Learning Distance for Sequences by Learning a Ground Metric Supplementary Material

Bing Su [1] **Ying Wu** [2]

1. Connections between the unified formulation and some other distances

Besides DTW and OPW, some other distance measures for sequences can also fit into our unified formulation.

Variants of DTW. Most variants of DTW actually impose additional or relaxed constraints on the feasible set and therefore fit into our formulation. For example, in (Ratanamahatana & Keogh, 2004), additional locality constraints $\boldsymbol{T}\mathbf{1}_{L_Y} \leq a\mathbf{1}_{L_X}, \boldsymbol{T}^T\mathbf{1}_{L_X} \leq a\mathbf{1}_{L_Y}$ are imposed to restrict the amount of alignment; in (Su et al., 2017), the continuity constraint is stricter by setting $\boldsymbol{T}\mathbf{1}_{L_Y} = \mathbf{1}_{L_X}$ or $\boldsymbol{T}^T\mathbf{1}_{L_X} = \mathbf{1}_{L_Y}$. Edit distances are originally defined for string sequences with discrete values, their extensions to multi-dimensional numeral sequences such as the edit distance with real penalty (Chen & Ng, 2004) and the time warp edit distance (Marteau, 2009) are achieved by modifying the dynamic programming process of DTW, where different costs are assigned to the movements in different directions in $\boldsymbol{T}$. These distances are defined recurrently without an explicit objective function.

Optimal Transport Distance (OT) (Villani, 2008). Originally, OT was developed for measuring two distributions. A sequence can be viewed as an empirical probability by taking its elements as independent supporting points. In this way, although the temporal information is lost, OT can be applied to measure the distance between sequences. It calculates a transport matrix $\boldsymbol{T}$ among elements in sequences which achieves the minimum energy to transport one sequence to another. t_{ij} takes value in the range of $[0, 1]$, which denotes the amount of probability to be transported from $\boldsymbol{x}_i$ to $\boldsymbol{y}_j$. Since the total probabilities to transport and to be transported are fixed, the constraints $\boldsymbol{T}\mathbf{1}_{L_Y} = \frac{1}{L_X}\mathbf{1}_{L_X}$ and $\boldsymbol{T}^T\mathbf{1}_{L_X} = \frac{1}{L_Y}\mathbf{1}_{L_Y}$ are imposed to $\boldsymbol{T}$. OT naturally has the form of Eq. (1) and Eq. (3) in the main text, where

$$\begin{aligned}&\mathscr{R}(\boldsymbol{T}) = 0;\\ &\boldsymbol{\Phi} = \{\boldsymbol{T} \in \mathbb{R}_+^{L_X \times L_Y} | \boldsymbol{T}\mathbf{1}_{L_Y} = \tfrac{1}{L_X}\mathbf{1}_{L_X},\\ &\quad \boldsymbol{T}^T\mathbf{1}_{L_X} = \tfrac{1}{L_Y}\mathbf{1}_{L_Y}\}\end{aligned}, \tag{1}$$

Sinkhorn distance (Cuturi, 2013). Solving the original OT is expensive. The Sinkhorn distance smooths the OT problem by adding an entropy regularization term to $\boldsymbol{T}$, and the resulting optimum can be efficiently determined by Sinkhorn's fixed point iterations. It instantiates the formulation Eq. (3) in the main text by setting:

$$\begin{aligned}&\mathscr{R}(\boldsymbol{T}) = \lambda(\sum_{i=1}^{N}\sum_{j=1}^{M} t_{ij}\log t_{ij});\\ &\boldsymbol{\Phi} = \{\boldsymbol{T} \in \mathbb{R}_+^{L_X \times L_Y} | \boldsymbol{T}\mathbf{1}_{L_Y} = \tfrac{1}{L_X}\mathbf{1}_{L_X},\\ &\quad \boldsymbol{T}^T\mathbf{1}_{L_X} = \tfrac{1}{L_Y}\mathbf{1}_{L_Y}\}\end{aligned}, \tag{2}$$

where λ is a preset balancing coefficient.

Variants of Sinkhorn. Several variants of the regularized OT distance modify the construction of the cost matrix $\boldsymbol{D}$ to incorporate additional information. In the Transportation L^p distance (Thorpe et al., 2017), $\boldsymbol{D}$ is constructed as

$$\boldsymbol{D} := [d^p(\boldsymbol{x}_i, \boldsymbol{y}_j) + \lambda|i - j|^p]_{ij}, \tag{3}$$

$d^p(\boldsymbol{x}_i, \boldsymbol{y}_j)$ is the distance between $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$ raised to power p. As shown in (Su & Hua, 2018), the OPW distance is also equivalent to the entropy-regularized OT with a ground metric $\tilde{\mathbf{D}}$:

$$\tilde{\mathbf{D}} = \boldsymbol{D} - \lambda_1\boldsymbol{E} + \lambda_2\frac{\boldsymbol{F}}{2\sigma^2}, \tag{4}$$

where $\boldsymbol{E} = \left[\frac{1}{(\frac{i}{N} - \frac{j}{M})^2 + 1}\right]_{ij}$, and $\boldsymbol{F} = \left[\ell^2(i, j)\right]_{ij}$.

In (Flamary et al., 2018), a dimensionality reduction method is built on the regularized OT for vector representations. The training vector samples of a class are used to represent the empirical probability of the class. The Sinkhorn distance between the empirical probabilities of two classes is used as the separability measure between the two class. This method cannot be directly extended to sequence data, because each supporting sample of the empirical probability of a sequence class is a vector sequence. To calculate the Sinkhorn distance between such two empirical

[1]Science & Technology on Integrated Information System Laboratory, Institute of Software, Chinese Academy of Sciences, Beijing, China [2]Department of Electrical and Computer Engineering, Northwestern University, Evanston, IL, USA. Correspondence to: Bing Su <subingats@gmail.com>.

Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

probabilities, each element of the cost matrix $\boldsymbol{D}$ should be a combinational meta-distance such as DTW or OPW between two sequence samples. The latent alignment or transport structures will block the back-propagation of the derivatives of the transformation. Therefore, automatic differentiation cannot be performed anymore.

2. Kernelized version of RVSML

A linear transformation may not be able to project sequences to their corresponding virtual target sequences adequately. We present a kernelized version of RVSML to learn implicit non-linear metric for sequence data.

We first reformulate the closed form solution for updating $\boldsymbol{W}$.

$$\boldsymbol{W}^* = \boldsymbol{A}^{-1}(\sum_{n=1}^{N}\sum_{i=1}^{L^n}\sum_{j=1}^{l^n} t_{ij}^n \boldsymbol{x}_i^n \boldsymbol{v}_j^{nT}), \tag{5}$$

where

$$\boldsymbol{A} = \sum_{n=1}^{N}\sum_{i=1}^{L^n}\sum_{j=1}^{l^n} t_{ij}^n \boldsymbol{x}_i^n \boldsymbol{x}_i^{nT} + \beta N \boldsymbol{I}. \tag{6}$$

Recall that

$$\boldsymbol{X}^n = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_{L^n}] \in \mathbb{R}^{b \times L^n},$$
$$\boldsymbol{V}^n = [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_{l^n}] \in \mathbb{R}^{b' \times l^n},$$
$$\boldsymbol{T}^n \in \mathbb{R}^{L^n \times l^n}.$$

For each training sequence, we construct a data sample matrix $\hat{\boldsymbol{X}^n}$ as a 1 by l^n tiling of copies of $\boldsymbol{X}^n$:

$$\hat{\boldsymbol{X}^n} = [\boldsymbol{X}^n, \cdots, \boldsymbol{X}^n] \in \mathbb{R}^{b \times (L^n l^n)}.$$

We construct a virtual sample matrix $\hat{\boldsymbol{V}^n}$ by copying each element L^n times and then concatenating column by column:

$$\hat{\boldsymbol{V}^n} = [\boldsymbol{v}_1^n, \cdots, \boldsymbol{v}_1^n, \cdots, \boldsymbol{v}_{l^n}^n, \cdots, \boldsymbol{v}_{l^n}^n] \in \mathbb{R}^{b' \times (l^n L^n)}.$$

We vectorize $\boldsymbol{T}^n$ by concatenating its columns:

$$\hat{\boldsymbol{T}^n} = [t_{1,1}^n, \cdots, t_{L^n,1}^n \cdots, t_{1,l^n}^n, \cdots, t_{L^n,l^n}^n] \in \mathbb{R}^{(L^n l^n) \times 1}.$$

We finally construct the overall data matrix $\hat{\boldsymbol{X}}$ as the concatenation of all $\hat{\boldsymbol{X}^n}$, $n = 1, \cdots, N$ by row:

$$\hat{\boldsymbol{X}} = [\hat{\boldsymbol{X}^1}, \cdots, \hat{\boldsymbol{X}^N}] \in \mathbb{R}^{b \times \sum_n (L^n l^n)}.$$

Similarly, the overall virtual matrix $\hat{\boldsymbol{V}}$ is constructed as the concatenation of all $\hat{\boldsymbol{V}^n}$, $n = 1, \cdots, N$ by row:

$$\hat{\boldsymbol{V}} = [\hat{\boldsymbol{V}^1}, \cdots, \hat{\boldsymbol{V}^N}] \in \mathbb{R}^{b' \times \sum_n (L^n l^n)}.$$

We also concatenate all $\hat{\boldsymbol{T}^n}$, $n = 1, \cdots, N$ into a long vector $\hat{\boldsymbol{T}}$:

$$\hat{\boldsymbol{T}} = [\hat{\boldsymbol{T}^1}^T, \cdots, \hat{\boldsymbol{T}^N}^T]^T \in \mathbb{R}^{\sum_n (L^n l^n) \times 1}.$$

Then the solution Eq. (5) can be reformulated as:

$$\boldsymbol{W}^* = (\hat{\boldsymbol{T}} \odot (\hat{\boldsymbol{X}}\hat{\boldsymbol{X}}^T) + \beta N \boldsymbol{I})^{-1}(\hat{\boldsymbol{T}} \odot (\hat{\boldsymbol{X}}\hat{\boldsymbol{V}}^T)), \tag{7}$$

where $\odot$ denotes element-wise product.

We further define $\hat{\boldsymbol{X}}_s$ and $\hat{\boldsymbol{V}}_s$ by

$$\hat{\boldsymbol{X}}_s = \hat{\boldsymbol{T}}^{\frac{1}{2}} \odot \hat{\boldsymbol{X}},$$
$$\hat{\boldsymbol{V}}_s = \hat{\boldsymbol{T}}^{\frac{1}{2}} \odot \hat{\boldsymbol{V}},$$

where $\hat{\boldsymbol{T}}^{\frac{1}{2}}$ should be understood as element-wise square root, then Eq. (7) can further be expressed as

$$\boldsymbol{W}^* = (\hat{\boldsymbol{X}}_s\hat{\boldsymbol{X}}_s^T + \beta N \boldsymbol{I})^{-1}(\hat{\boldsymbol{X}}_s\hat{\boldsymbol{V}}_s^T), \tag{8}$$

This solution is also equivalent to

$$\boldsymbol{W}^* = \hat{\boldsymbol{X}}_s(\hat{\boldsymbol{X}}_s^T\hat{\boldsymbol{X}}_s + \beta N \boldsymbol{I})^{-1}\hat{\boldsymbol{V}}_s^T, \tag{9}$$

Let $\phi(\boldsymbol{x})$ be a nonlinear transformation for the element $\boldsymbol{x}$ in sequences. $K(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x})^T\phi(\boldsymbol{x}')$ is the corresponding kernel. Following (Perrot & Habrard, 2015), Eq. (9) can be kernelized as

$$\boldsymbol{W}^* = \phi(\hat{\boldsymbol{X}}_s)(K_{\hat{\boldsymbol{X}}_s} + \beta N \boldsymbol{I})^{-1}\hat{\boldsymbol{V}}_s^T, \tag{10}$$

where $\phi(\hat{\boldsymbol{X}}_s) = [\phi(\hat{\boldsymbol{x}}_{si})]_i$ is generated by applying the transformation to all column vectors in $\hat{\boldsymbol{X}}_s$ and $K_{\hat{\boldsymbol{X}}_s} = \phi(\hat{\boldsymbol{X}}_s)^T\phi(\hat{\boldsymbol{X}}_s)$.

The transformed element $\boldsymbol{x}$ in any sequence with the projection Eq. (10) is given by

$$\begin{aligned} &\boldsymbol{W}^{*T}\phi(\boldsymbol{x}) = (\phi(\boldsymbol{x})^T\boldsymbol{W}^*)^T \\ &= (\phi(\boldsymbol{x})^T\phi(\hat{\boldsymbol{X}}_s)(K_{\hat{\boldsymbol{X}}_s} + \beta N \boldsymbol{I})^{-1}\hat{\boldsymbol{V}}_s^T)^T \\ &= (K_{\hat{\boldsymbol{X}}_s}(\boldsymbol{x})(K_{\hat{\boldsymbol{X}}_s} + \beta N \boldsymbol{I})^{-1}\hat{\boldsymbol{V}}_s^T)^T \end{aligned} , \tag{11}$$

where $K_{\hat{\boldsymbol{X}}_s}(\boldsymbol{x}) = [K(\boldsymbol{x}, \hat{\boldsymbol{x}}_{s1}), \cdots, K(\boldsymbol{x}, \hat{\boldsymbol{x}}_{s\sum_n(L^n l^n)})]$.

Once the transformation Eq. (10) is obtained, the projected sequences can be obtained by applying Eq. (11) to all the elements. $\boldsymbol{D}_{\boldsymbol{I}}^n(\boldsymbol{W})$ for any training-virtual sequence pair can then be calculated. Given $\boldsymbol{D}_{\boldsymbol{I}}^n(\boldsymbol{W})$, $\boldsymbol{T}^n$ can be updated in Step 5 without any modification. Therefore, to obtain the kernelized version of Alg. 1, we only need to replace the update of the transformation in Step 3 with the kernelized solutions Eq. (10) and Eq. (11).

Performing the kernel version of RVSML needs to process matrices with a size of $\sum_n(L^n l^n) \times \sum_n(L^n l^n)$, therefore is often prohibitive in practice due to the huge overhead on both time and memory.

3. Deep version of RVSML

The proposed RVSML can be extended to learn non-linear representations by replacing the linear transformation $\boldsymbol{W}\boldsymbol{X}$ with a non-linear transformation by a deep neural network. Let $h(\boldsymbol{X}, \theta)$ denotes the final layer activations of the network, where θ represents the set of parameters of the network, then the objective function of deep RVSML is formulated as follows by omitting the regularization on the transformation.

$$\begin{aligned} &\min_{\boldsymbol{W}} \frac{1}{N} \sum_{n=1}^{N} g_{\boldsymbol{I}}(h(\boldsymbol{X}^n, \theta), \boldsymbol{V}^n) = \frac{1}{N} \sum_{n=1}^{N} \langle \boldsymbol{T}^{n*}, \boldsymbol{D}_{\boldsymbol{I}}^n(h, \theta) \rangle \\ &s.t.\ \boldsymbol{T}^{n*} = arg \min_{\boldsymbol{T} \in \boldsymbol{\Phi}} \langle \boldsymbol{T}^n, \boldsymbol{D}_{\boldsymbol{I}}^n(h, \theta) \rangle + \mathscr{R}(\boldsymbol{T}^n) \end{aligned} \tag{12}$$

where $\boldsymbol{D}_{\boldsymbol{I}}^n(h, \theta)$ denotes the matrix of all the pairwise Euclidean distances between the non-linear representations generated by the network with the parameter set θ. The optimization of Eq. (12) follows the similar alternating procedures with the linear-RVSML. When the parameters of the deep network are fixed, The procedure for updating the alignments remain the same. That is, after the non-linear representations are obtained by the network, the alignments between any transformed sequences can be inferred by optimizing Eq. (14) in the main text. Differently, when the alignments are fixed, no closed form solution is available for updating the non-linear transformation. To update the parameters of the network, we simply calculate the subgradient of Eq. (12) w.r.t. θ and employ the back propagation algorithm to train the network. Through the chain rule, The subgradient w.r.t. θ can be obtained by the product of the gradient of Eq. (12) w.r.t. h and the gradient of h (the activation of the final layer) w.r.t. θ. Any type of neural network can be employed and given a network with a specific structure, the latter can be computed in a standard manner.

The two procedures are alternated until convergence. In this way, the neural network and the alignments can be jointly learned. However, deep networks generally requires a large amount of training sequences to estimate massive parameters. We leave the evaluation of the deep version of RVSML for future work.

4. Another virtual sequence generation method

The virtual sequences can be generated using various methods according to the desired properties of the metric, the prior knowledge on the data, etc. We present another virtual sequence generation method in this section.

Large-margin-based virtual sequences. In this approach, we construct a virtual sequence for a training sequence based on the relative location of the training sequence w.r.t. other sequences. Special attention should be pay to those sequences distributed near the boundaries among different classes. If we push any sequence near the boundaries to another sequence far away from the boundaries, the margins among different sequence classes would become larger.

Specifically, for each training sequence, we calculate its distances to the nearest training sequence from the same class and the nearest sequence from other classes w.r.t. a meta-distance measure as the smallest intra-class and inter-class distances, respectively. We also compute the sums of all pair-wise distances between this sequence and all other sequences from the same class and those from different classes as the overall intra-class and inter-class distances, respectively. We select the sequences whose overall inter-class distance is larger than the overall intra-class distance while the smallest inter-class distance is also larger than the smallest intra-class distance as candidates.

For each class, we select k training sequences of this class from the candidates with the largest differences between the overall inter-class and intra-class distances. These sequences are considered to have large margins with other classes and hence serve as the target sequences of this class. There are Ck target sequences in total. The virtual sequence for a training sequence is selected as the nearest target sequence of the same class. We summarize the generation process of such *large-margin (LM) based virtual sequences* in Alg. 1.

We denote the virtual sequences generated in the main text by *temporal-structure (TS) based virtual sequences*. We denote RVSML with TS-based virtual sequences and LM-based virtual sequences by RVSML-TS and RVSML-LM, respectively. RVSML-TS is equivalent to RVSML in the main text.

5. Influence of hyper-parameters

The RVSML framework has one hyper-parameter: β. The generation of the virtual sequences has one hyper-parameter: m, the number of elements in each virtual sequence. We evaluate the influence of them on RVSML instantiated by OPW on the MSR Action3D dataset. We first evaluate the influence of m by fixing β to 0.01. The performances as functions of m are shown in Fig. 1(a). We observe that a small m within the range of 4 to 8 works well. When $m = 1$, RVSML assumes that each sequence class contains only one temporal structure and degenerates into applying RVML by viewing elements in sequences as independent samples. As shown in Sec. 5 in the main text, RVSML with $m > 1$ outperforms RVML, this indicates that introducing more temporal structures helps to better explore the temporal information. However, since the dimen-

Algorithm 1 Generating LM-based Virtual Sequences

```
1: Input: A set of labeled training sequences $\{\boldsymbol{X}^n, z^n\}_{n=1}^N$; the number of target sequences per class $k$
2: Output: The associated virtual sequences $\{\boldsymbol{V}^n\}_{n=1}^N$
3: for $n = 1, \cdots, N$ do
4:     $s_o^a(n) = \sum_{n':z^{n'}=z^n, n'\neq n} g_{\boldsymbol{I}}(\boldsymbol{X}^n, \boldsymbol{X}^{n'})$;
5:     $s_o^r(n) = \sum_{n':z^{n'}\neq z^n} g_{\boldsymbol{I}}(\boldsymbol{X}^n, \boldsymbol{X}^{n'})$;
6:     $s_m^a(n) = \min_{n':z^{n'}=z^n, n'\neq n} g_{\boldsymbol{I}}(\boldsymbol{X}^n, \boldsymbol{X}^{n'})$;
7:     $s_m^r(n) = \min_{n':z^{n'}\neq z^n} g_{\boldsymbol{I}}(\boldsymbol{X}^n, \boldsymbol{X}^{n'})$;
8: end for
9: for $c = 1, \cdots, C$ do
10:     $\boldsymbol{\Theta} = \{n | z^n = c, s_o^r(n) > s_o^a(n), s_m^r(n) > s_m^a(n)\}$;
11:     select $k$ indexes $n_1, \cdots, n_k$ from $\boldsymbol{\Theta}$ with the largest $k$ values of $s_o^r(n) - s_o^a(n)$, for $n \in \boldsymbol{\Theta}$;
12:     $\boldsymbol{\Psi}_c = \{\boldsymbol{X}^n | n = n_1, \cdots, n_k\}$;
13: end for
14: for $n = 1, \cdots, N$ do
15:     $\boldsymbol{V}^n = f(\boldsymbol{X}^n, z^n) = arg \min_{\boldsymbol{V}\in\boldsymbol{\Psi}_{z^n}} g_{\boldsymbol{I}}(\boldsymbol{X}^n, \boldsymbol{V})$;
16: end for
```

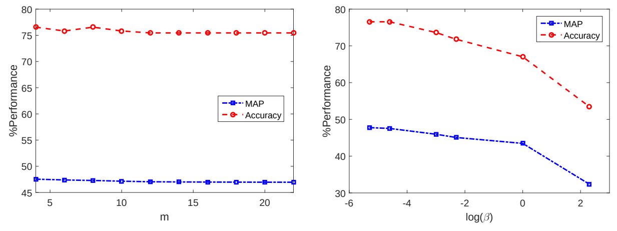

Figure 1. Performances of RVSML as functions of (a) m and (b) $log(\beta)$ on the MSR Action3D dataset.

sionality of elements in virtual sequences depends on m, the size of $\boldsymbol{W}$ increases with m. Therefore, the parameters in $\boldsymbol{W}$ may be too many to be adequately trained for large m. We then evaluate β by fixing m to 4. The performances as functions of m are illustrated in Fig. 1(b). Generally, as β is only a regularization coefficient, it seems that very small β leads to satisfactory results.

The generation of the LM-based virtual sequences has one hyper-parameter: k, the number of target sequences per class. For RVSML-LM, we evaluate k by fixing β to 0.005 and evaluate β by fixing k to 2. The performances as functions of the variables are shown in Fig. 2. We can observe that a small k works better and increasing k leads to performance degradation. When k is larger than the maximum number of training sequences among all classes, the virtual sequences for the training sequences that can be correctly classified with the Euclidean metric are themselves. As a result, the within-class variations may still be larger. When $k = 1$, the virtual sequence for the training sequences of the same class is the sequence that has the largest margin with other classes. If the distribution of the sequences is multi-modal, a linear transformation may not be able to move all sequences to a single center. Therefore, a small k coinciding with the number of modes is preferred. Again, only a very small β is enough to regularize the transformation. For the other two datasets, we fix β, m, and k to 0.0005, 4, and 1, respectively.

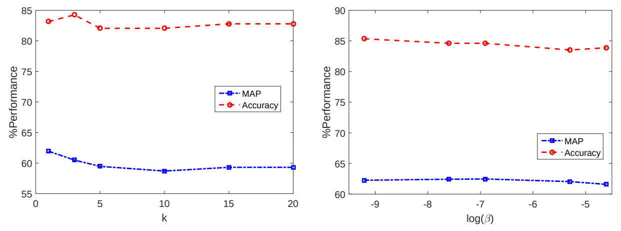

Figure 2. Performances of RVSML-LM as functions of (a) k and (b) $log(\beta)$ on the MSR Action3D dataset.

6. Evaluation of the LM-based virtual sequences on the MSR Action3D dataset

The comparisons of RVSML with the LM-based virtual sequences with other metric learning methods on the MSR Action3D dataset are presented in Tab. 1. Recall that RVSML-TS is equivalent to RVSML in the main text. We can observe that RVSML-LM performs best on this dataset instantiated by both the DTW distance and the OPW distance. With the OPW distance, RVSML-TS performs worse than some other metric learning methods. In this case, the original sequences can already be separated well, and forcibly projecting the elements to different unit vectors does not improve the performance. However, RVSML-TS outperforms RVML consistently. This indicates that exploring the temporal information for such projection still benefits. Due to the different properties and real distributions of the sequence data and different distance measures for sequences, different virtual sequences can lead to different effects on the learned metric. Selecting appropriate virtual sequences helps the RVSML framework to determine the metric that better captures the entire spatial-temporal separability of sequences from different classes.

7. Evaluation with another type of frame-wide features on the MSR Action3D dataset

We also employ another type of frame-wide features, i.e., the 60-dimensional motion features as used in (Lee et al.,

Table 1. Comparison of the proposed RVSML variants instantiated by (a) DTW and (b) OPW with other metric learning methods using the NN classifier with the (a) DTW and (b) OPW distance on the MSR Action3D dataset.

(a) DTW

Method	MAP	Accuracy
Ori (Su & Hua, 2018)	58.95	81.32
ITML (Davis et al., 2007)	59.19	80.95
LMNN (Weinberger & Saul, 2009)	54.14	80.95
SCML (Shi et al., 2014)	42.79	63.00
RVML (Perrot & Habrard, 2015)	57.41	80.95
LDMLT (Mei et al., 2014)	64.29	**84.98**
SWMD (Huang et al., 2016)	59.65	80.95
RVSML-TS	59.30	82.78
RVSML-LM	**64.97**	84.62

(b) OPW

Method	MAP	Accuracy
Ori (Su & Hua, 2018)	58.70	84.25
ITML (Davis et al., 2007)	59.48	83.52
LMNN (Weinberger & Saul, 2009)	32.73	82.42
SCML (Shi et al., 2014)	39.63	64.10
RVML (Perrot & Habrard, 2015)	44.58	73.63
LDMLT (Mei et al., 2014)	53.61	80.59
SWMD (Huang et al., 2016)	43.23	66.67
RVSML-TS	47.54	76.56
RVSML-LM	**62.25**	**85.35**

Table 2. Comparison of the proposed RVSML variants instantiated by (a) DTW and (b) OPW with other metric learning methods using the NN classifier with the (a) DTW and (b) OPW distance on the MSR Action3D dataset.

(a) DTW

Method	MAP	Accuracy
Ori (Su & Hua, 2018)	47.78	80.95
ITML (Davis et al., 2007)	48.91	80.22
LMNN (Weinberger & Saul, 2009)	54.05	80.59
RVML (Perrot & Habrard, 2015)	30.33	63.00
LDMLT (Mei et al., 2014)	52.11	81.32
SWMD (Huang et al., 2016)	46.83	80.22
RVSML	**59.89**	**86.08**

(b) OPW

Method	MAP	Accuracy
Ori (Su & Hua, 2018)	21.81	47.99
ITML (Davis et al., 2007)	17.50	40.66
LMNN (Weinberger & Saul, 2009)	**37.62**	**67.40**
RVML (Perrot & Habrard, 2015)	10.16	17.95
LDMLT (Mei et al., 2014)	21.57	48.35
SWMD (Huang et al., 2016)	17.98	41.76
RVSML	23.33	52.01

2017). The experimental setup remains the same as evaluating with the relative angle based features in the main text. The classification results in comparison with other metric learning methods are shown in Tab. 2. We can observe that RVSML instantiated by OPW obtains the second best results after LMNN by the 1-NN classifier with the OPW distance, while RVSML instantiated by DTW outperforms other metric learning methods significantly by a margin of about 5% by the 1-NN classifier with the DTW distance.

Table 4. Comparison of the training times.

Dataset	HAS
LDMLT (Mei et al., 2014)	10863.32 (247.0112)
SWMD (Huang et al., 2016)	1497.786 (65.3938)
RVSML(DTW)	212.3670 (32.0198)
RVSML(OPW)	150.2897 (8.7143)

8. Comparisons with other metric learning methods on the HAS dataset

“High-quality recordings of Australian Sign Language signs (HAS)” dataset (Kadous, 2002) consists of $2,565$ sequences from 95 Australian Sign Language sign types. The dataset is also included in the UCI Machine Learning Repository (Bache & Lichman, 2013). There are 27 sequence samples for each class. Each sequence is comprised of ordered 22-dimensional vectors. Following (Su et al., 2018), we split the sequences into five subsets and conduct experiments by five-fold cross validation. Each time four subsets are used for training and the remaining subset is used for testing. For the HAS dataset, the sequences have already been represented as a series of 22-dimensional feature vectors. The parameters λ_1, λ_2, and σ of OPW are fixed to 50, 0.1 and 1, respectively, as suggested in (Su & Hua, 2018).

The comparisons with other metric learning methods are presented in Tab. 3. The comparisons of training times on this dataset are presented in Tab. 4. In addition to the average performance measures, the standard deviations over different folds are shown in parentheses. We can observe that RVSML instantiated by DTW and OPW achieves slightly worse results than LDMLT, but outperforms other metric learning methods by the 1-NN classifier with the DTW and OPW distance, respectively. However, the training speeds of RVSMLs are more than five times faster than LDMLT.

Table 3. Comparison of the proposed RVSML instantiated by (a) DTW and (b) OPW with other metric learning methods using the NN classifier with the (a) DTW and (b) OPW distance on the HAS dataset.

(a) DTW

Method	MAP	Accuracy
Ori (Su & Hua, 2018)	48.87 (1.09)	86.95 (2.89)
ITML (Davis et al., 2007)	14.50 (1.58)	48.90 (3.69)
LMNN (Weinberger & Saul, 2009)	60.94 (1.08)	92.34 (1.88)
SCML (Shi et al., 2014)	45.85 (10.62)	80.82 (10.93)
RVML (Perrot & Habrard, 2015)	74.21 (1.45)	94.82 (2.07)
LDMLT (Mei et al., 2014)	**82.80** (1.28)	**96.60** (0.82)
SWMD (Huang et al., 2016)	47.16 (3.74)	85.05 (4.68)
RVSML	74.64 (1.47)	95.65 (2.01)

(b) OPW

Method	MAP	Accuracy
Ori[b] (Su & Hua, 2018)	43.29 (0.87)	81.09 (2.90)
ITML (Davis et al., 2007)	13.58 (1.38)	40.44 (2.24)
LMNN (Weinberger & Saul, 2009)	59.15 (1.03)	90.03 (2.23)
SCML (Shi et al., 2014)	39.44 (9.24)	74.47 (10.52)
RVML (Perrot & Habrard, 2015)	68.02 (1.34)	92.87 (2.47)
LDMLT (Mei et al., 2014)	**73.39** (0.87)	**95.00** (1.22)
SWMD (Huang et al., 2016)	41.99 (2.38)	79.22 (2.81)
RVSML	71.36 (1.05)	93.72 (2.45)

[b] In (Su & Hua, 2018), σ is set to 12 on this dataset. We fix σ to 1 on all datasets.

References

Bache, K. and Lichman, M. *UCI Machine Learning Repository*. http://archive.ics.uci.edu/ml, University of California, Irvine, School of Information and Computer Sciences, 2013.

Chen, L. and Ng, R. On the marriage of lp-norms and edit distance. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pp. 792–803. VLDB Endowment, 2004.

Cuturi, M. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in neural information processing systems*, pp. 2292–2300, 2013.

Davis, J. V., Kulis, B., Jain, P., Sra, S., and Dhillon, I. S. Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning*, pp. 209–216. ACM, 2007.

Flamary, R., Cuturi, M., Courty, N., and Rakotomamonjy, A. Wasserstein discriminant analysis. *Machine Learning*, 107(12), 2018.

Huang, G., Guo, C., Kusner, M. J., Sun, Y., Sha, F., and Weinberger, K. Q. Supervised word mover's distance. In *Advances in Neural Information Processing Systems*, pp. 4862–4870, 2016.

Kadous, M. W. *Temporal classification: extending the classification paradigm to multivariate time series*. PhD Thesis (draft), School of Computer Science and Engineering, University of New South Wales, 2002.

Lee, I., Kim, D., Kang, S., and Lee, S. Ensemble deep learning for skeleton-based action recognition using temporal sliding lstm networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 1012–1020. IEEE, 2017.

Marteau, P.-F. Time warp edit distance with stiffness adjustment for time series matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2): 306–318, 2009.

Mei, J., Liu, M., Karimi, H. R., and Gao, H. Logdet divergence-based metric learning with triplet constraints and its applications. *IEEE Transactions on Image Processing*, 23(11):4920–4931, 2014.

Perrot, M. and Habrard, A. Regressive virtual metric learning. In *Advances in Neural Information Processing Systems*, pp. 1810–1818, 2015.

Ratanamahatana, C. A. and Keogh, E. Making time-series classification more accurate using learned constraints. In *Proc. SIAM Int. Conf. Data Mining*, pp. 11–22. Lake Buena Vista, Florida, 2004.

Shi, Y., Bellet, A., and Sha, F. Sparse compositional metric learning. In *AAAI*, pp. 2078–2084, 2014.

Su, B. and Hua, G. Order-preserving optimal transport for distances between sequences. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

Su, B., Zhou, J., Ding, X., and Wu, Y. Unsupervised hierarchical dynamic parsing and encoding for action recognition. *IEEE Transactions on Image Processing*, 26(12): 5784–5799, 2017.

Su, B., Ding, X., Wang, H., and Wu, Y. Discriminative dimensionality reduction for multi-dimensional sequences. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(1):77–91, 2018.

Thorpe, M., Park, S., Kolouri, S., Rohde, G. K., and Slepčev, D. A transportation l^ p distance for signal analysis. *Journal of Mathematical Imaging and Vision*, 59(2):187–210, 2017.

Villani, C. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

Weinberger, K. Q. and Saul, L. K. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(Feb):207–244, 2009.