High Dimensional Pattern Recognition using Diffusion Maps and Earth Mover's Distance

Linh Lieu[*,a], Naoki Saito[**,a]

[a] *Department of Mathematics, University of California, Davis, CA 95616, USA*

Abstract

We propose a new method for matching, comparing, and discriminating datasets consisting of high-dimensional data (e.g., signals and images). Our approach first performs dimension reduction and feature extraction of training datasets using the diffusion maps developed by Coifman and Lafon [1, 2]. This leads to a compact representation of the given classes in the so-called "diffusion space" whose dimension is much lower than the original ambient space. In fact, each class in the diffusion space is represented as a set of cluster centroids called "signatures". Dimension reduction via diffusion maps offers the advantage of preserving the underlying geometry in the data. To classify an unlabeled test dataset, we extend (or embed) that dataset into the diffusion space constructed during the training stage, construct its signature, and then measure the "closeness" or similarity between the test signature and the class signatures using the Earth Mover's Distance (EMD) [3, 4], which is more robust than other measures. Finally, we will demonstrate the usefulness of our method using two practical real applications and compare the performance of the dimension reduction capability of our method with that of the standard Principal Component Analysis.

Key words: diffusion maps, graph Laplacian, Earth Mover's Distance

[*] Principal corresponding author

[**] Corresponding author

Email addresses: llieu@math.ucdavis.edu (Linh Lieu), saito@math.ucdavis.edu (Naoki Saito)

1. Introduction

Many problems in pattern recognition require analysis and comparison between datasets instead of those between individual data points. For example, in a visual speech recognition, two or more clips of recorded video are compared for similar patterns. Typically each video clip consists of a sequence of image frames. We can view each image as one data point and a video clip as a set of points in the image space. Thus, comparing two video clips is the same as comparing two sets of points in the image space. For a second example, consider the task of identifying an object on the ocean floor from a set of sonar waveforms reflected from the object. This requires comparing the set of waveforms of the unknown object to sets of waveforms of known objects. Then the object is identified if a match is made.

In this paper, we propose a method for classification problems where the data corresponding to an object is a set of points in a high dimensional space. Our proposed method consists of two main steps. The first step involves dimension reduction and feature extraction and the second involves discrimination and classification.

Modern technologies generate data that are often extremely high dimensional: even a small 128×128 image has dimension 16384. The variables describing the data (the data variates), however, are often highly correlated, at least locally. For example, many neighboring pixels in an image are highly correlated. This means that in many cases there exist lower-dimensional structures of the data. In other words, the data have low intrinsic dimensionality, and therefore, it is possible to find a low-dimensional representation for the data. By reducing the data dimensionality, we make analysis of the data much more efficient, and sometimes more accurate.

It is well known that classical algorithms for dimension reduction and feature extraction, such as Principal Component Analysis (PCA), are almost inapplicable to the analysis of high-dimensional data due to the *curse of dimensionality*. That is, their computational cost grows exponentially with the dimension. Moreover, the correlations between the data variates may only be local. Traditional methods such as PCA and Multidimensional Scaling are global methods, thus they may not provide a proper low dimensional representation for the data. M. Belkin and P. Niyogi [5, 6] introduced

the idea of using eigenvectors of the Laplacian on the graph constructed from the data for nonlinear dimension reduction. Here, each data point is treated as a node and the weight on an edge between two nodes represents the affinity between the two data points. Other nonlinear methods have been proposed in [7, 8, 9]. Nonlinear methods offer the advantage of preserving local geometry while achieving dimension reduction. In [6, 10], Belkin and Niyogi analyzed the local-neighborhood-preserving property of Laplacian eigenvectors. Unfortunately, the low-dimensional representation of the data obtained from Laplacian eigenvectors are highly sensitive to the sampling density of data (see [2] for examples). This is a serious drawback. For example, consider two video clips of a person speaking the word 'one' at two different speed. The set of image frames extracted from the slower video clip will have more images than that from the faster one, and thus it is a denser sampling set. We want to be able to identify the two sets of image frames as belonging to the same class. However, sensitivity to sampling density can cause distortion in the low-dimensional representation of the data, leading to high misclassification rate. R. R. Coifman and S. Lafon [1, 2] proposed Laplace-Beltrami normalization of the weights on the graph before constructing the Laplacian matrix. This makes the eigenvectors invariant to sampling density. In [1, 11], Coifman, Lafon, and Lee defined *diffusion maps* from the eigenvalues and eigenvectors of the Laplace-Beltrami normalized Laplacian matrix and provided an intuitive interpretation of how data clustering in a diffusion coordinate system is linked to a Markov chain on the weighted graph. In our proposed method, we shall apply diffusion maps to achieve dimension reduction. Our preference for diffusion maps over other nonlinear methods is mainly grounded in their invariance to sampling densities.

We note that Lafon, Keller, and Coifman have proposed in [12] a method for datasets matching similar to the problems in our consideration. Their algorithm includes using diffusion maps to reduce data dimensionality and then use the Hausdorff distance to measure the difference between any two sets. As we shall show in Section 5 below, Hausdorff distance is very sensitive to outliers. To remedy this drawback, we propose an approach that takes into consideration the distribution of the data in its lower-dimensional representation space (i.e., after dimension reduction has been done). Our approach involves constructing a *signature* (a discrete multidimensional

probability density function, or synonymous, a high-dimensional histogram) of each dataset after having embedded all datasets into a lower-dimensional space using diffusion maps. Then we determine the similarity between datasets by the *closeness* of their signatures in the Earth Mover's Distance (EMD). Our idea of applying EMD to discriminate discrete distributions originates from its success in applications to image retrieval from databases [3, 4].

The remainder of this paper is organized as follows: we first review the Diffusion Framework and Earth Mover's Distance. In Section 4 we describe our proposed method for datasets matching and provide suggestions for selecting basic parameters necessary under the Diffusion Framework. In Section 5 we give two examples of applications of our method together with experimental results. We will also provide a comparison of numerical results of our method and the method in [12]. In order to better understand the conditions under which a nonlinear technique (such as Diffusion Framework) is more appropriated for dimension reduction than a linear technique (such as PCA), we will compare numerical results of diffusion maps and PCA in Section 6. Finally we summarize our discoveries and draw conclusions in the last section.

2. Diffusion Framework

In this section, we review the construction of diffusion maps on a dataset and the properties that allow us to achieve meaningful dimensionality reduction. We will also review an algorithm proposed in [12] for extension of diffusion maps from the training data to the test data.

2.1. Diffusion maps

Diffusion maps are constructed from the eigenfunctions of an averaging operator – the *diffusion operator*. We assume here that our dataset $X = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\}$ lies in a space having a natural dissimilarity measure δ that gives a sense of affinity (or similarity) between any two points in X. This is a reasonable assumption to make in practice. For example, if X is a database of image patches each of which has 32×32 pixels, then δ may be the ℓ^2 norm between two image patches in $\mathbb{R}^{1024}$. Or, if X belongs to a

submanifold in $\mathbb{R}^n$, then δ may be the usual Euclidean distance. Following the work of Coifman and Lafon [1, 2], we construct the diffusion operator on X as follows. View the data points $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N$ as nodes of a weighted symmetric graph. Any two nodes $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are connected by an edge with weight $w_\varepsilon(\boldsymbol{x}_i, \boldsymbol{x}_j) \triangleq \mathrm{e}^{-(\delta(\boldsymbol{x}_i, \boldsymbol{x}_j)/\varepsilon)^2}$, $\varepsilon > 0$. The weight function w_ε gives the notion of local geometry to X. That is, it defines the notion of a local neighborhood at each point $\boldsymbol{x} \in X$ via the affinity between $\boldsymbol{x}$ and other points, and the value of the parameter ε specifies the size of this neighborhood. Moreover, as explained in [6], when the dataset X approximately lies on a submanifold, using the weights w_ε on the graph corresponds to an approximation of the heat kernel on the submanifold.

Applying the so-called graph-Laplacian normalization to w_ε yields the *diffusion kernel*

$$k(\boldsymbol{x}, \boldsymbol{y}) \triangleq \frac{w_\varepsilon(\boldsymbol{x}, \boldsymbol{y})}{d_\varepsilon(\boldsymbol{x})}, \tag{1}$$

where $d_\varepsilon(\boldsymbol{x}) \triangleq \sum_{\boldsymbol{y} \in X} w_\varepsilon(\boldsymbol{x}, \boldsymbol{y})$. The corresponding diffusion operator is

$$Af(\boldsymbol{x}) \triangleq \sum_{\boldsymbol{y} \in X} k(\boldsymbol{x}, \boldsymbol{y}) f(\boldsymbol{y}). \tag{2}$$

The kernel k is non-negative and row-stochastic (i.e., $\sum_{\boldsymbol{y} \in X} k(\boldsymbol{x}, \boldsymbol{y}) = 1$ for all $\boldsymbol{x} \in X$). Hence, it can be viewed as a transition matrix of a Markov process on X. The operator A is an averaging operator, since it is positivity-preserving (i.e., $Af \geq 0$ for any $f \geq 0$) and preserves constant functions. We can interpret the action of the operator A as 'diffusion' of information throughout the graph, and the Markov chain dictates the directions of fast and slow information propagation.

An important idea in the diffusion framework is to take larger powers of the operator A. For $t > 0$, raising the operator A to a power t is equivalent to running the Markov process forward by time t, which can be interpreted as letting information diffuse for a period of time t. The information propagates more easily and quickly among the regions of high affinity than those of low affinity. This is essentially how we can capture the local geometry of the data.

Let $k^{(t)}$ denotes the kernel of the operator A^t – the t^{th} power of the operator A. (Note that $k^{(t)}(\boldsymbol{x}, \boldsymbol{y})$ represents the probability of transition from $\boldsymbol{x}$ to $\boldsymbol{y}$ in t steps.)

The graph is connected by construction, therefore as $t \to +\infty$ the Markov process approaches a unique stationary distribution ϕ_0 [1], i.e, for any $\boldsymbol{x}, \boldsymbol{y} \in X$,

$$\lim_{t \to +\infty} k^{(t)}(\boldsymbol{x}, \boldsymbol{y}) = \phi_0(\boldsymbol{y}).$$

In practice we always work in the discrete setting, therefore we may view the operator A as a matrix whose rows are indexed by $\boldsymbol{x}$ and columns are indexed by $\boldsymbol{y}$. Then the stationary distribution $\boldsymbol{\phi}_0 = [\phi_0(\boldsymbol{x}_1), \ldots, \phi_0(\boldsymbol{x}_N)]$ is the left eigenvector of A corresponding to the top eigenvalue 1, i.e., $\boldsymbol{\phi}_0 A = \boldsymbol{\phi}_0$. It can be easily derived from equations (1) and (2) and the symmetry of w_ε that

$$\phi_0(\boldsymbol{x}) = \frac{d_\varepsilon(\boldsymbol{x})}{\sum_{\boldsymbol{z} \in X} d_\varepsilon(\boldsymbol{z})}.$$

With this, the *diffusion distance* between any two data points $\boldsymbol{x}$ and $\boldsymbol{y}$ is given by

$$\begin{aligned} D_t(\boldsymbol{x}, \boldsymbol{y})^2 &\triangleq \left\| k^{(t)}(\boldsymbol{x}, \cdot) - k^{(t)}(\boldsymbol{y}, \cdot) \right\|^2_{L^2(X, \frac{1}{\phi_0})} \\ &= \sum_{\boldsymbol{z} \in X} \frac{\left(k^{(t)}(\boldsymbol{x}, \boldsymbol{z}) - k^{(t)}(\boldsymbol{y}, \boldsymbol{z})\right)^2}{\phi_0(\boldsymbol{z})}. \end{aligned} \tag{3}$$

This is simply the weighted L^2 distance between $k^{(t)}(\boldsymbol{x}, \cdot)$ and $k^{(t)}(\boldsymbol{y}, \cdot)$. We observed earlier that $k^{(t)}(\boldsymbol{x}, \boldsymbol{z})$ is the probability of transition from $\boldsymbol{x}$ to $\boldsymbol{z}$ in t steps. Therefore it is easy to see that the diffusion distance between $\boldsymbol{x}$ and $\boldsymbol{y}$ measures the difference in how much connected or how strong in affinity these two nodes are to the rest of the graph at time (or step) t. In its definition, the diffusion distance $D_t(\boldsymbol{x}, \boldsymbol{y})$ takes into account all incidences relating $\boldsymbol{x}$ and $\boldsymbol{y}$. Consequently, it is robust to noise perturbations and hence a great tool for extracting the underlying geometry in the dataset X, especially when X is a low dimensional manifold lying in a high-dimensional space.

The diffusion distance is directly related to the eigenvalues and eigenvectors of the matrix A. In practice, we approximate $D_t(\cdot, \cdot)$ by using eigenvalues and eigenvectors of A. To see this, let us first do some preprocessing: conjugate the kernel k with $\sqrt{\phi_0}$ to obtain the symmetric kernel

$$\begin{aligned} \widetilde{k}(\boldsymbol{x}, \boldsymbol{y}) &\triangleq \sqrt{\phi_0(\boldsymbol{x})}\; k(\boldsymbol{x}, \boldsymbol{y})\, \frac{1}{\sqrt{\phi_0(\boldsymbol{y})}} \\ &= \frac{w_\varepsilon(\boldsymbol{x}, \boldsymbol{y})}{\sqrt{d_\varepsilon(\boldsymbol{x})}\sqrt{d_\varepsilon(\boldsymbol{y})}}. \end{aligned} \tag{4}$$

Let $\widetilde{A}$ be the operator with $\widetilde{k}$ as its kernel, i.e.,

$$\widetilde{A}f(\boldsymbol{x}) \triangleq \sum_{\boldsymbol{y}\in X} \widetilde{k}(\boldsymbol{x},\boldsymbol{y})f(\boldsymbol{y}).$$

It shares the same spectrum as A, and eigenvectors of A can be obtained from those of $\widetilde{A}$ via conjugation by $\sqrt{\phi_0}$. Suppose $\{\lambda_\ell\}$ are the eigenvalues (with $|\lambda_0| \geq |\lambda_1| \geq \cdots$) and the corresponding eigenvectors of $\widetilde{A}$ are $\left\{\widetilde{\phi}_\ell\right\}$, then the left and right eigenvectors of A corresponding to λ_ℓ are $\phi_\ell = \widetilde{\phi}_\ell \cdot \sqrt{\phi_0}$ and $\psi_\ell = \widetilde{\phi}_\ell/\sqrt{\phi_0}$, respectively.

The advantage of the operator $\widetilde{A}$ is that it is symmetric, positive semi-definite, and compact. Hence it has a discrete, non-increasing, non-negative spectra: $\lambda_0 = 1 > \lambda_1 \geq \lambda_2 \geq \cdots \geq 0$, and the orthonormal eigenvectors $\left\{\widetilde{\phi}_\ell\right\}$ form a basis for $L^2(X)$ (the eigenvector corresponding to top eigenvalue $\lambda_0 = 1$ is $\widetilde{\phi}_0 = \sqrt{\phi_0}$). The kernel $\widetilde{k}$ has spectral decomposition

$$\widetilde{k}(\boldsymbol{x},\boldsymbol{y}) = \sum_{j\geq 0} \lambda_j \widetilde{\phi}_j(\boldsymbol{x})\widetilde{\phi}_j(\boldsymbol{y}).$$

Hence,

$$k(\boldsymbol{x},\boldsymbol{y}) = \sum_{j\geq 0} \lambda_j \psi_j(\boldsymbol{x})\phi_j(\boldsymbol{y})$$

and

$$k^{(t)}(\boldsymbol{x},\boldsymbol{y}) = \sum_{j\geq 0} \lambda_j^t \psi_j(\boldsymbol{x})\phi_j(\boldsymbol{y}). \tag{5}$$

Now, $\{\phi_\ell\}$ and $\{\psi_\ell\}$ are biorthogonal (i.e., $\sum_{\boldsymbol{z}\in X} \phi_j(\boldsymbol{z})\psi_\ell(\boldsymbol{z}) = \delta_{j\ell}$, where $\delta_{j\ell}$ is the Kronecker delta), and

$$\phi_\ell(\boldsymbol{x}) = \phi_0(\boldsymbol{x})\psi_\ell(\boldsymbol{x}).$$

Thus,

$$\sum_{z\in X} \frac{\phi_j(\boldsymbol{z})\phi_\ell(\boldsymbol{z})}{\phi_0(\boldsymbol{z})} = \sum_{\boldsymbol{z}\in X} \phi_j(\boldsymbol{z})\psi_\ell(\boldsymbol{z}) = \delta_{j\ell}.$$

That is, $\{\phi_\ell\}$ is an orthonormal basis in $L^2(X, 1/\phi_0)$. Therefore, for fixed $\boldsymbol{x}$, the formula (5) can be interpreted as the expansion of the function $k^{(t)}(\boldsymbol{x},\cdot)$ in this basis, and the expansion coefficients are $\{\lambda_j^t\psi_j(\boldsymbol{x})\}$. Consequently, the formula (3) for the diffusion distance reduces to

$$D_t(\boldsymbol{x},\boldsymbol{y})^2 = \sum_{j\geq 1} \lambda_j^{2t} \left(\psi_j(\boldsymbol{x}) - \psi_j(\boldsymbol{y})\right)^2. \tag{6}$$

Note that $\psi_j(\boldsymbol{x}) = \widetilde{\phi}_j(\boldsymbol{x})/\widetilde{\phi}_0(\boldsymbol{x})$. In other words, the proper diffusion distance can be obtained by the eigenanalysis of the symmetrized operator $\widetilde{A}$ with the kernel $\widetilde{k}$ instead of the original averaging operator A with the transition kernel k. The summation in (6) starts at index $j = 1$ because $\psi_0 \equiv 1$.

In practice, we approximate the diffusion distance formula (6) by the following consideration. Since the eigenvalues λ_j's are non-increasing, the diffusion distance can be approximated to a relative accuracy $\tau > 0$ specified by the user by

$$D_t(\boldsymbol{x}, \boldsymbol{y})^2 \approx \sum_{j=1}^{s(\tau,t)} \lambda_j^{2t} \left(\psi_j(\boldsymbol{x}) - \psi_j(\boldsymbol{y})\right)^2, \tag{7}$$

where

$$s(\tau, t) \stackrel{\Delta}{=} \arg\max_{j \in \mathbb{N}} \{|\lambda_j|^t > \tau |\lambda_1|^t\}. \tag{8}$$

From this, the *diffusion map* is defined as

$$\Psi_t : \boldsymbol{x} \mapsto \begin{pmatrix} \lambda_1^t \psi_1(\boldsymbol{x}) \\ \lambda_2^t \psi_2(\boldsymbol{x}) \\ \vdots \\ \lambda_{s(\tau,t)}^t \psi_{s(\tau,t)}(\boldsymbol{x}) \end{pmatrix}. \tag{9}$$

It can be viewed as coordinates in a $s(\tau, t)$-dimensional Euclidean space characterized by the parameters ε, t, and τ. We shall call this space a *diffusion space*. Note that $s(\tau, t) \ll N$ and in our numerical experiments in Section 5, the values of $s(\tau, t)$ lie in the range $9 \leq s(\tau, t) \leq 18$ while $96 \leq N \leq 128$ in the problem involving classification of underwater objects and $7 \leq s(\tau, t) \leq 15$ while $990 \leq N \leq 1204$ in the lip-reading experiment (The value of $s(\tau, t)$ and N changes as we repeat each experiment using a different training set).

We use Ψ_t to embed our dataset into a diffusion space denoted by $\mathbb{R}^{s(\tau,t)}$. Note that *the usual Euclidean distance in this diffusion space is an approximation to the diffusion distance*. The key point here is that the diffusion map Ψ_t produces a low-dimensional representation of the data that highlights the underlying intrinsic local geometry in the data.

The final important thing to mention is the Laplace-Beltrami normalization of the edge weights $w_\varepsilon(\boldsymbol{x}, \boldsymbol{y})$ if the data X approximately lies on a submanifold $\mathcal{M}$ of $\mathbb{R}^n$

[1, 2, 12]. In this case, we replace w_ε with a normalized version

$$w_\varepsilon(\boldsymbol{x}, \boldsymbol{y}) \longleftarrow \frac{w_\varepsilon(\boldsymbol{x}, \boldsymbol{y})}{d_\varepsilon(\boldsymbol{x}) d_\varepsilon(\boldsymbol{y})}.$$

Then proceed to construct diffusion kernel as described in (1) above. In other words, we normalize the weights twice to construct diffusion kernel: first, the above Laplace-Beltrami normalization, and second, the graph Laplacian normalization. When the data points are sampled from $\mathcal{M}$ in a nonuniform manner, this normalization makes the transition matrix A approximate the Laplace-Beltrami diffusion operator on $\mathcal{M}$ and the embedding of the data points via diffusion maps invariant to the density distribution of the sampled data. In short, the Laplace-Beltrami normalization produces a spectral embedding that depends only on the geometry of $\mathcal{M}$ and not the density of the sampled data points.

2.2. Extension of diffusion maps to test data

Our main interest is in classifying newly obtained unlabeled data (also called *test* data) based on a classification rule learned from the labeled data at hand (i.e., *training* data). In order to make meaningful inference from the training data to the unlabeled test data, we need to have the same low-dimensional representation for both datasets. That is, we need to embed test data into the same diffusion space as the training data. Hence, it becomes necessary for us to extend the diffusion map computed on the training dataset to the test data. To perform this task, we employ the multiscale extension scheme proposed in [12], which is based on "geometric harmonics" originally introduced in [2, Chap. 3] and [15]. Let us call this scheme GHME (geometric harmonics multiscale extension) for short. We now review the GHME scheme.

The GHME scheme is an improvement of the Nyström extension method proposed in [16, 17]. Let X and Y denote the training set and the unlabeled test set, respectively. First consider the eigenvalues $\{\mu_\ell\}$ and orthonormal eigenfunctions $\{\varphi_\ell\}$ of a (symmetric) Gaussian kernel of width $\sigma > 0$ on the training set X:

$$\mu_\ell \varphi_\ell(\boldsymbol{x}) = \sum_{\boldsymbol{z} \in X} \mathrm{e}^{-\|\boldsymbol{x}-\boldsymbol{z}\|^2/\sigma^2} \varphi_\ell(\boldsymbol{z}), \quad \boldsymbol{x} \in X, \tag{10}$$

where the nonnegative eigenvalues $\{\mu_\ell\}$ are sorted in decreasing order. From Equation (10), the Nyström extension of φ_ℓ from X to $\boldsymbol{y} \in Y$ is defined as

$$\bar{\varphi}_\ell(\boldsymbol{y}) \triangleq \frac{1}{\mu_\ell} \sum_{\boldsymbol{z} \in X} \mathrm{e}^{-\|\boldsymbol{y}-\boldsymbol{z}\|^2/\sigma^2} \varphi_\ell(\boldsymbol{z}). \tag{11}$$

Since the eigenfunctions $\{\varphi_\ell\}$ form an orthonormal basis for $L^2(X)$, any function $f \in L^2(X)$ can be expanded as

$$f(\boldsymbol{x}) = \sum_\ell \langle f, \varphi_\ell \rangle \, \varphi_\ell(\boldsymbol{x}), \ \boldsymbol{x} \in X.$$

Thus the Nyström extension of f from X to $\boldsymbol{y} \in Y$ can be defined as

$$\bar{f}(\boldsymbol{y}) \triangleq \sum_\ell \langle f, \varphi_\ell \rangle \, \bar{\varphi}_\ell(\boldsymbol{y}).$$

We observe that the range of the extension in (11) is proportional to σ. If the ratio $\|\boldsymbol{x} - \boldsymbol{y}\|/\sigma$ is large for all $\boldsymbol{x} \in X$, then $\bar{\varphi}_\ell(\boldsymbol{y})$ will be numerically small and hence may not be meaningful. Hence the extension scale σ should be as large as possible. However, for large enough σ, the Gaussian kernel in (10) becomes ill-conditioned, i.e., μ_ℓ tends to 0 more quickly compared to the case where σ is small. Thus the Nyström extension in (11) will blow up. Furthermore, it is well known that the extension range depends on the smoothness of the function to be extended [2, Chap. 3], [15]. If f is fairly smooth, it can be extended far away from the training set. On the other hand, if f varies wildly on X, then it has limited extension range. To address the ill-condition issue, the GHME scheme considers the following approximate extension for f:

$$\bar{f}(\boldsymbol{z}) \triangleq \sum_{\ell:\ \eta\mu_\ell > \mu_0} \langle f, \varphi_\ell \rangle \, \bar{\varphi}_\ell(\boldsymbol{z}), \tag{12}$$

where $\eta > 0$ is some fixed condition number and $\boldsymbol{z} \in X \cup Y$. This extension $\bar{f}$ is well-defined on $X \cup Y$, but it is not equal to f on the training set X. Observe that if the value of σ decreases, the eigenvalues $\mu_\ell \to 0$ more slowly. This allows more terms in (12), making $\bar{f}$ a better approximation of f on X. Based on this observation, the GHME iteratively searches for an extension $\bar{f}$ that approximates f on X with an pre-set error tolerance $\varrho > 0$ by slowly decreasing the value of the extension scale σ.

The GHME scheme is summarized as follows:

Step 1: Suppose f is a function defined on the training set X and to be extended to a new dataset Y. Fix a condition number $\eta > 0$ and an error tolerance $\varrho > 0$. Set the extension scale $\sigma = \sigma_0$, for some large value σ_0.

Step 2: Compute eigenvalues $\{\mu_\ell\}$ and orthonormal eigenfunctions $\{\varphi_\ell\}$ of the Gaussian kernel of width σ and expand f (on the training set X) in this eigenbasis

$$f(\boldsymbol{x}) = \sum_\ell \langle f, \varphi_\ell \rangle \, \varphi_\ell(\boldsymbol{x}), \;\; \boldsymbol{x} \in X,$$

i.e., compute the coefficients $c_\ell \triangleq \langle f, \varphi_\ell \rangle$.

Step 3: On the training set X, approximate f by $\bar{f}$ defined in (12). Compute the approximation error

$$Err \triangleq \left(\sum_{\ell:\ \mu_0/\mu_\ell \geq \eta} |c_\ell|^2 \right)^{1/2}.$$

If $Err > \varrho$, set $\sigma \leftarrow \frac{1}{2}\sigma$ and return to Step 2. Otherwise, continue.

Step 4: For each ℓ such that $\mu_0/\mu_\ell < \eta$, compute the Nyström extension

$$\bar{\varphi}_\ell(\boldsymbol{y}) = \frac{1}{\mu_\ell} \sum_{\boldsymbol{x} \in X} \mathrm{e}^{-\|\boldsymbol{x}-\boldsymbol{y}\|^2/\sigma^2} \varphi_\ell(\boldsymbol{x}),$$

for all $\boldsymbol{y} \in Y$. And finally, compute the approximate extension $\bar{f}$

$$\bar{f}(\boldsymbol{y}) \triangleq \sum_{\ell:\ \mu_0/\mu_\ell < \eta} c_\ell \bar{\varphi}_\ell(\boldsymbol{y}).$$

3. Earth Mover's Distance

The definition of the Earth Mover's Distance (EMD) is based on the solution to a discrete *optimal mass transportation problem*. EMD represents the minimum cost of moving earth (or sand) from some source locations to fill up holes at some sink locations. In other words, given any two distributions of materials (or probability distributions), one of them can be viewed as a distribution of earth and the other a distribution of holes, then EMD between the two distributions is the minimum cost of rearranging the mass in one distribution to obtain the other. In the continuous setting, this problem is known

as the *Monge-Kantorovich optimal mass transfer* problem and has been well studied over the past 100 years; for an introductory reading on the problem, see e.g., [18]. The importance here is that EMD can be applied to measure the discrepancy between two multidimensional distributions.

In the discrete setting, the optimal mass transfer problem can be formulated as a linear optimization problem as follows [3, 4]: Suppose we have a source mass distribution $P = \{(\boldsymbol{p}_1, w_{\boldsymbol{p}_1}), \cdots, (\boldsymbol{p}_m, w_{\boldsymbol{p}_m})\}$ and a sink distribution $Q = \{(\boldsymbol{q}_1, w_{\boldsymbol{q}_1}), \cdots, (\boldsymbol{q}_n, w_{\boldsymbol{q}_n})\}$ in a high-dimensional space $\mathbb{R}^s$. In this setting, P and Q are called *signatures* and can be viewed as two distributions of feature vectors representing two objects. P is a signature of one object that consists of m clusters in $\mathbb{R}^s$ where $\boldsymbol{p}_i$ is the centroid of the ith cluster and $w_{\boldsymbol{p}_i}$ is the proportion of the object's feature vectors that belongs to the ith cluster. Similarly, Q is a signature of another object that consists of n clusters with the cluster centroid and the weight pairs $(\boldsymbol{q}_j, w_{\boldsymbol{q}_j})$, $j = 1, \ldots, n$.

Suppose the cost of moving one unit of mass from $\boldsymbol{p}_i$ to $\boldsymbol{q}_j$ is $c(\boldsymbol{p}_i, \boldsymbol{q}_j)$, and f_{ij} denotes the amount of mass flow from $\boldsymbol{p}_i$ to $\boldsymbol{q}_j$. Then, the transportation cost can be defined as:

$$\text{COST}(P, Q, \boldsymbol{F}) \triangleq \sum_{i=1}^{m} \sum_{j=1}^{n} c(\boldsymbol{p}_i, \boldsymbol{q}_j) f_{ij},$$

where $\boldsymbol{F} \triangleq [f_{ij}] \in \mathbb{R}^{m \times n}$. Then, the optimal mass transfer problem seeks the flow $\boldsymbol{F}^*$ that transfers the maximum allowable amount of earth to fill up the holes with minimum total transportation cost, i.e.,

$$\boldsymbol{F}^* = \arg\min_{\boldsymbol{F} \in S} \text{COST}(P, Q, \boldsymbol{F}),$$

where $\boldsymbol{F} \in S \subset \mathbb{R}^{m \times n}$ means that $\boldsymbol{F}$ must satisfy the following constraints:

(i) $f_{ij} \geq 0$, for all i, j;

(ii) $\sum_{j=1}^{n} f_{ij} \leq w_{\boldsymbol{p}_i}$, for all $1 \leq i \leq m$;

(iii) $\sum_{i=1}^{m} f_{ij} \leq w_{\boldsymbol{q}_j}$, for all $1 \leq j \leq n$; and

(iv) $\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij} = \min\left(\sum_{i=1}^{m} w_{\boldsymbol{p}_i}, \sum_{j=1}^{n} w_{\boldsymbol{q}_j}\right)$.

The constraint (i) ensures that one can only move earth from P to Q, not vice versa; (ii) that the amount of earth moved from P is no more than the sum of the weights $w_{\boldsymbol{p}_i}$; (iii) that the amount of earth received at Q is no more than the sum of the weights $w_{\boldsymbol{q}_j}$; and (iv) that the maximum allowable amount of earth is moved.

Once the optimal flow $\boldsymbol{F}^*$ from P to Q is found, EMD is then defined as the total cost normalized by the total flow:

$$\begin{aligned}\text{EMD}(P,Q) &\triangleq \frac{\text{COST}(P,Q,\boldsymbol{F}^*)}{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij}^*} \\ &= \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} c(\boldsymbol{p}_i,\boldsymbol{q}_j) f_{ij}^*}{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij}^*}.\end{aligned}$$

Notice that the normalization factor is the total weight of the smaller signature due to the constraint (iv). This normalization ensures that smaller signatures are not favored in the case when two signatures have different total weights. Furthermore, EMD is symmetric, i.e., $\text{EMD}(P,Q) = \text{EMD}(Q,P)$ for any two distributions P and Q.

4. Datasets Matching with Diffusion Maps and EMD

We now describe how diffusion maps and Earth Mover's Distance can be applied together to perform datasets matching. Our approach quantitatively determines the dissimilarity between any two sets of points of high dimensional nature (each set corresponds to an object). Our idea is the following: first, perform dimension reduction and feature extraction under the diffusion framework; then apply Earth Mover's Distance as a discriminant measure between sets; and finally, classify unlabeled sets via nearest neighbor in EMD distance.

4.1. Signature construction

As explained in Section 3, Earth Mover's Distance measures the discrepancy between two discrete distributions. To apply EMD as a datasets-discrimination measure, we need to construct a signature for each dataset. This involves two steps: first, cluster the feature vectors in the diffusion space; and second, form the signature $P = \{(\boldsymbol{p}_1, w_{\boldsymbol{p}_1}), \cdots, (\boldsymbol{p}_m, w_{\boldsymbol{p}_m})\}$, where $\{\boldsymbol{p}_j\}_{j=1}^m$ are cluster centroids and $w_{\boldsymbol{p}_j}$

is the density of cluster j, that is, the percentage of all feature vectors in the dataset that belong to cluster j.

For the clustering step, we apply the *Elongated K-means* algorithm [19]. Elongated K-means (*ekmeans*) was adapted from the original K-means algorithm by replacing the Euclidean distance with an Elongated distance in the computation of point-to-centroids distances. It was designed for detecting elongated (thin and long) clusters. More specifically, *ekmeans* algorithm would group points lying inside a thin long ellipsoid to form a cluster, as opposed to inside a sphere.

To motivate the consideration of *ekmeans*, let us examine the ideal scenario when the data consists of K clusters widely separated from each other [19]. In this case the matrix of $\widetilde{k}(\boldsymbol{x}, \boldsymbol{y})$ described in Section 2.1 (with rows re-ordered by clusters if necessary) is block diagonal with exactly K blocks. Thus, it has K eigenvectors associated with the largest eigenvalue 1, one eigenvector for each cluster. Each eigenvector has ones in the entries corresponding to the points in the cluster and zeros elsewhere. Suppose we perform a spectral embedding of the data into the top K eigenspace (i.e., the space spanned by the top K eigenvectors). The data would get mapped to K clusters at the K unit vectors on the coordinate axes. In general, rotations may occur, depending on the computation of the eigenvectors. In other words, any set of K mutually orthogonal vectors in the top K eigenspace is an admissible set of eigenvectors associated with eigenvalue 1. Furthermore, eigenvectors are usually normalized. These two facts translate to K elongated clusters lying along some K mutually orthogonal axes within the top K eigenspace (instead of on the coordinate axes).

We observe that when the data is embedded into the top q eigenspace with $q < K$, or equivalently, when we project the K elongated clusters down to the q-dimensional subspace spanned by the first q eigenvectors, the results are elongated clusters lying along radial directions and possibly some dense clusters near the origin. These clusters near the origin are the projection image of those clusters that lay elongated along the directions orthogonal to this q-dimensional subspace. On the other hand, suppose we embed the data into the top q eigenspace with $q > K$. We would find no additional cluster other than the K elongated clusters already accounted for. The reason behind this phenomenon is that each of the eigenvectors after the K^{th} eigenvector contains

mostly small numbers. In other words, large separations in the data are already captured in the top K eigenspace. Consequently, increasing the dimension of the embedding spectral space does not affect the clustering behaviors of the feature vectors.

ekmeans exploits the geometric properties of the eigenvectors of $\widetilde{k}(\boldsymbol{x}, \boldsymbol{y})$ to cluster the data and automatically determine the number of intrinsic clusters. That is, *ekmeans* does not require input of the number of clusters. To determine the number of intrinsic clusters automatically, *ekmeans* starts the clustering process in the top 2 eigenspace with three centroids initialized, two centroids at two different elongated clusters and one at the origin. If there are more than two elongated clusters, the centroid at the origin will be dragged to a cluster not accounted for. Then the algorithm moves the clustering process up to the top 3 eigenspace, adds a centroid at the origin, and repeats the process until no additional cluster is found. This clustering process stops at the top K eigenspace if there are K (intrinsic) clusters in the data.

In practice, the data we handle are usually not widely separated, and we do not have more than one eigenvector associated with the largest eigenvalue 1. Indeed, whether we can construct a diffusion map from eigenvectors of $\widetilde{k}(\boldsymbol{x}, \boldsymbol{y})$ depends solely on the uniqueness of the top eigenvector. So $\widetilde{k}(\boldsymbol{x}, \boldsymbol{y})$ will always have exactly one eigenvector associated with eigenvalue 1. However, the eigenvectors still have similar geometric properties if there are tight clusters in the data which causes a spectral gap at the $K + 1$-th eigenvalue (see [20]). Consequently, the diffusion maps (constructed from eigenvectors of $\widetilde{k}(\boldsymbol{x}, \boldsymbol{y})$ as defined in formula (9)) also have similar geometric properties. Therefore, *ekmeans* is applicable to our problem. Moreover, our accomplishment in utilizing *ekmeans* is twofold: to cluster the feature vectors and to determine the intrinsic dimensionality of the data. Since *ekmeans* determines the intrinsic number of clusters in the data based on geometric properties, this number can be considered as the intrinsic dimensionality of the data. We take advantage of this aspect of *ekmeans* to determine the dimension $s(\tau, t)$ of our diffusion space $\mathbb{R}^{s(\tau,t)}$. We explain how to select $s(\tau, t)$ in more details in Section 4.2.

Now, suppose we have N datasets, $X^1, \cdots, X^N$, and *ekmeans* has determined that each set X^i has K_i clusters. This means the cluster centroids for set X^i determined by *ekmeans* are vectors in the K_i-dimensional subspace of the embedding diffusion space

$\mathbb{R}^{s(\tau,t)}$ (where $s(\tau,t)$ is determined as described in Section 4.2). To bring all signatures into the same space $\mathbb{R}^{s(\tau,t)}$, we run K-means with Elongated distance once on each set of feature vectors in $\mathbb{R}^{s(\tau,t)}$ to reform the K_i clusters for each dataset X^i. We use the previous clustering result as an initial condition for the re-clustering process. With this, the re-clustering process converges very quickly.

4.2. Parameters selection

There are four parameters to be determined in the diffusion framework: the time scale ε for the diffusion kernel $k(\boldsymbol{x},\boldsymbol{y})$ defined in (1), the dimension $s(\tau,t)$ of the embedding diffusion space $\mathbb{R}^{s(\tau,t)}$ (our feature space), the error tolerance ϱ in approximating the extension of the diffusion coordinates, and the cutoff bound η for the condition number of the extension kernel. Clearly we have to select each of these parameters wisely.

The scale $\varepsilon > 0$ for the diffusion kernel must be chosen so that if we form a graph with Gaussian weights $w_\varepsilon(\boldsymbol{x},\boldsymbol{y})$ (defined in Section 2.1) on the edges between all pairs of points $\boldsymbol{x}$ and $\boldsymbol{y}$ then the graph is numerically connected. Connectedness of the graph is important because it guarantees the existence and uniqueness of the stationary distribution ϕ_0 of the Markov process on the graph via the graph-Laplacian normalization (see Appendix I in [12]), and the construction of the diffusion maps depends on the existence and uniqueness of ϕ_0. Therefore, ε must be large enough to ensure that every point in the graph is connected to at least one other point. However, it is clear that when ε is too large any affinity or dissimilarity between the data points is obscured, since w_ε converges to 1 as ε increases to infinity. In our numerical experiments, we select ε to be the mean of the Euclidean distances from each point to its k-nearest neighbor, where k equals to 5 percent of the total number of points in the training set. In other words, choose ε so that approximately 5 percent of all distances between pairs of points are less than or equal to ε. This means approximately 5 percent of all possible edges in the graph have weights greater than or equal to e^{-1}, the rest have smaller weights, i.e., the graph is sparse but not too sparse. The spectrum of the diffusion kernel decays relatively fast with this choice of ε. For example, in Fig. 1 we plot the largest 100 eigenvalues of the diffusion kernel taken from one trial of the lip-reading experiment.

The value of ε determined by our method was $\varepsilon = 740$. We see that the eigenvalues decrease quickly. Fast decay of the spectrum implies that any random walk initiated on the graph converges quickly to steady state and that the diffusion distance can be approximated more accurately with a smaller number of eigenfunctions. In other words, we will be able to detect clustering behaviors in the data with a small number of time steps t. In the numerical experiments below, we use $t = 1$. We do not need to set the Markov process forward in time. In these cases, having found the value ε appropriate for the data is enough for identifying grouping patterns in the data.

The dimension $s(\tau, t)$ of the embedding diffusion space can be determined by taking advantage of the geometrically grounded properties of the *ekmeans* algorithm. As described in Section 4.1, when we cluster each dataset using *ekmeans*, the intrinsic dimension of the dataset is automatically determined. Suppose our training set consists of a total of N datasets, $X^1, \cdots, X^N$, belonging to C different classes, *ekmeans* will find an intrinsic dimension K_i for each set X^i. This number K_j is also the intrinsic number of clusters in the set. As discussed in Section 4.1, this intrinsic number of K_j clusters does not change when the set X^j is embedded into a diffusion space of dimension higher than K_j. Therefore, it is natural to set the dimension of the embedding diffusion space for all of our data to be the maximum of K_j over all $j = 1, \cdots, N$, that is,

$$s(\tau, t) \stackrel{\Delta}{=} \max_{1 \le j \le N} k_j.$$

The choice of error tolerance ϱ is up to specific applications and personal judgment. However, we should keep in mind that small error limit means small extension range. Suppose we know a priori that our training set is a good representative of a manifold or data space (that is, there are no missing gap so that we can completely capture the shape of the manifold from the training data) and the unlabeled data lie on the manifold, then the approximation of the extension is fairly accurate, thus we can set ϱ to be small. A heuristic value to set for ϱ is one percent of the size of the test data. This gives on average a bound of 0.0001 for the error at each point where the extension is being computed. In our numerical experiments, we use this heuristic approach to set ϱ.

To determine a cutoff lower bound η for the condition number of the Gaussian

extension kernel $\mathrm{e}^{-\|\boldsymbol{x}-\boldsymbol{y}\|^2/\sigma^2}$ in (10), we have to keep in mind the approximation error tolerance ϱ. If ϱ is small, then η has to be large. In addition, as σ increases, the condition number of the kernel also increases. To predict how large η might get, we can take advantage of the symmetric kernel $\widetilde{k}(\boldsymbol{x},\boldsymbol{y})$, which we already computed from the Gaussian weights w_ε with ε optimally chosen for the data. Let κ be the condition number of $\widetilde{k}(\boldsymbol{x},\boldsymbol{y})$. It is easy to show that when $\sigma = \varepsilon$, the condition number of $\mathrm{e}^{-\|\boldsymbol{x}-\boldsymbol{y}\|^2/\sigma^2}$ is proportional to κ. Furthermore, as σ grows the condition number of the Gaussian kernel will only get worse. Thus, we can consider setting η larger than κ and inversely proportional to ϱ. In our numerical experiments, we set $\eta = \kappa/\varrho$, if κ is finite, and $\eta = 10^5/\varrho$ if κ is infinite.

4.3. Datasets Matching algorithm

We summarize our proposed method in the following algorithm:

Algorithm 4.1. *[Datasets Matching by Diffusion Maps and EMD]*

0. *Let X and Y denote the training data and the unlabeled data, respectively. Also, $X = \cup_i X^i$, where X^i is a set containing all signals characterizing one object, e.g., all image frames in one video sequence. Similarly, $Y = \cup_j Y^j$. There are C classes, and each X^i is known to belong of one of the C classes.*

1. *Signature construction in diffusion space:*

   i. *Construct the diffusion map Ψ_t* (9) *on the training data X, then embed X into a diffusion space $\mathbb{R}^{s(\tau,t)}$ (our feature space). The dimension $s(\tau,t)$ of the embedding diffusion space is determined by using* ekmeans *as described in Section 4.2.*

   ii. *Extend Ψ_t to the unlabeled data Y (this embeds Y into $\mathbb{R}^{s(\tau,t)}$).*

   iii. *For each i^{th} set of feature vectors in the training data, construct a signature $P^i = \{(\boldsymbol{p}_1^i, w_{\boldsymbol{p}_1}^i), \cdots, (\boldsymbol{p}_m^i, w_{\boldsymbol{p}_m}^i)\}$. Likewise, construct a signature $Q^j = \{(\boldsymbol{q}_1^j, w_{\boldsymbol{q}_1}^j), \cdots, (\boldsymbol{q}_n^j, w_{\boldsymbol{q}_n}^j)\}$ for each j^{th} unlabeled set.*

2. *Classification via EMD:*

i. Compute EMD between P^i and Q^j for all possible pairs (i, j). Define the cost of moving one unit mass from $\boldsymbol{p}_k^i$ to $\boldsymbol{q}_\ell^j$ to be

$$c(\boldsymbol{p}_k^i, \boldsymbol{q}_\ell^j) \triangleq \frac{1}{2}\|\boldsymbol{p}_k^i - \boldsymbol{q}_\ell^j\|^2,$$

where $||\cdot||$ is the Euclidean distance in the space $\mathbb{R}^{s(\tau,t)}$. Let $D_{ij} \triangleq EMD(P^i, Q^j)$.

ii. For each j, suppose $i_j \triangleq \arg\min_i D_{ij}$. Label Y^j with the label of X^{i_j}. That is, assign label by the nearest neighbor using EMD distance.

In Step 2.i we define the cost of moving one unit of mass from centroid $\boldsymbol{p}_k^i$ to centroid $\boldsymbol{q}_\ell^j$ to be proportional to the squared (instead of to the first power) of the Euclidean distance between the two centroids so as to give more preference to very close clusters.

5. Numerical Experiments and Results

We now show that our method can be applied to classification problems where the data characterizing each object consist of a set of signals instead of a single signal. We will show two examples of application. The first example is classification of underwater objects by analyzing Synthetic Aperture Sonar (SAS) waveforms reflected from the objects. The second example is a lip-reading application in which we identify the spoken word from a sequence of image frames extract from a silent video segment. We will also present a comparison in performance between our method and the LKC method in [12] (LKC stands for the first letter of the authors' last name: Lafon, Keller, and Coifman). In [12], the authors applied diffusion maps and GHME scheme to do dimensionality reduction and out-of-sample extension, much in the same manner as the first part in our approach. The main difference between our method and the LKC method is the use of EMD versus Hausdorff Distance (HD) to measure the distance between two sets. We recall that the HD between any two sets S_1 and S_2 is defined as

$$d_H(S_1, S_2) \triangleq \max\left(\max_{\boldsymbol{y}\in S_2}\min_{\boldsymbol{x}\in S_1}\|\boldsymbol{x}-\boldsymbol{y}\|, \max_{\boldsymbol{x}\in S_1}\min_{\boldsymbol{y}\in S_2}\|\boldsymbol{x}-\boldsymbol{y}\|\right),$$

where $\|\cdot\|$ denotes the Euclidean distance. In our proposed method, the classifier is nearest-neighbor in EMD. In LKC approach, the classifier is nearest-neighbor in HD.

5.1. Classification of underwater objects

The data in this example are collected from three different controlled experiments in a fresh water test pond at NSWC-PC. For details of the experiments, see [21]. In each of the three experiments, two objects were placed – either buried in the sand or proud – at the bottom of the pond. One of the objects was a sphere made of an iron casing filled with a different material each time. The other object was a solid aluminum cylinder of different length in each experiment. A sinusoidal pulse was transmitted across the floor of the pond and the reflected signal was recorded over a period of time at uniform time intervals. The data obtained contain waveforms reflected from the entire area of the pond floor. Waveforms corresponding to objects are extracted and processed using an improved version of the algorithm presented in [22]. This yields one set of rectangular blocks of waveforms per object.

Our goal is to identify objects according to their material compositions instead of shape. Let us name the sphere and the cylinder in Experiment 1 as $S1$ and $C1$; in Experiment 2 as $S2$ and $C2$; and in Experiment 3 as $S3$ and $C3$, respectively. Sphere $S1$ was filled with air, so we categorize it as one class with label **FeA** for iron-air. Spheres $S2$ and $S3$ were filled with silicone oil so we group them into another class with label **FeS** for iron-silicone. All three cylinders were of the same diameter and of the same material, so we grouped them into one class with label **Al** for aluminum. However, we would like to mention that $C1$ and $C2$ were of the same length while $C3$ was much shorter.

The waveform data is of extremely high dimension. Each rectangular block of waveforms is a 2D array of size 17 (cross range samples) by 600 (time samples). We treat it as a point in $\mathbb{R}^{17\times 600}$. Then we apply the steps in Algorithm 4.1 to identify the test object.

In our numerical experiment, we set aside a set of waveforms (corresponding to one object) to use as test data and train our algorithm on the remaining five sets. Then we cycle through all six objects, i.e., we repeat the classification process six times. Our classification results for all six runs are shown in Table 1. Using EMD, we were able to correctly identify spheres $S2$ and $S3$ as objects of class **FeS** and all three cylinders as objects of class **Al**. The mistake was made when $S1$ was labeled as **Al**. This error is

Table 1: Identification of Pond Objects

Object	Label by EMD	Label by HD
$C1$	**Al**	**Al**
$C2$	**Al**	**Al**
$C3$	**Al**	**FeA**
$S1$	**Al**	**Al**
$S2$	**FeS**	**Al**
$S3$	**FeS**	**FeS**

expected since the class **FeA** contains only one member $S1$. There is no training data for this class. On the other hand, the LKC method mislabeled the cylinder $C3$ as **FeA** object and the sphere $S2$ as **Al** object.

Let us examine the distribution of the feature vectors in the diffusion space. Fig. 2 shows the projection of the data onto top three diffusion coordinates. The diffusion coordinates in this figure were computed using all but the $S1$ data, i.e., all the cylinder data are displayed in blue, and the $S2$ and $S3$ data displayed in green were used to compute the diffusion coordinates. The $S1$ data (points displayed in red) were treated as test (left-out) data, which were embedded into the diffusion space by the GHME scheme described in Section 2.2.

From Fig. 2, we see that points corresponding to $S1$ are, on average, closer to the cylinder points. We recall that EMD is based on the distribution of points in the entire set. Therefore our method classified $S1$ as an **Al** object. The LKC method also classified $S1$ as an **Al** object. However, we can observe from its definition that HD is highly sensitive to outliers in general. To see this, let us examine the distribution of the feature vectors in the diffusion space during the experimental trial when $C3$ was left out as the test data. Fig. 3(a) shows the projection of the data onto top three diffusion coordinates for this case. Points corresponding to $C3$ (test data) are displayed in red. Blue points correspond to class **Al** objects ($C1$ and $C2$); green points correspond to class **FeS** objects ($S2$ and $S3$); and yellow points correspond to $S1$, i.e., class **FeA**. We see that the red points are on average close to the blue ($C1$ and $C2$) points. Thus,

using EMD we were able to correctly label object $C3$ as **Al**. However, LKC method mislabeled $C3$ as an **FeA** object. It is not difficult to see why this happened if we examine Fig. 3(b), a zoomed up version of Fig. 3(a). The HD between $C3$ and $S1$ is roughly the length of the arrow from the boundary of the cluster of yellow points to the other end of the red cluster, which is approximately 0.7. The HD between $C3$ and $C1$ (or similarly $C2$) is roughly the length of the arrow from the boundary of the blue cluster to the other end of the red cluster, which is approximately 1.0.

From this experiment, we conclude that using EMD makes our method more robust to noise and outliers than the LKC method which uses HD for sets discrimination.

5.2. Lip reading experiment

The objective of lip reading is to train a machine to automatically recognize the spoken words from the movements of the lips captured on silent video segments (no sound is involved). Much research effort has been devoted to this area. Many published algorithms involved sophisticated feature selection. In this example, our features are simply the diffusion coordinates, and the lips data used are collected from one speaker. More sophisticated feature selection might be necessary when more speakers, i.e., more variations in the lips, are involve.

Our main objective in this example is to illustrate the potential of our method for application to such problems. Moreover, a similar numerical experiment was done in [12]. By repeating the example, we hope to provide a good grounded comparison of our method to the LKC method in [12].

We recorded a subject speaking the first five digits ('one',...,'five') ten times using a Nikon Coolpix digital camera sampling at a rate of 60 frames per second. We then extracted the image frames from each movie clip and did some simple processing. First, we convert the images from color to gray scales ranging from 0 to 255. Then we cropped each image to a 55×70 pixels window around the lips to compensate for translations. (The speaker's nose was marked with a color marker to facilitate automatic cropping of the image frames). Each cropped frame is treated as a point in $\mathbb{R}^{55\times 70}$.

For each spoken digit, we randomly selected five image sequences from the ten in

Table 2: Lip-Reading Confusion Matrices. Dimension reduction by diffusion map.

	Recognition by EMD					Recognition by HD				
	Labeled as (% of times)									
Word	'1'	'2'	'3'	'4'	'5'	'1'	'2'	'3'	'4'	'5'
'one'	56.2	2.6	39.4	1.8	0	43.4	4.4	21	26.8	4.4
'two'	0.4	69.2	4	26.4	0	0	64	8.4	27.6	0
'three'	13	10.2	70	6.8	0	8.6	7.6	77.4	6.4	0
'four'	0	36.4	2.4	61.2	0	2	25.6	6.8	65.6	0
'five'	0	0	0	0	100	0	0	0	0	100

our collection to use as training data. This gives us a total of 25 sequences for training data and 25 for test data. We apply Algorithm 4.1 to identify the test sequences. Then we repeat the whole process 100 times. The recognition results over all 100 experimental trials are combined to compute the statistics shown in Table 2. The recognition errors range from 0% to 44.8% by using EMD and 0% to 56.6% by using HD, averaging 28.6% and 29.9% recognition error, respective. The largest error made by both methods was when identifying the word 'one'. This can be explained by large variations in the image sequences of the word 'one'. We display in Fig. 4 some images extracted from two sequences of the word 'one'. In the first sequence (top row) the speaker simply spoke the word with no emotion. In the second sequence (bottom row) the speaker was smiling while speaking.

Over all, recognition results by using EMD and HD are relatively comparable. However, HD did much worse than EMD when trying to identify the word 'one'. We recall that HD is highly sensitive to outliers, and the datasets corresponding to the word 'one' have many outliers.

6. Diffusion Maps versus Principal Component Analysis

In this section, we compare the performance in dimension reduction of diffusion maps to that of Principal Component Analysis (PCA). We repeat the two experiments

Table 3: Lip-Reading Confusion Matrices. Dimension reduction by PCA.

	Recognition by EMD					Recognition by HD				
	Labeled as (% of times)									
Word	'1'	'2'	'3'	'4'	'5'	'1'	'2'	'3'	'4'	'5'
'one'	89.8	0	10.2	0	0	81.6	1.4	12	4.2	0.8
'two'	0	98.2	0	1.8	0	0	92.2	0.6	7.2	0
'three'	0.4	3.2	95.2	1.2	0	1.2	6	91.2	1.6	0
'four'	0	7.6	0.4	92	0	1	9.4	2.6	87	0
'five'	0	0	0	0	100	0	0	0	0	100

described in the previous sections, this time using PCA instead of diffusion maps for dimension reduction.

First, we computed the eigenvectors (PCA vectors) of the covariance matrix of the training data. (See [23, Sec.2.1] for fast computation of PCA vectors). Then, we retain the top ten PCA vectors associated with the ten largest eigenvalues and use these as a basis for the reduced-dimension (PCA) space. The choice of ten is made based on the observation that the largest ten eigenvalues of the covariance matrix decreased the fastest. The rest of the eigenvalues are relatively small and decreased slowly. Using the ten basis vectors, we project both training and test data onto the corresponding 10-dimensional PCA space. Finally we apply the remaining steps in Algorithm 4.1 to construct signatures and then label the test data.

In the lip-reading application, using PCA for dimension reduction dramatically improved recognition results. In Table 3 we list results of classification using both EMD and HD. Comparing the recognition rates in Table 3 to those in Table 2, we see that PCA outperformed diffusion maps by at least 14%.

In the case of recognizing underwater objects, we found that PCA performed very poorly. Regardless of what waveform sets were used for training data (i.e., in all six repetitions of the experiment) PCA method projects the test data into a small region around the origin in the PCA space. For example, Fig. 5 shows the projection of all data onto top three PCA coordinates. In this figure, the PCA vectors were computed

using all but the $S2$ data ($S2$ data were left-out as test data). All cylinder data are displayed in blue, $S1$ in yellow, $S3$ in green, and $S2$ in red. We can see that the test ($S2$) data are clustered around the origin. This is the observed phenomenon throughout all six repetitions of the experiment. As a consequence of this degeneration, all three spheres were mislabeled.

Success and failure of PCA can be explained by the relationship between the data variates (the variables describing the data). In images, each pixel represents one variable. Since neighboring pixels are highly correlated, it is clear that there are very high correlations between the data variates. Since PCA provides a decorrelated coordinate system, projecting images of the lips onto a PCA space allows prominent features in the images to be more visible. In the case of the sonar data, the variates do not correlate in a linear manner. Consequently, PCA fails to provide a good representation for this type of data.

7. Conclusions

We have proposed an approach for datasets matching using diffusion maps for dimension reduction and feature extraction and EMD for sets discrimination. To illustrate the applicability of our proposed method, we have provided two examples of applications: silent lip-reading and classification of underwater objects. Both examples are real and tangible problems in the scientific community. To validate the applicability of our method, we have provided for comparison purpose numerical results of our method and the LKC method published in [12]. In the lip-reading application, both methods performed relatively similar. However, in the recognition of underwater objects application, our method correctly identified all objects and proved to be more stable.

Furthermore, we have also provided a comparison in the performance of diffusion maps to the performance of PCA in dimension reduction. We found that on the image data of the lips, where the data variates (the variables describing the data) are highly correlated, PCA performs better than diffusion maps. Perhaps this is because the decorrelated representation provided by PCA allows distinguishing features in the images to be more prominent. However, when the variables describing the data are not linearly

correlated (e.g., the case of waveforms reflected from underwater objects) then PCA fails to provide an adequate low-dimensional representation for the data. We are currently investigating why diffusion maps do not work as well as PCA when the input data are highly correlated as in the case of the lip-reading image frames.

Acknowledgments

This work was partially supported by the ONR grants N00014-06-1-0615, N00014-07-1-0166, the NSF grant DMS-0410406, and the NSF VIGRE grants DMS-0135345, DMS-0636297. A preliminary version of a subset of the material in this paper was presented at the SPIE Wavelets XII Conference, in San Diego, in August 2007 [24]. We have used the SPECTRAL Toolbox version 0.1 distributed on the web by Guido Sanguinetti and Jonathan Laidler to compute Elongated Kmeans. Finally, we would like thank Bradley Marchand for his help in processing the sonar data used in the numerical experiments.

References

[1] R. R. Coifman and S. Lafon, "Diffusion maps", *Appl. Comput. Harmon. Anal.*, **21**, pp. 5–30, July 2006.

[2] S. Lafon, "Diffusion Maps and Geometric Harmonics", Ph.D. Dissertation, Yale University, May 2004.

[3] Y. Rubner and C. Tomasi, *Perceptual Metrics for Image Database Navigation*, Kluwer Academic Publishers, Boston, 1999.

[4] Y. Rubner, C. Tomasi, L. J. Guibas, "The earth mover's distance as a metric for image retrieval", *Int. J. Comput. Vision*, **40**(2), pp. 99–121, 2000.

[5] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering", *Advances in Neural Information Processing Systems*, pp. 585–591, 2001.

[6] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation", *Neural Computation*, **15**(6), pp. 1373–1396, 2003.

[7] D. Donoho and C. Grimes, "Hessian eigenmaps: New locally linear embedding techniques for high-dimensional data", *Proc. Natl. Acad. Sci. USA*, bf 100(10), pp. 5591–5596, 2003.

[8] S. Roweis and L. Saul, "Nonlinear dimensionality reduction by locally linear embedding", *Science*, **290**, pp. 2323–2326, 2000.

[9] Z. Zhang and H. Zha, "Principal manifolds and nonlinear dimension reduction via local tangent space alignment", *SIAM J. Sci. Comput.*, **26**(1), pp. 313-338, 2005.

[10] M. Belkin and P. Niyogi, "Towards a theoretical foundation for Laplacian-based manifold methods", in *Learning Theory: 18th Annual Conference on Learning Theory, COLT 2005* (P. Auer and R. Meir, eds.), pp. 486–500, 2005.

[11] S. Lafon and A. B. Lee, "Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning and data set parameterization", *IEEE Trans. Pattern Anal. Machine Intell.*, **28**(9), pp. 1393–1403, 2006.

[12] S. Lafon, Y. Keller, R.R. Coifman, "Data fusion and multicue data matching by diffusion maps", *IEEE Trans. Pattern Anal. Machine Intell.*, **28**(11), pp. 1784–1797, 2006.

[13] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review", *ACM Comput. Surv.*, **31**(3), pp. 264–323, 1999.

[14] R. Kannan, S. Vempala, A. Vetta, "On Clusterings: Good, Bad and Spectral", *Proc. 41st Annual Symposium on Foundations of Computer Science*, pp. 367–377, 2000.

[15] R. R. Coifman and S. Lafon, "Geometric harmonics", *Appl. Comput. Harmon. Anal.*, **21**, pp. 31–52, July 2006.

[16] C. Fowlkes, S. Belongie, F. Chung, and J. Malik, "Spectral grouping using the Nyström method", *IEEE Trans. Pattern Anal. Machine Intell.*, **26**(2), pp. 214–225, 2004.

[17] Y. Bengio, J.-F. Paiement, P. Vincent, O. Delalleau, N. Le Roux, and M. Ouimet, "Out-of-sample extensions for LLE, Isomap, MDS, Eigenmaps, and spectral clustering", in *Advances in Neural Information Processing Systems*, **16**, pp. 177–184, MIT Press, 2004.

[18] L. C. Evans, "Partial Differential Equations and Monge-Kantorovich Mass Transfer" (lecture notes), `www.math.berkeley.edu/~evans/Monge-Kantorovich.survey.pdf`.

[19] G. Sanguinetti, J. Laidler, N. D. Lawrence, "Automatic Determination of the Number of Clusters Using Spectral Algorithms", *Proc. 2005 IEEE Workshop on Machine Learning for Signal Processing*, pp. 55–60, 2008.

[20] A. Y. Ng, M. I. Jordan, Y. Weiss, "On spectral clustering: Analysis and an algorithm", *Advances in Neural Information Processing Systems 14*, 2001.

[21] C. L. Nesbitt and J. L. Lopes, "Subcritical detection of an elongated target buried under a rippled interface", In *Oceans '04, MTS/IEEE Techno-Ocean '04*, **4**, pp. 1945–1952, 2004.

[22] B. Marchand, N. Saito, and H. Xiao, "Classification of objects in synthetic aperture sonar images", *Proc. 14th IEEE Statistical Signal Processing Workshop*, pp. 433–437, 2007.

[23] N. Saito, "Image approximation and modeling via least statistically dependent bases", *Pattern Recogn.*, **34**, pp. 1765–1784, 2001.

[24] L. Lieu and N. Saito, "Automated discrimination of shapes in high dimensions", in *Wavelets XII* (D. Van De Ville, V. K. Goyal, and M. Papadakis, eds.), *Proc. SPIE* **6701**, Paper # 67011V, 2007.

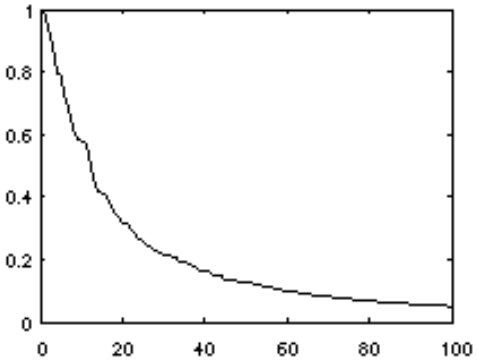


Figure 1: Largest 100 eigenvalues of the diffusion kernel in one trial of the lip-reading experiment. ($\varepsilon = 740$).

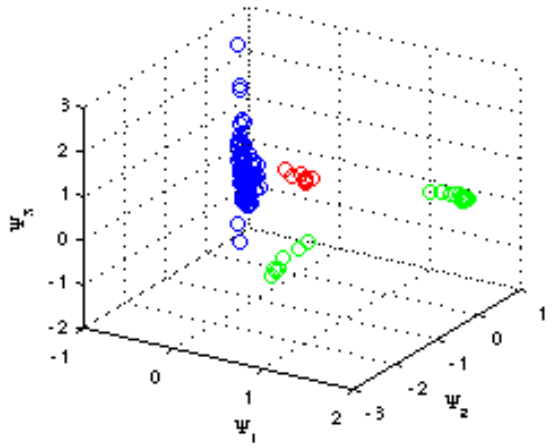
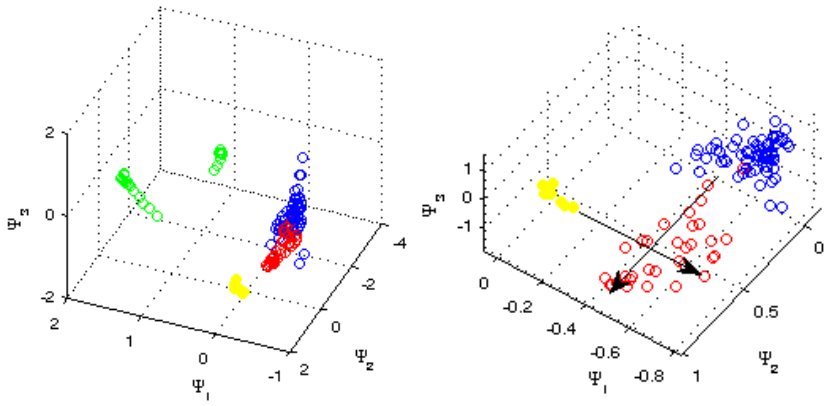


Figure 2: Top three diffusion coordinates. Training data are in blue (all three cylinders) and green ($S2$ and $S3$). Points displayed in red are the test data $S1$.



(a) Embedded points (b) Zoom up

Figure 3: (a) Top three diffusion coordinates. Training data are in blue ($C1$ and $C2$), yellow ($S1$), and green ($S2$ and $S3$). Points displayed in red are the test data $C3$. (b) The zoomed up version of (a).

Figure 4: Lip shapes of two different sequences of the word 'one'.

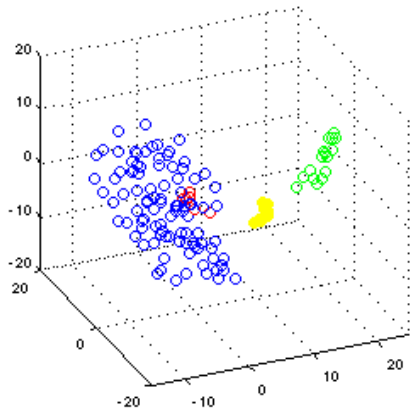


Figure 5: Top three PCA coordinates. Training data are in blue (cylinder data), yellow ($S1$), and green ($S3$). Points displayed in red are the test data $S2$.