

Learning DTW-Preserving Shapelets

Arnaud Lods, Simon Malinowski, Romain Tavenard, Laurent Amsaleg

▶ **To cite this version:**

Arnaud Lods, Simon Malinowski, Romain Tavenard, Laurent Amsaleg. Learning DTW-Preserving Shapelets. International Symposium on Intelligent Data Analysis, Oct 2017, London, United Kingdom. 10.1007/978-3-319-68765-0_17 . hal-01565207v1

HAL Id: hal-01565207

https://hal.science/hal-01565207v1

Submitted on 19 Jul 2017 (v1), last revised 24 Oct 2017 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Learning DTW-Preserving Shapelets

Arnaud Lods[1], Simon Malinowski[2], Romain Tavenard[3], and Laurent Amsaleg[4]

[1] IRISA, Rennes
[2] Univ. Rennes 1, IRISA
[3] Univ. Rennes 2, CNRS, UMR LETG, IRISA
[4] CNRS-IRISA

Abstract. Dynamic Time Warping (DTW) is one of the best similarity measures for time series, and it has extensively been used in retrieval, classification or mining applications. It is a costly measure, and applying it to numerous and/or very long times series is difficult in practice. Recently, Shapelet Transform (ST) proved to enable accurate supervised classification of time series. ST learns small subsequences that well discriminate classes, and transforms the time series into vectors lying in a metric space. In this paper, we adopt the ST framework in a novel way: we focus on learning, without class label information, shapelets such that Euclidean distances in the ST-space approximate well the true DTW. Our approach leads to an ubiquitous representation of time series in a metric space, where any machine learning method (supervised or unsupervised) and indexing system can operate efficiently.

1 Introduction

Time series analysis and mining is a wide research domain becoming increasingly popular over the last decades for tasks as diverse as classification, clustering, indexing or retrieval (see [4] for a survey). One popular similarity measure to compare time series is the Dynamic Time Warping (DTW), due to its capacity to cope with time shifts and warpings.Its complexity being quadratic with the length of time series, it is difficult to use DTW against very long time series and/or very large sets of time series. In turn, many research works have attempted to reduce that complexity and/or have tried to run DTW onto a very limited subset of candidate sequences [7, 9, 11, 12]. Note furthermore that DTW is not a distance as the triangular inequality does not hold, which induces sub-optimality when used with traditional optimizations for indexing, or with kernel-based classfiers for instance.

Recently, a new family of approaches, based on the concept of *shapelets* [5, 6, 13], has been proposed for time series classification. Shapelets are time series subsequences selected (or learned) so as to discriminate classes. Amongst these approaches, the Shapelet Transform (ST) [6] uses shapelets as surrogates for representing time series: each time series is projected against the set of shapelets, resulting in a vector in which components represent the distances between the time series and the shapelets.

This paper proposes a different approach at shapelet learning time: we do not try to best discriminate classes, but instead we aim at learning shapelets that best preserve the DTW. In other words, shapelets are selected such that the Euclidean distance between transformed time series best approximates the DTW between raw time series. The objective function we use to learn the shapelets is hence different from the ones of traditional works based on shapelets.

Learning shapelets to best approximate the true DTW between pairs of time series from a training set has several nice properties. First, it becomes possible to get a good estimation of the true DTW between any two time series by simply computing the Euclidean distance between the resulting shapelet transform vectors. Second, shapelet transformed vectors being a good proxy for the DTW, it becomes possible to use them, not only for supervised classification of time series, but also for many other tasks such as time series clustering, retrieval or indexing. This novel shapelet representation becomes quite ubiquitous as it can feed a wide range of machine learning or indexing methods for time series.

This paper first presents how to determine shapelets such that the DTW between all pairs of a training set is well captured in the high-dimensional space by the resulting vectors. We then demonstrate the validity of our approach by measuring how good surrogates are such vectors for the DTW. We also highlight the performance of this novel representation for time series clustering.

2 Related work

A very large number of contributions aim at reducing the cost of the DTW, either by relying on lower bounds or by applying sophisticated pruning strategies. All this helps, only to some extent [4, 11]. This paper, however, follows an entirely different direction as it builds on *vectorial representations* of time series. This related work section hence mainly focuses on such techniques and essentially discusses shapelet transforms for time series analysis.

Shapelets were introduced by Ye and Keogh in [13] for time series classification where a shapelet is an existing subsequence of a time series that best discriminate classes. Hills *et al.* proposed the *shapelet transform* in [6]. It consists in transforming a time series into a vector, its components representing the distances between the time series and shapelets determined beforehand. This vectorial representation of time series then feeds a classifier.Instead of using *existing* subsequences as shapelets, Grabocka *et al.* in [5] propose to rather forge the shapelets by learning the subsequences that minimize a classification loss. The learning step relies on a gradient descent.Then, the learnt shapelets are used to transform the time series into vectors, as proposed by Hills *et al.*.Unsupervised extraction of shapelets has also been proposed in the literature for clustering purposes. In [14], Zakaria *et al.* extract the shapelets so that they divide the set of time series into well separated groups. Zhang *et al.* [15] propose to combine the learning of shapelets with pseudo-class labels.

To the best of our knowledge, only a very recent work tackles the same objective as ours which is to learn a mapping such that the Euclidean distance

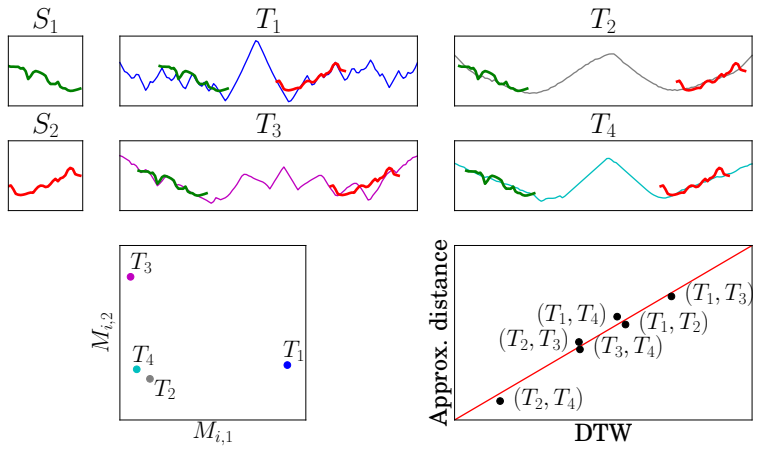


Fig. 1: Illustration of LDPS. Learned shapelets S_1 and S_2 are used to embed time series in a 2-dimensional space (*cf.* bottom-left figure), in which the Euclidean distance is a good approximation of the DTW between original time series. In real settings, more than 2 shapelets are learned to obtain higher-dimensional embeddings and hence richer representations.

between the transformed time series preserves at best the DTW between the original time series. Compared to the approach we describe in this paper, Lei *et al.* in [8] reach this goal with very different means, as no shapelets are involved in their work while they are at the core of the technique we describe here. Lei *et al.* learn a vectorial representation of time series such that the dot product between these representations well preserves the similarity between the original time series. The representations are obtained by matrix factorization. They also propose another learning strategy based on a gradient descent that is faster, but less accurate in preserving the original similarities. An extremely severe drawback of their contribution is that it learns the transformed time series, and not the transformation itself. It is therefore impossible to transform a new and unknown time series once a database of time series has been fully transformed. Their method can therefore not be used in the many applications where queries are not known in advance or where the database of time series has to be updated.

In contrast, with the approach detailed below, not only each time series in the database is transformed into a vector, but any unknown new time series that probes or that is to add to the dataset can undergo such a transform. It is the process of transforming the time series into a DTW-preserving high-dimensional vector that we overall learn.

3 Learning DTW-Preserving Shapelets (LDPS)

In this section, we detail LDPS an algorithm that embeds time series into a metric space such that the Euclidean distance in the transformed space approx-

imates the DTW in the original time series space. An illustration of the LDPS algorithm is schematically given in Figure 1.

3.1 Definitions and notations

We define here notions and quantities that are used in most of the papers that deal with time series shapelets [5, 6]. Let $\mathcal{T} = \{T_1, \ldots, T_N\}$ be a set of N time series. We assume here that all time series in $\mathcal{T}$ have the same length Q, but the method presented in this paper is also valid for time series of different lengths. A time series T_i in $\mathcal{T}$ is hence composed of Q elements: $T_i = T_{i,1}, \ldots, T_{i,Q}$. In the following, $T_{i,m:L}$ will denote the m^{th} segment of length L of T_i: $T_{i,m:L} = T_{i,m}, \ldots, T_{i,m+L-1}$.

Definition 1. *A shapelet S of length L is an ordered sequence of L values. In the following, $\mathcal{S}$ will denote a set of K such shapelets: $\mathcal{S} = \{S_1, \ldots, S_K\}$, where $S_k = S_{k,1:L}$ for all $k \in \{1, \ldots, K\}$.*

Definition 2. *The Euclidean score between S_k and $T_{i,j:L}$ is defined as*

$$D_{i,k,j} = \frac{1}{L} \sum_{l=1}^{L} (T_{i,j+l-1} - S_{k,l})^2. \tag{1}$$

The Euclidean shapelet match between S_k and T_i is defined as

$$M_{i,k} = \min_{j \in \{1,\ldots,Q-L+1\}} D_{i,k,j}\,. \tag{2}$$

The Shapelet Transform of time series has been defined in [6]. It consists in, given a set $\mathcal{S}$ of K shapelets, transforming T_i into a K-dimensional vector M_i whose components are $\{M_{i,k}\}_{1 \leq k \leq K}$ (Eqn. (2)).

3.2 Loss function to be minimized

Most works dealing with shapelets first select the best set of shapelets before doing the shapelet transformation. Best shapelets are selected to discriminate classes. In this paper, we adopt a different approach. We aim at learning a set of shapelets such that the Shapelet Transform preserves as well as possible the Dynamic Time Warping measure. In other words, we would like that the Euclidean distance in the transformed space approximates the DTW. We turn this problem into the minimization of a loss function, as explained in the following. Let $\mathcal{S}$ be a set of K shapelets. Let T_{i_1} and T_{i_2} be two time series in $\mathcal{T}$. The Shapelet Transform of T_{i_1} and T_{i_2} is denoted M_{i_1} and M_{i_2} respectively. The Dynamic Time Warping between T_{i_1} and T_{i_2} is denoted $DTW(T_{i_1}, T_{i_2})$. The loss $\mathcal{L}(T_{i_1}, T_{i_2})$ induced by the approximation of $DTW(T_{i_1}, T_{i_2})$ by the Euclidean distance between M_{i_1} and $M_{i_2}$is defined as:

$$\mathcal{L}(T_{i_1}, T_{i_2}) = \frac{1}{2} \left(DTW(T_{i_1}, T_{i_2}) - \beta \left\| M_{i_1} - M_{i_2} \right\|_2 \right)^2, \tag{3}$$

where β is a scale parameter that is learned by the proposed method, as explained below. The overall loss for a dataset $\mathcal{T}$ of N time series is given by:

$$\mathcal{L}(\mathcal{T}) = \frac{2}{N(N-1)} \sum_{i_1=1}^{N-1} \sum_{i_2=i_1+1}^{N} \mathcal{L}(T_{i_1}, T_{i_2}). \tag{4}$$

3.3 Stochastic gradient descent

The LDPS method aims at learning, for a training set $\mathcal{T}$ of time series, a set $\mathcal{S}$ of K shapelets and a scale parameter β that minimize the overall loss defined in Eqn. (4). For the sake of clarity, we assume here that the shapelets of $\mathcal{S}$ have the same length L, but the method can be straight-forwardly extended to shapelets of different lengths. We adopt the stochastic gradient descent framework to learn the $K \cdot L + 1$ coefficients that lead to minimize $\mathcal{L}(\mathcal{T})$. In this framework, the gradients of $\mathcal{L}(T_{i_1}, T_{i_2})$ with respect to these coefficients need to be computed.

If we denote $\hat{Y}_{i_1,i_2} = \|M_{i_1} - M_{i_2}\|_2$ and $\Delta_{i_1,i_2,k} = M_{i_1,k} - M_{i_2,k}$, we get:

$$\frac{\partial \mathcal{L}_{i_1,i_2}}{\partial \beta} = \hat{Y}_{i_1,i_2} \left(\beta \hat{Y}_{i_1,i_2} - DTW(T_{i_1}, T_{i_2}) \right) \tag{5}$$

$$\frac{\partial \mathcal{L}_{i_1,i_2}}{\partial S_{k,l}} = \frac{\partial \mathcal{L}_{i_1,i_2}}{\partial \hat{Y}_{i_1,i_2}} \frac{\partial \hat{Y}_{i_1,i_2}}{\partial \Delta_{i_1,i_2,k}} \left(\frac{\partial M_{i_1,k}}{\partial S_{k,l}} - \frac{\partial M_{i_2,k}}{\partial S_{k,l}} \right) \ \forall k, l. \tag{6}$$

Straight-forward derivations give:

$$\frac{\partial \mathcal{L}_{i_1,i_2}}{\partial \hat{Y}_{i_1,i_2}} = \beta \left(\beta \hat{Y}_{i_1,i_2} - DTW(T_{i_1}, T_{i_2}) \right) \tag{7}$$

$$\frac{\partial \hat{Y}_{i_1,i_2}}{\partial \Delta_{i_1,i_2,k}} = \frac{\Delta_{i_1,i_2,k}}{\|M_{i_1} - M_{i_2}\|_2} \qquad \forall M_{i_1} \neq M_{i_2} \tag{8}$$

$$\frac{\partial M_{i,k}}{\partial S_{k,l}} = \sum_j \frac{\partial M_{i,k}}{\partial D_{i,k,j}} \frac{\partial D_{i,k,j}}{\partial S_{k,l}}. \tag{9}$$

In practice, we extend the formula provided in Eqn. (8) in the case where $M_{i_1} = M_{i_2}$ by: $\frac{\partial \hat{Y}_{i_1,i_2}}{\partial \Delta_{i_1,i_2,k}} = 0$. We observe experimentally that this case is sufficiently rare not to impair the convergence process.

In our implementation, we do not use soft-minimum approximation as done in [5] for the computation of $M_{i,k}$. Indeed, we observe that authors of [5] tend to use an α parameter so large (in absolute value) that they almost end up with a hard minimum computation. We then consider the limit case when $\alpha \to -\infty$ of the soft-minimum formula to get back to a hard-minimum setup, which gives: $\frac{\partial M_{i,k}}{\partial D_{i,k,j}} = \delta_{j,j*}$,

where $j*$ is the argmin of Eqn. (2). Finally, for the computation of $\frac{\partial D_{i,k,j}}{\partial S_{k,l}}$, derivations from [5] can be used: $\frac{\partial D_{i,k,j}}{\partial S_{k,l}} = \frac{2}{L}(S_{k,l} - T_{i,j+l-1})$.

These gradients are used to update the coefficients at each iteration of the algorithm with a learning rate of α, like for any gradient descent algorithm.

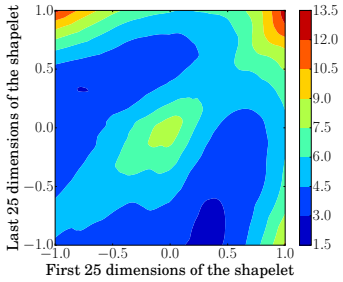


Fig. 2: Heat map of our loss function $\mathcal{L}$ with respect to shapelet coefficients for SwedishLeaf dataset. For visualization purposes, we consider a single shapelet with a single value assigned to the first 25 coefficients (x-axis) and another value for the remaining 25 coefficients (y-axis). Best viewed in color.

3.4 Model initialization

The loss function presented in Eqn. (4) is not convex, as illustrated in Figure 2. It is therefore of prime importance to ensure proper initialization of the model parameters for the optimization process not to get stuck in highly suboptimal local minima. In our setting, k-means clustering is used to generate the set of initial shapelets. Once the initial shapelets fixed, an initial value β_{init} is selected for β by randomly sampling a set $\mathcal{P}$ of 100 time series pairs and computing the corresponding optimal least square solution to the monodimensional regression problem that relates distances between Shapelet Transforms to DTW between original time series.We could, theoretically, update β the same way at each iteration, but this update rule has $O(|\mathcal{P}|)$ complexity, which contradicts our will to use stochastic gradient descent to ensure a fast update of the model.

3.5 Convolutional variant of LDPS

The Shapelet Transform on which we rely in this paper is very similar in spirit to what is learned by Convolutional Neural Networks. More precisely, the shapelet match presented in Definition 2 is very similar to a convolutional layer in a Neural Network. The computed Shapelet Transform corresponds to the output of a single-layer convolutional neural network with infinite max pooling in which the convolution operation would be replaced by a sliding window distance computation (and hence, the max pooling would be replaced by a min pooling). In this comparison, convolution filters are the equivalent of shapelets. We can then consider a unified framework in which both approaches can be used and compared experimentally. To do so, we introduce the convolutional shapelet match between a shapelet and a time series, which consists in using the following definition in place of Definition 2 (and its related Equations (1) and (2)):

Definition 3. *The convolutional score between S_k and $T_{i,j:L}$ is defined as*

$$D_{i,k,j} = \frac{1}{L} \langle S_k, T_{i,j:L} \rangle = \frac{1}{L} \sum_{l=1}^{L} S_{k,l} \cdot T_{i,j+l-1}. \tag{10}$$

The convolutional shapelet match between S_k and T_i is defined as

$$M_{i,k} = \min_{j \in \{1,\dots,Q-L+1\}} D_{i,k,j} \,. \tag{11}$$

As a consequence, the computation of $\frac{\partial D_{i,k,j}}{\partial S_{k,l}}$ for this convolutional variant of our model differs from the one presented above, and we get: $\frac{\partial D_{i,k,j}}{\partial S_{k,l}} = \frac{1}{L} T_{i,j+l-1}$.

In the following, we will refer to this convolutional variant of our model as LDPS-C, while the Euclidean one will be denoted LDPS-E.

Extending this analogy between neural networks and Shapelet models, our proposition can be seen as a *siamese* architecture [2] for Shapelets, *i.e.* two time series are provided as inputs to the same Shapelet model and distance between the corresponding outputs is used as a proxy for time series similarity. However, LDPS models are learned to minimize discrepancy between a target metric and the obtained distance whereas, in [2], the idea is to threshold the obtained distance for classification purposes.

3.6 Summary of the LDPS algorithm

A summary of the learning phase of the LDPS algorithm, i.e. the learning of $KL + 1$ coefficients (the set $\mathcal{S}$ of shapelets and the parameter β) is given in Algorithm 1. After this phase, the shapelet set $\mathcal{S}$ can be used to transform any time series into a vector of dimension K. The complexity of the shapelet transform (once the shapelets learned) is $\mathcal{O}(NLK)$, where N is the number of time series to transform.

Input : A set $\mathcal{T}$ of time series
The number K and length L of shapelets
The learning rate α for the gradient descent algorithm
Output: A set $\mathcal{S}$ of K shapelets of length L
The scale coefficient β

Initialize $\mathcal{S}$ and β according to Section 3.4
for $i \leftarrow 1$ **to** n_{iter} **do**
  Randomly pick two time series T_1 and T_2 from $\mathcal{T}$
  Compute the DTW between T_1 and T_2
  Compute the gradients of $\mathcal{S}$ and β from Eqs. (5) to (9)
  Update $\mathcal{S}$ and β (using their gradients and the learning rate α)
end

Algorithm 1: Learning phase of LDPS

4 Experimental results

In this section, we present experiments to evaluate the performance of our method. We first study the quality of DTW reconstruction reached by LDPS and then compare it to state-of-the-art competitors for a clustering task.

Experimental setup Following the principles of reproducible research, the Python code used in these experiments (including both variants of our model) is made publicly available for download[1].

Unless otherwise stated, each of our models makes use of shapelets of different lengths to better learn scale-specific patterns. Shapelet lengths L are set to 15%, 30% and 45% of time series lengths. Inspired by [5], we use a number K of shapelets for each length equal to $K = 10 \cdot \log(Q - L)$. Finally, as our method is stochastic, for each experiment, 5 different models are fitted. All models are fitted for 500,000 stochastic gradient descent steps, and we use the AdaGrad [3] algorithm to adapt the learning rate during the convergence process. Datasets used for the experiments are publicly available [1].

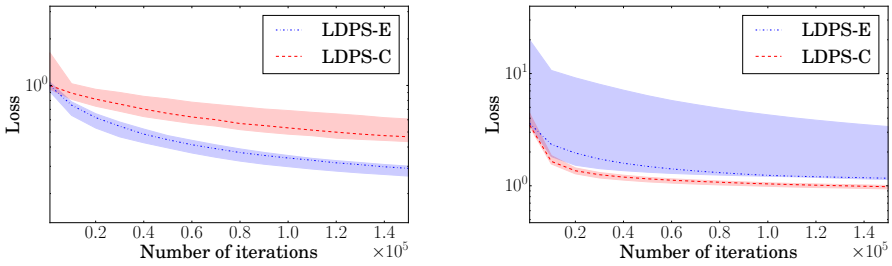
Comparison between LDPS-E and LDPS-C We analyze in this section the difference between LDPS-E and LDPS-C in terms of performance.

In practice, we observe that there does not seem to exist a consistently better variant on all datasets. Figure 3 presents model losses (*i.e.* mean squared DTW reconstruction error) as a function of the number of iterations. It shows that, for this criterion, LDPS-C outperform LDPS-E for Synthetic Control data set, while the opposite observation holds true when considering SwedishLeaf data set. Similar conclusions can be drawn when considering clustering performance as presented in Table 1.

Quality of DTW approximation Figure 4a presents the fit between DTW values and their approximations through the LDPS-E algorithm. Each dot in this figure corresponds to a pair of training time series. We can see that fully fitted model drastically improves the quality of DTW approximation over partially fitted ones. Another important point is to observe that all distance magnitudes are reproduced with similar accuracy, meaning that our method is able to reproduce both similarities and dissimilarities between time series. Finally, we observe in Figure 4b that there is no strong bias towards overestimation (resp. underestimation) observed for the fully fitted model, which indicates that the learned scale parameter β is reasonable.

Time series clustering with LDPS As LDPS embeds time series in a Euclidean space, it can be used for various machine learning tasks, including unsupervised ones, since class labels are not required to fit our models. We evaluate

[1] `https://github.com/rtavenar/LDPS/`



(a) SwedishLeaf (b) Synthetic Control

Fig. 3: Compared convergence of LDPS-E and LDPS-E on two different datasets. Shaded areas illustrate the loss span between best and worse models and dashed lines correspond to the median loss model for each variant. Best viewed in color.

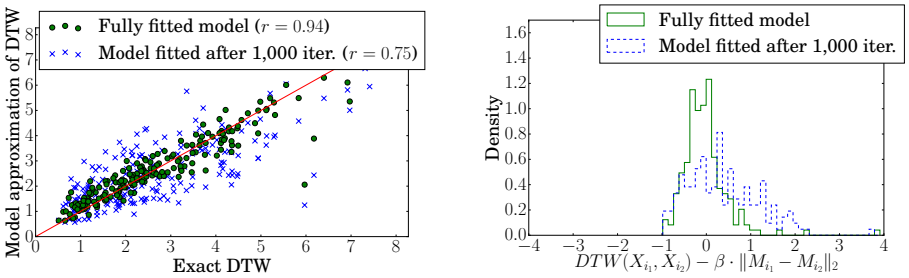


Fig. 4: Left: LDPS-E against exact DTW on dataset *SwedishLeaf*. r is the Pearson correlation coefficient. Right: Histogram of the difference between the exact DTW and the corresponding LDPS-E values.

in this section the quality of the clustering induced by LDPS. LDPS can be used for clustering by feeding a standard Euclidean k-means algorithm with the transformed time series.

Before going into clustering results, we address the issue of unsupervised model selection. The question we are asking here is the following: Is there a way to select a model that is likely to lead to a good clustering without using any *a priori* ground truth information? Figure 5 depicts the relationship between clustering quality, evaluated in terms of Normalized Mutual Information (NMI) score and model loss. The NMI score measures the coherence between the true labels of time series and the estimated cluster indices. In this figure, each dot corresponds to a partially fitted model. Dots that have high losses correspond to small numbers of iterations and the loss decreases when the number of iterations increases, as observed previously in Figure 3. An important point here is that we

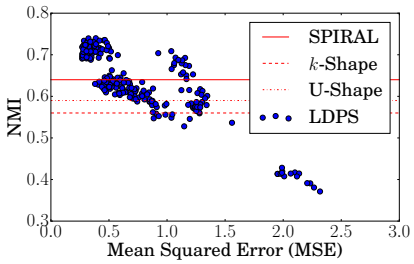


Fig. 5: Clustering quality as a function of model loss for dataset *SwedishLeaf*.

can observe a strong negative correlation between the loss associated to a model and clustering quality obtained with this model. This seems to indicate that the value of the loss can be used as a model selection criterion without using any ground truth information. For a given data set, several models can be learned (different initialization and variants of LDPS). The one leading to the smallest loss will be selected. This model selection criterion is applied in the clustering results presented below.

We conducted experiments on 15 datasets from the UEA & UCR repository [1]. Selected datasets cover a wide range of time series lengths, with varied numbers of classes and dataset sizes. For all these datasets, training and test sets are gathered, as we do not tackle the usual classification task in this piece of work. Performance of LDPS are compared with the following competitive methods. SPIRAL is the method proposed in [8] that has the same objective as LDPS but using a different approach to learn the transformation. It is combined with a k-means algorithm for clustering purposes. Reported results for SPIRAL have been obtained using the code made available by the authors[2]. k-Shape is a time series clustering algorithm proposed in [10] based on the cross-correlation measure. Reported results for k-Shape have been obtained using the `dtwclust` library of the R software, in which k-Shape is implemented. U-Shape corresponds to the clustering method using unsupervised shapelets presented in [14], and for which the code is available on a dedicated webpage[3]. For the sake of fair comparisons, we use the same shapelet lengths for this competitor and LDPS. All these competitor methods have been shown to be very efficient for time series clustering. Table 1 presents NMI scores for LDPS and competitive methods. Presented scores are medians obtained over 20 clustering runs for each method. Per-dataset performance as well as average ranks reported in this Table show the benefit of using LDPS models for this task, as they tend to get higher clustering performance. Moreover, one should note that contrary to k-Shape, our method is not specifically designed for clustering and could be used for many other machine learning tasks. Also, when compared to SPIRAL, LDPS has the key property

[2] `https://github.com/cecilialeiqi/SPIRAL`
[3] `https://sites.google.com/site/ushapelet/`

that it learns a transformation for time series that can later be applied to new data, which SPIRAL cannot do, hence strongly limiting its application scope.

Datasets	LDPS-E	LDPS-C	SPIRAL	U-Shape	k-Shape
CBF	**0.83**	0.71	0.39	0.61	0.76
CricketX	0.34	0.35	0.30	0.37	**0.38**
ElectricDevices	0.34	0.35	**0.35**	0.31	0.25
FaceAll	**0.63**	0.60	**0.63**	0.53	0.60
FaceFour	0.63	0.63	0.60	**1.00**	0.48
FiftyWords	0.68	0.64	**0.68**	0.56	0.66
Lightning2	**0.13**	0.12	0.08	0.05	0.11
Lightning7	**0.53**	**0.55**	0.48	0.50	**0.54**
OSULeaf	**0.42**	0.34	0.26	0.33	**0.42**
StarLightCurves	0.68	**0.68**	0.61	0.51	0.60
SwedishLeaf	**0.70**	0.63	0.64	0.59	0.56
SyntheticControl	0.97	**0.98**	0.81	0.83	0.72
Trace	**0.75**	**0.75**	0.50	0.73	**0.75**
TwoPatterns	0.69	**0.86**	0.11	0.32	0.30
UWaveGestureLibraryX	0.44	0.43	**0.47**	0.31	0.45
Average rank	2.13	2.33	3.40	3.87	3.20

Table 1: Comparison of Normalized Mutual Information (NMI) scores. Best performance is marked as bold. When the difference cannot be considered significant using a Mann-Whitney rank test with $p = 5\%$, several models are bolded.

5 Conclusion

In this paper, we present LDPS, an algorithm that aims at embedding time series into an Euclidean space, in which distances approximate the Dynamic Time Warping measure between raw time series. The embedding we design is based on the Shapelet Transform, that maps time series into high-dimensional vectors. The originality of our approach is that we learn shapelets using a stochastic gradient descent so that they best preserve the DTW between time series pairs. We show that the original DTW can be accurately captured by Euclidean distances in the transformed space. Clustering performance using this novel time series representation outperforms competitive methods designed specifically for this task. An interesting property of LDPS is that it leads to an ubiquitous time series representation that can feed a wide range of machine learning or indexing methods. As a future work, we will in particular aim at designing time series indexing schemes based on LDPS. As time series are embedded in a metric space, we can benefit from efficient indexing systems designed specifically in such spaces.

Bibliography

[1] Bagnall, A., Lines, J., Vickers, W., Keogh, E.: The UEA and UCR time series classification repository, `www.timeseriesclassification.com`

[2] Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., Shah, R.: Signature verification using a" siamese" time delay neural network. In: Advances in Neural Information Processing Systems. pp. 737–744 (1994)

[3] Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. JMLR 12 (2011)

[4] Esling, P., Agon, C.: Time-series data mining. ACM Comput. Surv. 45(1) (2012)

[5] Grabocka, J., Schilling, N., Wistuba, M., Schmidt-Thieme, L.: Learning time-series shapelets. In: Proc. KDD (2014)

[6] Hills, J., Lines, J., Baranauskas, E., Mapp, J., Bagnall, A.: Classification of time series by shapelet transformation. DMKD 28(4) (2014)

[7] Keogh, E., Ratanamahatana, C.A.: Exact Indexing of Dynamic Time Warping. KAIS 7 (2005)

[8] Lei, Q., Yi, J., Vaculín, R., Wu, L., Dhillon, I.S.: Similarity preserving representation learning for time series analysis (2017), `http://arxiv.org/abs/1702.03584`

[9] Lemire, D.: Faster retrieval with a two-pass dynamic-time-warping lower bound. Pattern Recognition 42(9) (2009)

[10] Paparrizos, J., Gravano, L.: k-shape: Efficient and accurate clustering of time series. In: Proc. SIGMOD (2015)

[11] Rakthanmanon, T., Campana, B., Mueen, A., Batista, G., Westover, B., Zhu, Q., Zakaria, J., Keogh, E.: Searching and mining trillions of time series subsequences under dynamic time warping. In: Proc. KDD (2012)

[12] Tan, C.W., Webb, G.I., Petitjean, F.: Indexing and classifying gigabytes of time series under time warping. In: Proc. Siam ICDM (2017)

[13] Ye, L., Keogh, E.: Time series shapelets: a new primitive for data mining. In: Proc. KDD (2009)

[14] Zakaria, J., Mueen, A., Keogh, E.: Clustering time series using unsupervised-shapelets. In: Proc. ICDM (2012)

[15] Zhang, Q., Wu, J., Yang, H., Tian, Y., Zhang, C.: Unsupervised feature learning from time series. In: Proc. IJCAI (2016)