Unsupervised Riemannian Clustering of Probability Density Functions

Alvina Goh and René Vidal

Center for Imaging Science, Johns Hopkins University
3400 North Charles Street, Baltimore, MD 21218, USA

Abstract. We present an algorithm for grouping families of probability density functions (pdfs). We exploit the fact that under the square-root re-parametrization, the space of pdfs forms a Riemannian manifold, namely the unit Hilbert sphere. An immediate consequence of this re-parametrization is that different families of pdfs form different submanifolds of the unit Hilbert sphere. Therefore, the problem of clustering pdfs reduces to the problem of clustering multiple submanifolds on the unit Hilbert sphere. We solve this problem by first learning a low-dimensional representation of the pdfs using generalizations of local nonlinear dimensionality reduction algorithms from Euclidean to Riemannian spaces. Then, by assuming that the pdfs from different groups are separated, we show that the null space of a matrix built from the local representation gives the segmentation of the pdfs. We also apply of our approach to the texture segmentation problem in computer vision.

Keywords: Manifold learning, manifold clustering, probability density functions.

1 Introduction

Over the past few decades, there has been a huge explosion in the amount of readily available information. In order to be able to efficiently process this abundance of data, probability density functions (pdf) are often employed to model complex datasets. This has led to the development of various metrics to measure the similarities between different pdfs. *Information geometry* refers to the study of the intrinsic geometric structures in the manifold of pdfs. The Riemannian structure of the space of pdfs was first introduced in [1]. Since then, there have been major breakthroughs in the theory of information geometry [2], and also in the development of computational tools for clustering that utilize the geometric structure of the Riemannian manifold [3,4,5].

An area in which probability density functions are commonly used in computer vision is texture analysis. Segmentation of different textures remains important for present-day applications and is the focus of considerable effort in the field. For example, it is vital to be able to perform automatic segmentation of different land cover from remotely sensed images in environmental management, as it is tedious for photo-interpreters to classify landscape manually. It is well-known that by convolving the image with a set of filters, it is possible to obtain a spectral pdf of the image which indicates the texture characteristics. Therefore, given a set of images with different textures, it is often desirable to be able to automatically segment the textures into similar classes by clustering the pdfs into different families. For arbitrary textures, we do not necessarily know the description of the pdfs, though it is reasonable to assume that similar textures will have similar pdfs.

The question we address in this paper is the following. Given a set of pdfs, how do we develop a computationally simple framework that allows us to group the pdfs into similar families? Since these pdfs are determined from data, they are non-parametric in nature. Therefore, in order to compute distances between two arbitrary pdfs, we impose a Riemannian structure on the manifold of pdfs. Unlike traditional clustering methods, we will not assume that the pdfs within each group are centered around a collection of cluster centers in the manifold. Instead, we will assume that the different groups of pdfs form different submanifolds in the Riemannian space. Therefore, we aim to develop a framework that exploits the Riemannian structure of the space of pdfs to cluster a given set of arbitrary pdfs into similar groups.

Our clustering framework makes use of nonlinear dimensionality reduction techniques. Nonlinear dimensionality reduction (NLDR) refers to the problem of finding a low-dimensional representation for a set of points lying in a nonlinear manifold embedded in a high-dimensional space. Existing NLDR techniques can be categorized into two main groups: global and local techniques. Global techniques attempt to preserve global properties of the data lying in a submanifold, similar to what Principal Component Analysis (PCA) [6] attempts to preserve for data lying in a linear subspace. Two of the best-known examples of this family of algorithms are ISOMAP [7] and Kernel PCA (KPCA) [8]. Local techniques are however based on the preservation of local properties which are obtained from the small neighborhoods around the datapoints. The key idea of such techniques is that by preserving the local properties of the data, one can also retain the global properties of the data. Locally linear embedding (LLE) [9], Laplacian eigenmaps (LE) [10] and Hessian LLE [11] fall under this category of algorithms. In this paper, we chose to use local NLDR techniques such as LLE, LE and HLLE. We will show that the segmentation of the data can be obtained from the null space of a matrix built from the local representation. This is a property that local NLDR methods offer but global NLDR methods such as ISOMAP do not.

Paper contributions The main contribution of this paper is the development of a framework for clustering different families of pdfs. By choosing the square-root representation, we reduce the problem to one of clustering data lying in different submanifolds of a unit sphere. As in our previous work [12], we learn a local representation of the data using generalization of the three NLDR techniques, namely Laplacian Eigenmaps (LE) [10], Locally Linear Embedding (LLE) [9], and Hessian LLE (HLLE) [11], from Euclidean to Riemannian spaces. We show that the null space of a matrix built from the local representation gives the segmentation of the pdfs. Our method is computationally simple and performs automatic segmentation without requiring user interaction.

2 Review of Local Nonlinear Dimensionality Reduction Methods in Euclidean Spaces

In this section, we review three local NLDR algorithms. Let $X = \{\boldsymbol{x}_i \in \mathcal{M}\}_{i=1}^n$ be a set of n data points sampled from a d-dimensional manifold $\mathcal{M}$ embedded in $\mathbb{R}^D$, $d \ll D$.

We assume that the n points are k**-connected**, i.e., for any two points $\boldsymbol{x}_i, \boldsymbol{x}_j \in X$ there is an ordered sequence of points in X having $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ as endpoints, such that any two consecutive points in the sequence have at least one k-nearest neighbor in common. The goal of dimensionality reduction is to find a set of vectors $\{\mathbf{y}_i \in \mathbb{R}^d\}_{i=1}^n$, such that nearby points remain close and distant points remain far.

Locally Linear Embedding (LLE) [9] assumes that the local neighborhood of a point in the manifold can be well approximated by the affine subspace spanned by the k-nearest neighbors of the point, and finds a low-dimensional embedding of the data based on these affine approximations. **Laplacian Eigenmaps (LE)** [10] are based on computing the low dimensional representation that best preserves *locality* instead of *local linearity* in LLE. **Hessian LLE (HLLE)** [11] bears substantial resemblance to LLE and LE, with the main difference being that the local neighborhood is represented by the tangent space at each point and the Laplacian matrix is replaced by the Hessian matrix. The main steps of these local NLDR algorithms are as follows:

1. *Nearest neighbor search:* For each data point $\boldsymbol{x}_i \in X$, find its k nearest neighbors (kNN) $\{\boldsymbol{x}_{i_j}\}_{j=1}^k$ according to the Euclidean distance.
2. *Construction of similarity matrix:* Construct a weighted graph whose elements encode the local geometry of the data. Define a similarity matrix M based on these weights. M is symmetric and positive semidefinite.
3. *Sparse eigenvalue problem*: Obtain the embedding coordinates, i.e., the columns of $Y = [\mathbf{y}_1, \ldots, \mathbf{y}_n]^\top \in \mathbb{R}^{n \times d}$, from the d (generalized) eigenvectors of the matrix M associated with its second to $(d+1)$-th smallest (generalized) eigenvalues. The vector of all ones, $\mathbf{1} \in \mathbb{R}^n$, is a eigenvector of M associated with eigenvalue 0.

We will now describe the construction of M for each NLDR algorithm in more detail.

Calculation of M in LLE

1. *Weight matrix*: Find a matrix of weights $W \in \mathbb{R}^{n \times n}$ whose entries W_{ij} minimize the reconstruction error

$$\varepsilon(W) = \sum_{i=1}^n \| \sum_{j=1}^n W_{ij} \boldsymbol{x}_j - \boldsymbol{x}_i \|^2 = \sum_{i=1}^n \operatorname{dist}^2(\widehat{\boldsymbol{x}}_i, \boldsymbol{x}_i) \tag{1}$$

subject to the constraints (i) $W_{ij} = 0$ if $\boldsymbol{x}_j$ is not a k-nearest neighbor of $\boldsymbol{x}_i$ and (ii) $\sum_{j=1}^n W_{ij} = 1$. In (1), $\widehat{\boldsymbol{x}}_i = \boldsymbol{x}_i + \sum_{j=1}^n W_{ij} \overrightarrow{\boldsymbol{x}_i \boldsymbol{x}_j}$ is the linear interpolation of $\boldsymbol{x}_i$ and its kNN. The solution to this problem can be computed as

$$\begin{bmatrix} W_{i\,i_1} & W_{i\,i_2} & \cdots & W_{i\,i_k} \end{bmatrix} = \frac{\mathbf{1}^\top C_i^{-1}}{\mathbf{1}^\top C_i^{-1} \mathbf{1}} \in \mathbb{R}^{1 \times k}, \tag{2}$$

where $\mathbf{1} \in \mathbb{R}^n$ is the vector of all ones, and $C_i \in \mathbb{R}^{k \times k}$ is the local Gram matrix at $\boldsymbol{x}_i$, i.e., $C_i(j,l) = (\boldsymbol{x}_j - \boldsymbol{x}_i) \cdot (\boldsymbol{x}_l - \boldsymbol{x}_i)$.
2. *Objective function*: Find vectors $\{\mathbf{y}_i \in \mathbb{R}^d\}_{i=1}^n$ that minimize the error

$$\phi(Y) = \sum_{i=1}^n \|\mathbf{y}_i - \sum_{j=1}^n W_{ij} \mathbf{y}_j\|^2 = \operatorname{trace}(Y^\top M Y), \tag{3}$$

subject to the constraints (i) $\sum_{i=1}^n \mathbf{y}_i = \mathbf{0}$ and (ii) $\frac{1}{n}\sum_{i=1}^n \mathbf{y}_i\mathbf{y}_i^\top = I$. The solution to this optimization problem is given by the d eigenvectors of $M = (I - W)^\top(I - W)$ associated with its second to $(d+1)$-th smallest eigenvalues.

Calculation of M in LE

1. *Weight matrix*: Construct a matrix of weights $W \in \mathbb{R}^{n\times n}$
$$W_{ij} = \exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/\sigma^2) \tag{4}$$
subject to the constraint $W_{ij} = 0$ if $\boldsymbol{x}_j$ is not a k-nearest neighbor of $\boldsymbol{x}_i$. The entries of W, W_{ij}, measure the proximity between two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.
2. *Objective function*: Find vectors $\{\mathbf{y}_i \in \mathbb{R}^d\}_{i=1}^n$ that minimize the error
$$\phi(Y) = \sum_{i,j} \|\mathbf{y}_i - \mathbf{y}_j\|^2 W_{ij} = \text{trace}(Y^\top MY) \tag{5}$$
subject to the constraints (i) $Y^\top D\mathbf{1} = \sum_{i=1}^n D_{ii}\mathbf{y}_i = \mathbf{0}$ (weighted low-dimensional coordinates centered at the origin) and (ii) $Y^\top DY = I$ (weighted low-dimensional coordinates having unit covariance). In Eq. (5), $M = D - W$ is the graph Laplacian matrix and D is a diagonal matrix whose entries are given by $D_{ii} = \sum_j W_{ij}$. The solution to this optimization problem is given by the d generalized eigenvectors of (M, D) associated with its second to $(d+1)$-th smallest generalized eigenvalues.

Calculation of M in HLLE

1. *Tangent coordinates*: For each data point $\boldsymbol{x}_i$, let $\{\boldsymbol{x}_{i_j}\}_{j=1}^k$ be its kNN. Form the D by D covariance matrix cov$(\boldsymbol{x}_i) = \frac{1}{k}\sum_{j=1}^k(\boldsymbol{x}_{i_j} - \bar{\boldsymbol{x}}_i)(\boldsymbol{x}_{i_j} - \bar{\boldsymbol{x}}_i)^\top$, where $\bar{\boldsymbol{x}}_i$ is the mean of the kNN. Perform an eigenanalysis of the matrix cov$(\boldsymbol{x}_i)$ to obtain the d eigenvectors $\{\mathbf{u}_q \in \mathbb{R}^D\}_{q=1}^d$. The tangent coordinates of the kNN are given by the d columns of the $k \times d$ matrix V given below, where $p = 1, \dots, k$ and $q = 1, \dots, d$
$$V_{pq} = (\boldsymbol{x}_{i_p} - \bar{\boldsymbol{x}}_i)^\top \mathbf{u}_q = \langle \boldsymbol{x}_{i_p} - \bar{\boldsymbol{x}}_i, \mathbf{u}_q\rangle. \tag{6}$$
2. *Objective function*: The embedding vectors are obtained based on the null vectors of a matrix M that indicates the Hessian quadratic cost. While we refer the reader to [11] for details on the estimation of M, the basic principle is as follows. We first locally estimate a Hessian operator h^i at each point $\boldsymbol{x}_i$ in the manifold in a least squares sense. In particular, consider a smooth function $f : \mathcal{M} \to \mathbb{R}$. We evaluate the function at all kNN of a point $\boldsymbol{x}_i$ in the manifold and stack these entries into a vector $\mathbf{f}_i$. It can be shown that $h^i\mathbf{f}_i$ approximates the entries of the Hessian, whose (p, q)-th entry is given by $\frac{\partial^2 f}{\partial V_p \delta V_q}$. These local estimates are then used to obtain an empirical estimate of the (i, j)-th entry of M as
$$M_{i,j} = \sum_l \sum_r ((h^l)_{r,i}(h^l)_{r,j}). \tag{7}$$
The embedding coordinates are then found by selecting a basis for the space spanned by d eigenvectors of M associated with its second to $(d+1)$-th smallest eigenvalues

with the restriction that it provides an orthonormal basis to a specific fixed neighborhood $\mathcal{N}$. Let U denote the $n \times d$ matrix associated with the second to $(d+1)$-th smallest eigenvectors where $U_{l,r}$ is the l-th entry in the r-th eigenvector of M. The embedding coordinates is obtained as $UR^{-\frac{1}{2}}$, where $R_{r,s} = \sum_{j \in \mathcal{N}} U_{j,r} U_{j,s}$.

3 Clustering Submanifolds of a Riemannian Space

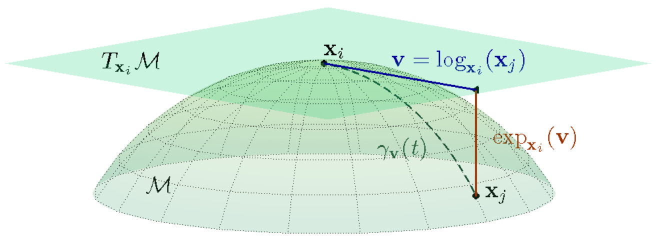
In this section, we present an algorithm for clustering and dimensionality reduction on Riemannian manifolds. We first present a brief summary of the theory of Riemannian manifolds in §3.1. For a more complete description, we refer the reader to [13]. We then illustrate how to extend existing NLDR algorithms to Riemannian manifolds in §3.2 by adopting the framework in [12]. Finally, in §3.3, we show that by making use of the mappings generated by NLDR, the problem of manifold clustering reduces to a central clustering problem, as proved in [12].

3.1 Review of Riemannian Manifolds

The NLDR techniques presented in §2 are applicable only in the presence of *one* manifold with *unknown structure*. Every operation is approximated by the corresponding Euclidean operation as the metric is unknown. However, for Riemannian manifolds with well-studied geometries, closed-form formulae for Riemannian operations are often available. The question now is to extend NLDR techniques for Riemannian manifolds in a way that takes into consideration the appropriate Riemannian structure. For this purpose, we adopt the framework developed in our previous work [12]. In this section, we will give an overview of Riemannian theory and show how the various operations such as interpolation on the manifold and computation of the mean and principal components are carried out.

A differentiable manifold $\mathcal{M}$ of dimension d is a topological space that is homeomorphic to the Euclidean space $\mathbb{R}^d$. Fig. 1 shows an example of a two-dimensional manifold, a smooth surface living in $\mathbb{R}^3$. The tangent space $T_{\boldsymbol{x}}\mathcal{M}$ at $\boldsymbol{x}$ is the vector space that contains the tangent vectors to all 1-D curves on $\mathcal{M}$ passing through $\boldsymbol{x}$. A *Riemannian metric* on a manifold $\mathcal{M}$ is a bilinear form which associates to each point $\boldsymbol{x} \in \mathcal{M}$, a differentiable varying inner product $\langle \cdot, \cdot \rangle_{\boldsymbol{x}}$ on the tangent space $T_{\boldsymbol{x}}\mathcal{M}$ at $\boldsymbol{x}$. The norm of a vector $\boldsymbol{v} \in T_{\boldsymbol{x}}\mathcal{M}$ is denoted by $\|\boldsymbol{v}\|_{\boldsymbol{x}}^2 = \langle \boldsymbol{v}, \boldsymbol{v} \rangle_{\boldsymbol{x}}$. The Riemannian distance $\text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ between two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ lying in the manifold is defined as the minimum length over all possible smooth curves on the manifold between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. The geodesic curve from $\boldsymbol{x}_i$ to $\boldsymbol{x}_j$, γ, is the smooth curve with minimum length.

Given a tangent vector $\boldsymbol{v} \in T_{\boldsymbol{x}}\mathcal{M}$, locally there exists a unique geodesic $\gamma_{\boldsymbol{v}}(t)$ starting at $\boldsymbol{x}$ with initial velocity $\boldsymbol{v}$, and this geodesic has constant speed equal to $\|\boldsymbol{v}\|_{\boldsymbol{x}}$. The *exponential map*, $\exp_{\boldsymbol{x}} : T_{\boldsymbol{x}}\mathcal{M} \to \mathcal{M}$ maps a tangent vector $\boldsymbol{v}$ to the point in the manifold that is reached at time 1 by the geodesic $\gamma_{\boldsymbol{v}}(t)$. The inverse of $\exp_{\boldsymbol{x}}$ is the *logarithm map* and denoted by $\log_{\boldsymbol{x}} : \mathcal{M} \to T_{\boldsymbol{x}}\mathcal{M}$. For two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in the manifold $\mathcal{M}$, the tangent vector to the geodesic curve from $\boldsymbol{x}_i$ to $\boldsymbol{x}_j$ is defined as $\boldsymbol{v} = \overrightarrow{\boldsymbol{x}_i \boldsymbol{x}_j} = \log_{\boldsymbol{x}_i}(\boldsymbol{x}_j)$, and the exponential map takes $\boldsymbol{v}$ to the point $\boldsymbol{x}_j = \exp_{\boldsymbol{x}_i}(\log_{\boldsymbol{x}_i}(\boldsymbol{x}_j))$. In addition, $\gamma_{\boldsymbol{v}}(0) = \boldsymbol{x}_i$ and $\gamma_{\boldsymbol{v}}(1) = \boldsymbol{x}_j$. The Riemannian distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is defined as $\text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \|\log_{\boldsymbol{x}_i}(\boldsymbol{x}_j)\|_{\boldsymbol{x}_i}$. Linear



$T_{\mathbf{x}_i}\mathcal{M}$
$\mathbf{x}_i$
$\mathbf{v} = \log_{\mathbf{x}_i}(\mathbf{x}_j)$
$\exp_{\mathbf{x}_i}(\mathbf{v})$
$\gamma_{\mathbf{v}}(t)$
$\mathcal{M}$
$\mathbf{x}_j$

Fig. 1. An example of a two-dimensional manifold. The tangent plane at $\boldsymbol{x}_i$, together with the exponential and logarithm maps relating $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, are also shown.

geodesic interpolation makes use of the exponential and logarithm maps and is given by $\widehat{\boldsymbol{x}} = \exp_{\boldsymbol{x}_i}(w\overrightarrow{\boldsymbol{x}_i\boldsymbol{x}_j})$, $w \in [0,1]$. Finally, the Riemannian metric, exponential and logarithm maps depend on the point $\boldsymbol{x}$ under consideration, hence the subscripts reflecting this dependency.

We will now briefly summarize how to calculate the mean and principal components of data points lying in a manifold. As defined by Fréchet in [14] and used in several recent works [15,16], the intrinsic mean $\overline{\mathbf{x}}$ is defined as the solution to the following minimization problem

$$\overline{\mathbf{x}} = \arg\min_{\boldsymbol{x}\in\mathcal{M}} \sum_{i=1}^{n} \operatorname{dist}(\boldsymbol{x},\boldsymbol{x}_i)^2 = \arg\min_{\boldsymbol{x}\in\mathcal{M}} \sum_{i=1}^{n} \|\log_{\boldsymbol{x}}(\boldsymbol{x}_i)\|_{\boldsymbol{x}}^2. \tag{8}$$

Note that, unlike in the Euclidean case, in general there is no closed form for $\overline{\mathbf{x}}$. Moreover, there is no guarantee that $\overline{\mathbf{x}}$ exists or is unique. However, one can show the existence and uniqueness of $\overline{\mathbf{x}}$ [17] by assuming that the data lie in a small enough neighborhood, i.e., the maximum distance between any $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is small enough. Furthermore, $\overline{\mathbf{x}}$ can be computed as shown in Algorithm 1.

Algorithm 1. (Intrinsic Mean)

```
Given data points $\boldsymbol{x}_1,\dots,\boldsymbol{x}_n \in \mathcal{M}$, a predefined threshold $\epsilon$, maximum number of iterations $T$,

1. Initialize $t = 1$, $\overline{\mathbf{x}}_1 = \boldsymbol{x}_i$ for a random $i$, $\boldsymbol{v} \neq 0 \in T_{\overline{\mathbf{x}}_1}\mathcal{M}$.
2. While $t \leq T$ or $\|\boldsymbol{v}\|_{\overline{\mathbf{x}}} \geq \epsilon$,
   (a) Compute tangent vector $\boldsymbol{v} = \frac{1}{n}\sum_{i=1}^{n} \log_{\overline{\mathbf{x}}_t}(\boldsymbol{x}_i)$.
   (b) Set $\overline{\mathbf{x}}_{t+1} = \exp_{\overline{\mathbf{x}}_t}(\boldsymbol{v})$
```

Given $\overline{\mathbf{x}}$, the calculation of principal components on a Riemannian manifold is not as straightforward as in the Euclidean case. It involves projecting a point onto a geodesic curve, which is also defined as a minimization problem for which existence and uniqueness are not ensured [15]. Again, by making the assumptions that the data lie in a small neighborhood, the projection can be shown to be unique. In [15], it is shown that finding principal components boils down to doing PCA in the tangent vectors $\log_{\overline{\mathbf{x}}}(\boldsymbol{x}_i) \in T_{\overline{\mathbf{x}}}\mathcal{M}$

Table 1. Comparison of Euclidean and Riemannian operations, where $\{\boldsymbol{x}_i\}_{i=1}^n$, are data points

Operation	Euclidean	Riemannian
Subtraction $\overrightarrow{\boldsymbol{x}_i\boldsymbol{x}_j}$	$\boldsymbol{x}_j - \boldsymbol{x}_i$	$\log_{\boldsymbol{x}_i}(\boldsymbol{x}_j)$
Addition $\boldsymbol{x}_j$	$\boldsymbol{x}_i + \overrightarrow{\boldsymbol{x}_i\boldsymbol{x}_j}$	$\exp_{\boldsymbol{x}_i}(\overrightarrow{\boldsymbol{x}_i\boldsymbol{x}_j})$
Distance $\text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j)$	$\Vert\overrightarrow{\boldsymbol{x}_i\boldsymbol{x}_j}\Vert = \Vert\boldsymbol{x}_j - \boldsymbol{x}_i\Vert$	$\Vert\log_{\boldsymbol{x}_i}(\boldsymbol{x}_j)\Vert_{\boldsymbol{x}_i} = \sqrt{\langle\log_{\boldsymbol{x}_i}(\boldsymbol{x}_j), \log_{\boldsymbol{x}_i}(\boldsymbol{x}_j)\rangle_{\boldsymbol{x}_i}}$
Mean $\overline{\mathbf{x}}$	$\sum_{i=1}^n \overrightarrow{\overline{\mathbf{x}}\boldsymbol{x}_i} = 0$	$\sum_{i=1}^n \log_{\overline{\mathbf{x}}}(\boldsymbol{x}_i) = 0$
Covariance $\text{cov}(\boldsymbol{x})$	$\frac{1}{n}\sum_{i=1}^n(\overrightarrow{\overline{\mathbf{x}}\boldsymbol{x}_i})(\overrightarrow{\overline{\mathbf{x}}\boldsymbol{x}_i})^\top$	$\frac{1}{n}\sum_{i=1}^n(\log_{\overline{\mathbf{x}}}(\boldsymbol{x}_i))(\log_{\overline{\mathbf{x}}}(\boldsymbol{x}_i))^\top$
Linear interpolation $\widehat{\boldsymbol{x}}$	$\boldsymbol{x}_i + w\overrightarrow{\boldsymbol{x}_i\boldsymbol{x}_j}$	$\exp_{\boldsymbol{x}_i}(w\overrightarrow{\boldsymbol{x}_i\boldsymbol{x}_j})$

Algorithm 2. (Principal Geodesic Analysis)

```
Given data points $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n \in \mathcal{M}$,

1. Compute intrinsic mean $\overline{\mathbf{x}}$ as in Algorithm 1.
2. Calculate the tangent vectors $\boldsymbol{v}_i = \log_{\overline{\mathbf{x}}}(\boldsymbol{x}_i)$about $\overline{\mathbf{x}}$.
3. Construct the sample covariance matrix $\text{cov}(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^n \boldsymbol{v}_i\boldsymbol{v}_i^\top$.
4. Perform eigenanalysis of the matrix $\text{cov}(\boldsymbol{x})$, with the eigenvectors $\{\mathbf{u}_i\}_{i=1}^d$ giving the principal directions. $\{\mathbf{u}_i\}_{i=1}^d$ forms an orthonormal basis for $T_{\overline{\mathbf{x}}}\mathcal{M}$.
```

about the mean $\overline{\mathbf{x}}$. This algorithm, known as Principal Geodesic Analysis (PGA), is summarized in Algorithm 2. Table 1 compares the standard operations in Euclidean and Riemannian spaces.

3.2 Extending NLDR to Riemannian Manifolds

Notice that the information about the local geometry of the manifold is essential only in the first two steps of each algorithm and therefore, modifications are made only to these two stages. The key issues are how to select the kNN and how to compute the matrix M representing the local geometry. As shown in [12], the former is straightforward, while the latter requires some thought. Given M, calculating the low-dimensional representation remains the same as in the Euclidean case. We let $X = \{\boldsymbol{x}_i \in \mathbb{R}^D\}_{i=1}^n$ be a set of n data points sampled from a known Riemannian manifold.

Selection of the Riemannian kNN The first step of any NLDR algorithm is the computation of the kNN associated with each data point. We define the kNN of $\boldsymbol{x}_i$ by incorporating the Riemannian distance, i.e., *the kNN of $\boldsymbol{x}_i$ are the k data points $\boldsymbol{x}_j$ that minimize* $\Vert\log_{\boldsymbol{x}_i}(\boldsymbol{x}_j)\Vert_{\boldsymbol{x}_i}$.

Riemannian Calculation of M for LLE The second step of LLE is to compute the matrix of weights $W \in \mathbb{R}^{n\times n}$. In order to do so, we will answer two main questions: 1) how does one express a point as a linear combination of its neighbors? and 2) what is the reconstruction cost? First of all, we know that from §3.1 that

$$\widehat{\boldsymbol{x}}_{Riem,i} = \exp_{\boldsymbol{x}_i}(\sum_{j=1}^{n} W_{ij} \log_{\boldsymbol{x}_i}(\boldsymbol{x}_j)). \tag{9}$$

is the geodesic linear interpolation of $\boldsymbol{x}_i$ by $\{\boldsymbol{x}_j\}_{j=1}^n$. Now, instead of minimizing the Euclidean error, we rewrite (1) to minimize the Riemannian reconstruction error and make use of the fact that $\exp$ and $\log$ are inverse mappings. Therefore, we have

$$\varepsilon_{Riem}(W) = \sum_{i=1}^{n} \| \log_{\boldsymbol{x}_i}(\widehat{\boldsymbol{x}}_{Riem,i}) \|_{\boldsymbol{x}_i}^2 = \sum_{i=1}^{n} \| \sum_{j=1}^{n} W_{ij} \log_{\boldsymbol{x}_i}(\boldsymbol{x}_j) \|_{\boldsymbol{x}_i}^2 \tag{10}$$

subject to $W_{ij} = 0$ if $\boldsymbol{x}_j$ is not a kNN of $\boldsymbol{x}_i$ and $\sum_j W_{ij} = 1$. Using similar manipulations as in the Euclidean case, the optimal weights are obtained as in (2), with the local Gram matrix $C_i \in \mathbb{R}^{k \times k}$ defined as

$$C_i(j,l) = \langle \log_{\boldsymbol{x}_i}(\boldsymbol{x}_j), \log_{\boldsymbol{x}_i}(\boldsymbol{x}_l) \rangle_{\boldsymbol{x}_i}. \tag{11}$$

M is then $(I - W)^\top (I - W)$.

Riemannian Calculation of M for LE Here, instead of attempting to write each data point as a linear combination of its kNN, we find a matrix of weights $W \in \mathbb{R}^{n \times n}$ whose entries W_{ij} measure the proximity between two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ as in (4). Therefore, modifying LE for Riemannian manifolds is less involved than in the case of LLE. Instead of using $\exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/\sigma^2)$ as in (4), we construct the weight matrix W using the Riemannian distance as

$$W_{ij} = \exp\Big(-\frac{\text{dist}_{Riem}(\boldsymbol{x}_i, \boldsymbol{x}_j)^2}{\sigma^2}\Big) = \exp\Big(-\frac{\| \log_{\boldsymbol{x}_i}(\boldsymbol{x}_j) \|_{\boldsymbol{x}_i}^2}{\sigma^2}\Big) \tag{12}$$

subject to the constraint $W_{ij} = 0$ if $\boldsymbol{x}_j$ is not a k-nearest neighbor of $\boldsymbol{x}_i$. As before, $M = D - W$ and D is a diagonal matrix, where $D_{ii} = \sum_j W_{ij}$.

Riemannian Calculation of M for HLLE The second step of HLLE involves computing the tangent coordinates for each $\boldsymbol{x}_i$ by applying Euclidean PCA to its neighbors. This implicitly assumes that these local points lie in a subspace. This assumption is no longer valid if $\boldsymbol{x}_i$ and its kNN lie in a Riemannian manifold. From §3.1, we know that in this case, calculating the principal geodesic components and the projection coordinates is not as simple as doing Euclidean PCA. There is a need to incorporate the correct Riemannian metric, mean and covariance matrix.

Again, let $\{\boldsymbol{x}_{i,j}\}_{j=1}^k$ denote the set of k-nearest neighbors of $\boldsymbol{x}_i$. First we calculate the intrinsic mean $\bar{\boldsymbol{x}}_i$ of the kNN (Algorithm 1). Next, we find the tangent vectors $\boldsymbol{v}_j = \log_{\bar{\boldsymbol{x}}_i}(\boldsymbol{x}_{i,j})$ about $\bar{\boldsymbol{x}}_i$ and the geodesic principal directions $\{\mathbf{u}_q\}_{q=1}^d$ using PGA (Algorithm 2). Since $\{\mathbf{u}_q \in \mathbb{R}^D\}_{q=1}^d$ is an orthonormal basis for $T_{\bar{\boldsymbol{x}}_i}\mathcal{M}$, we will rewrite the projection operator in (6) using the Riemannian metric. Thus the tangent coordinates of the kNN are given by the $k \times d$ matrix V, where

$$V_{pq} = \langle \log_{\bar{\boldsymbol{x}}_i}(\boldsymbol{x}_{i,p}), \mathbf{u}_q \rangle_{\bar{\boldsymbol{x}}_i},\ p = 1,..,k,\ q = 1,..,d. \tag{13}$$

Once the tangent coordinates are found, the estimation of the Hessian matrix M is the same as in the Euclidean case (7).

Calculation of the Embedding Coordinates The last step of NLDR is to find a Euclidean low-dimensional representation of the data points. As this step is independent of the Riemannian structure, one can find the embedding coordinates as described in §2. That is, the embedding coordinates are obtained based on the d (generalized) eigenvectors of the matrix M associated with its second to $(d+1)$-th smallest (generalized) eigenvalues. Finally, notice that if the Riemannian operations are available in closed-form, then extending NLDR to Riemannian manifolds do not require significant additional computational complexity.

3.3 Local Riemannian Manifold Clustering

In this section, we review the extension of NLDR algorithms for the purpose of clustering data lying in m submanifolds of a Riemannian space proposed [12]. We assume that the data is distributed in a k-disconnected union of m k-connected submanifolds of $\mathcal{M}$. Under this assumption, [12] shows that each of the m submanifolds will be mapped to a different point in $\mathbb{R}^m$. Proposition 1 states the main result of [12]. This proposition shows that in the case of a disconnected union of m k-connected submanifolds, the matrix M has at least m zero eigenvalues, whose eigenvectors give the clustering of the data. This is a generalized result that is applicable to Riemannian LLE, Riemannian LE and Riemannian HLLE. The interested reader is referred to [12] for the proof of Proposition 1.

Proposition 1 *Let $\{\boldsymbol{x}_i\}_{i=1}^n$ be a set of points drawn from a disconnected union of m k-connected d-dimensional submanifolds of a Riemannian manifold. Then, there exist m eigenvectors $\{\mathbf{u}_j\}_{j=1}^m$ in the null space of M such that $\mathbf{u}_j$ corresponds to the j-th group of points, i.e., $\mathbf{u}_{ij} = 1$ if the i-th data point is in the j-th group, and $\mathbf{u}_{ij} = 0$ otherwise.*

With real data, the assumption that the submanifolds are separated will obviously be violated. Therefore, the matrix M will be a perturbed version of the ideal case. However, it is well-known from perturbation theory [18] that if the perturbation is small or the eigengap is big, the eigenvectors $\boldsymbol{v}_j$ might not coincide completely with the indicator vectors $(\mathbf{0}, ., \mathbf{1}, ., \mathbf{0})^\top$ of the clusters, but do so up to a small error term. Hence, it is reasonable to expect that instead of mapping data points on m submanifolds to m points, the mapping will generate a collection of n points distributed around m cluster centers.

We see that there exists a mapping $g : \mathcal{M} \rightarrow \mathbb{R}^m$ that gives the membership of each point to the m submanifolds. This mapping is given by the rows of any basis for $\ker(M)$. However, notice that we do not necessarily obtain the set of membership vectors $\{\mathbf{u}_j\}$ when computing a basis for $\ker(M)$, but rather linear combinations of them, including the vector $\mathbf{1}$. In general, linear combinations of segmentation eigenvectors still contain the segmentation of the data. Hence, we can cluster the data into m groups by applying k-means to the columns of a matrix whose rows are the m eigenvectors in the null space of M. Algorithm 3 summarizes our dimensionality reduction and clustering algorithm for m submanifolds of a Riemannian space.

Algorithm 3. (Unsupervised Clustering and Dimensionality Reduction on Riemannian Manifolds)

Given data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathcal{M}$,

1. *Nearest neighbors:* Find the kNN of each data point $\boldsymbol{x}_i$ according to the Riemannian distance.
2. *Construction of M:* For each NLDR algorithm, construct the appropriate M described in §3.2.
3. *Clustering:* Compute the m eigenvectors $\{\mathbf{u}_j\}_{j=1}^m$ of M associated with its m smallest eigenvalues and apply k-means to the rows of $[\mathbf{u}_1, \cdots, \mathbf{u}_m]$ to cluster the data into m different groups.
4. *Low-dimensional embedding:* Apply NLDR to each group to obtain a low-dimensional embedding for each submanifold.

4 Riemannian Analysis of Probability Density Functions

In this section, we will show how to impose a Riemannian structure on the space of pdfs. We will adopt the work of [5], which proposes a "spherical" version of the Fisher-Rao metric that allows for closed-form expressions for the various Riemannian operations.

The class of constrained non-negative continuous functions under study here is the set of pdfs defined below. Without loss of generality, we can assume that these functions are defined on the interval $[0, T]$. Therefore, the set $\boldsymbol{\mathcal{P}}$ of pdfs is given by

$$\boldsymbol{\mathcal{P}} = \{\boldsymbol{p} : [0, T] \to \mathbb{R} | \forall s, \boldsymbol{p}(s) \geq 0, \int_0^T \boldsymbol{p}(s) ds = 1\}. \tag{14}$$

The question of how to regard the space of pdfs as a differential manifold endowed with a Riemannian metric and a family of affine connections has a long history behind it. Nevertheless, it remains an active and important research area. Treating statistical structures as geometric structures has the advantage that geometric structures remain invariant under coordinate transforms. [1] first introduces the Riemannian structure formed by the statistical manifold where each point in the manifold denotes a pdf. In addition, [1] also shows that the *Fisher-Rao* metric determines a Riemannian metric. The Fisher-Rao metric is later shown to be the *unique intrinsic metric* on the statistical manifold in [19]. This study of probability and information via differential geometry is known as *information geometry*. The reader is referred to the seminal work of [2] for a complete description.

We will consider the manifold $\boldsymbol{\mathcal{P}}$ of pdfs on the interval $[0, T]$. For any point $\boldsymbol{p}_i \in \boldsymbol{\mathcal{P}}$, the Fisher-Rao metric is defined as

$$\langle \boldsymbol{q}_j, \boldsymbol{q}_k \rangle_{\boldsymbol{p}_i} = \int_0^T \boldsymbol{q}_j(s) \boldsymbol{q}_k(s) \frac{1}{\boldsymbol{p}_i(s)} ds, \tag{15}$$

where $\boldsymbol{q}_j, \boldsymbol{q}_k \in T_{\boldsymbol{p}_i}(\boldsymbol{\mathcal{P}})$ are tangent vectors and $T_{\boldsymbol{p}_i}(\boldsymbol{\mathcal{P}})$ is the set containing the functions tangent to $\boldsymbol{\mathcal{P}}$ at the point $\boldsymbol{p}_i$. This representation turns out to be extremely difficult to work with as ensuring the geodesic between two elements lies on $\boldsymbol{\mathcal{P}}$ is not easy [5].

Even though the space $\boldsymbol{\mathcal{P}}$ turns out to be difficult to work with, we know that it is not the only possible representation for pdfs and in addition, we also know that the Fisher-Rao metric is the only metric that is invariant to re-parameterizations (essentially coordinate transforms) of the functions [19]. There are many different re-parameterizations of pdfs that are equivalent representations. These include cumulative distribution functions $\int_0^s \boldsymbol{p}(t)dt$, log density functions $\log \boldsymbol{p}(s)$ and square-root density functions $\sqrt{\boldsymbol{p}(s)}$. Each of these parameterizations will lead to a different resulting manifold. Depending on the representation, the resulting Riemannian structure can have varying degrees of complexity and numerical techniques may be required to compute geodesics on the manifold. For example, [3] chooses the log density representation and uses a shooting technique to find geodesics on this space. However, this space has a complicated Riemannian structure and the numerical method used in [3] sometimes leads to large errors. Therefore, the natural question to ask now is, is it possible to use a re-parameterization such that the resulting manifold is simple and the Riemannian operations are easy, preferably closed-form, to compute? Once an efficient representation is found, the corresponding Fisher-Rao metric, which depends on the tangent vector, will then be used as the Riemannian metric.

In a recent work [5], it is proved that by using the square-root representation, the resulting manifold is a unit sphere in a Hilbert space with the Fisher-Rao metric being the usual $\mathbb{L}^2$ metric. Therefore, the various Riemannian operations such as geodesics, exponential maps, logarithmic maps are available in closed form. This is the most efficient representation found to date. The square-root density function is defined as $\boldsymbol{\psi} = \sqrt{\boldsymbol{p}}$, where $\boldsymbol{\psi}$ is assumed to be non-negative to ensure uniqueness. The space of such functions is defined as:

$$\boldsymbol{\Psi} = \{\boldsymbol{\psi} : [0,T] \rightarrow \mathbb{R} | \forall s, \boldsymbol{\psi}(s) \geq 0, \int_0^T \boldsymbol{\psi}^2(s)ds = 1\}. \tag{16}$$

From (16), it is easy to see that the functions $\boldsymbol{\psi}$ lie on a unit sphere. In addition, $\boldsymbol{\Psi}$ forms a convex subset of the unit sphere. The advantage of choosing the square-root density becomes immediately obvious, as many of the Riemannian expressions for the unit sphere are well-known and closed-form. By making use of the representation in (16), we can rewrite (15) and obtain the Fisher-Rao metric as

$$\langle \boldsymbol{v}_j, \boldsymbol{v}_k \rangle_{\boldsymbol{\psi}_i} = \int_0^T \boldsymbol{v}_j(s)\boldsymbol{v}_k(s)ds, \tag{17}$$

where $\boldsymbol{v}_j, \boldsymbol{v}_k \in T_{\boldsymbol{\psi}_i}\boldsymbol{\Psi}$ are tangent vectors. Now, for any two functions $\boldsymbol{\psi}_i, \boldsymbol{\psi}_j \in \boldsymbol{\Psi}$, the geodesic distance between these two points on a unit sphere is simply the angle between them, i.e.,

$$\text{dist}(\boldsymbol{\psi}_i, \boldsymbol{\psi}_j) = \cos^{-1}\langle \boldsymbol{\psi}_i, \boldsymbol{\psi}_j \rangle = \cos^{-1}\big(\int_0^T \boldsymbol{\psi}_i(s)\boldsymbol{\psi}_j(s)ds\big), \tag{18}$$

where $\langle \cdot, \cdot \rangle$ is the normal dot product between points in the sphere under the $\mathbb{L}^2$ metric.

From the differential geometry of the sphere, the exponential map is defined as

$$\exp_{\boldsymbol{\psi}_i}(\boldsymbol{v}) = \cos(\|\boldsymbol{v}\|_{\boldsymbol{\psi}_i})\boldsymbol{\psi}_i + \sin(\|\boldsymbol{v}\|_{\boldsymbol{\psi}_i})\frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_{\boldsymbol{\psi}_i}}, \tag{19}$$

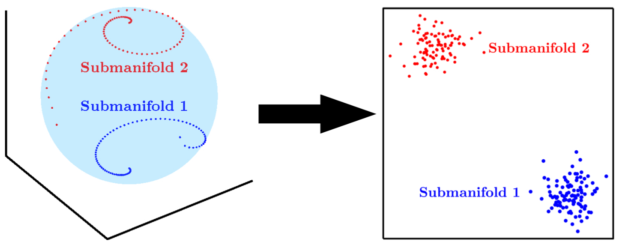


Fig. 2. Illustration of how probability density functions are clustered using our algorithm. Each point in the manifold denotes a pdf and different groups are mapped into different clusters.

where $\boldsymbol{v} \in T_{\boldsymbol{\psi}_i}(\boldsymbol{\Psi})$ is a tangent vector at $\boldsymbol{\psi}_i$ and $\|\boldsymbol{v}\|_{\boldsymbol{\psi}_i} = \sqrt{\langle \boldsymbol{v}, \boldsymbol{v}\rangle_{\boldsymbol{\psi}_i}} = (\int_0^T \boldsymbol{v}(s)\boldsymbol{v}(s)ds)^{\frac{1}{2}}$. In order to ensure that the exponential map is a bijective function, we restrict $\|\boldsymbol{v}\|_{\boldsymbol{\psi}_i} \in [0, \pi]$. The logarithm map from $\boldsymbol{\psi}_i$ to $\boldsymbol{\psi}_j$ is then given by

$$\overrightarrow{\boldsymbol{\psi}_i\boldsymbol{\psi}_j} = \log_{\boldsymbol{\psi}_i}(\boldsymbol{\psi}_j) = \frac{\mathbf{u}}{(\int_0^T \mathbf{u}(s)\mathbf{u}(s)ds)^{\frac{1}{2}}} \cos^{-1}\langle \boldsymbol{\psi}_i, \boldsymbol{\psi}_j\rangle, \tag{20}$$

with $\mathbf{u} = \boldsymbol{\psi}_j - \langle \boldsymbol{\psi}_i, \boldsymbol{\psi}_j\rangle \boldsymbol{\psi}_i$.

By substituting the closed-form formulae in this section into the respective operations in Algorithm 3, it is immediately clear that we are able to perform unsupervised clustering of probability density functions. Fig. 2 illustrates the overall idea of our approach.

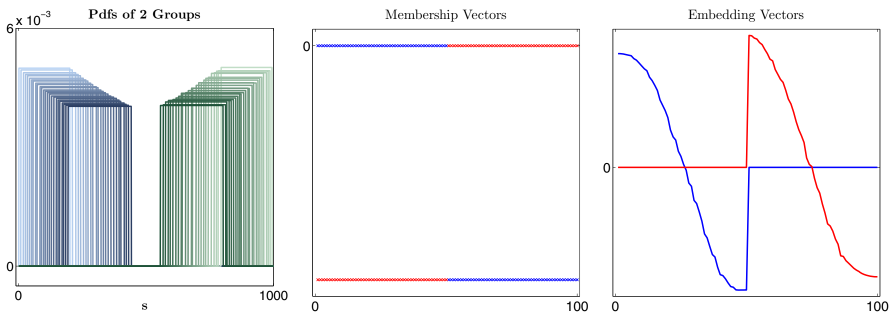
5 Experiments

In this section, we evaluate the performance of the proposed algorithm on both synthetic and real data. Experiments on synthetic data are performed on mixtures of uniform pdfs, while experiments on real data involve the segmentation of images based on texture.

5.1 Synthetic Examples

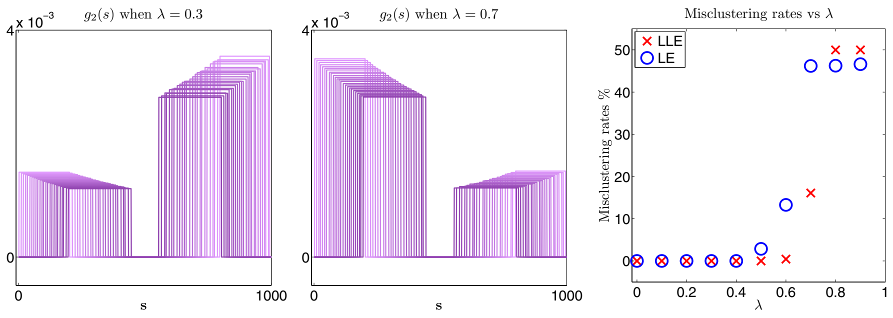
We will first evaluate the performance of Algorithm 3 for clustering two groups of uniform pdfs with 50 pdfs in each group. Fig. 3(a) shows these two groups of pdfs, with the first group f_1 in blue and the second group f_2 in green, defined on the interval $[0, 1000]$. Each different shade of blue or green denotes a different element of its group. Both groups are generated by shifting the intervals in which the probability is not zero and increasing the bandwidths. Let $\mathcal{U}[\alpha, \beta]$ be the uniform distribution on the interval $[\alpha, \beta]$. The pdfs in f_1 are $f_{1,i} = \mathcal{U}[a_i, b_i], i = 1, \ldots, 50$, where $a_i = 4(i-1) + \lambda_1$, $b_i = 195 + 5i + \lambda_2$. The pdfs in f_2 are $f_{2,j} = \mathcal{U}[c_j, d_j], j = 1, \ldots, 50$, where $c_j = 805 - 5j - \lambda_3$, $d_j = 1004 - 4j - \lambda_4$. $\{\lambda_k\}_{k=1}^4$ are drawn from $\mathcal{U}[0, 4]$. Figs. 3(b)-3(c) show that when we apply Riemannian LLE, the two smallest eigenvectors indicate the membership of each group whereas the next two eigenvectors are the embedding vectors.

Next, we validate the performance of our algorithm on two groups of pdfs, one with uniform distributions and the other one with mixtures of uniform distributions. The



(a) 2 groups of uniform pdfs (b) Membership eigenvectors (c) Embedding eigenvectors

Fig. 3. Applying Riemannian LLE to clustering two groups of uniform pdfs shown in (a). Both pdfs are uniform distributions with shifting centers and varying bandwidths.



(a) (b) (c)

Fig. 4. Clustering two groups of pdfs $g_1 = f_1$ and $g_2 = \lambda f_1 + (1-\lambda) f_2$. Figs. 4(a)-4(b) show g_2 when $\lambda = 0.3, 0.7$. Fig. 4(c) shows the misclustering rates of LLE and LE when λ varies.

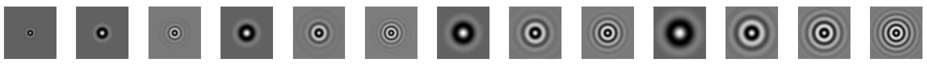


Fig. 5. Schmid filter bank that we use to generate the textons and in turn the histograms

groups have 50 pdfs each and are constructed as follows. Let f_1 be the first set of pdfs in blue shown in Fig. 3(a) and f_2 be the second set in green. We set the first group to $g_1 = f_1$ and the second group to $g_2 = \lambda f_1 + (1-\lambda) f_2$, where $\lambda \in [0, 1]$. Figs. 4(a)-4(b) show g_2 when λ is equal to 0.3 and 0.7. Since noise is introduced in the generation of f_1 and f_2, we repeat this experiment over 500 trials. It is easy to see that when λ approaches 1, the group g_2 merges into g_1. From Fig. 4(c), we see that when λ is small, the misclustering rate is 0%. However, as λ approaches 1, the distance between g_1 and g_2 decreases and the misclustering approaches 50%.

Table 2. Misclustering rates in % for two-class segmentation

Algorithm	Set 1	Set 2	Set 3	Set 4
Riemannian LLE	0	0	1.63	0
Riemannian LE	0	0	19.68	22.9

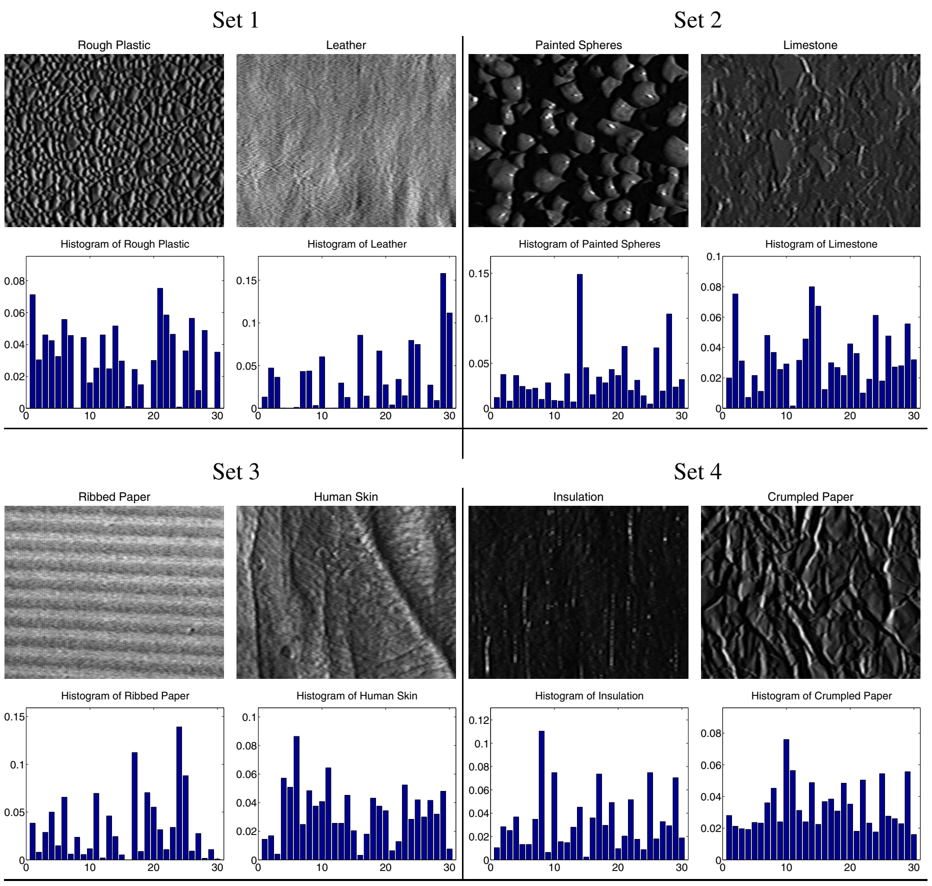


Fig. 6. Textures and corresponding histograms used in the two-class clustering experiments

We test our algorithm on 4 sets of data containing 2 different textures each. There are 92 images in each texture class. In these experiments, the number of nearest neighbors is set to 10. Fig. 6 shows these 4 sets with a typical example of the 2 different textures and the corresponding histograms in each set. Table 2 shows the misclustering percentage of LLE and LE for each set.

5.2 Texture Clustering

We also test our proposed algorithm in the segmentation of different textures. From the Columbia-Utrecht Reflectance and Texture Database (CUReT) found at `http://www1.cs.columbia.edu/CAVE//software/curet/`, we obtain samples of different textures and each grayscale image contains only one texture. In order to construct a histogram that reflects the texture statistics in an image, we will calculate what is commonly known as *textons* [20]. This is done by first applying a filter bank to all images in the training set. We use the Schmid [21] filter banks shown in Fig. 5. This will provide us with a feature vector $f(x, y)$ of dimension 13 at each pixel. Next, we apply k-means to all the vectors in the entire dataset to get 30 cluster centers, also known as the textons. For each image in the dataset, we then compute a histogram that contains the number of pixels corresponding to each one of these 30 bins. This is done by assigning a pixel (x, y) to bin i if the feature vector $f(x, y)$ is closest to cluster center $i = 1, \ldots, 30$, according to the Euclidean distance in $\mathbb{R}^{13}$.

Finally, we test our algorithm on a set of data containing 3 different textures. Fig. 7 shows a typical example of the different textures and the corresponding histograms in each set. The error produced by LLE in clustering is 5.43% whereas LE is significantly higher at 30.07%.

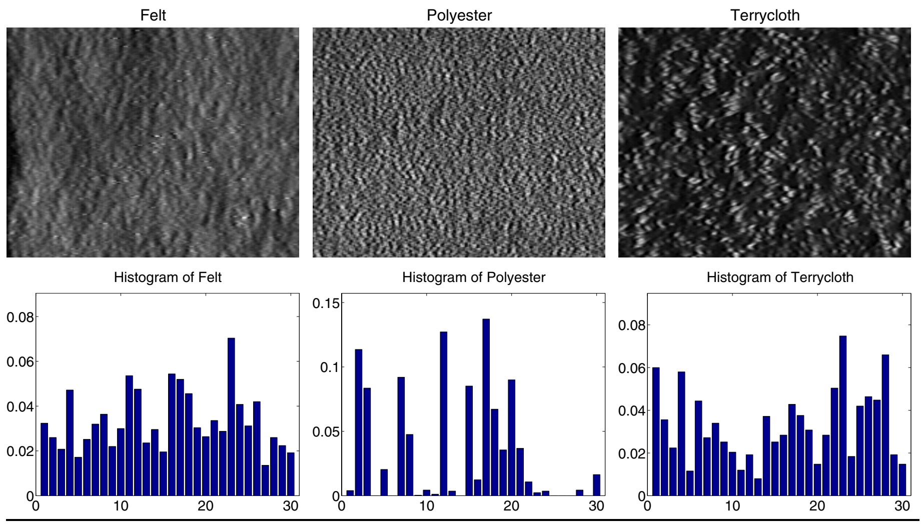


Fig. 7. Textures and corresponding histograms used in the three-class clustering experiment

6 Conclusion

We presented an algorithm to perform clustering of probability density functions. Our method takes into consideration the Riemannian structure of the square-root representation. Results on synthetic and real data are encouraging.

Acknowledgments. The authors thank Rizwan A. Chaudhry for useful discussions. This work has been supported by startup funds from JHU, by grants NSF CAREER IIS-0447739, NSF EHS-0509101, and ONR N00014-05-10836, and by contract JHU APL-934652.

References

1. Rao, C.R.: Information and accuracy attainable in the estimation of statistical parameters. Bull. Calcutta Math. Soc. 37, 81–89 (1945)
2. Amari, S.: Differential-Geometrical Methods in Statistics. Springer, Heidelberg (1985)
3. Mio, W., Badlyans, D., Liu, X.: A computational approach to Fisher information geometry with applications to image analysis. In: Rangarajan, A., Vemuri, B.C., Yuille, A.L. (eds.) EMMCVPR 2005. LNCS, vol. 3757, pp. 18–33. Springer, Heidelberg (2005)
4. Srivastava, A., Joshi, S., Mio, W., Liu, X.: Statistical shape analysis: clustering, learning, and testing. IEEE Transactions on PAMI 27(4), 590–602 (2005)
5. Srivastava, A., Jermyn, I., Joshi, S.: Riemannian analysis of probability density functions with applications in vision. In: IEEE CVPR (2007)
6. Hotelling, H.: Analysis of a complex of statistical variables into principal components. Journal of Educational Psychology 24, 417–441 (1933)
7. Tenenbaum, J.B., de Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science 290(5500), 2319–2323 (2000)
8. Schölkopf, B., Smola, A.: Learning with Kernels. MIT Press, Cambridge (2002)
9. Roweis, S., Saul, L.: Think globally, fit locally: Unsupervised learning of low dimensional manifolds. Journal of Machine Learning Research 4, 119–155 (2003)
10. Belkin, M., Niyogi, P.: Laplacian eigenmaps and spectral techniques for embedding and clustering. In: Neural Information Processing Systems, pp. 585–591 (2002)
11. Donoho, D., Grimes, C.: Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. PNAS 100(10), 5591–5596 (2003)
12. Goh, A., Vidal, R.: Clustering and dimensionality reduction on Riemannian manifolds. In: IEEE CVPR (2008)
13. do Carmo, M.P.: Riemannian Geometry. Birkhäuser, Boston (1992)
14. Frechet, M.: Les elements aleatoires de nature quelconque dans un espace distancie. Annales De L'Institut Henri Poincare 10, 235–310 (1948)
15. Fletcher, P.T., Joshi, S.: Riemannian geometry for the statistical analysis of diffusion tensor data. Signal Processing 87(2) (2007)
16. Pennec, X., Fillard, P., Ayache, N.: A Riemannian framework for tensor computing. International Journal of Computer Vision 66(1), 41–46 (2006)
17. Karcher, H.: Riemannian center of mass and mollifier smoothing. Communications on Pure and Applied Mathematics 30(5), 509–541 (1977)
18. Horn, R., Johnson, C.: Matrix Analysis. Cambridge University Press, Cambridge (1985)
19. Cencov, N.N.: Statistical decision rules and optimal inference. In: Translations of Mathematical Monographs, vol. 53. AMS (1982)
20. Varma, M., Zisserman, A.: A statistical approach to texture classification from single images. International Journal of Computer Vision 62(1-2), 61–81 (2005)
21. Schmid, C.: Constructing models for content-based image retrieval. In: IEEE Conference on Computer Vision and Pattern Recognition (2001)