Soft-DTW: a Differentiable Loss Function for Time-Series

Marco Cuturi [1] **Mathieu Blondel** [2]

Abstract

We propose in this paper a differentiable learning loss between time series, building upon the celebrated dynamic time warping (DTW) discrepancy. Unlike the Euclidean distance, DTW can compare time series of variable size and is robust to shifts or dilatations across the time dimension. To compute DTW, one typically solves a minimal-cost alignment problem between two time series using dynamic programming. Our work takes advantage of a smoothed formulation of DTW, called soft-DTW, that computes the soft-minimum of all alignment costs. We show in this paper that soft-DTW is a *differentiable* loss function, and that both its value and gradient can be computed with quadratic time/space complexity (DTW has quadratic time but linear space complexity). We show that this regularization is particularly well suited to average and cluster time series under the DTW geometry, a task for which our proposal significantly outperforms existing baselines (Petitjean et al., 2011). Next, we propose to tune the parameters of a machine that outputs time series by minimizing its fit with ground-truth labels in a soft-DTW sense.

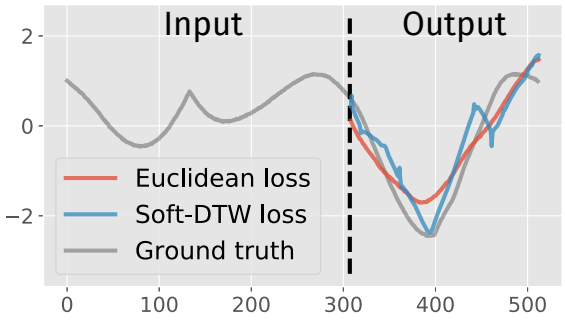


Figure 1. Given the first part of a time series, we trained two multi-layer perceptron (MLP) to predict the entire second part. Using the ShapesAll dataset, we used a Euclidean loss for the first MLP and the soft-DTW loss proposed in this paper for the second one. We display above the prediction obtained for a given test instance with either of these two MLPs in addition to the ground truth. Oftentimes, we observe that the soft-DTW loss enables us to better predict sharp changes. More time series predictions are given in Appendix F.

1. Introduction

The goal of supervised learning is to learn a mapping that links an input to an output objects, using examples of such pairs. This task is noticeably more difficult when the output objects have a structure, *i.e.* when they are not vectors (Bakir et al., 2007). We study here the case where each output object is a *time series*, namely a family of observations indexed by time. While it is tempting to treat time as yet another feature, and handle time series of vectors as the concatenation of all these vectors, several practical issues arise when taking this simplistic approach: Time-indexed phenomena can often be stretched in some areas along the time axis (a word uttered in a slightly slower pace than usual) with no impact on their characteristics; varying sampling conditions may mean they have different lengths; time series may not synchronized.

The DTW paradigm. Generative models for time series are usually built having the invariances above in mind: Such properties are typically handled through latent variables and/or Markovian assumptions (Lütkepohl, 2005, Part I,§18). A simpler approach, motivated by geometry, lies in the direct definition of a discrepancy between time series that encodes these invariances, such as the Dynamic Time Warping (DTW) score (Sakoe & Chiba, 1971; 1978). DTW computes the best possible alignment between two time series (the optimal alignment itself can also be of interest, see e.g. Garreau et al. 2014) of respective length n and m by computing first the $n \times m$ pairwise distance matrix between these points to solve then a dynamic program (DP) using Bellman's recursion with a quadratic (nm) cost.

The DTW geometry. Because it encodes efficiently a useful class of invariances, DTW has often been used in a discriminative framework (with a k-NN or SVM classifier) to predict a real or a class label output, and engineered to run

[1]CREST, ENSAE, Université Paris-Saclay, France [2]NTT Communication Science Laboratories, Seika-cho, Kyoto, Japan. Correspondence to: Marco Cuturi <marco.cuturi@ensae.fr>, Mathieu Blondel <mathieu@mblondel.org>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

faster in that context (Yi et al., 1998). Recent works by Petitjean et al. (2011); Petitjean & Gançarski (2012) have, however, shown that DTW can be used for more innovative tasks, such as time series *averaging* using the DTW discrepancy (see Schultz & Jain 2017 for a gentle introduction to these ideas). More generally, the idea of synthetising time series centroids can be regarded as a first attempt to *output* entire time series using DTW as a fitting loss. From a computational perspective, these approaches are, however, hampered by the fact that DTW is not differentiable and unstable when used in an optimization pipeline.

Soft-DTW. In parallel to these developments, several authors have considered smoothed modifications of Bellman's recursion to define smoothed DP distances (Bahl & Jelinek, 1975; Ristad & Yianilos, 1998) or kernels (Saigo et al., 2004; Cuturi et al., 2007). When applied to the DTW discrepancy, that regularization results in a *soft-DTW* score, which considers the *soft-minimum* of the distribution of *all costs* spanned by *all* possible alignments between two time series. Despite considering all alignments and not just the optimal one, soft-DTW can be computed with a minor modification of Bellman's recursion, in which all $(\min, +)$ operations are replaced with $(+, \times)$. As a result, both DTW and soft-DTW have quadratic in time & linear in space complexity with respect to the sequences' lengths. Because soft-DTW can be used with kernel machines, one typically observes an increase in performance when using soft-DTW over DTW (Cuturi, 2011) for classification.

Our contributions. We explore in this paper another important benefit of smoothing DTW: unlike the original DTW discrepancy, soft-DTW is *differentiable* in all of its arguments. We show that the gradients of soft-DTW w.r.t to all of its variables can be computed as a by-product of the computation of the discrepancy itself, with an added quadratic storage cost. We use this fact to propose an alternative approach to the DBA (DTW Barycenter Averaging) clustering algorithm of (Petitjean et al., 2011), and observe that our smoothed approach significantly outperforms known baselines for that task. More generally, we propose to use soft-DTW as a *fitting term* to compare the output of a machine synthesizing a time series segment with a ground truth observation, in the same way that, for instance, a regularized Wasserstein distance was used to compute barycenters (Cuturi & Doucet, 2014), and later to fit discriminators that output histograms (Zhang et al., 2015; Rolet et al., 2016). When paired with a flexible learning architecture such as a neural network, soft-DTW allows for a differentiable end-to-end approach to design predictive and generative models for time series, as illustrated in Figure 1. Source code is available at `https://github.com/mblondel/soft-dtw`.

Structure. After providing background material, we show in §2 how soft-DTW can be differentiated w.r.t the locations of two time series. We follow in §3 by illustrating how these results can be directly used for tasks that require to output time series: averaging, clustering and prediction of time series. We close this paper with experimental results in §4 that showcase each of these potential applications.

Notations. We consider in what follows multivariate discrete time series of varying length taking values in $\Omega \subset \mathbb{R}^p$. A time series can be thus represented as a matrix of p lines and varying number of columns. We consider a differentiable substitution-cost function $\delta : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}_+$ which will be, in most cases, the quadratic Euclidean distance between two vectors. For an integer n we write $[\![n]\!]$ for the set $\{1, \dots, n\}$ of integers. Given two series' lengths n and m, we write $\mathcal{A}_{n,m} \subset \{0,1\}^{n \times m}$ for the set of (binary) alignment matrices, that is paths on a $n \times m$ matrix that connect the upper-left $(1,1)$ matrix entry to the lower-right (n,m) one using only $\downarrow, \rightarrow, \searrow$ moves. The cardinal of $\mathcal{A}_{n,m}$ is known as the delannoy$(n-1, m-1)$ number; that number grows exponentially with m and n.

2. The DTW and soft-DTW loss functions

We propose in this section a unified formulation for the original DTW discrepancy (Sakoe & Chiba, 1978) and the Global Alignment kernel (GAK) (Cuturi et al., 2007), which can be both used to compare two time series $\mathbf{x} = (x_1, \dots, x_n) \in \mathbb{R}^{p \times n}$ and $\mathbf{y} = (y_1, \dots, y_m) \in \mathbb{R}^{p \times m}$.

2.1. Alignment costs: optimality and sum

Given the cost matrix $\Delta(\mathbf{x}, \mathbf{y}) \coloneqq \big[\delta(x_i, y_j)\big]_{ij} \in \mathbb{R}^{n \times m}$, the inner product $\langle A, \Delta(\mathbf{x}, \mathbf{y}) \rangle$ of that matrix with an alignment matrix A in $\mathcal{A}_{n,m}$ gives the score of A, as illustrated in Figure 2. Both DTW and GAK consider the costs of all possible alignment matrices, yet do so differently:

$$\begin{aligned} \mathrm{DTW}(\mathbf{x}, \mathbf{y}) &\coloneqq \min_{A \in \mathcal{A}_{n,m}} \langle A, \Delta(\mathbf{x}, \mathbf{y}) \rangle, \\ k_{\mathrm{GA}}^{\gamma}(\mathbf{x}, \mathbf{y}) &\coloneqq \sum_{A \in \mathcal{A}_{n,m}} e^{-\langle A, \Delta(\mathbf{x}, \mathbf{y}) \rangle / \gamma}. \end{aligned} \tag{1}$$

DP Recursion. Sakoe & Chiba (1978) showed that the Bellman equation (1952) can be used to compute DTW. That recursion, which appears in line 5 of Algorithm 1 (disregarding for now the exponent γ), only involves $(\min, +)$ operations. When considering kernel k_{GA}^{γ} and, instead, its integration over all alignments (see e.g. Lasserre 2009), Cuturi et al. (2007, Theorem 2) and the highly related formulation of Saigo et al. (2004, p.1685) use an old algorithmic appraoch (Bahl & Jelinek, 1975) which consists in *(i)* replacing all costs by their neg-exponential; *(ii)* replace $(\min, +)$ operations with $(+, \times)$ operations. These two recursions can be in fact unified with the use of a soft-

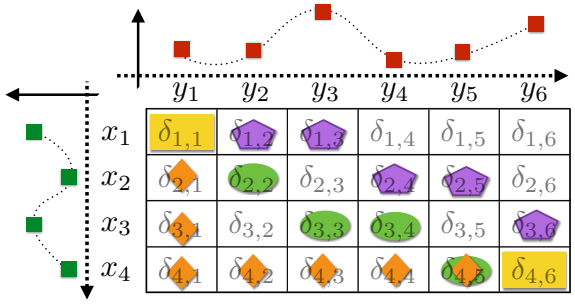


Figure 2. Three alignment matrices (orange, green, purple, in addition to the top-left and bottom-right entries) between two time series of length 4 and 6. The cost of an alignment is equal to the sum of entries visited along the path. DTW only considers the optimal alignment (here depicted in purple pentagons), whereas soft-DTW considers all delannoy$(n-1, m-1)$ possible alignment matrices.

minimum operator, which we present below.

Unified algorithm Both formulas in Eq. (1) can be computed with a single algorithm. That formulation is new to our knowledge. Consider the following generalized $\min$ operator, with a smoothing parameter $\gamma \geq 0$:

$$\min{}^{\gamma}\{a_1, \dots, a_n\} := \begin{cases} \min_{i \leq n} a_i, & \gamma = 0, \\ -\gamma \log \sum_{i=1}^{n} e^{-a_i/\gamma}, & \gamma > 0. \end{cases}$$

With that operator, we can define γ-soft-DTW:

$$\boxed{\mathbf{dtw}_\gamma(\mathbf{x}, \mathbf{y}) := \min{}^{\gamma}\{\langle A, \Delta(\mathbf{x}, \mathbf{y})\rangle, A \in \mathcal{A}_{n,m}\}.}$$

The original DTW score is recovered by setting γ to 0. When $\gamma > 0$, we recover $\mathbf{dtw}_\gamma = -\gamma \log k^{\gamma}_{\mathrm{GA}}$. Most importantly, and in either case, $\mathbf{dtw}_\gamma$ can be computed using Algorithm 1, which requires (nm) operations and (nm) storage cost as well . That cost can be reduced to $2n$ with a more careful implementation if one only seeks to compute $\mathbf{dtw}_\gamma(\mathbf{x}, \mathbf{y})$, but the backward pass we consider next requires the entire matrix R of intermediary alignment costs. Note that, to ensure numerical stability, the operator $\min^{\gamma}$ must be computed using the usual log-sum-exp stabilization trick, namely that $\log \sum_i e^{z_i} = (\max_j z_j) + \log \sum_i e^{z_i - \max_j z_j}$.

2.2. Differentiation of soft-DTW

A small variation in the input $\mathbf{x}$ causes a small change in $\mathbf{dtw}_0(\mathbf{x}, \mathbf{y})$ or $\mathbf{dtw}_\gamma(\mathbf{x}, \mathbf{y})$. When considering $\mathbf{dtw}_0$, that change can be efficiently monitored only when the optimal alignment matrix $A^\star$ that arises when computing $\mathbf{dtw}_0(\mathbf{x}, \mathbf{y})$ in Eq. (1) is unique. As the minimum over a finite set of linear functions of Δ, $\mathbf{dtw}_0$ is therefore locally differentiable w.r.t. the cost matrix Δ, with gradient $A^\star$, a fact that has been exploited in all algorithms designed to

Algorithm 1 Forward recursion to compute $\mathbf{dtw}_\gamma(\mathbf{x}, \mathbf{y})$ and intermediate alignment costs

Inputs: $\mathbf{x}, \mathbf{y}$, smoothing $\gamma \geq 0$, distance function δ
$r_{0,0} = 0; r_{i,0} = r_{0,j} = \infty; i \in [\![n]\!], j \in [\![m]\!]$
for $j = 1, \dots, m$ **do**
  for $i = 1, \dots, n$ **do**
    $r_{i,j} = \delta(x_i, y_j) + \min^{\gamma}\{r_{i-1,j-1}, r_{i-1,j}, r_{i,j-1}\}$
  end for
end for
Output: $(r_{n,m}, R)$

average time series under the DTW metric (Petitjean et al., 2011; Schultz & Jain, 2017). To recover the gradient of $\mathbf{dtw}_0(\mathbf{x}, \mathbf{y})$ w.r.t. $\mathbf{x}$, we only need to apply the chain rule, thanks to the differentiability of the cost function:

$$\nabla_{\mathbf{x}} \mathbf{dtw}_0(\mathbf{x}, \mathbf{y}) = \left(\frac{\partial \Delta(\mathbf{x}, \mathbf{y})}{\partial \mathbf{x}}\right)^T A^\star, \qquad (2)$$

where $\partial \Delta(\mathbf{x}, \mathbf{y}) / \partial \mathbf{x}$ is the Jacobian of Δ w.r.t. $\mathbf{x}$, a linear map from $\mathbb{R}^{p \times n}$ to $\mathbb{R}^{n \times m}$. When δ is the squared Euclidean distance, the transpose of that Jacobian applied to a matrix $B \in \mathbb{R}^{n \times m}$ is ($\circ$ being the elementwise product):

$$(\partial \Delta(\mathbf{x}, \mathbf{y}) / \partial \mathbf{x})^T B = 2\left((\mathbf{1}_p \mathbf{1}_m^T B^T) \circ \mathbf{x} - \mathbf{y} B^T\right).$$

With continuous data, $A^\star$ is almost always likely to be unique, and therefore the gradient in Eq. (2) will be defined almost everywhere. However, that gradient, when it exists, will be discontinuous around those values $\mathbf{x}$ where a small change in $\mathbf{x}$ causes a change in $A^\star$, which is likely to hamper the performance of gradient descent methods.

The case $\gamma > 0$. An immediate advantage of soft-DTW is that it can be explicitly differentiated, a fact that was also noticed by Saigo et al. (2006) in the related case of edit distances. When $\gamma > 0$, the gradient of Eq. (1) is obtained via the chain rule,

$$\nabla_{\mathbf{x}} \mathbf{dtw}_\gamma(\mathbf{x}, \mathbf{y}) = \left(\frac{\partial \Delta(\mathbf{x}, \mathbf{y})}{\partial \mathbf{x}}\right)^T \mathbb{E}_\gamma[A], \qquad (3)$$

$$\text{where} \quad \mathbb{E}_\gamma[A] := \frac{1}{k^{\gamma}_{\mathrm{GA}}(\mathbf{x}, \mathbf{y})} \sum_{A \in \mathcal{A}_{n,m}} e^{-\langle A, \Delta(\mathbf{x}, \mathbf{y})/\gamma \rangle} A,$$

is the average alignment matrix A under the Gibbs distribution $p_\gamma \propto e^{-\langle A, \Delta(\mathbf{x}, \mathbf{y})\rangle/\gamma}$ defined on all alignments in $\mathcal{A}_{n,m}$. The kernel $k^{\gamma}_{\mathrm{GA}}(\mathbf{x}, \mathbf{y})$ can thus be interpreted as the normalization constant of p_γ. Of course, since $\mathcal{A}_{n,m}$ has exponential size in n and m, a naive summation is not tractable. Although a Bellman recursion to compute that average alignment matrix $\mathbb{E}_\gamma[A]$ exists (see Appendix A) that computation has *quartic* $(n^2 m^2)$ complexity. Note that

this stands in stark contrast to the quadratic complexity obtained by Saigo et al. (2006) for edit-distances, which is due to the fact the sequences they consider can only take values in a *finite* alphabet. To compute the gradient of soft-DTW, we propose instead an algorithm that manages to remain *quadratic* (nm) in terms of complexity. The key to achieve this reduction is to apply the chain rule in *reverse* order of Bellman's recursion given in Algorithm 1, namely back-propagate. A similar idea was recently used to compute the gradient of ANOVA kernels in (Blondel et al., 2016).

2.3. Algorithmic differentiation

Differentiating algorithmically $\mathbf{dtw}_\gamma(\mathbf{x}, \mathbf{y})$ requires doing first a forward pass of Bellman's equation to store all intermediary computations and recover $R = [r_{i,j}]$ when running Algorithm 1. The value of $\mathbf{dtw}_\gamma(\mathbf{x}, \mathbf{y})$—stored in $r_{n,m}$ at the end of the forward recursion—is then impacted by a change in $r_{i,j}$ exclusively through the terms in which $r_{i,j}$ plays a role, namely the triplet of terms $r_{i+1,j}, r_{i,j+1}, r_{i+1,j+1}$. A straightforward application of the chain rule then gives

$$\underbrace{\frac{\partial r_{n,m}}{\partial r_{i,j}}}_{e_{i,j}} = \underbrace{\frac{\partial r_{n,m}}{\partial r_{i+1,j}}}_{e_{i+1,j}} \frac{\partial r_{i+1,j}}{\partial r_{i,j}} + \underbrace{\frac{\partial r_{n,m}}{\partial r_{i,j+1}}}_{e_{i,j+1}} \frac{\partial r_{i,j+1}}{\partial r_{i,j}} + \underbrace{\frac{\partial r_{n,m}}{\partial r_{i+1,j+1}}}_{e_{i+1,j+1}} \frac{\partial r_{i+1,j+1}}{\partial r_{i,j}},$$

in which we have defined the notation of the main object of interest of the backward recursion: $e_{i,j} \coloneqq \frac{\partial r_{n,m}}{\partial r_{i,j}}$. The Bellman recursion evaluated at $(i+1, j)$ as shown in line 5 of Algorithm 1 (here $\delta_{i+1,j}$ is $\delta(x_{i+1}, y_j)$) yields :

$$r_{i+1,j} = \delta_{i+1,j} + \min{}^\gamma\{r_{i,j-1}, r_{i,j}, r_{i+1,j-1}\},$$

which, when differentiated w.r.t $r_{i,j}$ yields the ratio:

$$\frac{\partial r_{i+1,j}}{\partial r_{i,j}} = e^{-r_{i,j}/\gamma} \Big/ \left(e^{-r_{i,j-1}/\gamma} + e^{-r_{i,j}/\gamma} + e^{-r_{i+1,j-1}/\gamma}\right).$$

The logarithm of that derivative can be conveniently cast using evaluations of $\min^\gamma$ computed in the forward loop:

$$\begin{aligned}\gamma \log \frac{\partial r_{i+1,j}}{\partial r_{i,j}} &= \min{}^\gamma\{r_{i,j-1}, r_{i,j}, r_{i+1,j-1}\} - r_{i,j} \\ &= r_{i+1,j} - \delta_{i+1,j} - r_{i,j}.\end{aligned}$$

Similarly, the following relationships can also be obtained:

$$\gamma \log \frac{\partial r_{i,j+1}}{\partial r_{i,j}} = r_{i,j+1} - r_{i,j} - \delta_{i,j+1},$$
$$\gamma \log \frac{\partial r_{i+1,j+1}}{\partial r_{i,j}} = r_{i+1,j+1} - r_{i,j} - \delta_{i+1,j+1}.$$

We have therefore obtained a *backward* recursion to compute the entire matrix $E = [e_{i,j}]$, starting from $e_{n,m} = \frac{\partial r_{n,m}}{\partial r_{n,m}} = 1$ down to $e_{1,1}$. To obtain $\nabla_{\mathbf{x}} \mathbf{dtw}_\gamma(\mathbf{x}, \mathbf{y})$, notice that the derivatives w.r.t. the entries of the cost matrix Δ can be computed by $\frac{\partial r_{n,m}}{\partial \delta_{i,j}} = \frac{\partial r_{n,m}}{\partial r_{i,j}} \frac{\partial r_{i,j}}{\partial \delta_{i,j}} = e_{i,j} \cdot 1 = e_{i,j}$, and therefore we have that

$$\nabla_{\mathbf{x}} \mathbf{dtw}_\gamma(\mathbf{x}, \mathbf{y}) = \left(\frac{\partial \Delta(\mathbf{x}, \mathbf{y})}{\partial \mathbf{x}}\right)^T E,$$

where E is exactly the average alignment $\mathbb{E}_\gamma[A]$ in Eq. (3). These computations are summarized in Algorithm 2, which, once Δ has been computed, has complexity nm in time and space. Because $\min^\gamma$ has a $1/\gamma$-Lipschitz continuous gradient, the gradient of $\mathbf{dtw}_\gamma$ is $2/\gamma$-Lipschitz continuous when δ is the squared Euclidean distance.

Algorithm 2 Backward recursion to compute $\nabla_{\mathbf{x}} \mathbf{dtw}_\gamma(\mathbf{x}, \mathbf{y})$

Inputs: $\mathbf{x}, \mathbf{y}$, smoothing $\gamma \geq 0$, distance function δ
$(\cdot, R) = \mathbf{dtw}_\gamma(\mathbf{x}, \mathbf{y})$, $\Delta = [\delta(x_i, y_j)]_{i,j}$
$\delta_{i,m+1} = \delta_{n+1,j} = 0, i \in [\![n]\!], j \in [\![m]\!]$
$e_{i,m+1} = e_{n+1,j} = 0, i \in [\![n]\!], j \in [\![m]\!]$
$r_{i,m+1} = r_{n+1,j} = -\infty, i \in [\![n]\!], j \in [\![m]\!]$
$\delta_{n+1,m+1} = 0, e_{n+1,m+1} = 1, r_{n+1,m+1} = r_{n,m}$
for $j = m, \dots, 1$ **do**
  for $i = n, \dots, 1$ **do**
    $a = \exp \frac{1}{\gamma}(r_{i+1,j} - r_{i,j} - \delta_{i+1,j})$
    $b = \exp \frac{1}{\gamma}(r_{i,j+1} - r_{i,j} - \delta_{i,j+1})$
    $c = \exp \frac{1}{\gamma}(r_{i+1,j+1} - r_{i,j} - \delta_{i+1,j+1})$
    $e_{i,j} = e_{i+1,j} \cdot a + e_{i,j+1} \cdot b + e_{i+1,j+1} \cdot c$
  end for
end for
Output: $\nabla_{\mathbf{x}} \mathbf{dtw}_\gamma(\mathbf{x}, \mathbf{y}) = \left(\frac{\partial \Delta(\mathbf{x}, \mathbf{y})}{\partial \mathbf{x}}\right)^T E$

3. Learning with the soft-DTW loss

3.1. Averaging with the soft-DTW geometry

We study in this section a direct application of Algorithm 2 to the problem of computing Fréchet means (1948) of time series with respect to the $\mathbf{dtw}_\gamma$ discrepancy. Given a family of N times series $\mathbf{y}_1, \dots, \mathbf{y}_N$, namely N matrices of p lines and varying number of columns, $m_1, \dots, m_N$, we are interested in defining a single barycenter time series $\mathbf{x}$ for that family under a set of normalized weights $\lambda_1, \dots, \lambda_N \in \mathbb{R}_+$ such that $\sum_{i=1}^N \lambda_i = 1$. Our goal is thus to solve approximately the following problem, in which we have assumed that $\mathbf{x}$ has fixed length n:

$$\min_{\mathbf{x} \in \mathbb{R}^{p \times n}} \sum_{i=1}^N \frac{\lambda_i}{m_i} \mathbf{dtw}_\gamma(\mathbf{x}, \mathbf{y}_i). \tag{4}$$

Note that each $\mathbf{dtw}_\gamma(\mathbf{x}, \mathbf{y}_i)$ term is divided by m_i, the length of $\mathbf{y}_i$. Indeed, since $\mathbf{dtw}_0$ is an increasing (roughly linearly) function of each of the input lengths n and m_i, we follow the convention of normalizing in practice each discrepancy by $n \times m_i$. Since the length n of $\mathbf{x}$ is here fixed across all evaluations, we do not need to divide the objective of Eq. (4) by n. Averaging under the soft-DTW geometry results in substantially different results than those that can be obtained with the Euclidean geometry (which can only be used in the case where all lengths $n = m_1 = \cdots =$

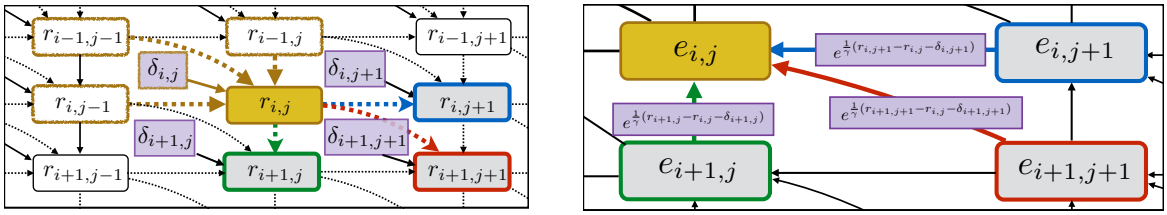
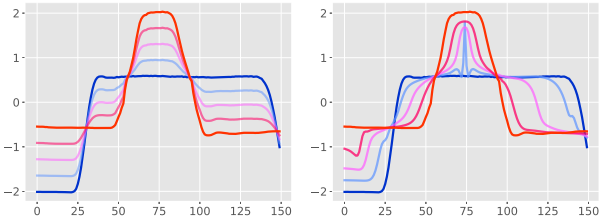


Figure 3. Sketch of the computational graph for soft-DTW, in the forward pass used to compute $\mathbf{dtw}_\gamma$ (left) and backward pass used to compute its gradient $\nabla_{\mathbf{x}} \mathbf{dtw}_\gamma$ (right). In both diagrams, purple shaded cells stand for data values available before the recursion starts, namely cost values (left) and multipliers computed using forward pass results (right). In the left diagram, the forward computation of $r_{i,j}$ as a function of its predecessors and $\delta_{i,j}$ is summarized with arrows. Dotted lines indicate a $\min^\gamma$ operation, solid lines an addition. From the perspective of the final term $r_{n,m}$, which stores $\mathbf{dtw}_\gamma(\mathbf{x}, \mathbf{y})$ at the lower right corner (not shown) of the computational graph, a change in $r_{i,j}$ only impacts $r_{n,m}$ through changes that $r_{i,j}$ causes to $r_{i+1,j}$, $r_{i,j+1}$ and $r_{i+1,j+1}$. These changes can be tracked using Eq. (2.3,2.3) and appear in lines 9-11 in Algorithm 2 as variables a, b, c, as well as in the purple shaded boxes in the backward pass (right) which represents the recursion of line 12 in Algorithm 2.

m_N are equal), as can be seen in the intuitive interpolations we obtain between two time series shown in Figure 4.

Non-convexity of $\mathbf{dtw}_\gamma$. A natural question that arises from Eq. (4) is whether that objective is convex or not. The answer is negative, in a way that echoes the non-convexity of the k-means objective as a function of cluster centroids locations. Indeed, for any alignment matrix A of suitable size, each map $\mathbf{x} \mapsto \langle A, \Delta(\mathbf{x}, \mathbf{y}) \rangle$ shares the same convexity/concavity property that δ may have. However, both $\min$ and $\min^\gamma$ can only preserve the *concavity* of elementary functions (Boyd & Vandenberghe, 2004, pp.72-74). Therefore $\mathbf{dtw}_\gamma$ will only be concave if δ is concave, or become instead a (non-convex) (soft) minimum of convex functions if δ is convex. When δ is a squared-Euclidean distance, $\mathbf{dtw}_0$ is a piecewise quadratic function of $\mathbf{x}$, as is also the case with the k-means energy (see for instance Figure 2 in Schultz & Jain 2017). Since this is the setting we consider here, all of the computations involving barycenters should be taken with a grain of salt, since we have no way of ensuring optimality when approximating Eq. (4).

Smoothing helps optimizing $\mathbf{dtw}_\gamma$. Smoothing can be regarded, however, as a way to "convexify" $\mathbf{dtw}_\gamma$. Indeed, notice that $\mathbf{dtw}_\gamma$ converges to the sum of all costs as $\gamma \to \infty$. Therefore, if δ is convex, $\mathbf{dtw}_\gamma$ will gradually become convex as γ grows. For smaller values of γ, one can intuitively foresee that using $\min^\gamma$ instead of a minimum will smooth out local minima and therefore provide a better (although slightly different from $\mathbf{dtw}_0$) optimization landscape. We believe this is why our approach recovers better results, **even when measured in the original $\mathbf{dtw}_0$ discrepancy**, than subgradient or alternating minimization approaches such as DBA (Petitjean et al., 2011), which can, on the contrary, get more easily stuck in local minima. Evidence for this statement is presented in the experimental section.



(a) Euclidean loss (b) Soft-DTW loss ($\gamma = 1$)

Figure 4. Interpolation between two time series (red and blue) on the Gun Point dataset. We computed the barycenter by solving Eq. (4) with (λ_1, λ_2) set to (0.25, 0.75), (0.5, 0.5) and (0.75, 0.25). The soft-DTW geometry leads to visibly different interpolations.

3.2. Clustering with the soft-DTW geometry

The (approximate) computation of $\mathbf{dtw}_\gamma$ barycenters can be seen as a first step towards the task of clustering time series under the $\mathbf{dtw}_\gamma$ discrepancy. Indeed, one can naturally formulate that problem as that of finding centroids $\mathbf{x}_1, \ldots, \mathbf{x}_k$ that minimize the following energy:

$$\min_{\mathbf{x}_1, \ldots, \mathbf{x}_k \in \mathbb{R}^{p \times n}} \sum_{i=1}^{N} \frac{1}{m_i} \min_{j \in [[k]]} \mathbf{dtw}_\gamma(\mathbf{x}_j, \mathbf{y}_i). \tag{5}$$

To solve that problem one can resort to a direct generalization of Lloyd's algorithm (1982) in which each centering step and each clustering allocation step is done according to the $\mathbf{dtw}_\gamma$ discrepancy.

3.3. Learning prototypes for time series classification

One of the de-facto baselines for learning to classify time series is the k nearest neighbors (k-NN) algorithm, combined with DTW as discrepancy measure between time series. However, k-NN has two main drawbacks. First, the time series used for training must be stored, leading to potentially high storage cost. Second, in order to com-

pute predictions on new time series, the DTW discrepancy must be computed with all training time series, leading to high computational cost. Both of these drawbacks can be addressed by the nearest centroid classifier (Hastie et al., 2001, p.670), (Tibshirani et al., 2002). This method chooses the class whose barycenter (centroid) is closest to the time series to classify. Although very simple, this method was shown to be competitive with k-NN, while requiring much lower computational cost at prediction time (Petitjean et al., 2014). Soft-DTW can naturally be used in a nearest centroid classifier, in order to compute the barycenter of each class at train time, and to compute the discrepancy between barycenters and time series, at prediction time.

3.4. Multistep-ahead prediction

Soft-DTW is ideally suited as a loss function for any task that requires time series outputs. As an example of such a task, we consider the problem of, given the first $1, \dots, t$ observations of a time series, predicting the remaining $(t+1), \dots, n$ observations. Let $\mathbf{x}^{t,t'} \in \mathbb{R}^{p \times (t'-t+1)}$ be the submatrix of $\mathbf{x} \in \mathbb{R}^{p \times n}$ of all columns with indices between t and t', where $1 \leq t < t' < n$. Learning to predict the segment of a time series can be cast as the problem

$$\min_{\theta \in \Theta} \sum_{i=1}^{N} \mathbf{dtw}_\gamma \left(f_\theta(\mathbf{x}_i^{1,t}), \mathbf{x}_i^{t+1,n} \right),$$

where $\{f_\theta\}$ is a set of parameterized function that take as input a time series and outputs a time series. Natural choices would be multi-layer perceptrons or recurrent neural networks (RNN), which have been historically trained with a Euclidean loss (Parlos et al., 2000, Eq.5).

4. Experimental results

Throughout this section, we use the UCR (University of California, Riverside) time series classification archive (Chen et al., 2015). We use a subset containing 79 datasets encompassing a wide variety of fields (astronomy, geology, medical imaging) and lengths. Datasets include class information (up to 60 classes) for each time series and are split into train and test sets. Due to the large number of datasets in the UCR archive, we choose to report only a summary of our results in the main manuscript. Detailed results are included in the appendices for interested readers.

4.1. Averaging experiments

In this section, we compare the soft-DTW barycenter approach presented in §3.1 to DBA (Petitjean et al., 2011) and a simple batch subgradient method.

Experimental setup. For each dataset, we choose a class at random, pick 10 time series in that class and compute their barycenter. For quantitative results below, we repeat this procedure 10 times and report the averaged results. For each method, we set the maximum number of iterations to 100. To minimize the proposed soft-DTW barycenter objective, Eq. (4), we use L-BFGS.

Table 1. Percentage of the datasets on which the proposed soft-DTW barycenter is achieving lower DTW loss (Equation (4) with $\gamma = 0$) than competing methods.

	Random initialization	Euclidean mean initialization
Comparison with DBA		
$\gamma = 1$	40.51%	3.80%
$\gamma = 0.1$	93.67%	46.83%
$\gamma = 0.01$	100%	79.75%
$\gamma = 0.001$	97.47%	89.87%
Comparison with subgradient method		
$\gamma = 1$	96.20%	35.44%
$\gamma = 0.1$	97.47%	72.15%
$\gamma = 0.01$	97.47%	92.41%
$\gamma = 0.001$	97.47%	97.47%

Qualitative results. We first visualize the barycenters obtained by soft-DTW when $\gamma = 1$ and $\gamma = 0.01$, by DBA and by the subgradient method. Figure 5 shows barycenters obtained using random initialization on the ECG200 dataset. More results with both random and Euclidean mean initialization are given in Appendix B and C.

We observe that both DBA or soft-DTW with low smoothing parameter γ yield barycenters that are spurious. On the other hand, a descent on the soft-DTW loss with sufficiently high γ converges to a reasonable solution. For example, as indicated in Figure 5 with DTW or soft-DTW ($\gamma = 0.01$), the small kink around $x = 15$ is not representative of any of the time series in the dataset. However, with soft-DTW ($\gamma = 1$), the barycenter closely matches the time series. This suggests that DTW or soft-DTW with too low γ can get stuck in bad local minima.

When using Euclidean mean initialization (only possible if time series have the same length), DTW or soft-DTW with low γ often yield barycenters that better match the shape of the time series. However, they tend to overfit: they absorb the idiosyncrasies of the data. In contrast, soft-DTW is able to learn barycenters that are much smoother.

Quantitative results. Table 1 summarizes the percentage of datasets on which the proposed soft-DTW barycenter achieves lower DTW loss when varying the smoothing parameter γ. The actual loss values achieved by different methods are indicated in Appendix G and Appendix H.

As γ decreases, soft-DTW achieves a lower DTW loss than other methods on almost all datasets. This confirms our

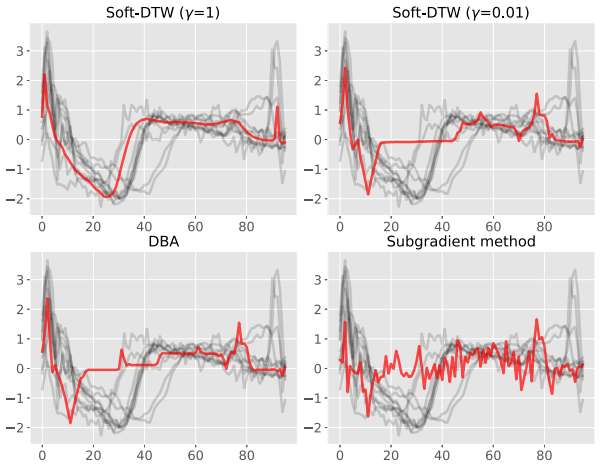
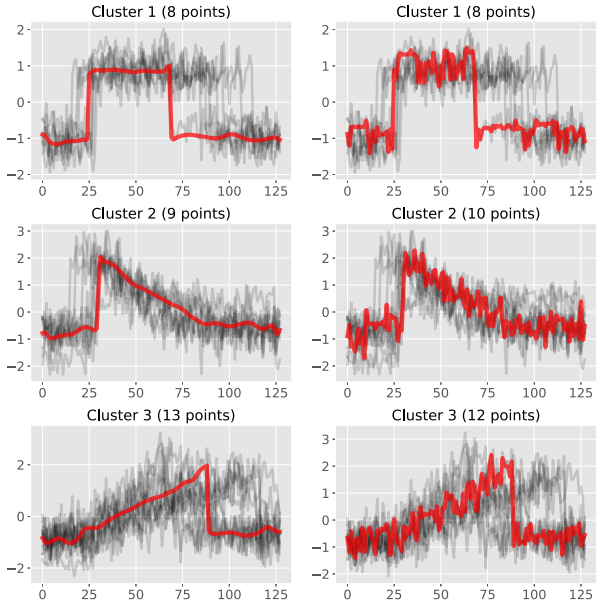


Figure 5. Comparison between our proposed soft barycenter and the barycenter obtained by DBA and the subgradient method, on the ECG200 dataset. When DTW is insufficiently smoothed, barycenters often get stuck in a bad local minimum that does not correctly match the time series.



(a) Soft-DTW ($\gamma = 1$) (b) DBA

Figure 6. Clusters obtained on the CBF dataset when plugging our proposed soft barycenter and that of DBA in Lloyd's algorithm. DBA absorbs the idiosyncrasies of the data, while soft-DTW can learn much smoother barycenters.

claim that the smoothness of soft-DTW leads to an objective that is better behaved and more amenable to optimization by gradient-descent methods.

4.2. k-means clustering experiments

We consider in this section the same computational tools used in §4.1 above, but use them to cluster time series.

Experimental setup. For all datasets, the number of clusters k is equal to the number of classes available in the dataset. Lloyd's algorithm alternates between a centering step (barycenter computation) and an assignment step. We set the maximum number of outer iterations to 30 and the maximum number of inner (barycenter) iterations to 100, as before. Again, for soft-DTW, we use L-BFGS.

Qualitative results. Figure 6 shows the clusters obtained when runing Lloyd's algorithm on the CBF dataset with soft-DTW ($\gamma = 1$) and DBA, in the case of random initialization. More results are included in Appendix E. Clearly, DTW absorbs the tiny details in the data, while soft-DTW is able to learn much smoother barycenters.

Quantitative results. Table 2 summarizes the percentage of datasets on which soft-DTW barycenter achieves lower k-means loss under DTW, i.e. Eq. (5) with $\gamma = 0$. The actual loss values achieved by all methods are indicated in Appendix I and Appendix J. The results confirm the same trend as for the barycenter experiments. Namely, as γ decreases, soft-DTW is able to achieve lower loss than other methods on a large proportion of the datasets. Note that we have not run experiments with smaller values of γ than 0.001, since $\mathbf{dtw}_{0.001}$ is very close to $\mathbf{dtw}_0$ in practice.

4.3. Time-series classification experiments

In this section, we investigate whether the smoothing in soft-DTW can act as a useful regularization and improve classification accuracy in the nearest centroid classifier.

Experimental setup. We use 50% of the data for training, 25% for validation and 25% for testing. We choose γ from 15 log-spaced values between 10^{-3} and 10.

Quantitative results. Each point in Figure 7 above the diagonal line represents a dataset for which using soft-DTW for barycenter computation rather than DBA improves the accuracy of the nearest centroid classifier. To summarize, we found that soft-DTW is working better or at least as well as DBA in 75% of the datasets.

4.4. Multistep-ahead prediction experiments

In this section, we present preliminary experiments for the task of multistep-ahead prediction, described in §3.4.

Experimental setup. We use the training and test sets predefined in the UCR archive. In both the training and test sets, we use the first 60% of the time series as input and the remaining 40% as output, ignoring class information. We then use the training set to learn a model that predicts the outputs from inputs and the test set to evaluate results with both Euclidean and DTW losses. In this experiment, we focus on a simple multi-layer perceptron (MLP) with one

Table 2. Percentage of the datasets on which the proposed soft-DTW based k-**means** is achieving lower DTW loss (Equation (5) with $\gamma = 0$) than competing methods.

	Random initialization	Euclidean mean initialization
Comparison with DBA		
$\gamma = 1$	15.78%	29.31%
$\gamma = 0.1$	24.56%	24.13%
$\gamma = 0.01$	59.64%	55.17%
$\gamma = 0.001$	77.19%	68.97%
Comparison with subgradient method		
$\gamma = 1$	42.10%	46.44%
$\gamma = 0.1$	57.89%	50%
$\gamma = 0.01$	76.43%	65.52%
$\gamma = 0.001$	96.49%	84.48%

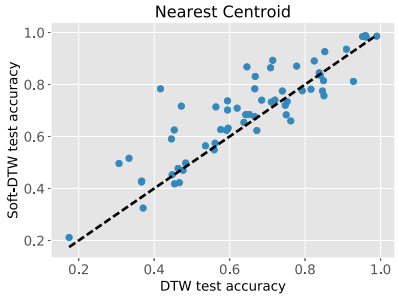


Figure 7. Each point above the diagonal represents a dataset where using our soft-DTW barycenter rather than that of DBA improves the accuracy of the nearest nearest centroid classifier. This is the case for **75%** of the datasets in the UCR archive.

hidden layer and sigmoid activation. We also experimented with linear models and recurrent neural networks (RNNs) but they did not improve over a simple MLP.

Implementation details. Deep learning frameworks such as Theano, TensorFlow and Chainer allow the user to specify a custom backward pass for their function. Implementing such a backward pass, rather than resorting to automatic differentiation (autodiff), is particularly important in the case of soft-DTW: First, the autodiff in these frameworks is designed for vectorized operations, whereas the dynamic program used by the forward pass of Algorithm 2 is inherently element-wise; Second, as we explained in §2.2, our backward pass is able to re-use log-sum-exp computations from the forward pass, leading to both lower computational cost and better numerical stability. We implemented a custom backward pass in Chainer, which can then be used to plug soft-DTW as a loss function in any network architecture. To estimate the MLP's parameters, we used Chainer's implementation of Adam (Kingma & Ba, 2014).

Qualitative results. Visualizations of the predictions obtained under Euclidean and soft-DTW losses are given in Figure 1, as well as in Appendix F. We find that for simple one-dimensional time series, an MLP works very well, showing its ability to capture patterns in the training set. Although the predictions under Euclidean and soft-DTW losses often agree with each other, they can sometimes be visibly different. Predictions under soft-DTW loss can confidently predict abrupt and sharp changes since those have a low DTW cost as long as such a sharp change is present, under a small time shift, in the ground truth.

Table 3. Averaged rank obtained by a multi-layer perceptron (MLP) under Euclidean and soft-DTW losses. Euclidean initialization means that we initialize the MLP trained with soft-DTW loss by the solution of the MLP trained with Euclidean loss.

Training loss	Random initialization	Euclidean initialization
When evaluating with DTW loss		
Euclidean	3.46	4.21
soft-DTW ($\gamma = 1$)	3.55	3.96
soft-DTW ($\gamma = 0.1$)	3.33	3.42
soft-DTW ($\gamma = 0.01$)	2.79	2.12
soft-DTW ($\gamma = 0.001$)	**1.87**	**1.29**
When evaluating with Euclidean loss		
Euclidean	**1.05**	**1.70**
soft-DTW ($\gamma = 1$)	2.41	2.99
soft-DTW ($\gamma = 0.1$)	3.42	3.38
soft-DTW ($\gamma = 0.01$)	4.13	3.64
soft-DTW ($\gamma = 0.001$)	3.99	3.29

Quantitative results. A comparison summary of our MLP under Euclidean and soft-DTW losses over the UCR archive is given in Table 3. Detailed results are given in the appendix. Unsurprisingly, we achieve lower DTW loss when training with the soft-DTW loss, and lower Euclidean loss when training with the Euclidean loss. Because DTW is robust to several useful invariances, a small error in the soft-DTW sense could be a more judicious choice than an error in an Euclidean sense for many applications.

5. Conclusion

We propose in this paper to turn the popular DTW discrepancy between time series into a full-fledged loss function between ground truth time series and outputs from a learning machine. We have shown experimentally that, on the existing problem of computing barycenters and clusters for time series data, our computational approach is superior to existing baselines. We have shown promising results on the problem of multistep-ahead time series prediction, which could prove extremely useful in settings where a user's actual loss function for time series is closer to the robust perspective given by DTW, than to the local parsing of the Euclidean distance.

Acknowledgements. MC gratefully acknowledges the support of a *chaire de l'IDEX Paris Saclay*.

References

Bahl, L and Jelinek, Frederick. Decoding for channels with insertions, deletions, and substitutions with applications to speech recognition. *IEEE Transactions on Information Theory*, 21(4):404–411, 1975.

Bakir, GH, Hofmann, T, Schölkopf, B, Smola, AJ, Taskar, B, and Vishwanathan, SVN. *Predicting Structured Data*. Advances in neural information processing systems. MIT Press, Cambridge, MA, USA, 2007.

Bellman, Richard. On the theory of dynamic programming. *Proceedings of the National Academy of Sciences*, 38(8): 716–719, 1952.

Blondel, Mathieu, Fujino, Akinori, Ueda, Naonori, and Ishihata, Masakazu. Higher-order factorization machines. In *Advances in Neural Information Processing Systems 29*, pp. 3351–3359. 2016.

Boyd, Stephen and Vandenberghe, Lieven. *Convex Optimization*. Cambridge University Press, 2004.

Chen, Yanping, Keogh, Eamonn, Hu, Bing, Begum, Nurjahan, Bagnall, Anthony, Mueen, Abdullah, and Batista, Gustavo. The ucr time series classification archive, July 2015. www.cs.ucr.edu/~eamonn/time_series_data/.

Cuturi, Marco. Fast global alignment kernels. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 929–936, 2011.

Cuturi, Marco and Doucet, Arnaud. Fast computation of Wasserstein barycenters. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 685–693, 2014.

Cuturi, Marco, Vert, Jean-Philippe, Birkenes, Oystein, and Matsui, Tomoko. A kernel for time series based on global alignments. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, volume 2, pp. II–413, 2007.

Fréchet, Maurice. Les éléments aléatoires de nature quelconque dans un espace distancié. In *Annales de l'institut Henri Poincaré*, volume 10, pp. 215–310. Presses universitaires de France, 1948.

Garreau, Damien, Lajugie, Rémi, Arlot, Sylvain, and Bach, Francis. Metric learning for temporal sequence alignment. In *Advances in Neural Information Processing Systems*, pp. 1817–1825, 2014.

Hastie, Trevor, Tibshirani, Robert, and Friedman, Jerome. *The Elements of Statistical Learning*. Springer New York Inc., 2001.

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Lasserre, Jean B. *Linear and integer programming vs linear integration and counting: a duality viewpoint*. Springer Science & Business Media, 2009.

Lloyd, Stuart. Least squares quantization in pcm. *IEEE Trans. on Information Theory*, 28(2):129–137, 1982.

Lütkepohl, Helmut. *New introduction to multiple time series analysis*. Springer Science & Business Media, 2005.

Parlos, Alexander G, Rais, Omar T, and Atiya, Amir F. Multi-step-ahead prediction using dynamic recurrent neural networks. *Neural networks*, 13(7):765–786, 2000.

Petitjean, François and Gançarski, Pierre. Summarizing a set of time series by averaging: From steiner sequence to compact multiple alignment. *Theoretical Computer Science*, 414(1):76–91, 2012.

Petitjean, François, Ketterlin, Alain, and Gançarski, Pierre. A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recognition*, 44 (3):678–693, 2011.

Petitjean, François, Forestier, Germain, Webb, Geoffrey I, Nicholson, Ann E, Chen, Yanping, and Keogh, Eamonn. Dynamic time warping averaging of time series allows faster and more accurate classification. In *ICDM*, pp. 470–479. IEEE, 2014.

Ristad, Eric Sven and Yianilos, Peter N. Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):522–532, 1998.

Rolet, A., Cuturi, M., and Peyré, G. Fast dictionary learning with a smoothed Wasserstein loss. *Proceedings of AISTATS'16*, 2016.

Saigo, Hiroto, Vert, Jean-Philippe, Ueda, Nobuhisa, and Akutsu, Tatsuya. Protein homology detection using string alignment kernels. *Bioinformatics*, 20(11):1682–1689, 2004.

Saigo, Hiroto, Vert, Jean-Philippe, and Akutsu, Tatsuya. Optimizing amino acid substitution matrices with a local alignment kernel. *BMC bioinformatics*, 7(1):246, 2006.

Sakoe, Hiroaki and Chiba, Seibi. A dynamic programming approach to continuous speech recognition. In *Proceedings of the Seventh International Congress on Acoustics, Budapest*, volume 3, pp. 65–69, 1971.

Sakoe, Hiroaki and Chiba, Seibi. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans. on Acoustics, Speech, and Sig. Proc.*, 26:43–49, 1978.

Schultz, David and Jain, Brijnesh. Nonsmooth analysis and subgradient methods for averaging in dynamic time warping spaces. *arXiv preprint arXiv:1701.06393*, 2017.

Tibshirani, Robert, Hastie, Trevor, Narasimhan, Balasubramanian, and Chu, Gilbert. Diagnosis of multiple cancer types by shrunken centroids of gene expression. *Proceedings of the National Academy of Sciences*, 99(10): 6567–6572, 2002.

Yi, Byoung-Kee, Jagadish, HV, and Faloutsos, Christos. Efficient retrieval of similar time sequences under time warping. In *Data Engineering, 1998. Proceedings., 14th International Conference on*, pp. 201–208. IEEE, 1998.

Zhang, C., Frogner, C., Mobahi, H., Araya-Polo, M., and Poggio, T. Learning with a Wasserstein loss. *Advances in Neural Information Processing Systems 29*, 2015.