Data Min Knowl Disc (2013) 26:310–331
DOI 10.1007/s10618-012-0251-4

Using derivatives in time series classification

Tomasz Górecki · Maciej Łuczak

Received: 3 April 2011 / Accepted: 16 January 2012 / Published online: 1 February 2012

Abstract Over recent years the popularity of time series has soared. Given the widespread use of modern information technology, a large number of time series may be collected during business, medical or biological operations, for example. As a consequence there has been a dramatic increase in the amount of interest in querying and mining such data, which in turn has resulted in a large number of works introducing new methodologies for indexing, classification, clustering and approximation of time series. In particular, many new distance measures between time series have been introduced. In this paper, we propose a new distance function based on a derivative. In contrast to well-known measures from the literature, our approach considers the general shape of a time series rather than point-to-point function comparison. The new distance is used in classification with the nearest neighbor rule. In order to provide a comprehensive comparison, we conducted a set of experiments, testing effectiveness on 20 time series datasets from a wide variety of application domains. Our experiments show that our method provides a higher quality of classification on most of the examined datasets.

Keywords Dynamic Time Warping · Derivative Dynamic Time Warping · Data mining · Time series

Responsible editor: Eamonn Keogh.

T. Górecki (✉)
Faculty of Mathematics and Computer Science, Adam Mickiewicz University,
Umultowska 87, 61-614 Poznań, Poland
e-mail: tomasz.gorecki@amu.edu.pl

M. Łuczak
Department of Civil and Environmental Engineering, Koszalin University of Technology,
Śniadeckich 2, 75-453 Koszalin, Poland
e-mail: mluczak@wbiis.tu.koszalin.pl

1 Introduction

Time-series classification has been studied extensively by machine learning and data mining communities, resulting in a variety of different approaches, ranging from neural (Petridis and Kehagias 1997) and Bayesian networks (Pavlovic et al. 1999) through HMM-AR models (Penny and Roberts 1999) to genetic algorithms and support vector machines (Eads et al. 2002). Similarly we can find many distance measures for similarity of time series data (a very good overview can be found in Ding et al. 2008). Nevertheless, the simple method combining the nearest neighbor (1NN) classifier and some form of Dynamic Time Warping (DTW) distance has been shown to be one of the best performing time-series classification techniques (Ding et al. 2008).

DTW is a classical distance measure well suited to the task of comparing time series (Berndt and Clifford 1994). It differs from Euclidean distance (ED) by allowing the vector components being compared to "drift" from exactly corresponding positions. It is an algorithm for measuring similarity between two sequences (e.g. time series) which may vary in time or speed. The sequences are "warped" non-linearly in the time dimension to determine a measure of their similarity independent of certain non-linear variations in the time dimension. In our work we used the original DTW, which is parameter free. The most straightforward similarity measure for time series is the ED (metric) and its variants based on the common L_p norms. In this work we used the L_2 norm. ED has several advantages (Keogh and Kasetty 2003). The complexity of evaluating this measure is linear, and it is easy to implement and indexable with any access method, and in addition is parameter free. The L_2 norm is competitive with other more complex approaches, especially if the size of the training set is relatively large (in Ding et al. 2008 and elsewhere it has been empirically shown that simple ED is competitive with or superior to many of the complex distance measures, and has the very important triangular inequality property). However, since the mapping between the points of two time series is fixed, this distance is very sensitive to some distortion in the data (offset and scale (amplitude)), and is unable to handle local time shifting. Two subsequences of a time series may be very similar but at different offsets and/or scales, and thus report a larger distance than warranted.

Effectiveness of the nearest neighbor method depends on the distance measure (metric, similarity measure) used to compare objects in the classification process. At present, in the domain of time series classification, distance functions are used which mostly do point-to-point comparison in a time series. The measures often reduce such distortions as occur if two time series do not have the same length or are locally out of phase, etc. Nevertheless the perfect case is if the compared time series are similar as functions—if the point values are identical.

It seems that in the classification domain there could be objects for which function value comparison is not sufficient. There could be cases where assignment to one of the classes depends on the general shape of objects (signals, functions) rather than on strict function value comparison. Especially for time series it seems that some variability in the "time" domain could have a great influence on the classification process.

In mathematics, an object associated with a function that responds to its variability in "time" is the derivative of the function. The function's derivative determines areas where the function is constant, increases or decreases, and the intensity of the changes.

The derivative determines the general shape of the function rather than the value of the function at an actual point. The derivative shows what happens in the neighborhood of the point. In the case of time series, it means that the function derivative considers the behavior of a time series before and after some point in "time".

It seems that especially in the case of time series such an approach to classification can be very effective. We cannot expect that it is sufficient to compare time series only as their derivatives. Although such datasets exist (e.g. the data set from Sect. 3.1), in most cases the classification result more or less depends on function value comparison. It seems that the best approach is to create a method which considers both function values of time series (point to point comparison) and values of the derivative of the function (general shape comparison). The intensity of the influence of these two approaches should be parameterized. Then we can expect that for different datasets of time series the method will select the appropriate intensity of these two kinds of comparisons and give the best classification results.

In this paper we construct a distance measure that considers the two above-mentioned approaches to time series classification. Thanks to this we are able to deal with situations where examinated sequences are not different enough. For any fixed distance function there is formed a new parameterized family of distance measures, where the fixed distance measure is used both to compute distances of time series (function values) and their variability in "time" (distances of their derivatives). The new distance functions so constructed are used in the nearest neighbor classification method.

For different datasets the parameters are chosen in the learning phase by a cross-validation (leave-one-out) method. The distance measure is then used for classification of a test data set.

The remainder of the paper is organized as follows. We first give an overview of the methods on which the creation of our new distance measure was based (Sect. 2). In the same section we explain what is new in our approach. In Sect. 3 we review the concept of time series. At the end of that section we introduce our distance based on derivatives. In Sect. 4 we describe how to optimize the calculation. The datasets used are described at the beginning of Sect. 5. Later in that section we describe the experimental setup. The results of the research are illustrated with graphs showing the differences between the classifiers in the same section. Section 5 also contains the results of our experiments on the described real datasets. Section 6 contains the results of comparison of our method with related approaches, with statistical analysis of the differences in classification accuracy. We conclude in Sect. 7 with discussion of possible future extensions of the work.

2 Related works and new contribution

The use of derivatives in time series classification is not a novelty. Their use with DTW was proposed by Keogh and Pazzani (2001). However they used only the distance between the derivatives, rather than the point-to-point distance between the time series. They called their method Derivative Dynamic Time Warping (DDTW). They tried to remove a major weakness of DTW. DTW tends to produce "singularities", i.e., alignments of a single point of a series with multiple points of another series.

Their proposal reduces the singularity phenomenon. 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. Keogh and Pazzani (2001) performed experiments only on three datasets, and therefore it is hard to say that the efficiency of the method was unequivocally confirmed. They did not test it in the context of the classification of time series. Subsequent works, however, used DDTW and showed its effectiveness in practice (recently Luan et al. 2010; Mokhtar et al. 2010). On the basis of this method Kulbacki et al. 2002 created a measure which took into account the point-to-point distance between time series. The measure of the distance between time series was the product of the ED between two time series and the square of the difference of the estimated derivatives. This approach also does not include the possibility of controlling the extent to which the derivative affects the result of classification. The authors named their method Value-Derivative Dynamic Time Warping (VDDTW). The authors discussed the application of this method to motion classification, but did not perform any simulations. Benedikt et al. (2008, 2010) have gone even further. They introduced a weighted sum, in which they also took into consideration second derivative. They called their method Weighted Derivative Dynamic Time Warping (WDTW). However they did not propose any methods for the choice of parameters. They showed only a certain system of parameters, which were shown to be effective on the examined example (face recognition).

The methods described above either do not have any parameters or the parameters are integrated into the internal distance function d used in DTW (d is a distance between two points of time series). In contrast, in our method the parameters are outside the base distance functions dist (dist is a distance between time series). Thanks to this approach, tuning of parameters is computationally simpler. Our method can use any distance measure as the distance component, in particular DTW and ED. We can also use a different point-to-point distance and another for derivatives. The parametric approach makes it possible to adapt to the data set, but without overtraining. As was demonstrated by the simulations, on multiple datasets the approach with only the derivative or only point-to-point distance does not work—an intermediate parameter is needed. In spite of the need to search for a parameter, the computational complexity is actually like for a method without a parameter. In fact, particularly for DTW, there is no deterioration in performance on any data set. The method works well on a wide range of datasets (we tested many different datasets). For all these reasons in combination, our method seems to be (especially with DTW) a universal (and very good) method for the classification of the time series, able to identify for which of the data sets the impact of the derivative is helpful and to what extent. At the same time our method preserves the computational complexity of the output method—there is no computational overhead. The proposed distance inherits the lower bounds defined for the original distance as well as the important triangular inequality property of the original distance. Therefore, the proposed distance would also be indexed by a large number of techniques that rely on metric properties.

Additionally, all related previous approaches were applied to one of a few domains. We conducted a comprehensive study on a much larger scale, which provided general knowledge that the use of derivatives can generally produce accuracy improvements for ED and DTW.

3 Methods

A time series is a sequence of observations which are ordered in time or space (Box et al. 2008). Time is called the independent variable.

For simplicity and without any loss of generality, we assume that time is discrete. Formally, a time series data is defined as a sequence of pairs $T = [(\boldsymbol{x}_1, t_1), (\boldsymbol{x}_2, t_2), \ldots, (\boldsymbol{x}_n, t_n)]$ $(t_1 < t_2 < \cdots < t_n)$ where each $\boldsymbol{x}_i$ is a data point in d-dimensional feature space, and each t_i is the time stamp at which $\boldsymbol{x}_i$ occurs. If the sampling rates of two time series are the same, we can omit the time stamps and consider them as a sequences of d-dimensional data points. Such a representation is called the raw representation of the time series. The number of data points n in given time series is called its length. In the rest of this article we will only use one-dimensional time series, which are conveniently represented in the form $x(i)$, $i = 1, 2, \ldots, n$.

3.1 An illustrative example

Let us consider a data set of time series consisting of three classes of objects. For every four random numbers $x_1, x_2, y_1, y_2 \in [0, 1]$ we can construct three kinds of signals (Fig. 1).

The first class consists of rectangular signals, the second of increasing signals, and the third of decreasing signals. Assignment to one of the classes does not depend on a function value comparison, but on the general variability of the signals—their shape.

For example, in Fig. 2 (left) there are two signals with the same class of objects—rectangular signals. In a distance measure that compares only values of functions, the Euclidean metric for example, these two signals are very far from each other. Nevertheless, in the same metric but comparing not the functions but their derivatives, the signals are identical—the function derivatives are constant functions with a value of 0 on the “period” [0, 1]. But for signals from two different classes, for a rectangular and an increasing one, we can see that the derivative separates them accurately (Fig. 2 (right)).

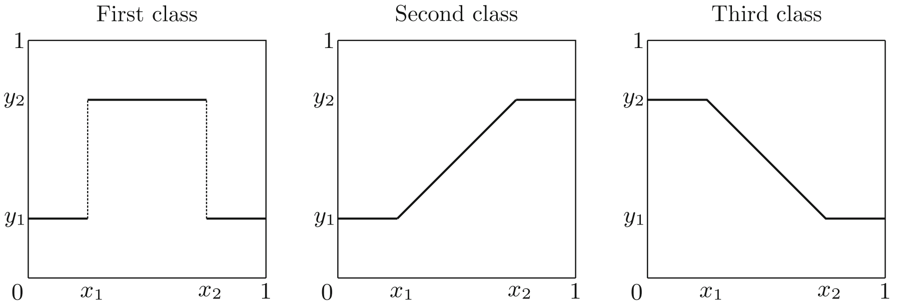

Fig. 1 Example data set—three classes of signals

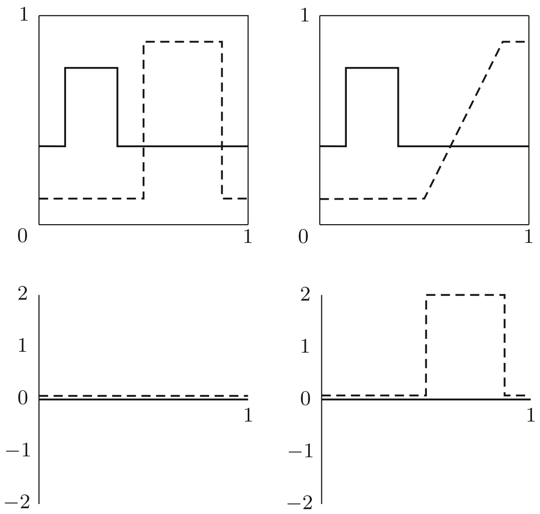

Fig. 2 Signals from the example data set (*top*) and their derivative functions (*bottom*)

3.2 Dynamic Time Warping distance

Suppose we have two time series

$$x = \{x(i) \colon i = 1, 2, .., n\} \text{ and } y = \{y(j) \colon j = 1, 2, .., m\}$$

of length n and m respectively. To align two sequences using DTW we construct an $n \times m$ matrix where the element (i, j) of the matrix contains the distance $d(x(i), y(j))$ between the two points $x(i)$ and $y(j)$. In this paper, we will call the distance d the internal distance of DTW. Each matrix element (i, j) corresponds to the alignment between the points $x(i)$ and $y(j)$. A warping path W is a contiguous set of matrix elements that defines a mapping between x and y. The kth element of W is defined as $w_k = (i, j)_k$ so we have

$$W = w_1, w_2, \ldots, w_k, \ldots, w_K, \quad \max(n, m) \le K \le n + m - 1.$$

The warping path is subject to three constraints:

- $w_1 = (1, 1)$ and $w_K = (n, m)$ (boundary conditions);
- For $w_k = (a, b)$ and $w_{k-1} = (a', b')$, $a - a' \le 1$ and $b - b' \le 1$ (continuity);
- For $w_k = (a, b)$ and $w_{k-1} = (a', b')$, $a - a' \ge 0$ and $b - b' \ge 0$ (monotonicity).

As DTW distance we take the path which minimizes the warping cost:

$$\mathrm{DTW}(x, y) = \min_W \left\{ \sum_{k=1}^{K} w_k \right\}.$$

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

$$\gamma(i, j) = d(x(i), y(j)) + \min\{\gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1)\}.$$

3.3 Distance based on derivative

If dist is a distance measure for two time series f and g, a new distance measure $\widehat{\text{dist}}_{ab}$ is defined by

$$\widehat{\text{dist}}_{ab}(f, g) := a\,\text{dist}(f, g) + b\,\text{dist}(f', g'), \tag{1}$$

where f', g' are discrete derivatives of f, g, and $a, b \in [0, 1]$ are parameters. The discrete derivative of a time series f with length n is defined by

$$f'(i) = f(i) - f(i-1), \quad i = 2, 3, \ldots, n \tag{2}$$

where f' is a time series with the length $n - 1$.

Each distance function dist which we use in our method we will call the base distance. In this work, we use two base distances: ED and DTW.

The distance function $\widehat{\text{dist}}_{ab}$ is used with the 1NN method in the classification process. The parameters a, b are chosen in the learning phase. In this paper we use the cross-validation (leave-one-out) method on the learning data set in the process of parameter tuning.

Note that there is nothing in the way that the distance between time series and the derivatives in (1) differed. For example, DTW between the derivatives and ED between the time series. However in this paper we always take the same distance function. For an arbitrary distance function dist, we will denote the new distance measure by DD_{dist} (derivative distance), for example DD_{DTW} or DD_{ED}.

We chose the simplest and the most common definition of the discrete derivative of (discrete) function. In papers concerning time series classification using derivatives we can find several other definitions. For example: $f'(i) = \frac{f(i)-f(i-1)+\frac{f(i+1)-f(i-1)}{2}}{2}$ in Keogh and Pazzani (2001) or $f'(i) = \frac{f(i+1)-f(i-1)}{2}$ in Gullo et al. (2009). In the second paper we can read some arguments that proposed there two-point derivative estimator can be in some cases better than the first three-point one. We examined a few different definitions in our method and some other distances using derivative and results were ambiguous (see Appendix).

Following all these suggestions and seeking the simplest method we chose the formula (2). If the same length of the derivative and the function is needed (for example in other distance functions using derivative compared in Sect. 6) the assumption $f'(1) = f'(2)$ is made.

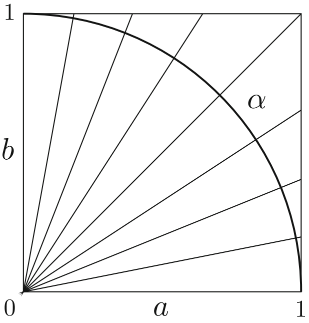

Fig. 3 Dependence of parameters a, b and α

4 Optimization

4.1 Parameter dimension

We do not have to check all values of $a, b \in [0, 1]$. If $a_1 = ca_2$ and $b_1 = cb_2$, where $c > 0$ is a constant (i.e. points (a_1, b_1), (a_2, b_2) are linearly dependent), we have

$$\widehat{\mathrm{dist}}_{a_1b_1}(f_1, g_1) \overset{=}{\underset{>}{<}} \widehat{\mathrm{dist}}_{a_1b_1}(f_2, g_2) \iff \widehat{\mathrm{dist}}_{a_2b_2}(f_1, g_1) \overset{=}{\underset{>}{<}} \widehat{\mathrm{dist}}_{a_2b_2}(f_2, g_2)$$

so, we can choose points (a, b) on any continuous line between the points $(0, 1)$ and $(1, 0)$. For example, it can be a straight line or a quarter of a circle:

$$\begin{array}{l} a = (1-\alpha), \\ b = \alpha, \end{array} \quad \alpha \in [0, 1]; \quad \begin{array}{l} a = \cos\alpha, \\ b = \sin\alpha, \end{array} \quad \alpha \in \left[0, \tfrac{\pi}{2}\right].$$

In the second case, we have equal distances between parameters (Fig. 3), so we choose that parameterization as more appropriate for a research work. However, especially if the subset of the parameters α is dense enough, the choice of parameterization should not be critical. In the next part of the paper we will use one parameter α instead of a, b.

4.2 Parameter tuning

In the training phase we have to tune the parameter α. Let $A \subset \left[0, \frac{\pi}{2}\right]$ be a finite subset of k parameters. We have to compute the cross-validation (leave-one-out) error rate on the learning data set for every $\alpha \in A$ and choose the parameter for the smallest value of the error (1NN method). However, we can do this in several ways, differing in computational and memory complexity.

1. For every parameter $\alpha \in A$ we do cross-validation on a training data set. If the training set consists of n elements, then for every α we have to compute $n(n-1)$ values of distance measure dist for a function and the same number for its derivative. Thus the method has a time complexity of $O(k\,n^2)$.
   This is the simplest but the most time-consuming method. If we know nothing more than the number of distance function computations, we will have to follow this approach. However, the structure of the new distance measure $\widehat{\mathrm{dist}}$ allows some optimizations.

2. Note that for every parameter α (all parameters a, b in the distance function $\widehat{\text{dist}}_{ab}$ (Eq. 1)) the base distance measures dist do not depend on these parameters. In a cross-validation process for every parameter the same distance function values are computed. We can avoid this by computing the distance values once and keeping them in memory. Before the cross-validation phase, we can compute distance matrices for the distance measure dist (for all pairs of training data set elements) for a function and its derivative. Then, using these matrices, for every parameter we compute the distance measure $\widehat{\text{dist}}$. This way, the calculation has a time complexity of $O(n^2)$. A disadvantage of this method is that we have to reserve a computer memory for distance matrices with complexity of $O(n^2)$.
3. In the two above procedures, the tuning of the parameter is as follows. We fix a parameter α and then (by cross-validation) for every element of a training data set we calculate an appropriate number $(n-1)$ of distance functions $\widehat{\text{dist}}$. We repeat this for every parameter α. However, we can proceed in the opposite direction.

   First, we fix one element e of the training data set. Then we take an element $e_1 \neq e$ and for every parameter α we compute the distance function $d = \widehat{\text{dist}}_{ab}(e_1, e)$ and put its value in a vector D (with k elements). Note that we need to compute only two distance measures dist (for the function and its derivative). Then for the next element $e_2 \neq e$ from the training set, we repeat the procedure and obtain a new distance vector d_2. Now we compare the positions of the vectors D and d_2. In every position of vector D we put the smaller one. We create a vector L (with k elements) in whose positions we put labels of elements e_1, e_2 corresponding to values of the vector D. We repeat the procedure for the next elements of the training set ($e_i \neq e, i = 3, 4, \ldots, n$). As a result we obtain vector L with labels of nearest neighbors of the element e for every parameter α. This is one step of the cross-validation process. We performed classification of element e for all α parameters. Note that we computed the distance function dist only $2(n-1)$ times. Repeating the procedure for all elements from the training data set, we obtain the cross-validation error rate for all parameters. The code of the algorithm is presented in Sect. 5.1.

   The above algorithm has all the advantages and none of the disadvantages of the previous two methods. The number of computations of distance functions dist has a complexity of $O(n^2)$ and we do not have to keep distance matrices in memory. Because the most time-consuming part of the algorithm is usually calculations of the distance functions dist, the computation time depends to a small degree on the number of parameters (especially for large values of n). Thanks to this fact, we can choose a large subset of parameters in the cross-validation process without increasing the computation time of the parameter tuning phase.

4.3 Lower bound and triangular inequality

For many distance measures it is possible to find their lower bound (Keogh 2002). Then the lower bound can be used in the nearest neighbor method to speed up computations. We can also find a lower bound of our new distance measure. If low is a lower bound of a distance dist, then

$$\widehat{\mathrm{low}}_{ab}(f, g) := a\,\mathrm{low}(f, g) + b\,\mathrm{low}(f', g')$$

is a lower bound of the distance $\widehat{\mathrm{dist}}_{ab}$ (see Appendix for proof).

If the base distance dist is a metric, then the new distance $\widehat{\mathrm{dist}}$ is also a metric. If dist is not a metric but obeys the triangular inequality, then the distance $\widehat{\mathrm{dist}}$ obeys the triangular inequality as well:

$$\widehat{\mathrm{dist}}_{ab}(f, g) \leq \widehat{\mathrm{dist}}_{ab}(f, h) + \widehat{\mathrm{dist}}_{ab}(h, g)$$

(see Appendix for proof).

5 Experimental results

5.1 Experimental setup

We performed experiments on 20 data sets. Information on the data sets used is presented in Table 1.

The data sets originate from the UCR Time Series Classification/Clustering Homepage (Keogh et al. 2006).

Table 1 Summary of data sets

Data set	Number of classes	Size of training set	Size of testing set	Time series length
50 words	50	450	455	270
Adiac	37	390	391	176
Beef	5	30	30	470
CBF	3	30	900	128
Coffee	2	28	28	286
ECG	2	100	100	96
Face (all)	14	560	1,690	131
Face (four)	4	24	88	350
Fish	7	175	175	463
Gun-point	2	50	150	150
Lightning-2	2	60	61	637
Lightning-7	7	70	73	319
Olive oil	4	30	30	570
OSU leaf	6	200	242	427
Swedish leaf	15	500	625	128
Synthetic control	6	300	300	60
Trace	4	100	100	275
Two patterns	4	1,000	4,000	128
Wafer	2	1,000	6,174	152
Yoga	2	300	3,000	426

We chose two distance measures to use with the new distance function. Note, however, that our method can work with any distance measure. The ED is the most straightforward similarity measure for time series, and DTW is one of the most effective distance functions for time series. Thus we have two similarity measures denoted by DD_{ED} and DD_{DTW}. For each data set we calculated the classification error rate on a test subset (to learn the model we used a training subset, leave-one-out, 1NN method). We found all parameters using the training subset. An appropriate distribution of the training and test sets was proposed by the authors of the repository (each data set is divided into a training and testing subset).

In the case of the data set in Sect. 3.1 we generated 200 observations from each class separately for the training and testing subset. In total we obtained 600 observations in each subset. The length of each time series was 100. Points x_1, x_2 were drawn from the discrete uniform distribution $U[1, 100]$, and y_1, y_2 from the continuous uniform distribution $U(0, 1)$.

We use the cross-validation (leave-one-out) method to find the best parameter α in our classifier. If the minimal error rate is the same for more than one value of parameter α we choose the smallest one. We implemented the most effective algorithm from those discussed in Sect. 4.2(3). From the set $[0, \frac{\pi}{2}]$ is chosen the finite subset of parameters α, from 0 to $\frac{\pi}{2}$ with fixed step 0.01. The code of the algorithm in MATLAB is presented below.

```
% e - list of time series in learning data set(cell vector of vectors)
% labels - vector of labels of elements of list e
% dist - base distance function (ED, DTW)

step = 0.01;
alpha = 1 : step : pi/2;
a = cos(alpha);
b = sin(alpha);

n = length(e);
k = length(alpha);
mistakes(1 : k) = 0; % vector of numbers of misclassified elements

for i = 1 : n
 D(1 : k) = inf; % vector of minimal distances
 L(1 : k) = 0; % vector of ‘minimal’ labels
 for j = [1 : i-1, i+1 : n] % leave-one-out
   d = a * dist(e{j}, e{i}) + b * dist(diff(e{j}), diff(e{i}));
   D(d < D) = d(d < D);
   L(d < D) = labels(j);
 end
 mistakes = mistakes + (L ~= labels(i));
end
errors = mistakes / n; % error rates for every parameter alpha
```

5.2 Main results

Let us return briefly to the previously discussed example (Sect. 3.1). In Fig. 4 we see that we have similar conclusions using our method both with Euclidean (DD_{ED})

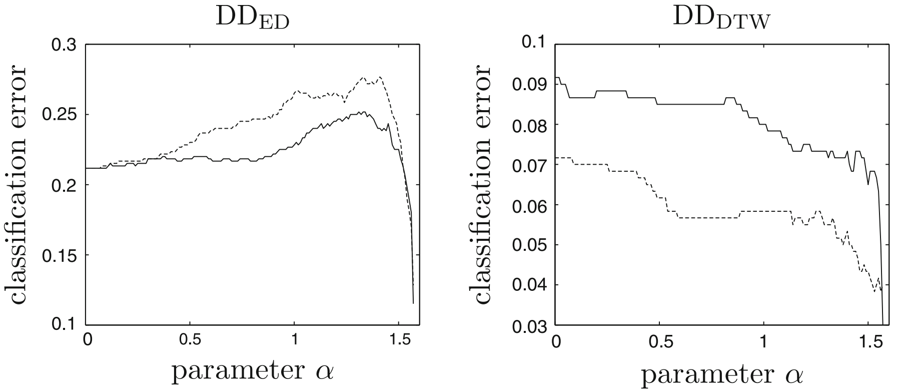

Fig. 4 Dependence of classification error on the participation of the derivative (parameter α) for the example data set (*dashed line* CV error, *solid line* test error)

Table 2 Testing error rates

Data set	ED	DED	$\mathrm{DD}_{\mathrm{ED}}$	DTW	DDTW	$\mathrm{DD}_{\mathrm{DTW}}$	$\frac{\mathrm{DD}_{\mathrm{ED}}-\mathrm{ED}}{\mathrm{ED}}$	$\frac{\mathrm{DD}_{\mathrm{DTW}}-\mathrm{DTW}}{\mathrm{DTW}}$
50Words	36.92	48.35	36.92	30.99	30.33	24.62	0.00	−20.57
Adiac	38.87	41.94	37.60	39.64	41.18	30.18	−3.29	−23.87
Beef	46.67	43.33	43.33	50.00	43.33	43.33	−7.14	−13.33
CBF	14.78	66.00	14.78	0.33	45.33	0.33	0.00	0.00
Coffee	25.00	14.29	7.14	17.86	17.86	10.71	−71.43	−40.00
ECG	12.00	15.00	10.00	23.00	13.00	17.00	−16.67	−26.09
Face (all)	28.64	28.52	28.64	19.23	13.37	9.82	0.00	−48.92
Face (four)	21.59	47.73	21.59	17.05	40.91	17.05	0.00	0.00
Fish	21.71	18.86	19.43	16.57	8.00	5.71	−10.53	−65.52
Gun-point	8.67	8.67	8.67	9.33	1.33	2.00	0.00	−78.57
Lightning-2	24.59	49.18	27.87	13.11	36.07	13.11	13.33	0.00
Lightning-7	42.47	68.49	43.84	27.40	46.58	32.88	3.23	20.00
Olive oil	13.33	20.00	13.33	13.33	20.00	13.33	0.00	0.00
OSU leaf	48.35	60.74	47.93	40.91	11.57	12.40	−0.85	−69.70
Swedish leaf	21.12	44.16	21.12	20.80	11.84	9.76	0.00	−53.08
Synthetic control	12.00	65.33	12.00	0.67	50.00	0.67	0.00	0.00
Trace	24.00	41.00	23.00	0.00	1.00	0.00	−4.17	0.00
Two patterns	9.33	61.28	9.33	0.00	0.33	0.00	0.00	0.00
Wafer	0.45	0.94	0.47	2.01	2.04	2.01	3.57	0.00
Yoga	16.97	25.07	17.37	16.37	17.97	14.40	2.36	−12.02
Mean							−4.58	−21.58

and DTW ($\mathrm{DD}_{\mathrm{DTW}}$) distance. In both cases, the smallest classification error is obtained with a considerable share of derivatives. In the case of DTW we have a continuous decline in error with increasing share of the derivative. However, when

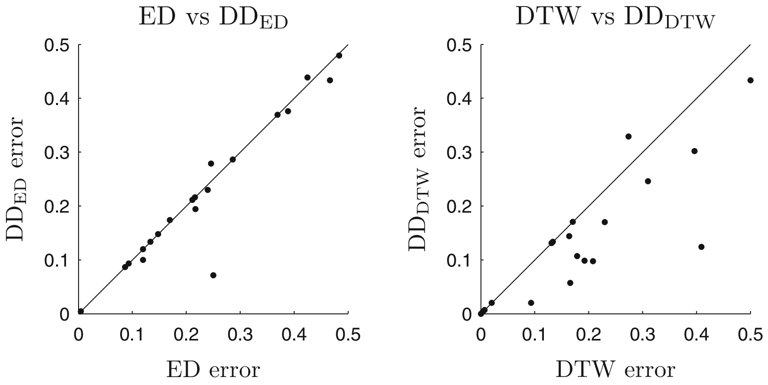

Fig. 5 Comparison of test errors

using the ED we observe an initial increase in error, and then a relatively rapid decline.

The main result is shown in Table 2. In the columns ED and DTW we have (absolute) error rates on the test subset with 1NN method for Euclidean and DTW distance respectively. In the columns DED and DDTW we have (absolute) error rates for methods only with derivatives. In the columns DD$_{\mathrm{ED}}$ and DD$_{\mathrm{DTW}}$ we have absolute error rates, while in the next two columns we have relative error rates. Note that in the first four columns are the results for the distance function which in fact are components of our method. ED is in fact DD$_{\mathrm{ED}}$ with $a = 1,\ b = 0\ (\alpha = 0)$, DED is DD$_{\mathrm{ED}}$ with $a = 0,\ b = 1\ (\alpha = \frac{\pi}{2})$, DTW is DD$_{\mathrm{DTW}}$ with $a = 1,\ b = 0\ (\alpha = 0)$, and DDTW is DD$_{\mathrm{DTW}}$ with $a = 0,\ b = 1\ (\alpha = \frac{\pi}{2})$.

We clearly see that in both cases we obtain a significant reduction in the average relative error of classification (the last row of Table 2). In the case of ED the reduction amounts to 4.58, while for DTW reduction is as high as 21.58. Interestingly, this reduction occurs in the first case for most data sets, while in the case of DTW the error is reduced or remains unchanged. For ED there is improvement in the case of 7 data sets, while 4 are worse, and for 9 there is no difference. However in the case of DTW there is improvement for 11 data sets, worsening for only one, and for 8 there is no difference. Figure 5 presents a graphical comparison of DD methods and its base distance measures that suggests that our method is clearly superior to others on most of the examined data sets.

Of course, the interesting question is that of what derivative contribution in the final distance measure is optimal. Could we obtain some arbitrary quantity that determines for all cases that such and such participation will give us the best result of classification? The answer to this question is negative, as is illustrated by Figs. 6 and 7. We see that in both cases the optimal share of the derivative may be zero, average, or that it exclusively should be used.

Pursuing this approach, we can ask whether at least for the same data for the methods of DD$_{\mathrm{ED}}$ and DD$_{\mathrm{DTW}}$, we can offer a universal value which will determine an optimal contribution to the derivative in the final distance. The answer is negative, as is confirmed by Fig. 8. In this case we see that, although the increasing importance

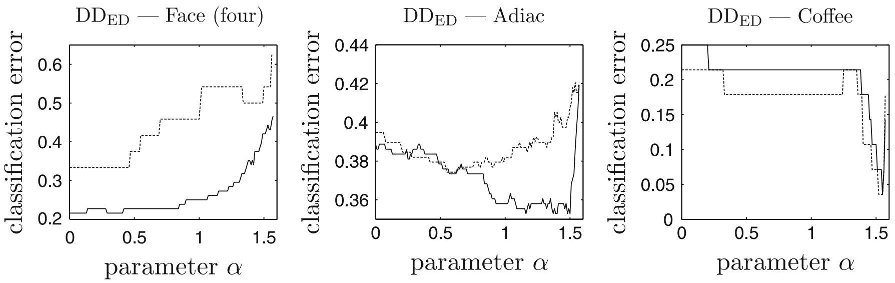

Fig. 6 Dependence of classification error on the participation of the derivative (parameter α) for method $\mathrm{DD_{ED}}$ (*dashed line* CV error, *solid line* test error)

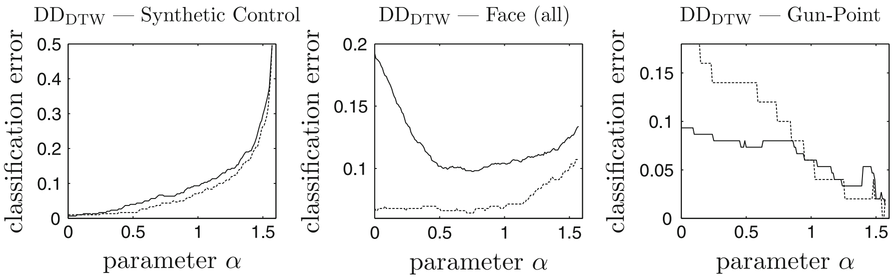

Fig. 7 Dependence of classification error on the participation of the derivative (parameter α) for method $\mathrm{DD_{DTW}}$ (*dashed line* CV error, *solid line* test error)

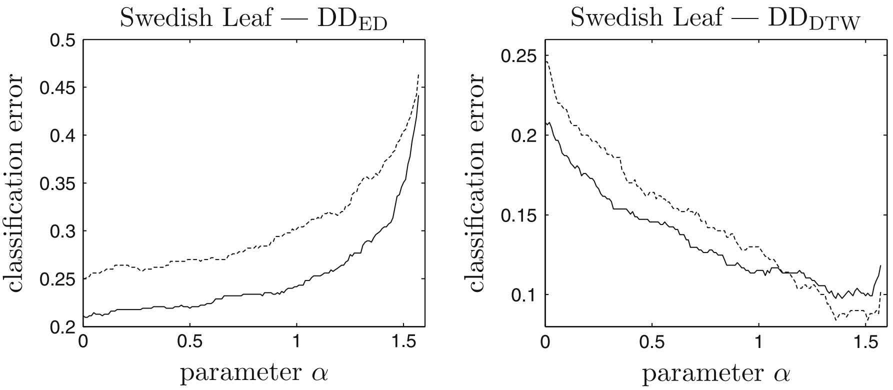

Fig. 8 Dependence of classification error on the participation of the derivative (parameter α) for the data set *swedishleaf* (*dashed line* CV error, *solid line* test error)

of the derivative in the case of method $\mathrm{DD_{ED}}$ worsens the classification, in the case of $\mathrm{DD_{DTW}}$ it improves (significantly). Thus we see that for each data set and each method, we should select the optimal derivative share independently of others.

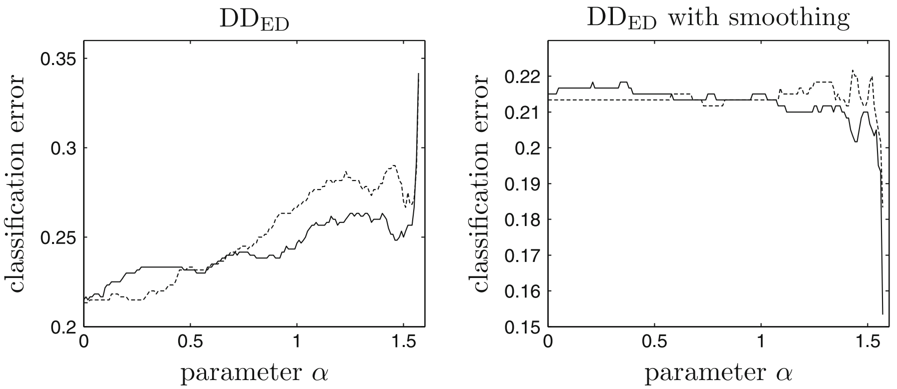

Fig. 9 Dependence of classification error on the participation of the derivative (parameter α) for the example data set with noise and DD_{ED} method without and with smoothing (*dashed line* CV error, *solid line* test error)

5.3 Noise and smoothing

The function derivative is very sensitive to even small changes of function values. Adding noise to a signal has a small influence on function value comparison, but it can have a great influence on function derivative comparison. We have to take this into consideration in our method. We can use any smoothing method before using our distance measure. We do not need to smooth the function before function value comparison, but only before function derivative comparison. We did so with the example in Sect. 3.1 and for ED. The result is shown in Fig. 9. On the left subplot is the classification error rate after we added noise with normal distribution $N(0, 0.001)$. We can see that if participation of the derivative increases the error rate also increases. On the right subplot is the situation after we smoothed the signal before function derivative comparison (the signal for function value comparison includes the noise). To smooth the signal we used a moving average with a length of 10. We see that then the shape of the classification error curve is similar to the shape before addition of noise (Fig. 4, left subplot).

Some slight smoothing before function derivative comparison can also produce a good result for data sets of signals without noise. We used the above procedure (with a moving average length of 3) to the data sets in the paper. For our method with ED both the mean of relative error rates decreased (from -4.58 to -7.69) and more data sets were classified better than by the method without smoothing. For our method with DTW distance the mean of relative error rates increased a little (from -21.58 to $-17, 94$) but also more data sets were classified better (Fig. 10).

We have to admit that the length of the moving average was taken arbitrarily. For other lengths the results may be different. We have to be very careful when smoothing signals for function derivative comparison. Changing the signal by smoothing can affect information which is important to the classification process. Too much smoothing can eliminate the influence of derivative function comparison.

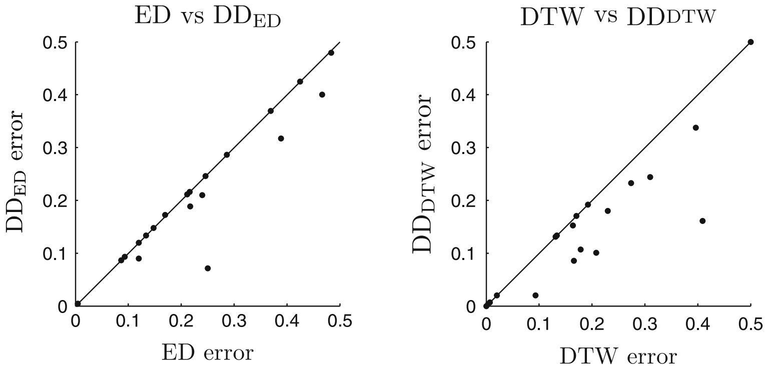

Fig. 10 Comparison of test errors for smoothing versions of DD$_{\mathrm{ED}}$ and DD$_{\mathrm{DTW}}$

6 Comparison with related works

As we can see in Table 2 and Fig. 5, DD$_{\mathrm{DTW}}$ with 1NN seems a fairly good universal classifier of time series. It is interesting to compare the method with other ones using DTW distance and derivatives. We compare DD$_{\mathrm{DTW}}$ to the following distance functions known from the literature (mentioned in Sect. 2). All these distance functions are in fact DTW distances with a different internal distance function d.
DTW—standard DTW distance that compares only values of functions:

$$d(f(i), g(j)) = \big(f(i) - g(j)\big)^2;$$

DDTW—Derivative DTW (Keogh and Pazzani 2001)—DTW distance that compares only values of derivatives:

$$d(f(i), g(j)) = \big(f'(i) - g'(j)\big)^2;$$

VDDTW—Value-Derivative DTW (Kulbacki et al. 2002)—DTW distance that compares both functions and derivatives and combines them multiplicatively:

$$d(f(i), g(j)) = \big(f(i) - g(j)\big)^2\big(f'(i) - g'(j)\big)^2;$$

WDTW—Weighted DTW (Benedikt et al. 2008)—DTW distance that compares functions, first and second derivatives, and combines them additively (the weights are fixed at 1, 2, 2 as suggested in the above paper):

$$d(f(i), g(j)) = \big(f(i) - g(j)\big)^2 + 2\big(f'(i) - g'(j)\big)^2 + 2\big(f''(i) - g''(j)\big)^2.$$

In all distance functions we use the same derivative formula (2) (for different derivative estimators results, see Appendix).

The results of the comparison are shown in Table 3. In the columns we have (absolute) error rates on the test subset with the 1NN method for the previously described

Table 3 Testing error rates (absolute) in comparing classifiers

Data set	DTW	DDTW	VDDTW	WDTW	DD_{DTW}
50Words	30.99	30.33	32.09	27.25	**24.62**
Adiac	39.64	41.18	33.25	36.57	**30.18**
Beef	50.00	**43.33**	46.67	50,00	**43.33**
CBF	**0.33**	45.33	6.78	25.56	**0.33**
Coffee	17.86	17.86	**10.71**	21.43	**10.71**
ECG	23.00	**13.00**	14.00	15.00	17.00
Face (all)	19.23	13.37	12.13	**9.23**	9.82
Face (four)	**17.05**	40.91	38.64	27.27	**17.05**
Fish	16.57	8.00	8.57	11.43	**5.71**
Gun-point	9.33	**1.33**	2.67	**1.33**	2.00
Lightning-2	**13.11**	36.07	34.43	24.59	**13.11**
Lightning-7	**27.40**	46.58	47.95	35.62	32.88
Olive oil	**13.33**	20.00	**13.33**	**13.33**	**13.33**
OSU leaf	40.91	**11.57**	25.62	36.36	12.40
Swedish leaf	20.80	11.84	12.48	13.92	**9.76**
Synthetic control	**0.67**	50.00	8.33	26.33	**0.67**
Trace	**0.00**	1.00	**0.00**	1.00	**0.00**
Two patterns	**0.00**	0.33	0.03	**0.00**	**0.00**
Wafer	2.01	2.04	2.56	**1.69**	2.01
Yoga	16.37	17.97	17.77	15.08	**14.40**

Table 4 Average relative error rates

X	DTW	DDTW	VDDTW	WDTW
Mean	−21.58	−33.72	−32.29	−27.21

distance functions (the best results are bolded). We see that our classifier is slightly worse than one of the compared classifiers only on 6 of the 20 examined data sets (once than DTW, three times than DDTW and three times than WDTW). Our method should always gives the results not worse than DTW and DDTW (because it contains these models as components). Sometimes this is not happening. The blame for this lies not always perfect parameter selection method, which in this case is the CV. Simply a method sometimes do not find correctly the optimal values of parameter. This is due to a situation in which a learning set do not fully reflect the structure of the test set.

As a measure of relative performance we may use the average of the relative error rates ($\frac{DD_{DTW}-X}{X}$) across data sets. We can find this information in Table 4.

We can see that we obtain an average reduction in the relative error rates ranging from 21.58 (DTW) to 33.72% (DDTW). These differences seem significant.

However, to confirm statistically the quality of our classifier, in the next step we test the null-hypothesis that all classifiers perform the same and the observed differences are merely random. We used the Iman and Davenport (1980) version of the F-test,

Table 5 Mean ranks

DTW	DDTW	VDDTW	WDTW	DD_{DTW}
3.25	3.60	3.30	3.05	1.80

which is a non-parametric equivalent of the repeated-measures ANOVA. The F-test is recommended because it is less conservative than other tests (Looney 1998; Demšar 2006). It ranks the classifiers for each data set separately, the best performing classifier receiving the rank of 1, the second best the rank of 2 and so on (in case of ties average ranks are assigned).

Let R_{ij} be the rank of the jth of K methods on the ith of N data sets and let $R_j = \frac{1}{N}\sum_{i=1}^{N} R_{ij}$.

The Iman and Davenport test compares the mean ranks $R_1, R_2, \ldots, R_K$ (given in Table 5) of classifiers and is based on the statistic

$$S = \frac{(N-1)S_1}{N(K-1)-S_1} \tag{3}$$

where

$$S_1 = \frac{12N}{K(K+1)}\sum_{i=1}^{K} R_i^2 - 3N(K+1)$$

is the Friedman (1937, 1940) statistic. The statistic S is distributed according to the F-distribution with $K-1$ and $(K-1)(N-1)$ degrees of freedom. In our case $N=20$ and $K=5$. The value of statistic S given by (3) is equal to 4.62. The corresponding critical value is equal to 2.49 for $\alpha = 0.05$. We see that the null-hypothesis that all classifiers give the same results is rejected (the p-value is 0.002) and we can proceed with a post-hoc test.

Hence in the next step we can use the Nemenyi (1963) test procedure, in which all classifiers are compared to each other. The Nemenyi test is similar to the Tukey test for ANOVA and is used when all classifiers are compared to each other. The performance of two classifiers is significantly different at the experimentwise error rate α if

$$|R_i - R_j| > q(\alpha, K, \infty)\left(\frac{K(K+1)}{12N}\right)^{1/2}, \\ i = 1, \ldots, K-1,\ \ j = i+1, \ldots, K, \tag{4}$$

where the values of $q(\alpha, K, \infty)$ are based on the Studentized range statistic (Hollander and Wolfe 1973; Demšar 2006). However, Demšar (2006) showed that when all classifiers are compared with a control classifier, we can use, instead of the Nemenyi test, one of the general procedures for controlling the family-wise error in multiple hypothesis testing, such as the Bonferroni–Dunn correction (Dunn 1961). Although these methods are generally conservative and sometimes lack power, in this specific case they are more powerful than the Nemenyi test, since the latter adjusts the critical value for making $K(K-1)/2$ comparisons, whereas when comparing with a control we make only $K-1$ comparisons. The power of the post-hoc test is much greater when all classifiers are compared only to a control classifier and not between themselves. Hence we should not make pairwise comparisons when we in fact only test whether a newly proposed method is better than the existing ones. To perform this test we can use the inequality (4), but using the critical values for $\alpha/(K-1)$.

The right-hand side of the inequality (4) is equal to 1.249 for $N = 20$, $K = 5$ and $\alpha = 0.05/4$. Using the mean ranks from Table 5 we see that our classifier is significantly better than each of the compared ones at the experimentwise error rate $\alpha = 0.05$.

7 Conclusions and future work

In this paper we have introduced and studied a new time series distance based on derivative. We used this distance to classify time series with the 1NN method. Our research showed that our method gives very good results. On many data sets our distance measure is superior to both the ED and the DTW. The proposed method, thanks to a parametrical approach, makes it possible to choose an appropriate model for any data set. Due to the high nonlinearity, the method does not easily lead to a rigorous theoretical analysis. However the experiments that we have conducted justify the power and usefulness of our method.

We are currently adapting our distance measure to account for the second derivative and higher. In addition, we are experimenting with the use of different distance measures for the value and for derivatives. We are also looking for functions other than derivative which can be used in a similar manner (for example Batista et al. 2011 introduced complexity of time series, which is the length of the line obtained by stretching them).

8 Appendix

8.1 Lower bound and triangular inequality proof

Let us define the distance function $\widehat{\mathrm{dist}}_{ab}$ by

$$\widehat{\mathrm{dist}}_{ab}(f, g) := a\,\mathrm{dist}_1(f, g) + b\,\mathrm{dist}_2(f', g'), \quad a, b \in [0, 1],$$

where dist_1, dist_2 are distances between time series and their derivatives.

If low_1, low_2 are lower bounds of the distances dist_1, dist_2:

$$\mathrm{low}_1(f, g) \le \mathrm{dist}_1(f, g),$$
$$\mathrm{low}_2(f', g') \le \mathrm{dist}_2(f', g'),$$

then

$$\begin{aligned}\widehat{\mathrm{low}}_{ab}(f,g) &:= a\,\mathrm{low}_1(f,g) + b\,\mathrm{low}_2(f',g') \leq \\ &\leq\ a\,\mathrm{dist}_1(f,g) + b\,\mathrm{dist}_2(f',g') = \\ &=\ \widehat{\mathrm{dist}}_{ab}(f,g).\end{aligned}$$

Therefore, $\widehat{\mathrm{low}}_{ab}$ is a lower bound of the distance function $\widehat{\mathrm{dist}}_{ab}$.

If dist_1, dist_2 obey the triangular inequality:

$$\begin{aligned}\mathrm{dist}_1(f,g) &\leq \mathrm{dist}_1(f,h) + \mathrm{dist}_1(h,g),\\ \mathrm{dist}_2(f',g') &\leq \mathrm{dist}_2(f',h') + \mathrm{dist}_2(h',g'),\end{aligned}$$

then

$$\begin{aligned}\widehat{\mathrm{dist}}_{ab}(f,g) &= a\,\mathrm{dist}_1(f,g) + b\,\mathrm{dist}_2(f',g') \leq \\ &\leq a(\mathrm{dist}_1(f,h) + \mathrm{dist}_1(h,g)) + b(\mathrm{dist}_2(f',h') + \mathrm{dist}_2(h',g')) = \\ &= (a\,\mathrm{dist}_1(f,h) + b\,\mathrm{dist}_2(f',h')) + (a\,\mathrm{dist}_1(h,g) + b\,\mathrm{dist}_2(h',g')) = \\ &= \widehat{\mathrm{dist}}_{ab}(f,h) + \widehat{\mathrm{dist}}_{ab}(h,g).\end{aligned}$$

Therefore, the distance function $\widehat{\mathrm{dist}}_{ab}$ obeys the triangular inequality.

8.2 Derivative influence comparison

We use three derivative estimators:

$$f'(i) = f(i) - f(i-1) \tag{1}$$

$$f'(i) = \frac{f(i+1) - f(i-1)}{2} \tag{2}$$

$$f'(i) = \frac{f(i) - f(i-1) + \frac{f(i+1)-f(i-1)}{2}}{2} \tag{3}$$

with methods: DDTW, VDDTW, WDTW and $\mathrm{DD}_{\mathrm{DTW}}$. The results of the comparison are shown in Table 6. In the columns we have (absolute) error rates on the test subset with the 1NN method for the previously described distance functions and derivative estimators. The last row describes the number of wins (including ties) of each derivative for each distance function separately.

To confirm statistically that the quality of classifiers does not depend on the kind of derivative we test the null-hypothesis that all methods perform the same and the observed differences are merely random. We used the Iman and Davenport version of the F-test described in Sect. 6. We obtain the p-value 0.534 for DDTW, 0.778 for VDDTW, 0.329 for WDTW and 0.641 for $\mathrm{DD}_{\mathrm{DTW}}$. We see that the null-hypothesis that all derivatives give the same results is not rejected for all classifiers, hence all methods do not statistically differ.

Table 6 Comparison of the derivative formulas

	DDTW			VDDTW			WDTW			DD$_{\text{DTW}}$		
	(1)	(2)	(3)	(1)	(2)	(3)	(1)	(2)	(3)	(1)	(2)	(3)
50Words	30.33	30.77	30.77	32.09	32.75	32.31	27.25	27.03	27.03	24.62	25.05	24.18
Adiac	41.18	40.15	41.43	33.25	32.23	33.50	36.57	35.81	36.83	30.18	31.20	30.43
Beef	43.33	53.33	46.67	46.67	50.00	46.67	50.00	50.00	50.00	43.33	43.33	43.33
CBF	45.33	38.89	40.67	6.78	3.56	4.22	25.56	3.22	10.67	0.33	0.33	0.33
Coffee	17.86	10.71	17.86	10.71	10.71	10.71	21.43	21.43	21.43	10.71	10.71	10.71
ECG	13.00	16.00	17.00	14.00	16.00	12.00	15.00	17.00	13.00	17.00	23.00	15.00
Face (all)	13.37	12.96	12.66	12.13	11.48	11.78	9.23	8.22	8.05	9.82	18.93	18.76
Face (four)	40.91	34.09	37.50	38.64	42.05	40.91	27.27	29.55	27.27	17.05	17.05	17.05
Fish	8.00	8.57	10.29	8.57	7.43	6.86	11.43	10.86	11.43	5.71	8.00	5.14
Gun-point	1.33	0.67	0.67	2.67	1.33	2.67	1.33	1.33	1.33	2.00	3.33	4.00
Lightning-2	36.07	31.15	32.79	34.43	21.31	37.70	24.59	11.48	14.75	13.11	13.11	14.75
Lightning-7	46.58	46.58	42.47	47.95	45.21	41.10	35.62	26.03	27.40	32.88	30.14	32.88
Olive oil	20.00	26.67	20.00	13.33	20.00	13.33	13.33	13.33	13.33	13.33	13.33	13.33
OSU leaf	11.57	12.40	11.57	25.62	26.03	24.38	36.36	37.19	37.19	12.40	13.22	11.16
Swedish leaf	11.84	10.24	11.52	12.48	10.40	11.84	13.92	12.64	13.28	9.76	8.48	11.20
Synthetic control	50.00	34.00	43.33	8.33	10.33	7.33	26.33	4.33	10.67	0.67	0.67	0.67
Trace	1.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
Two patterns	0.33	0.10	0.25	0.03	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Wafer	2.04	2.48	2.21	2.56	2.16	2.63	1.69	2.09	2.03	2.01	2.01	2.01
Yoga	17.97	18.43	17.70	17.77	18.43	18.50	15.80	15.87	15.93	14.40	14.03	13.60
Wins	7	10	7	7	9	10	9	14	10	13	12	14

References

Batista G, Wang X, Keogh E (2011) A complexity-invariant distance measure for time series. In: Eleventh SIAM international conference on data mining (SDM'2011), Mesa, USA

Benedikt L, Cosker D, Rosin PL, Marshal D (2008) Facial dynamics in biometric identification. In: BMVC, vol 2, pp 235– 241

Benedikt L, Cosker D, Rosin PL, Marshal D (2010) Assessing the uniqueness and permanence of facial actions for use in biometric applications. IEEE Trans Syst Man Cybernet A Syst Humans 40(3):449–460

Berndt DJ, Clifford J (1994) Using dynamic time warping to find patterns in time series. In: AAAI workshop on knowledge discovery in databases, pp 229–248

Box GEP, Jenkins GM, Reinsel GC (2008) Time series analysis: forecasting and control. Wiley, New York

Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30

Ding H, Trajcevski G, Scheuermann P, Wang X, Keogh E (2008) Querying and mining of time series data: experimental comparison of representations and distance measures. In: Proceedings of 34th international conference on very large data bases, pp 1542–1552

Dunn OJ (1961) Multiple comparisons among means. J Am Stat Assoc 56:52–64

Eads D, Hill D, Davis S, Perkins S, Ma J, Porter R, Theiler J (2002) Genetic algorithms and support vector machines for time series classification. In: Proc Int Soc Optic Eng 4787:74–85

Friedman M (1937) The use of ranks to avoid the assumption of normality implicit in the analysis of variance. J Am Stat Assoc 32:675–701

Friedman M (1940) A comparison of alternative tests of significance for the problem of m rankings. Ann Math Stat 11:86–92

Gullo F, Ponti F, Tagarelli A, Greco S (2009) A time series representation model for accurate and fast similarity detection. Pattern Recogn 42(11):2998–3014

Hollander M, Wolfe DA (1973) Nonparametric statistical methods. Wiley, New York

Iman RL, Davenport JM (1980) Approximations of the critical region of the Freidman statistic. Commun Stat Theory Methods 9:571–595

Keogh E (2002) Exact indexing of dynamic time warping. In 28th International Conference on Very Large Data Bases 406-417

Keogh E, Kasetty E (2003) On the need for time series data mining benchmarks: a survey and empirical demonstration. Data Min Knowl Discov 7(4):349–371

Keogh E, Pazzani M (2001) Dynamic time warping with higher order features. In: Proceedings of SIAM international conference on data mining (SDM'2001), Chicago, USA

Keogh E, Xi X, Wei L, Ratanamahatana CA (2006) The UCR Time Series Classification/Clustering Homepage: http://www.cs.ucr.edu/~eamonn/time_series_data/

Kulbacki M, Segen J, Bak A (2002) Unsupervised learning motion models using dynamic time warping. In: Proceedings of the intelligent information systems 2002 symposium, Sopot, 2002

Looney SW (1998) A statistical technique for comparing the accuracies of several classifiers. Pattern Recogn Lett 8:5–9

Luan F, Li K, Ma S (2010) The algorithm of online handwritten signature verification based on improved DTW. Int J Model Identif Control 10(1-2):81–86

Mokhtar N, Arof H, Iwahashi M (2010) One dimensional image processing for eye tracking using derivative dynamic time warping. Sci Res Essays 5(19):2947–2952

Nemenyi PB (1963) Distribution-free multiple comparisons. PhD thesis, Princeton University

Pavlovic V, Frey BJ, Huang TS (1999) Time-series classification using mixed-state dynamic Bayesian networks. In: Proceedings of IEEE conference on computer vision and pattern recognition, vol 2, pp 2609–2615

Penny W, Roberts S (1999) Dynamic models for nonstationary signal segmentation. Comput Biomed Res 32(6):483–502

Petridis V, Kehagias A (1997) Predictive modular neural networks for time series classification. Neural Netw 10(1):31–49