Warping Resilient Time Series Embeddings

Anish Mathew [1] **Deepak P** [2] **Sahely Bhadra** [1]

Abstract

Time series are ubiquitous in real world problems and computing distance between two time series is often required in several learning tasks. Computing similarity between time series by ignoring variations in speed or warping is often encountered and dynamic time warping (DTW) is the state of the art. However DTW is not applicable in algorithms which require kernel or vectors. In this paper, we propose a mechanism named WaRTEm to generate vector embeddings of time series such that distance measures in the embedding space exhibit resilience to warping. Therefore, WaRTEm is more widely applicable than DTW. WaRTEm is based on a twin auto-encoder architecture and a training strategy involving warping operators for generating warping resilient embeddings for time series datasets. We evaluate the performance of WaRTEm and observed more than 20% improvement over DTW in multiple real-world datasets.

1. Introduction

Time series is a specific type of observation that contains a sequence of points, and the sequence as a whole represents a data point in many applications such as healthcare, energy and manufacturing. Due to this structure, solving problems involving time series still remains a challenging task. As an example, consider a stream of speech and two variants of it; one in which the same sequence of text is read out in a different speed, and another in which the sentence sequences are shuffled but is spoken at the same speed as the original. The original speech needs to be intuitively considered similar to the first and dissimilar to the second; this is so since shuffling of sentences changes the semantics of the speech whereas speed variations do not. In other words, a similarity measure between time series sequences should be reasonably invariant to speed/phase shifts, but be sensitive to the ordering in the sequence. Speed/phase shifts are often referred to as *warping*; dynamic time warping(Berndt & Clifford, 1994), or DTW for short, has been an extremely successful distance measure between time series owing to its ability to be resilient to warping. It has been so successful that it has been widely used in a variety of domains (Mueen & Keogh, 2016) that involve time series data such as biometrics, anthropology and finance.

Majority of machine learning algorithms expect data objects to be represented as feature-vectors. The data vectors or embeddings are expected to be meaningful within the vector spaces that they reside in; in other words, similarities between vectors using reasonable vector similarities/distances (e.g., euclidean) are expected to reflect semantic relationships between data points. Vector/Matrix processing methods include classical analytics methods such as those for non-negative matrix factorization (Xu et al., 2003), locally linear embedding (Roweis & Saul, 2000) and a large majority of deep learning methods. While research into deep learning has evolved mechanisms for identifying space-invariant and localized features (e.g., convolutional units (Krizhevsky et al., 2012)) and long-range sequential dependencies (e.g., LSTM (Greff et al., 2017)), DTW still remains a very competitive method (Bagnall et al., 2017).

While time series data may be naively considered as vectors by ignoring the sequential information, such vectors don't yield a meaningful representation within their vector spaces. This is so since much of the meaningfulness is embedded within the sequential information as well as the expectation of warping invariance as modelled within bespoke time series similarity measures such as DTW. Consequently, as recognized in earlier work (Lei et al., 2017), this scenario makes the many machine learning models built for feature-vectors inapplicable to time series data. As in that paper, we consider the route of converting a time series dataset into multi-dimensional vectors such that distances/similarities in the resultant vector space are meaningful. The vast library of matrix/vector oriented machine learning algorithms can then be applied on the resultant dataset of embeddings.

Our Contributions: Our contributions are as follows:

- We develop two *warping operators* which transform time series data into their warped variations.
- We propose a *novel neural architecture based on auto-*

[1]Computer Science and Engineering, Indian Institute of Technology, Palakkad, India [2]Queen's University Belfast, UK. Correspondence to: Anish Mathew <anishmm1997@gmail.com>, Deepak P <deepaksp@acm.org>.

Proceedings of the Time Series Workshop at ICML 2019.

encoders that can leverage a time series dataset along with warping variations in order to convert time series data into meaningful warping resilient vector embeddings.

- Through an empirical evaluation over real world datasets, we illustrate that such vector embeddings yield 1-NN classification accuracies that are competitive to DTW 1-NN over corresponding time series. We also show that the they yield better accuracies than the original time series when used with other models developed for static data.

2. Related Work

Most related to our work is a recent work (Lei et al., 2017) addressing the same task as ours, that of converting a time series dataset into a set of vector embeddings. Starting from a dataset of n time series, they build an $n \times n$ matrix of pair-wise similarities that is defined using DTW distances. This is done by performing $\mathcal{O}(n \times log(n))$ DTW computations, and approximating the other entries in the matrix using a low-rank assumption. If a time series is of length m, a DTW computation takes time of the order of $\mathcal{O}(m^2)$. This makes just the matrix creation step a process in $\mathcal{O}(n \times log(n) \times m^2)$. The similarity matrix is then subject to symmetric matrix factorization so that a vector embedding matrix $\mathbf{X}$ is learnt such that $\mathbf{X}\mathbf{X}^T$ approximates the similarity matrix well. This calls for eigen decomposition which is superlinear in n as well. This makes it impractical for most large datasets. In contrast to this work, the method we propose is linear in both n and m, making our method significantly more scalable.

There are two more recent methods that consider learning embeddings to approximate DTW distances. The first, called *Jiffy* (Shanmugam et al., 2018), targets the case of multi-variate time series, and proposes a convolutional neural architecture. We differ from their task setting in that we consider the univariate setting for devising our method. The second method (Lods et al., 2017) models the embedding task as one of identifying a specified number of shapelets of specified lengths (both being method parameters) that enable transforming time series data into embeddings closely approximating DTW distances. Our model is not constrained to produce shapelets, and targets to train a neural architecture that embodies the transformation within itself.

Another recent work (Franceschi et al., 2019) makes use of convolutional neural networks in a framework heavily inspired by word2vec (Mikolov et al., 2013). They train a neural network so that it learns an embedding for a time series subsequence that would be closer to the embedding of the larger sequence containing it (positive example) than to a different random series (negative example). SVM classifiers trained over such embeddings are shown to be better than DTW 1-NN classifier in accuracy. In contrast with their intent of learning representations that are more suitable to train SVMs, our intent is to ensure that the distances between embeddings are meaningful locally. This intent entails a different evaluation target, that of optimizing for performance of a 1-NN (or k-NN) classifier over the embeddings.

3. WaRTEm: Proposed Method

Consider a time-series dataset $\mathcal{T} = \{T_1, \ldots, T_n\}$, where each individual T_i represents a time series of length m. Our time series embedding method, WaRTEm (short for Warping Resilient Time Series Embeddings), transforms this dataset into a set of vector embeddings that reside in a vector space $\mathbb{R}^d$ (d being pre-specified), with each T_i being assigned to its own embedding $V_i \in \mathbb{R}^d$, the collection of embeddings being denoted by $\mathcal{V}$.

$$\{T_1, \ldots, T_n\} \xrightarrow{\text{WaRTEm}} \{V_1, \ldots, V_n\}$$

To ensure that $\mathcal{V}$ comprises meaningful representations, we would like simple vector distances among them to be semantically meaningful like warping invariant distance measures such as DTW over corresponding time series in $\mathcal{T}$. On the lines of manifold learning methods for dimensionality reduction (e.g., SNE (Hinton & Roweis, 2003), LLE (Roweis & Saul, 2000)), we look to maintain warping resilience only within local neighborhoods. Accordingly, our evaluation is performed by comparing 1-NN classification accuracies over vectors in $\mathcal{V}$ using Euclidean distance vis-a-vis DTW 1-NN over $\mathcal{T}$.

We first introduce a set of warping operators that transform a time series into warped versions. This is followed by outlining our twin auto-encoder architecture that can utilize time series and their transformations to learn vector embeddings for time series data.

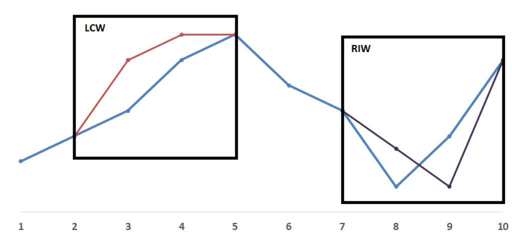

Figure 1. Warping Operators Example: The warping operators LCW and RIW are illustrated by the changes they effect within the warping windows. Blue indicates the original time series, with others the respective warped versions.

3.1. Warping Operators

Our warping operators are intended to transform a time series into another one such that the pair are warped variants of each other. In other words, the pair would be expected to be judged as very similar under warping aware distance measures such as DTW. Each of our warping operators involve deleting off a point/value from the original series, and adding a point/value to the *left* or *right* of it to restore the original length.

Copy Warping: Consider a window of four points from a time series T denoted as $[p_1, p_2, p_3, p_4]$; this is the warping focus window. The *left* and *right* copy warping operators are illustrated as below:

$$Left : [p_1, p_2, p_3, p_4] \rightarrow [p_1, p_3, p_4] \rightarrow [p_1, p_3, p_4, p_4]$$

$$Right : [p_1, p_2, p_3, p_4] \rightarrow [p_1, p_2, p_4] \rightarrow [p_1, p_1, p_2, p_4]$$

The *LeftCopyWarp* (LCW) operator deletes off p_2 from the window and duplicates p_4 to restore the length, whereas RCW deletes off p_3 and duplicates p_1. In other words, LCW shrinks the left side of the window and extends the right endpoint to a plateau, whereas the vice versa is the case for RCW. It may be noted that a time series T and it's warped variant $LCW(T)$ or $RCW(T)$ differ only in the values that they take within the warping focus window.

Interpolation Warping: Consider a warping focus window $[p_1, p_2, p_3, p_4]$ as earlier. The variants of the interpolation warping are as follows:

$$Left : [p_1, p_2, p_3, p_4] \rightarrow [p_1, p_3, p_4] \rightarrow [p_1, p_3, \frac{p_3 + p_4}{2}, p_4]$$

$$Right : [p_1, p_2, p_3, p_4] \rightarrow [p_1, p_2, p_4] \rightarrow [p_1, \frac{p_1 + p_2}{2}, p_2, p_4]$$

The LIW operator, as in the case of LCW, shrinks the left side of the window, but then extends the right side by a slope (as against a plateau in LCW). This slope is formed by adding a point that is midway between p_3 and p_4 both in terms of its value and placement. RIW is simply the mirror image of LIW.

Figure 1 illustrates examples of series formed by two of our warping operators to help visualize the changes effected by them.

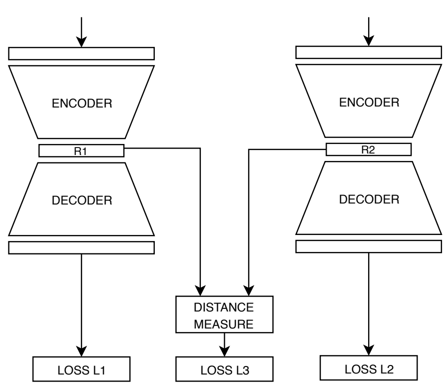

Figure 2. Twin Auto-encoder Architecture

3.2. Twin Auto-Encoder Architecture

The WaRTEm neural network architecture comprising twin auto-encoders (AEs) is illustrated in Figure 2. Our network is modelled to take a pair of time-series sequences as input. Each time series sequence in the input pair is passed through a separate convolutional AE (shown side by side in Fig. 2). As is typical of AEs, the respective time series get converted into an internal representation (aka code) through the encoder, with the decoder expected to re-construct the original input to high accuracy from the code. $L1$ and $L2$ indicate the conventional reconstruction losses for the separate AEs. The twinning between the AEs is achieved through the introduction of a new loss term, $L3$ which is designed as the squared euclidean distance $\|R_1 - R_2\|_2^2$ between the codes (R_1 and R_2) corresponding to the pair of input time series. To learn embeddings that cater to a different similarity measure, WaRTEm can be adapted by designing a corresponding loss term between R_1 and R_2. As indicated, $L3$ is propagated back through the encoder parts of the AEs, and does not affect the decoder weights. In other words, in addition to training the separate auto-encoders to reconstruct their respective inputs, we also try to ensure that the codes corresponding to the time series pairs are close to each other. The way this maps to our intent of learning warping resilient time series embeddings will be outlined in our warping-based training strategy in the next section.

The encoder part of the AE comprises a sequence of pairs of 1d convolution and maxpooling layers followed by a final fully connected layer, whereas the decoder analogously uses upsampling and pairs of 1d convolution layers.

3.3. Training Strategy and Embeddings

The training strategy indicates the manner in which we use the time series dataset $\mathcal{T}$ in training our twin AE architecture. The twin AE architecture is motivated by our observation based on empirical studies that *warping resilience* is quite complex for a single AE to learn (using, for example, a variant of denoising AE). Thus, we specialize the task to two, viz., *leftward* and *rightward* warping resilience, so separate AEs can learn them separately.

For each time series $T \in \mathcal{T}$ and for each directionality of warping (*left* or *right*), we generate a warped variant. Consider the choice of *left* direction; we first sample a random integer r between 0 and $(0.5 \times length(T))$. We then progressively perform r leftward warpings over randomly chosen warping focus windows, choosing LCW or LIW depending on the experiment, to generate a warped variant of T, denoted as $L(T)$. The pair $[L(T), T]$ thus generated forms an element of the training dataset for our twin AE. Analogously, the choice of *right* direction yields a warped variant $R(T)$, forming a training pair $[T, R(T)]$. It may be noted that each T thus generates two training pairs, one for *left* and another for *right*. The construction of the ordering in the pairs is pertinent to the separation of warping resilience learning; the left entry in the pair is either a left-warped variant or the original series, but never a right-warped variant (and similarly for the right entry). This creates a warping directionality co-ordination between the AEs which helps separate the nature of learnings within the respective AEs.

The training process is continued for as many epochs as needed; we use a held-out dataset to compute loss trends across epochs to effect early stopping. Then, each time series T is passed through each of the AEs separately to generate their left and right AE codes, which are then averaged to be used as the corresponding embedding V. This completes the description of WaRTEm transformation from time series to embeddings.

4. Experiments

For all experiments, datasets were taken from the UCR Time Series Classification Archive (Dau et al., 2018). 1NN Euclidean and 1NN DTW accuracies reported in table 1 were calculated from the corresponding error rates provided in the archive. We use the provided train test split for all models to ensure fair comparison. For all experiments, we set the representation length as 20% of the series length. Different datasets give best accuracies at different code lengths but we found that 20% works well overall. Three models, corresponding to Copy warping, Interpolation warping and a combination of both are explored. For the combination, at each focus window, CW or IW is chosen with equal probability. For each model, we train the network with 10 different initializations and take the average 1NN accuracy. Same initialization seeds are used across the three methods. The best accuracy among the three is reported under WaRTEm-NN in table 1.

Table 1 shows that WaRTEm provides better 1NN performance than both Euclidean 1NN and DTW 1NN for most datasets. Moreover, WaRTEm provides a way to use numerous static data models with time series datasets. For instance, Table 2 reports the performances of a simple 3-layer neural network classifier and XGBoost (Chen & Guestrin, 2016) on these datasets. The network comprises of 3 fully-connected layers with $max(10, \lfloor L/10 \rfloor)$, 50 and $n\_classes$ nodes respectively, where L is the length of the input series and $n\_classes$ the number of classes. For XGBoost, the maximum depth and number of estimators is decided through 5-fold cross-validation on the train data. For both the models, for each input dataset/embeddings, training is carried out 10 times and best test accuracy is selected. For WaRTEm, average accuracy over 10 such embeddings (learned in the previous experiment) is reported.

As shown in table 2, both WaRTEm-DL and WaRTEm-XGB outperform their counterparts on 6 out of 10 datasets. WaRTEm gives overall best performance (marked in bold) for half the datasets investigated. These results suggest

Table 1. Comparing accuracy of 1NN classifier using 1)Euclidean distance on lower order representation given by WaRTEm (WaRTEm-NN), 2)Euclidean distance on original time series(Eucl-NN) and 3)DTW distance on original time series(DTW-NN). Best accuracy for each dataset is in bold.

Dataset	WaRTEm-NN	Eucl-NN	DTW-NN
ArrowHead	77.43 ± 1.81	**80.00**	70.29
Beef	$\mathbf{67.33 \pm 5.73}$	66.67	63.33
EthanolLevel	$\mathbf{29.68 \pm 2.95}$	27.4	27.6
FaceAll	70.32 ± 4.02	71.36	**80.77**
InsectWbSnd	$\mathbf{58.02 \pm 0.43}$	56.16	35.51
Lightning7	64.93 ± 3.58	57.53	**72.6**
SHandGenCh2	$\mathbf{92.22 \pm 1.89}$	76.17	80.17
SHandMovCh2	$\mathbf{78.42 \pm 2.48}$	36.89	58.44
SwedishLeaf	$\mathbf{81.01 \pm 1.37}$	78.88	79.2
Yoga	83.19 ± 0.55	83.03	**83.63**

Table 2. Comparing accuracy of deep learning based classifier trained on 1)lower order representation given by WaRTEm (WaRTEm-DL) and 2)original time series(DL) and XGBoost on 1) WaRTEm embedding (WaRTEm-XGB) 2)original time series (XGB)

Dataset	WaRTEm-DL	DL	WaRTEm-XGB	XGB
ArrowHead	76.72 ± 3.03	**77.59**	66.38 ± 4.76	60.92
Beef	36.21 ± 5.61	34.48	$\mathbf{65.86 \pm 8.78}$	65.52
EthanolLevel	54.51 ± 7.51	**81.96**	34.75 ± 3.77	51.3
FaceAll	14.42 ± 2.19	11.96	65.62 ± 2.3	**82.53**
InsectWbSnd	62.03 ± 0.8	**62.25**	59.59 ± 1.49	61.85
Lightning7	$\mathbf{65.42 \pm 3.65}$	56.94	63.19 ± 4.86	56.94
SHandGenCh2	$\mathbf{94.61 \pm 0.87}$	65.44	92.09 ± 1.6	91.15
SHandMovCh2	33.41 ± 3.18	20.94	$\mathbf{68.4 \pm 1.9}$	53.9
SwedishLeaf	$\mathbf{83.08 \pm 0.73}$	81.09	80.37 ± 1.49	82.37
Yoga	79.67 ± 1.36	**81.73**	79.84 ± 0.79	79.46

that WaRTEm embeddings are suitable for using with static models and can help leverage such static models for time series analysis.

5. Conclusion

We propose WaRTEm, a model for generating lower dimensional vector embeddings for time series data such that the embeddings exhibit resilience to warping in local neighborhoods. To this end, we introduce two warping operators which transform time series into their warped variations, and a novel neural architecture which along with our proposed training strategy, can convert time series data into their lower dimensional warping resilient embeddings. In addition to outperforming DTW and Euclidean distance on NN tasks, the study also demonstrates that our embeddings lend themselves well to other static learning models.

Acknowledgements: The first and second authors were partially supported by the D-DIEM project funded by UKIERI.

References

Bagnall, A., Lines, J., Bostrom, A., Large, J., and Keogh, E. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 31(3):606–660, 2017.

Berndt, D. J. and Clifford, J. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pp. 359–370. Seattle, WA, 1994.

Chen, T. and Guestrin, C. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794. ACM, 2016.

Dau, H. A., Keogh, E., Kamgar, K., Yeh, C.-C. M., Zhu, Y., Gharghabi, S., Ratanamahatana, C. A., Yanping, Hu, B., Begum, N., Bagnall, A., Mueen, A., and Batista, G. The ucr time series classification archive, October 2018. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/.

Franceschi, J.-Y., Dieuleveut, A., and Jaggi, M. Unsupervised scalable representation learning for multivariate time series. *arXiv preprint arXiv:1901.10738*, 2019.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., and Schmidhuber, J. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28 (10):2222–2232, 2017.

Hinton, G. E. and Roweis, S. T. Stochastic neighbor embedding. In *Advances in neural information processing systems*, pp. 857–864, 2003.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Lei, Q., Yi, J., Vaculin, R., Wu, L., and Dhillon, I. S. Similarity preserving representation learning for time series analysis. *arXiv preprint arXiv:1702.03584*, 2017.

Lods, A., Malinowski, S., Tavenard, R., and Amsaleg, L. Learning dtw-preserving shapelets. In *International Symposium on Intelligent Data Analysis*, pp. 198–209. Springer, 2017.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pp. 3111–3119, 2013.

Mueen, A. and Keogh, E. Extracting optimal performance from dynamic time warping. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2129–2130. ACM, 2016.

Roweis, S. T. and Saul, L. K. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500): 2323–2326, 2000.

Shanmugam, D., Blalock, D., and Guttag, J. Jiffy: A convolutional approach to learning time series similarity. 2018.

Xu, W., Liu, X., and Gong, Y. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pp. 267–273. ACM, 2003.