A multi-scale smoothing kernel for measuring time-series similarity

A. Troncoso, M. Arias, J.C. Riquelme

1. Introduction

Time-series analysis is an important problem with application in domains as diverse as engineering, medicine, astronomy or finance [11,29]. In particular, the problem of time-series classification and prediction is attracting a lot of attention among researchers. One of the most successful and popular methods for classification and prediction are kernel-based methods such as support vector machines (SVM) [26,12,35,25]. Despite their popularity, there seem to be only a handful of kernels designed for time-series. This paper tries to fill this gap, and proposes a kernel exclusively designed for time-series. Moreover, using a standard trick, we are able to convert our kernel into a distance for time-series, therefore allowing us to use our kernel in distance-based algorithms as well.

A crucial aspect when dealing with time-series is to find a good measure, either a kernel similarity or a distance, that captures the essence of the time-series according to the domain of application.

For example, Euclidean distance between time-series is commonly used due to its computational efficiency; however, it is very brittle and small shifts in one time-series can result in huge changes in the Euclidean distance. Therefore, more sophisticated distances have been devised and designed to be more robust to small fluctuations of the input time-series. Notably, Dynamic Time Warping (DTW) [30] is held as the state-of-the-art method for comparing the similarity among time-series. The DTW is very powerful in the sense that it can deal optimally with contractions, expansions and shifts in time-series in addition to being able to handle time-series of different lengths. Unfortunately, computing the DTW distance is prohibitively costly for many practical applications [33]. Moreover, it cannot be used to define a positive definite kernel since it violates the triangle inequality [3].

Therefore, researchers are coming up with distances for time-series that approximate the DTW at lower computational costs either by adding global path constraints [30,36] or by reducing the number of instances e.g. in nearest neighbor classification [35].

In this paper we introduce a new kernel, called Multi-Scale Smoothing Kernel (MUSS). The basic idea behind our kernel is to take into account many smoothed versions of the time-series and compute the similarity of the time-series as the aggregation of the similarities of the multiple smoothed versions of the original time-series. The underlying idea is that by smoothing the original time-series we will get rid of slight perturbations, and so the basic trends will become apparent and more easily detected. The main strength of this kernel is the integration of multiple time-scales, that is, at a high level, the MUSS kernel is a combination of linear kernels obtained by

**Abstract**

In this paper a kernel for time-series data is introduced so that it can be used for any data mining task that relies on a similarity or distance metric. The main idea of our kernel is that it should recognize as highly similar time-series that are essentially the same but may be slightly perturbed from each other: for example, if one series is shifted with respect to the other or if it slightly misaligned. Namely, our kernel tries to focus on the shape of the time-series and ignores small perturbations such as misalignments or shifts. First, a recursive formulation of the kernel directly based on its definition is proposed. Then it is shown how to efficiently compute the kernel using an equivalent matrix-based formulation. To validate the proposed kernel three experiments have been carried out. As an initial step, several synthetic datasets have been generated from UCR time-series repository and the KDD challenge of 2007 with the purpose of validating the kernel-derived distance over shifted time-series. Also, the kernel has been applied to the original UCR time-series to analyze its potential in time-series classification in conjunction with Support Vector Machines. Finally, two real-world applications related to ozone concentration in atmosphere and electricity demand have been considered.

**Keywords:**
- Kernel
- Similarity
- Distance
- Time-series classification
using several smoothed versions over different scales from the original time-series. In a sense, the kernel-derived distance that is proposed here tries to fix the brittleness of Euclidean distance without incurring in the high computational costs of DTW. Moreover, our kernel can easily be adapted to deal with multidimensional time series by considering multi-variate versions of the point-wise distance between time-series. In addition, we can derive a distance metric from the kernel definition that satisfies the triangle inequality.

The main goal of the proposed kernel is to recognize as similar time-series that may be slightly perturbed from one another. Namely, it tries to focus on the shape of the time-series and not so much on the details. It is conceivable that small errors in measurement or delivery of data may result in slight shifts or misalignments of time-series. Consequently, any data that is sent through complicated machinery can suffer from this type of misalignment as for example astrometric data, and could benefit from our kernel.

In this work, two ways of computing the kernel are presented: a recursive formulation and an equivalent matrix-based formulation. To evaluate the proposed kernel three experiments have been carried out. As an initial step, several synthetic datasets have been generated from UCR time-series repository [20] and the KDD challenge of 2007 [19], with the purpose of validating our kernel-derived distance over shifted time-series. In particular, a comparison with DTW and Euclidean distances shows that our kernel-derived distance outperforms the Euclidean distance and is competitive with respect to the DTW distance while having a much lower computational cost. The DTW distance is designed to deal with misalignments and shifts optimally. Therefore, our objective is not to beat the DTW, but to approach its performance without incurring its high computational cost. On the other hand, the Euclidean distance has been considered as baseline distance. In the second experiment, the proposed kernel has been applied to the original UCR time-series [20] to analyze its performance. In this case, the proposed kernel shows a remarkable performance when compared with a kernel based on DTW [10] and a linear kernel. Finally, two real-world applications related to ozone concentration in atmosphere and electricity demand have been considered to show the performance of the MUSS kernel over very specific datasets. In this case, an accuracy ranging from 97% to 99% has been obtained.

The paper is structured as follows. Section 2 presents the most relevant related works found in literature. Section 3 describes our time-series kernel and its corresponding derived distance. The experimental results are presented in Sections 4–6. Finally, Section 7 concludes with a summary of our main contributions and possible directions for future work.

2. Related work

Similarity and distance measures for time-series are a crucial ingredient in solving time-series classification and forecasting problems [29,15]. For this reason, many distances have been proposed. For example, [1] defines a distance between two time-series representing the convexities/concavities of two shape contours. In [4] the authors modify the Euclidean distance with a correction factor based on the complexity of the input time-series.

The success and popularity of Support Vector Machines has motivated researchers to design kernels that capture similarity between time-series and sequences. For example, [32] define a kernel for the particular task of handwritten character recognition. In this work, the authors approximate each time-series by a linear combination of piecewise polynomial functions and the kernel is based on the product of the coefficients and functions that form part of this approximation. Another family of kernels for time-series based on Echo State Network [23,27] with a deterministic reservoir architecture is proposed in [6]. Their kernel is defined by Gaussian kernel with the $L_2$ distance between the corresponding readouts for each time-series from the same reservoir.

It is well-known that the DTW distance is not a distance in a strict sense as it does not satisfy the triangle inequality and, therefore, it cannot be used to define a positive definite kernel [3]. Despite this disadvantage, many variants of DTW and definitions of kernels based on DTW have been recently proposed in the literature. As an example, [17] use Gaussian kernel and the DTW distance with a special support vector machine, which has the ability to handle non positive-definite kernel matrices.

Another example of the use of (a weighted variant of) DTW for time-series classification, this time based on nearest neighbors is [18]. The weights penalize instances with higher phase difference between a reference point and a testing point with the purposes of minimizing the distortion caused by outliers.

More recently, non-linear kernels have been proposed for time-series classification. In [10] a new kernel based on the DTW distance is defined by global alignments (GA-DTW). In particular, the kernel is defined as the sum of the exponential function of the distances for all possible alignments. However, this kernel has a high computational cost and similar constraints on alignments to that of [30] are presented to speed-up the computation in [8]. The same author presents another kernel based on the idea that similar time-series should be fit well by the same models [9]. The author used autoregressive models and thus the name of autoregressive kernel. In particular, these two global alignment-based and autoregressive kernels defined in [8] and [9] have been recently used in machineolfaction applications in conjunction with SVM [33]; An extension of SVM based on nonlinear dynamical systems theory is presented in [10]; here it is shown that these non-linear methods perform better and faster than the DTW distance-based methods. However, linear kernels may still be preferred over their more accurate non-linear counterparts due to their interpretability, computational efficiency and the lack of metaparameters that need tuning.

A kernel for periodic time-series arising in the field of astronomy is presented in [34]. This kernel is similar to a global alignment kernel as it consists of the sum of the exponential function of the inner products for all possible shifts of a time series instead of alignments.

Finally, another kernel for time-series is proposed in [22]. In particular, the time series are represented with a summarizing smooth curve in a Hilbert space and the learning method of the kernel is based on Gaussian processes.

In the last years, several approaches have been proposed to combine multiple kernels instead of using a single kernel. In [28] a combination of kernels for long-term time-series forecasting is presented. In particular, a kernel that takes into account the seasonality of the time-series to improve the performance of the predictor is combined with the well-known Gaussian or rational quadratic kernel. A detailed description can be found in [14].

Due to the fact that annotation of class labels in time-series is very expensive, researchers are exploring the semi-supervised methodology to the problem of time-series classification. The main strategies within this line of work are the extension of well-known semi-supervised techniques for static data classification to time-series problems [21], and the definition of new distances for time-series that work well in semi-supervised classification [7].

3. Kernel description

This section presents the notation used in this paper and also provides the definitions underlying the proposed kernel.
3.1. Preliminaries

**Definition 1.** Time-series. A time-series $X$ is a set of temporally sorted sequence of real values. In this work, $X = \{x_1, \ldots, x_N\}$, where $N$ is the length of the time-series.

**Definition 2.** Subsequence time-series. A subsequence of length $k$ of a time-series $X = \{x_1, \ldots, x_N\}$ is a time-series $X^k = \{x_j, x_{j+1}, \ldots, x_{j+k-1}\}$ for $1 \leq j \leq N-k+1$.

**Definition 3.** $(k, j)$-Order partial sum. A $(k, j)$-order partial sum of a time-series $X$, $S^{k,j}_X$, is the sum of the values of the $X^k$ subsequence of time-series of length $k$. That is

$$S^{k,j}_X = \sum_{i=0}^{k-1} x_{j+i} = x_j + x_{j+1} + \cdots + x_{j+k-1}.$$

**Definition 4.** $k$-Order partial sum time-series. A $k$-order partial sum time-series is a time-series $S^k_X$ whose values are $S^{k,j}_X$ for $1 \leq j \leq N-k+1$, that is, the sum of all the values of the subsequence of length $k$ of the time-series $X$.

For example, the $(k, j)$-order partial sums and the $k$-order partial sum time-series for $X = \{3, 2, 4, 1\}$ time-series are

$$S^{2,1}_X = 3 + 2 = 5,$$
$$S^{2,2}_X = 2 + 4 = 6,$$
$$S^{2,3}_X = 4 + 1 = 5,$$
$$S^{3,1}_X = 3 + 2 + 4 = 9,$$
$$S^{3,2}_X = 2 + 4 + 1 = 7,$$
$$S^{4,1}_X = 3 + 2 + 4 + 1 = 10,$$  
$S^4_X = \{10\}$

3.2. Motivation

The main motivation in the definition of the MUSS kernel proposed in this paper is to obtain a similarity measure for time-series that yields high values when two time-series $X$ and $Y$ have the same shape but may be slightly perturbed from each other. As an illustration, Fig. 1 shows two time-series that are in fact similar in the sense that they follow the same trend, however, the Euclidean distance between the two time-series is high and so it fails at detecting the similarity. As a consequence, using the Euclidean distance in distance-based classification algorithms may lead to poor results. The purpose of this work is to propose a kernel that yields high similarity for time-series that have similar shapes with a reasonable time complexity.

Fig. 2 shows the different time-series composed by the values of the partial sums of the $X$ and $Y$ time-series for orders $k \in \{1, 20, 60, 200\}$. Note that when the order is $k=1$ the time-series are the original $X$ and $Y$ time-series. It can be readily observed that the greater the order of partial sums is, the smoother the time-series become. In fact, the partial sums are essentially unnormalized window-average smoothing over the original series.

The MUSS kernel proposed here is obtained by adding the inner products of partial sum time-series over all possible orders. It is not necessary to discover the best alignment between two time-series, in contrast to the DTW distance, as all partial sums will be included in the kernel definition. Moreover, as it is shown in Section 4 the MUSS kernel can be calculated very fast when compared to the DTW distance since finding an optimal alignment is not necessary.  

3.3. Definition of the kernel

Let $X$ and $Y$ be two time-series of length $N$. Let $U^X$ and $U^Y$ be two upper triangular matrices defined as

$$U^X = [U^X_1, \ldots, U^X_N]$$

and

$$U^Y = [U^Y_1, \ldots, U^Y_N],$$

where $U^X_i$ and $U^Y_i$ are the $i$-th rows of the matrices $U^X$ and $U^Y$, respectively, which are defined by the elements $u^X_i$ and $u^Y_i$ as follows:

$$u^X_i = \begin{cases} S^X_i & \text{if } 1 \leq j \leq N-i+1 \\ 0 & \text{if } j > N-i+1 \end{cases}$$

and

$$u^Y_i = \begin{cases} S^Y_i & \text{if } 1 \leq j \leq N-i+1 \\ 0 & \text{if } j > N-i+1 \end{cases}$$

Finally, the MUSS kernel is defined as the sum of the scalar products among the rows of the $U^X$ and $U^Y$ matrices. That is,

$$Kernel(X, Y) = \sum_{i=1}^{N} (U^X_i, U^Y_i)$$

where $U^X_i$ and $U^Y_i$ are defined by Eqs. (1)–(4) and $(\cdot, \cdot)$ is the scalar product of two vectors in $\mathbb{R}^N$. It is obvious that the function defined by Eq. (5) is indeed a kernel as it can be represented by a inner product in the high-dimensional feature space $\phi(\cdot)$ defined as follows:

$$Kernel(X, Y) = \langle \phi(X), \phi(Y) \rangle$$

where

$$\phi: \mathbb{R}^N \longrightarrow \mathbb{R}^{N^2}$$

and

$$X \longrightarrow \phi(X) = (U^X_1, \ldots, U^X_N)$$

Next, we show an illustrative example for the time-series $X = \{3, 2, 4, 1\}$ and $Y = \{1, -1, 0, 2\}$. Firstly, the $U^X$ and $U^Y$ matrices comprising the partial sums of the $X$ and $Y$ time-series have to be computed. It is necessary only to estimate the partial sums for all the orders greater than or equal to 2 as the 1 order partial sums are the own values of the time-series. The 2-order partial sums for $X$ and $Y$ are $S^X_2 = \{5, 6, 5\}$ and $S^Y_2 = \{0, -1, 2\}$, respectively. Analogously, the 3 and 4 order partial sums are $S^X_3 = \{9, 7\}$, $S^Y_3 = \{0, 1, 1\}$, $S^X_4 = \{10\}$, and $S^Y_4 = \{\}$.  

![Fig. 1. Example of two similar time-series. These time-series represent measurements taken each second over an 11 minute period. One of the time-series (thin, dotted blue) has been obtained from the other (thick, red continuous) as follows: within each minute block, the values of the original (red) time-series have been circularly shifted with an offset of 30 measurements. The Euclidean distance, as illustrated by the shaded regions between these two time-series, is high. On the other hand our kernel should recognize both time-series as very similar. Plot is best seen in color. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)](image-url)
3.4.1. Recursive scheme

ditional cost. In this subsection these two schemes are described in different schemes to compute the kernel at a far lower computation given above. However, an exhaustive analysis provides two
t since a priori there is no reason why one should pay more attention to one particular scale. Plot is best seen in color.

This scheme is based on the fact that the $i$-th row of the $U^X$ matrix can be recursively obtained from the $(i-1)$-th row and the original values of the time-series $X$. Continuing the previous example, the following relationships hold:

$$ U_{11}^X = 5 = U_{1,1}^X + x_2 = 3 + 2 $$
$$ U_{2,2}^X = 6 = U_{2,2}^X + x_3 = 2 + 4 $$
$$ U_{3,3}^X = 5 = U_{1,3}^X + x_4 = 4 + 1 $$
$$ U_{4,4}^X = 10 = U_{4,4}^X + x_4 = 9 + 1 $$

In general, the $(ij)$-element of the $U^X$ matrix can be obtained from the recursive formula as follows:

$$ U_{ij}^X = U_{i-1,j}^X + x_{i+j-1} $$

3.4.2. Matrix-based scheme

This scheme is based on the observation that each row $U_i^X$ of the matrix $U^X$ can be obtained by multiplying a matrix $M_i$ by the row vector composed of the values of the time-series. That is,

$$ U_i^X = (x_1, \ldots, x_N) \cdot M_i $$

where $M_i = \{a_{ij}\}$ is the $N \times N$ matrix capturing the partial sums of order $i$ and the $A_i = \{a_{ij}\}_{1 \leq j \leq N}$ matrix is defined by $a_{ij} = 1$ for all $I_i$ such that $1 \leq i \leq i$ and $1 \leq j \leq N-i+1$, and 0 otherwise.

The matrix $M_i$ has two well-differentiated blocks, a first block, $A_i$, comprising a diagonal band matrix of width $i$ composed of 1's and two null triangular matrices of dimensions $(N-i) \times (N-i)$ and a second block, composed of a null matrix of dimensions $N \times (i-1)$. It can be observed that the structure of the $M_i$ matrix is mainly due to the definition of the $i$-order partial sums as the diagonal band of width $i$ represents the $i$-order partial sums for the different subsequence time-series of length $i$. On the other hand, the null matrix of dimension $N \times (i-1)$ is because of the columns of the $U^X$ matrix are equal to 0 when $j > N-i+1$.

$S^X_2 = \{10\}$ and $S^Y_2 = \{2\}$. Therefore, the matrices are

$$ U^X = \begin{bmatrix} 3 & 2 & 4 & 1 \\ 5 & 6 & 5 & 0 \\ 9 & 7 & 0 & 0 \\ 10 & 0 & 0 & 0 \end{bmatrix}, \quad U^Y = \begin{bmatrix} 1 & -1 & 0 & 2 \\ 0 & -1 & 2 & 0 \\ 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 0 \end{bmatrix} $$

The second step consists in calculating the scalar products of the rows of the $U^X$ matrix and the corresponding rows of the $U^Y$ matrix. That is,

$$ \langle U_{1,1}^X, U_{1,1}^Y \rangle = 3 \cdot 1 + 2 \cdot (-1) + 4 \cdot 0 + 1 \cdot 2 = 3 $$
$$ \langle U_{2,2}^X, U_{2,2}^Y \rangle = 5 \cdot 0 + 6 \cdot (-1) + 5 \cdot 2 = 4 $$
$$ \langle U_{3,3}^X, U_{3,3}^Y \rangle = 9 \cdot 0 + 7 \cdot 1 = 7 $$
$$ \langle U_{4,4}^X, U_{4,4}^Y \rangle = 10 \cdot 2 = 20 $$

where $U^X_i$ and $U^Y_i$ are the $i$ rows of the $U^X$ and $U^Y$ matrices, respectively.

Finally, the MUSS kernel is defined as the sum of the above-mentioned scalar products. Therefore,

$$ \text{Kernel}(X, Y) = (3 + 4 + 7 + 20) = 34 $$

3.4. Speeding-up the computation

The MUSS kernel can be computed following from the definition given above. However, an exhaustive analysis provides two different schemes to compute the kernel at a far lower computational cost. In this subsection these two schemes are described in detail.

3.4.1. Recursive scheme

This scheme is based on the fact that the $i$-th row of the $U^X$ matrix can be recursively obtained from the $(i-1)$-th row and the original values of the time-series $X$. Continuing the previous example, the

\[ S^X_2 = \{10\} \text{ and } S^Y_2 = \{2\}. \]

Therefore, the matrices are

\[ U^X = \begin{bmatrix} 3 & 2 & 4 & 1 \\ 5 & 6 & 5 & 0 \\ 9 & 7 & 0 & 0 \\ 10 & 0 & 0 & 0 \end{bmatrix}, \quad U^Y = \begin{bmatrix} 1 & -1 & 0 & 2 \\ 0 & -1 & 2 & 0 \\ 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 0 \end{bmatrix} \]

The second step consists in calculating the scalar products of the rows of the $U^X$ matrix and the corresponding rows of the $U^Y$ matrix. That is,

\[ \langle U_{1,1}^X, U_{1,1}^Y \rangle = 3 \cdot 1 + 2 \cdot (-1) + 4 \cdot 0 + 1 \cdot 2 = 3 \]
\[ \langle U_{2,2}^X, U_{2,2}^Y \rangle = 5 \cdot 0 + 6 \cdot (-1) + 5 \cdot 2 = 4 \]
\[ \langle U_{3,3}^X, U_{3,3}^Y \rangle = 9 \cdot 0 + 7 \cdot 1 = 7 \]
\[ \langle U_{4,4}^X, U_{4,4}^Y \rangle = 10 \cdot 2 = 20 \]

where $U^X_i$ and $U^Y_i$ are the $i$ rows of the $U^X$ and $U^Y$ matrices, respectively.

Finally, the MUSS kernel is defined as the sum of the above-mentioned scalar products. Therefore,

\[ \text{Kernel}(X, Y) = (3 + 4 + 7 + 20) = 34 \]

3.4.2. Matrix-based scheme

This scheme is based on the observation that each row $U_i^X$ of the matrix $U^X$ can be obtained by multiplying a matrix $M_i$ by the row vector composed of the values of the time-series. That is,

\[ U_i^X = (x_1, \ldots, x_N) \cdot M_i \]

where $M_i = \{a_{ij}\}$ is the $N \times N$ matrix capturing the partial sums of order $i$ and the $A_i = \{a_{ij}\}_{1 \leq j \leq N}$ matrix is defined by $a_{ij} = 1$ for all $I_i$ such that $1 \leq i \leq i$ and $1 \leq j \leq N-i+1$, and 0 otherwise.

The matrix $M_i$ has two well-differentiated blocks, a first block, $A_i$, comprising a diagonal band matrix of width $i$ composed of 1’s and two null triangular matrices of dimensions $(N-i) \times (N-i)$ and a second block, composed of a null matrix of dimensions $N \times (i-1)$. It can be observed that the structure of the $M_i$ matrix is mainly due to the definition of the $i$-order partial sums as the diagonal band of width $i$ represents the $i$-order partial sums for the different subsequence time-series of length $i$. On the other hand, the null matrix of dimension $N \times (i-1)$ is because of the columns of the $U^X$ matrix are equal to 0 when $j > N-i+1$. 

Fig. 2. Example of $k$-order partial sum time-series for $k = 1, 20, 60, 200$ on the time-series of Fig. 1. The reader should notice that as $k$ increases, the smoothed versions of the original time-series become more and more similar. Our kernel is computed as the sum of the similarities over all possible $k$. 

We define
\[
\text{Kernel}(X, Y) = \sum_{i=1}^{N} (U_i^i, U_i^j)
\]
\[
= \sum_{i=1}^{N} U_i^i \cdot (U_i^j)^t
\]
\[
= \sum_{i=1}^{N} (X \cdot M_i) \cdot (Y \cdot M_i)^t
\]
\[
= \sum_{i=1}^{N} X \cdot (M_i \cdot M_i^t) \cdot Y^t
\]
\[
= X \left( \sum_{i=1}^{N} M_i \cdot M_i^t \right) Y^t
\]

where \(X\) and \(Y\) are the row and column vectors composed by the values of the time-series \(X\) and \(Y\), respectively, and \((\cdot)^t\) denotes the transpose. Let \(Z\) be the matrix defined by
\[
Z = \sum_{i=1}^{N} M_i \cdot M_i^t
\]
and thus \(\text{Kernel}(X, Y) = X \cdot Z \cdot Y^t\).

It can be easily seen that the \(Z\) matrix is symmetric since it is the sum of symmetric matrices. Moreover, it is positive semi-definite because it defines an inner product (cf. Eq. (6)). On the other hand, this matrix only depends on the length of the time-series \(N\). Thus, the matrix can be precomputed just once when computing the pair-wise kernel of a whole dataset providing a reduction of the computing time. Most of kernels for time series proposed in the literature cannot be represented by a matrix, which is not depending on the values of the time series \([6,10,34]\). Moreover, the structure of the \(Z = [z_{ij}]_{1 \leq i, j \leq N}\) matrix can be obtained by unwrapping Eq. (8):
\[
z_{ij} = \begin{cases} 
& i \cdot (N-j-1) \quad \text{if } j \geq i \\
& j \cdot (N-i-1) \quad \text{if } i > j 
\end{cases}
\]

Continuing with the previous example, it can be observed that, for all \(i = 1, \ldots, 4\)
\[
U_i^i = (3, 2, 4, 1) \cdot M_i
\]
where \(M_i\) is the identity matrix of dimension \(4 \times 4\) and the remaining matrices and the resulting \(Z\) are
\[
M_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad M_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}
\]
\[
M_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \quad Z = \begin{bmatrix} 4 & 3 & 2 & 1 \\ 3 & 6 & 4 & 2 \\ 2 & 4 & 6 & 3 \\ 1 & 2 & 3 & 4 \end{bmatrix}
\]

It should be noted that a distance metric can be obtained from any positive definite kernel \(\text{Ker}\) using the standard transformation described in the following Eq. [31]:
\[
d(u, v) = \text{Ker}(u, u) + \text{Ker}(v, v) - 2 \cdot \text{Ker}(u, v).
\]
Therefore, when this work refers to the MUSS kernel as a distance it really means the derived distance from the kernel.

4. Results

This section presents the results obtained by the application of the MUSS kernel to forty datasets for measuring the similarity in shifted time-series. Section 4.1 provides a detailed description of all datasets used in the experiments. In Section 4.2 the kernel has been applied to forty time-series to validate its potential for separating classes in shifted time series. Finally, a statistical analysis of the results is reported in Section 4.3.

4.1. Description of datasets

The MUSS kernel has been initially tested on several synthetic datasets from the UCR time-series repository [20] and from data from the 2007 KDD Challenge [19]. In particular, a synthetic dataset derived from each available dataset in the aforementioned repositories has been created in the following way: for each different class existing in a dataset, a time-series belonging to this class is chosen uniformly at random. Then, shifted time-series copies with varying offsets are created from the selected chosen time-series having the same label as the original time-series the copies were created from.

Time-series lengths in our datasets range from 60 to 637, with the average and the median being 282.1 and 272.5, respectively. Computation times are highly sensitive to the time-series length, especially for the DTW algorithm, which is quadratic in this parameter. Relevant information about these datasets is summarized in Tables 1 and 2. The class labels for the test KDD datasets are not available.

4.2. Validation

A statistic based on pair-wise distances has been developed to show how well the MUSS kernel is able to separate classes in time-series. Let \(D\) be a labeled dataset of \(M\) time-series of the same length \(N\). Let \(\tau(X)\) be the class of the time-series \(X \in D\). Then, the SM separation measure is defined as follows:
\[
\text{SM} = \frac{(\text{INTRA} - \text{INTER})}{\text{MAX}}
\]
where INTRA and INTER are the average pair-wise distance of time-series belonging to the same and to different classes, respectively, and MAX is the maximum pair-wise distance over the whole dataset. Namely, let \(A = \{(X, Y) | X, Y \in D, \tau(X) = \tau(Y)\}\) and \(B = \{(X, Y) | X, Y \in D, \tau(X) \neq \tau(Y)\}\). That is, \(A\) is a set of pairs of time series that belong to the same class, and \(B\) is the set of pairs of time-series that belong to different classes. Then,
\[
\text{INTRA} = \frac{1}{|A|} \sum_{(X, Y) \in A} d(X, Y)
\]
\[
\text{INTER} = \frac{1}{|B|} \sum_{(X, Y) \in B} d(X, Y)
\]
\[
\text{MAX} = \max_{X, Y \in D} d(X, Y)
\]

where \(d\) is any distance defined in \(\mathbb{R}^N \times \mathbb{R}^N\) and \(N\) is the length of the time-series in \(D\). Observe that the higher the value for this measure, the better the separation among classes obtained by distance \(d\) is. A positive value of the SM measure means that, on average, time-series belonging to the same class are closer than the time-series of different classes. A negative value of the SM measure means the opposite, that is, pairs of series of different classes tend to be closer than the pairs of the same class. Notice that for negative or values of the SM measure close to 0, any distance-based classifier such as Nearest Neighbors is doomed to fail.

Table 3 presents a comparison of the separation statistic and computation time of the following distances: the Euclidean distance, the one derived from the MUSS kernel (which we call MUSS distance), and the DTW distance. The comparison is over the synthetic 40 datasets from UCR and KDD repositories. The computation times have
been obtained with a laptop of 8 GB of RAM and an Intel Core i7 processor running at 2.7 GHz. The distance that better separates the existing classes for each dataset is marked in bold style. It can be seen that the average of the separation measure for the MUSS kernel is higher than that of the Euclidean distance and similar to DTW distance for both classes.

We further analyze some of the rows of Table 3, in particular, the rows corresponding to the FISH and KDD-03 datasets. These two rows represent the opposite behavior of the SD measure: for FISH all distances have negative values in the SM measure, while for KDD-03 all of them perform quite well. We try to explain why this should be the case in what follows. Remember that in order to generate the synthetic datasets, we chose one representative of each class at random and then created circularly shifted copies from each representative as time-series of the same class. Depending on the chosen representatives, some of the resulting synthetic datasets are thus inconsistent in their labels and therefore all distance measures perform poorly. Fig. 3 shows one example of this (poor behavior in FISH), and Fig. 4 shows a case of good behavior across all distances (KDD-03).

Table 4 further summarizes Table 3, where the reader can observe that the behavior of the MUSS distance is similar to that of DTW on average (1.75 versus 1.70), and both outperform the Euclidean distance (1.75 and 1.70 versus 2.55). From this table, it can be noticed that the MUSS kernel reaches the highest rank in 14 datasets, the second position and third positions in 21 and 4 datasets, respectively, the DTW distance reaches the highest rank in 24 datasets, the second position for 4 datasets (kdd13, kdd16, Coffee, Face (Four)) and the third position for the 12 remaining datasets, and finally, the Euclidean distance obtains the highest ranking just for 2 datasets (kdd13 and FISH), the second position for 14 datasets and the third position for the remaining 24 datasets. The average ranking for each distance is summarized in Table 4. It can be observed that DTW and the MUSS kernel present similar behavior on average.

Table 5 shows the wins matrix for pairs of distances over the 40 datasets. That is, in how many datasets a distance separates better than another distance. It should be read as follows: if row $i$ and column $j$ contains number $m$, then distance $i$ has beaten distance $j$ a total of $m$ times. It can be noticed that the MUSS distance beats the Euclidean distance in 34 datasets, and beats the DTW distance in 16 datasets. The DTW distance wins in 28 datasets to the Euclidean distance and in 24 datasets to the MUSS kernel, and finally, the Euclidean distance wins in 6 and 12 datasets to the MUSS and DTW distances, respectively.

4.3 Statistical analysis

A statistical analysis has been conducted to evaluate the significance of the MUSS kernel, following the non-parametric procedures discussed in [13]. Non-parametric tests have to be selected because the conditions of normality and homoscedasticity are not met. All tests applied in this work have been obtained by using the open-source platform KEEL [2].

Friedman and Iman–Davenport (ID) tests have been applied to assess whether there are global differences in the separation measure obtained by the three distances compared. The $p$-values obtained by both tests at the level of significance $\alpha = 0.05$ are $1.12 \times 10^{-4}$ for the Friedman test, and $4.25 \times 10^{-5}$ for the ID test. As the $p$-values obtained from both of the tests are lower than the level of significance considered, it can be stated that there exist significant differences among the results obtained by three distances and a post-hoc statistical analysis is required.

The Holm and Shaffer tests have been applied to perform pairwise comparisons. Table 6 shows the sorted $p$-values obtained by all comparisons for two levels of significance ($\alpha = 0.05$ and $\alpha = 0.10$). Both of the tests allow concluding that the MUSS distance is better than the Euclidean distance and similar to DTW distance for both levels of significance, as the two tests reject the hypotheses for the MUSS kernel and DTW distance versus Euclidean distance but do not reject the null hypotheses for the MUSS kernel versus DTW distance.

5. Time-series classification

In our second experiment, we use Support Vector Machines to perform classification using three different kernels: our MUSS kernel, the GA-DTW kernel based on global alignments in [8], and a linear kernel. We compare the prediction accuracy achieved by these three kernels over the 20 datasets of the UCR repository [20].
### Table 3
Separation measure among classes and computing times when using several distances.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Separation measure</th>
<th>CPU times (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Euclidean distance</td>
<td>MUSS distance</td>
</tr>
<tr>
<td>kdd01</td>
<td>0.004</td>
<td>0.010</td>
</tr>
<tr>
<td>kdd02</td>
<td>0.076</td>
<td>0.128</td>
</tr>
<tr>
<td>kdd03</td>
<td>0.499</td>
<td>0.637</td>
</tr>
<tr>
<td>kdd04</td>
<td>0.388</td>
<td>0.670</td>
</tr>
<tr>
<td>kdd05</td>
<td>0.193</td>
<td>0.389</td>
</tr>
<tr>
<td>kdd06</td>
<td>0.013</td>
<td>0.200</td>
</tr>
<tr>
<td>kdd07</td>
<td>0.159</td>
<td>0.229</td>
</tr>
<tr>
<td>kdd08</td>
<td>0.131</td>
<td>0.208</td>
</tr>
<tr>
<td>kdd09</td>
<td>-0.037</td>
<td>-0.041</td>
</tr>
<tr>
<td>kdd10</td>
<td>0.012</td>
<td>0.109</td>
</tr>
<tr>
<td>kdd11</td>
<td>0.017</td>
<td>0.000</td>
</tr>
<tr>
<td>kdd12</td>
<td>0.364</td>
<td>0.387</td>
</tr>
<tr>
<td>kdd13</td>
<td>-0.028</td>
<td>-0.009</td>
</tr>
<tr>
<td>kdd14</td>
<td>0.083</td>
<td>0.095</td>
</tr>
<tr>
<td>kdd15</td>
<td>-0.048</td>
<td>-0.007</td>
</tr>
<tr>
<td>kdd16</td>
<td>0.019</td>
<td>0.044</td>
</tr>
<tr>
<td>kdd17</td>
<td>-0.036</td>
<td>0.004</td>
</tr>
<tr>
<td>kdd18</td>
<td>0.157</td>
<td>0.058</td>
</tr>
<tr>
<td>kdd19</td>
<td>-0.009</td>
<td>-0.024</td>
</tr>
<tr>
<td>kdd20</td>
<td>0.089</td>
<td>0.147</td>
</tr>
<tr>
<td>50 Words</td>
<td>0.155</td>
<td>0.196</td>
</tr>
<tr>
<td>Adiac</td>
<td>-0.042</td>
<td>-0.040</td>
</tr>
<tr>
<td>Beef</td>
<td>0.696</td>
<td>0.894</td>
</tr>
<tr>
<td>CBF</td>
<td>0.150</td>
<td>0.311</td>
</tr>
<tr>
<td>Coffee</td>
<td>-0.015</td>
<td>0.111</td>
</tr>
<tr>
<td>ECG</td>
<td>0.045</td>
<td>0.054</td>
</tr>
<tr>
<td>FISH</td>
<td>-0.004</td>
<td>0.008</td>
</tr>
<tr>
<td>Face (all)</td>
<td>0.007</td>
<td>0.110</td>
</tr>
<tr>
<td>Face (four)</td>
<td>-0.005</td>
<td>0.016</td>
</tr>
<tr>
<td>Gun-point</td>
<td>0.136</td>
<td>0.113</td>
</tr>
<tr>
<td>Lighting-2</td>
<td>0.126</td>
<td>0.152</td>
</tr>
<tr>
<td>Lighting-7</td>
<td>0.137</td>
<td>0.268</td>
</tr>
<tr>
<td>OSU leaf</td>
<td>0.229</td>
<td>0.346</td>
</tr>
<tr>
<td>Olive oil</td>
<td>-0.063</td>
<td>-0.041</td>
</tr>
<tr>
<td>Swedish leaf</td>
<td>0.104</td>
<td>0.144</td>
</tr>
<tr>
<td>Trace</td>
<td>0.300</td>
<td>0.303</td>
</tr>
<tr>
<td>Two patterns</td>
<td>0.113</td>
<td>0.165</td>
</tr>
<tr>
<td>Synth. control</td>
<td>0.103</td>
<td>0.293</td>
</tr>
<tr>
<td>Wafer</td>
<td>0.126</td>
<td>0.172</td>
</tr>
<tr>
<td>Yoga</td>
<td>-0.000</td>
<td>0.073</td>
</tr>
<tr>
<td>Average</td>
<td>0.114</td>
<td>0.168</td>
</tr>
</tbody>
</table>

**Fig. 3.** The left plot shows the 7 representatives chosen for each of the 7 classes found in the FISH dataset. The right plot shows in red are the 9 copies made out of the first representative, and in blue are the 9 copies made out of the second representative. Since the representatives are, in fact, quite alike, it turns out that the copies are further apart from their representatives than from other copies of other representatives, and therefore any reasonable distance measure is going to have low scores in the SM measure. Plots are best seen in color. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)
The results in this section show that our kernel is able to achieve good results for general time-series (not necessarily shifted). We have used the implementation LIBSVM [5] of SVM due to its wide acceptance in the literature. Code for the GA-DTW kernel has been taken from.1

Table 7 shows the accuracy of the three methods. The last row presents the average and the standard deviation in brackets for all datasets. It can be appreciated that the MUSS kernel is better than the linear kernel and similar to the GA-DTW kernel having in addition the lowest standard deviation.

On the other hand, Table 8 reports the time (in seconds) needed for the kernel computations. The MUSS kernel proposed here is much faster than the GA-DTW kernel and as fast as the linear kernel. In fact, the GA-DTW kernel is not applicable to high-dimensional real-world problems that require the computation of the kernel for many instances.

These results show that our MUSS kernel achieves comparable accuracy to the DTW-based kernel but at a much lower computational cost.

6. Real-world applications: ozone concentration in atmosphere and electricity demand classification

Finally, we present results in two real-world applications: atmospheric pollutants (ozone), and electricity demand. The pattern recognition in ozone time-series data is an important task as it allows governments to activate alert protocols and implement good environmental policies if high ozone concentration levels are predicted. On the other hand, electricity-producing companies are interested in predicting spikes of demand in order to schedule the energy production to maximize their profit.

Ozone time-series have been retrieved from a meteorological station placed in the outskirts of Seville city (Spain), providing 312 times series composed of 168 hourly records each. The dataset is classified into two classes corresponding to high and low ozone level periods (165 and 147 time-series, respectively). The time-series have been split into a training set of 218 time-series, and a test set of 94 time-series, preserving the proportion between two existing classes.

Electricity demand time-series from 2008 to 2012 for the Spanish electricity market have been collected [24], providing 261 times series composed of 168 hourly records each. Four datasets, called Demand-2, Demand-3, Demand-4 and Demand-5, have been used depending on the necessary number of classes that an electrical engineering can consider useful to determine the behavior of the demand. The time-series data have been split in training set (182, 182, 181 and 181 time-series for each one of 4 datasets, respectively) and test set (79, 19, 80 and 80, respectively) preserving the proportion among existing classes.

---

1 http://www.iip.iit.i.kyoto-u.ac.jp/member/cuturi/Code/DTWkernel.cpp.
Tables 8 and 10 show the percentage of time-series correctly classified in the test sets and the time in seconds obtained by the application of the SVM classifier for all real-world datasets when using several kernels. The last row presents the average and the standard deviation in brackets for all datasets. It can be observed that the MUSS kernel presents the better results in accuracy for all datasets. Moreover, the MUSS kernel is 440 times faster than the GA-DTW kernel approximately and it provides a mean error of classification by 3% and 5% lower than the remaining kernels.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>GA-DTW kernel</th>
<th>Linear kernel</th>
<th>MUSS kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td>50 Words</td>
<td>3820.55</td>
<td>1.54</td>
<td>2.86</td>
</tr>
<tr>
<td>Adiac</td>
<td>1183.33</td>
<td>0.85</td>
<td>1.45</td>
</tr>
<tr>
<td>Beef</td>
<td>52.13</td>
<td>0.03</td>
<td>0.17</td>
</tr>
<tr>
<td>CRF</td>
<td>77.81</td>
<td>0.14</td>
<td>0.18</td>
</tr>
<tr>
<td>Coffee</td>
<td>16.49</td>
<td>0.02</td>
<td>0.07</td>
</tr>
<tr>
<td>ECG</td>
<td>23.57</td>
<td>0.05</td>
<td>0.08</td>
</tr>
<tr>
<td>FISH</td>
<td>1654.97</td>
<td>0.38</td>
<td>1.34</td>
</tr>
<tr>
<td>Face (all)</td>
<td>3297.67</td>
<td>2.98</td>
<td>4.61</td>
</tr>
<tr>
<td>Face (four)</td>
<td>50.72</td>
<td>0.04</td>
<td>0.10</td>
</tr>
<tr>
<td>Gun-point</td>
<td>33.37</td>
<td>0.05</td>
<td>0.08</td>
</tr>
<tr>
<td>Lighting-2</td>
<td>382.17</td>
<td>0.09</td>
<td>0.61</td>
</tr>
<tr>
<td>Lighting-7</td>
<td>132.32</td>
<td>0.06</td>
<td>0.23</td>
</tr>
<tr>
<td>OSULeaf</td>
<td>2127.90</td>
<td>0.50</td>
<td>1.50</td>
</tr>
<tr>
<td>Olive oil</td>
<td>75.73</td>
<td>0.03</td>
<td>0.24</td>
</tr>
<tr>
<td>Swedish leaf</td>
<td>1210.00</td>
<td>1.31</td>
<td>1.96</td>
</tr>
<tr>
<td>Trace</td>
<td>194.04</td>
<td>0.10</td>
<td>0.29</td>
</tr>
<tr>
<td>Two-patterns</td>
<td>12,666.86</td>
<td>11.64</td>
<td>17.76</td>
</tr>
<tr>
<td>Synth. control</td>
<td>84.69</td>
<td>0.27</td>
<td>0.40</td>
</tr>
<tr>
<td>Wafer</td>
<td>26,289.65</td>
<td>20.26</td>
<td>30.52</td>
</tr>
<tr>
<td>Yoga</td>
<td>28,925.00</td>
<td>6.20</td>
<td>11.18</td>
</tr>
<tr>
<td>Average</td>
<td>4114.95 (426.96)</td>
<td>2.33 (0.25)</td>
<td>3.78 (0.38)</td>
</tr>
</tbody>
</table>

7. Conclusions

In this paper we have presented the MUSS kernel for time-series data and its associated distance metric. Initial experiments show promise in detecting similarity between time-series. The MUSS kernel has been compared to the Euclidean distance as a reference distance and the DTW distance as one of the most competitive distances that exist in the literature. Also, the MUSS kernel has been used in conjunction with the SVM classifier to time-series classification and compared with the GA-DTW kernel [8] and the linear kernel. The kernel has shown to be efficient in separating different classes in time-series from well-known repositories, and also, it has been successfully applied to real-world time-series. In particular, a low error for the classification of the ozone in low and high concentrations in the atmosphere and classification of the electricity demand has been obtained.

For future work, we would like to generalize our MUSS kernel to multivariate time-series and time-series that differ in length. Moreover, we would like to adapt our ideas so that they can be used in a stream setting where time-series keep growing unboundedly.

Acknowledgments

This work is partially supported by the Spanish Ministry of Science and Technology contracts TIN2011-27479-C04-03 and TIN2011-28956-C02, by the Generalitat de Catalunya 2009-SGR-1428 (LARCA), by the Junta de Andalucía P12-TIC-1728, by the Pablo de Olavide University APB813097 and by the EU PASCAL2 Network of Excellence (FP7-ICT-216886). Part of this work was done while A.T. and M.A. were at Columbia University in New York and University of California in San Diego, supported in part by grant BE-DGR 2011 of the Generalitat de Catalunya. We are grateful for the fruitful discussions held with Charles Elkan, Eamonn Keogh and Ramón Huerta during A.T. and M.A.’s stay at UCSD. Finally, we also acknowledge constructive comments given by anonymous reviewers that undoubtedly helped to improve this paper.

References