On-line Elastic Similarity Measures for Time Series

Izaskun Oregi[a], Aritz Pérez[b], Javier Del Ser[a,b,c], Jose A. Lozano[b,d]

[a]*TECNALIA, 48160 Derio, Spain*
[b]*Basque Center for Applied Mathematics (BCAM), 48009 Bilbao, Spain*
[c]*Department of Communications Engineering, University of the Basque Country UPV/EHU, 48013 Bilbao, Spain*
[d]*Department of Computer Science and Artificial Intelligence*
University of the Basque Country UPV/EHU, 20018 Donostia/San-Sebastián, Spain

ABSTRACT

The way similarity is measured among time series is of paramount importance in many data mining and machine learning tasks. For instance, Elastic Similarity Measures are widely used to determine whether two time series are similar to each other. Indeed, in off-line time series mining, these measures have been shown to be very effective due to their ability to handle time distortions and mitigate their effect on the resulting *distance*. In the on-line setting, where available data increase continuously over time and not necessary in a stationary manner, stream mining approaches are required to be fast with limited memory consumption and capable of adapting to different stationary intervals. In this sense, the computational complexity of Elastic Similarity Measures and their lack of flexibility to accommodate different stationary intervals, make these similarity measures incompatible with the requirements mentioned. To overcome these issues, this paper adapts the family of Elastic Similarity Measures – which includes Dynamic Time Warping, Edit Distance, Edit Distance for Real Sequences and Edit Distance with Real Penalty – to the on-line setting. The proposed adaptation is based on two main ideas: a forgetting mechanism and the incremental computation. The former makes the similarity consistent with streaming time series characteristics by giving more importance to recent observations, whereas the latter reduces the computational complexity by avoiding unnecessary computations. In order to assess the behavior of the proposed similarity measure in on-line settings, two different experiments have been carried out. The first aims at showing the efficiency of the proposed adaptation, to do so we calculate and compare the computation time for the elastic measures and their on-line adaptation. By analyzing the results drawn from a distance-based streaming machine learning model, the second experiment intends to show the effect of the forgetting mechanism on the resulting similarity value. The experimentation shows, for the aforementioned Elastic Similarity Measures, that the proposed adaptation meets the memory, computational complexity and flexibility constraints imposed by streaming data.

1. Introduction

In recent years, time series have extended to many scientific and social domains such as medicine, manufacturing industry, energy consumption and geophysics, among others (Fard et al., 2017; Villar-Rodriguez et al., 2017; Yu and Chen, 2007). In order to extract valuable information or respond to the specific needs and challenges of these areas of application, the scientific

e-mail: izaskun.oregui@tecnalia.com (Izaskun Oregi), aperez@bcamath.org (Aritz Pérez), javier.delser@tecnalia.com (Javier Del Ser), ja.lozano@ehu.eus (Jose A. Lozano)

community has made a great effort to develop different time series mining and machine learning models (Aghabozorgi et al., 2015; Esling and Agon, 2012; Lotte et al., 2007; Mondal et al., 2018). When analyzing this particular type of data, distance-based classification, clustering, anomaly detection and motif discovery algorithms (Liao, 2005; Jeong et al., 2011; Aggarwal and Reddy, 2016; Panagiotakis et al., 2018; Mueen and Chavoshi, 2015) – have played a central role. In these approaches the choice of the similarity measure is essential for their performance. Hence, in recent years a diverse collection of similarity measures has been proposed in the literature to further enrich the existing portfolio (Parziale et al., 2018; Cinti et al., 2017; Oregi et al., 2017a; Zhou and De la Torre, 2016; Zhao and Itti, 2018).

To measure similarity among sequences, L_p-norm distances and Elastic Similarity Measures (ESMs) are the most commonly used in distance-based learning models. On the one hand, L_p-norm distances such as Manhattan distance ($p = 1$), Euclidean distance ($p = 2$) and Chebyshev distance ($p = \infty$) are characterized by their easy implementation and low computational cost. However, they cannot manage sequences with different lengths nor can they handle mismatches/gaps between time series. On the other hand, the family of ESMs – including, among others, Dynamic Time Warping (DTW) and Edit-based measures of similarity – are able to shrink or stretch the time axis to find the best alignment between the time series and obtain the smallest distance between them. Unfortunately, the computational complexity of ESMs is quadratic with the time series length, and their implementation is not as straightforward as that of non-elastic L_p-norms. Nonetheless, results in (Ding et al., 2008; Lines and Bagnall, 2015) – where an extensive set of time series supervised classification experiments were performed – showed that the use of ESM tends to outperform the Euclidean distance based classifiers. Standing on this empirical evidence, ESM have consolidated as the reference measures of similarity for distance-based classification and clustering tasks over time series data (Rakthanmanon et al., 2012; Petitjean et al., 2011; Yu et al., 2011).

Up to this point, we have assumed that data are permanently stored in memory, enabling algorithms to access them any number of times. However, in many real-world applications this assumption does not hold. Instead, data are produced at high speed creating massive amounts of samples that must be analyzed as fast as possible, and then discarded, so as to deal with memory limitations. In addition to the (time and memory) computational complexity, the structure of arriving samples may change i.e., they may no longer be stationary. Consequently conventional procedures for off-line learning are not valid for the on-line analysis of evolving data. In this sense, the research community is currently proposing new models that might efficiently deal with the challenges arising from streaming data (Gama, 2010; Krempl et al., 2014). Regarding shape-based models in the on-line setting, L_p-norm and correlation-based distances have become dominant measures of similarity between time series (Beringer and Hüllermeier, 2006; Rodrigues et al., 2008; Yeh et al., 2007). The reason why they are so often used might rely on the fact that L_p-norm and correlation-based distances provide an exact incremental expression with minimal memory consumption and low computational efforts. In contrast, the computational complexity of ESMs makes the computation of these measures unaffordable when dealing with long time series.

Considering the virtues of ESMs, a straightforward question is how to adapt ESMs to streaming time series so they can be used in any on-line distance-based machine learning model. This manuscript aims at addressing this paradigm by proposing an On-line Elastic Similarity Measure (OESM), which hinges on two essential characteristics: the first is a forgetting mechanism, which allows OESM to deal with different stationary intervals, whereas the second corresponds to a spotted property of elastic similarities that is exploited to compute the measure within a constant time and fixed amount of consumed memory. Two experiments will be conducted to examine different aspects of the proposed OESM. First, we will test the computational efficiency of the OESM in terms of complexity. Then we will study the effect of the forgetting mechanism by analyzing the results drawn by a 1-NN classifier over transitions between stationary stream intervals. Built upon preliminary findings reported in (Oregi et al., 2017b), this work extends this previous work by proposing a complete framework for computing on-line similarity measures that considers any kind of inner cost functions and similarity-based on-line learning scenarios. We describe in depth the concepts behind the design of the proposed OESM, and discuss its performance under different configurations over a much broader set of experiments.

The remainder of the paper is organized as follows: Section 2 provides a review and background information on the conventional ESMs. Section 3 gives a detailed description of the proposed OESM. Section 4 describes the experimentation and Section 5 discusses the obtained results. Finally, in Section 6 we summarize the main contributions and outline future research lines derived from this work.

2. A Review on Elastic Similarity Measures

The mathematical formulation of ESMs is expressed as an optimization problem. As we have already mentioned, the essence of ESM resides in the ability to stretch or compress the time axis in order to minimize the influence of local time shifts in the resulting measure of similarity. In doing so, ESMs are based on a restricted number of time series alignments or paths to each of which a *punctuation* is allocated. These path punctuations (or path weights) represent the degree of dissimilarity of each proposed alignment. So, the optimization problem seeks to find, among different time series alignments, that with minimum weight.

2.1. ESM definition

Consider two time series $X^m = (x_1, ..., x_i, ..., x_m)$ and $Y^n = (y_1, ..., y_j, ..., y_n)$, where m and n represent the number of points in each sequence, and let a **path** (p) be the sequence of $(i, j) \in [1, m] \times [1, n]$ pairs that represent the alignment between x_i and y_j observations. A path, $p = \{(i_1, j_1), ..., (i_Q, j_Q)\}$ of length Q, is an **allowed path** if:

i) $(i_1, j_1) = (1, 1)$,
ii) $(i_Q, j_Q) = (m, n)$ and
iii) $(i_q - i_{q-1}, j_q - j_{q-1}) \in \{(1, 0), (1, 1), (0, 1)\}$ for $q = 2, ..., Q$.

That is, p is an allowed path if it is an ordered sequence of alignments going from $(1, 1)$ to (m, n) concatenating $\uparrow$, $\nearrow$, $\rightarrow$ unitary steps.

Let $c(x_i, y_j)$ be the **cost function**, which quantifies the dissimilarity between x_i and y_j, then

$$w(p) = \sum_{(i,j)\in p} c(x_i, y_j) \tag{1}$$

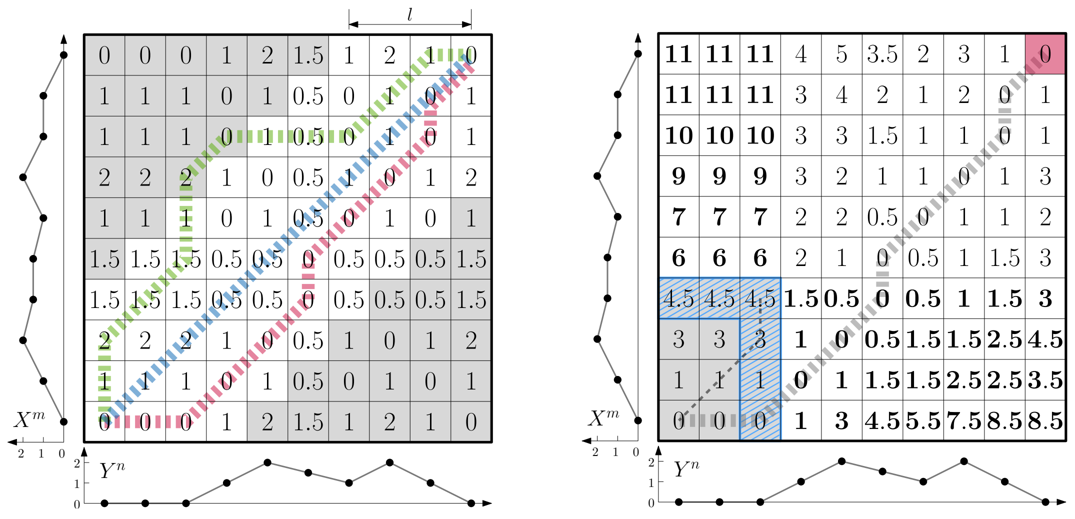
is the weight of an allowed path.

Definition 1. *Given a cost function, the ESM between X^m and Y^n time series is defined as:*

$$D(X^m, Y^n) = \min_{p\in\mathcal{P}} w(p), \tag{2}$$

where $\mathcal{P}$ is the set of allowed paths in the $[1, m] \times [1, n]$ lattice.

When clear from the context, we will hereafter use $D_{m,n}$ to denote the elastic measure of similarity between two sequences of length m and n, that is $D_{m,n} = D(X^m, Y^n)$. Similarly, we will use $c_{i,j}$ to denote the cost function, $c(x_i, y_j)$. Besides, the path $p^* \in \mathcal{P}$ satisfying the condition, $D_{m,n} = \sum_{(i,j)\in p^*} c_{i,j}$, will be referred to as **optimal path**.



(a) Time series alignment and Sakoe-Chiba Band.

(b) Recursive computation.

Fig. 1. DTW similarity measure between two generic time series, X^m and Y^n, with $m = n = 10$. The cost function corresponds to the Euclidean distance, $c(x_i, y_j) = \|x_i - y_j\|_2$. (a) Different examples of allowed paths over $[1, 10] \times [1, 10]$ lattice, where cell values show the cost function and grey areas the Sakoe-Chiba constraints. Under Sakoe-Chiba, the green path is forbidden, consequently it is excluded from the search space $\mathcal{P}$. (b) $F^{4,3}$ frontier (blue area) over 10×10 dimension measure matrix. The dashed lines illustrate the optimal path to $D_{4,3}$ and $D_{10,10}$.

Figure 1.a shows, for two time series with $m = n = 10$ points, a small sample of allowed paths (blue, green and red dashed lines), where the value in the lattice cells corresponds to the Euclidean distance. If we compute the weight of all $p \in \mathcal{P}$, it is easy to see that the optimal path corresponds to the red path, for which $D_{10,10} = 0$.

2.2. *ESM computation*

Because the number of allowed paths in $\mathcal{P}$ grows exponentially with the size of the time series (m and n), brute force is not the best option to consider when solving Equation (2). Instead, we can make use of dynamic programming techniques to solve the optimization problem (Bellman and Dreyfus, 2015). As such, the ESM is computed using the recursion:

$$D_{m,n} = c_{m,n} + \min\{D_{m-1,n}, D_{m-1,n-1}, D_{m,n-1}\}, \tag{3}$$

where $D_{0,0} = 0$ and $D_{0,j} = D_{i,0} = \infty$ for $i = 1, 2, ..., m$ and $j = 1, 2, ..., n$. As a result, the computational complexity is reduced to $O(m \times n)$. This computation technique is shown in Figure 1.b, where each cell, (i, j), corresponds to $D_{i,j}$, the red cell corresponds to $D_{m,n}$ and the dashed line illustrates the optimal path. When using Expression (3), p^* can be obtained by going backwards across the matrix.

Even though the recursive equation reduces the running time of the ESM, it is still unaffordable in many practical situations where time series are significantly large. In order to address this problem, several techniques have been proposed either to reduce the number of ESM computations in learning methods (Keogh, 2002; Lin et al., 2003; Sakurai et al., 2005) or to speed up the ESM computation itself (Keogh and Pazzani, 2000; Salvador and Chan, 2007) refining the resolution of the time series at hand.

In the line of relaxing the complexity of ESM, two constraint-based methods have been especially notable in the literature: the Sakoe-Chiba (Sakoe and Chiba, 1978) and the Itakura parallelogram (Itakura, 1975) strategies. In order to reduce the number of operations required in the ESM computation, these techniques limit the number of paths in $\mathcal{P}$ simply by adding some constraints to its definition. Actually, these constraints discard the subset of paths of higher length and (on average) with higher weights. In particular, the allowed paths considered by these strategies are composed of (i_q, j_q) pairs that must satisfy $|i_q - j_q| \leq g(i, j)$ for all $q = 1, ..., Q$, where $g : \mathbb{N}^2 \to \mathbb{N}$ is the constraint function. In the case of Itakura parallelogram, $g(i, j)$ is defined in such a way that the lattice $[1, m] \times [1, n]$ is limited to a parallelogram with $g(1, 1) = g(m, n) = 0$. In the case of Sakoe-Chiba $g(i, j) = l$ for any (i, j) point in the lattice $[1, m] \times [1, n]$. Note that, in such a way, $g(i, j)$ restricts the lattice to a band around the lattice diagonal, for which l is referred to as the **band width**. An illustrative example of the Sakoe-Chiba constraints is shown in Figure 1.a for $l = 3$. Either the Sakoe-Chiba band or the Itakura parallelogram compute the ESM in linear time with respect to the time series length. However, we would like to point out that this type of constraints can also improve the performance of ESM. This fact was proven in (Ratanamahatana and Keogh, 2004), where the effect of the Sakoe-Chiba band was analyzed by measuring the accuracy of 1-NN classifiers. By varying the value of the band width, the authors showed that narrow constraints improve the performance of the DTW-based classifiers over a benchmark of seven time series datasets. In this work we focus on the Sakoe-Chiba strategy, because it is less restrictive than Itakura's parallelogram, although both techniques can be utilized in the on-line setting.

2.3. ESM set and cost functions

Although the ESM family is a broad set of proposed measures of similarity, in this paper we will focus on the most commonly used alternatives, namely, the Dynamic Time Warping (DTW (Berndt and Clifford, 1994)), the Edit Distance (EDIT (Marzal and Vidal, 1993)), the Edit Distance for Real Sequences (EDR (Chen et al., 2005)) and the Edit Distance with Real penalty (ERP (Chen and Ng, 2004)). The ground differences among these ESMs reside in the data type they can handle, as well as in the definition of the cost function (see Table 1):

1. For the well-known DTW, the cost function corresponds to the Euclidean Distance (ED), or alternatively to the squared ED. In consequence, the DTW is utilized to calculate the similarity between two numerical time series.

2. From a different perspective, the EDIT distance quantifies the similarity between two character sequences by counting the number of operations needed to convert one string into another. As shown in Table 1, the penalty for an operation is usually set to 1.

3. Given two numerical time series, EDR and ERP also count the number of operations needed to convert two sequences, where the equality holds whenever the absolute difference between the two time series falls within the threshold $\delta > 0$. Given the definitions in Table 1, note that:

   - just as in the EDIT distance, the cost of each EDR operation is 1,
   - the penalty for ERP is, however, proportional to the Euclidean distance.

Table 1. Definition of the ESMs cost function, $c_{i,j} = c(x_i, y_j)$**.**

Elastic Similarity Measure	Cost Function	Data Type
Dynamic Time Warping (DTW)	$c(x_i, y_j) = \|x_i - y_j\|_2$	Numerical
Edit Distance (EDIT)	$c(x_i, y_j) = \begin{cases} 0 & \text{if } x_i = y_j \\ 1 & \text{otherwise} \end{cases}$	Character
Edit Distance for Real sequences (EDR)	$c(x_i, y_j; \delta) = \begin{cases} 0 & \text{if } \lvert x_i - y_j \rvert \leq \delta \\ 1 & \text{otherwise} \end{cases}$	Numerical
Edit Distance with Real Penalty (ERP)	$c(x_i, x_j; \delta) = \begin{cases} 0 & \text{if } \lvert x_i - y_j \rvert \leq \delta \\ \lvert x_i - y_j \rvert & \text{otherwise} \end{cases}$	Numerical

3. On-line Elastic Similarity Measures

In this section, we introduce the On-line Similarity Measure (OESM). Just as in the off-line version, the OESM is expressed as an optimization problem. To make its computation compatible with on-line setting requirements – fast execution and limited memory consumption –, we propose the use of an incremental computation technique. Based on dynamic programming principles similar to those in (Zhou et al., 2013), the incremental computation avoids unnecessary computations thanks to a set of thoroughly selected and stored measurements we call *frontier*. The adaptation – and so the definition – of ESM to on-line settings includes a memory function. By under-weighting the contribution of past events, the memory function is set to forget the past and fit OESM to recent events. Moreover, this forgetting mechanism can be modified depending on the characteristics of the problem at hand. In this paper, two problem scenarios are considered: the **batch pattern** and the **on-line pattern**. The first illustrates the case where one of the time series (e.g., X^m) is stored in memory and only Y^n sequence receive new examples, as in streaming classification or early classification problems (Mori et al., 2017). The second corresponds to the case where both, X^m and Y^n receive new data points, as occurs in clustering tasks for streaming time series (Rodrigues et al., 2008).

In what follows, the memory function and the incremental computation are introduced and described, with an emphasis placed on the required adaptations to treat on-line or batch pattern scenarios.

The OESM has been made available at `http://bitbucket.org/izaskun_oregui/ESM`. The code has been written in Python 2.7 and can be used to solve either the batch or the on-line pattern scenario. The implemented cost functions correspond to DTW, EDR, ERP and EDIT distances.

3.1. On-line setting and OESM definition

In the off-line setting, one of the most extended assumptions in distance-based classification and clustering algorithms is that every time series (either in the learning or the prediction stage) belong to a certain group characterized by an underlying time series shape or structure. However, this assumption may not hold in on-line learning scenarios, where streaming time series can switch from one structure to another over time. In order to make OESM consistent with streaming time series evolution, we propose a simple yet effective forgetting mechanism that mitigates, by means of a memory function, the contribution of earlier time series alignments in Equation (1).

Consider X^m and Y^n time series and a path, $p \in \mathcal{P}$. In the on-line setting, the weight of a path p is redefined to be

$$w_\rho(p) = \sum_{(i,j)\in p} \rho^{f(i,j)} c_{i,j}, \tag{4}$$

where $f : \mathbb{N}^2 \to \mathbb{R}$ is the memory function, and $\rho \in (0, 1]$ the memory parameter. Note that the definition given above is just a generalization of Equation (1). Indeed, if $\rho = 1$, the weight of the path in Equation (4) is equivalent to Equation (1).

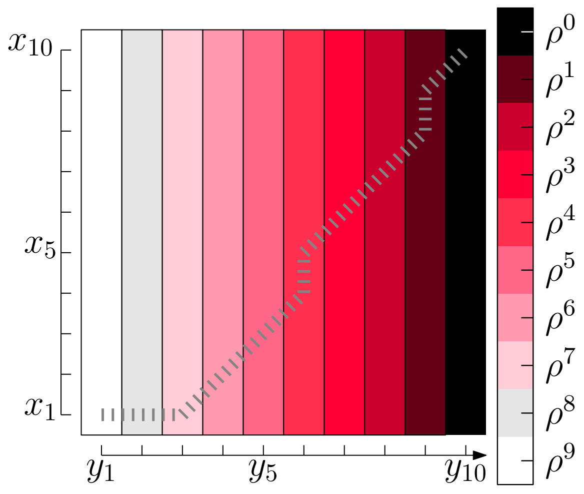
The formal definition of the OESM is given in Definition 2. The reader should note that the adaptation of the ESM to on-line setting is produced just by replacing Equation (4) in (2).

Definition 2. *(On-line Elastic Similarity Measure) Let X^m and Y^n be two time series and $w_\rho(p)$ the weight of a path p for a given memory parameter $\rho \in (0, 1]$. The OESM between Y^n and X^m time series is defined as:*

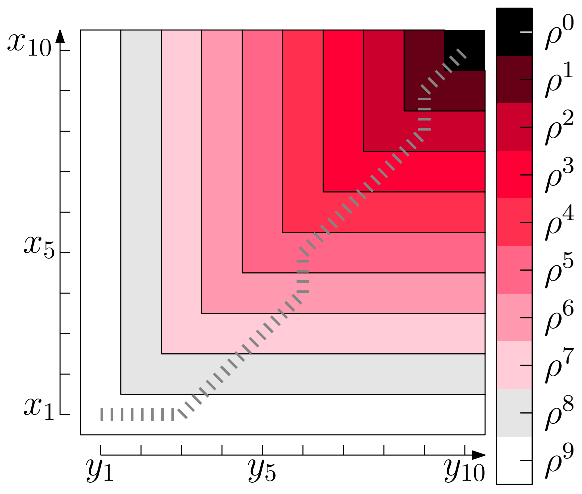
$$D_\rho(X^m, Y^n) = \min_{p\in\mathcal{P}} w_\rho(p), \tag{5}$$

where $\mathcal{P}$ is the set of allowed paths.

Next we introduce the adaptation of the memory function to the particularities of the batch and on-line pattern scenarios.



(a) Batch Pattern.



(b) On-line Pattern.

Fig. 2. Representation of the memory function over the $[1,10]\times[1,10]$ lattice given two time series $X^{10} = (x_1,\dots,x_i,\dots,x_{10})$ and $Y^{10} = (y_1,\dots,y_j,\dots,y_{10})$, for the batch (left) and on-line pattern scenarios (right). The color shapes represent the cells sharing the same values of $\rho^{f(i,j)}$. Similarly to Figure 1, the gray dashed line denotes the warping path p.

Batch pattern scenario

As in Definition (5), let Y^n be a streaming time series and X^m a sequence permanently stored in memory. In this scenario, the memory function in Equation (5) is given by:

$$f(i,j) = n - j. \tag{6}$$

An example of the function is shown in Figure 2.a, where coloured bands illustrate all (i, j) pairs in the lattice $[1,10]\times[1,10]$ sharing the same $f(i, j)$ value from $(10,10)$. Note that, by modulating the cost function with Equation (6), the last alignments contribute more significantly to the weight of a given path than those that are far from the current sample, (m,n) (see Equation (4)). That is, the contribution of $c(i, j)\rho^{f(i,j)}$ to the resulting similarity value depends on the difference between n and j. Since in the batch pattern scenario X^m does not evolve over time, note that the proposed forgetting mechanism works with the evolution of Y^n (see Figure 2), enabling OESM to forget its past.

On-line pattern scenario

In the on-line pattern scheme, the memory function corresponds to:

$$f(i,j) = \max\{m - i, n - j\}. \tag{7}$$

An example of the function is shown in Figure 2.b. for $m = n = 10$. Again, the coloured bands in the plot denote all $(i, j) \in [1,m]\times[1,n]$ sharing the same $f(i, j)$ value from (m,n). Moreover, the contribution of a (i, j) pair to the memory weight does not depend on the path but on the proximity to the (m,n) point. In this case, the proposed memory function makes OESM forget past events of X^m and Y^n streams in accordance with their evolution rate. The reader should note that the proposed memory function corresponds to the Chebyshev (or L_∞-norm) distance.

OESM convergence

Consider the definition of the memory function either in Equation (6) or in (7), and assume that $\exists M > 0$ so it meets $c_{i,j} \leq M$, for any (i, j) in the $[1, m] \times [1, n]$ lattice. Hence, it is easy to see that Equation (4) is upper-bounded by a geometric series

$$w_\rho(p) \leq \sum_{i=0}^{\max\{m,n\}-1} \rho^i M.$$

As it is known, when m (or/and n) tends to infinity, the geometric series converge if $|\rho| < 1$. As a consequence, the measure of similarity (Equation (5)) would be upper-bounded by

$$D_{m,n} \leq \frac{M}{1-\rho}, \text{ as long as } \rho < 1.$$

Specifically, since $c_{i,j} \leq 1$ for EDIT and EDR similarity measures, we can set $M = 1$. As a result, $D_{m,n} \leq (1-\rho)^{-1}$ in these particular cases.

3.2. Incremental computation

Following the arguments in Section 2.2, we can solve the OESM optimization problem in Equation (5) applying dynamic programming techniques. In this case, the recursive equation that provides the solution to OESM is given by

$$D_{m,n} = c_{m,n} + \min \left\{ \begin{array}{l} \rho^{f(m-1,n)} D_{m-1,n}, \\ \rho^{f(m-1,n-1)} D_{m-1,n-1}, \\ \rho^{f(m,n-1)} D_{m,n-1} \end{array} \right\}, \tag{8}$$

where $D_{0,0} = 0$ and $D_{0,j} = D_{i,0} = \infty$ for $i = 1, 2, ..., m$ and $j = 1, 2, ..., n$. As in the previous section, the computational complexity of the OESM is $O(m \times n)$, which results infeasible for two time series that grow without bounds. In order to alleviate the complexity and adapt the computation to the on-line setting, we use the following observation which is straightforwardly derived from Equation (8):

Observation 1. *When computing the ESM, we are not only calculating the final result, $D_{m,n}$, but also the intermediate $D_{i,j}$ measurements for all $i = 1, ..., m-1$ and $j = 1, ..., n-1$. In other words, the ESM between two time series X^m and Y^n, in turn, requires the computation of the ESM between all $X^i = (x_1, ..., x_i)$ and $Y^j = (y_1, ..., y_j)$ subsequences, for $i = 1, ..., m-1$ and $j = 1, ..., n-1$.*

In line with the observation above, let us define the **frontier** as

$$\mathbf{F}^{m,n} = \{D_{i,n}\}_{i=1}^{m} \cup \{D_{m,j}\}_{j=1}^{n}, \tag{9}$$

which will be used to avoid unnecessary computations when new data samples arrive.

Turning our attention to streaming scenarios, consider an evolving scenario where streaming time series evolve from X^r to X^m and from Y^s to Y^n after receiving $X^{r+1,m} = (x_{r+1}, ..., x_m)$ and $Y^{s+1,n} = (y_{s+1}, ..., y_m)$ data samples, respectively. The goal of the incremental computation is to calculate $D_{m,n}$ by demanding minimum computational resources. To this end, we assume that before receiving $X^{r+1,m}$ and $Y^{s+1,n}$, we store the frontier $\mathbf{F}^{r,s}$. By definition, $D_{m,n}$ requires the computation of the whole measure matrix, an $m \times n$ dimension matrix, whose elements correspond to $D_{i,j}$ for all $i \in \{1, ..., m\}$ and $j \in \{1, ..., n\}$. However, in this particular situation we can apply Equation (8) recursively until $\mathbf{F}^{r,s}$ is reached; rather than propagating the recurrence back to $D_{0,0} = 0$.

An example of the incremental computation is shown in Figure 1.b, where $D_{10,10}$ (red cell) is computed given $\mathbf{F}^{4,3}$ (blue cells). Note that storing the frontier, we avoid the computation of 4×3 intermediate results, specifically, those shadowed values in the figure. Likewise for the general case, where given the frontier $\mathbf{F}^{r,s}$, $r \times s$ computations are avoided. Hence, the computational complexity of the incremental computation is $O(m \cdot n - r \cdot s)$.

Next we adapt the incremental computation described above in order to use it in the batch and on-line pattern scenarios.

Batch pattern scenario

Consider the batch pattern scenario illustrated in Figure 3, where the batch pattern X^m is fixed and stored permanently in memory, whereas the streaming sequence evolves from Y^s to Y^n. Since the length of the batch pattern is fixed, the given frontier corresponds to

$$\mathbf{F}^{m,s} = \{D_{i,s}\}_{i=1}^{m}, \tag{10}$$

the blue area in Figure 3.

The OESM computation for the batch pattern scenario consists of two main steps: (1) compute and (2)store. That is, first $D_{m,n}$ is computed using Equation (8) and $F^{m,s}$. Then, $\mathbf{F}^{m,n}$ is stored – according to Expression (10) – so as to use it in future similarity measurements. The computation-storing procedure is described as a pseudo-code in Algorithm 1 and illustrated in Figure 3.

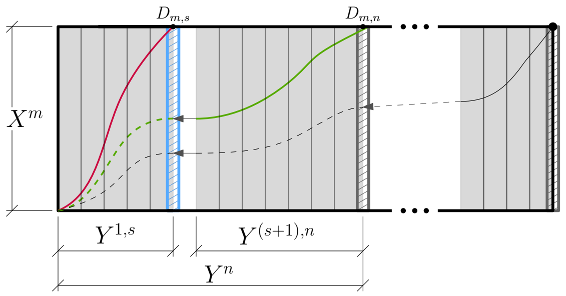


Fig. 3. OESM computation procedure for a continuous flow of query data samples in the batch pattern scenario, where red and green curves illustrate the optimal paths to $D_{m,s}$ and $D_{m,n}$ respectively, the black solid lines the memory function, and the vertical shadowed blue and grey areas the frontier.

We would like to highlight that in the batch pattern scenario the streaming time series is not required to be stored for computing the similarity measure incrementally. Consequently, the space complexity of the batch pattern scenario is constant $O(m)$, thus meeting the memory limitation requirement of the on-line learning algorithms. Regarding the running time, in this case the complexity is given by $O(m \cdot (n - s))$, where $n - s$ is the number of samples in the arriving chunk.

Algorithm 1 OESM pseudo-code for the batch pattern scenario.

```
1: Require: $\mathbf{F}^{m,s}$, $X^m$ , $\rho$.
2: for newly arriving chunk $Y^{s,n}$ do
3:     ▷ Step 1
4:     Compute $D_{m,n}$ from $\mathbf{F}^{m,s}$ using Equation (8).
5:     ▷ Step 2
6:     Store frontier $\mathbf{F}^{m,n} = \{D_{m,s+j}\}_{j=1}^{n-s}$.
7:     Redefine $s \leftarrow n$.
```

On-line pattern scenario

Consider the frontier as per Equation (9) and the recurrence in Expression (8). Since both X^m and Y^n grow in the on-line pattern scenario, they must be stored so as to compute $D_{m,n}$ incrementally. Specifically, we need past sequences, $X^{1,r}$ and $Y^{1,s}$, to compute the $D_{i,j}$ elements in the measure matrix corresponding to $(i, j) \in [1, r] \times [s + 1, n]$ and $(i, j) \in [r + 1, m] \times [1, s]$ (i.e., $D_{i,j}$ entries highlighted in bold font in Figure 7.b). Consequently, the memory required to compute OESM in the on-line pattern scenario lacks an upper bound as the time series grow.

To overcome this issue, we embrace the use of Sakoe-Chiba constraints, so that in the on-line pattern scenario the proposed OESM approach blends together the incremental computation procedure previously described in this section, and the Sakoe-Chiba strategy. Formally, let us consider the evolving scenario described in Figure 4, where l is the Sakoe-Chiba band width and the blue area corresponds to the stored frontier given by

$$\mathbf{F}^{r,s} = \{D_{i,s}\}_{i=\max\{1,r-l\}}^{r} \cup \{D_{r,j}\}_{j=\max\{1,s-l\}}^{s}. \tag{11}$$

As in the previous example, the on-line computation procedure is shown in Figure 4 and Algorithm 2, where line 4 corresponds to the computation step and lines 6 and 7 to the storage step. In this context, first $D_{m,n}$ is computed by applying the recurrence shown in Equation (8) until the frontier is reached. Due to Sakoe-Chiba band constraints, we assume $D_{i,j} = \infty$ for those $i \in \{\max\{1, r - l\}, ..., m\}$ and $j \in \{\max\{1, s - l\}, ..., n\}$ not satisfying $|i - j| \leq l$. Then, according to the definition in (11), the frontier $\mathbf{F}^{m,n}$ is stored. In addition, the last l points of each time series must also be saved in this scenario, i.e., we would need $X^{\max\{1,n-l\},m}$ and $Y^{\max\{1,m-l\},n}$ subsequences for future OESM computations.

Once the length of the time series (m and n) is greater than l, due to the imposed constraints the on-line pattern computation procedure leads to a limited space complexity of $O(l)$. Likewise, the constraints reduce the running time from quadratic to linear with chunk size, $O(l \cdot \max\{m - r, n - s\})$. According to Figure 4, the time series must grow similarly in order for OESM to be a consistent computation procedure[1]. If that does not hold, we should take the rate in which the time series grows for proper adaptation of the memory function and the Sakoe-Chiba constraints. Hence, in the on-line pattern scenario, we will assume that

[1]When one time series grows significantly faster than the other, eventually $|m - n| > l$. In this case, we store points of X^m and Y^n that might never be used in the computation of the similarity measure.

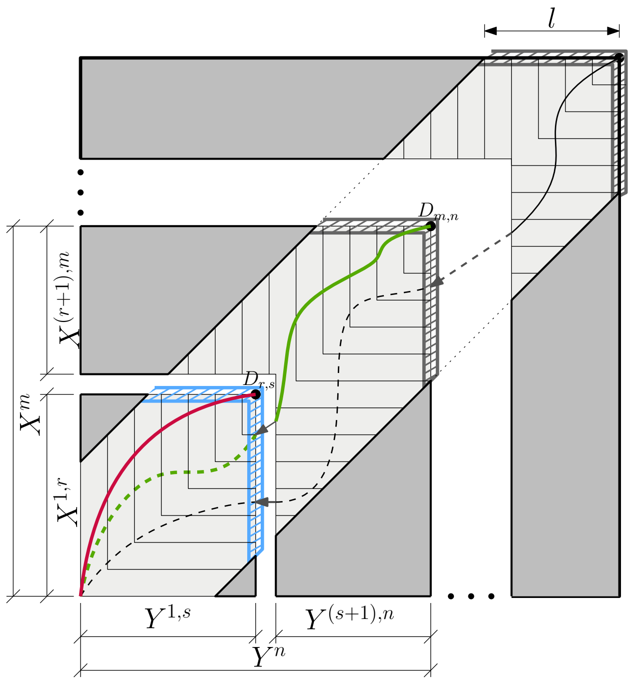


Fig. 4. In the on-line pattern scenario, OESM computation procedure for a continuous flow of on-line pattern and query data samples. The red and green curves illustrate the optimal paths to $D_{m,s}$ and $D_{m,n}$ respectively. According to Equation (6), black solid lines show equidistant (i, j) points. The shadowed blue and grey areas correspond to the frontier.

time series evolve so the arriving chunks satisfy $|m - n| \leq l$.

Algorithm 2 OESM pseudo-code for the batch pattern scenario.

```
1: Require: $\mathbf{F}^{m,s}$, $X^m$ , $\rho$.
2: for newly arriving chunk $Y^{s,n}$ do
3:     ▷ Step 1
4:     Compute $D_{m,n}$ from $\mathbf{F}^{m,s}$ using Equation (8).
5:     ▷ Step 2
6:     Store frontier $\mathbf{F}^{m,n} = \{D_{m,s+j}\}_{j=1}^{n-s}$.
7:     Redefine $s \leftarrow n$.
```

4. Experimental Setup

In this section we describe the experiments used to analyze, in terms of efficiency and the forgetting mechanism, the proposed OESM in both the batch and on-line pattern scenarios.

4.1. Efficiency

In order to shed light on the efficiency of OESM, we compare its computation time in the batch and on-line pattern scenarios with the computation time required by the conventional ESM with the Sakoe-Chiba constraints. In this context, two sets of experiments are proposed. Considering a constant size of arriving chunks, the first set of experiments aims at analyzing the efficiency over time, i.e., we compare the running time as the length of the time series grows. In the second set of experiments we consider arriving chunks of increasing size and the objective is to analyze the running time required to compute the similarity assuming that the initial length of the time series is kept fixed.

To carry these experiments out, we compute the measure of similarity between two randomly generated streaming time series drawn from a uniform distribution, $U(0,1)$.

- **Experiment 1.** Let $b = 1$ be the length of the arriving chunks. Either in the on-line pattern scheme (OESM) or for the conventional ESM study, the streaming time series evolve from an initial size of 3 points to a final length of 70 points. In both cases the width of Sakoe-Chiba band has been set at 35. Regarding the batch pattern scheme (OESM), the length of the streaming time series ranges from 3 to 70, while the length of the batch time series is fixed at 35.

- **Experiment 2.** In this second experiment we analyze the running time of the OESMs as a function of the length of the arriving chunks. For both the on-line or off-line setting, we compute the similarity measure for length $b \in [1, 50]$ assuming that the length of the stored time series is $m = n = 35$. As in the previous experiment, the on-line pattern OESM and the conventional ESM are run under Sakoe-Chiba constraints, where the band length has been set at $l = 35$.

As we have mentioned at the end of Section 2, the difference among ESM variants resides in the definition of the cost function. For this set of experiments, we assume that the time needed to compute the cost functions listed in Table 1 is similar. As a consequence, there is no need for replicating the experiments with all EMS variants. For these experiments, the considered cost function corresponds to the ED, i.e., the results we show further correspond to those issued from the conventional DTW and its on-line adaptation.

4.2. Forgetting mechanism

In this section we present the experimentation used to test the behavior of the proposed forgetting mechanism when dealing with non-stationary streaming time series. The conducted experimentation aims to evaluate whether it allows OESMs to neglect the past and accommodate to incoming stationary intervals. We have decided to evaluate the forgetting mechanism by means of a supervised classification problem, which allows to quantify the behavior objectively, in terms of the accuracy. For this purpose we use a 1-NN classifier, one of the simplest and most widely utilized distance-based approaches.

The experiments are carried out with specially designed non-stationary time series streams. As we subsequently explain in more detail, these streams are created by concatenating sequences contained in the Time Series Classification Archive (Chen et al., 2015) databases.

4.2.1. Generation of non-stationary streaming time series

The generation process uses time series contained in the UCR archive as a substrate from which to compose the streaming time series for our experimental benchmark. Precisely, this repository collects real and synthetic datasets, each consisting of multiple time series that have – from a set of possible labels – a class label associated with each of them. In this sense, let $\mathbf{U} = \{(U_k, c_k)\}_{k=1}^{K}$

denote a UCR dataset, where $c_k \in \{1, ..., C\}$ represents the label of the $U_k = (u_1^k, ..., u_v^k)$ time series, C is the total number of class labels, and K is the total number of time series in the dataset. Drawing time series randomly from $\mathbf{U}$ and concatenating them, we construct a non-stationary time series stream following a predefined duration of stationary intervals (the assigned label keeps constant) which, in turn, establishes the position where label transitions occur. The steps involved in the creation of non-stationary streaming time series can be summarized as follows:

1. First, P time series sharing the same class label are selected uniformly at random from the selected dataset. The selected sequences are z-normalized[2] and concatenated in order to model a stationary interval. Parameter P denotes the periodicity of the stationary interval, i.e., the number of time series of a given class concatenated before a label change is held.

2. Stationary intervals are assembled together (one after another) in such a way that two consecutive stationary intervals do not share the same class label. That is, stationary intervals are concatenated forcing a transition in each union. We refer as T to the total number of transitions in a streaming time series.

An example showing the generation process of a streaming time series from a generic binary-class dataset $\mathbf{U}$ is depicted in Figure 5. In this example, the streaming time series parameters are $T = 4$ transitions and $P = 2$ time series per stationary interval (periodicity).

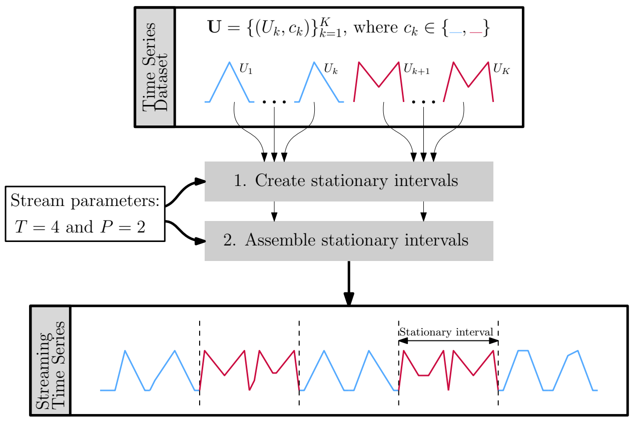

Time Series Dataset
$\mathbf{U} = \{(U_k, c_k)\}_{k=1}^{K}$, where $c_k \in \{\_, \_\}$
U_1
U_k
U_{k+1}
U_K
Stream parameters: $T = 4$ and $P = 2$
1. Create stationary intervals
2. Assemble stationary intervals
Streaming Time Series
Stationary interval

Fig. 5. Stream time series generation process. The upper box shows a binary-class dataset U, where c_k (red or blue) denotes the ground truth label of U_k time series. Given the number of transitions $T = 4$ and the periodicity $P = 2$, the box at the bottom illustrates the resulting time series stream, where the vertical dashed lines indicates the transition point.

For each scenario (batch pattern and on-line pattern) and UCR dataset in Table 2, $N_{\text{CP}} = 50$ different on-line classification problems have been created. For each dataset $\mathbf{U}$, these classification problems consist of a query stream and a set of reference (streaming) time series that are defined as follows:

[2]Data normalization has been widely acknowledged as a major open issue in stream data mining by the related literature. This work assumes that non-stationary streaming time series are composed by normalized sequences so that the performance of the proposed adaptation becomes unbiased with respect to this aspect and focuses strictly on the OESM adaptability.

Table 2. Main characteristics of the utilized UCR datasets.

Dataset (**U**)	Time series length (v)	Number of classes (C)	Sakoe-Chiba band (l)
`Gun Point`	150	2	0
`Two Lead ECG`	82	2	4
`Wafer`	152	2	4
`Synthetic Control`	60	6	4
`Two Patterns`	128	4	5
`Plane`	144	7	9
`CBF`	128	3	14
`Faces UCR`	131	14	16
`Symbols`	398	6	32

- **Batch pattern scenario.** The query stream of the batch pattern scenario is a time series with $T = 14$ transitions and a periodicity of $P = 1$. The batch pattern set (i.e., the reference time series set) is just a subset of **U**, where all class labels are equally represented each with 10 randomly selected time series.

- **On-line pattern scenario.** The query stream of the on-line pattern scenario consists of $T = 9$ transitions with a periodicity of $P = 3$. In this case, the on-line pattern set (i.e., the reference stream set) consists of $10 \cdot C$ stationary streams (10 per class label), each created by concatenating 30 time series with equal labels (that is, with $T = 0$ and $P = 30$).

For the sake of fairness, the reference and query streams do not share any original time series, i.e., in the classification problem design, the random sampling is made without replacement.

4.2.2. Gold-standard model

In order to compare our approach, we consider as gold-standard a classifier based on the ESM that, in addition, uses the query stream information (total number of transitions T and the periodicity P) to reset the measure of similarity in order to adapt to the incoming stationary interval. By doing so, note that the gold-standard classifier can be considered the perfect model as i) it forgets old stationary intervals immediately, and ii) it adapts to new intervals by taking all query stream points information into account.

4.2.3. Memory parameter

The selection of an appropriate value for the memory parameter is not a trivial task, as it roughly depends on the problem at hand. In our case we will set the memory parameter taking into account the length v of the time series within every UCR dataset **U** (second column in Table 2). The considered memory parameters are given by

$$\rho \in \{0.0001^{1/v}, 0.1^{1/v}, 0.5^{1/v}, 1^{1/v}\}. \tag{12}$$

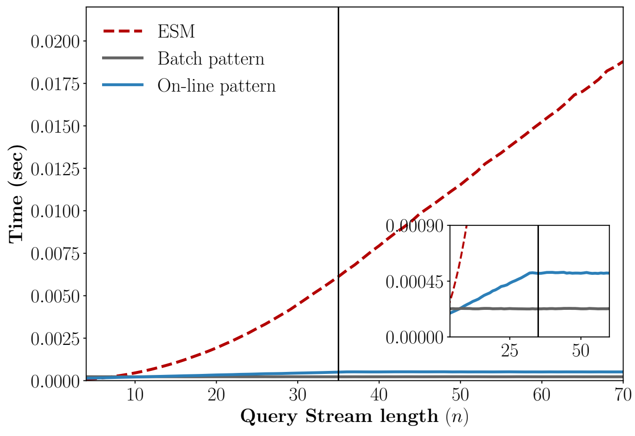
In words, by using these values the contribution of the time series point v steps prior to the sample at hand is weighted by $\{0.0001, 0.1, 0.5, 1\}$. We will hereafter refer to the selected values as short ($\rho = 0.0001^{1/v}$), middle ($\rho = 0.1^{1/v}$), large ($\rho = 0.5^{1/v}$) and full ($\rho = 1$) range memory. It is important to recall that the OESM is completely equivalent to the ESM when $\rho = 1$, hence full range memory OESM and ESM will be used indistinctly.

5. Results and Discussion

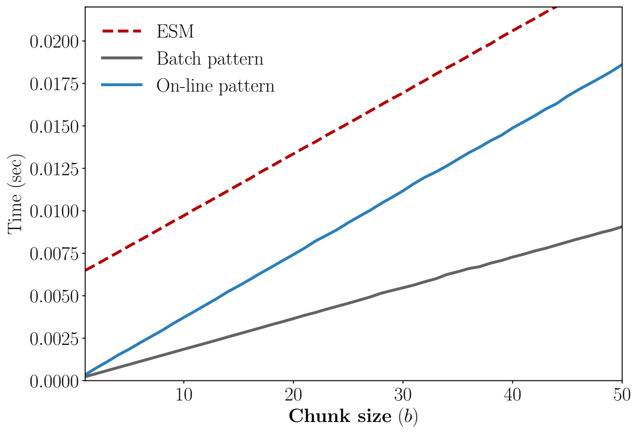
In this section we present and comment on the results of the conducted experiments.

5.1. Efficiency

Results for the running time are shown in Figure 6. The plot on the left illustrates the results for Experiment 1, whereas the plot on the right shows the results for Experiment 2 (see Section 4.1). After 100 independent runs of each experiment, the plots show the average computational time (in seconds) required by each scenario[3]: the conventional ESM with Sakoe-Chiba band (red dashed), the on-line pattern (blue) and the batch pattern (grey). In Figure 6.a, the vertical solid line indicates the point where the Sakoe-Chiba band is exceeded.



(a) **Experiment 1.**



(b) **Experiment 2.**

Fig. 6. Average running times of conventional ESM and proposed OESM under different parameters. Number of runs 100**.**

- **Experiment 1:** As can be seen in the plot, the running time required by the conventional ESM (i.e., the ESM computed as per Equation (3)) grows in a quadratic manner when the query stream length n is lower than or equal to the Sakoe-Chiba band width l, and linearly once $n > l$ (see red dashed curve). Regarding the running time of the on-line pattern scenario, note that it grows linearly when $n \leq l$. However, once $n > l$ the complexity remains constant (blue solid line). For the batch pattern scenario, the computational time is constant for any length of the time series (gray solid line). Observe that once n exceeds the Sakoe-Chiba band, the running time required by the on-line pattern is twice the running time for the batch pattern scenario (gray versus blue solid lines). This is just the consequence of imposing $l = m = 35$. When a new sample arrives, the number of elements to be computed in the distance matrix is equal to the number of elements in the frontier. In this case, 35 in the batch pattern scheme and 69 in the on-line pattern.

[3]The results are obtained using a Python 2.7 naive code run on a single i7 core at 3.10 GHz.

- **Experiment 2:** Due to the Sakoe-Chiba band constraints, the running time of both the conventional ESM and the on-line pattern grow linearly. Moreover, note that the curves are nearly parallel (compare red dashed and blue solid lines). The required time also increases linearly for pattern scenario (see gray solid line). The chosen time series length and the Sakoe-Chiba band width make the computational complexity $O(b)$ for on-line and batch pattern scenarios. However, note that the slopes are quite different (compare blue and gray solid lines).

5.2. Reaction capacity and predictive performance

Given the $N_{\text{CP}} = 50$ classification problems built from each UCR dataset in Table 2, and the values for the memory parameter ρ in Equation (12); in this section we discuss, for both the batch and on-line pattern scenarios, the results issued from the OESM-based and gold-standard classifiers. To this end, we use the average accuracy and the average ranking defined in the following section.

5.2.1. Evaluation method

In order to numerically quantify the accuracy of each classifier – either in the batch or in the on-line pattern scenario–, we have considered that a classifier correctly predicts the class label of an arriving query point if the predicted label coincides with the label of the stationary interval it belongs to. Consider the memory parameter ρ, and let $\mathbf{U}_\kappa$ be the κ-th generated classification problem built from the UCR dataset $\mathbf{U}$. For the i-th arriving query stream point (recall that the size of the arriving chunk is $b = 1$), the predictive performance (or accuracy) is computed as follows:

$$\text{ACC}_{\mathbf{U}_\kappa}(i;\rho) = \begin{cases} 1 & \text{if} \quad \text{PL}(i) = \text{TL}(i), \\ 0 & \quad\text{otherwise,} \end{cases} \tag{13}$$

where PL(i) and TL(i) are, for the arriving point i, the predicted label and true label, respectively. Note that, TL(i) is the same for all i belonging to the same stationary interval.

As already known, the accuracy of the 1-NN classifier strongly depends on the selected reference and query streams. To account for this statistical variability, the average accuracy, $\overline{\text{ACC}}_{\mathbf{U}}(i;\rho)$, is computed over the N_{CP} classification problems built on $\mathbf{U}$.

Given $\mathbf{U}$ and a set of memory parameters, suppose we compute the average accuracy for each ρ in the set. In order to extract an overall view of the influence of the memory parameter ρ across all the classification problems, we have analyzed the experimental results right after a label transition (i.e., at the *beginning* of a stationary interval), in the *middle* or at the *end* of the stationary interval. To do so, we apply the following process to each part (*beginning*, *middle*, *end*):

1. For each i in the corresponding part and memory parameter ρ, we compute the the ranking produced by the average accuracy.

2. We compute the average ranking for the average accuracies obtained in the previous step. In this context, we will use $\bar{\text{r}}_1(\mathbf{U},\rho)$, $\bar{\text{r}}_2(\mathbf{U},\rho)$ and $\bar{\text{r}}_3(\mathbf{U},\rho)$ to denote the average ranking in the *beginning*, *middle* and *end* parts respectively.

5.2.2. Results

For the sake of brevity, we only show a representative sample of the experiment results in this section, specifically those corresponding to the DTW cost function. The results issued from the remaining cost functions – EDIT, EDR and ERP – have been made available in `http://bitbucket.org/izaskun_oregui/ESM`.

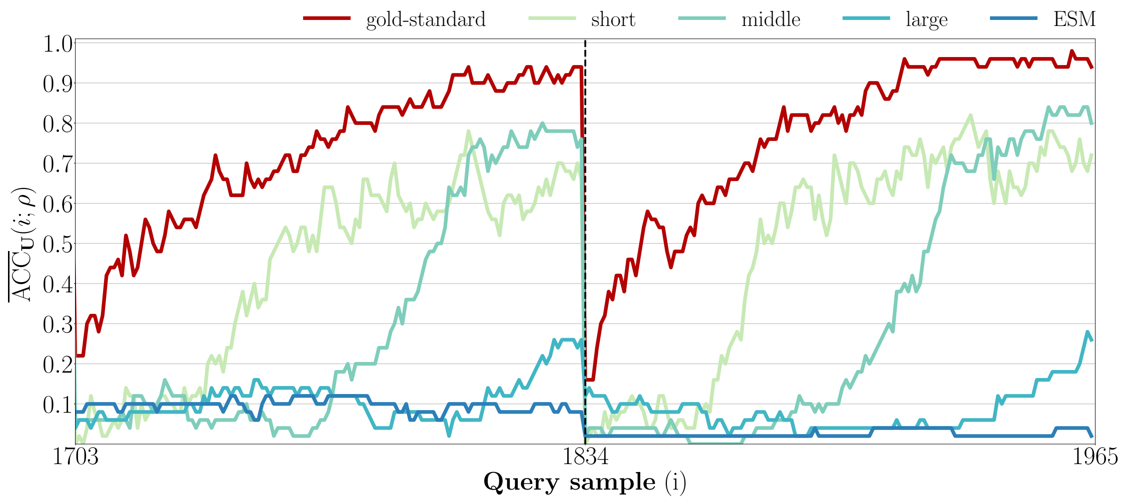
Batch pattern scenario

The outcomes for the batch pattern scenario are shown in Figure 7.

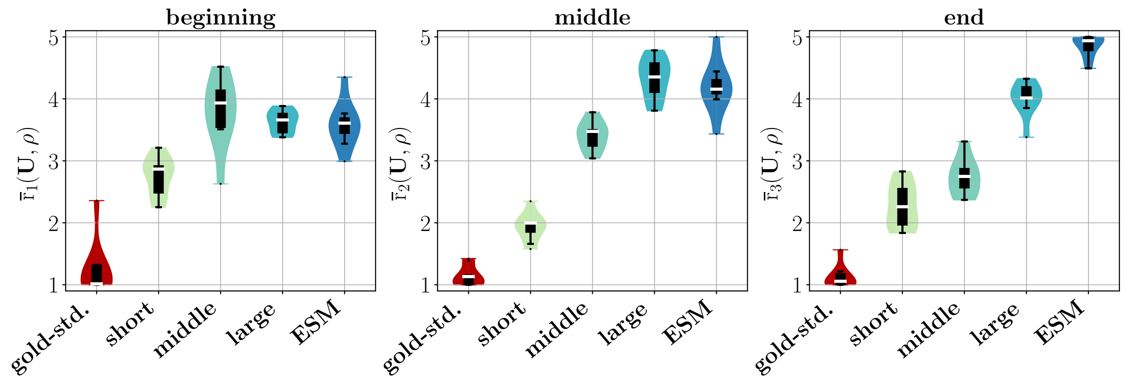
Considering all values of the memory parameter ρ and the classification problems built on `Faces UCR` dataset, Figure 7.a illustrates the average accuracy, $\overline{\mathrm{ACC}_{\mathbf{U}}}(i;\rho)$, over the last 2 simulated stationary intervals. As can be observed, the results drawn from the gold-standard classifier are the best throughout the intervals (red curve). Moreover, note that the scores issued from this model show a continuous progress that starts from an initial value of 0.2 to 0.95, in both intervals. On the contrary, the results issued from the ESM classifier do not present any progress. Precisely, the obtained scores are those of random guessing. Likewise, the results from the large-range memory OESM-based classifier are those of random guessing (expect at the end of the stationary intervals where results slightly improve). In contrast to the gold-standard model, where the measure of similarity is reset immediately after streams transitions, large-range and ESM models are highly influenced by the past alignments, and as a result these classifiers are unable to adapt to streams transitions. A balance – between the steady progress of the gold-standard model and the impasse of the large-range and ESM models – is observed for short- and middle-range memory models. In these cases, the smaller the memory parameter value is, the sooner the accuracy increases due to its capacity to forget. However, in certain cases this involves a penalty in score once we go ahead on the stationary interval. Indeed, this noted behavior is consistent with the stability-plasticity dilemma formulated in the context of incremental learning over non-stationary streams, which states that there is a trade-off between the learning capability of a predictive model and its flexibility and speed to capture evolving concepts along the stream (Ditzler et al., 2015). Finally, note that the curves seems to be noisier as ρ decreases. This is in part due to the influence of the high difference between past and present alignments in the OESM.

In order to see whether the observed evolution of $\overline{\mathrm{ACC}_{\mathbf{U}}}(i;\rho)$ can be extrapolated to all $\mathbf{U}$ in the benchmark, Figure 7.b shows, the empirical average ranking distribution across all $\mathbf{U}$ in Table 2. Three plots are included to focus on the (*beginning*,*middle*,*end*) parts of the stationary intervals. According to the aforementioned in Section 4.2.2, the gold-standard is the best positioned model in all partitions (red violin plot). For the OESM-based models, the ranking varies as we progress on the stationary interval as follows:

- ***Beginning*:** Intermediately after every stream transition, scores from the OESM-based classifiers are those from random guessing. In this case the short-range memory model is slightly better than the rest of the OESM-based classifiers (middle-range, large-range and ESM). This is directly related to the fact that short-range memory model reacts sooner than the rest.



(a)



(b)

Fig. 7. Performance of the 1-NN classifier for the batch pattern scenario and different values of ρ: (a) Accuracy score over the last 2 simulated stationary intervals of the non-stationary classification problems built on `Faces UCR` time series; (b) Average raking distribution from the last 5 simulated stationary intervals considering all datasets in Table 2. The depicted violin plots represent for each partition (*beginning*,*middle*,*end*), the average ranking distribution. The horizontal white line denotes the median from all considered datasets.

- ***Middle***: A considerable improvement is observed for the short-range memory model and, to a lesser extend, for the middle-range memory classifier. Again, this behaviour is in line with the aforementioned idea: the smaller ρ is, the sooner the reaction of the accuracy score.

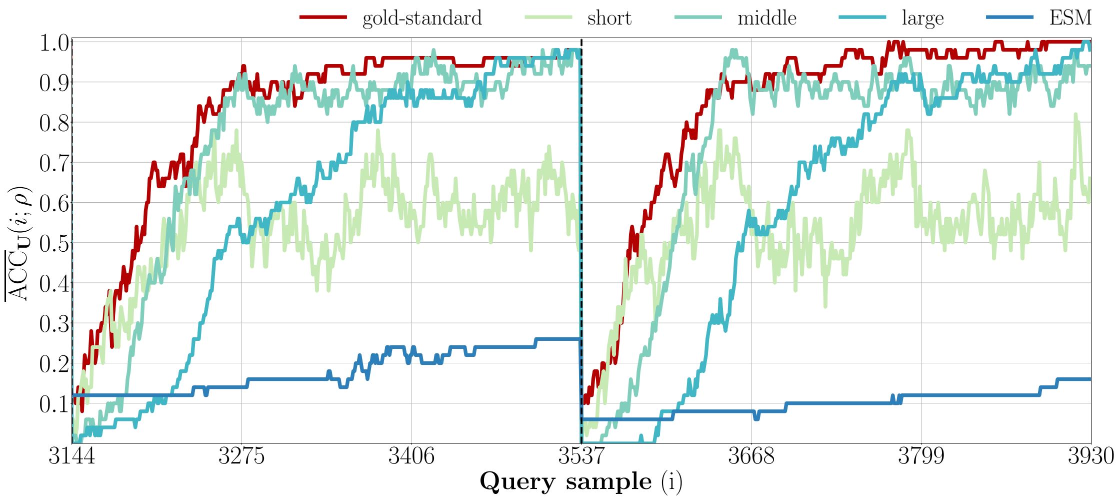
- ***End***: Note from Figure 7.a that while the average accuracy issued from the short-range memory model is stabilized, the middle-range memory model accuracy continues to improve. Consequently, the middle-range memory model is, in terms of average ranking, the model which improves the most in this partition.

Regarding the average ranking distribution for the large-range and ESM classifiers, note that they get worse. This is an expected result since the scores drawn from these models do not progress throughout the interval.

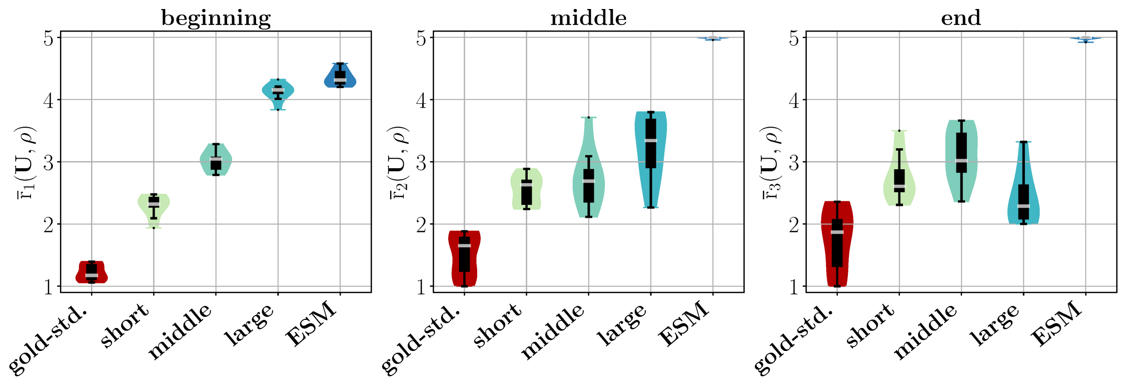
On-line pattern scenario

The results of the on-line pattern scenario are shown in Figure 8.

Given the streaming time series built on `Faces UCR` dataset, Figure 8.a illustrates the average accuracy issued from the gold-standard and OESM-based classifiers. As can be observed, the accuracy score is very similar to the behaviour described for the batch pattern scenario, where models with smaller values of the memory parameter ρ react sooner to transitions than those with larger values. Moreover, the penalty in the scores of most forgetful models is also more evident than in the previous scenario (observe results issued from the short-range memory model). Again, since ESM model is fully influenced by the past events, the results drawn from this model are still the same as those from random guessing. However, the large-range memory model is able to accommodate to streams transitions. This is due to the periodicity increase i.e., in contrast to the batch pattern scenario where $P = 1$, in the on-line pattern scenario $P = 3$, hence, the large-range memory model has more *time* to forget past events and to accommodate to contemporaneous stationary intervals.



(a)



(b)

Fig. 8. Predictive performance of the 1-NN classifier for proposed ρ values in the on-line pattern scenario: (a) Given the classification problem built `Faces UCR` dataset, average accuracy over the last 2 simulated stationary intervals; (b) Considering all datasets in Table 2 and the last 5 simulated stationary intervals, the violinplots in the figure represent for each part (*beginning*, *middle*, *end*) the average ranking distribution, where the horizontal white line denotes the median.

Similarly to the discussion on the previous scenario, the results considering all the datasets in Table 2 are summarized in Figure 8.b. In this case, the analysis of each partition is outlined below:

- ***Beginning*****:** As in the batch pattern scenario, the gold-standard model is the best at the beginning of the stationary interval. In contrast, the OESM models (short-range, middle-range, large-range and ESM models) are no longer similar. In this case, the average rank worsens as the memory parameter value increases. As the length of the partition increases, OESM models have more time to react and consequently to accommodate to streams transitions being those models with small ρ values the first to react.

- ***Middle*****:** Again, gold-standard is the best positioned model. Regarding the OESM-based classifiers, middle- and large-range models are those improving the most. This is true because their accommodation is slower and also because the score penalty is lower for those models with higher ρ values. Finally, note that the ESM is the worst model for all the considered dataset.

- ***End*****:** In this case, the average ranking median is 2 for the conventional gold-standard model, meaning that, in some dataset, results drawn from the OESM can be as accurate as the conventional model. As aforementioned, the large-range memory model is able to accommodate to stationary interval with lower penalty (in the score) than models with smaller ρ values. As a result, the improvement of the large-range memory model is noteworthy. The ESM is once again the worst model.

6. Conclusions

In this work, we have proposed the On-line Elastic Similarity Measure (OESM), namely, an adaptation of conventional ESMs suited to measure the similarity between time series in on-line settings. Our proposed OESM is able to compute measures of similarity incrementally, and incorporates a novel forgetting mechanism to adapt to changes in the streaming time series. We have shown that the proposed adaptation is conceptually simple, that it can be computed using limited computational resources, and that it is able to neglect the past so as to focus on newly arriving points. Regarding this last issue, the results drawn from 1-NN classifiers have shown that the behavior of the proposed forgetting mechanisms is in line with the stability-plasticity dilemma identified in incremental learning models over non-stationary data streams (Ditzler et al., 2015).

For these reasons, the proposed OESM opens many research lines in the pattern recognition field, in particular, it accommodates a direct adaptation of distance-based methodologies to the on-line setting. In other words, OESM can be used to accommodate any learning algorithm hinging on distances of similarity between time series, such as nearest-neighbor classifiers, kernel-based classifiers, k-medoids or hierarchical clustering, to evolving scenarios. Moreover, the proposed formulation can accommodate other possible inner cost functions $c(x_i, y_j)$ apart from those in DTW, EDR, EDIT and others alike.

With the aim of giving insights into the influence of the memory parameter ρ, the parameter value setting has been made based on the UCR time series length. As previously mentioned, different ρ values yield different behaviors in the prediction performance. As future work, we propose to develop a strategy to select the best value of ρ given the characteristics of a given problem. We are currently developing a methodology for stream transition detection based on the proposed OESM. We also plan to apply the proposed OESM to other supervised and unsupervised classification problems – such as early classification or clustering – where streaming time series are involved. Finally, we will elaborate different normalization techniques suited for on-line environments, so that the proposed OESM can be applied to streaming contexts without any prior normalization assumption.

Acknowledgments

This work has been supported by the Basque Government through the EMAITEK program as well as through the ELKARTEK program (TEKINTZE project, ref. KK-2018/00104). Aritz Pérez and José A. Lozano are both supported by the Basque Government through the BERC 2018-2021 program and by the Spanish Ministry of Economy and Competitiveness MINECO through BCAM Severo Ochoa excellence accreditation SEV-2017-0718. José A. Lozano is also supported by Spanish Ministry of Economy and Competitiveness MINECO through TIN2016-78365-R, and Aritz Pérez through TIN2017-82626-R funded by (AEI/FEDER, UE).

References

Aggarwal, C.C., Reddy, C.K., 2016. Data clustering: algorithms and applications. CRC Press.
Aghabozorgi, S., Shirkhorshidi, A.S., Wah, T.Y., 2015. Time-series clustering–a decade review. Information Systems 53, 16–38.
Bellman, R.E., Dreyfus, S.E., 2015. Applied dynamic programming. Princeton university press.
Beringer, J., Hüllermeier, E., 2006. Online clustering of parallel data streams. Data & Knowledge Engineering 58, 180–204.
Berndt, D.J., Clifford, J., 1994. Using dynamic time warping to find patterns in time series, in: Workshop on Knowledge Discovery in Databases, Seattle, WA. pp. 359–370.
Chen, L., Ng, R., 2004. On the marriage of lp-norms and edit distance, in: Proceedings of the 30th International Conference on Very Large Data Bases, VLDB Endowment. pp. 792–803.
Chen, L., Özsu, M.T., Oria, V., 2005. Robust and fast similarity search for moving object trajectories, in: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, ACM. pp. 491–502.
Chen, Y., Keogh, E., Hu, B., Begum, N., Bagnall, A., Mueen, A., Batista, G., 2015. The ucr time series classification archive. `www.cs.ucr.edu/~eamonn/time_series_data/`.
Cinti, A., Bianchi, F.M., Rizzi, A., 2017. A novel algorithm for online inexact string matching and its fpga implementation. arXiv preprint arXiv:1712.03560 .
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. Proceedings of the Very Large Data Bases Endowment 1, 1542–1552.
Ditzler, G., Roveri, M., Alippi, C., Polikar, R., 2015. Learning in nonstationary environments: A survey. IEEE Computational Intelligence Magazine 10, 12–25.
Esling, P., Agon, C., 2012. Time-series data mining. ACM Computing Surveys 45, 12.
Fard, M.J., Pandya, A.K., Chinnam, R.B., Klein, M.D., Ellis, R.D., 2017. Distance-based time series classification approach for task recognition with application in surgical robot autonomy. The International Journal of Medical Robotics and Computer Assisted Surgery 13.
Gama, J., 2010. Knowledge discovery from data streams. CRC Press.
Itakura, F., 1975. Minimum prediction residual principle applied to speech recognition. IEEE Transactions on Acoustics, Speech, and Signal Processing 23, 67–72.
Jeong, Y.S., Jeong, M.K., Omitaomu, O.A., 2011. Weighted dynamic time warping for time series classification. Pattern Recognition 44, 2231–2240.
Keogh, E., 2002. Exact indexing of dynamic time warping, in: Proceedings of the 28th International Conference on Very Large Data Bases, VLDB Endowment. pp. 406–417.
Keogh, E.J., Pazzani, M.J., 2000. Scaling up dynamic time warping for datamining applications, in: Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM. pp. 285–289.
Krempl, G., Žliobaite, I., Brzeziński, D., Hüllermeier, E., Last, M., Lemaire, V., Noack, T., Shaker, A., Sievi, S., Spiliopoulou, M., et al., 2014. Open challenges for data stream mining research. ACM SIGKDD Explorations Newsletter 16, 1–10.
Liao, T.W., 2005. Clustering of time series data – a survey. Pattern Recognition 38, 1857–1874.
Lin, J., Keogh, E., Lonardi, S., Chiu, B., 2003. A symbolic representation of time series, with implications for streaming algorithms, in: Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, ACM. pp. 2–11.
Lines, J., Bagnall, A., 2015. Time series classification with ensembles of elastic distance measures. Data Mining and Knowledge Discovery 29, 565–592.
Lotte, F., Congedo, M., Lécuyer, A., Lamarche, F., Arnaldi, B., 2007. A review of classification algorithms for eeg-based brain–computer interfaces. Journal of Neural Engineering 4, R1.
Marzal, A., Vidal, E., 1993. Computation of normalized edit distance and applications. IEEE Transactions on Pattern Analysis and Machine Intelligence 15, 926–932.
Mondal, T., Ragot, N., Ramel, J.Y., Pal, U., 2018. Comparative study of conventional time series matching techniques for word spotting. Pattern Recognition 73, 47–64.
Mori, U., Mendiburu, A., Dasgupta, S., Lozano, J.A., 2017. Early classification of time series by simultaneously optimizing the accuracy and earliness. IEEE Transactions on Neural Networks and Learning Systems .
Mueen, A., Chavoshi, N., 2015. Enumeration of time series motifs of all lengths. Knowledge and Information Systems 45, 105–132.
Oregi, I., Del Ser, J., Pérez, A., Lozano, J.A., 2017a. Nature-inspired approaches for distance metric learning in multivariate time series classification, in: IEEE Congress on Evolutionary Computation, IEEE. pp. 1992–1998.

Oregi, I., Pérez, A., Del Ser, J., Lozano, J.A., 2017b. On-line dynamic time warping for streaming time series, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer. pp. 591–605.

Panagiotakis, C., Papoutsakis, K., Argyros, A., 2018. A graph-based approach for detecting common actions in motion capture data and videos. Pattern Recognition 79, 1–11.

Parziale, A., Diaz, M., Ferrer, M.A., Marcelli, A., 2018. Sm-dtw: Stability modulated dynamic time warping for signature verification. Pattern Recognition Letters .

Petitjean, F., Ketterlin, A., Gançarski, P., 2011. A global averaging method for dynamic time warping, with applications to clustering. Pattern Recognition 44, 678–693.

Rakthanmanon, T., Campana, B., Mueen, A., Batista, G., Westover, B., Zhu, Q., Zakaria, J., Keogh, E., 2012. Searching and mining trillions of time series subsequences under dynamic time warping, in: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM. pp. 262–270.

Ratanamahatana, C.A., Keogh, E., 2004. Everything you know about dynamic time warping is wrong, in: 3rd International Workshop on Mining Temporal and Sequential Data, Seattle, WA. pp. 50–60.

Rodrigues, P.P., Gama, J., Pedroso, J., 2008. Hierarchical clustering of time-series data streams. IEEE Transactions on Knowledge and Data Engineering 20, 615–627.

Sakoe, H., Chiba, S., 1978. Dynamic programming algorithm optimization for spoken word recognition. IEEE Transactions on Acoustics, Speech, and Signal Processing 26, 43–49.

Sakurai, Y., Yoshikawa, M., Faloutsos, C., 2005. Ftw: fast similarity search under the time warping distance, in: Proceedings of the 24th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, ACM. pp. 326–337.

Salvador, S., Chan, P., 2007. Toward accurate dynamic time warping in linear time and space. Intelligent Data Analysis 11, 561–580.

Villar-Rodriguez, E., Del Ser, J., Oregi, I., Bilbao, M.N., Gil-Lopez, S., 2017. Detection of non-technical losses in smart meter data based on load curve profiling and time series analysis. Energy 137, 118–128.

Yeh, M.Y., Dai, B.R., Chen, M.S., 2007. Clustering over multiple evolving streams by events and correlations. IEEE Transactions on Knowledge and Data Engineering 19.

Yu, D., Yu, X., Hu, Q., Liu, J., Wu, A., 2011. Dynamic time warping constraint learning for large margin nearest neighbor classification. Information Sciences 181, 2787–2796.

Yu, S.N., Chen, Y.H., 2007. Electrocardiogram beat classification based on wavelet transformation and probabilistic neural network. Pattern Recognition Letters 28, 1142–1150.

Zhao, J., Itti, L., 2018. shapedtw: Shape dynamic time warping. Pattern Recognition 74, 171–184.

Zhou, F., De la Torre, F., 2016. Generalized canonical time warping. IEEE Transactions on Pattern Analysis and Machine Intelligence 38, 279–294.

Zhou, F., De la Torre, F., Hodgins, J.K., 2013. Hierarchical aligned cluster analysis for temporal clustering of human motion. IEEE Transactions on Pattern Analysis and Machine Intelligence 35, 582–596.