Distortion-Free Nonlinear Dimensionality Reduction

Yangqing Jia, Zheng Wang, and Changshui Zhang

State Key Laboratory of Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology (TNList)
Department of Automation, Tsinghua University, Beijing, China
{jiayq06,w-z04}@mails.tsinghua.edu.cn, zcs@mail.tsinghua.edu.cn

Abstract. Nonlinear Dimensionality Reduction is an important issue in many machine learning areas where essentially low-dimensional data is nonlinearly embedded in some high-dimensional space. In this paper, we show that the existing Laplacian Eigenmaps method suffers from the distortion problem, and propose a new distortion-free dimensionality reduction method by adopting a local linear model to encode the local information. We introduce a new loss function that can be seen as a different way to construct the Laplacian matrix, and a new way to impose scaling constraints under the local linear model. Better low-dimensional embeddings are obtained via constrained concave convex procedure. Empirical studies and real-world applications have shown the effectiveness of our method.

1 Introduction

Dimensionality reduction is an important issue in many pattern recognition and machine learning areas, where essentially low dimensional data is often embedded in some high dimensional space. Early works on dimensionality reduction are mostly linear methods such as PCA [1] and MDS [2], and have been widely applied and discussed. However, in recent years, researchers have realized that in many situations the data lies on a low-dimensional manifold embedded in the feature space, and the embedding is often difficult to be captured by the simple linear model. Thus, nonlinear dimensionality reduction (NLDR) is believed to be more powerful to preserve the low dimensional information under such situations.

Several NLDR methods have been proposed in the recent years, such as ISOMAP [3], Locally Linear Embedding (LLE) [4], local tangent space alignment (LTSA) [5], maximum variance unfolding (MVU) [6], and so on. In this paper, we will mainly focus on methods that rely on Laplacian matrices, the most representative work of which is the Laplacian Eigenmaps [7]. It constructs a Laplacian matrix by constructing a graph over the data set, and calculates the low-dimensional coordinates via generalized eigenvalue decomposition. The method has been simplified to a linear version called Locality Preserving Projections [8], which benefits from both the nonlinear graph structure and the simplicity of the linearity.

One of the key points for Laplacian Eigenmaps is how to construct the Laplacian matrix. Most commonly, it is constructed based on K-nearest neighbor and a Gaussian kernel to calculate the pairwise similarity between data points. Although its theoretical foundations and relationship with the Laplacian-Beltrami operator have been well studied (see *e.g.* [9]), it may not be universally the best option. Specifically, we will discuss in this paper that it has a *distortion problem*: it tends to drive the data points far from the center of the low-dimensional coordinate and may also expand the small “holes” in the intrinsic distribution of the data.

In this paper, we propose a different way to construct the Laplacian matrix by explicitly defining a linear model in each local area of the data. Next, we use the local gradients obtained from the linear model to impose additional scaling constraints to solve the distortion problem. We adopt the *constrained concave convex procedure* (CCCP) to solve the optimization problem, and show the performance of the new dimensionality reduction both visually and quantitatively.

The remainder of this paper is organized as follows: Section 2 introduces the basic notations and the distortion problem of Laplacian Eigenmaps. Section 3 discusses the loss function of our method, leading to the construction of a new Laplacian matrix, and the implementation of the scaling criterions. Section 4 shows experimental results comparing our method and the closely related methods with discussions. Finally, Section 5 concludes the paper.

2 Laplacian Eigenmaps and the Distortion Problem

In this section, after defining necessary notations, we will discuss the distortion problem of the Laplacian Eigenmaps.

2.1 Notations

Formally, in nonlinear dimensionality reduction problems, we are given a set of D -dimensional data points $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\} \subset \mathbb{R}^D$. The task of NLDR is to find a corresponding set $\mathcal{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n\} \subset \mathbb{R}^d$ ($d \ll D$), where $\mathbf{y}_i$ is the low dimensional representation of $\mathbf{x}_i$. The coordinates of each low dimension can also be seen as being calculated from a mapping function $f^i : \mathbb{R}^D \rightarrow \mathbb{R}$, where $i = 1, \dots, d$. Thus, we define the dimension-wise set $\mathcal{F} = \{\mathbf{f}^1, \mathbf{f}^2, \dots, \mathbf{f}^d\} \subset \mathbb{R}^n$, where the j -th element of $\mathbf{f}^i$ is the i -th dimensional value of $\mathbf{y}_j$.

The Laplacian Eigenmaps [7] method finds the low-dimensional representation by minimizing the weighted sum of the squared distances between neighboring data points:

$$\mathcal{J}(\mathbf{f}_1, \mathbf{f}_2, \dots, \mathbf{f}_d) = \sum_{k=1}^d (\mathbf{f}^k)^\top \mathbf{L} \mathbf{f}^k = \sum_{k=1}^d \sum_{i,j=1}^n (f_i^k - f_j^k)^2 w_{ij} \quad . \quad (1)$$

$\mathbf{L}$ is the graph Laplacian matrix with its (i, j) -th element

$$L_{ij} = \begin{cases} d_i - w_{ii}, & \text{if } i = j \\ -w_{ij}, & \text{otherwise} \end{cases} \quad , \quad (2)$$

where $d_i = \sum_j w_{ij}$, and w_{ij} is the similarity between data point $\mathbf{x}_i$ and $\mathbf{x}_j$. There are several different ways to calculate the similarity, among which the weighted exponential similarity is most often used: let d_{ij} be the distance between two data points $\mathbf{x}_i$ and $\mathbf{x}_j$, the weight is calculated as

$$w_{ij} = \exp \left[-d_{ij}^2 / (2\sigma^2) \right] , \quad (3)$$

where σ is a parameter that controls the width of the Gaussian function. Usually, a precedent K -nearest neighborhood is calculated to force $w_{ij} = 0$ if $\mathbf{x}_i$ and $\mathbf{x}_j$ are not neighbors.

To recover the low-dimensional coordinates, Belkin *et al.* [7] proposed to solve the following optimization problem:

$$\begin{aligned} \min \quad & \sum_{k=1}^d (\mathbf{f}^k)^\top \mathbf{L} \mathbf{f}^k \\ \text{s.t.} \quad & (\mathbf{f}^i)^\top \mathbf{D} \mathbf{f}^j = \delta_{ij}, \quad (\mathbf{f}^i)^\top \mathbf{D} \mathbf{1} = 0, \quad 1 \leq i, j \leq d , \end{aligned} \quad (4)$$

where δ_{ij} is the *kronecker delta* function that takes value 1 if $i = j$ and 0 otherwise. $\mathbf{D}$ is the diagonal degree matrix with the i -th diagonal element be d_i . This is further solved by the following generalized eigenvalue decomposition problem:

$$\mathbf{L} \mathbf{f} = \lambda \mathbf{D} \mathbf{f} , \quad (5)$$

and the d eigenvectors corresponding to the smallest non-zero eigenvalue are the low-dimensional coordinates (The eigenvector corresponding to eigenvalue 0 is constant vector $\mathbf{1}$ and is discarded).

We note here that solving the eigenvalue decomposition problem $\mathbf{L} \mathbf{f} = \lambda \mathbf{f}$ is also applicable. From a spectral clustering view [10], the former is equivalent to minimizing the Normalized Cut [11] criterion on the corresponding graph defined by the Laplacian matrix, and the latter is equivalent to minimizing the Ratio Cut [12] criterion.

2.2 The Distortion Problem

To show the distortion problem, we construct a toy *clover* data shown in Fig. 1(a) and consider using Laplacian Eigenmaps to recover the 2-dimensional coordinates¹. From the recovered coordinates under different σ values shown in Fig. 1(b), we can see that Laplacian Eigenmaps may not recover the low-dimensional data satisfactorily: the circular structure of the data is expanded and the data area with higher density (the blue dots) is driven far from the center of the low-dimensional embedding. This is essentially different from the ground truth. Another example is the recovered embedding of the Swiss roll, see Fig. 3. Laplacian Eigenmaps tends to expand the “holes” in the intrinsic data distribution, and squeeze the data points into lines of high-density areas, which produces distorted results.

¹ For details about constructing the toy data, see Section 4.

Fig. 1. The distortion problem of Laplacian Eigenmaps. (a) The original toy clover data; (b) recovered coordinates using Laplacian Eigenmaps with $\sigma = 0.05, 0.1, 1$, and $+\infty$ respectively; (c) example of a one-dimensional uniformly spaced data (blue points on the left) showing that Laplacian Eigenmaps tend to drive the recovered coordinates (red points on the right) to the two ends, very similar to performing clustering.

The distortion problem roots largely in two aspects. First, the loss function of Laplacian Eigenmaps considers minimizing the pairwise distance between neighboring data points only and does not consider any isometric information, thus it may essentially neglect the intrinsic shape of the data. Second, Laplacian Eigenmaps can be seen as minimizing $\mathbf{f}^\top \mathbf{L} \mathbf{f}$ and maximizing $\mathbf{f}^\top \mathbf{D} \mathbf{f}$ at the same time. Although maximizing $\mathbf{f}^\top \mathbf{D} \mathbf{f}$ successfully removes the scaling factor and avoids trivial solution $\mathbf{f} = \mathbf{0}$, it also tends to give large function values for data points in the dense area because they have large degree values d_i . Even for uniformly distributed data, Laplacian Eigenmaps still tends to drive the data to the two ends of the low-dimensional embedding, leaving a very low density area in the middle, see Fig. 1(c) for example.

In other words, Laplacian Eigenmaps works more similar to clustering than dimensionality reduction if we see the low-dimensional coordinates as class labels: it tries to recover the labels as determinate as possible, and avoids the central ambiguous area. Admittedly, this is in some cases desired and natural, such as in spectral clustering (actually, Normalized Cuts [11] is equivalent to Laplacian Eigenmaps plus discretization), it may not be an optimal choice for dimensionality reduction since it produces unnecessary distortion that especially affects data visualization.

In the next section, we aim to solve the distortion problem by both changing the loss function and the constraints.

3 The Proposed Method

In this section, we will first propose a new approach to construct the Laplacian matrix for dimensionality reduction, and then impose additional constraints to solve the distortion problem.

Table 1. A collection of the notations involving f , for better clarification

$f : \mathbb{R}^D \rightarrow \mathbb{R}$, and f^k	the mapping function, the superscript is used if we specify the function for the k -th dimension.
f_i , and f_i^k	the (k -th dimensional) function value for the i -th data $\mathbf{x}_i$.
$\mathbf{f}$, and $\mathbf{f}^k$	the (k -th dimensional) vector for all the data points.
$\mathbf{f}_i$, and $\mathbf{f}_i^k$	a $(K+1) \times 1$ local column vector whose first K elements are $[f_j]$ for all $\mathbf{x}_j \in \mathcal{N}_i$ and the last element is f_i . See Section 3.2.
$\tilde{\mathbf{f}}$	an $n(K+1) \times 1$ concatenated label vector, $\tilde{\mathbf{f}} = [\mathbf{f}_1^\top, \mathbf{f}_2^\top, \dots, \mathbf{f}_n^\top]^\top$.

3.1 The Local Linear Model and the Loss

To simplify the representation of symbols, we consider reducing the dimensionality to $d = 1$: let $\mathbf{f} = [f_1, f_2, \dots, f_n] \in \mathbb{R}^n$ be such a mapping that each f_i is the low-dimensional coordinate of $\mathbf{x}_i$. This can also be seen from a functional-view as $f_i = f(\mathbf{x}_i)$ where f is the mapping functional. In most real-world applications, we can assume f to be differentiable, *i.e.*, if there are enough data points, then in the small neighborhood $\mathcal{N}_i$ of $\mathbf{x}_i$ (we use K -nearest neighbor to define the neighborhood), we may use the first-order Taylor expansion to approximate the function as:

$$f_j \approx f_i + \nabla f(\mathbf{x}_i)^\top (\mathbf{x}_j - \mathbf{x}_i), \quad \forall \mathbf{x}_j \in \mathcal{N}_i. \quad (6)$$

In other words, if we find a mapping $\mathbf{f}$ that represents the local coordinates well, we expect the function values f_j in the neighborhood of $\mathbf{x}_i$ to behave linearly with respect to $\mathbf{x}_j - \mathbf{x}_i$ and f_i . Thus, a direct criterion to evaluate the fitness of a mapping $\mathbf{f}$ is to minimize the sum of the least-square loss in each neighborhood of the data points:

$$\mathcal{J}(\mathbf{f}) = \sum_{i=1}^n \sum_{j \in \mathcal{N}_i} [f_j - f_i - \nabla f(\mathbf{x}_i)^\top (\mathbf{x}_j - \mathbf{x}_i)]^2. \quad (7)$$

The thought of the loss function is inspired by the two most related works, namely Laplacian Eigenmaps [7] and Locally Linear Embedding (LLE) [4]. We discuss the relationship and difference here.

LLE. One may think that our derivation is similar to LLE at first glance, which assumes that each data point can be linearly reconstructed from its neighboring data points. In other words, for LLE, a reconstruction weight matrix W is calculated from the data so that $\sum_j W_{ij} \mathbf{x}_j$ gives the least-square estimation of $\mathbf{x}_i$ under the constraint $\sum_j W_{ij} = 1$. However, LLE focuses on the linear relationship from the neighborhood points and considers little about the direct relationship between the high and low coordinates, while in our model we explicitly adopt a function that generates the embedding from the high-dimensional coordinates. Thus the two methods are conceptually different. Further, when

the neighbor number K is larger than the dimensionality d , such reconstruction may not be unique. Even when unique solution exists, eliminating the suboptimal reconstruction weights may still lose information, as has been pointed out in [13]. Thus LLE may only be able to preserve comparatively weak information. Instead, by defining a local linear model $f_j = \mathbf{w}_i^T(\mathbf{x}_j - \mathbf{x}_i) + f_i$, we may preserve richer information than LLE: assume that the least-square estimations are precise, then any weight $\{W_{ij}\}$ that satisfies $\mathbf{x}_i = \sum_j W_{ij}\mathbf{x}_j$ will automatically satisfy $f_i = \sum_j W_{ij}f_j$ under the local linear model. Thus, we also expect our method to perform better than LLE in utilizing the local information.

Laplacian Eigenmaps. The Laplacian Eigenmaps method uses a slightly different criterion, which attempts to minimize the distance between nearby points with weight factor W_{ij} , *i.e.*, to minimize $\sum_j W_{ij}(f_i - f_j)^2$ in each local area. However, this introduces an additional parameter σ in the heat kernel that needs careful tuning, and may dilate the small holes in the intrinsic structure of the data as we have indicated in Section 2, which can further be seen experimentally in Fig. 3. We will experimentally show that our loss criterion empirically works better in Section 4. Theoretically, our loss criterion and the one of Laplacian Eigenmaps are essentially different: since the low-dimensional coordinates of nearby data points are essentially different (otherwise they will collapse to a constant value), we argue that it may not be optimal to impose penalty on any changes of the low-dimensional coordinates in the local area. Instead, we adopt a local linear model to allow reasonable changes, and only impose penalty if the low-dimensional embedding does not follow the model. This may be more reasonable than simply minimizing the distance of the neighbors.

3.2 Constructing the Laplacian Matrix

The above idea faces some difficulty to implement: there does not exist an explicit analytical representation of the gradient ∇f , since usually we only find the mapping value on the n data points. Thus, we denote the gradient vector at each data point $\mathbf{x}_i$ by an additional hidden parameter $\mathbf{w}_i \in \mathbb{R}^D$, and equivalently rewrite the loss function as:

$$\mathcal{J}(\mathbf{f}, \mathbf{w}_1, \dots, \mathbf{w}_n) = \sum_{i=1}^n \sum_{j \in \mathcal{N}_i} [f_j - f_i - \mathbf{w}_i^T(\mathbf{x}_j - \mathbf{x}_i)]^2. \quad (8)$$

Then, the remaining problem is how to estimate each $\mathbf{w}_i$. Since $\mathbf{w}_i$ is an ancillary parameter and we want to find $\mathbf{f}$ ultimately, we consider using $\mathbf{f}$ to analytically represent the gradient $\mathbf{w}_i$. That is, if given the function value $\mathbf{f}$, how to estimate the gradients $\{\mathbf{w}_i\}$ at each data point $\mathbf{x}_i$?

A straightforward thought may be to adopt the least-square estimation of the linear model at each neighborhood as $\mathbf{w}_i$. However, in most real-world applications, we often face a high dimensionality D (which may be significantly larger than the neighbor number K) and a comparatively small number of data points in the local area. This may cause the simple least-square estimation to be

unstable especially when there is noise in the data. Thus, we add an additional regularizer $\|\mathbf{w}_i\|^2$ in a SVM-like way to perform structural risk minimization. Mathematically, we aim to estimate $\mathbf{w}_i$ to be the optimum solution to the following regularized least-square problem:

$$\arg \min_{\mathbf{w}_i} [\lambda \|\mathbf{w}_i\|^2 + \sum_{j \in \mathcal{N}_i} (f_j - f_i - \mathbf{w}_i^\top (\mathbf{x}_j - \mathbf{x}_i))^2] , \quad (9)$$

where λ is a constant weight parameter. The thought is inspired by the supervised local learning algorithms [14] and semi-supervised Local Learning Regularization [15]. However, instead of interpreting the local model as a “classifier” and using the classification error as the loss function, we use the local model to do linear regression, which is more reasonable for dimensionality reduction, where the low-dimensional coordinates take continuous values.

Such estimation can further be written in a closed form. Define matrix X_i to be the matrix $[\mathbf{x}_j - \mathbf{x}_i]$ for all $\mathbf{x}_j \in \mathcal{N}_i$, local vector $\mathbf{f}_i$ to be a $(K+1) \times 1$ column vector whose first K elements are $[f_j]$ for all $\mathbf{x}_j \in \mathcal{N}_i$ and the last element is f_i , and a $K \times (K+1)$ ancillary matrix $H = [I_{K \times K}, -\mathbf{1}_{K \times 1}]$. Then, the optimization problem Eqn. (9) can be written as:

$$\arg \min_{\mathbf{w}_i} \lambda \|\mathbf{w}_i\|^2 + \|H\mathbf{f}_i - X_i^\top \mathbf{w}_i\|^2 . \quad (10)$$

Take the derivative with respect to $\mathbf{w}_i$ and set it to be zero, we get

$$\mathbf{w}_i = A_i \mathbf{f}_i, \quad \text{where } A_i = (\lambda I + X_i X_i^\top)^{-1} X_i H . \quad (11)$$

Substitute (11) into (8), the loss function $\mathcal{J}$ can then be written in a closed form of $\mathbf{f}$ as

$$\mathcal{J}(\mathbf{f}) = \sum_{i=1}^n \mathbf{f}_i^\top M_i \mathbf{f}_i , \quad (12)$$

where the matrix M_i is

$$M_i = H^\top (I - X_i^\top A_i)^\top (I - X_i^\top A_i) H . \quad (13)$$

Next, we define an $n(K+1) \times 1$ concatenated label vector $\tilde{\mathbf{f}} = [\mathbf{f}_1^\top, \mathbf{f}_2^\top, \dots, \mathbf{f}_n^\top]^\top$, an $n(K+1) \times n(K+1)$ concatenated block-diagonal matrix

$$M = \begin{bmatrix} M_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & M_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & M_n \end{bmatrix} , \quad (14)$$

and an $n(K+1) \times n$ selection matrix S whose elements take 0-1 values and there is only one 1 in each row so that $\tilde{\mathbf{f}} = S\mathbf{f}$. Then, the loss function is written as

$$\mathcal{J}(\mathbf{f}) = \mathbf{f}^\top \mathbf{L} \mathbf{f}, \quad \text{where } \mathbf{L} = S^\top M S . \quad (15)$$

It is not difficult to prove the following property:

Theorem 1. *The matrix $\mathbf{L}$ is a Laplacian matrix, i.e., it is positive semi-definite and satisfies $\mathbf{L}\mathbf{1} = \mathbf{0}$ where $\mathbf{1}$ and $\mathbf{0}$ are $n \times 1$ constant-value column vectors.*

Proof. See the Appendix.

3.3 Trivial Solution Elimination

Similar to Laplacian Eigenmaps, simply minimizing the loss function $\mathbf{f}^\top \mathbf{L} \mathbf{f}$ leads to a trivial solution $\mathbf{f} = \mathbf{0}$. Thus, we add scaling constraints to eliminate such trivial solutions. To eliminate the scaling factor, we make an isometric assumption that the norm of the local gradient vector at each data point $\mathbf{x}_i$ is 1, i.e., $\|\mathbf{w}_i\|^2 = 1$, inspired by the idea of PCA. This leads to n quadratic constraints $\mathbf{f}_i^\top A_i^\top A_i \mathbf{f}_i = 1$, where $i = 1, \dots, n$. As a relaxation and also for faster optimization, we may only place constraint on the sum of all the local gradients' norms as $\frac{1}{n} \sum_{i=1}^n \|\mathbf{w}_i\|^2 = 1$, which further leads to $\mathbf{f}^\top \mathbf{D} \mathbf{f} = 1$, where the scalar matrix $\mathbf{D}$ is calculated as

$$\mathbf{D} = S^\top \begin{bmatrix} A_1^\top A_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & A_2^\top A_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & A_n^\top A_n \end{bmatrix} S. \quad (16)$$

When the reduced dimensionality is larger than 1, we constraint each two low-dimensional mappings $\mathbf{f}^{k_1}$ and $\mathbf{f}^{k_2}$ to be such that their local gradient vectors at each point $\mathbf{x}_i$ to be orthogonal to each other, i.e., $\mathbf{w}_i^{k_1} \perp \mathbf{w}_i^{k_2}$, inspired by the idea of PCA. This leads to the constraints $(\mathbf{f}_i^{k_1})^\top A_i^\top A_i \mathbf{f}_i^{k_2} = 0$, where $i = 1, \dots, n$. Similar to the discussion above, we consider relaxing these constraints to the sum over the n points as $\sum_{i=1}^n (\mathbf{f}_i^{k_1})^\top A_i^\top A_i \mathbf{f}_i^{k_2} = 0$. It is easy to see that this leads to the constraint

$$(\mathbf{f}^{k_1})^\top \mathbf{D} \mathbf{f}^{k_2} = 0, \quad \text{for } k_1 \neq k_2. \quad (17)$$

Further, to remove the degree of translation freedom, we require the coordinates to be centered in the origin, i.e., to require $(\mathbf{f}^k)^\top \mathbf{1} = 0$.

Note that although looking similar, our constraint is different from Laplacian Eigenmaps, as the definitions of the matrix $\mathbf{D}$ in the two methods are different. It is worth pointing out the following theorem:

Theorem 2. *The matrix $\mathbf{D}$ is a Laplacian matrix.*

Proof. See the Appendix.

This leads to a different optimization approach, which we will observe in the next subsection.

3.4 Optimization

We use the following way to recover the d low-dimensional coordinates sequentially: for the i -th dimension ($1 \leq i \leq d$), we summarize the optimization task as follows:

$$\begin{aligned} \min_{\mathbf{f}^i} \quad & (\mathbf{f}^i)^\top \mathbf{L} \mathbf{f}^i \\ \text{s.t.} \quad & (\mathbf{f}^i)^\top \mathbf{1} = 0, \quad (\mathbf{f}^i)^\top \mathbf{D} \mathbf{f}^i = 1 \\ & (\mathbf{f}^i)^\top \mathbf{D} \mathbf{f}^j = 0, \quad \forall 1 \leq j < i. \end{aligned} \quad (18)$$

This is a non-convex problem because of the constraint $(\mathbf{f}^i)^\top \mathbf{D} \mathbf{f}^i = 1$. Note that generalized eigenvalue decomposition $\mathbf{L} \mathbf{f} = \lambda \mathbf{D} \mathbf{f}$ is not applicable to solve this problem. We explain this in detail. In Laplacian Eigenmaps, the constraint $(\mathbf{f}^i)^\top \mathbf{1} = 0$ will be automatically satisfied by eliminating the eigenvector $\mathbf{1}$ corresponding to generalized eigenvalue 0 (note that the degree matrix in Laplacian Eigenmaps is positive definite). However, in our method, the matrix $\mathbf{D}$ is defined differently and is also a Laplacian matrix. This leads to $\mathbf{L} \mathbf{1} \equiv \mathbf{D} \mathbf{1} \equiv 0$, which means that $\mathbf{1}$ is a generalized eigenvector corresponding to any value in $\mathbb{R}$. Most GEVD solvers may suffer from severe numerical problems performing such tasks.

Instead, to get a stable and satisfying solution, we directly solve the problem using Constrained Concave Convex Procedure (CCCP) [16]. CCCP works in an iterative way: at each iteration, the 1-st order Taylor expansion is used to approximate the non-convex constraints, and the problem is thus approximated by a convex optimization subproblem. The optimum solution to the subproblem is then used as the initial value of the next iteration. This procedure is repeated until convergence, which has been theoretically guaranteed in [16]. Specifically, we rewrite the constraint $(\mathbf{f}^i)^\top \mathbf{D} \mathbf{f}^i = 1$ to a convex one $(\mathbf{f}^i)^\top \mathbf{D} \mathbf{f}^i \leq 1$ and a concave one $(\mathbf{f}^i)^\top \mathbf{D} \mathbf{f}^i \geq 1$. At each iteration, assume that the initial value is $\mathbf{f}^{i,\text{old}}$, the concave constraint is approximated as

$$2(\mathbf{f}^i)^\top \mathbf{D} \mathbf{f}^{i,\text{old}} - (\mathbf{f}^{i,\text{old}})^\top \mathbf{D} \mathbf{f}^{i,\text{old}} \geq 1. \quad (19)$$

Thus the subproblem given initial value $\mathbf{f}^{i,\text{old}}$ is

$$\begin{aligned} \min_{\mathbf{f}^i} \quad & (\mathbf{f}^i)^\top \mathbf{L} \mathbf{f}^i \\ \text{s.t.} \quad & (\mathbf{f}^i)^\top \mathbf{1} = 0, \quad (\mathbf{f}^i)^\top \mathbf{D} \mathbf{f}^i \leq 1 \\ & 2(\mathbf{f}^i)^\top \mathbf{D} \mathbf{f}^{i,\text{old}} - (\mathbf{f}^{i,\text{old}})^\top \mathbf{D} \mathbf{f}^{i,\text{old}} \geq 1 \\ & (\mathbf{f}^i)^\top \mathbf{D} \mathbf{f}^j = 0, \quad \forall 1 \leq j < i. \end{aligned} \quad (20)$$

It is a standard *quadratic programming* (QP) problem that can be solved efficiently.

Further, it is worth indicating that although CCCP only guarantees local optimum, in our experiments we have reached identical solutions from multiple runs with different initial values, implying that the optimization procedure may already be stable and good enough for real-world applications. Still, a theoretical analysis on the optimization method is one of our further interests. One

shortcoming using CCCP is that it may require some time to converge. In our experiments, the method runs about half a minute for a 2000-point data set using Matlab code and Mosek to carry out the optimization on a P4 2G computer, which is a time a bit longer than that spent by LE.

3.5 Extension for Out-of-sample Data

To predict the low-dimensional coordinates of a new data point $\mathbf{x}$ that does not belong to the original data set $\mathcal{X}$, we propose the following induction approach: first find its neighborhood $\mathcal{N}_{\mathbf{x}}$ in the original data set, and then find the low-dimensional embedding $f(\mathbf{x})$ that simultaneously fits the local linear models on all its neighbor points well by minimizing the following criterion:

$$f(\mathbf{x}) = \arg \min_{f_{\mathbf{x}}} \sum_{\mathbf{x}_i \in \mathcal{N}_{\mathbf{x}}} [(f_{\mathbf{x}} - f_i) - \mathbf{w}_i^{\top}(\mathbf{x} - \mathbf{x}_i)]^2 . \quad (21)$$

Take the derivative with respect to $f_{\mathbf{x}}$ and set it to be zero, the result can be written in a closed form as

$$f(\mathbf{x}) = \frac{1}{K} \sum_{\mathbf{x}_i \in \mathcal{N}_{\mathbf{x}}} [f_i + \mathbf{w}_i^{\top}(\mathbf{x} - \mathbf{x}_i)] , \quad (22)$$

where K is the number of neighbor points.

4 Experiments

In this section, we perform dimensionality reduction for data visualization on toy and real-world data sets. For all the experiments, the parameter σ in Laplacian Eigenmaps is chosen via a line search from $\{1/16, 1/4, 1, 4, 16, \infty\} \times \sigma_0$ where σ_0 is the squared root of the mean pairwise distance between data points; the parameter λ in our method is chosen from $\{0.01, 0.1, 1, 10\}$.

4.1 Toy Data Sets

We generate a toy *clover* data set with two essential dimensions generated by $\{x = (1+0.3 \sin(3t)) \cos(t), y = (1+0.3 \sin(3t)) \sin(t)\}$ where t takes 80 uniformly spaced values from $[0, 2\pi)$. The data is then mapped to a 20 dimensional space with a randomly generated linear transform. We test our method against two most related algorithms namely Laplacian Eigenmaps and LLE² to recover the true 2-dimensional embedding using neighbor number $K = 10$. The result of Laplacian Eigenmaps has been shown in Fig. 1(b). The result of LLE and our method are shown in Fig. 2. Although all three methods keep the neighboring information well, our method clearly preserves better shape information, while the other two methods show poorer reconstructions.

Next we experiment on four kinds of Swiss roll data sets that have appeared in the previous literature. *Swiss Roll* is the classical Swiss roll data set;

² The LLE code used here is from <http://www.cs.toronto.edu/~roweis/lle/code.html>.

Fig. 2. The result of LLE and our method on the toy clover data. Figure (c) shows the local gradient vectors estimated by our method.

Fig. 3. Experimental results on the Swiss roll data sets. In the first row, the first figure shows a 3-dimensional roll structure and the other two shows the true 2-dimensional coordinates of the corresponding data. The 2nd to the 4th rows are the results of LLE, Laplacian Eigenmaps, and our method respectively.

Swiss Roll-H cuts a squared hole on the intrinsic data structure; *Swiss Roll-N* has a narrow width and a comparatively long roll; *Swiss Roll-II* embeds a π -shaped set of data into the 3-dimensional space. The data and results are shown in Fig. 3. We can observe visually that our method better preserves the

low-dimensional structure under all conditions, while LLE may give globally distorted recovery and Laplacian Eigenmaps tend to “squeeze” the data into lines of high-density areas as we have analyzed in the previous part of the paper.

4.2 Quantitative Comparison

For the two data sets, since we have the true low-dimensional coordinates of the data, we are able to calculate the *procrustes measure* (PM) [17] between the true coordinates and the recovered ones for each method for quantitative comparison. Procrustes measure between two sets of points determines a linear transformation (including translation, reflection, orthogonal rotation, and scaling) and uses the sum of squared errors as a goodness-of-fit measure, which in our case shows the accuracy of the dimensionality reduction algorithms. A smaller PM value indicates more accurate recovery. We also calculate the local procrustes measure, which is the mean of the procrustes measures in each local neighborhood, to compare the accuracy of the dimensionality reduction algorithms in the local area. PCA is adopted as a baseline. Table 2 shows the results of different methods on the toy data sets (if one algorithm needs parameter tuning, the best result is reported here). The result verifies the superiority of our method to preserve the intrinsic data structure both locally and globally.

Table 2. The procrustes measure (PM) and local procrustes measure (LPM) on the toy data sets. LE denotes Laplacian Eigenmaps and DFDR is our distortion-free dimensionality reduction method.

Data set	PM				LPM			
	PCA	LLE	LE	DFDR	PCA	LLE	LE	DFDR
clover	0.1464	0.0303	0.0637	1.006×10^{-4}	0.0433	0.0167	0.0396	2.188×10^{-4}
Swiss Roll	0.6891	0.1973	0.2090	0.0032	0.1078	0.4525	0.2726	0.0051
Swiss Roll-H	0.6791	0.1232	0.1678	0.0047	0.1136	0.1022	0.3519	0.0077
Swiss Roll-N	0.8988	0.4738	0.5116	0.0016	0.4255	0.2917	0.4434	0.0187
Swiss Roll-II	0.6634	0.2066	0.2550	0.0054	0.1201	0.1316	0.3316	0.0068

4.3 Real-World Data

In this part, we perform dimensionality reduction on real-world data sets. We randomly select 2000 digital TWOs from the MNIST database [18] for dimensionality reduction and 1000 for out-of-sample test, and show the results in Fig. 4. For computation speed consideration, a precedent PCA is adopted to save 95% of the energy before NLDR is applied. It can be seen that both results reveal the intrinsic structure of the low-dimensional embedding. Next, we adopt the sequential smiling face data from [19] to test our method. The one-dimensional embedding and the original images are shown in Fig. 5. Our method recovers the low-dimensional coordinates well and is consistent with the original data. This can be seen visually: in the recovered coordinates, a larger gap between

Fig. 4. The result of dimensionality reduction on the digital TWO images

Fig. 5. Recovered coordinates from a sequential smiling face images. The images are borrowed from [19]. The numbers in (b) and (c) shows the Euclidean distance between related images.

data points such as in the rectangle marked “b” and “c” corresponds to a larger distance between the corresponding images. It is comparatively difficult for us to provide a quantitative evaluation (and comparison against other NLDR algorithms) like we did on the toy data, as we do not know what low-dimensional coordinates are “optimal”. However, the experiments have revealed that our method does obtain good dimensionality reduction results visually and preserves the local information well.

5 Conclusion

Dimensionality reduction has been an important issue in the machine learning field to reveal the low-dimensional structure of the data. The main contribution of our paper lies in two aspects: (1) we discussed the distortion problem of the classical Laplacian Eigenmaps method and proposed an alternative way to construct the Laplacian matrix via a local linear model; (2) we proposed a new way to reduce the dimensionality to obtain a better result both visually and numerically, and avoid the distortion problem successfully. In the future, we will explore the theoretical nature of the proposed method as well as developing more efficient algorithms to solve the optimization task.

Acknowledgement

This work is supported by National 863 Project (No. 2006AA01Z121) and NSFC (Grant No. 60675009) of China.

References

1. Jolliffe, I.: *Principal Component Analysis*. Springer, Heidelberg (2002)
2. Cox, T., Cox, M.: *Multidimensional Scaling*. Chapman and Hall, Boca Raton (2001)
3. Tenenbaum, J., Silva, V., Langford, J.: A Global Geometric Framework for Non-linear Dimensionality Reduction. *Science* 290(5500), 2319–2323 (2000)
4. Roweis, S., Saul, L.: Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science* 290(5500), 2323–2326 (2000)
5. Zhang, Z., Zha, H.: Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. *SIAM J. Sci. Comp.* 26(1), 313–338 (2005)
6. Weinberger, K., Saul, L.: Unsupervised Learning of Image Manifolds by Semidefinite Programming. *Intl. J. Computer Vision* 70(1), 77–90 (2006)
7. Belkin, M., Niyogi, P.: Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. *Neural Computation* 15(6), 1373–1396 (2003)
8. He, X., Niyogi, P.: Locality preserving projections. *Advances in Neural Information Processing Systems* (2003)
9. Belkin, M., Niyogi, P.: Towards a theoretical foundation for laplacian-based manifold methods. In: Auer, P., Meir, R. (eds.) *COLT 2005. LNCS (LNAI)*, vol. 3559, pp. 486–500. Springer, Heidelberg (2005)
10. Ding, C., Simon, H., Jin, R., Li, T.: A learning framework using Green's function and kernel regularization with application to recommender system. In: *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 260–269 (2007)
11. Shi, J., Malik, J.: Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22(8), 888–905 (2000)
12. Hagen, L., Kahng, A.: New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 11(9), 1074–1085 (1992)
13. Zhang, Z., Wang, J.: MLLE: Modified Locally Linear Embedding Using Multiple Weights. *Advances in Neural Information Processing Systems* (2007)

14. Bottou, L., Vapnik, V.: Local learning algorithms. *Neural Computation* 4(6), 888–900 (1992)
15. Wu, M., Schölkopf, B.: Transductive classification via local learning regularization. In: *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 624–631 (2007)
16. Smola, A., Vishwanathan, S., Hofmann, T.: Kernel methods for missing variables. In: *Proc. International Workshop on Artificial Intelligence and Statistics* (2005)
17. Seber, G.: *Multivariate Observations*. Wiley, Chichester (1984)
18. LeCun, Y.: The MNIST database of handwritten digits. NEC Research Institute, <http://yann.lecun.com/exdb/mnist/index.html>
19. Gashler, M., Ventura, D., Martinez, T.: Iterative non-linear dimensionality reduction with manifold sculpting. In: *Advances in Neural Information Processing Systems* 20, pp. 513–520 (2008)

Appendix: Proof to the Theorems

Proof to Theorem 1: To prove that $\mathbf{L}$ is positive semi-definite, we first write $\mathbf{L}$ as $\mathbf{L} = P^\top P$, where we define the ancillary matrix P as

$$P = \begin{bmatrix} (I - X_1^\top A_1)H & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & (I - X_2^\top A_2)H & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & (I - X_n^\top A_n)H \end{bmatrix} S, \quad (23)$$

where the selection matrix S is defined in Section 3.2. For any arbitrary vector $\mathbf{f}$, we have

$$\mathbf{f}^\top \mathbf{L} \mathbf{f} = \|P \mathbf{f}\|^2 \geq 0, \quad (24)$$

thus $\mathbf{L}$ is positive semi-definite.

To prove $\mathbf{L}\mathbf{1} = \mathbf{0}$, notice that the row sum of S is 1, *i.e.*, $S\mathbf{1} = \mathbf{1}$ (with a slight abuse of the notation, we use $\mathbf{1}$ to denote all the column vectors with proper dimensions), so we have

$$\mathbf{L}\mathbf{1} = S^\top M S \mathbf{1} = S^\top M \mathbf{1}. \quad (25)$$

Thus we can prove $\mathbf{L}\mathbf{1} = \mathbf{0}$ if we can prove $M\mathbf{1} = \mathbf{0}$. This is further equivalent to proving $M_i \mathbf{1} = \mathbf{0}$, $\forall i = 1, \dots, n$. Define the symmetric matrix $B_i = (I - X_i^\top A_i)^\top (I - X_i^\top A_i)$, from Eqn. 13, we get

$$\begin{aligned} M_i \mathbf{1} &= H^\top (I - X_i^\top A_i)^\top (I - X_i^\top A_i) H \mathbf{1} \\ &= \begin{bmatrix} I \\ -\mathbf{1}^\top \end{bmatrix} B_i \begin{bmatrix} I - \mathbf{1} \end{bmatrix} \mathbf{1} \\ &= \begin{bmatrix} B_i & -B_i \mathbf{1} \\ -\mathbf{1}^\top B_i & \mathbf{1}^\top B_i \mathbf{1} \end{bmatrix} \mathbf{1} \\ &= \mathbf{0}. \end{aligned} \quad (26)$$

Thus, we have

$$\mathbf{L}\mathbf{1} = S^\top M \mathbf{1} = S^\top \mathbf{0} = \mathbf{0}, \quad (27)$$

and the theorem is proved.

Proof to Theorem 2: the proof is similar to the proof to Theorem 1. To prove that D is positive semi-definite, we use $D = \tilde{P}^\top \tilde{P}$ where P is defined as:

$$P = \begin{bmatrix} A_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & A_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & A_n \end{bmatrix} S . \quad (28)$$

To prove $D\mathbf{1} = \mathbf{0}$, it is equivalent to prove $A_i^\top A_i \mathbf{1} = \mathbf{0}$, $\forall i = 1, \dots, n$. To see this, we have

$$\begin{aligned} A_i^\top A_i \mathbf{1} &= H^\top [(\lambda I + X_i X_i^\top)^{-1} X_i]^\top (\lambda I + X_i X_i^\top)^{-1} X_i H \\ &= \begin{bmatrix} I \\ -\mathbf{1}^\top \end{bmatrix} \hat{B}_i \begin{bmatrix} I & -\mathbf{1} \end{bmatrix} \mathbf{1} \\ &= \begin{bmatrix} \hat{B}_i & -\hat{B}_i \mathbf{1} \\ -\mathbf{1}^\top \hat{B}_i & \mathbf{1}^\top \hat{B}_i \mathbf{1} \end{bmatrix} \mathbf{1} \\ &= \mathbf{0} , \end{aligned} \quad (29)$$

where we define $\hat{B}_i$ as $\hat{B}_i = [(\lambda I + X_i X_i^\top)^{-1} X_i]^\top (\lambda I + X_i X_i^\top)^{-1} X_i$. Thus, the theorem is proved.