Data Analysis with Soft-DTW

Ilan Bacry
MVA
ilan.bacry20@gmail.com

Théo Basseras
MVA
theo.basseras@student.isae-supaero.fr

December 9, 2025

Abstract

Dynamic Time Warping (DTW) is one of the most classical discrepancies for time-series comparison. Unlike the Euclidean distance, DTW is robust to temporal distortions and can align sequences that are stretched, compressed, or phase-shifted. However, DTW is not differentiable, which limits its integration into gradient-based learning methods. Soft-DTW, introduced by Cuturi and Blondel (2017), provides a smooth relaxation of DTW that preserves its alignment geometry while making the loss differentiable. This report reviews DTW, presents the soft-DTW formulation and its gradient computation, and discusses applications to averaging, clustering, classification, and forecasting.

1 Introduction

Comparing time series is a fundamental problem in data analysis. Because real-world signals often exhibit temporal variability (such as local delays, speed changes, or phase shifts), classical pointwise distances like the Euclidean metric become ineffective, as they assume strict one-to-one alignment.

Dynamic Time Warping (DTW) addresses this issue by allowing non-linear alignments that preserve temporal order while compensating for local distortions. However, a major limitation of classical DTW is that it is not differentiable, which prevents its use in gradient-based learning methods such as neural networks. In addition, DTW is sensitive to amplitude and offset differences, which motivates standard preprocessing steps such as centering and normalization.

To address these limitations, Cuturi and Blondel [1] introduced soft-DTW, a smooth and differentiable relaxation of DTW that preserves its alignment geometry while enabling gradient-based optimization.

In this article, we first review classical DTW, then introduce soft-DTW and its gradient. We next describe the experimental protocols and present the corresponding numerical results. We conclude with a brief discussion of the method's limitations and future directions.

2 Dynamic Time Warping (DTW)

2.1 Formal Definition and Warping Paths

Let $x = (x_1, \dots, x_n)$ and $y = (y_1, \dots, y_m)$ be two time series. For each pair (i, j), let $\delta(x_i, y_j)$ be the local alignment cost, and let $\Delta(x, y)$ denote the $n \times m$ matrix with entries $\Delta_{ij} = \delta(x_i, y_j)$.

A warping path is represented by a binary matrix $A = (a_{ij}) \in \{0, 1\}^{n \times m}$, where $a_{ij} = 1$ indicates that the pair (i, j) is in the alignment. Valid paths (denoted $\mathcal{A}_{n,m}$) satisfy boundary conditions $(1, 1)$ and (n, m), are monotone and allow only steps $(1, 0)$, $(0, 1)$, or $(1, 1)$.

The DTW distance is then defined as

$$\mathrm{DTW}(x, y) = \min_{A \in \mathcal{A}_{n,m}} \sum_{i,j} a_{ij}\, \delta(x_i, y_j).$$

2.2 Limitations of Euclidean Distance for Time-Series Comparison

The Euclidean distance is a natural first choice to compare two time series, but it suffers from key limitations. It enforces strict pointwise alignment: the i-th sample of one series is compared only to the i-th sample of the other. Consequently, small temporal distortions (local shifts, stretches, or compressions) can lead to large Euclidean costs, even when the underlying patterns are similar. Two sequences that differ only by a slight delay or speed variation may therefore appear highly dissimilar under Euclidean distance despite sharing the same structure.

2.3 Preprocessing Requirements: Centering and Normalization

Another essential issue is amplitude and offset variability. If the two time series do not have the same scale or baseline level, Euclidean distance becomes dominated by these differences rather than by their temporal patterns. For this reason, it is crucial to **center** (remove the mean) and **normalize** (rescale the variance or amplitude) of the signals before comparing them. Only after this preprocessing step does DTW meaningfully capture the similarity between the underlying patterns.

2.4 DTW: A Shape-Aware Alternative

DTW overcomes these issues by allowing nonlinear alignments of the time axis. Instead of enforcing strict pointwise comparisons, it matches samples through an admissible warping path, which enables the comparison of sequences based on their temporal shape rather than their sampling synchrony.

2.5 Differentiability Issue

Despite its advantages, classical DTW has a critical limitation: it is **not differentiable**. The minimization over discrete warping paths results in a piecewise-constant objective that lacks gradients, preventing its use as a loss function in neural network training or any gradient-based optimization procedure. This non-differentiability motivates the smoothed relaxation introduced in the next section.

3 Soft-DTW

3.1 Definition

Definition 1 (Soft-min operator)**.** *For $a = (a_1, \dots, a_n) \in \mathbb{R}^n$ and $\gamma \geq 0$, the soft-min operator is*

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

Definition 2 (Soft-DTW)**.** *Let $x = (x_1, \dots, x_n) \in \mathbb{R}^n$ and $y = (y_1, \dots, y_m) \in \mathbb{R}^m$ be two time series. Let $\mathcal{A}_{n,m}$ denote the set of admissible binary alignment matrices (as defined in Section 2). For a pointwise cost function $\delta : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, define the pairwise cost matrix $\Delta(x, y) \in \mathbb{R}^{n \times m}$ by*

$$\Delta_{ij} = \delta(x_i, y_j) \quad \textit{(typically } \delta(x_i, y_j) = \|x_i - y_j\|^2\textit{)}.$$

The soft-DTW discrepancy *between x and y, for a smoothing parameter $\gamma \geq 0$, is*

$$\mathrm{dtw}_\gamma(x, y) = \min^{\gamma} \left\{ \langle A, \Delta(x, y) \rangle_F \ : \ A \in \mathcal{A}_{n,m} \right\},$$

where $\min^{\gamma}$ is the soft-min operator and $\langle \cdot, \cdot \rangle_F$ the Frobenius inner product.

Remark. For $\gamma = 0$, dtw_0 coincides with classical DTW. For $\gamma > 0$, the soft-min smooths the minimum over alignments, making dtw_γ differentiable with respect to x and y.

3.2 Forward Recursion

Soft-DTW can be computed via dynamic programming by replacing the hard minimum in the classical DTW recursion with the soft-min operator. Let $r_{i,j}$ denote the dynamic programming table with boundary conditions $r_{0,0} = 0$ and $r_{i,0} = r_{0,j} = +\infty$ for all $i, j \geq 1$. The forward recursion is

$$r_{i,j} = \delta(x_i, y_j) + \min^{\gamma}\{ r_{i-1,j-1},\ r_{i-1,j},\ r_{i,j-1} \},$$

and the soft-DTW value is $\mathrm{dtw}_\gamma(x,y) = r_{n,m}$.

The forward dynamic algorithm is detailed in Algorithm 1.

3.3 Gradient Computation

Lemma 1 (Gradient of Soft-DTW)**.** *Let $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$ be two time series, and let $\delta(x_i, y_j) = \|x_i - y_j\|^2$ be the squared Euclidean cost. Let $E = (e_{ij}) \in \mathbb{R}^{n\times m}$ be the gradient matrix with entries $e_{ij} = \frac{\partial\, \mathrm{dtw}_\gamma(x,y)}{\partial \delta_{ij}}$. Then the gradient of soft-DTW with respect to x is:*

$$\nabla_x \,\mathrm{dtw}_\gamma(x,y) = 2\left[(E\mathbf{1}_m)\circ x - Ey\right],$$

where $\mathbf{1}_m \in \mathbb{R}^m$ is a vector of ones, $\circ$ denotes element-wise multiplication, and Ey denotes the matrix-vector product $(Ey)_i = \sum_{j=1}^m e_{ij} y_j$.

Proof.

$$\begin{aligned}
\mathrm{dtw}_\gamma(x,y) &= f(\delta_{11},\dots,\delta_{nm}),\\
\frac{\partial\, \mathrm{dtw}_\gamma}{\partial x_k} &= \sum_i \sum_j \frac{\partial\, \mathrm{dtw}_\gamma}{\partial \delta_{ij}} \frac{\partial \delta_{ij}}{\partial x_k}\\
&= \sum_i \sum_j e_{ij} \frac{\partial \delta_{ij}}{\partial x_k},\\
e_{ij} &:= \frac{\partial\, \mathrm{dtw}_\gamma}{\partial \delta_{ij}},\\
\nabla_x \,\mathrm{dtw}_\gamma(x,y) &= \left(\frac{\partial\, \mathrm{dtw}_\gamma}{\partial x_1},\dots,\frac{\partial\, \mathrm{dtw}_\gamma}{\partial x_n}\right),\\
\text{and for } \delta_{ij} &= \|x_i - y_j\|^2,\\
\frac{\partial\, \mathrm{dtw}_\gamma}{\partial x_k} &= \sum_{j=1}^m e_{kj}\cdot 2(x_k - y_j)\\
&= 2\big[x_k (E\mathbf{1}_m)_k - (Ey)_k\big],\\
\nabla_x \,\mathrm{dtw}_\gamma(x,y) &= 2\left[(E\mathbf{1}_m)\circ x - Ey\right],\\
E &= (e_{ij})_{1\le i\le n,\ 1\le j\le m}.
\end{aligned}$$

So now to get the gradient we only have to compute E: Let's first remark that

$$e_{i,j} = \frac{\partial r_{n,m}}{\partial \delta_{i,j}} = \frac{\partial r_{n,m}}{\partial r_{i,j}} \cdot \frac{\partial r_{i,j}}{\partial \delta_{i,j}} = \frac{\partial r_{n,m}}{\partial r_{i,j}} \cdot 1 = \frac{\partial r_{n,m}}{\partial r_{i,j}}$$

So computing E is equivalent to computing $\frac{\partial r_{n,m}}{\partial r_{i,j}}$, so let's find a recursive relation to compute $\frac{\partial r_{n,m}}{\partial r_{i,j}}$ using the chain rule and supposing $\gamma > 0$:

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

We suppose $\gamma > 0$ by applying the dynamical Programming (as shown in line 4 of Algorithm$_1$) :

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

as $\gamma > 0$ we can differentiate w.r.t $r_{i,j}$, hence we get:

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

We apply the logarithm and get:

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

Similarly, the following relationships can also be obtained:

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

so we get

$$e_{i,j} = e_{i+1,j} e^{\frac{r_{i+1,j} - \delta_{i+1,j} - r_{i,j}\cdot}{\gamma}} + e_{i,j+1} e^{\frac{r_{i,j+1} - r_{i,j} - \delta_{i,j+1}}{\gamma}} + e_{i+1,j+1} e^{\frac{r_{i+1,j+1} - r_{i,j} - \delta_{i+1,j+1}}{\gamma}}$$

Since $r_{i+1,j+1}$, $r_{i,j}$, $r_{i,j+1}$, $r_{i+1,j}$, $\delta_{i+1,j+1}$, $\delta_{i,j+1}$, $\delta_{i+1,j}$ are all known we get our backward recursion since $e_{n,m} = 1$

Thus we obtain the second algorithm from the paper (indices i and j run backward from n to 1 and m to 1). See Algorithm 2.

Since $\nabla_x \text{dtw}_\gamma(x, y) = 2\left[(E\mathbf{1}_m) \circ x - Ey\right]$ We compute E recursively using algorithm 2 and we get the gradient.

4 Experimental protocols

In this section, we reproduce several experiments originally introduced in [1]. Our objective is to describe the experimental protocols used to assess the behavior of soft-DTW in practical time series analysis scenarios.

4.1 Averaging with Soft-DTW

The goal of this first experiment is to examine how the Soft-DTW barycenter behaves as the smoothing parameter γ varies. We reproduce the qualitative results of [1] and study the transition from a highly smoothed regime to a nearly non-smoothed one. This experiment highlights how γ influences both the optimization landscape and the stability of the resulting average.

Given a set of N univariate time series $y_1, \ldots, y_N$, the soft-DTW barycenter is defined as the solution of

$$\min_{x \in \mathbb{R}^n} \sum_{i=1}^{N} \frac{\lambda_i}{m_i} \text{dtw}_\gamma(x, y_i), \tag{1}$$

where:

- $x \in \mathbb{R}^n$ is the barycenter we want to estimate (fixed length n);
- y_i is the i-th time series (length m_i);
- dtw_γ is Soft-DTW with smoothing parameter γ;
- $\lambda_i \geq 0$ are weights with $\sum_{i=1}^{N} \lambda_i = 1$;
- the division by m_i follows the normalization of [1].

Non-convexity. We remark that this problem is in general non-convex as it is shown on these curves. And so we do not have any theoretical guarantees of the existence and the convergence toward the minimum of our gradient descent algorithm.

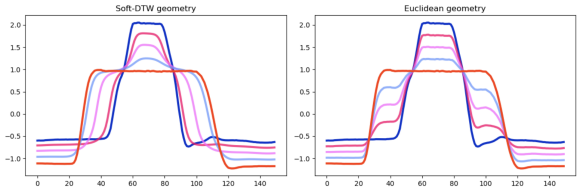

Figure 1: Non-convexity of the barycenter function due to the non-convexity of the soft-dtw

Smoothness. We also remark that $\min^\gamma\{a_1, \ldots, a_n\}\big|_{\gamma=\infty} = -\gamma - a_i/\gamma = a_i = \delta_{r,l}$ hence we get that when γ is big enough ($\gamma \gg \|a_i\|_\infty$) if δ is convex we get that dtw_γ convex so we can get guarantee of existence and that our gradient descent algorithm converge toward the minimum.

The minimization is performed using the L-BFGS algorithm. For $\gamma = 0$, this reduces to the standard non-smooth, non-convex DTW objective. For $\gamma > 0$, the loss becomes differentiable, and larger γ values yield smoother optimization landscapes.

Protocol. On the dataset `ECG200` (UCR), we sample 10 series from a randomly selected class and initialize the barycenter with their Euclidean mean. We then compute Soft-DTW barycenters for $\gamma = 1$ and $\gamma = 0.01$.

4.2 Clustering with Soft-DTW

We now investigate the effect of the smoothing parameter γ in the context of k-means clustering under the Soft-DTW geometry. As in the [1], the goal is to evaluate whether smoothed barycenters lead to more stable and representative cluster prototypes than those obtained with standard DTW or related baselines.

Given a set of time series $y_1, \ldots, y_N$ and a fixed number of clusters k, the Soft-DTW k-means objec-

tive reads:

$$\min_{x_1,\dots,x_k} \sum_{i=1}^{N} \frac{1}{m_i} \min_{j\in\{1,\dots,k\}} \mathrm{dtw}_\gamma(x_j, y_i),$$

where each x_j is a barycenter of fixed length and the inner minimization corresponds to the assignment step.

Following Lloyd's algorithm [2], we alternate between two phases: (i) assigning each time series to its closest prototype under dtw_γ, and (ii) updating each prototype by computing the Soft-DTW barycenter of its assigned cluster.

Protocol. We follow the setup used in the averaging experiment. We use the `CBF` dataset, select one class, and initialize $k = 3$ clusters, corresponding to the three underlying shapes (Cylinder, Bell, Funnel). Cluster centroids are initialized by randomly selecting three time series, and the Soft-DTW k-means algorithm is run for a fixed number of iterations.

As in Lloyd's algorithm [2], we alternate between:

- **Assignment step**: each time series is assigned to the closest centroid under dtw_γ.
- **Update step**: each centroid is recomputed as the Soft-DTW barycenter of its assigned cluster.

We run the algorithm using $\gamma = 1$.

4.3 Learning prototypes for time-series classification

To avoid the storage and computational cost of k-NN for time series, the [1] use a prototype-based classifier in the spirit of [3] and [4]. Each class is represented by a single prototype obtained as a Soft-DTW barycenter of its training examples, and prediction is performed by assigning a test series to the nearest prototype under Soft-DTW.

Protocol. We evaluate this nearest-centroid classifier on the 79 datasets of the UCR time-series archive. For several smoothing parameters γ, class barycenters are computed on the training set, γ is selected using validation accuracy, and final classification accuracy is reported on the test set.

Practical limitations. Because computing Soft-DTW barycenters is computationally expensive, we did not reproduce this experiment.

4.4 Multistep-Ahead Prediction

The final experiment evaluates Soft-DTW as a loss function for forecasting the future segment of a time series from its initial observations. The task can be formulated as learning a neural network f_θ that maps the first part of a sequence to its remaining part. More precisely, for each series x_i, we denote by $x_i^{1,t}$ the first t observations and by $x_i^{t+1,n}$ the remaining ones. Training amounts to solving:

$$\min_\theta \sum_{i=1}^{N} dtw_\gamma\left(f_\theta(x_i^{1,t}),\, x_i^{t+1,n}\right),$$

where γ controls the smoothness of the Soft-DTW loss.

Protocol. Following the same protocol as in the [1], each time series is split using the first 60% as input and the remaining 40% as output, ignoring class labels. We train a simple one-hidden-layer MLP with sigmoid activation to predict future values from past ones. Unlike the [1], which evaluates this experiment over the entire UCR archive, we restrict our analysis to the `ECG200` dataset for computational reasons. Models are trained under different losses (Euclidean and Soft-DTW with $\gamma \in \{1, 0.1, 0.01, 0.001\}$), and evaluation on the test set is performed using the DTW metric.

Difference with [1]. The original paper reports DTW and Euclidean test losses. Given the high computational cost of Soft-DTW (particularly for small γ), we only reproduce the DTW evaluation, which already required significant training time on `ECG200`.

5 Numerical results

In this section, we implement the experimental framework introduced above. The corresponding results will be presented and discussed in the next section.

5.1 Averaging with Soft-DTW

We implemented this task and the full code for this experiment is provided in the Python notebook `barycenter.ipynb`. The results are below :

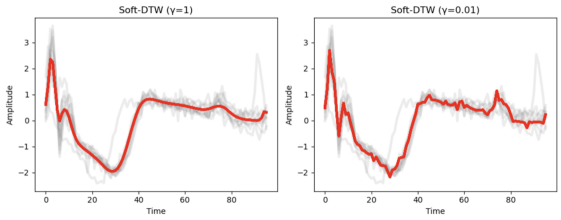

Figure 2: Soft-DTW barycenters for $\gamma = 1$ (left) and $\gamma = 0.01$ (right). Grey curves: input `ECG200` time series. Red curve: estimated barycenter.

The horizontal axis corresponds to time and the vertical axis to the signal amplitude.

Case $\gamma = 1$: The left panel of Figure 2 shows that a large smoothing parameter produces a very smooth barycenter. It captures the global ECG shape while filtering out local fluctuations, reflecting the stabilizing effect of Soft-DTW when γ is large.

Case $\gamma = 0.01$: The right panel shows that with a very small smoothing parameter, the barycenter still recovers the main ECG shape but becomes less smooth and inherits local oscillations. This is expected: when γ approaches zero, the objective behaves like standard DTW, which is more irregular and sensitive to small variations.

Discussion. Overall, our results reproduce the behaviour reported in the original Soft-DTW paper: a large γ yields a smooth and stable barycenter, while a very small γ introduces local oscillations. This confirms that the smoothing parameter is crucial for obtaining meaningful barycenters under the DTW geometry.

5.2 Clustering with Soft-DTW

We implemented this clustering task and the full code for this experiment is provided in the Python notebook `kmeans.ipynb`. The results are shown below.

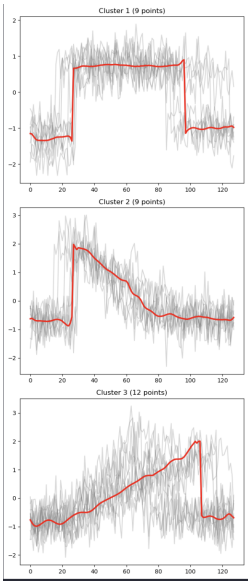

Figure 3: Soft-DTW k-means clustering on the `CBF` dataset ($K = 3$). Grey curves are cluster members and the red curve is the Soft-DTW barycenter.

Figure 3 displays the three clusters obtained with Soft-DTW k-means on the `CBF` dataset. On each graph, the horizontal axis denotes time and the vertical axis amplitude.

Cluster 1: Cylinder. This cluster contains 9 time series. Its centroid exhibits the characteristic plateau of the Cylinder class: a sharp rise, a flat segment, and a symmetric drop. Soft-DTW effectively smooths the noise and isolates the Cylinder patterns.

Cluster 2: Bell. This cluster (9 series) has a centroid with the typical Bell shape: a rapid rise followed by a smooth, monotonic decline. Soft-DTW effectively filters the noise and recovers a stable representative prototype.

Cluster 3 : Funnel shape. The third cluster contains 12 time series and corresponds to the *Funnel* class. The barycenter shows a progressive increase throughout the sequence, followed by a sharp drop near the end. This centroid accurately reflects the asymmetric structure of the Funnel pattern, while filtering out the amplitude variability and noise present within the cluster.

Discussion. Overall, the three clusters recovered by Soft-DTW k-means correspond precisely to the expected Cylinder, Bell, and Funnel shapes of the `CBF` dataset. The centroids are smooth, interpretable, and robust to noise. These results match exactly those reported in Section 4.2 of the original Soft-DTW paper [1], confirming that the smoothing parameter leads to stable, meaningful and highly representative cluster prototypes.

5.3 Multistep-Ahead Prediction

We implemented this multi–step ahead prediction task and the complete code for the experiment is available in the Python notebook `multi-ahead-prediction.ipynb`. The corresponding results are reported below.

Loss	**Rand. init**	**Euc. init**
Euclidean	6.3	7.1
Soft-DTW ($\gamma = 1$)	5.9	6.02
Soft-DTW ($\gamma = 0.1$)	5.4	5.8
Soft-DTW ($\gamma = 0.01$)	4.8	5.6
Soft-DTW ($\gamma = 0.001$)	5.7	6.1

Table 1: DTW test error for multistep prediction on ECG200.

Discussion. Contrary to [1], our results do not exhibit a monotonically improving trend as γ decreases: while $\gamma = 1$ improves over the Euclidean loss, very small values such as $\gamma = 0.001$ lead to a *higher* DTW test error. This discrepancy is expected given the radically different experimental conditions. In our setting, the model is trained and evaluated on a *single* UCR dataset (ECG200), which is small, noisy, and short in length. Training on a single dataset induces high variance, making the effect of γ much less stable than in [1], where results are averaged over many datasets with diverse temporal structures.

Another key factor is optimization difficulty. For small values of γ, the Soft-DTW loss becomes sharply peaked and the gradient signal degrades, making optimization significantly harder. In practice, convergence often required more than 3 000 epochs, and even then the model may not reach the same energy level as for larger γ. This explains why excessively small values (e.g., $\gamma = 0.001$) lead to worse performance in our experiments.

Finally, because each Soft-DTW forward and backward pass relies on a quadratic dynamic program, training is computationally expensive. Running the full protocol of [1] (multiple initializations, multiple γ values, and evaluations under DTW and Euclidean losses across the entire UCR archive) would have been prohibitive. Our objective was therefore not to reproduce the exact numerical values, but to study how Soft-DTW behaves on ECG200 and to observe how optimization becomes increasingly unstable as γ decreases.

6 Discussion of the paper

Although Soft-DTW provides a differentiable relaxation of DTW and enables gradient-based learning, it also introduces several limitations.

Choice of the smoothing parameter. Soft-DTW is highly sensitive to γ: large values oversmooth the loss, while small values make optimization unstable. Selecting γ is thus critical and usually requires costly validation.

Computational complexity. Soft-DTW inherits the quadratic time and memory costs of DTW, since both the forward and backward passes rely on dynamic programming. This makes neural network training significantly slower than with standard pointwise losses, especially on long sequences.

A classical way to reduce this cost is to constrain the alignment to a diagonal band, as in the Sakoe–

Chiba window [6]. Applying such restricted alignment regions to Soft-DTW could substantially improve its practical scalability.

Scalability of barycenter computation. Computing soft-DTW barycenters requires evaluating the dynamic program for each time series at every iteration, yielding a per-iteration cost of $O(NT^2)$ [1]. This rapidly becomes prohibitive on large datasets or when several values of γ must be tested. This difficulty is consistent with observations in the DTW literature: even for the classical DTW barycenter, naive computation is widely acknowledged to be too expensive to scale, which has motivated the development of approximate multi-scale methods such as FastDTW [5].

References

[1] M. Cuturi and M. Blondel (2017). *Soft-DTW: a Differentiable Loss Function for Time-Series*. ICML 2017.

[2] S. Lloyd (1982). *Least squares quantization in PCM*. IEEE Transactions on Information Theory.

[3] T. Hastie, R. Tibshirani, J. Friedman (2001). *The Elements of Statistical Learning*. Springer.

[4] R. Tibshirani, T. Hastie, B. Narasimhan, G. Chu (2002). *Diagnosis of multiple cancer types by shrunken centroids*. PNAS.

[5] S. Salvador and P. Chan (2007). *Toward Accurate Dynamic Time Warping in Linear Time and Space*. SIAM International Conference on Data Mining (SDM).

[6] H. Sakoe and S. Chiba (1978). *Dynamic Programming Algorithm Optimization for Spoken Word Recognition*. IEEE Transactions on Acoustics, Speech, and Signal Processing, 26(1): 43–49.

A Algorithmic Details for Soft-DTW

This appendix contains the forward and backward dynamic programming recursions used to compute $\mathrm{dtw}_\gamma(x, y)$ and its gradient with respect to x.

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

Inputs: x, y, smoothing $\gamma \geq 0$, distance δ
$r_{0,0} = 0; \quad r_{i,0} = r_{0,j} = +\infty \quad$ for $i \in [\![n]\!]$, $j \in [\![m]\!]$
for $j = 1, \ldots, m$ **do**
  for $i = 1, \ldots, n$ **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: $(r_{n,m}, R)$

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

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