Learning Euclidean-to-Riemannian Metric for Point-to-Set Classification

Zhiwu Huang[1,2], Ruiping Wang[1], Shiguang Shan[1], Xilin Chen[1]
[1]Key Laboratory of Intelligent Information Processing of Chinese Academy of Sciences (CAS),
Institute of Computing Technology, CAS, Beijing, 100190, China
[2]University of Chinese Academy of Sciences, Beijing, 100049, China
zhiwu.huang@vipl.ict.ac.cn, {wangruiping, sgshan, xlchen}@ict.ac.cn

Abstract

In this paper, we focus on the problem of point-to-set classification, where single points are matched against sets of correlated points. Since the points commonly lie in Euclidean space while the sets are typically modeled as elements on Riemannian manifold, they can be treated as Euclidean points and Riemannian points respectively. To learn a metric between the heterogeneous points, we propose a novel Euclidean-to-Riemannian metric learning framework. Specifically, by exploiting typical Riemannian metrics, the Riemannian manifold is first embedded into a high dimensional Hilbert space to reduce the gaps between the heterogeneous spaces and meanwhile respect the Riemannian geometry of the manifold. The final distance metric is then learned by pursuing multiple transformations from the Hilbert space and the original Euclidean space (or its corresponding Hilbert space) to a common Euclidean subspace, where classical Euclidean distances of transformed heterogeneous points can be measured. Extensive experiments clearly demonstrate the superiority of our proposed approach over the state-of-the-art methods.

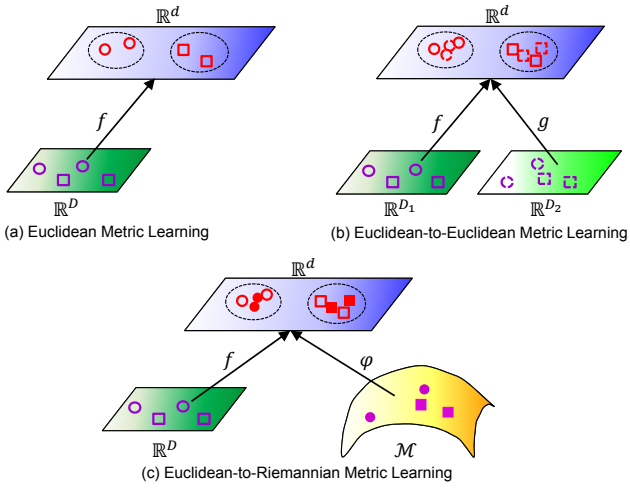

Figure 1. Illustration of conventional *Euclidean Metric Learning*, *Euclidean-to-Euclidean Metric Learning* and our proposed *Euclidean-to-Riemannian Metric Learning*. $\mathbb{R}^D/\mathbb{R}^{D_1}/\mathbb{R}^{D_2}$, $\mathcal{M}$ and $\mathbb{R}^d$ indicate Euclidean space, Riemannian manifold and common subspace respectively. f, g, φ denote the transformations, and different shapes (i.e., circles and rectangles) represent classes.

1. Introduction

In computer vision and pattern recognition field, objects (e.g., images) are often described in terms of feature vectors, each of which can be considered as a point lying in Euclidean space $\mathbb{R}^D$. Designed for point-to-point classification, a large number of metric learning methods [6, 5, 24, 15, 27, 25] make great efforts to learn an appropriate point-to-point distance metric in Euclidean space(s). In recent years, increasing approaches [4, 22, 3, 28] are proposed to define or learn a point-to-set distance metric for point-to-set classification, which is performed to match single points against sets of correlated points.

By treating sets as subspace-based models, Nearest Feature Subspace (NFS) [4], K-local Hyperplane (Convex) Distance Nearest Neighbor (HKNN, CKNN) [22] and Nearest Hyperdisk (NHD) [3] classifiers have attempted to measure point-to-set distance. More recently, Point-to-Set Distance Metric Learning (PSDML) [28] is proposed to learn a proper metric between pairs of single points in Euclidean space to get more accurate point-to-set distance for classification. However, few of these methods consider the non-Euclidean geometry of the space of their subspace-based set models, which typically reside on certain Riemannian manifolds [8, 7, 23]. As is well known, ignoring the underlying data structure and distributions in the learning stage will probably lead to undesirable metrics.

In light of the success of modeling sets as points on specific Riemannian manifolds [8, 7, 23], we propose to learn a distance metric between points in Euclidean space $\mathbb{R}^D$ and points on Riemannian manifold $\mathcal{M}$ for point-to-set classification. As illustrated in Fig.1(c), the new distance metric is

1063-6919/14 $31.00 © 2014 IEEE
DOI 10.1109/CVPR.2014.217

learned by seeking different transformations f, φ from the heterogeneous spaces $\mathbb{R}^D$ and $\mathcal{M}$ to a common Euclidean subspace $\mathbb{R}^d$. However, the gaps of $\mathbb{R}^D$ and $\mathcal{M}$ is so large that learning the mappings on them is not trivial. Therefore, we design a unified framework to first embed the manifold $\mathcal{M}$ into a high dimensional Hilbert space where Euclidean geometry applies. During the Hilbert space embedding, we also encode the Riemannian geometry of $\mathcal{M}$ by exploiting typical Riemannian metrics [8, 7, 1]. Finally, we learn the distance metric on the Hilbert space and the Euclidean space $\mathbb{R}^D$ (or its corresponding Hilbert space). To our best knowledge, this paper makes the first attempt to propose a *Euclidean-to-Riemannian Metric Learning* framework for the problem of the point-to-set classification.

2. Metric Learning Methods

In metric learning, the learned distance metric is usually defined as Mahalanobis distance, which is the squared Euclidean distance after applying the learned linear transformation(s) on original Euclidean space(s). According to the number of the original spaces, traditional metric learning methods can be categorized into *Euclidean metric learning* and *Euclidean-to-Euclidean metric learning*.

As shown in Fig.1(a), the Euclidean metric learning methods [6, 5, 20, 24, 28] intend to learn a metric or a transformation f of object features from an original Euclidean space $\mathbb{R}^D$ to a new one $\mathbb{R}^d$. For example, Davis *et al*. [5] presented Information Theoretic Metric Learning (ITML) by adopting an information-theoretic formulation to learn a metric on one original Euclidean space. Weinberger *et al*. [24] proposed Large Margin Nearest Neighbor (LMNN) to learn a transformation from one Euclidean space to a new one for K-Nearest Neighbor (KNN) by pulling neighboring objects of the same class closer and pushing others further. Recently, Zhu *et al*. [28] developed Point-to-Set Distance Metric Learning (PSDML) to learn a metric between points in a single Euclidean space for more appropriate point-to-set distance to apply in point-to-set classification task.

In contrast, as depicted in Fig.1(b), Euclidean-to-Euclidean metric learning methods [2, 17, 15, 27, 25, 14] are designed to learn a metric or multiple transformations f, g mapping object features from multiple original Euclidean spaces, say $\mathbb{R}^{D_1}$, $\mathbb{R}^{D_2}$, to a common subspace $\mathbb{R}^d$. For instance, Quadrianto *et al*. [17] proposed a metric learning method to seek multiple projections with neighborhood preserving constraint for multi-view data in multiple original Euclidean spaces. McFee and Lanckriet [15] applied multiple kernel/metric learning technique to integrate different object features from multiple Euclidean spaces to a unified Euclidean space. Zhai *et al*. [27] developed a method to learn two transformations of cross-view objects from two different Euclidean spaces to a common subspace by integrating them into a joint graph regularization.

3. Background

In this section, we review the Riemannian metrics for several Riemannian manifolds and introduce the existing Euclidean-to-Riemannian metrics in previous literatures.

3.1. Riemannian metric

In computer vision, sets (of images) are typically modeled as linear subspaces [26, 4, 8], affine subspaces [22, 7, 28] or covariance matrices [23, 21], which lie on specific Riemannian manifolds, namely Grassmannian manifold, Affine Grassmannian manifold and Symmetric Positive Definite (SPD) matrix manifold respectively. Such different Riemannian manifolds are often accompanied with different Riemannian metrics as follows.

Grassmannian metric: A Grassmannian manifold $\mathcal{G}(D^{'}, D)$ is defined as the space of $D^{'}$-dimensional linear subspaces of Euclidean space $\mathbb{R}^D$ [8]. An element of $\mathcal{G}(D^{'}, D)$ is represented by an orthonormal matrix $\boldsymbol{U} \in \mathbb{R}^{D\times D^{'}}$, whose column vectors span the subspace. For two points $\boldsymbol{U}_i$, $\boldsymbol{U}_j$ on the manifold $\mathcal{G}(D^{'}, D)$, their distance can be measured by one of the most popular metrics namely Projection metric [8]:

$$d(\boldsymbol{U}_i, \boldsymbol{U}_j) = 2^{-1/2}\|\boldsymbol{U}_i\boldsymbol{U}_i^T - \boldsymbol{U}_j\boldsymbol{U}_j^T\|_F. \tag{1}$$

Affine Grassmannian metric: An Affine Grassmannian manifold $\mathcal{AG}(D^{'}, D)$ is the space of $D^{'}$-dimensional affine subspaces of Euclidean space $\mathbb{R}^D$, which is simply a linear subspace with an offset [7]. Therefore, each point on $\mathcal{AG}(D^{'}, D)$ is denoted by an affine span $\boldsymbol{A}$ with an orthonormal matrix $\boldsymbol{U} \in \mathbb{R}^{D\times D^{'}}$ adding the offset $\boldsymbol{\mu} \in \mathbb{R}^D$ (i.e., the mean of one set of Euclidean points) from the origin. In this case, the distance of two points $\boldsymbol{A}_i$ and $\boldsymbol{A}_j$ on the manifold $\mathcal{AG}(D^{'}, D)$ is measured by [7]:

$$\begin{aligned} d(\boldsymbol{A}_i, \boldsymbol{A}_j) = 2^{-1/2}(&\|\boldsymbol{U}_i\boldsymbol{U}_i^T - \boldsymbol{U}_j\boldsymbol{U}_j^T\|_F \\ &+ \|(\boldsymbol{I} - \boldsymbol{U}_i\boldsymbol{U}_i^T)\boldsymbol{\mu}_i - (\boldsymbol{I} - \boldsymbol{U}_j\boldsymbol{U}_j^T)\boldsymbol{\mu}_j\|_F). \end{aligned} \tag{2}$$

where $\boldsymbol{I} \in \mathbb{R}^{D\times D}$ is the identity matrix.

Symmetric Positive Definite metric: The space of $(D \times D)$-dimensional Symmetric Positive Definite (SPD) matrices is mostly studied when endowed with a Riemannian metric and thus forms a Riemannian manifold Sym_D^+ [16, 1]. On the manifold Sym_D^+, the distance between two points $\boldsymbol{C}_i$, $\boldsymbol{C}_j \in \mathbb{R}^{D\times D}$ can be measured by the Log-Euclidean Distance (LED) [1], which is a true geodesic distance and expressed by Euclidean computations in the domain of matrix logarithms:

$$d(\boldsymbol{C}_i, \boldsymbol{C}_j) = \|log(\boldsymbol{C}_i) - log(\boldsymbol{C}_j)\|_F. \tag{3}$$

3.2. Euclidean-to-Riemannian metric

For point-to-set classification, several works [4, 22, 3] represented the sets as subspace-based models to define

various point-to-set distances, which can be regarded as Euclidean-to-Riemannian metrics by treating each point x_i in Euclidean space $\mathbb{R}^D$ and modeling each set model as one element (with the same notation as above) on a specific Riemannian manifold $\mathcal{M}$.

Euclean-to-Grassmannian metric: In Nearest Feature Subspace (NFS) classifier [4], the sets are modeled as linear subspaces, which lie on Grassmannian manifold. The NFS calculates the distance between Euclidean point $\boldsymbol{x}_i$ and the linear subspace representation $\boldsymbol{U}_j$ in the following:

$$d(\boldsymbol{x}_i, \boldsymbol{U}_j) = \|\boldsymbol{x}_i - \boldsymbol{U}_j \boldsymbol{U}_j^T \boldsymbol{x}_i\|_F. \tag{4}$$

Euclean-to-AffineGrassmannian metric: K-local Hyperplane Distance Nearest Neighbor (HKNN) algorithm [22] models the sets as affine subspaces lying on Affine Grassmannian manifold. It defines the distance between the Euclidean point $\boldsymbol{x}_i$ and the affine subspace $\boldsymbol{A}_j$ as follows:

$$d(\boldsymbol{x}_i, \boldsymbol{A}_j) = \min_{\alpha} \|(\boldsymbol{U}_j \alpha + \boldsymbol{\mu}_j) - \boldsymbol{x}_i\|_2. \tag{5}$$

where α is a vector of coordinates for the points within $\boldsymbol{A}_j$.

Euclidean-to-SPD metric: Classical Mahalanobis Distance (MD) can be used to define the distance between the Euclidean point $\boldsymbol{x}_i$ and the covariance matrix $\boldsymbol{C}_j$, which actually is a SPD matrix and thus residing on SPD manifold:

$$d(\boldsymbol{x}_i, \boldsymbol{C}_j) = \sqrt{(\boldsymbol{x}_i - \boldsymbol{\mu}_j)^T \boldsymbol{C}_j^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu}_j)}. \tag{6}$$

where $\boldsymbol{\mu}_j$ is the mean of the samples in the set.

4. Learning Euclidean-to-Riemannian Metric

In this section, we first present our proposed framework of Learning Euclidean-to-Riemannian Metric (LERM) for point-to-set classification and then introduce its implementations in three Euclidean-to-Riemannian cases. Lastly, we will also provide a brief discussion on the relationship of our framework with other traditional learning schemes.

4.1. Problem formulation

As described in Sec.3, with the usual representations for the points and the sets, the task of point-to-set classification is formulated as the problem of matching Euclidean points with Riemannian points. In this paper, we denote a collection of Euclidean points by $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\} \subset \mathbb{R}^D$ with class labels $\{l_1^x, l_2^x, \ldots, l_m^x\}$ and a collection of Riemannian points by $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n\} \subset \mathcal{M}$ with class labels $\{l_1^y, l_2^y, \ldots, l_n^y\}$, where $\mathbf{y}_j$ may come in the form of $\boldsymbol{U}_j, \boldsymbol{A}_j, \boldsymbol{C}_j$ as denoted in Sec.3, depending on the type of the corresponding Riemannian manifold.

Given one pair of Euclidean point $\boldsymbol{x}_i$ and Riemannian point $\mathbf{y}_j$, we use $d(\boldsymbol{x}_i, \mathbf{y}_j)$ to represent the distance between them. To achieve an appropriate distance metric between the two heterogeneous points, we need to learn two transformation functions f and φ, which respectively map the Euclidean points and Riemannian points to a common subspace. The learned distance metric reduces to classicial Euclidean distance in the common subspace as:

$$d(\boldsymbol{x}_i, \mathbf{y}_j) = \sqrt{(f(\boldsymbol{x}_i) - \varphi(\mathbf{y}_j))^T (f(\boldsymbol{x}_i) - \varphi(\mathbf{y}_j))}. \tag{7}$$

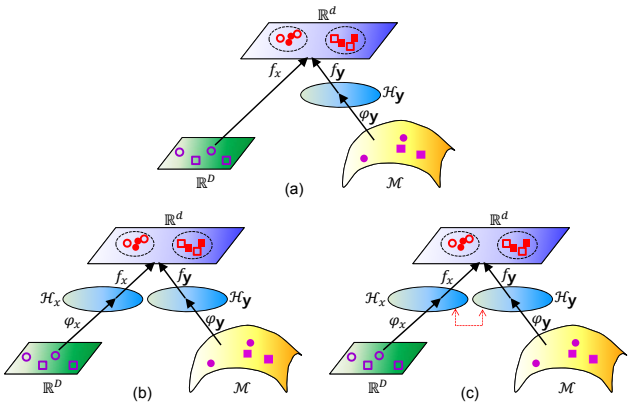

Figure 2. Three Mapping Modes. $\mathbb{R}^D$, $\mathcal{M}$, $\mathcal{H}_x/\mathcal{H}_y$, $\mathbb{R}^d$ represent Euclidean space, Riemannian manifold, Hilbert space and common subspace. f_x/f_y, φ_x/φ_y denote linear and nonlinear transformation functions, different shapes represent classes.

However, the gaps of Euclidean space and Riemannian manifold is too large to embed them into the common subspace straightforwardly. To deal with this problem, we first embed the Riemannian manifold into a high dimensional Hilbert space, which adheres to Euclidean geometry and thus reduces the heterogeneity between the original spaces. In addition, by exploiting typical Riemannian metrics [8, 7, 1], this Hilbert space embedding can account for the geometry of the Riemannian manifold [11]. After the embedding, multiple mappings are learned from the Hilbert space and the original Euclidean space (or its corresponding Hilbert space) to the final common subspace. Accordingly, we design three Mapping Modes to embed the Hilbert space(s) and learn the mappings in different ways.

Mapping Mode I: As shown in Fig.2(a), this mode first maps the Riemannian points to a Hilbert space $\mathcal{H}_y$ by a nonlinear transformation function $\varphi_y : \mathcal{M} \to \mathcal{H}_y$. In general, since the form of φ_y is usually implicit, we exploit kernel functions based on Riemannian metrics [8, 7, 1] to define its inner product. The transformed Riemannian points in the Hilbert space $\mathcal{H}_y$ and the Euclidean points in original Euclidean space $\mathbb{R}^D$ can be further projected into a common subspace $\mathbb{R}^d$ by two linear projection functions f_x and f_y. Consequently, the final mappings for Euclidean points and Riemannian points are $f = f_x$ and $\varphi = f_y \circ \varphi_y$.

Mapping Mode II: As illustrated in Fig.2(b), the second mode implicitly maps the Euclidean points by a implic-

it mapping $\varphi_x : \mathbb{R}^D \rightarrow \mathcal{H}_x$ to one Hilbert space $\mathcal{H}_x$ for obtaining richer representations of the original data. Additionally, this mode also transforms the Remiannian points to another Hilbert space $\mathcal{H}_{\mathsf{y}}$. The heterogeneous points are then projected from the two different Hilbert spaces $\mathcal{H}_x, \mathcal{H}_{\mathsf{y}}$ to a common subspace $\mathbb{R}^d$ by the final transformation functions $f = f_x \circ \varphi_x$ and $\varphi = f_{\mathsf{y}} \circ \varphi_{\mathsf{y}}$.

Mapping Mode III: As depicted in Fig.2(c), this mode simultaneously applies *single-space* and *cross-space* kernel functions to define the inner products of the implicit mappings φ_x and φ_{y} for Euclidean points and Riemannian points transforming to the corresponding Hilbert spaces $\mathcal{H}_x, \mathcal{H}_{\mathsf{y}}$. The *single-space* kernel functions are based on the classical Euclidean metric or Riemannian metrics [8, 7, 1], whereas the *cross-space* kernel functions are defined by the basic Euclidean-to-Riemannian metrics [4, 22]. With the final mappings $f = f_x \circ \varphi_x$ and $\varphi = f_{\mathsf{y}} \circ \varphi_{\mathsf{y}}$, the Euclidean and Riemannian points are transformed through the Hilbert spaces $\mathcal{H}_x, \mathcal{H}_{\mathsf{y}}$ to the final common subspace $\mathbb{R}^d$ for measuring their Euclidean distances.

4.2. Objective function

We formulate a objective function for our Euclidean-to-Riemannian metric learning framework as follows:

$$\min_{f,\varphi} \{D(f,\varphi) + \lambda_1 G(f,\varphi) + \lambda_2 T(f,\varphi)\} \tag{8}$$

where $D(f,\varphi)$ is the distance constraint item defined on the collections of similarity and dissimilarity constraints, $G(f,\varphi)$ is discriminant geometry constraint item and $T(f,\varphi)$ is transformation constraint item, which are regularizers defined on the target parameter functions f and φ, $\lambda_1 > 0, \lambda_2 > 0$ are the balancing parameters.

Distance constraint: This constraint item is defined in the way such that the distances between the Euclidean points and the Riemannian points with the similarity (dissimilarity) constraints are minimized (maximized). In this paper, we adopt classical expression of the sum of squared distances to define this item as:

$$\begin{aligned} D(f,\varphi) &= \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} \boldsymbol{Z}(i,j)\|f(\boldsymbol{x}_i) - \varphi(\mathbf{y}_j)\|^2, \\ \boldsymbol{Z}(i,j) &= \begin{cases} 1, & \text{if } l_i^x = l_j^{\mathsf{y}}, \\ -1, & \text{if } l_i^x \neq l_j^{\mathsf{y}}. \end{cases} \end{aligned} \tag{9}$$

where $\boldsymbol{Z}(i,j)$ indicates the heterogeneous points $\boldsymbol{x}_i$ and $\mathbf{y}_j$ are relevant or irrelevant inferred from the class label. To balance the effect of similarity constraints and dissimilarity ones, we normalize the elements of $\boldsymbol{Z}$ by averaging them with the total number of similar/dissimilar pairs.

Discriminant geometry constraint: This constraint item aims to preserve Euclidean geometry and Riemannian geometry respectively for the Euclidean and Riemannian points. Thus, it can be defined as: $G(f,\varphi) = G_x(f) + G_{\mathsf{y}}(\varphi)$, which are Euclidean and Riemannian geometry preserving items formulated as:

$$\begin{aligned} G_x(f) &= \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \boldsymbol{Z}_x(i,j)\|f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j)\|^2, \\ \boldsymbol{Z}_x(i,j) &= \begin{cases} d_{ij}, & \text{if } l_i^x = l_j^x \text{ and } k_1(i,j), \\ -d_{ij}, & \text{if } l_i^x \neq l_j^x \text{ and } k_2(i,j), \\ 0, & \text{else.} \end{cases} \end{aligned} \tag{10}$$

$$\begin{aligned} G_{\mathsf{y}}(\varphi) &= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \boldsymbol{Z}_{\mathsf{y}}(i,j)\|\varphi(\mathbf{y}_i) - \varphi(\mathbf{y}_j)\|^2, \\ \boldsymbol{Z}_{\mathsf{y}}(i,j) &= \begin{cases} d_{ij}, & \text{if } l_i^{\mathsf{y}} = l_j^{\mathsf{y}} \text{ and } k_1(i,j), \\ -d_{ij}, & \text{if } l_i^{\mathsf{y}} \neq l_j^{\mathsf{y}} \text{ and } k_2(i,j), \\ 0, & \text{else.} \end{cases} \end{aligned} \tag{11}$$

where $d_{ij} = exp(\|\boldsymbol{z}_i - \boldsymbol{z}_j\|^2/\sigma^2)$, $\boldsymbol{z}$ indicates $\boldsymbol{x}$ or $\mathbf{y}$. $k_1(i,j)$/$k_2(i,j)$ means point i is in the k_1/k_2 neighborhood of point j or point j is in the k_1/k_2 neighborhood of point i.

Transformation constraint: Since Euclidean distance will be used in the target common subspace where all dimensions are uniformly treated, it is reasonable to require the feature vectors satisfy isotropic distribution. Thus, this constraint item can be expressed in terms of unit covariance:

$$T(f,\varphi) = \frac{1}{2}(\|f(\boldsymbol{X})\|_F^2 + \|\varphi(\mathbf{Y})\|_F^2). \tag{12}$$

4.3. Iterative optimization

We propose an iterative method to minimize the objective function in Eq.8. According to the proposed Mapping Mode I,II,III, the learned transformation functions are $f = f_x(\circ\varphi_x)$ and $\varphi = f_{\mathsf{y}} \circ \varphi_{\mathsf{y}}$, where the linear projections $f_x(\boldsymbol{x}_i) = \boldsymbol{V}_x^T\boldsymbol{x}_i$ and $f_{\mathsf{y}}(\mathbf{y}_j) = \boldsymbol{V}_{\mathsf{y}}^T\mathbf{y}_j$. Without loss of generality, we here just consider the case $f = f_x \circ \varphi_x$ for simplicity. By taking inner products with a parameter in the Hilbert spaces, we find $f(\boldsymbol{x}_i) = \boldsymbol{V}_x^T\varphi_x(\boldsymbol{x}_i) = \sum_j \boldsymbol{W}_x^T\varphi_x(\boldsymbol{x}_j)\varphi_x(\boldsymbol{x}_i) = \boldsymbol{W}_x^T\boldsymbol{K}_{x\cdot i}$, where $\boldsymbol{K}_x(\boldsymbol{x}_i,\boldsymbol{x}_j) = \langle\varphi_x(\boldsymbol{x}_i),\varphi_x(\boldsymbol{x}_j)\rangle$. Employing this kernel trick, the function φ can also be reduced to $\varphi(\mathbf{y}_j) = \boldsymbol{W}_{\mathsf{y}}^T\boldsymbol{K}_{\mathsf{y}\cdot j}$, where $\boldsymbol{K}_{\mathsf{y}}(\mathbf{y}_i,\mathbf{y}_j) = \langle\varphi_{\mathsf{y}}(\mathbf{y}_i),\varphi_{\mathsf{y}}(\mathbf{y}_j)\rangle$. Then, Eq.9, Eq.10 and Eq.11 can be re-written in matrix formulations as:

$$\begin{aligned} D(\boldsymbol{W}_x,\boldsymbol{W}_{\mathsf{y}}) = \frac{1}{2}(&\boldsymbol{W}_x^T\boldsymbol{K}_x\boldsymbol{B}_x'\boldsymbol{K}_x^T\boldsymbol{W}_x + \boldsymbol{W}_{\mathsf{y}}^T\boldsymbol{K}_{\mathsf{y}}\boldsymbol{B}_{\mathsf{y}}'\boldsymbol{K}_{\mathsf{y}}^T\boldsymbol{W}_{\mathsf{y}} \\ &- 2\boldsymbol{W}_x^T\boldsymbol{K}_x\boldsymbol{Z}\boldsymbol{K}_{\mathsf{y}}\boldsymbol{W}_{\mathsf{y}}^T). \end{aligned} \tag{13}$$

$$\begin{aligned} G_x(\boldsymbol{W}_x) &= \boldsymbol{W}_x^T\boldsymbol{K}_x\boldsymbol{B}_x\boldsymbol{K}_x^T\boldsymbol{W}_x - \boldsymbol{W}_x^T\boldsymbol{K}_x\boldsymbol{Z}_x\boldsymbol{K}_x^T\boldsymbol{W}_x \\ &= \boldsymbol{W}_x^T\boldsymbol{K}_x\boldsymbol{L}_x\boldsymbol{K}_x^T\boldsymbol{W}_x. \end{aligned} \tag{14}$$

$$\begin{aligned} G_{\mathsf{y}}(\boldsymbol{W}_{\mathsf{y}}) &= \boldsymbol{W}_{\mathsf{y}}^T\boldsymbol{K}_{\mathsf{y}}\boldsymbol{B}_{\mathsf{y}}\boldsymbol{K}_{\mathsf{y}}^T\boldsymbol{W}_{\mathsf{y}} - \boldsymbol{W}_{\mathsf{y}}^T\boldsymbol{K}_{\mathsf{y}}\boldsymbol{Z}_{\mathsf{y}}\boldsymbol{K}_{\mathsf{y}}^T\boldsymbol{W}_{\mathsf{y}} \\ &= \boldsymbol{W}_{\mathsf{y}}^T\boldsymbol{K}_{\mathsf{y}}\boldsymbol{L}_{\mathsf{y}}\boldsymbol{K}_{\mathsf{y}}^T\boldsymbol{W}_{\mathsf{y}}. \end{aligned} \tag{15}$$

where $\boldsymbol{B}_x^{'}$, $\boldsymbol{B}_{\mathsf{y}}^{'}$, $\boldsymbol{B}_x$ and $\boldsymbol{B}_{\mathsf{y}}$ are diagonal matrices with $\boldsymbol{B}_x^{'}(i,i) = \sum_{j=1}^{n} \boldsymbol{Z}(i,j)$, $\boldsymbol{B}_{\mathsf{y}}^{'}(j,j) = \sum_{i=1}^{m} \boldsymbol{Z}(i,j)$, $\boldsymbol{B}_x(i,i) = \sum_{j=1}^{m} \boldsymbol{Z}_x(i,j)$, $\boldsymbol{B}_{\mathsf{y}}(i,i) = \sum_{j=1}^{n} \boldsymbol{Z}_{\mathsf{y}}(i,j)$.

Initialization. We define the within-class and between-class templates for $\boldsymbol{Z}$, $\boldsymbol{Z}_x$ and $\boldsymbol{Z}_{\mathsf{y}}$. Take $\boldsymbol{Z}$ for example, the within-class and between-class templates are formulated as:

$$\boldsymbol{Z}^w(i,j) = \begin{cases} 1, & \text{if } l_i^x = l_j^{\mathsf{y}}, \\ 0, & \text{if } l_i^x \neq l_j^{\mathsf{y}}. \end{cases}, \boldsymbol{Z}^b(i,j) = \begin{cases} 0, & \text{if } l_i^x = l_j^{\mathsf{y}}, \\ 1, & \text{if } l_i^x \neq l_j^{\mathsf{y}}. \end{cases} \tag{16}$$

Replacing $\boldsymbol{Z}$ with $\boldsymbol{Z}^w$ and $\boldsymbol{Z}^b$ in Eq.13, we get the within-class and the between-class templates for $D(\boldsymbol{W}_x, \boldsymbol{W}_{\mathsf{y}})$ denoted by $D^w(\boldsymbol{W}_x, \boldsymbol{W}_{\mathsf{y}})$ and $D^b(\boldsymbol{W}_x, \boldsymbol{W}_{\mathsf{y}})$.

Likewise, we get the within and between-class templates for G_x and G_{y} in Eq.14 and Eq.15 respectively denoted by G_x^w, G_x^b, G_{y}^w, G_{y}^b. Then we can initialize $\boldsymbol{W}_x$ and $\boldsymbol{W}_{\mathsf{y}}$ by maximizing the sum of the between-class templates while minimizing the sum of the within-class templates as:

$$\begin{aligned} &\max_{\boldsymbol{W}_x, \boldsymbol{W}_{\mathsf{y}}} \{\boldsymbol{D}^b(\boldsymbol{W}_x, \boldsymbol{W}_{\mathsf{y}}) + \lambda_1 G^b(\boldsymbol{W}_x, \boldsymbol{W}_{\mathsf{y}})\}, \\ &s.t. \quad \boldsymbol{D}^w(\boldsymbol{W}_x, \boldsymbol{W}_{\mathsf{y}}) + \lambda_1 G^w(\boldsymbol{W}_x, \boldsymbol{W}_{\mathsf{y}}) = 1. \end{aligned} \tag{17}$$

where $G^b = G_x^b + G_{\mathsf{y}}^b$ and $G^w = G_x^w + G_{\mathsf{y}}^w$. Then, it can reach the optimization by a standard generalized eigenvalue problem that can be solved using any eigensolver[1].

Fix $\boldsymbol{W}_{\mathsf{y}}$ to update $\boldsymbol{W}_x$. Let $Q(\boldsymbol{W}_x, \boldsymbol{W}_{\mathsf{y}})$ denote the objective function in Eq.8. Differentiating $Q(\boldsymbol{W}_x, \boldsymbol{W}_{\mathsf{y}})$ in Eq.8 (combined with Eq.13, 14, 15) w.r.t.$\boldsymbol{W}_x$ and setting it to zero, we have the following equation:

$$\begin{aligned} \frac{\partial Q(\boldsymbol{W}_x, \boldsymbol{W}_{\mathsf{y}})}{\partial \boldsymbol{W}_x} &= \boldsymbol{K}_x \boldsymbol{B}_x^{'} \boldsymbol{K}_x^T \boldsymbol{W}_x - \boldsymbol{K}_x \boldsymbol{Z} \boldsymbol{K}_{\mathsf{y}}^T \boldsymbol{W}_{\mathsf{y}} \\ &+ 2\lambda_1 \boldsymbol{K}_x \boldsymbol{L}_x \boldsymbol{K}_x^T \boldsymbol{W}_x + 2\lambda_2 \boldsymbol{K}_x \boldsymbol{K}_x^T \boldsymbol{W}_x = 0. \end{aligned} \tag{18}$$

Then we obtain the analytical solution as follows:

$$\boldsymbol{W}_x = (\boldsymbol{K}_x(\boldsymbol{B}_x^{'} + 2\lambda_1 \boldsymbol{L}_x + 2\lambda_2 \boldsymbol{I})\boldsymbol{K}_x^T)^{-1} \boldsymbol{K}_x \boldsymbol{Z} \boldsymbol{K}_{\mathsf{y}}^T \boldsymbol{W}_{\mathsf{y}}. \tag{19}$$

Fix $\boldsymbol{W}_x$ to update $\boldsymbol{W}_{\mathsf{y}}$. Similarly, by differentiating $Q(\boldsymbol{W}_x, \boldsymbol{W}_{\mathsf{y}})$ in Eq.8 (with Eq.13, 14, 15) w.r.t.$\boldsymbol{W}_x$ and setting it to zero, we obtain the solution:

$$\boldsymbol{W}_{\mathsf{y}} = (\boldsymbol{K}_{\mathsf{y}}(\boldsymbol{B}_{\mathsf{y}}^{'} + 2\lambda_1 \boldsymbol{L}_{\mathsf{y}} + 2\lambda_2 \boldsymbol{I})\boldsymbol{K}_{\mathsf{y}}^T)^{-1} \boldsymbol{K}_{\mathsf{y}} \boldsymbol{Z} \boldsymbol{K}_x^T \boldsymbol{W}_x. \tag{20}$$

We alternate the above updates of $\boldsymbol{W}_x$ and $\boldsymbol{W}_{\mathsf{y}}$ for several iterations to search an optimal solution. Although providing a theoretical proof of uniqueness or convergence of the proposed iterative optimization is hard, we found it can make our objective function Eq.8 converge to a stable and desirable solution after tens of iterations. As for more detailed experimental results with quantitative analysis, we refer the reader to the Supplementary Material.

[1]Please kindly refer to our Supplementary Material for more details.

4.4. Implementations for different Euclidean-to-Riemannian cases

According to which type of manifold set models lie on, it generates Euclidean-to-Grassmannian (**EG**), Euclidean-to-AffineGrassmannian (**EA**) and Euclidean-to-SPD (**ES**) cases. In all cases, as mentioned in Sec.4.1, the mapping of Mapping Mode I for the Euclidean points is linear while the one of Mapping Mode II is non-linear transformation, whose inner product can be defined by the Gaussian Radial Basis Function (RBF), i.e., $\boldsymbol{K}_x^{'}(i,j) = exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/2\sigma^2)$. Inspired by [11], to define positive definite kernels on manifolds, we replace the Euclidean distance by specific Riemannian metric or Euclidean-to-Riemannian metric on given cases. Specifically, let $\boldsymbol{K}_{\mathsf{y}}^{'}$ and $\boldsymbol{K}_{x\mathsf{y}}^{'}$ denote the Riemannian metric based kernel and the Euclidean-to-Riemannian metric based kernel respectively. The final kernels $\boldsymbol{K}_{\mathsf{y}} = \boldsymbol{K}_{\mathsf{y}}^{'}$ in Mapping Mode I and II while $\boldsymbol{K}_x = \boldsymbol{K}_x^{'}$ in Mapping Mode II. For Mapping Mode III, the final kernels $\boldsymbol{K}_x = [\boldsymbol{K}_x^{'}, \boldsymbol{K}_{x\mathsf{y}}^{'}]$, $\boldsymbol{K}_{\mathsf{y}} = [\boldsymbol{K}_{\mathsf{y}}^{'}, (\boldsymbol{K}_{x\mathsf{y}}^{'})^T]$.

Take the **ES** case for example, we employ the Riemannian metric in Eq.3 to define a Gaussian kernel for points on SPD manifold as following:

$$\boldsymbol{K}_{\mathsf{y}}^{'}(i,j) = exp(-d^2(\boldsymbol{C}_i, \boldsymbol{C}_j)/2\sigma^2) \tag{21}$$

In Mapping Mode III, for both Euclidean and Riemannian points, we also define a Euclidean-to-SPD Gaussian kernel by using the traditional Mahalanobis Distance in Eq.6:

$$\boldsymbol{K}_{x\mathsf{y}}^{'}(i,j) = exp(-d^2(\boldsymbol{x}_i, \boldsymbol{C}_j)/2\sigma^2) \tag{22}$$

Likewise, for the **EG** case and the **EA** case, we can define their Riemannian metric based kernels and Euclidean-to-Riemannian metric based kernels by exploiting corresponding metrics in Eq.1, Eq.2, Eq.4 and Eq.5 respectively.

4.5. Discussion

Generally speaking, our metric learning framework can be viewed as a two-stage procedure, which firstly embeds the heterogeneous spaces into Hilbert spaces, and then learns the multiple transformations from the Hilbert spaces to a common subspace. Hence, existing Kernelized Multi-View Learning (KML) methods, e.g., Kernel Partial Least Squares (KPLS) [18], Kernel Canonical Correlation Analysis (KCCA) [9] and Kernel Generalized Multiview Analysis (KGMA) [19], can be easily used as alternatives in the second stage. Compared with our method, such KML methods commonly work on a couple of single-space kernels in traditional way, and thus can only couple with our proposed Mapping Mode II. Besides, most of them only take account of positive pairs, while ours takes both positive and negative pairs as constraints. Furthermore, the KML methods are not designed to learn an optimal distance metric but for some other objectives such as maximum correlations or covariances in the common subspace.

Table 1. Image-to-set object classification results (%) on ETH-I2S.

methods	Image-Set	Set-Image
NNC	60.83 ± 5.54	77.50 ± 7.72
NCA [6]	70.83 ± 4.39	77.29 ± 4.95
ITML [5]	66.04 ± 4.72	77.29 ± 5.68
LFDA [20]	64.79 ± 4.55	78.75 ± 3.78
LMNN [24]	68.96 ± 4.44	80.00 ± 5.74
NFS [4]	45.00 ± 5.91	77.08 ± 5.29
HKNN [22]	48.75 ± 4.52	66.25 ± 9.76
CKNN [22]	55.21 ± 2.25	77.08 ± 8.10
MD	57.29 ± 3.44	62.71 ± 10.51
PSDML [28]	64.17 ± 7.78	81.25 ± 6.51
KPLS[18]-**ES2**	74.58 ± 5.88	81.25 ± 5.97
KCCA[9]-**ES2**	83.54 ± 4.76	92.08 ± 8.21
KGMA[19]-**ES2**	87.50 ± 1.39	**100.00 ± 0.00**
LERM-ES1	**91.25 ± 2.91**	**100.00 ± 0.00**
LERM-ES2	**91.88 ± 1.18**	**100.00 ± 0.00**
LERM-ES3	89.17 ± 2.74	**100.00 ± 0.00**

5. Experiments

In this section, we present extensive experiments to demonstrate the effectiveness of our proposed Leaning Euclidean-to-Riemannian Metric (LERM) method for two point-to-set classification tasks, including image-to-set object classification and still-to-video face recognition.

We compare LERM (implementing in three Euclidean-to-Riemannian cases **EG**, **EA**, **ES**) with Nearest Neighborhood Classifier (NNC), four state-of-the-art metric learning methods NCA [6], ITML [5], LFDA [20] and LMNN [24] and five point-to-set based methods NFS [4], HKNN [22], CKNN [22], Mahalanobis Distance (MD) and PSDML [28]. We also validate three kernelized multi-view methods KPLS [18], KCCA [9] and KGMA (i.e., KGML-DA) [19], which are coupled with our proposed Mapping Mode II. Except NFS, HKNN, CKNN, the source code of all above methods are obtained from the original authors.

For fair comparison, we use PCA to reduce the data dimension by preserving 95% of data energy for all methods. For NCA, the output dimension is equal to that of PCA. For ITML, the lower/upper bound distance are set to the mean distance minus/plus standard variance and its other parameters are set to default values. In LFDA, the neighborhood number k is set to 7. For LMNN, the number of neighborhood is set to 5, the output dimension is equal to that of PCA, the maximum iteration times is set to 500, the portion of training samples in validation is 30%. For HKNN, the regularization parameter λ is set to 50. For PSDML, we set its parameters $\nu = 1$, $\lambda = 0.8$. For KPLS, KCCA and KGMA, the parameters are tuned for the best results. For our method LERM, we set the parameters $\lambda_1 = 0.01, \lambda_2 = 0.1$, the neighborhood number $k_1 = 1, k_2 = 20$, all the kernel widths σ is specified from the mean of distances, and the number of iterations is set to 30.

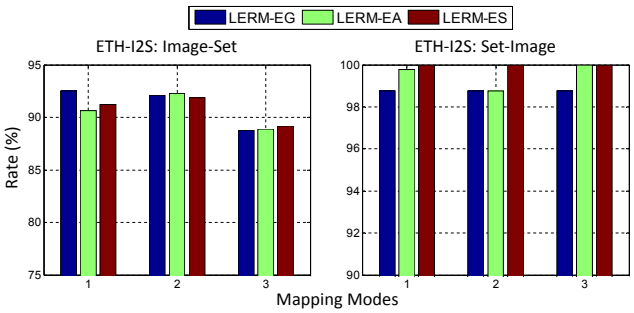

Figure 3. The comparisons of our method LERM in three cases (**EG, EA and ES**) with three Mapping Modes on ETH-I2S.

5.1. Image-to-set object classification

As a point-to-set classification task, image-to-set (I2S) object classification matches single images against sets of images from objects. We privately construct ETH-I2S based on ETH-80 [13] object data set, which contains images of 8 object categories. We resized the images to 64×64 and used their raw intensity as input features. On this dataset, we conducted ten cross validation experiments, i.e., 10 randomly selected gallery/probe combinations. We randomly chosen 24 image sets and 48 images as training data while using 48 images and 48 image sets for testing.

Tab.1 summarizes testing results in the I2S object classification scenario. In **image-set** classification, we enroll single images in gallery and image sets in probe for testing. In **set-image** classification, we identify the probe images with the gallery image sets. Each reported rate is an average plus a standard variance over the ten runs of cross validation. In the table, we list the results of our method LERM in **ES** case with three proposed Mapping Modes. These results show that our novel method LERM significantly outperform both the point-to-set based methods and the state-of-the-art metric learning methods. The results of kernelized multiview methods such as KPLS coupled with Mapping Mode II in **ES** case also validate our method is a good metric learning framework for exploiting multi-view kernels. Fig.3 illustrates the average matching rates of three Euclidean-to-Riemannian cases (i.e., **EG, EA, ES**) with different Mapping Modes of our method. From the comparison results, we can see that our method in various cases with three Mapping modes are comparable in this task.

5.2. Still-to-video face recognition

Still-to-video (S2V) face recognition matches still face images with sets of faces in videos, which can be considered as a point-to-set classification task. The YTC-S2V dataset is privately collected by us from the YouTube Celebrities [12], which has $1,910$ video clips of 47 subjects. Following [23], each face in videos was resized to a 20×20 image and preprocessed by histogram equalization. We also conducted

Table 2. Still-to-video face recognition results (rank-1 recognition rate (%)) on YTC-S2V and COX-S2V dataset.

Methods	YTC-S2V		COX-S2V					
	Still-Video	Video-Still	Still-Video1	Still-Video2	Still-Video3	Video1-Still	Video2-Still	Video3-Still
NNC	52.88 ± 2.33	69.15 ± 2.90	9.96 ± 0.61	7.14 ± 0.68	17.37 ± 6.16	7.60 ± 0.69	5.81 ± 0.42	26.37 ± 9.32
NCA [6]	51.74 ± 3.11	60.35 ± 3.09	39.14 ± 1.33	31.57 ± 1.56	**57.57 ± 2.03**	37.71 ± 1.45	32.14 ± 2.20	58.86 ± 1.87
ITML [5]	47.62 ± 1.73	59.72 ± 4.27	19.83 ± 1.62	18.20 ± 1.80	36.63 ± 2.69	26.66 ± 1.74	25.21 ± 2.10	47.57 ± 2.87
LFDA [20]	57.83 ± 2.67	73.12 ± 2.87	21.41 ± 1.77	22.17 ± 1.77	43.99 ± 1.74	40.54 ± 1.32	33.90 ± 1.40	61.40 ± 1.45
LMNN [24]	55.02 ± 2.71	70.99 ± 3.25	34.44 ± 1.02	30.03 ± 1.36	**58.06 ± 1.35**	37.84 ± 1.79	35.77 ± 1.37	**63.33 ± 2.06**
NFS [4]	53.27 ± 2.75	60.21 ± 5.99	9.99 ± 1.17	5.90 ± 0.92	22.23 ± 1.60	11.64 ± 0.81	6.51 ± 0.57	31.67 ± 1.48
HKNN [22]	36.94 ± 2.71	48.01 ± 4.80	4.70 ± 0.33	3.70 ± 0.44	12.70 ± 1.00	6.34 ± 0.50	4.64 ± 0.52	20.41 ± 0.80
CKNN [22]	54.45 ± 3.30	68.94 ± 2.19	7.93 ± 0.56	5.47 ± 0.52	14.83 ± 1.06	8.89 ± 0.75	5.67 ± 0.67	26.24 ± 0.82
MD	56.09 ± 2.91	53.83 ± 5.45	12.81 ± 0.85	10.53 ± 0.79	22.06 ± 1.20	5.34 ± 0.55	1.74 ± 0.42	3.40 ± 1.04
PSDML [28]	55.30 ± 1.90	61.21 ± 5.24	12.14 ± 1.04	9.43 ± 1.41	25.43 ± 1.29	7.04 ± 0.60	4.14 ± 0.52	29.86 ± 1.69
KPLS[18]-**ES2**	54.02 ± 2.61	64.89 ± 5.18	20.21 ± 2.03	16.21± 1.45	27.23 ± 1.80	14.83 ± 1.93	11.61 ± 1.10	23.99 ± 2.43
KCCA[9]-**ES2**	55.80 ± 3.12	67.80 ± 3.71	38.60 ± 1.39	33.20± 1.77	53.26 ± 0.80	36.39 ± 1.61	30.87 ± 1.77	50.96 ± 1.44
KGMA[19]-**ES2**	58.19 ± 4.00	89.36 ± 6.88	41.89 ± 1.54	38.29 ± 1.90	52.87 ± 1.61	38.03 ± 2.42	33.29 ± 1.67	50.06 ± 1.29
LERM-ES1	64.20 ± 2.90	93.83 ± 2.50	19.60 ± 1.30	16.06 ± 0.94	28.54 ± 2.02	23.73 ± 1.32	19.77 ± 1.21	38.60 ± 1.26
LERM-ES2	**68.75 ± 2.72**	**95.53 ± 2.92**	**45.44 ± 1.12**	**42.10 ± 1.08**	**57.04 ± 1.10**	**41.89 ± 1.95**	**37.27 ± 1.88**	53.50 ± 1.25
LERM-ES3	**69.11 ± 2.99**	**96.60 ± 3.36**	**45.71 ± 2.05**	**42.80 ± 1.86**	**58.37 ± 3.31**	**49.07 ± 1.53**	**44.16 ± 0.94**	**63.83 ± 1.58**

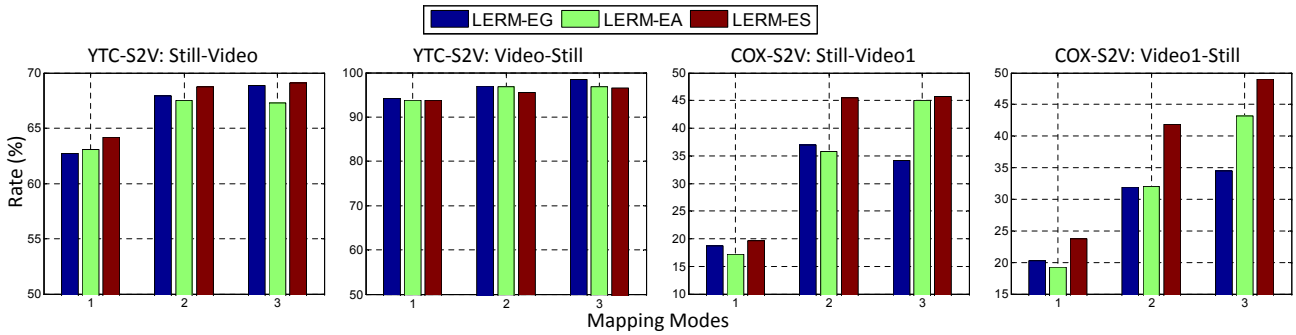

Figure 4. The comparisons of our method LERM in three cases (**EG, EA and ES**) with three Mapping Modes on YTC-S2V and COX-S2V.

ten cross validation experiments, in each of which one person had 3 randomly chosen videos for the gallery and others for probes. To design a S2V scenario, we randomly selected still images from the videos. The COX-S2V [10] is a public still and video face dataset from 1,000 subjects. *Based on it, we reconstruct a new version*[2] *by adding more videos captured by another camera*. In this dataset, each subject has 1 still images and 3 video clips, which are enrolled into 3 sets of videos namely *Video1, Video2, Video3* respectively. The still and video faces were resized to 48×60 and processed by histogram equalization. In the protocol of this dataset, 300 subjects are randomly selected and their still and video data are all used for training. From the rest 700 subjects, the still images and the three sets of videos (i.e.,*Video1-3*) are used for different testings detailed below.

As shown in Tab.2 and Fig.4 [3], for **still-video** recognition, we provide the rank-1 recognition rate of related methods by querying probe videos with gallery still images. We also conduct the rank-1 **video-still** recognition by identifying probe images with gallery videos. In Tab.2, each reported rate is a mean value plus a standard variance over the ten runs of testing. Since little image variance exists in *Video3* of COX-S2V dataset, our method LERM can not take advantage of the set modeling thus its performances are just comparable to those of NCA and LMNN. The other results consistently show that LERM achieves much higher recognition rate than other metric learning methods. We attribute this to our method exploring the specific non-Euclidean geometry of the space of set models, which typically reside on a certain type of Riemannian manifold. Besides, compared with other kernelized multiview methods, the results exhibit that our approach LERM is the most promising one for exploiting multi-view kernels. The results of Fig.4 show that new method LERM-ES3 (**ES** case, Mapping Mode III) outperforms other combinations of three cases and three Mapping Modes. This is because that **ES** case models each set as a covariance matrix, which is more robust set modeling than linear and affine subspaces [23]. In addition, the Mapping Mode III exploits the richest representations of original Euclidean points and Riemannian points.

[2] http://vipl.ict.ac.cn/resources/datasets/cox-face-dataset/cox-face

[3] Please kindly refer to our Supplementary Material for more results.

Table 3. Average training and testing time (seconds) on YTC-S2V.

Method	NCA	ITML	LMNN	PSDML	**LERM-ES3**
Training	7761	523.2	282.4	55.78	**3.615**
Testing	46.39	110.8	45.44	**4.437**	**8.865**

5.3. Computational cost

We compare the training and the testing time of different metric learning methods on an Intel(R) Core(TM) i5-3210M (2.5GHz) PC. The average training and testing time for total subjects on the YTC-S2V dataset is listed in Tab.3. The results show that our method LERM is much faster than other methods in training. Since comparing points with set models is typically cheaper than matching points with sets of points when those sets are very large, the PSDML and LERM methods are faster than others in testing.

6. Conclusion

In this paper, we propose a novel metric learning framework called Learning Euclidean-to-Riemannian Metric (LERM) for point-to-set classification. Different from other metric learning methods, our method learns the metric between Euclidean space and Riemannian manifold. Extensive experiments for image-to-set object classification and still-to-video face recognition demonstrate that the proposed method impressively outperforms and is faster than the state-of-the-art metric learning methods in both training and testing. In the future, this framework can be extended for metrics between sets or even different kinds of sets.

Acknowledgements

The work is partially supported by Natural Science Foundation of China under contracts nos.61025010, 61222211, 61379083, and 61390511.

References

[1] V. Arsigny, P. Fillard, X. Pennec, and N. Ayache. Geometric means in a novel vector space structure on symmetric positive-definite matrices. *SIAM J. Matrix Analysis and Applications*, 29(1):328–347, 2007.

[2] M. M. Bronstein, A. M. Bronstein, F. Michel, and N. Paragios. Data fusion through cross-modality metric learning using similarity-sensitive hashing. In *CVPR*, 2010.

[3] H. Cevikalp, B. Triggs, and R. Polikar. Nearest hyperdisk methods for high-dimensional classification. In *ICML*, 2008.

[4] J. Chien and C. Wu. Discriminant waveletfaces and nearest feature classifiers for face recognition. *IEEE T-PAMI*, 24(12):1644–1649, 2002.

[5] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon. Information-theoretic metric learning. In *ICML*, 2007.

[6] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood components analysis. In *NIPS*, 2004.

[7] J. Hamm and D. D. Lee. Extended grassmann kernels for subspace-based learning. In *NIPS*, 2008.

[8] J. Hamm and D. D. Lee. Grassmann discriminant analysis: a unifying view on subspace-based learning. In *ICML*, 2008.

[9] D. Hardoon, S. Szedmak, and J. Shawe-Taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural Computation*, 16(12):2639–2664, 2004.

[10] Z. Huang, S. Shan, H. Zhang, H. Lao, A. Kuerban, and X. Chen. Benchmarking still-to-video face recognition via partial and local linear discriminant analysis on COX-S2V dataset. In *ACCV*, 2012.

[11] S. Jayasumana, R. Hartley, M. Salzmann, H. Li, and M. Harandi. Kernel methods on the riemannian manifold of symmetric positive definite matrices. In *CVPR*, 2013.

[12] M. Kim, S. Kumar, V. Pavlovic, and H. Rowley. Face tracking and recognition with visual constraints in real-world videos. In *CVPR*, 2008.

[13] B. Leibe and B. Schiele. Analyzing appearance and contour based methods for object categorization. In *CVPR*, 2003.

[14] J. Masci, M. Bronstein, A. Bronstein, and J. Schmidhuber. Multimodal similarity-preserving hashing. *IEEE T-PAMI*, 36(4):824–830, 2013.

[15] B. McFee and G. Lanckriet. Learning multi-modal similarity. *JMLR*, 12:491–523, 2011.

[16] X. Pennec, P. Fillard, and N. Ayache. A riemannian framework for tensor computing. *IJCV*, 66(1):41–66, 2006.

[17] N. Quadrianto and C. H. Lampert. Learning multi-view neighborhood preserving projections. In *ICML*, 2011.

[18] A. Sharma and D. Jacobs. Bypassing synthesis: PLS for face recognition with pose, low-resolution and sketch. In *CVPR*, 2011.

[19] A. Sharma, A. Kumar, H. Daume, and D. Jacobs. Generalized multiview analysis: A discriminative latent space. In *CVPR*, 2012.

[20] M. Sugiyama. Dimensionality reduction of multimodal labeled data by local fisher discriminant analysis. *JMLR*, 8:1027–1061, 2007.

[21] R. Vemulapalli, J. K. Pillai, and R. Chellappa. Kernel learning for extrinsic classification of manifold features. In *CVPR*, 2013.

[22] P. Vincent and Y. Bengio. K-local hyperplane and convex distance nearest neighbor algorithms. In *NIPS*, 2001.

[23] R. Wang, H. Guo, L. S. Davis, and Q. Dai. Covariance discriminative learning: A natural and efficient approach to image set classification. In *CVPR*, 2012.

[24] K. Q. Weinberger and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. *JMLR*, 10:207–244, 2009.

[25] P. Xie and E. P. Xing. Multi-modal distance metric learning. In *IJCAI*, 2013.

[26] O. Yamaguchi, K. Fukui, and K. Maeda. Face recognition using temporal image sequence. In *FG*, 1998.

[27] X. Zhai, Y. Peng, and J. Xiao. Heterogeneous metric learning with joint graph regularization for cross-media retrieval. In *AAAI*, 2013.

[28] P. Zhu, L. Zhang, W. Zuo, and D. Zhang. From point to set: extend the learning of distance metrics. In *ICCV*, 2013.