

Classification Constraint Dimensionality Reduction

Raviv Raich, *Member, IEEE*, S. B. Damelin *Senior Member, IEEE*,
Jose A. Costa, *Member, IEEE* and Alfred O. Hero III, *Fellow, IEEE*

Abstract

In this paper, we evaluate the contribution of the classification constrained dimensionality reduction (CCDR) algorithm to the performance of several classifiers. We present extensions to previously introduced CCDR algorithms to multiple hypotheses. We investigate classification performance using the CCDR algorithm on hyper-spectral satellite imagery data. We demonstrate the performance gain for both local and global classifiers and demonstrate a 10% improvement of the k -nearest neighbors algorithm performance. We present a connection between intrinsic dimension estimation and the optimal embedding dimension obtained using the CCDR algorithm.

Index Terms

Classification, Computational Complexity, Constraint, Dimensionality reduction, Embedding, High Dimensional Data, Kernel, K-Nearest Neighbor, Linear, Manifold, Non-Linear, Probability, Out of Sample Approximation.

This work was supported, in part by NSF-DMS-0555839 and NSF-DMS-0439734

Classification Constraint Dimensionality Reduction

I. INTRODUCTION

In classification theory, the main goal is to find a mapping from an observation space \mathcal{X} consisting of a collection of points in some containing Euclidean space $\mathbb{R}^{d'}$, $d' \geq 1$ into a set consisting of several different integer valued hypotheses. In some problems, the observations from the set \mathcal{X} lie on a $d \leq d'$ manifold \mathcal{M} and Whitney's theorem tells us that provided that this manifold is smooth enough, there exists an embedding of \mathcal{M} into \mathbb{R}^{2d+1} . This motivates the approach taken by kernel methods in classification theory, such as support vector machines [?] for example. Our interest is in finding an embedding of \mathcal{M} into a lower dimensional Euclidean space.

Fig. 1. PCA of a two-classes classification problem.

Dimensionality reduction of high dimensional data, was addressed in classical methods such as principal component analysis (PCA) [?] and multidimensional scaling (MDS) [?], [?]. In PCA, an eigendecomposition of the $d \times d$ empirical covariance matrix is performed and the data points are linearly projected along the $0 < m \leq d$ eigenvectors with the largest eigenvalues. A problem that may occur with PCA for classification is demonstrated in Fig. ???. When the information that is relevant for classification is present only in the eigenvectors associated with the small eigenvalues (e_2 in the figure), removal of such eigenvectors may result in severe degradation in classification performance. In MDS, the goal is to find a lower dimensional embedding of the original data points that preserves the relative distances between all the data points. The two later methods suffer greatly when the manifold is nonlinear. For example, PCA will not be able to offer dimensionality reduction for classification of two classes lying each on one of two concentric circles.

In the seminal paper of Tenenbaum *et al* [?], Isomap, a global dimensionality reduction algorithm was introduced taking into account the fact that data points may lie on a lower

dimensional manifold. Unlike MDS, geodesic distances (distances that are measured along the manifold) are preserved by Isomap. Isomap utilizes the classical MDS algorithm, but instead of using the matrix of Euclidean distances, it uses a modified version of it. Each point is connected only to points in its local neighborhood. A distance between a point and another point outside its local neighborhood is replaced with the sum of distances along the shortest path in graph. This procedure modifies the squared distances matrix replacing Euclidian with geodesic distances.

In [?], Belkin and Niyogi present a related Laplacian eigenmap dimensionality reduction algorithm. The algorithm performs a minimization on the weighted sum of squared-distances of the lower-dimensional data. Each weight multiplying the squared-distances of two low-dimensional data points is inversely related to distance between the corresponding two high-dimensional data points. Therefore, small distance between two high-dimensional data points results in small distance between two low-dimensional data points. To preserve the geodesic distances, the weight of the distance between two points that do not share a local neighborhood is set to zero.

We refer the interested reader to the references below and those cited therein for a list of some of the most commonly used additional algorithms within the class of *manifold learning* algorithms and their different advantages relevant to our work. Locally Linear Embedding (LLE) [?], Laplacian Eigenmaps [?], Hessian Eigenmaps (HLLE) [?], Local Space Tangent Analysis [?], Diffusion Maps [?] and Semidefinite Embedding (SDE) [?].

The algorithms mentioned above, consider the problem of learning a lower-dimensional embedding of the data. In classification, such algorithms can be used to preprocess high-dimensional data before performing the classification. This could potentially allow for a lower computational complexity of the classifier. In some cases, dimensionality reduction results increase the computational complexity of the classifier. In fact, support vector machines suggest the opposing strategy: data points are projected onto a higher-dimensional space and classified by a low computational complexity classifier. To guarantee a low computational complexity of the classifier of the low-dimensional data, a classification constrained dimensionality reduction (CCDR) algorithm was introduced in [?]. The CCDR algorithm is an extension of Laplacian eigenmaps [?] and it incorporates class label information into the cost function, reducing the distance between points with similar label.

In [?] the CCDR algorithm was only studied for two classes and its performance was illustrated

for simulated data. In this paper, we introduce an extension of the algorithm to the multi-class problem and present experimental results for the Landsat MSS imagery data [?]. We study the algorithm performance as its various parameters, (e.g., dimension, label importance, and local neighborhood), are varied. We study the performance of CCDR as preprocessing prior to implementation of several classification algorithms such as k -nearest neighbors, linear classification, and neural networks. We demonstrate a 10% improvement over the k -nearest neighbors algorithm performance benchmark for this dataset. We address the issue of dimension estimation and its effect on classification performance.

The organization of this paper is as follows. Section ?? presents the multiple-class CCDR algorithm. Section ?? provides a study of the algorithm using the Landsat dataset and Section ?? summarizes our results.

II. DIMENSIONALITY REDUCTION

Let $\mathcal{X}_n = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ be a set of n points constrained to lie on an m -dimensional submanifold $\mathcal{M} \subseteq \mathbb{R}^d$. In dimensionality reduction, our goal is to obtain a lower-dimensional embedding $\mathcal{Y}_n = \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n\}$ (where $\mathbf{y}_i \in \mathbb{R}^m$ with $m < d$) that preserves local geometry information such that processing of the lower dimensional embedding \mathcal{Y}_n yields comparable performance to processing of the original data points \mathcal{X}_n . Alternatively, we would like learn the mapping $f : \mathcal{M} \subseteq \mathbb{R}^d \rightarrow \mathbb{R}^m$ that maps every data point \mathbf{x}_i to $\mathbf{y}_i = f(\mathbf{x}_i)$ such that some geometric properties of the high-dimensional data are preserved in the lower dimensional embedding. The first question that comes to mind is how to select f , or more specifically how to restrict the function f so that we can still achieve our goal.

A. Linear dimensionality reduction

1) *PCA*: When principal component analysis (PCA) is used for dimensionality reduction, one considers a linear embedding of the form

$$\mathbf{y}_i = f(\mathbf{x}_i) = \mathbf{A}\mathbf{x}_i,$$

where \mathbf{A} is $m \times d$. This embedding captures the notion of proximity in the sense that close points in the high dimensional space map to close points in the lower dimensional embedding,

i.e., $\|\mathbf{y}_i - \mathbf{y}_j\| = \|\mathbf{A}(\mathbf{x}_i - \mathbf{x}_j)\| \leq \|\mathbf{A}\| \|\mathbf{x}_i - \mathbf{x}_j\|$. Let

$$\bar{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i$$

and

$$\mathcal{C}_x = \frac{1}{n} \sum_{i=1}^n (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T.$$

Similarly, let

$$\bar{\mathbf{y}} = \frac{1}{n} \sum_{i=1}^n \mathbf{y}_i$$

and

$$\mathcal{C}_y = \frac{1}{n} \sum_{i=1}^n (\mathbf{y}_i - \bar{\mathbf{y}})(\mathbf{y}_i - \bar{\mathbf{y}})^T.$$

Since $\mathbf{y}_i = \mathbf{A}\mathbf{x}_i$, we have $\bar{\mathbf{y}} = \mathbf{A}\bar{\mathbf{x}}$ and $\mathcal{C}_y = \mathbf{A}\mathcal{C}_x\mathbf{A}^T$. In PCA, the goal is to find the projection matrix \mathbf{A} that preserves most of the energy in the original data by solving

$$\max_{\mathbf{A}} \text{tr}\{\mathcal{C}_y(\mathbf{A})\} \quad \text{s.t.} \quad \mathbf{A}\mathbf{A}^T = \mathbf{I},$$

which is equivalent to

$$\max_{\mathbf{A}} \text{tr}\{\mathbf{A}\mathcal{C}_x\mathbf{A}^T\} \quad \text{s.t.} \quad \mathbf{A}\mathbf{A}^T = \mathbf{I}. \quad (1)$$

The solution to (??), is given by $\mathbf{A} = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_m]^T$, where \mathbf{u}_i is the eigenvector of \mathcal{C}_x corresponding to its i th largest eigenvalue. When the data lies on an m -dimensional hyperplane, the matrix \mathcal{C}_x has only m positive eigenvalues and the rest are zero. Furthermore, every \mathbf{x}_i belongs to $\bar{\mathbf{x}} + \text{span}\{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_m\} \subseteq \mathbb{R}^d$. In this case, the mapping PCA finds $f(\mathbf{x}) = \mathbf{A}\mathbf{x}$ is one-to-one and satisfies $\|f(\mathbf{x}_i) - f(\mathbf{x}_j)\| = \|\mathbf{A}(\mathbf{x}_i - \mathbf{x}_j)\| = \|\mathbf{x}_i - \mathbf{x}_j\|$. Therefore, the lower embedding preserves all the geometry information in the original dataset \mathcal{X} . We would like to point out that PCA can be written as

$$\max_{\{\mathcal{Y}\}} \sum_{i=1}^n \|\mathbf{y}_i - \mathbf{y}_j\|^2 \quad \text{s.t.} \quad \mathbf{y}_i = \mathbf{A}\mathbf{x}_i \text{ and } \mathbf{A}\mathbf{A}^T = \mathbf{I},$$

2) *MDS*: Multidimensional Scaling (MDS) differs from PCA in the way the input is provided to it. While in PCA, the original data \mathcal{X} is provided, the classical MDS requires only the set of all Euclidean pairwise distances $\{\|\mathbf{x}_i - \mathbf{x}_j\|_2\}_{i=1, j>i}^{n-1}$. As MDS uses only pairwise distances, the solution it finds is given up to translation and unitary transformation. Let $\mathbf{x}'_i = \mathbf{x}_i - \mathbf{c}$, the Euclidean distance $\|\mathbf{x}'_i - \mathbf{x}'_j\|$ is the same as $\|\mathbf{x}_i - \mathbf{x}_j\|$. Let \mathbf{U} be an arbitrary unitary matrix \mathbf{U} satisfying $\mathbf{U}^T \mathbf{U} = \mathbf{I}$ and define $\mathbf{x}'_i = \mathbf{U} \mathbf{x}_i$. The distance $\|\mathbf{x}'_i - \mathbf{x}'_j\|$ is equal to $\|\mathbf{U}(\mathbf{x}_i - \mathbf{x}_j)\|$, which by the invariance of the Euclidean norm to a unitary transformation equals to $\|\mathbf{x}_i - \mathbf{x}_j\|$. Denote the pairwise squared-distance matrix by $[\mathbf{D}_2]_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|^2$. By the definition of Euclidean distance, the matrix \mathbf{D}_2 satisfies

$$\mathbf{D}_2 = \mathbf{1}\phi^T + \phi\mathbf{1}^T - 2\mathbf{X}^T \mathbf{X}, \quad (2)$$

where $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n]$ and $\phi = [\|\mathbf{x}_1\|^2, \|\mathbf{x}_2\|^2, \dots, \|\mathbf{x}_n\|^2]^T$. To verify (??), one can examine the ij -th term of \mathbf{D}_2 and compare with $\|\mathbf{x}_i - \mathbf{x}_j\|^2$. Denote the $n \times n$ matrix $\mathbf{H} = \mathbf{I} - \mathbf{1}\mathbf{1}^T/n$. Multiplying both sides of \mathbf{D}_2 with \mathbf{H} in addition to a factor of $-\frac{1}{2}$, yields

$$-\frac{1}{2}\mathbf{H}\mathbf{D}_2\mathbf{H} = (\mathbf{X}\mathbf{H})^T(\mathbf{X}\mathbf{H}),$$

which is key to MDS, i.e., Cholesky decomposition of $-\frac{1}{2}\mathbf{H}\mathbf{D}_2\mathbf{H}$ yields \mathbf{X} to within a translation and a unitary transformation. Consider the eigendecomposition $-\frac{1}{2}\mathbf{H}\mathbf{D}_2\mathbf{H} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$. Therefore, a rank d \mathbf{X} can be obtained as $\mathbf{X} = \mathbf{\Lambda}_d^{\frac{1}{2}}\mathbf{U}_d^T$, where $\mathbf{\Lambda}_d = \text{diag}\{[\lambda_1, \lambda_2, \dots, \lambda_d]\}^{\frac{1}{2}}$ and $\mathbf{U}_d = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_d]$. Note that $\mathbf{X}\mathbf{H}$ is a translated version of \mathbf{X} , in which every column \mathbf{x}_i is translated to $\mathbf{x}_i - \bar{\mathbf{x}}$.

To use MDS for dimensionality reduction, we can consider a two step process. First a square-distance matrix \mathbf{D}_2 is obtained from the high-dimensional data \mathcal{X} . Then, MDS is applied to \mathbf{D}_2 to obtain a low-dimensional ($m < d$) embedding by $\mathbf{X}_m = \mathbf{\Lambda}_m^{\frac{1}{2}}\mathbf{U}_m^T = \mathbf{X}\mathbf{U}_m\mathbf{U}_m^T$. In the absence of noise, this procedure provides an affine transformation to the high-dimensional data and thus can be regarded as a linear method.

B. Nonlinear dimensionality reduction

Linear maps are limited as they cannot preserve the geometry of nonlinear manifolds.

1) *ISOMAP*: In [?], Tenenbaum *et al* find a nonlinear embedding that rather than preserving the Euclidean distance between points on a manifold, preserves the geodesic distance between points on the manifold. Similar to MDS where a lower dimensional embedding is found to preserve the Euclidean distances of high dimensional data, ISOMAP finds a lower dimensional embedding that preserves the geodesic distances between high-dimensional data points.

2) *Kernel PCA*:

3) *Laplacian Eigenmaps*: Belkin and Niyogi's Laplacian eigenmaps dimensionality reduction algorithm [?] takes a different approach. They consider a nonlinear mapping f that minimize the Laplacian

$$\arg \min_{\|f\|_{L^2(\mathcal{M})}=1} \int_{\mathcal{M}} \|\nabla f\|^2. \quad (3)$$

Since the manifold is not available but only data point on it are, the lower dimensional embedding is found by minimizing the graph Laplacian given by

$$\sum_{i=1}^n w_{ij} \|\mathbf{y}_i - \mathbf{y}_j\|^2, \quad (4)$$

where w_{ij} is the ij th element of the adjacency matrix which is constructed as follows: For $k \in \mathbb{N}$, a k -nearest neighbors graph is constructed with the points in \mathcal{X}_n as the graph vertices. Each point \mathbf{x}_i is connected to its k -nearest neighboring points. Note that it suffices that either \mathbf{x}_i is among \mathbf{x}_j 's k -nearest neighbors or \mathbf{x}_j is among \mathbf{x}_i 's k -nearest neighbors for \mathbf{x}_i and \mathbf{x}_j to be connected. For a fixed scale parameter $\epsilon > 0$, the weight associated with the two points \mathbf{x}_i and \mathbf{x}_j satisfies

$$w_{ij} = \begin{cases} \exp \{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\epsilon\} & \text{if } \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ are connected} \\ 0 & \text{otherwise.} \end{cases}$$

III. CLASSIFICATION CONSTRAINED DIMENSIONALITY REDUCTION

A. Statistical framework

To put the problem in a classification context, we consider the following model. Let $\mathcal{X}_n = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ be a set of n points sampled from an m -dimensional submanifold $\mathcal{M} \subseteq \mathbb{R}^d$. Each point $\mathbf{x}_i \in \mathcal{M}$ is associate with a class label $c_i \in \mathcal{A} = \{0, 1, 2, \dots, L\}$, where $c_i = 0$ corresponds to the case of unlabeled data. We assume that pairs $(\mathbf{x}_i, c_i) \in \mathcal{M} \times \mathcal{A}$ are i.i.d. drawn from a joint distribution

$$P(\mathbf{x}, c) = p_x(\mathbf{x}|c)P_c(c) = P_c(c|\mathbf{x})p_x(\mathbf{x}), \quad (5)$$

where $p_x(\mathbf{x}) > 0$ and $p_x(\mathbf{x}|c) > 0$ (for $\mathbf{x} \in \mathcal{M}$) are the marginal and the conditional probability density functions, respectively, satisfying $\int_{\mathcal{M}} p_x(\mathbf{x}) d\mathbf{x} = 1$, $\int_{\mathcal{M}} p_x(\mathbf{x}|c) d\mathbf{x} = 1$ and $P_c(c) > 0$ and $P_c(c|\mathbf{x}) > 0$ are the a priori and a posteriori probability mass functions of the class label, respectively, satisfying $\sum_c P_c(c) = 1$ and $\sum_c P_c(c|\mathbf{x}) = 1$. While we consider unlabeled points of the form $(\mathbf{x}_i, 0)$ similar labeled points, we still make the following distinction. Consider the following mechanism for generating an unlabeled point. First, a class label $c \in \{1, 2, \dots, L\}$ is generated from the labeled a priori probability mass function $P'_c(c) = P(c|c \text{ is labeled}) = P_c(c) / \sum_{c'=1}^L P_c(c')$. Then \mathbf{x}_i is generated according to $p_x(\mathbf{x}|c)$. To treat c as an unobserved label, we marginalize $P(\mathbf{x}, c|c \text{ is labeled}) = p_x(\mathbf{x}|c)P'_c(c)$ over c :

$$p_x(\mathbf{x}|c=0) = \sum_{q=1}^L p_x(\mathbf{x}|c=q)P'_c(q) = \frac{\sum_{q=1}^L p_x(\mathbf{x}|c=q)P_c(q)}{\sum_{c'=1}^L P_c(c')}. \quad (6)$$

This suggests that the conditional PDF of unlabeled points $f_x(\mathbf{x}|c=0)$ is uniquely determined by the class priors and the conditionals for labeled point. We would like to point out that this is one of few treatments that can be offered for unlabeled point. For example, in anomaly detection, one may want to associate the unlabeled point with contaminated data points, which can be represented as a density mixture of $p_x(\mathbf{x}|c=0)$ and $\gamma(\mathbf{x})$ (e.g., $\gamma(\mathbf{x})$ is uniform in \mathcal{X}).

In classification constraint dimensionality reduction, our goal is to obtain a lower-dimensional embedding $\mathcal{Y}_n = \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n\}$ (where $\mathbf{y}_i \in \mathbb{R}^m$ with $m < d$) that preserves local geometry and that encourages clustering of points of the same class label. Alternatively, we would like to find a mapping $\mathbf{f}(\mathbf{x}, c) : \mathcal{M} \times \mathcal{A} \rightarrow \mathbb{R}^m$ for which $\mathbf{y}_i = \mathbf{f}(\mathbf{x}_i, c_i)$ that is smooth and that clusters points of the same label.

We introduce the class label indicator for data point \mathbf{x}_i as $c_{ki} = I(c_i = k)$, for $k = 1, 2, \dots, L$ and $i = 1, 2, \dots, n$. Note that when point \mathbf{x}_i is unlabeled $c_{ki} = 0$ for all k . Using the class indicator, we can write the number of point in class k as $n_k = \sum_{i=1}^n c_{ki}$. If all points are labeled, then $n = \sum_{k=1}^L n_k$.

B. Linear dimensionality reduction for classification

1) *LDA*: Restricting the discussion to linear maps, one can extend PCA to take into account label information using the multi-class extension to Fisher's linear discriminant analysis (LDA). Instead of maximizing the data covariance matrix, LDA maximizes the ratio of the between-class-covariance to within-class-covariance. In other words, we obtain a linear transformation

$\mathbf{y}_i = f(\mathbf{x}_i, c_i) = \mathbf{A}\mathbf{x}_i$ with matrix \mathbf{A} that is the solution to the following maximization:

$$\max_{\mathbf{A}} \text{tr}\{\mathbf{A}\mathbf{C}_B\mathbf{A}^T\} \quad \text{s.t.} \quad \mathbf{A}\mathbf{C}_W\mathbf{A}^T = \mathbf{I}, \quad (7)$$

where

$$\mathbf{C}_B = \frac{1}{n} \sum_{k=1}^L n_k (\bar{\mathbf{x}}^{(k)} - \bar{\mathbf{x}})(\bar{\mathbf{x}}^{(k)} - \bar{\mathbf{x}})^T$$

is the between-class-covariance matrix, $\bar{\mathbf{x}}^{(k)} = \sum_i c_{ki} \mathbf{x}_i / n_k$ is the k th class center, $\bar{\mathbf{x}} = \sum_i \mathbf{x}_i / n$ is the center point of the dataset,

$$\mathbf{C}_W = \frac{1}{n} \sum_{k=1}^L n_k \mathbf{C}_W^{(k)},$$

is the within-class-covariance, and

$$\mathbf{C}_W^{(k)} = \frac{\sum_{i=1}^n c_{ki} (\mathbf{x}_i - \bar{\mathbf{x}}^{(k)})(\mathbf{x}_i - \bar{\mathbf{x}}^{(k)})^T}{n_k}$$

is within-class- k covariance matrix. In Fig. ??, LDA selects an embedding that projects the data onto \mathbf{e}_2 since the maximum distance between classes is achieved along with a minimum class variance when projecting the data onto \mathbf{e}_2 . We are interested in exploring a strategy that maximizes class separation in the lower dimensional embedding.

2) *Marginal Fisher Analysis*: Recent work [?], presents the marginal Fisher analysis (MFA), which is a method that minimizes the ratio between intraclass compactness and interclass separability. In its basic formulation MFA is a linear embedding, in which $\mathbf{y}_i = \mathbf{A}\mathbf{x}_i$. Another aspect of the method is that it considers two classes. The kernel trick is use to provide a nonlinear extension to MFA. To construct the cost function, two quantities are of interest: intraclass compactness and interclass separability. The intraclass compactness can be written as

$$\sum_{i,j} w_{ij} \|\mathbf{y}_i - \mathbf{y}_j\|^2, \quad (8)$$

where w_{ij} is given by

$$w_{ij} = \left(\sum_k c_{ki} c_{kj} \right) I(\mathbf{x}_i \in N_{k_1}^+(\mathbf{x}_j) \text{ or } \mathbf{x}_j \in N_{k_1}^+(\mathbf{x}_i)) \quad (9)$$

and $N_k^+(\mathbf{x})$ denote the k -nn neighborhood of \mathbf{x} within the same class as \mathbf{x} . Note that the term $\sum_k c_{ki} c_{kj}$ is one if \mathbf{x}_i and \mathbf{x}_j have the same label and zero otherwise. Similarly, the interclass

separability can be written as

$$\sum_{i,j} w_{ij} \|\mathbf{y}_i - \mathbf{y}_j\|^2, \quad (10)$$

where w_{ij} is given by

$$w_{ij} = \left(1 - \sum_k c_{ki} c_{kj}\right) I(\mathbf{x}_i \in N_{k_2}^-(\mathbf{x}_j) \text{ or } \mathbf{x}_j \in N_{k_2}^-(\mathbf{x}_i)) \quad (11)$$

and $N_k^-(\mathbf{x})$ denote the k -nn neighborhood of \mathbf{x} outside the class of \mathbf{x} .

IV. DIMENSIONALITY REDUCTION FOR CLASSIFICATION ON NONLINEAR MANIFOLDS

Here, we review the CCDD algorithm [?] and its extension to multi-class classification.

To cluster lower dimensional embedded points of the same label we associate each class with a class center namely $\mathbf{z}_k \in \mathbb{R}^m$. We construct the following cost function:

$$J(\mathcal{Z}_L, \mathcal{Y}_n) = \sum_{ki} c_{ki} \|\mathbf{z}_k - \mathbf{y}_i\|^2 + \frac{\beta}{2} \sum_{ij} w_{ij} \|\mathbf{y}_i - \mathbf{y}_j\|^2, \quad (12)$$

where $\mathcal{Z}_L = \{\mathbf{z}_1, \dots, \mathbf{z}_L\}$ and $\beta \geq 0$ is a regularization parameter. We consider two terms on the RHS of (??). The first term corresponds to the concentration of points of the same label around their respective class center. The second term is as in (??) or as in Laplacian Eigenmaps [?] and controls the smoothness of the embedding over the manifold. Large values of β produce an embedding that ignores class labels and small values of β produce an embedding that ignores the manifold structure. Training data points will tend to collapse into the class centers, allowing many classifiers to produce perfect classification on the training data without being able to control the generalization error (i.e., classification error of the unlabeled data). Our goal is to find \mathcal{Z}_L and \mathcal{Y}_n that minimize the cost function in (??).

Let \mathbf{C} be the $L \times n$ class membership matrix with c_{ki} as its ki -th element, $\mathbf{Z} = [\mathbf{z}_1, \dots, \mathbf{z}_L, \mathbf{y}_1, \dots, \mathbf{y}_n]$, and $\mathbf{0}$ be the $L \times L$ all zeroes matrix and

$$\mathbf{G} = \begin{bmatrix} \mathbf{0} & \mathbf{C} \\ \mathbf{C}^T & \beta \mathbf{W} \end{bmatrix}.$$

Minimization over \mathbf{Z} of the cost function in (??) can be expressed as

$$\begin{aligned} \min_{\mathbf{Z}} \quad & \text{tr}(\mathbf{Z} \mathbf{L} \mathbf{Z}^T), \\ \mathbf{Z} \mathbf{D} \mathbf{1} = & \mathbf{0} \\ \mathbf{Z} \mathbf{D} \mathbf{Z}^T = & \mathbf{I} \end{aligned} \quad (13)$$

where $\mathbf{D} = \text{diag}\{\mathbf{G}\mathbf{1}\}$ and $\mathbf{L} = \mathbf{D} - \mathbf{G}$. To prevent the lower-dimensional points and the class centers from collapsing into a single point at the origin, the regularization $\mathbf{Z}\mathbf{D}\mathbf{Z}^T = \mathbf{I}$ is introduced. The second constraint $\mathbf{Z}\mathbf{D}\mathbf{1} = \mathbf{0}$ is constructed to prevent a degenerate solution, e.g., $\mathbf{z}_1 = \dots = \mathbf{z}_L = \mathbf{y}_1 = \dots = \mathbf{y}_n$. This solution may occur since $\mathbf{1}$ is in the null-space of the Laplacian \mathbf{L} operator, i.e., $\mathbf{L}\mathbf{1} = \mathbf{0}$. The solution to (??) can be expressed in term of the following generalized eigendecomposition

$$\mathbf{L}^{(n)}\mathbf{u}_k^{(n)} = \lambda_k^{(n)}\mathbf{D}^{(n)}\mathbf{u}_k^{(n)}, \quad (14)$$

where $\lambda_k^{(n)}$ is the k th eigenvalue and $\mathbf{u}_k^{(n)}$ is its corresponding eigenvector. Note that we include $^{(n)}$ to emphasize the dependence on the n data points. Without loss of generality we assume $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_{n+L}$. Specifically, matrix \mathbf{Z} is given by $[\mathbf{u}_2, \mathbf{u}_3, \dots, \mathbf{u}_{m+1}]^T$, where the first L columns correspond to the coordinates of the class centers, i.e., $\mathbf{z}_k = \mathbf{Z}\mathbf{e}_k$, and the following n columns determine the embedding of the n data points, i.e., $\mathbf{y}_t = \mathbf{Z}\mathbf{e}_{L+t}$. We use \mathbf{e}_i to denote the canonical vector such that $[\mathbf{e}_i]_s = 1$ for element $s = i$ and zero otherwise.

A. Classification and computational complexity

In classification, the goal is to find a classifier $a_x(\mathbf{x}) : \mathcal{M} \rightarrow \mathcal{A}$ based on the training data that minimizes the generalization error:

$$\hat{a} = \arg \min_{a \in \mathcal{F}} E[I(a(\mathbf{x}) \neq a)], \quad (15)$$

where the expectation is taken w.r.t. the pair (\mathbf{x}, a) . Since only samples from the joint distribution of \mathbf{x} and a are available, we replace the expectation with a sample average w.r.t. the training data $\frac{1}{n} \sum_{i=1}^n I(a(\mathbf{x}_i) \neq a_i)$. During the minimization, we search over a set of classifiers $a_x(\mathbf{x}) : \mathcal{M} \subseteq \mathbb{R}^d \rightarrow \mathcal{A}$, which is defined over a domain in \mathbb{R}^d . In our framework, we suggest replacing a classifier $a_x(\mathbf{x}) : \mathcal{M} \subseteq \mathbb{R}^d \rightarrow \mathcal{A}$ with dimensionality reduction via CCDR $f(\mathbf{x}) : \mathcal{M} \subseteq \mathbb{R}^d \rightarrow \mathbb{R}^m$ followed by a classifier on the lower-dimensional space $a_y(\mathbf{y}) : \mathbb{R}^m \rightarrow \mathcal{A}$, i.e., $a_x = a_y \circ f$. The first advantage is that the search space for the minimization in (??) defined over a d -dimensional space can be reduced to an m -dimensional space. This results in significant savings in computational complexity if the complexity associated with the process of obtaining f can be made low. In general, the classifier set \mathcal{F} has to be rich enough to attain a lower generalization error. The other advantage of our method lies in the fact that CCDR is designed

to cluster points of the same label thus allowing for a linear classifier or other low complexity classifiers. Therefore, further reduction in the size of class \mathcal{F} can be achieved in addition to the reduction due to a lower-dimensional domain. To classify a new data point, one has to apply CCDR to a new data point. If it is done brute force, the point is added to the set of training points with no label a new matrix W' is formed and an eigendecomposition is carried out.

When performing CCDR, each of the $n(n-1)/2$ terms of the form $\{\|\mathbf{x}_i - \mathbf{x}_j\|^2\}$ requires one summation and d multiplications leading to computational complexity of the order $O(dn^2)$. Construction of a K -nearest neighbors graph requires $O(kn)$ comparisons per point and therefore a total of $O(kn^2)$. The total number of operations involved in constructing the graph is therefore $O((k+d)n^2)$. Next, an eigendecomposition is applied to W' , which is an $(L+n) \times (L+n)$ matrix. The associated computation complexity is $O(n^3)$. Therefore, the overall computational complexity of CCDR is $O(n^3)$. This holds for both training and classification as explained earlier. We are interested in reducing computational complexity in training the classifier and in classification. For that purpose, we consider an out-of-sample extension of CCDR.

V. OUT-OF-SAMPLE EXTENSION

We start by rearranging the generalized eigendecomposition of the Laplacian in (??) as

$$\mathbf{G}^{(n)} \mathbf{u}_l^{(n)} = (1 - \lambda_l^{(n)}) \mathbf{D}^{(n)} \mathbf{u}_l^{(n)}, \quad (16)$$

and recall that $\mathbf{u}_l^{(n)} = [\mathbf{z}_1(l), \mathbf{z}_1(l), \dots, \mathbf{z}_1(l), \mathbf{y}_1(l), \mathbf{y}_2(l), \dots, \mathbf{y}_n(l)]^T$. Since we consider an m -dimensional embedding, we are only interested in eigenvectors $\mathbf{u}_2, \dots, \mathbf{u}_{m+1}$. The $L+i$ equation (row) for $i = 1, 2, \dots, n$ in the eigendecomposition in (??) can be written as

$$\mathbf{y}_i^{(n)}(l) = \frac{1}{1 - \lambda_l^{(n)}} \frac{\sum_k c_{ki} \mathbf{z}_k^{(n)}(l) + \beta \sum_j K(\mathbf{x}_i, \mathbf{x}_j) \mathbf{y}_j^{(n)}(l)}{\sum_k c_{ki} + \beta \sum_j K(\mathbf{x}_i, \mathbf{x}_j)}. \quad (17)$$

Similarly, the k th equation (row) of (??) for $k = 1, 2, \dots, L$ is given by

$$\mathbf{z}_k^{(n)}(l) = \frac{\sum_i c_{ki} \mathbf{y}_i^{(n)}(l)}{(1 - \lambda_l^{(n)}) n_k}. \quad (18)$$

Our interest is in finding a mapping $\mathbf{f}(\mathbf{x}, c)$ that in addition to mapping every \mathbf{x}_i to \mathbf{y}_i , can perform an out-of-sample extension, i.e., is well-defined outside the set \mathcal{X} . We consider the following out-of-sample extension expression

$$\mathbf{f}_l^{(n)}(\mathbf{x}, c) = \frac{1}{1 - \lambda_l^{(n)}} \frac{I(c \neq 0) \mathbf{z}_c^{(n)}(l) + \beta \sum_j K(\mathbf{x}, \mathbf{x}_j) \mathbf{y}_j^{(n)}(l)}{I(c \neq 0) + \beta \sum_j K(\mathbf{x}, \mathbf{x}_j)}, \quad (19)$$

where $\mathbf{z}^{(n)}$ is the same as in (??). This formula can be explain as follows. First, the lower dimensional embedding $\mathbf{y}_1^{(n)}, \dots, \mathbf{y}_n^{(n)}$ and the class centers $\mathbf{z}_1^{(n)}, \dots, \mathbf{z}_L^{(n)}$ are obtained through an the eigendecomposition in (??). Then, the embedding outside the sample set \mathcal{X} is calculated via (??). By comparison of $\mathbf{f}_l^{(n)}(\mathbf{x}_i, c_i)$ evaluated through (??) with (??), we have $\mathbf{f}_l^{(n)}(\mathbf{x}_i, c_i) = \mathbf{y}_i^{(n)}(l)$. This suggests that the out-of-sample extension coincides with the solution, we already have for the mapping at the the data points \mathcal{X} . Moreover, using this result one can replace all $\mathbf{y}_i^{(n)}$ with $\mathbf{f}_l^{(n)}(\mathbf{x}_i, c_i)$ in (??) and obtain the following generalization of the eigendecomposition in (??):

$$\mathbf{f}_l^{(n)}(\mathbf{x}, c) = \frac{1}{1 - \lambda_l^{(n)}} \frac{I(c \neq 0) \mathbf{z}_c^{(n)}(l) + \beta \sum_j K(\mathbf{x}, \mathbf{x}_j) \mathbf{f}_l^{(n)}(\mathbf{x}_j, c_j)}{I(c \neq 0) + \beta \sum_j K(\mathbf{x}, \mathbf{x}_j)}, \quad (20)$$

and

$$\mathbf{z}_k^{(n)}(l) = \frac{\sum_i c_{ki} \mathbf{f}_l^{(n)}(\mathbf{x}_i, c_i)}{(1 - \lambda_l^{(n)}) n_k}. \quad (21)$$

In [?], it is proposed that if the out-of-sample solution to the eigendecomposition problem associated with kernel PCA converge, it is given by the solution to the asymptotic equivalent of the eigendecomposition. Using similar machinery, we can provide a similar result suggesting that if $\mathbf{f}_l^{(n)}(\mathbf{x}, c) \rightarrow \mathbf{f}_l^{(\infty)}(\mathbf{x}, c)$ as $n \rightarrow \infty$, then the asymptotic equivalents to (??) and (??) should provide the solution to the limit of $\mathbf{f}_l^{(n)}(\mathbf{x}, c)$. The asymptotic analogues to (??) and (??) are described in the following. The mapping for labeled data $f_l(\mathbf{x}, c) : \mathcal{M} \times \mathcal{A} \rightarrow \mathbb{R}$ for $c = 0, 1, 2, \dots, L$ equivalent to equation (??) is

$$f_l(\mathbf{x}, c) = \frac{1}{1 - \lambda_l} \frac{I(c \neq 0) z_c(l) + \beta' \sum_{c'=0}^L \int_{\mathcal{M}} K(\mathbf{x}, \mathbf{x}') f_l(\mathbf{x}', c') P(\mathbf{x}', c') d\mathbf{x}'}{I(c \neq 0) + \beta' \int_{\mathcal{M}} K(\mathbf{x}, \mathbf{x}') p(\mathbf{x}') d\mathbf{x}'} \quad (22)$$

where $z_c(l)$ for $c = 1, 2, \dots, L$ is equivalent to (??)

$$z_c(l) = \frac{\int_{\mathcal{M}} f_l(\mathbf{x}, c) p(\mathbf{x}|c) d\mathbf{x}}{1 - \lambda_l}, \quad (23)$$

and $\beta' = \beta n$. Since we are interested in an m -dimensional embedding, we consider only $l = 1, 2, \dots, m$, i.e., the eigenvectors that correspond to the m smallest eigenvalues. To guarantee that the relevant eigenvectors are unique (up to a multiplicative constant), we require $\lambda_1 < \lambda_2 < \dots < \lambda_{m+1} \leq \lambda_{m+2} \leq \dots \lambda_n$.

The out-of-sample extension given by (??), can be useful in a couple of scenario. The first, is in classification of new unlabeled samples. We assume that $\{\mathbf{y}_j\}_{j=1}^n$, $\{\mathbf{z}_k\}_{k=1}^L$, and $\{\lambda_l\}_{l=1}^m$

are already obtained based on labeled (or partially labeled) training data and we would like to embed a new unlabeled data point. We consider using (??) with $c = 0$, i.e., we can use $\mathbf{f}(\mathbf{x}, 0)$ to map a new sample \mathbf{x} to \mathbb{R}^m . The obvious immediate advantage is the savings in computational complexity as we avoid performing addition eigendecomposition that includes the new point.

The second scenario involves the out-of-sample extension for labeled data. The goal here is not to classify the data since the label is already available. Instead, we are interested in the training phase in the case of large n for which the eigendecomposition is infeasible. In this case, a large amount of labeled training data is available but due to the heavy computational complexity associated with the eigendecomposition in (??) (or by (??)), the data cannot be processed. In this case, we are interested in developing a resampling method, which integrates $\mathbf{f}_l^{(n)}(\mathbf{x}, c)$ obtained for different subsamples of the complete data set. We illustrate the idea of the method in the the appendix.

A. Classification Algorithms

We consider three widespread algorithms: k -nearest neighbors, linear classification, and neural networks. A standard implementation of k -nearest neighbors was used, see [?, p. 415]. The linear classifier we implemented is given by

$$\hat{c}(\mathbf{y}) = \arg \max_{c \in \{\mathcal{A}_1, \dots, \mathcal{A}_L\}} \mathbf{y}^T \boldsymbol{\alpha}^{(c)} + \alpha_0^{(c)}$$

$$[\boldsymbol{\alpha}^{(\mathcal{A}_k)}, \alpha_0^{(\mathcal{A}_k)}] = \arg \min_{[\boldsymbol{\alpha}, \alpha_0]} \sum_{i=1}^n (\mathbf{y}_i^T \boldsymbol{\alpha} + \alpha_0 - c_{ki})^2,$$

for $k = 1, \dots, L$. The neural network we implemented is a three-layer neural network with d elements in the input layer, $2d$ elements in the hidden layer, and 6 elements in the output layer (one for each class). Here d was selected using the common PCA procedure, as the smallest dimension that explains 99.9% of the energy of the data. A gradient method was used to train the network coefficients with 2000 iterations. The neural net is significantly more computationally burdensome than either linear or k -nearest neighbors classifications algorithms.

B. Data Description

In this section, we examine the performance of the classification algorithms on the benchmark label classification problem provided by the Landsat MSS satellite imagery database [?]. Each

sample point consists of the intensity values of one pixel and its 8 neighboring pixels in 4 different spectral bands. The training data consists of 4435 36-dimensional points of which, 1072 are labeled as 1) red soil, 479 as 2) cotton crop, 961 as 3) grey soil, 415 as 4) damp grey soil, 470 are labeled as 5) soil with vegetation stubble, and 1038 are labeled as 6) very damp grey soil. The test data consists of 2000 36-dimensional points of which, 461 are labeled as 1) red soil, 224 as 2) cotton crop, 397 as 3) grey soil, 211 as 4) damp grey soil, 237 are labeled as 5) soil with vegetation stubble, and 470 are labeled as 6) very damp grey soil. In the following, each classifier is trained on the training data and its classification is evaluated based on the entire sample test data. In Table ??, we present “best case” performance of neural networks, linear classifier, and k -nearest neighbors in three cases: no dimensionality reduction, dimensionality reduction via PCA, and dimensionality reduction via CCDR. The table presents the minimum probability of error achieved by varying the tuning parameters of the classifiers. The benefit of using CCDR is obvious and we are prompted to further evaluate the performance gains attained using CCDR.

	Neural Net.	Lin.	k -nearest neigh.
No dim. reduc.	83 %	22.7 %	9.65 %
PCA	9.75 %	23 %	9.35 %
CCDR	8.95 %	8.95 %	8.1 %

TABLE I

CLASSIFICATION ERROR PROBABILITY

C. Regularization Parameter β

As mentioned earlier, the CCDR regularization parameter β controls the contribution of the label information versus the contribution of the geometry described by the sample. We apply CCDR to the 36-dimensional data to create a 14-dimensional embedding by varying β over a range of values. For justification of our choice of $d = 14$ dimensions see Section ?. In the process of computing the weights w_{ij} for the algorithm, we use k -nearest neighbors with $k = 4$ to determine the local neighborhood. Fig. ?? shows the classification error probability (dashed lines) for the linear classifier vs. β after preprocessing the data using CCDR with $k = 4$ and

dimension 14. We observe that for a large range of β the average classification error probability is greater than 0.09 but smaller than 0.095. This performance competes with the performance of k -nearest neighbors applied to the high-dimensional data, which is presented in [?] as the leading classifier for this benchmark problem. Another observation is that for small values of β (i.e., $\beta < 0.1$) the probability of error is constant. For such small value of β , classes in the lower-dimensional embedding are well-separated and are well-concentrated around the class centers. Therefore, the linear classifier yields perfect classification on the training set and fairly low constant probability of error on the test data is attained for low value of β . When β is increased, we notice an increase in the classification error probability. This is due to the fact that the training data become non separable by any linear classifier as β increases.

We perform a similar study of classification performance for k -nearest neighbors. In Fig. ??, classification probability error is plotted (dotted lines) vs. β . Here, we observed that an average error probability of 0.086 can be achieved for $\beta \approx 0.5$. Therefore, k -nearest neighbors preceded by CCDR outperforms the straightforward k -nearest neighbors algorithm. We also observe that when β is decreased the probability of error is increased. This can be explained as due to the ability of k -nearest neighbors to utilize local information, i.e., local geometry. This information is discarded when β is decreased.

We conclude that CCDR can generate lower-dimensional data that is useful for global classifiers, such as the linear classifier, by using a small value of β , and also for local classifiers, such as k -nearest neighbors, by using a larger value β and thus preserving local geometry information.

Fig. 2. Probability of incorrect classification vs. β for a linear classifier (dotted line \circ) and for the k -nearest neighbors algorithm (dashed line \diamond) preprocessed by CCDR. 80% confidence intervals are presented as \times for the linear classifier and as $+$ for the k -nearest neighbors algorithm.

D. Dimension Parameter

While the data points in \mathcal{X}_n may lie on a manifold of a particular dimension, the actual dimension required for classification may be smaller. Here, we examine classification performance

Fig. 3. Probability of incorrect classification vs. CCDR’s dimension for a linear classifier (dotted line \circ) and for the k -nearest neighbors algorithm (dashed line \diamond) preprocessed by CCDR. 80% confidence intervals are presented as \times for the linear classifier and as $+$ for the k -nearest neighbors algorithm.

as a function of the CCDR dimension. Using the entropic graph dimension estimation algorithm in [?], we obtain the following estimated dimension for each class:

class	1	2	3	4	5	6
dimension	13	7	13	10	6	13

Therefore, if an optimal nonlinear embedding of the data could be found, we suspect that a dimension greater than 13 may not yield significant improvement in classification performance. Since CCDR does not necessarily yield an optimal embedding, we choose CCDR embedding dimension as $d = 14$ in Section ??.

In Fig. ??, we plot the classification error probability (dotted line) vs. CCDR dimension and its confidence interval for a linear classifier. We observed decrease in error probability as the dimension increases. When the CCDR dimension is greater than 5, the error probability seems fairly constant. This is an indication that CCDR dimension of 5 is sufficient for classification if one uses the linear classifier with $\beta = 0.5$, i.e., linear classifier cannot exploit geometry.

We also plot the classification error probability (dashed line) vs. CCDR dimension and its confidence interval for k -nearest neighbors classifier. Generally, we observe decrease in error probability as the dimension increases. When the CCDR dimension is greater than 5, the error probability seems fairly constant. When CCDR dimension is three, classifier error is below 0.1. On the other hand, minimum possibility of error obtained at CCDR dimension 12-14. This is remarkable agreement with the dimension estimate of 13 obtained using the entropic graph algorithm of [?].

E. CCDR’s k -Nearest Neighbors Parameter

The last parameter we examine is the CCDR’s k -nearest neighbors parameter. In general, as k increases non-local distances are included in the lower-dimensional embedding. Hence, very large k prevents the flexibility necessary for dimensionality reduction on (globally) non-linear (but locally linear) manifolds.

Fig. 4. Probability of incorrect classification vs. CCDR's k -nearest neighbors parameter for a linear classifier (dotted line \circ) and for the k -nearest neighbors algorithm (dashed line \diamond) preprocessed by CCDR. 80% confidence intervals are presented as \times for the linear classifier and as $+$ for the k -nearest neighbors algorithm.

In Fig. ??, the classification probability of error for the linear classifier (dotted line) is plotted vs. the CCDR's k -nearest neighbors parameter. A minimum is obtained at $k = 3$ with probability of error of 0.092. The classification probability of error for k -nearest neighbors (dashed line) is plotted vs. the CCDR's k -nearest neighbors parameter. A minimum is obtained at $k = 4$ with probability of error of 0.086.

VI. CONCLUSION

In this paper, we presented the CCDR algorithm for multiple classes. We examined the performance of various classification algorithms applied after CCDR for the Landsat MSS imagery dataset. We showed that for a linear classifier, decreasing β yields improved performance and for a k -nearest neighbors classifier, increasing β yields improved performance. We demonstrated that both classifiers have improved performance on the much smaller dimension of CCDR embedding space than when applied to the original high-dimensional data. We also explored the effect of k in the k -nearest neighbors construction of CCDR weight matrix on classification performance. CCDR allows reduced complexity classification such as the linear classifier to perform better than more complex classifiers applied to the original data. We are currently pursuing an out-of-sample extension to the algorithm that does not require rerunning CCDR on test and training data to classify new test point.

APPENDIX I

PROOF OF ...

Appendix one text goes here.

APPENDIX II

We consider the following generalized eigendecomposition

$$\mathbf{W}'\mathbf{u}_l = \lambda_l \text{diag}\{\mathbf{W}'\mathbf{1}\}\mathbf{u}_l. \quad (24)$$

For the ϵ -ball or K -nn neighborhoods, one assumes a local neighborhood, i.e., asymptotically the number of neighbors in a given neighborhood (e.g., K) becomes negligible as compared to the total number of points: $\frac{K}{n} \rightarrow 0$ as $n \rightarrow \infty$. Using order notation, we can say that $K \sim o(n)$. An interesting setting is when the number of points in a neighborhood of a point is increasing with n , but the size of the neighborhood decreasing. This can be achieved, for example, by setting $K \sim O(1/\sqrt{n})$. The implication is that the weighting matrix has $O(n)$ zero-valued elements at each column and only $O(K) \sim o(n)$ number of elements are $O(1)$ (when $0 \leq \mathbf{W}'_{ij} \leq 1$).

When every point is labeled, we have precisely one 1 in each column of \mathcal{C} , which allows us to make the same assumptions we made in the previous paragraph on the matrix \mathcal{M} regarding the order of the number of nonzero elements. While the nonzero elements in \mathcal{C} are 1 and therefore $O(1)$, it is not obvious that the same holds for $\beta\mathbf{W}$ as β may vary.

The generalized eigendecomposition in (??) is equivalent to the following standard eigendecomposition

$$\mathbf{G}\mathbf{v}_l = \lambda_l\mathbf{v}_l, \quad (25)$$

where $\mathbf{G} = \text{diag}\{\mathbf{W}'\mathbf{1}\}^{-\frac{1}{2}}\mathbf{W}'\text{diag}\{\mathbf{W}'\mathbf{1}\}^{-\frac{1}{2}}$ and $\mathbf{v}_l = \text{diag}\{\mathbf{W}'\mathbf{1}\}^{\frac{1}{2}}\mathbf{u}_l$. We make the following assumption:

- 1) The matrix \mathbf{W}' is $n \times n$, symmetric, with only $O(K)$ of the coefficients of each column being $O(1)$ (the rest of its coefficients are 0).
- 2) The elements of \mathbf{u}_l are $O(\alpha)$, where α is selected so that the elements \mathbf{v}_l are $O(1)$. The elements of \mathbf{v}_l would form the l -th coordinate for every point. This condition would allow for a pointwise convergence of the mapping of $f^{(n)}(\cdot)$ at \mathbf{x}_l to some nonzero \mathbf{y}_l .

Therefore:

- 1) The matrix \mathbf{G} is $n \times n$, symmetric, with only $O(K)$ of the coefficients of each column being $O(1/K)$ (the rest of its coefficients are 0).
- 2) By the eigendecomposition,

$$\lambda_l = \frac{\mathbf{u}_l^T \mathbf{G} \mathbf{u}_l}{\mathbf{u}_l^T \mathbf{u}_l}. \quad (26)$$

Since \mathbf{G} is of order $O(1/K)$ with only $O(K)$ coefficients in each row being nonzero, λ_l is of order $O(1)$.

To find the embedding of a new point \mathbf{x}_{n+1} we consider the eigendecomposition of a matrix that is extended as follows

$$\begin{bmatrix} \mathbf{G} + \delta\mathbf{G} & \mathbf{g} \\ \mathbf{g}^T & g \end{bmatrix} \begin{bmatrix} \mathbf{u}_l + \delta\mathbf{u}_l \\ \xi_l \end{bmatrix} = (\lambda_l + \delta\lambda_l) \begin{bmatrix} \mathbf{u}_l + \delta\mathbf{u}_l \\ \xi_l \end{bmatrix} \quad (27)$$

By the argument used to determine the order of the elements of \mathbf{G} , the elements of \mathbf{g} are also $O(1/K)$ for only $O(K)$ and the rest are zero. Similarly, g is $O(1/K)$. However, $\delta\mathbf{G}$ is $O(1/K^2)$ for only $O(K^2)$ of its n^2 elements. Since the new eigenvector $[\mathbf{u}_l^T + \delta\mathbf{u}_l^T, \xi_l]^T$ is the collection of the value of the l th coordinate of every vector. The last element ξ_l is the l coordinate of the $(n+1)$ th point. Therefore, to find the embedding of the $(n+1)$ th point, we need to find ξ_l for $1 \leq l \leq d$, as $\mathbf{y}_{n+1} = \text{diag}\{\mathcal{M}\mathbf{1}\}^{\frac{1}{2}}[\xi_1, \dots, \xi_d]^T$. We start with the last row of (??):

$$\mathbf{g}^T(\mathbf{u}_l + \delta\mathbf{u}_l) + g\xi_l = (\lambda_l + \delta\lambda_l)\xi_l. \quad (28)$$

After collecting the terms with ξ_l at the LHS, we have

$$\lambda_l(1 + \frac{\delta\lambda_l - g}{\lambda_l})\xi_l = \mathbf{g}^T(\mathbf{u}_l + \delta\mathbf{u}_l) \quad (29)$$

Taking the dominant terms on both side, we have

$$\xi_l = \frac{\mathbf{g}^T \mathbf{u}_l}{\lambda_l} + o(\alpha), \quad (30)$$

which is $O(\alpha)$, i.e., the same order as the elements of \mathbf{u}_l . Note that if $1/K \rightarrow 0$ as $N \rightarrow \infty$, then $g \sim o(1)$. To verify the result in (??), it remains to show that $\delta\mathbf{u}_l \sim o(\alpha)$ and that $\delta\lambda_l \sim o(1)$.

We continue exploring this result in the context of the dimensionality reduction problem.

Expressing ξ_l in terms of \mathcal{M} , we have

$$\xi_l = \frac{\sum_i \mathcal{M}_{i,n+1}[\mathbf{u}_l]_i}{\lambda_l \sqrt{\sum_s \mathcal{M}_{s,n+1} \sum_s \mathcal{M}_{i,s}}} + o(\alpha). \quad (31)$$

To compute the embedding of the $(n+1)$ th point, we need to express the results in terms of the generalized eigenvectors \mathbf{v}_l rather than the standard eigenvectors \mathbf{u}_l :

$$[y_{n+1}]_l = [\text{diag}\{\mathcal{M}\mathbf{1}\}^{-\frac{1}{2}}]_{n+1,n+1} \xi_l = \frac{\sum_i \mathcal{M}_{i,n+1}[\text{diag}\{\mathcal{M}\mathbf{1}\}^{\frac{1}{2}} \mathbf{v}_l]_i}{\lambda_l \sum_s \mathcal{M}_{s,n+1} \sqrt{\sum_s \mathcal{M}_{i,s}}} \quad (32)$$

$$= \frac{\sum_i \mathcal{M}_{i,n+1} \sqrt{\sum_s \mathcal{M}_{i,s}} [\mathbf{v}_l]_i}{\lambda_l \sum_s \mathcal{M}_{s,n+1} \sqrt{\sum_s \mathcal{M}_{i,s}}} \quad (33)$$

$$= \frac{\sum_i \mathcal{M}_{i,n+1} [\mathbf{v}_l]_i}{\lambda_l \sum_s \mathcal{M}_{s,n+1}} \quad (34)$$

$$= \frac{\sum_i \mathcal{M}_{i,n+1} [\mathbf{y}_i]_l}{\lambda_l \sum_s \mathcal{M}_{s,n+1}}. \quad (35)$$

For brevity we omit $o(\alpha)$. In vector format, the embedding of the new point \mathbf{x}_{n+1} is

$$\mathbf{y}_{n+1} = \mathbf{\Lambda}^{-1} \frac{\sum_i \mathcal{M}_{i,n+1} \mathbf{y}_i}{\sum_s \mathcal{M}_{s,n+1}}. \quad (36)$$

This formula offers a method of out-of-sample extension for new samples for Laplacian eigenmaps and CCDR. In contrast, the formula in (??) appears in [?] in the context of out-of-sample extension to Laplacian eigenmaps. The way to correctly utilize (??) is by recalling the transformation back to the formulation of the generalized eigen transformation (i.e., from \mathbf{u}_l to \mathbf{v}_l). When no label is given, we can replace \mathcal{M} with \mathbf{W} :

$$\mathbf{y}_{n+1} = \mathbf{\Lambda}^{-1} \frac{\sum_i \mathbf{W}_{i,n+1} \mathbf{y}_i}{\sum_s \mathbf{W}_{s,n+1}}, \quad (37)$$

which is the out-of-sample extension for Laplacian eigenmaps. Ignoring $\mathbf{\Lambda}^{-1}$, \mathbf{y}_{n+1} the embedding of the new point \mathbf{x}_{n+1} is an average (or even a convex combination) of the embedding of its neighbors. When some labels are given then \mathcal{M} is given by (??), and therefore

$$\mathbf{y}_{n+1} = \mathbf{\Lambda}^{-1} \frac{\beta \sum_i \mathbf{W}_{i,n+1} \mathbf{y}_i + \sum_k c_{k,n+1} \mathbf{z}_k}{\beta \sum_s \mathbf{W}_{s,n+1} + \sum_k c_{k,n+1}}. \quad (38)$$

An interesting observation is that this formula is only indirectly dependent on the label of point \mathbf{x}_i for $1 \leq i \leq n$ through \mathbf{z}_k . The only label that directly controls the embedding of the new point is its own label. If the new sample belongs to class k , we have

$$\mathbf{y}_{n+1} = \mathbf{\Lambda}^{-1} \left(\beta \frac{\sum_i \mathbf{W}_{i,n+1} \mathbf{y}_i}{\beta \sum_s \mathbf{W}_{s,n+1} + 1} + \frac{\mathbf{z}_k}{\beta \sum_s \mathbf{W}_{s,n+1} + 1} \right). \quad (39)$$

For a very small β , we have $\mathbf{y}_{n+1} \approx \mathbf{\Lambda}^{-1} \mathbf{z}_k$, i.e., a new point is mapped to its class center when geometry information is ignored. For very large β , we have (??) and the embedding is controlled only geometry information.

Next, we examine $\delta\lambda_l$ and $\delta\mathbf{u}_l$. By taking the top part of (??) and multiplying on the left by \mathbf{u}_l , we can obtain an equation for $\delta\lambda_l$:

$$\delta\lambda_l \left(1 + \frac{\mathbf{u}_l^T \delta\mathbf{u}_l}{\mathbf{u}_l^T \mathbf{u}_l} \right) = \frac{\mathbf{u}_l^T \delta\mathbf{G} \mathbf{u}_l}{\mathbf{u}_l^T \mathbf{u}_l} + \frac{\mathbf{u}_l^T \delta\mathbf{G} \delta\mathbf{u}_l}{\mathbf{u}_l^T \mathbf{u}_l} + \frac{\mathbf{u}_l^T \mathbf{g} \xi_l}{\mathbf{u}_l^T \mathbf{u}_l} \quad (40)$$

Following our assumptions on $\delta\mathbf{u}_l$, we collect only dominating terms and obtain the following expression for $\delta\lambda_l$:

$$\delta\lambda_l = \frac{\mathbf{u}_l^T \delta\mathbf{G} \mathbf{u}_l}{\mathbf{u}_l^T \mathbf{u}_l} + \frac{(\mathbf{u}_l^T \mathbf{g})^2}{\lambda_l \mathbf{u}_l^T \mathbf{u}_l} + o(1/n) = \frac{\mathbf{u}_l^T (\delta\mathbf{G} + \frac{\mathbf{g}\mathbf{g}^T}{\lambda_l}) \mathbf{u}_l}{\mathbf{u}_l^T \mathbf{u}_l} + o(1/n). \quad (41)$$

By taking the top part of (??), we can obtain an equation for $\delta \mathbf{u}_l$:

$$(\mathbf{G} + \delta \mathbf{G} - (\lambda_l + \delta \lambda_l) \mathbf{I}) \delta \mathbf{u}_l = -(\delta \mathbf{G} - \delta \lambda_l \mathbf{I}) \mathbf{u}_l - \mathbf{g} \xi_l. \quad (42)$$

Substituting the ξ_l from (??) and reorganizing, we have

$$(\mathbf{G} + \delta \mathbf{G} - (\lambda_l + \delta \lambda_l) \mathbf{I}) \delta \mathbf{u}_l = -\left(\mathbf{I} - \frac{\mathbf{u}_l \mathbf{u}_l^T}{\mathbf{u}_l^T \mathbf{u}_l}\right) \left(\delta \mathbf{G} + \frac{\mathbf{g} \mathbf{g}^T}{\lambda_l}\right) \mathbf{u}_l. \quad (43)$$

Collecting the dominant terms, we have

$$(\mathbf{G} - \lambda_l \mathbf{I}) \delta \mathbf{u}_l = -\left(\mathbf{I} - \frac{\mathbf{u}_l \mathbf{u}_l^T}{\mathbf{u}_l^T \mathbf{u}_l}\right) \left(\delta \mathbf{G} + \frac{\mathbf{g} \mathbf{g}^T}{\lambda_l}\right) \mathbf{u}_l. \quad (44)$$

Note that the RHS depends on the perturbation terms $\delta \mathbf{G}$ and \mathbf{g} and is of order $O(\alpha/K)$ for only $O(K)$ terms.

APPENDIX III

PROOF OF THE OUT OF SAMPLE APPROXIMATION

In this section, we illustrate the method of approximation of out of sample extensions by selecting subsequences of the original data set and forming sequences of good approximants to the original function. We recall that we are concerned with the out-of-sample extension for labeled data. The goal here is not to classify the data since the label is already available. Instead, we are interested in the training phase in the case of large n for which the eigendecomposition is infeasible. In this case, a large amount of labeled training data is available but due to the heavy computational complexity associated with the eigendecomposition in (??) (or by (??)), the data cannot be processed. In this case, we are interested in developing a resampling method, which approximates $\mathbf{f}_l^{(n)}(\mathbf{x}, c)$ obtained for different subsamples of the complete data set.

Throught this section, we assume without loss of generality that l and c are fixed. Throughtout, C denotes a positive absolute constant which may take on different values at different times. Suppose that

$$y_j^n = y + O\left(\frac{1}{\varepsilon_n}\right), \quad n \rightarrow \infty$$

uniformly for all $1 \leq j \leq n$ where y is some absolute constant and ε_n is a strictly increasing positive sequence with limit infinity at infinity. Let δ_n be a positive, strictly increasing sequence with limit infinity at infinity and satisfying

$$\frac{\delta_n}{\varepsilon_n} \rightarrow 0, \quad n \rightarrow \infty.$$

Let F_Q denote the space of all $Q : \mathbb{R} \rightarrow [0, \infty)$ with Q strictly increasing with $Q(0) = 0$ and let C_n denote the space of all positive sequences c_n with limit zero at infinity. Define for $n \geq 1$,

$$K(x, y, c_n, Q) := \exp\left(-\frac{Q(\|x - y\|)}{c_n}\right), \quad x, y \in M, \quad c_n \in C_n, \quad Q \in F_Q.$$

Let $n \geq 1$, $X_n := \{x_j : 1 \leq j \leq n\}$ a collection of points in M and for $c_n \in C_n$ and $Q \in F_Q$, let

$$f^n(x) := \sum_{j=1}^{[\delta_n]} K(x, x_j, c_n, Q) y_j^n, \quad x \in M, \quad x_j \in X_n.$$

Here, $[x]$ denotes the largest integer $\leq x$, $x \in M$. Following is our main estimate which lays the foundation of our method but is also of independent interest.

Let $\gamma_1, \gamma_2 > 0$. For $1 \leq m < n$, let δ_m be any strictly increasing positive sequence with limit infinity at infinity, let $Q \in F_Q$ and $c_n \in C_n$. Then, uniformly for all large enough m and uniformly for all $x \in M$ satisfying

$$|x - x_j| \geq Q^{-1}(j^{1+\gamma_1+\gamma_2} c_n), \quad x_j \in X_n \setminus X_m$$

we have

$$|f^n(x) - f^m(x)| = O\left(\frac{1}{\delta_m}\right)^{\max(\gamma_1, \gamma_2)} + O\left(\frac{\delta_m}{\varepsilon_m}\right).$$

The essence of this estimate is that it says that if we choose ANY subsequence $X_{m(n)}$ of X_n with $1 \leq m(n) < n$ and with $m(n)$ positive, strictly increasing and having limit infinity at infinity, then $f^{m(n)}(x)$ is asymptotically equal to $f^n(x)$ for all x away from X_m in neighborhoods which typically shrink for large m at a precise rate determined by the kernel.

Proof We first observe that we may write

$$f^n(x) - f^m(x) = I_1(x) + I_2(x)$$

where

$$I_1(x) := \sum_{j=1}^{[\delta_m]} K(x, x_j, c_n, Q) (y_j^n - y_j^m).$$

and

$$I_2(x) := \sum_{j=[\delta_m]+1}^{[\delta_n]} K(x, x_j, c_n, Q) y_j^n.$$

We begin with the estimate of I_1 . Now as $Q \in F_Q$, $c_n \in C_n$, we have easily that for every $x \in M$ and $x_j \in X_n$ that $K(x, x_j, c_n, Q)$ is uniformly bounded from above, so using this and the asymptotics of y_j^n , we see that we have

$$|I_1|(x) \leq C \sum_{j=1}^{[\delta_m]} |(y_j^n - y_j^m)| = O\left(\frac{\delta_m}{\varepsilon_m}\right).$$

Next, we write

$$I_2(x) = I_{2,1}(x) + I_{2,2}(x)$$

where

$$I_{2,1}(x) := \sum_{[\delta_m]+1 \leq j \leq [\delta_n], j: |x-x_j| \geq Q^{-1}(j^{1+\gamma_1+\gamma_2}c_n)}$$

and

$$I_{2,2}(x) := \sum_{[\delta_m]+1 \leq j \leq [\delta_n], j: |x-x_j| < Q^{-1}(j^{1+\gamma_1+\gamma_2}c_n)}$$

Now, we see that

$$|I_{2,1}|(x) \leq C \sum_{j=[\delta_m]+1}^{[\delta_n]} \frac{1}{j^{1+\gamma_1+\gamma_2}}, x \in M.$$

Assume without loss of generality that $\gamma_1 \geq \gamma_2$. Then we can continue the estimate above by

$$|I_{2,1}|(x) \leq C \left(\frac{1}{\delta_m}\right)^{\gamma_1} \sum_{j=1}^{\infty} \frac{1}{j^{1+\gamma_2}}$$

and so we see that indeed

$$|I_{2,1}|(x) = O\left(\frac{1}{\delta_m}\right)^{\max(\gamma_1, \gamma_2)}$$

as required. To complete the proof, we see that by definition, $I_{2,2}(x)$ is in fact empty for $x \in M$ satisfying

$$|x - x_j| \geq Q^{-1}(j^{1+\gamma_1+\gamma_2}c_n), x_j \in X_n \setminus X_m.$$

We remark that we may indeed adapt the proof to allow c_n to be replaced by c_j everywhere in the definition of the kernel. In this sense, we obtain kernels which also update with j .

Name Biography text here.