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

Mini-Project (ML for Time Series)

Clémence Grislain clemence.grislain@eleves.enpc.fr
Roman Plaud roman.plaud@eleves.enpc.fr

MVA 2022/2023

1 Introduction and contributions

Dynamic Time Warping (DTW) measures similarity between time series data by temporally aligning two series to minimize the Euclidean distance. DTW can compare time series of different lengths and is robust to timeline changes such as shifting, dilation, contraction, etc. It's widely used for classification, clustering, and pattern recognition. However, one may be interested in having a loss function that can be differentiated, especially for neural network pipelines.

The paper [CB18] introduces Soft-DTW, a smoothed and differentiable version of DTW. Soft-DTW relaxes the minimization problem of DTW by computing a soft minimum over all possible alignments. Soft-DTW has major advantages: Soft-DTW can be used in all optimization problems requiring differential loss to approximate DTW.
In this project, we studied the properties of the Soft-DTW metric both theoretically and in practical applications for time series classification and prediction tasks. We explored its benefits compared to methods based directly on DTW or Euclidean distance.
We work with the GitHub repository attached to the original paper. This implementation allowed us to avoid recoding all the basic functions, such as the actual Soft-DTW function (value and gradient computation) and the Soft-DTW barycenter based on it. However, this repository did not include any examples or code to reproduce the results obtained in the paper. Therefore, the attached notebook is almost entirely our own production.
Our work focuses on reproducing the performances of the paper on specific datasets *50words* for classification and *ECG5000* for time series prediction, both are extracted from UCR database [DBK+19]. We also conduct experiments on the parameter γ of Soft-DTW and attempted to understand the behavior of classification and prediction results when varying parameter with respect to the results obtained using Euclidean distance and DTW.
In terms of work distribution, both of us worked on both subjects, but Clémence focused more on the classification task, while Roman focused more on the prediction task. Overall, the work was equally shared.

2 Method

To understand the theoretical construction of Soft-DTW, we consider a framework in which we are given two time series $\mathbf{x} = (x_1, \dots, x_n) \in \mathbb{R}^{n \times p}$ and $\mathbf{y} = (y_1, \dots, y_m) \in \mathbb{R}^{m \times p}$, where $(n, m, p) \in \mathbb{N}^3$.

2.1 DTW

DTW computes a correspondance between elements of $\mathbf{x}$ and of $\mathbf{y}$ which is called a path. A path can be mathematically described as follows :

$$P = ((i_1, j_1), \dots, (i_{K_p}, j_{K_p})) \in (\mathbb{N} \times \mathbb{N})^{K_p}, K_p \in \mathbb{N}$$

$$(i_k, j_k) \in P \Leftrightarrow x[i_k] \text{ is matched with } y[j_k]$$

This path must be **continuous** (it can only do step of size 1), **monotonic** (it can only go up and right, and diagonally up right) and **bounded** (it starts/ends by matching together the first/last elements of both time series). A path can therefore be represented uniquely as an alignment matrix $A_P \in \{0,1\}^{n \times m}$. The 3 conditions mentioned above can be rewritten as regarding A_P as : A_P represents a path on a n × m matrix that connect the upper-left $(1,1)$ matrix entry to the lower-right (n,m) one using only moves of size 1 going right, down or diagonal right (no up or left move). Then, let us consider a cost function δ to evaluate path P and $\Delta(\mathbf{x}, \mathbf{y}) = (\delta(x_i, y_j))_{i,j} \in \mathbb{R}^{n \times m}$, we compute

$$w(P) = \sum_{k=1}^{K_p} \delta(x[i_k], y[j_k]) = \langle A_P, \Delta(\mathbf{x}, \mathbf{y}) \rangle = w(A_P)$$

And then we can define DTW as :

$$DTW(\mathbf{x}, \mathbf{y}) = \min_{P \in \mathcal{P}} w(P) = \min_{A \in \mathcal{A}} w(A)$$

DTW can be computed easily taking advantages of Dynamic programming tools [BC94]. However, as briefly mentioned in introduction, gradient of DTW, when it exists, will be discontinuous around the values $\mathbf{x}$ where a small change in $\mathbf{x}$ causes a change in the path, which is likely to drastically reduce the performance of gradient descent method. This last point motivates the introduction of a smoothed DTW, which would be differentiable everywhere and whose gradient would be continuous.

2.2 Soft-DTW

Let us consider the generalized minimum operator, with a smoothing parameter $\gamma \geq 0$:

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

Therefore, we can define the Soft-DTW :

$$SDTW^{\gamma}(\mathbf{x}, \mathbf{y}) = \min_{A \in \mathcal{A}}^{\gamma} \langle A, \Delta(\mathbf{x}, \mathbf{y}) \rangle$$

When $\gamma > 0$ we can derive the gradient w.r.t. $\mathbf{x}$ as follows :

$$\nabla_x SDTW^{\gamma}(\mathbf{x}, \mathbf{y}) = \nabla_x \left(-\gamma \log \sum_{A \in \mathcal{A}} \exp\left(\frac{-\langle A, \Delta(x,y) \rangle}{\gamma} \right) \right) = \left(\frac{\partial \Delta(x,y)}{\partial x} \right)^{\mathbf{T}} \frac{\sum_{A \in \mathcal{A}} A \exp\left(\frac{-\langle A, \Delta(x,y) \rangle}{\gamma} \right)}{\sum_{A \in \mathcal{A}} \exp\left(\frac{-\langle A, \Delta(x,y) \rangle}{\gamma} \right)}$$

This approach yields a closed form expression of the gradient, but its computation cost is quartic $O(n^2 m^2)$. The main achievement of the paper is that it provides a nm-complexity algorithm which performs a forward Bellmann pass which both compute the $SDTW^{\gamma}$ and stores all intermediate costs. A backward pass is then performed which results in the actual computation of the gradient.

3 Data

To perform our experiments, we use two datasets from UCR Archive [DBK+19].

- **Classification**: We work with '50words' dataset to test the different classification methods. This dataset comprises height profiles of words extracted from the George Washington library, as documented by T. Rath. It includes a total of 905 data, consisting of 450 training samples and 455 testing samples. We selected this dataset as we already manipulated it in the first practical session of the course. Moreover, this dataset poses a challenging classification problem due to its large number of classes and imbalanced distribution of training data. Due to the inherent nature of words, data collection does not imply any errors such as noise or problems coming from a device. To that extent, very few preprocessing is needed. We only rescale words so that each one of them is normalized - zero-mean, and standard deviation of one. The time series also have the same length.
- **Prediction**: We work with 'ECG5000' dataset to test the multi-step ahead prediction models. This dataset contains information about a person's heartbeats, thus one dimensional time series. The data are taken from a 20-hour long electrocardiogram (ECG) called the BIDMC Congestive Heart Failure Database (chfdb) in which researchers randomly selected 5,000 heartbeats -4500 for test set and 500 for training set. ECG data mainly suffer from various sources of noise, which can originate from the patient's breathing or movement, or from the sensor itself. These phenomena can lead to change point variation -especially of the mean- or outliers. Yet, the downloaded dataset was already pre-processed in two ways:
  - extract each heartbeat
  - make each heartbeat equal length using interpolation

  As we are not working on full ECG signal but segmented heartbeats, the change point issue is no longer present. We chose this dataset because of the small length of its time series (140), which allows for fast training and inference of the models and thus implies faster experiments on the model's parameters.

Remark The fact that both datasets have time series of equal length allowed us to compare methods based on DTW and $SDTW^{\gamma}$ with methods based on Euclidean distance, which require equal lengths of training data.

4 Results

4.1 Centroids and classification

To perform time series classification, a very efficient approach is to implement a k-Nearest Neighbors algorithm using DTW loss. However, at prediction time, the computational cost of K-NN grows with the size of the dataset ($|train| \times |test|$). In a resource-constrained situation, a solution would be to compute for each class a representative from the training set. At prediction time, we can perform a nearest centroid classifier -a 1-NN only with the centroids- which has a time complexity ($|centroids| \times |test|$).
The computation of representatives and the assignment rule at prediction time are critical in nearest centroid classification. In this work, we define representatives as barycenters with respect to a given metric, and use the same metric during prediction. We explore nearest centroid classification using three different metrics: Mean Squared Error (MSE), DTW, and $SDTW^{\gamma}$.

4.1.1 Centroid computation

To compute the centroids, we use pre-implemented algorithms. Specifically, we used the DTW barycenter computation algorithm DBA [PKG11] provided by the `tslearn` library , and the $SDTW^\gamma$ barycenter algorithm provided by the source code attached with the paper [CB18]. (The MSE barycenters are just term by term averaging).

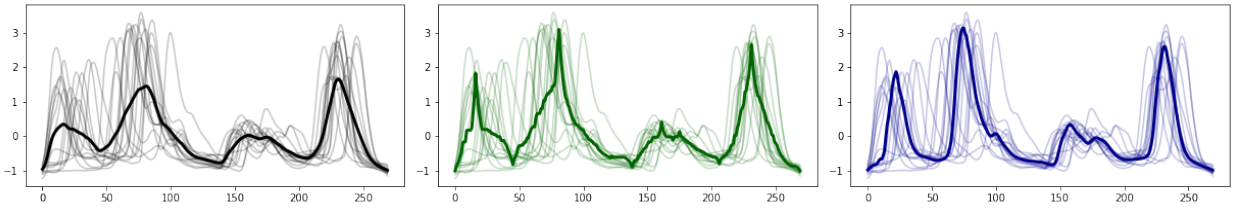

Figure 1: Centroids of a category from *50words* dataset (label 6), using MSE (left), DTW (center), $SDTW^\gamma$ ($\gamma = 1$) (right)

Figure 1 displays the centroids for the label 6 of the *50words* dataset, which contains 15 training data, as MSE, DTW and $SDTW^\gamma$ barycenters. We observe that the centroid computed with the MSE method is oversmoothed. At the contrary, the one computed with the DBA algorithm is overshaped, but it captures the characteristics of the class in a better way. The $SDTW^\gamma$ centroid appears to offer a good balance between the two previous methods, preserving the singularities of the class while maintaining smoothness.

4.1.2 Nearest Centroid Classification

Table 1 summarises the accuracies on the test set of the K-NN and of the different centroids classifications fitted on the entire training set, using the best parameters of the cross validation.

Remark: we used cross validation on a reduced training set (339 data) and validation set (111 data) to determine the optimal parameters for K-NN and $SDTW^\gamma$, namely $k = 1$ and $\gamma \approx 0.37$.

Algorithm	KNN	Centroids MSE	Centroids DTW	Centroids $SDTW^\gamma$
Accuracy	0.7209	0.5165	0.6264	**0.7275**

Table 1: Accuracy on the test set of *50words* (k=1 for K-NN $\gamma = 0.37$ for centroids with $SDTW^\gamma$)

In accordance with the results of the paper, we observed that using $SDTW^\gamma$ improves the performance of the nearest centroid classifier by approximately 11% for the *50words* dataset. Furthermore, nearest centroid classifier based on $SDTW^\gamma$ achieves similar results as the K-Nearest Neighbour algorithm, while being less memory-intensive. This is due to the regularization effect of the smoothing in$SDTW^\gamma$.

4.2 Multi-step ahead prediction

As explained in the introduction, because of its differentiability $SDTW^\gamma$ is really suited for tasks that require a time series output. As in the paper, we focus here on multi-step ahead prediction, that is knowing the beginning of a time series, we want to predict the following part of it. We work in a framework in which we know the first 60% of it and want to predict the 40% masked part. Following [CB18] we implement a simple two-layers perceptron, with a sigmoid activation

function, whose input and output dimension are fixed by the length of the time series, and the hidden dimension is fixed to 256. We train our network on the dataset *ECG5000* and experiment different hyperparameters, especially the choice of the loss function. Rather than detailing too much about specificities of the neural network, we focus here on the results.

4.2.1 Trade-off at stake : qualitative results

In the context of time series prediction, it is desirable to achieve both a low DTW score and a low MSE. However, as illustrated in Figure 2, there are two unwanted scenarios. The first plot shows a prediction made by a neural network trained with $SDTW^{\gamma}$ and a small γ value (0.001) (we recall that the smaller γ is the more it approximates DTW) , which results in a prediction with a very low DTW score but with a time offset that may need to be avoided. Conversely, the second plot shows a prediction made by a neural network trained with MSE Loss, resulting in a prediction with a low MSE but with oversmoothing that erases the sharpness of the output, another drawback that needs to be avoided. In the next section, we demonstrate that $SDTW^{\gamma}$ with an appropriate γ value strikes a balance between DTW and MSE.

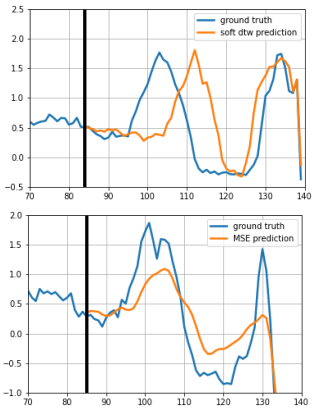

Figure 2: Time series prediction using $SDTW^{\gamma}$ (top) and MSE (bottom)

4.2.2 Quantitative results

For gamma ranging from 0.001 to 1 we train 10 models trained on 10 different random train/validation splits. For each γ we evaluate our 10 models on the test set using two different metrics : DTW and MSE, we then average them and compute 95% confidence interval, we observe in Figure 3 :

- On a wide range of γ, $SDTW^{\gamma}$ outperforms MSE-trained network on DTW evaluation
- On MSE evaluation, the more γ increase the more $SDTW^{\gamma}$ closes up to MSE-trained network
- When $\gamma \to 0$ we observe instable behaviour and high DTW values.

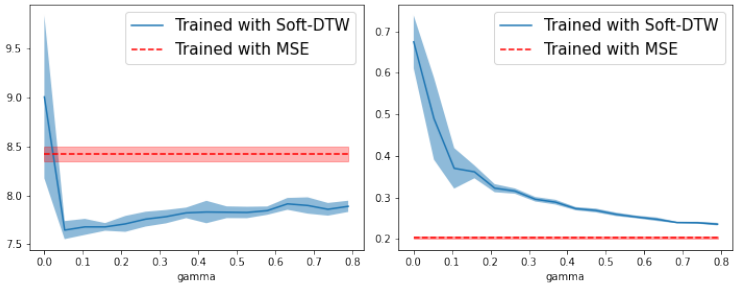

Figure 3: DTW (left), MSE (right) on test set for variouts training split

These statements allow us to say that $SDTW^{\gamma}$ is a very good proxy to DTW as well as MSE. It possesses the invariance to timeline changes characteristic of DTW while allowing for smoothness regularization. Therefore, it yields results that performs a tradeoff between the two measures and effectively avoids the both drawbacks expressed in previous parts, that is to say, time offset for DTW and oversmoothing for MSE.

References

[BC94] Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370, July 1994.

[CB18] Marco Cuturi and Mathieu Blondel. Soft-dtw: a differentiable loss function for time-series, 2018.

[DBK+19] Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. The ucr time series archive, 2019.

[PKG11] Francois Petitjean, Alexandre Ketterlin, and Pierre Gancarski. A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recognition*, 44(3):678–693, 2011.