=1 (*1*) A=1 (*1*)	F=.98 (*1*) A=.98 (*1*)	F=.77 (*3*) A=.77 (*3*)	F=.91 (*3*) A=.98 (*3*)	F=.65 (*1*) A=.70 (*1*)
L_2 on DWT	F=.90 (*5*) A=.89 (*5*)	F=.95 (*2*) A=.96 (*2*)	F=1 (*1*) A=1 (*1*)	F=1 (*1*) A=1 (*1*)	F=.90 (*1*) A=.93 (*1*)	F=.88 (*1*) A=.97 (*1*)	F=.65 (*1*) A=.80 (*2*)
L_2 on CHEBY	F=.71 (*1*) A=.71 (*1*)	F=1 (*1*) A=1 (*1*)	F=1 (*1*) A=1 (*1*)	F=1 (*1*) A=1 (*1*)	F=.91 (*1*) A=.93 (*1*)	F=.85 (*1*) A=.97 (*1*)	F=.61 (*1*) A=.77 (*2*)
DTW on SD	F=.99 (*5*) A=.99 (*5*)	F=1 (*1*) A=1 (*1*)	F=1 (*1*) A=1 (*1*)	F=1 (*1*) A=1 (*1*)	F=1 (*1*) A=1 (*1*)	F=.87 (*1*) A=.97 (*1*)	F=.68 (*4*) A=.73 (*1*)
DTW on PLA	F=.95 (*3*) A=.95 (*3*)	F=1 (*1*) A=1 (*1*)	F=1 (*1*) A=1 (*1*)	F=1 (*1*) A=1 (*1*)	F=1 (*1*) A=1 (*1*)	F=.86 (*1*) A=.97 (*1*)	F=.68 (*4*) A=.77 (*1*)
DTW on PAA	F=.99 (*1*) A=.99 (*1*)	F=1 (*1*) A=1 (*1*)	F=1 (*1*) A=1 (*1*)	F=1 (*1*) A=1 (*1*)	F=1 (*1*) A=1 (*1*)	F=.87 (*1*) A=.97 (*1*)	F=.67 (*2*) A=.73 (*2*)
DTW on SAX	F=.78 (*1*) A=.74 (*1*)	F=1 (*1*) A=1 (*1*)	F=.98 (*5*) A=.99 (*5*)	F=1 (*1*) A=1 (*1*)	F=.99 (*1*) A=.99 (*1*)	F=.80 (*2*) A=.95 (*1*)	F=.63 (*3*) A=.80 (*1*)
DTW on APCA	F=.95 (*3*) A=.95 (*3*)	F=1 (*1*) A=1 (*1*)	F=1 (*1*) A=1 (*1*)	F=1 (*1*) A=1 (*1*)	F=1 (*1*) A=1 (*1*)	F=.86 (*1*) A=.97 (*1*)	F=.68 (*4*) A=.77 (*1*)
DTW on DSA	**F=1 (*1*)** **A=1 (*1*)**	**F=1 (*1*)** **A=1 (*1*)**	**F=1 (*1*)** **A=1 (*1*)**	**F=1 (*1*)** **A=1 (*1*)**	**F=1 (*1*)** **A=1 (*1*)**	**F=.95 (*1*)** **A=.98 (*1*)**	**F=.75 (*1*)** **A=.90 (*1*)**

models obviously lead to higher efficiency and, at the same time, lower accuracy. DFT and APCA were always by far slower than the other methods. Our DSA was close to the fastest methods in most cases; in particular, compared to Chebyshev, the larger the dataset the faster was modeling by DSA with respect to modeling by Chebyshev polynomials. It is important to note

that, since DSA shares with PAA, PLA, SD and SAX the same asymptotic time complexity (i.e., linear with the series length), the time differences between DSA and these fast methods should be not really relevant in practical contexts.

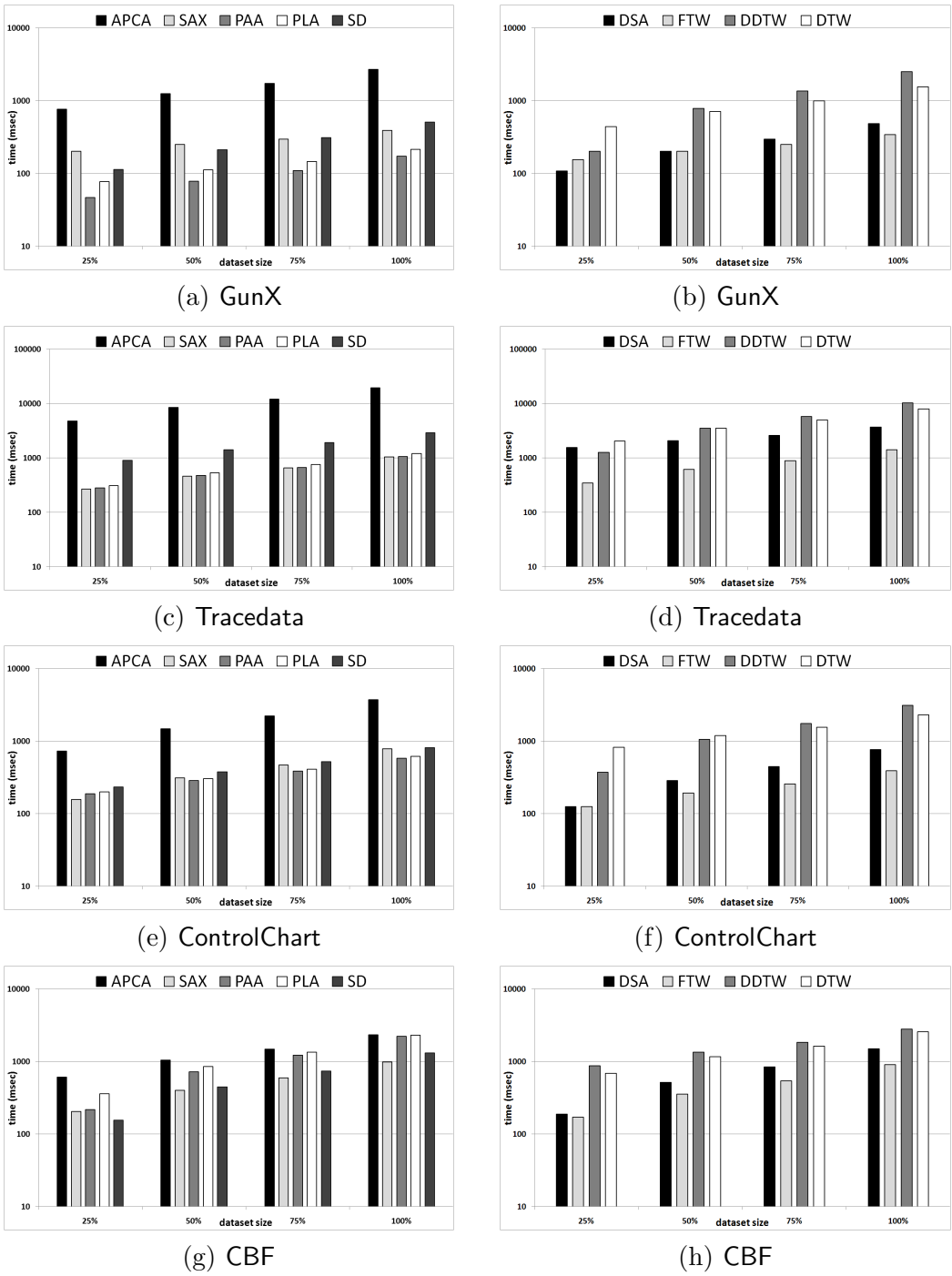
Table 5
Summary of best time performances (msecs) in time series modeling

	GunX	Trace data	Control Chart	CBF	Twopat	Mixed BagSh.	Ovarian Cancer
DFT	2,835	8,276	1,652	3,679	4,701	286,829	570,056
DWT	8	13	9	5	17	48	181
CHEBY	58	58	173	87	232	46	*14*
SD	7	15	14	17	24	65	262
PLA	4	7	*2*	3	*4*	21	46
PAA	*2*	*3*	*2*	*2*	*4*	*14*	41
SAX	8	13	11	11	19	56	67
APCA	1,758	7,151	412	657	2,358	68,739	135,982
DSA	**15**	**40**	**27**	**31**	**52**	**143**	**391**

5.3.2 Performances in time series clustering

We also evaluated the time performances for the clustering task, including in this stage the time required by the series modeling task as well; for the sake of brevity, we present here results obtained by the K-Means algorithm, and we focus on time warping-aware representations.

Figures 2–3 and the summary reported in Table 6 show that DTW on DSA drastically improved the clustering performances of basic DTW and DDTW; clearly, this was a consequence of the dimensionality reduction due to the segmentation performed by DSA. More surprisingly, DSA behaved very close to the fastest competing methods: indeed, it is interesting to note that the performance difference between DSA and PAA, SAX, PLA and SD was not as evident as in the modeling performances previously observed; in particular, DSA-based clustering was even faster than PAA, SAX, PLA and SD on Twopat (the largest dataset) and Mixed-BagShapes (the dataset with the highest number of classes). This suggests that our DSA is able to yield a time series representation that might require more time to be computed, but generally is more accurate yet convenient to fit the whole task of clustering.



(a) GunX (b) GunX

(c) Tracedata (d) Tracedata

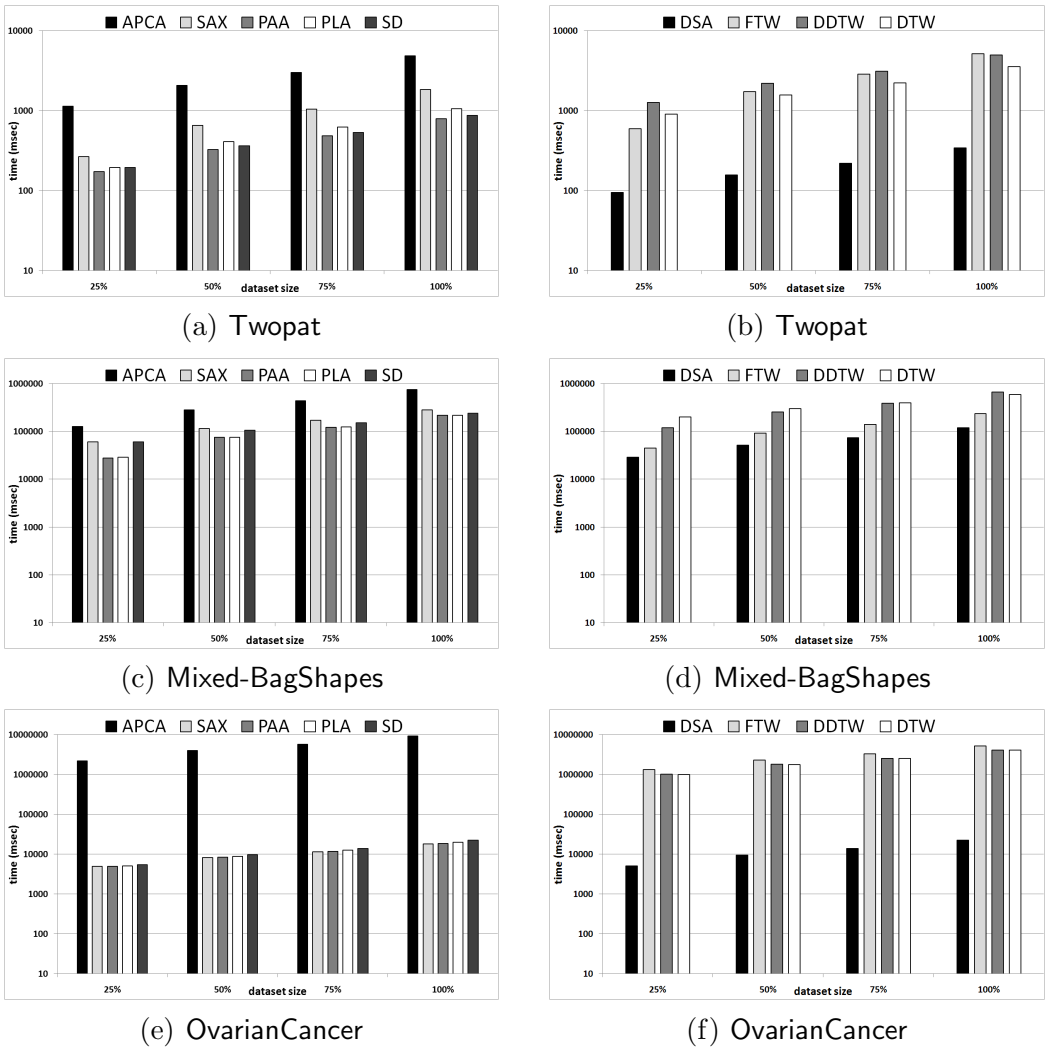
(e) ControlChart (f) ControlChart

(g) CBF (h) CBF

Fig. 2. Time performances in time series modeling and clustering: GunX, Tracedata, ControlChart, CBF

5.4 Summary of results and discussion

We evaluated the capabilities of our DSA as well as the competing methods in supporting similarity detection within clustering and classification frameworks. Of course, the extent to which such frameworks actually led to good



(a) Twopat (b) Twopat

(c) Mixed-BagShapes (d) Mixed-BagShapes

(e) OvarianCancer (f) OvarianCancer

Fig. 3. Time performances in time series modeling and clustering: Twopat, Mixed-BagShapes, OvarianCancer

solutions depend on how each of the following critical aspects was devised: *i)* the preprocessing scheme, *ii)* the similarity method and its application in relation to a representation model, and *iii)* the clustering/classification algorithms.

Since the focus of our work is on a compact representation of derivative-based features of time series and its impact on similarity detection, it should be emphasized that the evaluation frameworks are indeed "parametric" with respect to the algorithms. In order to provide a complete specification of our evaluation frameworks, we conducted experiments using standard clustering and classification algorithms mainly because of their simplicity, applicability, and relatively less dependence on algorithmic parameters.

Facing with the experimental results presented in the above sections and having the focus on point *ii)*, we can summarize the main remarks of our study as follows:

Table 6
Summary of best time performances (msecs) in time series modeling and clustering

	GunX	Trace data	Control Chart	CBF	Twopat	Mixed BagSh.	Ovarian Cancer
DTW	1,548	8,018	2,298	2,564	3,548	594,446	4,071,815
DDTW	2,518	10,345	3,117	2,789	4,971	664,885	4,098,329
FTW	368	1,491	*405*	*926*	5,254	237,823	5,252,045
DTW on SD	511	2,912	814	1,317	870	240,744	22,655
DTW on PLA	253	1,286	652	2,342	1,103	218,272	20,588
DTW on PAA	*197*	*1,103*	601	2,262	*812*	*216,196*	18,685
DTW on SAX	413	1,113	805	1,132	1,941	282,312	*18,403*
DTW on APCA	2,865	19,503	3,793	2,563	4,864	749,003	9,282,532
DTW on DSA	**521**	**3,733**	**792**	**1,587**	**377**	**121,402**	**23,821**

- Applying the dynamic time warping (DTW) to DSA sequences leads to clustering and classification solutions that are more accurate than those obtained by using DTW on the original time series. The advantage taken by DSA is essentially due to the combination of a derivative-based feature generation with dimensionality reduction by segmentation. In this way, DTW on DSA performs as good as or better than the major methods for time series representation and dimensionality reduction (i.e., SAX, APCA, PLA, SD, PAA, Chebyshev polynomials, etc.) and even than DDTW alone.
- Modeling a time series into a DSA sequence is reasonably fast if compared to other methods of dimensionality reduction, which is supported by a computational complexity that is linear with the series length; most importantly, the trade-off between accuracy and compactness of DSA sequences makes performing similarity detection more advantageous.

6 Application: Profiling of electricity company customers

We briefly present here a real case study on electricity customer profiling. This is part of our ongoing research on fraud detection in electricity customer data in the context of a research project subsidized by the ENEL Italian electricity power company.

6.1 Description

An electrical load profile represents an electricity consumption pattern for a customer or a group of customers over a given time period. The use of load profiles for electricity settlement has been identified in many countries as an effective solution to avoid the prohibitive costs of putting interval metering into every electricity customer.
We were granted to access customer data from ENEL over a two-month period. This customer data consists of two parts. The main part corresponds to load profiles which represent the amount of electricity consumption (Watt-hour) over a quarter-hourly period. The second part is represented by various meta-data associated to each customer load profile, which concerns settlement information and loading conditions, such as type of customer (i.e., domestic, commercial, and industrial customers), type of electronic meter, load power, location (e.g., urban, rural locations).
We aimed to analyze the consumption behavior of customers according to their load profiles and associated meta-data. It should be noted that meta-data represents a potentially useful resource for the task in hand, although they cannot directly used as a-priori knowledge on the customers' behavior; indeed, similar loading conditions may yield different load shapes of the customers.

6.2 Framework

We devised a simple framework to profile the customers' behavior, which consists of three main phases. In the first phase, an initial organization of the data is obtained. Customer load profiles are modeled based on DSA and grouped together using the agglomerative hierarchical clustering algorithm equipped with the DTW distance. The whole dendrogram is built and finally cut at an appropriate level corresponding to the best quality partition; in this case, the quality of a partition is measured as the difference between the inter-cluster distance (cluster separation) and the intra-cluster distance (cluster cohesiveness), i.e., the difference between the averages of the pair-wise distances of the objects between and within the clusters, respectively.
Once an initial organization of the data has been obtained, the next phase is to characterize the customer groups by two steps: *i)* computing a load profile prototype for each cluster, and *ii)* learning a model on the meta-data associated to the customers of each cluster. Finally, the third phase is to train a K-NN classifier based on the initial clustering solution. This classifier and the characterization of the customer groups establish a knowledge base which will be used on new load profiles in order to support further analysis for anomaly detection.

6.3 Evaluation

We conducted preliminary experiments on a data set of 2,864 customer load profiles, associated with a set of meta-data represented by 5 types of electronic

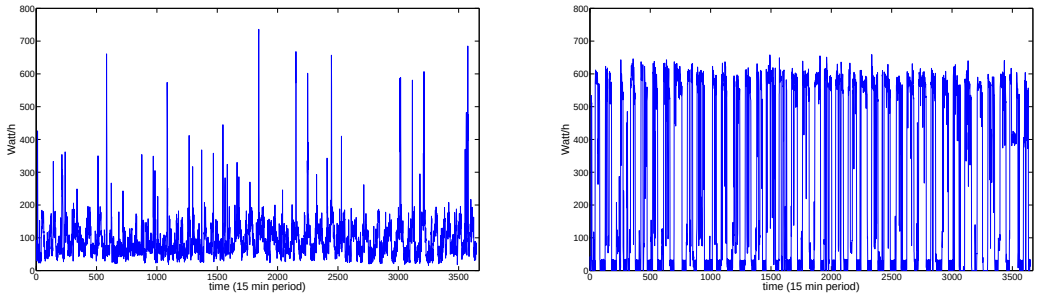


Fig. 4. Example load profiles: domestic customer (on the left) and non-domestic customer (on the right)

meter, 25 types of customer, about 20 different settlement information on load power, 15 districts and 20 different locations.

The UPGMA algorithm was performed on the set of (preprocessed) load profiles and the resulting dendrogram was cut at the level corresponding to a 14-cluster partition. For each of the resulting clusters, a load profile prototype was computed using our DSA-based cluster representative approach; Figure 4 shows two examples of load profile over a one-week period, which represent a cluster of domestic customers and a cluster of non-domestic customers, respectively. Moreover, a decision tree (C4.5) classifier was learned on the meta-data associated to the customers within any given cluster. We observed in each cluster that the electronic meter type, the customer type and the settlement load power were selected in that order as test attributes in the highest levels of the decision tree, whereas the leaf nodes mostly corresponded to districts and locations.

The initial organization of the customer load profiles obtained via clustering was used as training set for K-NN classification. We tested the K-NN classifier by varying the neighborhood size K (up to 5) and the validation setting (cross-validation, percentage split). We obtained 0.86 of accuracy and 0.77 of F-measure averaged over K, with peaks of around 0.93 for both measures. Overall, these preliminary experiments provided results which have been recognized as somehow interesting by practitioners from the ENEL electricity power company and encouraged us to plug the methodology described in a more complex context of fraud detection.

7 Conclusion

In this paper we proposed DSA, a representation model to support accurate and fast similarity detection in time series. DSA is able to transform a time series into a compact yet feature-rich sequence by combining the notions of derivative estimation, segmentation and segment modeling. We experimentally evaluated DSA in clustering and classification frameworks, and compared it to the state-of-the-art similarity measures and dimensionality reduction methods.

Experiments conducted on various benchmark and real-world datasets have shown that performing dynamic time warping on DSA sequences leads to a good trade-off between effectiveness and efficiency in time series similarity detection.

We plan to test the applicability of our approach to different application contexts, including biomedical domains. Further aspects we would like to study concern the feasibility of extending DSA to multidimensional time series and the relation between denoising and differentiation of time series in specific domains from a more complex perspective of regularization functions.

Acknowledgments

We are grateful to the anonymous reviewers for their valuable suggestions which helped to improve the quality of this paper.

References

[1] D. J. Berndt and J. Clifford. Using Dynamic Time Warping To Find Patterns in Time Series. In *Proc. AAAI Workshop on Knowledge Discovery in Databases*, pages 359–370, 1994.

[2] E. H. Bristol. Swinging door trending: adaptive trending recording. In *Proc. ISA National Conf.*, pages 749—753, 1990.

[3] C. S. Burrus, R. A. Gopinath, and H. Guo. *Introduction to Wavelets and Wavelet Transforms: A Primer*. Prentice Hall, 1997.

[4] Y. Cai and R. Ng. Indexing Spatio-Temporal Trajectories with Chebyshev Polynomials. In *Proc. ACM SIGMOD Conf.*, pages 599–610, 2004.

[5] K. Chakrabarti, E. Keogh, S. Mehrotra, and M. Pazzani. Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases. *ACM Transactions on Database Systems*, 27(2):188–228, 2002.

[6] K. Chan and A. Fu. Efficient Time Series Matching by Wavelets. In *Proc. ICDE Conf.*, pages 126–133, 1999.

[7] L. Chen and R. Ng. On The Marriage of Lp-norms and Edit Distance. In *Proc. VLDB Conf.*, pages 792–803, 2004.

[8] L. Chen, M. T. Özsu, and V. Oria. Robust and Fast Similarity Search for Moving Object Trajectories. In *Proc. ACM SIGMOD Conf.*, pages 491–502, 2005.

[9] E. Diday. The dynamic clusters method in nonhierarchical clustering. *Journal on Computer and Information Science*, 2(1):61–88, 1973.

[10] S. Greco, M. Ruffolo, and A. Tagarelli. Effective and Efficient Similarity Search in Time Series. In *Proc. ACM CIKM Conf.*, pages 808–809, 2006.

[11] F. Gullo, G. Ponti, A. Tagarelli, G. Tradigo, and P. Veltri. MSPtool: A Versatile Tool for Mass Spectrometry Data Preprocessing. In *Proc. CBMS Conf.*, pages 209–214, 2008.

[12] S. A. Imtiaz, M. A. A. Shoukat Choudhury, and S. L. Shah. Building Multivariate Model from Compressed Data. *Industrial Engineering Chemistry Research and Development*, 46(2):481–491, 2007.

[13] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, 1988.

[14] K. V. Kanth, D. Agrawal, and A. Singh. Dimensionality Reduction for Similarity Searching in Dynamic Databases. In *Proc. ACM SIGMOD Conf.*, pages 166–176, 1998.

[15] E. Keogh. Exact Indexing of Dynamic Time Warping. In *Proc. VLDB Conf.*, pages 406–417, 2002.

[16] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. *Knowledge and Information Systems*, 3(3):263–286, 2001.

[17] E. Keogh and M. Pazzani. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In *Proc. KDD Conf.*, pages 239–241, 1998.

[18] E. Keogh and M. Pazzani. Scaling up Dynamic Time Warping for Datamining Applications. In *Proc. ACM KDD Conf.*, pages 285–289, 2000.

[19] E. Keogh and M. Pazzani. Derivative Dynamic Time Warping. In *Proc. SIAM Int. Conf. on Data Mining*, 2001.

[20] E. Keogh, X. Xi, L. Wei, and C. A. Ratanamahatana. The UCR Time Series Classification/Clustering Homepage. *http://www.cs.ucr.edu/∼eamonn/time_series_data/*.

[21] S. W. Kim, S. Park, and W. W. Chu. An Indexed-Based Approach for Similarity Search Supporting Time Warping in Large Sequence Databases. In *Proc. ICDE Conf.*, pages 607–614, 2001.

[22] F. Korn, H. V. Jagadish, and C. Faloutsos. Efficiently Supporting Ad Hoc Queries in Large Datasets of Time Sequences. In *Proc. ACM SIGMOD Conf.*, pages 289–300, 1997.

[23] T. Warren Liao. Clustering of Time Series Data—A Survey. *Pattern Recognition*, 38:1857–1874, 2005.

[24] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A Symbolic Representation of Time Series, with Implications for Streaming Algorithms. In *Proc. ACM SIGMOD Conf.*, pages 2–11, 2003.

[25] J. C. Mason and D. Handscomb. *Chebyshev Polynomials*. Chapman & Hall, 2003.

[26] T. M. Mitchell. *Machine Learning*. Computer Sciences Series, McGraw-Hill, 1997.

[27] J. S. Morris, K. R. Coombes, J. Koomen, K. A. Baggerly, and R. Kobayashi. Feature extraction and quantification for mass spectrometry in biomedical applications using the mean spectrum. *Bioinformatics*, 21(9):1764–1775, 2005.

[28] T. Pavlidis and S.L. Horowitz. Segmentation of Plane Curves. *IEEE Transactions on Computers*, 23(8):860–870, 1974.

[29] E. F. Petricoin 3rd, A. M. Ardekani, B. A. Hitt, P. Levine, V. A. Fusaro, and S. Steinberg. Use of proteomic patterns in serum to identify ovarian cancer. *Lancet*, 9306(359):572–577, 2002.

[30] L. Rabiner and B.-H. Juang. *Fundamentals of Speech Recognition*. Englewood Cliffs, N. J., 1993.

[31] D. Rafiei and A. Mendelzon. Similarity-based queries for time series data. In *Proc. ACM SIGMOD Conf.*, pages 13–25, 1997.

[32] D. Rafiei and A. Mendelzon. Efficient Retrieval of Similar Time Sequences Using DFT. In *Proc. FODO Conf.*, 1998.

[33] Y. Sakurai, M. Yoshikawa, and C. Faloutsos. FTW: Fast Similarity Search under the Time Warping Distance. In *Proc. ACM PODS Conf.*, pages 326–337, 2005.

[34] N. F. Thornhill, M. A. A. Shoukat Choudhury, and S. L. Shah. The impact of compression on data-driven process analyses. *Process Control*, 14(4):389–398, 2004.

[35] A. N. Tikhonov and V. Y. Arsenin. *Solution of Ill-posed Problems*. Winston & Sons, Washington D.C., 1977.

[36] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, 1979.

[37] M. Vlachos, D. Gunopulos, and G. Kollios. Discovering Similar Multidimensional Trajectories. In *Proc. ICDE Conf.*, pages 673–684, 2002.

[38] M. Wagner, D. Naik, and A. Pothen. Protocols for disease classification from mass spectrometry data. *Proteomics*, 3(9):1692–1698, 2003.

[39] Y. Wu, D. Agrawal, and A. Abbadi. A Comparison of DFT and DWT Based Similarity Search in Time-Series Databases. In *Proc. ACM CIKM Conf.*, pages 488–495, 2000.

[40] F. Xiaodong, C. Changling, L. Changling, and S. Huihe. An Improved Process Data Compression Algorithm. In *Proc. Intelligent Control and Automation Conf.*, pages 2190—2193, 2002.

[41] B. K. Yi and C. Faloutsos. Fast Time Sequence Indexing for Arbitrary Lp Norms. In *Proc. VLDB Conf.*, pages 385–394, 2000.

[42] B. K. Yi, H. V. Jagadish, and C. Faloutsos. Efficient Retrieval of Similar Time Sequences Under Time Warping. In *Proc. ICDE Conf.*, pages 201–208, 1998.

Appendix A: DDTW and DSA derivative estimation models

Approximating the derivative of a given series plays an essential role in the DDTW method as well as in our DSA. We have described both the DDTW and DSA derivative estimation models in Section 3.1 (Eq. 1 and Eq. 2). Here we present some experimental results which show a comparison of these two models concerning their *i)* performance in approximating real derivatives of standard functions and *ii)* impact on the performance of DSA and DDTW.

Evaluation of the approximation of real derivative functions. Let $f(t) : \Re \rightarrow \Re$ be a continuous function and $f'(t) : \Re \rightarrow \Re$ be its first derivative. Let $R = [t_1, \ldots, t_n] \in \Re^n$ denote a sequence of real values, over which we suppose to define a sequence T and the corresponding sequence T' of derivative values. Formally, let $T = [x_1, \ldots, x_n]$ and $T' = [x'_1, \ldots, x'_n]$, such that $x_h = f(t_h)$ and $x'_h = f'(t_h), \forall h \in [1..n]$. Given T, we also denote with $\hat{T}' = [\hat{x}'_1, \ldots, \hat{x}'_n]$ the derivative version of T which is obtained by a certain estimation model.

We compute the average approximation error of $\hat{T}'$ (i.e., the estimated derivative sequence) with respect to T' (i.e., the actual derivative sequence) as follows:

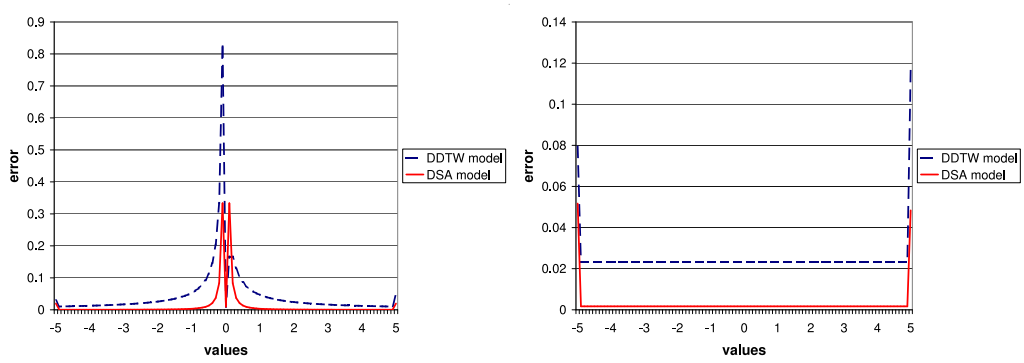
$$\mathcal{E}(\hat{T}', T') = \frac{1}{n} \sum_{h=1}^{n} \frac{|\hat{x}'_h - x'_h|}{|x'_h|}$$

Figure 5 shows a comparison between the DDTW and DSA derivative estimation models on four example functions, namely a cubic polynomial, an exponential function, a sine wave, and the Gaussian function. Table 7 reports on the average approximation errors (in percentage) obtained by using the two models on the selected functions. It can be noted that the DSA derivative estimation model produces an average error of approximation which is always lower than the error by the DDTW model.

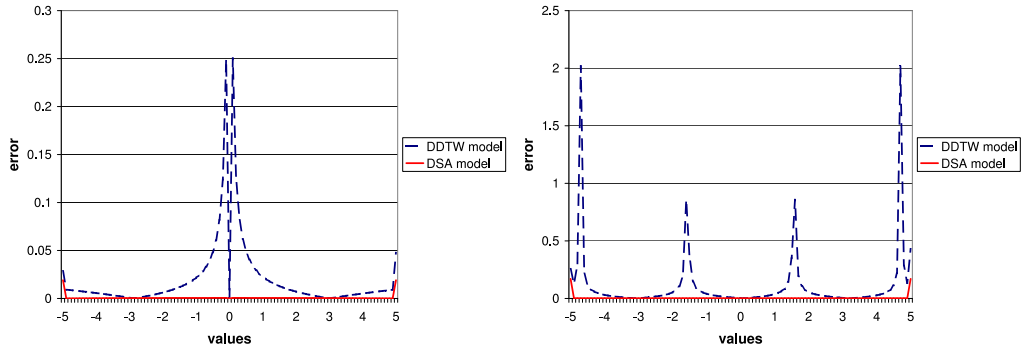
Table 7
Average approximation errors on derivative estimation. Each function is valued on 101 points over the range $[-5, +5]$.

function	*DDTW model*	*DSA model*
cubic	4.52%	1.12%
exponential	2.48%	0.26%
Gaussian	1.99%	0.08%
sine	11.48%	0.5%

Impact on the performance of DSA and DDTW. In the main experimental sections we have presented clustering results obtained on the various data sets by using DSA and DDTW. Since both methods are characterized



(a) cubic (b) exponential



(c) Gaussian (d) sin wave

Fig. 5. Approximation errors on derivative estimation: DDTW model vs. DSA model

by a distinct model of derivative estimation, it was also interesting to gain an insight into the impact of this model on the performance of DSA and DDTW. For this purpose, we exchanged the respective models of derivative estimation in DSA and DDTW and then repeated the relative experimental evaluation in clustering frameworks.

Table 8 shows the clustering results obtained by K-Means and UPGMA when DDTW was equipped with the DSA derivative estimation model, and compares these results with those previously reported in Table 2 and Table 3. The modified version of DDTW led to better performances than the original DDTW method in most cases, in particular Mixed-BagShapes (4% by UPGMA and 2% by K-Means), Twopat (3% by UPGMA and 1% by K-Means), and ControlChart (2%).

Analogously, Table 9 reports the clustering results when DSA was equipped with the DDTW derivative estimation model, and compares them with the original performances of DSA. Again, the DSA derivative estimation model prevailed against the DDTW one in most cases—OvarianCancer (5% by UPGMA and 4% by K-Means), CBF and GunX (4% by K-Means and 2% by K-Means), Mixed-BagShapes (3% by K-Means and 2% by UPGMA), and Twopat (2%).

Table 8
DDTW-based clustering results by varying the derivative estimation model

clustering algorithm	derivative estimation	GunX	Tracedata	ControlChart	CBF	Twopat	Mixed-Bag Shapes	Ovarian Cancer
K-Means	DSA (Eq. 2)	*.90*	*1*	*.91*	*.95*	*.96*	*.78*	*.63*
	DDTW (Eq. 1)	.89	1	.89	.96	.95	.76	.62
UPGMA	DSA (Eq. 2)	*.72*	*.78*	*.56*	*.48*	*.67*	*.45*	*.64*
	DDTW (Eq. 1)	.72	.76	.54	.49	.64	.41	.63

Table 9
DSA-based clustering results by varying the derivative estimation model

clustering algorithm	derivative estimation	GunX	Tracedata	ControlChart	CBF	Twopat	Mixed-Bag Shapes	Ovarian Cancer
K-Means	DSA (Eq. 2)	*.92*	*1*	*.90*	*.96*	*.97*	*.78*	*.75*
	DDTW (Eq. 1)	.90	1	.91	.92	.95	.75	.71
UPGMA	DSA (Eq. 2)	*.73*	*.82*	*.54*	*.60*	*.67*	*.51*	*.73*
	DDTW (Eq. 1)	.69	.80	.56	.62	.65	.49	.68

The above results led us to conclude that while the DSA derivative estimation model can enhance the DDTW method in practice; conversely, the DDTW derivative estimation model does not bring any beneficial effect to (in general, it may negatively affect) the performance of the DSA method.

Appendix B: Impact of preprocessing on similarity detection

As we have discussed in Section 4.3, smoothing was performed prior to the mining tasks in order to handle noise in the raw data, regardless of the particular representation method or distance measure used. Smoothing turned out to be useful for all the methods on every dataset—except for the OvarianCancer case. The intuition that skipping the smoothing stage would cause a decrease in performing similarity detection was supported by experimental evidence when we tried to directly classify the original (i.e., non-smoothed) data. Indeed, as shown for some prominent methods in Table 10, the decrease would be significantly high in most datasets, with peaks of around 50% on ControlChart, CBF, and Twopat.

Another important remark is that the relative performances of most of the various methods (including our DSA) do not vary substantially whether or not smoothing is performed. This indicates that the representation model and similarity/distance measure play a more important role than the preprocessing operations in determining the best approach(es) to similarity detection in time series.

A special remark should also be made on the OvarianCancer dataset which is huge-dimensional and largely affected by noisy factors, like most of mass spectra datasets (cf. Section 5.1). On this dataset, DSA performed far better

Table 10
K-Means clustering performance reduction in case of no smoothing

	GunX	Trace data	Control Chart	CBF	Twopat	Mixed-BagSh.
FTW	-21%	-16%	-41%	-29%	-23%	–
DTW	-16%	-2%	-46%	-48%	-53%	-4%
DDTW	-19%	–	-47%	-57%	-50%	–
DWT	-6%	-17%	-37%	-35%	-15%	–
SD	-5%	-19%	-45%	-46%	-47%	–
PLA	-14%	-1%	-47%	-46%	-35%	-3%
PAA	-7%	-2%	-46%	-46%	-30%	-5%
SAX	-22%	-2%	-43%	-47%	-25%	-4%
APCA	-18%	-5%	-48%	-42%	-46%	-4%
DSA	-19%	–	-48%	-54%	-51%	–

than DDTW, precisely +13% by K-Means, +10% by UPGMA and 5% by K-NN, in terms of F-measure (cf. Tables 2–4).

Table 11 summarizes the best preprocessing setups for DSA and the other methods on the various datasets, using the K-Means algorithm; we left the best setups for UPGMA and K-NN out of the presentation, since they resulted fairly similar to those obtained by K-Means in most datasets. In the table, term MA (resp., EXP) stands for moving average (resp., exponential smoothing) and is followed by the value set for δ (resp., ω) and the number of iterations.

Appendix C: Dynamic kernel clustering

We briefly present a preliminary investigation on the use of a more general notion of cluster prototype in time series clustering. For this purpose, we considered a *dynamic kernel clustering* algorithm. Dynamic kernel clustering [9] falls into the family of partitional clustering, and represents a generalization of centroid-based and medoid-based partitional clustering algorithms. Indeed, a major difference between dynamic kernel clustering and more popular algorithms (such as, e.g., K-Means and K-Medoids) consists in the notion of cluster prototype, which can be a centroid, a medoid, or even a whole set of points or objects. To conduct an experimental evaluation of dynamic kernel clustering in our context, we chose the following notion of cluster prototype based on a set of medoids: given any cluster C in the current partition, the

Table 11
Summary of the preprocessing setups providing the best clustering results by K-Means

	GunX	Trace data	Control Chart	CBF	Twopat	Mixed-BagSh.	Ovarian Cancer
LCSS	MA δ=9 it=3	MA δ=5 it=3	EXP ω=0.3 it=1	No smooth.	EXP ω=0.1 it=4	No smooth.	No smooth.
EDR	No smooth.	EXP ω=0.3 it=5	EXP ω=0.7 it=1	EXP ω=0.7 it=1	No smooth.	EXP ω=0.1 it=3	No smooth.
ERP	EXP ω=0.1 it=5	EXP ω=0.1 it=1	EXP ω=0.7 it=3	EXP ω=0.1 it=5	MA δ=5 it=3	EXP ω=0.5 it=5	No smooth.
FTW	EXP ω=0.3 it=3	No smooth.	EXP ω=0.7 it=3	EXP ω=0.1 it=3	EXP ω=0.3 it=3	EXP ω=0.5 it=2	No smooth.
DTW	EXP ω=0.1 it=3	No smooth.	EXP ω=0.9 it=1	No smooth.	EXP ω=0.9 it=5	EXP ω=0.1 it=5	No smooth.
DDTW	EXP ω=0.9 it=3	EXP ω=0.1 it=1	EXP ω=0.3 it=5	EXP ω=0.5 it=5	EXP ω=0.1 it=1	EXP ω=0.1 it=1	No smooth.
DFT	EXP ω=0.2 it=4	EXP ω=0.1 it=3	MA δ=9 it=2	EXP ω=0.1 it=1	EXP ω=0.2 it=4	MA δ=5 it=1	No smooth.
DWT	EXP ω=0.1 it=2	EXP ω=0.6 it=2	EXP ω=0.3 it=1	EXP ω=0.1 it=3	EXP ω=0.6 it=3	EXP ω=0.1 it=2	No smooth.
CHEBY	EXP ω=0.1 it=3	EXP ω=0.9 it=5	EXP ω=0.7 it=5	EXP ω=0.5 it=1	MA δ=9 it=3	EXP ω=0.9 it=3	No smooth.
SD	EXP ω=0.7 it=5	MA δ=9 it=5	EXP ω=0.6 it=4	EXP ω=0.3 it=4	EXP ω=0.3 it=5	EXP ω=0.9 it=4	No smooth.
PLA	EXP ω=0.1 it=1	MA δ=5 it=1	EXP ω=0.5 it=1	EXP ω=0.5 it=1	EXP ω=0.3 it=1	EXP ω=0.4 it=1	No smooth.
PAA	EXP ω=0.1 it=1	EXP ω=0.5 it=1	EXP ω=0.7 it=3	EXP ω=0.7 it=1	EXP ω=0.1 it=1	EXP ω=0.1 it=1	No smooth.
SAX	EXP ω=0.3 it=3	EXP ω=1 it=1	EXP ω=0.4 it=3	EXP ω=0.5 it=1	EXP ω=0.1 it=1	EXP ω=0.5 it=3	No smooth.
APCA	EXP ω=0.1 it=3	EXP ω=0.7 it=5	EXP ω=0.7 it=3	No smooth.	EXP ω=0.5 it=1	EXP ω=0.1 it=3	No smooth.
DSA	EXP ω=0.9 it=3	EXP ω=0.1 it=1	EXP ω=0.2 it=2	EXP ω=0.4 it=3	EXP ω=0.2 it=2	EXP ω=0.1 it=2	No smooth.

prototype (or kernel) of C is the set of the H objects in C such that the sum of the distances of all the objects in C to such H objects is minimized.

Table 12 summarizes the (average) F-measure scores obtained by dynamic kernel clustering and compares them to the corresponding results obtained by K-Means on the various datasets. For each dataset and method, the table contains three values separated by the symbol '/':

- the first value refers to dynamic kernel clustering with $H = 2$;
- the second value refers to dynamic kernel clustering with $H = \sqrt{|C|}$, for each cluster C;
- the third value refers to K-Means (which is extracted from Table 2).

In Table 12 we can observe that there were no significant differences between dynamic kernel with $H = 2$ and K-Means. In most cases, the obtained F-measure scores were identical or very close ($\pm 1\%$ on average) to those obtained by K-Means; in particular, as far as our DSA, the maximum improvement of quality with respect to K-Means occurred in Mixed-BagShapes, which was about 2%.

Setting the kernel with $H = \lceil \sqrt{|C|} \rceil$, for each cluster C, led to further improvements of the clustering quality with respect to K-Means in most cases. On average, the observed improvements were about 2-3%, with maximum increases of the quality up to 4-5% in Mixed-BagShapes and OvarianCancer. Indeed, as we expected, the dynamic kernel clustering revealed to be particularly advantageous for those datasets where the separation between the classes is less sharp (e.g., Mixed-BagShapes and OvarianCancer). In this case, using a set of medoids as cluster prototype was effective to better represent the clusters and improve the clustering quality.

However, the improvements in terms of clustering quality provided by the dynamic kernel algorithm appeared to be not enough large to justify a runtime behavior that is much more costly than K-Means. Moreover, the higher computational complexity of the dynamic kernel algorithm represents an efficiency issue that may limit the applicability to large datasets and real use cases.

Table 12
Summary of average quality results (F-measure) by dynamic kernel clustering and comparison with K-Means clustering results

	GunX	Trace data	Control Chart	CBF	Twopat	Mixed BagSh.	Ovarian Cancer
LCSS	.58/.62/.59	.32/.32/.30	.50/.52/.50	.79/.79./.79	.37/.37/.36	.33/.35/.32	.35/.39/.34
EDR	.53/.53/.54	.72/.74/.74	.90/.91/.88	.86/.86/.86	.44/.44/.42	.69/.73/.70	.63/.63/.58
ERP	.73/.73/.72	.64/.65/.62	.77/.77/.76	.59/.59/.58	.38/.41/.39	.49/.53/.48	.34/.37/.34
DTW	.67/.68/.66	.77/.81/.78	.86/.86/.87	.89/.90/.89	.95/.95/.95	.78/.80/.77	.58/.63/.60
DDTW	.90/.91/.89	.99/1/1	.89/.89/.89	.96/.97/.97	.94/.96/.95	.78/.78/.76	.62/.66/.62
FTW	.75/.74/.74	.91/.92/.90	.80/.80/.81	.67/.67/.67	.57/.56/.55	.75/.75/.73	.57/.62/.58
L_2 on DFT	.65/.64/.63	.78/.78/.77	.78/.80/.78	.66/.69/.67	.42/.43/.39	.70/.71/.70	.41/.41/.36
L_2 on DWT	.59/.60/.61	.69/.69/.67	.76/.79/.76	.76/.76/.74	.37/.37/.36	.70/.71/.68	.40/.40/.36
L_2 on CHEBY	.56/.58/.57	.72/.74/.72	.70/.72/.70	.68/.70/.69	.40/.42/.38	.73/.75/.72	.36/.38/.34
DTW on SD	.68/.69/.67	.95/.96/.95	.85/.87/.85	.89/.89/.87	.87/.91/.89	.77/.79/.76	.69/.71/.69
DTW on PLA	.74/.74/.73	.78/.79/.77	.89/.91/.89	.88/.87/.87	.75/.76/.75	.77/.77/.74	.62/.65/.60
DTW on PAA	.69/.69/.68	.80/.80/.78	.87/.88/.87	.88/.89/.86	.72/.75/.73	.76/.78/.75	.60/.61/.59
DTW on SAX	.73/.75/.73	.76/.77/.77	.83/.84/.83	.87/.87/.87	.70/.70/.69	.73/.74/.71	.61/.61/.58
DTW on APCA	.76/.78/.77	.80/.80/.81	.88/.88/.89	.84/.84/.83	.90/.92/.91	.78/.79/.74	.57/.59/.55
DTW on DSA	.91/.94/.92	1/1/1	.89/.91/.90	.95/.95/.96	.97/.98/.97	.80/.83/.78	.76/.79/.75