Lin T, Liu Y, Wang B *et al.* Nonlinear dimensionality reduction by local orthogonality preserving alignment. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 31(3): 512–524 May 2016. DOI 10.1007/s11390-016-1644-4

Nonlinear Dimensionality Reduction by Local Orthogonality Preserving Alignment

Tong Lin, Yao Liu, Bo Wang, Li-Wei Wang, *Senior Member, CCF, Member, ACM*, and Hong-Bin Zha, *Member, CCF, ACM, IEEE*

The Key Laboratory of Machine Perception (Ministry of Education), School of Electrical Engineering and Computer Sciences, Peking University, Beijing 100871, China

E-mail: {lintong, liuyao}@pku.edu.cn; wangbo1204@gmail.com; {wanglw, zha}@cis.pku.edu.cn

Received December 1, 2015; revised March 31, 2016.

Abstract We present a new manifold learning algorithm called Local Orthogonality Preserving Alignment (LOPA). Our algorithm is inspired by the Local Tangent Space Alignment (LTSA) method that aims to align multiple local neighborhoods into a global coordinate system using affine transformations. However, LTSA often fails to preserve original geometric quantities such as distances and angles. Although an iterative alignment procedure for preserving orthogonality was suggested by the authors of LTSA, neither the corresponding initialization nor the experiments were given. Procrustes Subspaces Alignment (PSA) implements the orthogonality preserving idea by estimating each rotation transformation separately with simulated annealing. However, the optimization in PSA is complicated and multiple separated local rotations may produce globally contradictive results. To address these difficulties, we first use the pseudo-inverse trick of LTSA to represent each local orthogonal transformation with the unified global coordinates. Second the orthogonality constraints are relaxed to be an instance of semi-definite programming (SDP). Finally a two-step iterative procedure is employed to further reduce the errors in orthogonal constraints. Extensive experiments show that LOPA can faithfully preserve distances, angles, inner products, and neighborhoods of the original datasets. In comparison, the embedding performance of LOPA is better than that of PSA and comparable to that of state-of-the-art algorithms like MVU and MVE, while the runtime of LOPA is significantly faster than that of PSA, MVU and MVE.

Keywords manifold learning, dimensionality reduction, semi-definite programming, Procrustes measure

1 Introduction

Manifold learning is a large class of nonlinear dimensionality reduction methods operated in an unsupervised manner, with each method attempting to preserve a particular geometric quantity such as distances, angles, proximity, or local patches. Since the two pioneering work published on *Science* in 2000, Isomap[1] and LLE[2-3], manifold learning[4] has been a significant topic in data visualization and pattern classification. Today the huge amount of data coming from imaging devices, bioinformatics, and financial applications are usually high-dimensional; thus there is an imperative need to overcome the "curse of dimensionality"[5]. A direct solution is the dimensionality reduction approach that transforms the high-dimensional data into a low-dimensional embedding space. However, traditional methods like PCA and MDS fail to discover nonlinear or curved structures of the input data. In contrast, manifold learning methods are suitable for unfolding the nonlinear structures into a flat low-dimensional embedding space. Therefore, these methods have found a wide variety of applications, for instance, microarray gene expression, 3D body pose recovery, face recognition and facial expression transferring (see [6] for some recent applications based on manifold alignment).

According to the methodology in [7], existing mani-

Regular Paper

Special Section of CVM 2016

This work was supported by the National Basic Research 973 Program of China under Grant No. 2011CB302202, the National Natural Science Foundation of China under Grant Nos. 61375051 and 61075119, and the Seeding Grant for Medicine and Information Sciences of Peking University of China under Grant No. 2014-MI-21.

fold learning methods can be roughly divided into three categories: 1) distance-preserving methods, including Isomap[1], MVU[8-9], MVE[10], and RML[11]; 2) angle-preserving methods, e.g., conformal eigenmaps[12]; and 3) proximity-preserving methods, such as LLE[2-3], HLLE[13], Laplacian Eigenmaps (LE)[14], LTSA[15], and NPPE[16], which align local weights or neighborhood for each data point into a global coordinates space. Due to recent advancement, here we point out that there exists the fourth category: 4) patch-preserving methods, such as LMDS[17], and MLE[18], which align each linear patch of moderate size with other patches in order to construct the global representation. In addition, several special methods occurred to be seemingly excluded by the four main categories, such as manifold sculpting[19] and NeRV[20].

Most previous manifold learning methods focus on one particular perspective in order to preserve a single geometric quantity. In this way, for instance, a proximity-preserving method often performs poorly when viewed from other perspectives such as maintaining distances and angles. A basic question was addressed by [21]: how do we define a *faithful* embedding that preserves the local structure of neighborhoods on the manifold? In other words, can we find some fundamental clues to handle distances, angles, and neighborhoods in a comprehensive way? The proposed answer was the Procrustes measure, which computes the distance between two configurations of points after one of the configuration is rotated and translated to best match the other. As the translation vector can be omitted by centering each point set, the computation of fitting errors in Procrustes measure boils down to finding the best rotation (orthogonal) matrix. Then two algorithms, greedy Procrustes (GP) and Procrustes subspaces alignment (PSA), were developed to minimize the suggested measure. GP is a progressive method that relies on the selection of a basis point and the embeddings produced by GP may not maintain the global structure of the input data (e.g., the cylinder data of Fig.3 in [21]). On the other hand, PSA performs the global embedding by finding each local orthogonal transformation separately with complicated simulated annealing (SA) and then aligning multiple local PCA subspaces together. However, there is a risk that these local orthogonal transformations may produce an incompatible global embedding since each orthogonal transformation is estimated separately.

We agree that the Procrustes measure[21] is one reasonable clue to be preserved in manifold learning. To circumvent the difficulties in PSA, in this paper we propose a new algorithm called Local Orthogonality Preserving Alignment (LOPA). The main contributions are highlighted as follows.

- With the pesudo-inverse trick proposed in [15], we render the Procrustes measure minimization problem into an orthogonality constraint problem with respect to an unknown global embedding. This sidesteps the problem of handling multiple local orthogonal transformations occurred in the PSA method[21].
- The orthogonality constraint problem is further relaxed into an easier trace constraint problem, which can be efficiently solved by any semidefinite programming (SDP) tool. The relaxation model is a loose approximation to the original Procrustes problem, but is acceptable in practice. The original problem imposes the exact orthogonal constraints on each local transformation, which is too hard to satisfy. Thus our optimization is easier than the complicated simulated annealing used in PSA[21].
- We report experimental results on synthetic and real-world datasets, supporting our claims of better preserving geometrical properties like distances and angles, and demonstrating the faster speed of LOPA than the state-of-the-art methods such as PSA, MVU, and MVE.

The rest of the paper is organized as follows. We first discuss some criteria in manifold learning and describe our models in Section 2. Section 3 is devoted to numerically solving the proposed optimization problem, and experimental results are presented in Section 4. Section 5 concludes this paper.

2 Criteria and Models

Generally there are two ways to handle multiple geometric quantities in a comprehensive manner for manifold learning. The first one is to preserve the Riemannian metrics, a fundamental notion in Riemannian geometry[22], which determine inner products on tangent spaces at every point. The work in [23] was the first attempt to use Riemannian metrics as a criterion in manifold learning. An algorithm was developed to augment the output of any embedding method with Riemannian metrics estimated by the Laplace-Beltrami operator, which can be regarded as a post-processing to the results obtained by any existing manifold learning method. However, this work did not provide any new manifold learning algorithm to preserve Riemannian metrics. Here we present a simple model to directly preserve inner products in each neighborhood, and show the inherent difficulties in its optimization.

Given a dataset $\boldsymbol{X} = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_n) \in \mathbb{R}^{m\times n}$ with each data point $\boldsymbol{x}_i$ is an m-dimensional column vector, the goal of dimensionality reduction is to transform $\boldsymbol{X}$ to $\boldsymbol{Y} = [\boldsymbol{y}_1, \dots, \boldsymbol{y}_n] \in \mathbb{R}^{d\times n}$ ($d \ll m$), where each $\boldsymbol{y}_i$ is the low-dimensional representation of $\boldsymbol{x}_i$. For each data point $\boldsymbol{x}_i$, we denote $\boldsymbol{X}_i = [\boldsymbol{x}_{i_1}, \dots, \boldsymbol{x}_{i_k}]$ as its k nearest neighbors (including itself by setting $\boldsymbol{x}_{i_1} = \boldsymbol{x}_i$), and the neighborhood indices are represented as $\Omega_i = [i_1, \dots, i_k]$. Here k is a predefined parameter. A direct model to preserve inner products in each neighborhood can be formulated as the following minimization problem of finding the optimal $\boldsymbol{Y}$:

$$\min_{\boldsymbol{Y}} \sum_{i=1}^{n} \sum_{j,\,l} \left(\langle \boldsymbol{y}_j - \boldsymbol{y}_i, \boldsymbol{y}_l - \boldsymbol{y}_i\rangle - \langle \boldsymbol{x}_j - \boldsymbol{x}_i, \boldsymbol{x}_l - \boldsymbol{x}_i\rangle\right)^2$$
$$\text{s.t.}\ \ j, l \in \Omega_i,\ \ j, l \neq i. \qquad (1)$$

Here $\langle\cdot,\cdot\rangle$ denotes the inner product of two vectors. A similar formula occurred in MVU[8-9], but an equivalent formulation of local isometry, i.e., preserving pairwise distances, is used in the final MVU implementation. We show that the minimization of (1) leads to a standard least squares (LS) problem:

$$\begin{aligned}&\sum_{i=1}^{n} \|\boldsymbol{H}_k^{\mathrm{T}} \boldsymbol{S}_i^{\mathrm{T}} \boldsymbol{Y}^{\mathrm{T}} \boldsymbol{Y} \boldsymbol{S}_i \boldsymbol{H}_k - \boldsymbol{H}_k^{\mathrm{T}} \boldsymbol{S}_i^{\mathrm{T}} \boldsymbol{X}^{\mathrm{T}} \boldsymbol{X} \boldsymbol{S}_i \boldsymbol{H}_k\|_F^2\\ &= \sum_{i=1}^{n} \|\boldsymbol{Q}_i^{\mathrm{T}} \boldsymbol{Z} \boldsymbol{Q}_i - \boldsymbol{W}_i\|_F^2\\ &= \sum_{i=1}^{n} \|\boldsymbol{A}_i \boldsymbol{z} - \boldsymbol{w}_i\|^2\\ &= \|\boldsymbol{A}\boldsymbol{z} - \boldsymbol{w}\|^2,\end{aligned}$$

where $\boldsymbol{H}_k \doteq \boldsymbol{I} - \boldsymbol{e}\boldsymbol{e}^{\mathrm{T}}/k$ is a centering matrix of size k-by-k, $\boldsymbol{I}$ (or $\boldsymbol{I}_k$) is an identity matrix (of size k-by-k), $\boldsymbol{e}$ is a column vector of all 1s (in a proper dimension), and $\boldsymbol{S}_i$ is a 0-1 selection matrix for choosing the neighborhood $\boldsymbol{X}_i$. Note that $\boldsymbol{Q}_i \doteq \boldsymbol{S}_i \boldsymbol{H}_k$, $\boldsymbol{Z} \doteq \boldsymbol{Y}^{\mathrm{T}}\boldsymbol{Y}$, $\boldsymbol{W}_i \doteq \boldsymbol{H}_k^{\mathrm{T}} \boldsymbol{S}_i^{\mathrm{T}} \boldsymbol{X}^{\mathrm{T}} \boldsymbol{X} \boldsymbol{S}_i \boldsymbol{H}_k$ can be computed beforehand. We define the operator $\boldsymbol{vec}(\boldsymbol{A})$ that stacks all the columns of $\boldsymbol{A}$ and outputs a long column vector. Thereby in the above equations, $\boldsymbol{z} \doteq \boldsymbol{vec}(\boldsymbol{Z})$, $\boldsymbol{w}_i \doteq \boldsymbol{vec}(\boldsymbol{W}_i)$, and $\boldsymbol{A}$ and $\boldsymbol{w}$ are formed by stacking all $\boldsymbol{A}_i$ and $\boldsymbol{w}_i$ together respectively. The notation $\|\cdot\|_F$ denotes the matrix Frobenius norm. Note that we use the well-known equality $\boldsymbol{vec}(\boldsymbol{A}\boldsymbol{X}\boldsymbol{B}) = (\boldsymbol{B}^{\mathrm{T}} \otimes \boldsymbol{A})\boldsymbol{vec}(\boldsymbol{X})$ in the step of obtaining $\boldsymbol{z}$ from $\boldsymbol{Z}$, where $\otimes$ denotes the Kronecker product of two matrices.

However, this LS problem is rank-deficient in solving an n^2-dimensional vector $\boldsymbol{z}$ with only nk^2 equations, thus having an infinite number of solutions in most cases such that $k^2 \ll n$. As $\mathrm{rank}(\boldsymbol{Y}) = d$ for common cases when $d < n$, the result $\boldsymbol{Y}$ obtained by eigendecomposition of $\boldsymbol{Z} = \boldsymbol{Y}^{\mathrm{T}}\boldsymbol{Y}$ is usually a poor embedding. One remedy is to explicitly incorporate the rank constraint of $\boldsymbol{Y}$ into the LS problem. But the fixed rank or low rank LS problem poses great challenges for finding reasonable embedding for high-dimensional datasets. Furthermore, the huge sizes of $\boldsymbol{A} \in \mathbb{R}^{nk^2\times n^2}$ and $\boldsymbol{w} \in \mathbb{R}^{nk^2\times 1}$ can be problematic in storage even for a small dataset. For instance, the number of matrix entries in $\boldsymbol{A}$ is 64 billion if $n = 1\,024$ and $k = 8$.

Notice that the complexity of above inner product model (1) essentially comes from the quadratic term $\boldsymbol{Y}^{\mathrm{T}}\boldsymbol{Y} \in \mathbb{R}^{n\times n}$ in the inner product representation. In order to reduce the complexity, we resort to an alternative way based on local alignments preserving orthogonality (or isometry). The Local Tangent Space Alignment (LTSA) method[15] provides an elegant framework for neighborhood alignments:

$$\min_{\boldsymbol{Y},\{\boldsymbol{L}_i\}} \sum_{i=1}^{n} \|\boldsymbol{Y}\boldsymbol{S}_i\boldsymbol{H}_k - \boldsymbol{L}_i\boldsymbol{\Theta}_i\|_F^2, \qquad (2)$$

where $\boldsymbol{\Theta}_i \in \mathbb{R}^{d\times k}$ is the d-dimensional PCA coordinate representation for $\boldsymbol{X}_i$, and $\boldsymbol{L}_i \in \mathbb{R}^{d\times d}$ is a local affine transformation. The cost function of (2) is then the first order about $\boldsymbol{Y}$ (rather than the second order of $\boldsymbol{Y}^{\mathrm{T}}\boldsymbol{Y}$ in the above inner product model). Then using a pseudo-inverse trick, for each fixed $\boldsymbol{Y}$, the optimal affine transformation can be represented as $\boldsymbol{L}_i = \boldsymbol{Y}\boldsymbol{S}_i\boldsymbol{H}_k\boldsymbol{\Theta}_i^{\dagger}$ where $\boldsymbol{\Theta}_i^{\dagger}$ is the Moore-Penrose generalized inverse of $\boldsymbol{\Theta}_i$. Hence the cost function of (2) can be formulated as a trace $tr(\boldsymbol{Y}\boldsymbol{B}\boldsymbol{Y}^{\mathrm{T}})$, where $\boldsymbol{B} \doteq \sum_{i=1}^{N} \boldsymbol{S}_i\boldsymbol{H}_k(\boldsymbol{I}-\boldsymbol{\Theta}_i^{\dagger}\boldsymbol{\Theta}_i)(\boldsymbol{I}-\boldsymbol{\Theta}_i^{\dagger}\boldsymbol{\Theta}_i)^{\mathrm{T}}\boldsymbol{H}_k^{\mathrm{T}}\boldsymbol{S}_i^{\mathrm{T}} \in \mathbb{R}^{n\times n}$ (see the detailed derivations in [15]). Finally, by imposing the "bogus" unit covariance constraint $\boldsymbol{Y}\boldsymbol{Y}^{\mathrm{T}} = \boldsymbol{I}_d$, the LTSA algorithm obtains the optimal $\boldsymbol{Y}$ given by the eigenvectors corresponding to the d smallest positive eigenvalues of $\boldsymbol{B}$. However, general linear transformations can *not* preserve local geometric quantities such as distances and angles.

A straightforward extension is to restrict the linear transformations $\boldsymbol{L}_i$ in the set of orthogonal matrices, leading to our LOPA model:

$$\min_{\boldsymbol{Y},\{\boldsymbol{L}_i\}} \sum_{i=1}^{n} \|\boldsymbol{Y}\boldsymbol{S}_i\boldsymbol{H}_k - \boldsymbol{L}_i\boldsymbol{\Theta}_i\|_F^2$$
$$\text{s.t.}\ \ \boldsymbol{L}_i\boldsymbol{L}_i^{\mathrm{T}} = \boldsymbol{I}_d,\quad i = 1, \dots, n. \qquad (3)$$

The LOPA model (3) is similar to PSA, except that PSA directly aligns the high-dimensional input data $\boldsymbol{X}_i$

without the use of PCA projection. Again using the pseudo-inverse trick to represent $\boldsymbol{L}_i$, the LOPA model can be rewritten as

$$\min_{\boldsymbol{Y}} tr(\boldsymbol{Y}\boldsymbol{B}\boldsymbol{Y}^{\mathrm{T}})$$
$$\text{s.t. } \boldsymbol{Y}\boldsymbol{C}_i\boldsymbol{Y}^{\mathrm{T}} = \boldsymbol{I}_d,\ i = 1, \ldots, n, \quad (4)$$

where $\boldsymbol{C}_i \doteq \boldsymbol{G}_i\boldsymbol{G}_i^{\mathrm{T}} \in \mathbb{R}^{n\times n}$ with $\boldsymbol{G}_i \doteq \boldsymbol{S}_i\boldsymbol{H}_k\boldsymbol{\Theta}_i^{\dagger} \in \mathbb{R}^{n\times d}$. An earlier work of the ONPP[24] method shares a similar idea:

$$\min_{\boldsymbol{Y}} tr(\boldsymbol{Y}\boldsymbol{M}\boldsymbol{Y}^{\mathrm{T}})$$
$$\text{s.t. } \boldsymbol{Y} = \boldsymbol{V}^{\mathrm{T}}\boldsymbol{X}, \boldsymbol{V}^{\mathrm{T}}\boldsymbol{V} = \boldsymbol{I}_d,$$

where $\boldsymbol{M} \in \mathbb{R}^{n\times n}$ is a known matrix, and the embedding $\boldsymbol{Y}$ is obtained by an orthogonal transformation of $\boldsymbol{X}$. However, ONPP has only one orthogonality constraint and essentially is a linear projection.

3 Optimizations

3.1 Orthogonality Constraint Problems

The LOPA model (4) is a minimization problem with multiple matrix orthogonality constraints. Minimization with orthogonality constraints[25] plays an important role in many applications of science and engineering, such as polynomial optimization, combinatorial optimization, eigenvalue problems, sparse PCA, p-harmonic flows, 1-bit compressive sensing, and matrix rank minimization (see [25] for descriptions of some recent applications). Three types of problems are considered in [25]:

$$\min_{\boldsymbol{X}} \mathcal{F}(\boldsymbol{X}) \quad \text{s.t. } \boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} = \boldsymbol{I},$$
$$\min_{\boldsymbol{X}} \mathcal{F}(\boldsymbol{X}) \quad \text{s.t. } \boldsymbol{X}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{X} = \boldsymbol{K},$$
$$\min_{\boldsymbol{X}_1,\ldots,\boldsymbol{X}_q} \mathcal{F}(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_q)$$
$$\text{s.t. } \boldsymbol{X}_i^{\mathrm{T}}\boldsymbol{M}_i\boldsymbol{X}_i = \boldsymbol{K}_i, i = 1, \ldots, q,$$

where $\mathcal{F}$ is a known differentiable function, and $\boldsymbol{M}$, $\boldsymbol{M}_i$, and $\boldsymbol{K}_i$ are given positive definite and nonsingular symmetric matrices. It is generally difficult to solve these problems because the orthogonality constraints can lead to many local minimizers and several types of these problems are NP-hard. No guarantee can be made for obtaining the global minimizer, except for a few simple cases such as finding the extreme eigenvalues.

Generally, the approaches to solve orthogonality constraint problems can be roughly classified into two categories[25]: 1) feasible methods that strictly satisfy the orthogonality constraints during iterations, including matrix re-orthogonalization and generating trial points along geodesics, and 2) infeasible methods that relax the constraints by penalizing their violations and thus generate infeasible intermediate points, such as various penalty, augmented Lagrangian, and SDP relaxation methods.

In this paper, the LOPA model (4) is solved by an infeasible method, since all the orthogonality constraints are rarely strictly satisfied except for a few intrinsically flat datasets with zero Gaussian curvature everywhere, such as the Swiss role data. Specifically, the SDP relaxation method is used to solve the LOPA problem, with details given in the following subsection.

3.2 Relaxation Models for LOPA

A most straightforward way to simplify (4) is to replace the multiple constraints with just a single combined constraint $\boldsymbol{Y}\boldsymbol{C}\boldsymbol{Y}^{\mathrm{T}} = \boldsymbol{I}_d$, where $\boldsymbol{C} = \sum_{i=1}^{n} \boldsymbol{C}_i/n$. This simplification can be verified by the Lagrangian function:

$$\mathcal{L}(\boldsymbol{Y}, \{\boldsymbol{W}_i\}) = tr(\boldsymbol{Y}\boldsymbol{B}\boldsymbol{Y}^{\mathrm{T}}) - \frac{1}{n}\sum_{i=1}^{n} tr\big(\boldsymbol{W}_i(\boldsymbol{Y}\boldsymbol{C}_i\boldsymbol{Y}^{\mathrm{T}} - \boldsymbol{I}_d)\big),$$

where each $\boldsymbol{W}_i$ is a Lagrangian multiplier matrix. If assuming all the multiplier matrices are identical and thus can be rewritten as $\boldsymbol{W}$, then the penalization term can be written as

$$tr\Big(\boldsymbol{W}\Big(\boldsymbol{Y}\left(\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{C}_i\right)\boldsymbol{Y}^{\mathrm{T}} - \boldsymbol{I}_d\Big)\Big).$$

Thus we can obtain an overly simplified model:

$$\min_{\boldsymbol{Y}} \ tr(\boldsymbol{Y}\boldsymbol{B}\boldsymbol{Y}^{\mathrm{T}}) \quad \text{s.t. } \boldsymbol{Y}\boldsymbol{C}\boldsymbol{Y}^{\mathrm{T}} = \boldsymbol{I}_d.$$

If considering each dimension of $\boldsymbol{Y}$, then the optimal $\boldsymbol{Y}$ is simply given by the eigenvectors corresponding to the d smallest positive generalized eigenvalues of $(\boldsymbol{B}, \boldsymbol{C} + \delta\boldsymbol{I}_n)$. Here $\delta\boldsymbol{I}_n$ is a small regularization term to avoid singularity. This overly simplified model is not amenable to embedding curved manifold data, though yielding satisfactory results for intrinsically flat data like Swiss roll.

A more practical way is to replace the difficult orthogonal constraints by easier trace constraints, leading to the following relaxation model:

$$\min_{\boldsymbol{Y}} tr(\boldsymbol{Y}\boldsymbol{B}\boldsymbol{Y}^{\mathrm{T}})$$

$$\text{s.t. } tr(\boldsymbol{Y}\boldsymbol{C}_i\boldsymbol{Y}^{\mathrm{T}}) = d,\ i = 1, \ldots, n. \tag{5}$$

Compared with the rigid orthogonality constraint $\boldsymbol{Y}\boldsymbol{C}_i\boldsymbol{Y}^{\mathrm{T}} = \boldsymbol{I}_d$, the trace constraint $tr(\boldsymbol{Y}\boldsymbol{C}_i\boldsymbol{Y}^{\mathrm{T}}) = d$ at each data point only loosely specifies the sum of the diagonals of $\boldsymbol{Y}\boldsymbol{C}_i\boldsymbol{Y}^{\mathrm{T}}$. By setting $\boldsymbol{K} \doteq \boldsymbol{Y}^{\mathrm{T}}\boldsymbol{Y} \in \mathbb{R}^{n\times n}$ and using the trace property $tr(\boldsymbol{ABC}) = tr(\boldsymbol{BCA}) = tr(\boldsymbol{CAB})$, the model (5) can be rewritten as

$$\begin{aligned} &\min_{\boldsymbol{K}} tr(\boldsymbol{BK}) \\ &\text{s.t. } \boldsymbol{K} \succeq 0,\quad tr(\boldsymbol{C}_i\boldsymbol{K}) = d,\ i = 1, \ldots, n, \end{aligned} \tag{6}$$

where $\boldsymbol{K} \succeq 0$ means that it is a positive semi-definite matrix. Note that $\boldsymbol{K}$ is of rank d by its definition.

3.3 Connection to MVU

It is interesting to connect the LOPA model (6) with the MVU model[8-9], which is given by:

$$\begin{aligned} &\max_{\boldsymbol{K}}\ tr(\boldsymbol{K}) \\ &\text{s.t. } \boldsymbol{K} \succeq 0,\quad tr(\boldsymbol{e}\boldsymbol{e}^{\mathrm{T}}\boldsymbol{K}) = 0, \\ &\boldsymbol{K}_{ii} - 2\boldsymbol{K}_{ij} + \boldsymbol{K}_{jj} = \boldsymbol{D}_{ij},\ j \in \Omega_i, \end{aligned} \tag{7}$$

where $\boldsymbol{D}_{ij} \doteq \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2$ is the squared distance between two neighbors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. The last constraint in (7) is just $\|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2 = \boldsymbol{D}_{ij}$ represented by $\boldsymbol{K}$, implying that the main purpose of MVU is to preserve distances between any two neighbor points. The second constraint enforces that the embeddings of all data points should be centered on the origin:

$$\begin{aligned} \sum_i \boldsymbol{y}_i = \boldsymbol{0} &\Rightarrow \sum_{i,j} \boldsymbol{y}_i^{\mathrm{T}}\boldsymbol{y}_j = 0 \\ &\Rightarrow \boldsymbol{Ye} = 0 \\ &\Rightarrow tr(\boldsymbol{e}^{\mathrm{T}}\boldsymbol{Y}^{\mathrm{T}}\boldsymbol{Y}\boldsymbol{e}) = 0 \\ &\Rightarrow tr(\boldsymbol{e}\boldsymbol{e}^{\mathrm{T}}\boldsymbol{K}) = 0. \end{aligned}$$

The objective function of MVU is derived as follows:

$$\begin{aligned} tr(\boldsymbol{K}) &= tr(\boldsymbol{Y}^{\mathrm{T}}\boldsymbol{Y}) \\ &= \sum_i \|\boldsymbol{y}_i\|^2 \\ &= \frac{1}{2n}\sum_{i,j}(\|\boldsymbol{y}_i\|^2 + \|\boldsymbol{y}_j\|^2 - 2\boldsymbol{y}_i^{\mathrm{T}}\boldsymbol{y}_j) \\ &= \frac{1}{2n}\sum_{i,j}\|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2, \end{aligned}$$

where the zero mean constraint is used in the third equality. Therefore, it is clear that MVU attempts to unfold the curved manifold by maximizing the averaged squared distance between any two embedding points (need not to be k-nearest neighbors) under the distance preserving constraint, thus getting its algorithmic name.

We can see that the objective function $\max tr(\boldsymbol{K}) = \min tr(-\boldsymbol{IK})$ of MVU (7) is similar to $\min tr(\boldsymbol{BK})$ of LOPA (6). However, there are approximately $nk/2$ constraints of pairwise distances in the MVU model. In contrast, LOPA (6) has only n constraints, having lower complexity than MVU since solving SDP problems is the most expensive step in both LOPA and MVU. Most algorithms for solving SDPs are based on interior-point methods, including CSDP[26], SDPT3[27], and SeDuMi[28]. As a rough rule-of-thumb, interior-point methods solve semidefinite problems in about 5~50 iterations[29], and the number of iterations seems to grow slowly with the size of the problem. Thereby the computational complexity of a single iteration is often studied for SDP problems. However there is no unified complexity for each iteration of different computation methods. For example, CSDP requires $O(m(n^2m + n^3) + m^3 + n^3)$ time in each iteration, where m is the number of equity constraints and n is the size of the symmetric semidefinite matrix to be solved. SDPT3 takes approximately $O(4mn^3 + m^2n^2)$ complexity in each iteration using the Hadamard product formula with the AHO search direction. Clearly the running time is much faster for an SDP problem with a smaller size of constraints.

3.4 Solution to LOPA

It is well known that the LOPA model (6) is a standard formulation of semi-definite programming (SDP) and the optimal $\boldsymbol{K}$ can be solved by any off-the-shelf convex optimization toolbox like SDPT3[27], CSDP[26], and SeDuMi[28]. In general, the obtained $\boldsymbol{K}$ may not satisfy the rank d constraint coming from the definition $\boldsymbol{K} = \boldsymbol{Y}^{\mathrm{T}}\boldsymbol{Y}$. Then by eigenvalue-decomposition of $\boldsymbol{K}$, we get an initial solution of the embedding, $\boldsymbol{Y}_0 = \boldsymbol{V}\boldsymbol{D}^{\frac{1}{2}}$, where $\boldsymbol{D}$ and $\boldsymbol{V}$ are the top d eigenvalue diagonal matrix and the corresponding eigenvectors of $\boldsymbol{K}$, respectively. Here $\boldsymbol{D}^{\frac{1}{2}}$ denotes the diagonal matrix formed by the square roots of the top d eigenvalues.

Recall that we only solved the relaxation version (6) to approximate the original LOPA problem (4). Starting from the initial SDP solution $\boldsymbol{Y}_0$, it is usually possible to find better $\boldsymbol{Y}$ such that both the cost function $tr(\boldsymbol{Y}\boldsymbol{B}\boldsymbol{Y}^{\mathrm{T}})$ and the penalty terms $\|\boldsymbol{Y}\boldsymbol{C}_i\boldsymbol{Y}^{\mathrm{T}} - \boldsymbol{I}_d\|_F^2$ can be further decreased. Here we directly use the two-step iterative procedure suggested by the Appendix of [15]:

1) For a fixed $\boldsymbol{Y}$, solve $\min_{\boldsymbol{L}_i} \|\boldsymbol{Y}\boldsymbol{S}_i\boldsymbol{H}_k - \boldsymbol{L}_i\boldsymbol{\Theta}_i\|_F^2$ to obtain an optimal orthogonal transformation $\boldsymbol{L}_i$ for each $\boldsymbol{x}_i$. This is the standard orthogonal Procrustes problem (see Algorithm 12.4.1 of [30]), with solution given by $\boldsymbol{L}_i = \boldsymbol{U}_i\boldsymbol{V}_i^{\mathrm{T}}$ where $\boldsymbol{Y}\boldsymbol{S}_i\boldsymbol{H}_k\boldsymbol{\Theta}_i^{\mathrm{T}} = \boldsymbol{U}_i\boldsymbol{\Sigma}_i\boldsymbol{V}_i^{\mathrm{T}}$ is the singular value decomposition (SVD).

2) For the fixed $\{\boldsymbol{L}_i\}$, $i = 1, \ldots, n$, solve the least squares (LS) problem $\min_{\boldsymbol{Y}} \sum_i \|\boldsymbol{Y}\boldsymbol{S}_i\boldsymbol{H}_k - \boldsymbol{L}_i\boldsymbol{\Theta}_i\|_F^2$ to update $\boldsymbol{Y}$.

3) Repeat the above two steps until the updates in $\boldsymbol{Y}$ are rather small, formally $\|\boldsymbol{Y}^{(t+1)} - \boldsymbol{Y}^{(t)}\|_F / \|\boldsymbol{Y}^{(t)}\|_F \leqslant \epsilon$, or $t \geqslant t_{\max}$.

The entire LOPA procedure is given in Algorithm 1.

4 Experiments

We compare our algorithm with other dimensionality reduction methods, including PCA, LLE, LTSA, Isomap, PSA, MVU, and MVE. Aside from the results on synthetic data, we also show visualization results on pose varying data and motion sequence. Moreover classification performance is evaluated on low-dimensional embeddings of five diverse datasets.

To produce a quantitative evaluation, we introduce four averaged measures reflecting geometric changes before and after the embedding in each neighborhood. These measures are relative errors in distances ($R_{k\mathrm{dist}}$), relative errors in angles ($R_{k\mathrm{angl}}$), relative errors in inner products ($R_{k\mathrm{inner}}$) among any three neighbors, and change rates in k-nearest neighborhood ($R_{k\mathrm{nn}}$):

$$R_{k\mathrm{dist}} = \frac{\sum_{i=1}^n \sum_{j=2}^k |(\boldsymbol{x}_{i_j} - \boldsymbol{x}_i)^2 - (\boldsymbol{y}_{i_j} - \boldsymbol{y}_i)^2|}{\sum_{i=1}^n \sum_{j=2}^k (\boldsymbol{x}_{i_j} - \boldsymbol{x}_i)^2},$$

$$R_{k\mathrm{angl}} = \frac{\sum_{i=1}^n \sum_{j=3}^k |\angle \boldsymbol{x}_{i_j}\boldsymbol{x}_i\boldsymbol{x}_{i_2} - \angle \boldsymbol{y}_{i_j}\boldsymbol{y}_i\boldsymbol{y}_{i_2}|}{\sum_{i=1}^n \sum_{j=3}^k |\angle \boldsymbol{x}_{i_j}\boldsymbol{x}_i\boldsymbol{x}_{i_2}|},$$

$$R_{k\mathrm{inner}} = \frac{\sum_{i=1}^n \sum_{j=2}^k |\langle \boldsymbol{x}_{i_j}, \boldsymbol{x}_i\rangle - \langle \boldsymbol{y}_{i_j}, \boldsymbol{y}_i\rangle|}{\sum_{i=1}^n \sum_{j=2}^k |\langle \boldsymbol{x}_{i_j}, \boldsymbol{x}_i\rangle|},$$

$$R_{k\mathrm{nn}} = \frac{1}{kn}\sum_{i=1}^n (k - |\Omega(\boldsymbol{x}_i) \cap \Omega(\boldsymbol{y}_i)|).$$

4.1 Synthetic Data

Although viewed as "toy data" and shown over and over again in the manifold learning literature, synthetic datasets are often self-explanatory to grasp basic properties of each method. If one algorithm performs poorly on synthetic data, nobody would believe its good embedding on real-world datasets. Here five synthetic datasets are used to perform dimensionality reduction from 3D to 2D: Swiss roll, Swiss hole, punctured sphere, twin peaks, and toroidal helix. Every dataset has 800 data points, and the number of neighbors k is set to 8. Two situations are considered, without noise or with Gaussian noise. In the latter case, we add Gaussian noise $\mathcal{N}(0, c^2\sigma^2)$ on each dimension of the coordinates, where $\sigma = 0.03$ in our experiments and c is an average distance within one neighborhood for each dataset.

Fig.1 shows the visualization results. We can see that LLE and LTSA cannot maintain distances and angles due to their proximity-preserving nature. Other five methods including LOPA attempt to preserve distances or isometry, performing poorly on the punctured sphere because unfolding this curved data into flat will greatly violate the distance preserving criterion. In comparison, Isomap yields unsatisfactory or poor results on Swiss roll, Swiss hole, and punctured sphere; PSA fails to unfold Swiss roll, punctured sphere, and toroidal helix. The results offered by LOPA, MVU and MVE are very similar except that on twin peaks, LOPA performs better.

Algorithm 1. LOPA Algorithm for Nonlinear Dimensionality Reduction

Input: a high-dimensional data $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{m\times n}$ with each data point $\boldsymbol{x}_i \in \mathbb{R}^m$ and the number of data points is n
Parameters: k is the number of neighbors for each $\boldsymbol{x}_i$, and d is the target low dimension satisfying $d \ll m$
Output: a low-dimensional data $\boldsymbol{Y} = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n] \in \mathbb{R}^{d\times n}$ with each $\boldsymbol{y}_i$ being the low-dimensional representation of $\boldsymbol{x}_i$

- Denote $\boldsymbol{X}_i = [\boldsymbol{x}_{i_1}, \ldots, \boldsymbol{x}_{i_k}]$ as the k nearest neighbors of $\boldsymbol{x}_i$, and $\boldsymbol{S}_i$ is a 0-1 selection matrix such that $\boldsymbol{X}_i = \boldsymbol{X}\boldsymbol{S}_i$. $\boldsymbol{H}_k = \boldsymbol{I} - \boldsymbol{e}\boldsymbol{e}^{\mathrm{T}}/k$ is a centering matrix of size k-by-k with $\boldsymbol{e} = [1, \ldots, 1]^{\mathrm{T}}$. $\boldsymbol{\Theta}_i \in \mathbb{R}^{d\times k}$ is the d-dimensional PCA representation of each $\boldsymbol{X}_i$, with its Moore-Penrose generalized inverse denoted by $\boldsymbol{\Theta}_i^{\dagger}$.
- $\boldsymbol{C}_i = \boldsymbol{G}_i\boldsymbol{G}_i^{\mathrm{T}} \in \mathbb{R}^{n\times n}$ where $\boldsymbol{G}_i = \boldsymbol{S}_i\boldsymbol{H}_k\boldsymbol{\Theta}_i^{\dagger} \in \mathbb{R}^{n\times d}$. $\boldsymbol{B} = \sum_{i=1}^N \boldsymbol{S}_i\boldsymbol{H}_k(\boldsymbol{I} - \boldsymbol{\Theta}_i^{\dagger}\boldsymbol{\Theta}_i)(\boldsymbol{I} - \boldsymbol{\Theta}_i^{\dagger}\boldsymbol{\Theta}_i)^{\mathrm{T}}\boldsymbol{H}_k^{\mathrm{T}}\boldsymbol{S}_i^{\mathrm{T}} \in \mathbb{R}^{n\times n}$.
- Use any SDP tool to solve the problem (6) with respect to a positive semidefinite matrix $\boldsymbol{K} \in \mathbb{R}^{n\times n}$: $\min_{\boldsymbol{K}} tr(\boldsymbol{B}\boldsymbol{K})$ s.t. $\boldsymbol{K} \succeq 0$, $tr(\boldsymbol{C}_i\boldsymbol{K}) = d$, $i = 1, \ldots, n$.
- Obtain an initial solution $\boldsymbol{Y}_0 = \boldsymbol{V}\boldsymbol{D}^{\frac{1}{2}}$, where $\boldsymbol{D}$ and $\boldsymbol{V}$ are the top d eigenvalue diagonal matrix and the corresponding eigenvectors of $\boldsymbol{K}$, respectively.
- Run the two-step iterations to further refine the solution: 1) obtain a local orthogonal transformation $\boldsymbol{L}_i$ for each $\boldsymbol{x}_i$ by solving $\min_{\boldsymbol{L}_i} \|\boldsymbol{Y}\boldsymbol{S}_i\boldsymbol{H}_k - \boldsymbol{L}_i\boldsymbol{\Theta}_i\|_F^2$ with a fixed $\boldsymbol{Y}$; 2) solve the least squares (LS) problem $\min_{\boldsymbol{Y}} \sum_i \|\boldsymbol{Y}\boldsymbol{S}_i\boldsymbol{H}_k - \boldsymbol{L}_i\boldsymbol{\Theta}_i\|_F^2$ to update $\boldsymbol{Y}$ when the set of $\{\boldsymbol{L}_i\}_{i=1}^n$ is fixed; 3) stop the iterations at the maximum iteration number or the updates in $\boldsymbol{Y}$ are relatively small.

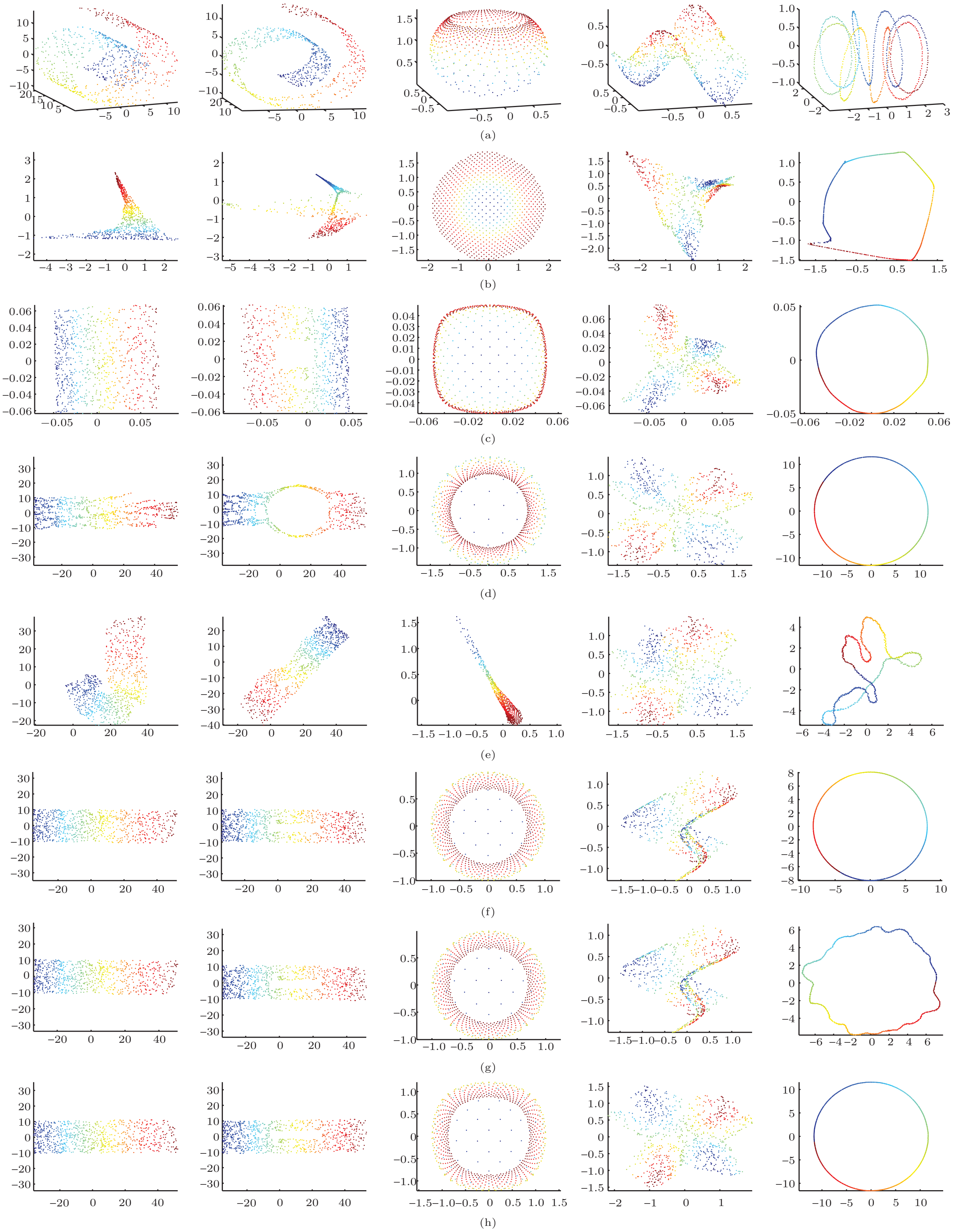


Fig.1. 3D to 2D results on synthetic data. (a) Input. (b) LLE. (c) LTSA. (d) Isomap. (e) PSA. (f) MVU. (g) MVE. (h) LOPA. From left to right: Swiss roll, Swiss hole, punctured sphere, twin peaks, and toroidal helix.

The results of the four geometric measures on synthetic datasets are listed in Table 1∼Table 4, showing that LOPA outperforms the other methods significantly in $R_{k\text{dist}}$, $R_{k\text{angl}}$, and $R_{k\text{inner}}$. Table 5 shows the running time on a PC with 2.5 GHz CPU and 4 G RAM with all algorithms implemented in Matlab. Note that the implementations of LOPA, MVU and MVE use SDPT3[27] to solve SDP problems for fairness. It is clear that LOPA runs much faster than PSA, MVU, and MVE.

Table 1. $R_{k\text{dist}}$

Dataset	σ	LLE (%)	LTSA (%)	Isomap (%)	PSA (%)	MVU (%)	MVE (%)	LOPA (%)
Swiss roll	0.00	99.53	100.00	48.14	18.13	4.22	1.67	**0.42**
	0.03	99.00	100.00	44.98	16.54	8.98	6.14	**1.09**
Swiss hole	0.00	99.45	100.00	41.81	18.42	3.84	1.64	**0.42**
	0.03	99.08	100.00	38.95	19.49	8.77	10.09	**0.94**
Punctured sphere	0.00	80.91	99.83	79.93	53.59	44.70	32.15	**24.06**
	0.03	80.80	99.83	79.87	57.70	44.64	32.26	**23.72**
Twin peaks	0.00	185.82	99.83	32.76	25.62	27.53	34.38	**9.94**
	0.03	53.81	99.85	36.33	21.67	26.86	28.19	**10.29**
Toroidal helix	0.00	97.07	99.77	99.62	14.92	1.13	**0.18**	0.65
	0.03	97.07	99.77	99.49	8.74	1.16	**0.22**	0.67

Table 2. $R_{k\text{angl}}$

Dataset	σ	LLE (%)	LTSA (%)	Isomap (%)	PSA (%)	MVU (%)	MVE (%)	LOPA (%)
Swiss roll	0.00	51.27	34.98	20.60	11.23	0.64	0.45	**0.41**
	0.03	48.04	37.86	21.35	10.08	4.28	4.35	**4.14**
Swiss hole	0.00	38.83	35.16	27.20	10.18	0.67	0.50	**0.44**
	0.03	34.45	36.70	31.32	13.06	4.75	4.95	**4.17**
Punctured sphere	0.00	9.74	43.75	25.87	32.27	23.60	20.52	**7.94**
	0.03	9.76	43.75	25.99	45.45	23.38	20.67	**8.94**
Twin peaks	0.00	29.53	11.88	21.17	16.98	18.39	8.77	**6.50**
	0.03	14.41	14.94	22.84	11.98	15.70	9.99	**5.92**
Toroidal helix	0.00	4.36	**0.06**	4.36	2.55	3.49	4.36	3.08
	0.03	4.33	**0.24**	4.33	1.74	2.94	3.74	3.23

Table 3. $R_{k\text{inner}}$

Dataset	σ	LLE (%)	LTSA (%)	Isomap (%)	PSA (%)	MVU (%)	MVE (%)	LOPA (%)
Swiss roll	0.00	99.53	100.00	64.00	29.26	4.24	1.61	**0.57**
	0.03	98.92	100.00	55.58	26.08	8.79	7.06	**2.30**
Swiss hole	0.00	99.46	100.00	66.48	30.01	3.76	1.55	**0.55**
	0.03	99.07	100.00	62.94	30.66	8.36	9.47	**2.05**
Punctured sphere	0.00	82.83	99.88	95.71	69.04	89.09	65.82	**34.76**
	0.03	82.85	99.88	96.05	54.11	88.77	65.72	**34.73**
Twin peaks	0.00	251.56	99.83	50.12	42.73	34.67	34.05	**16.24**
	0.03	70.63	99.84	53.56	34.83	32.21	28.64	**16.04**
Toroidal helix	0.00	97.06	99.77	100.05	15.31	1.01	**0.24**	0.68
	0.03	97.06	99.77	100.10	8.78	0.97	**0.32**	0.85

Table 4. $R_{k\text{nn}}$

Dataset	σ	LLE (%)	LTSA (%)	Isomap (%)	PSA (%)	MVU (%)	MVE (%)	LOPA (%)
Swiss roll	0.00	53.58	38.19	14.56	12.31	0.56	**0.22**	0.23
	0.03	48.45	39.39	13.89	7.48	1.17	1.66	**0.55**
Swiss hole	0.00	36.02	36.13	17.98	9.77	0.52	0.20	**0.13**
	0.03	33.53	37.20	21.13	10.16	1.06	1.44	**0.50**
Punctured sphere	0.00	**12.55**	60.08	34.78	38.77	27.92	34.80	24.83
	0.03	**12.30**	60.11	34.73	63.80	28.29	34.73	26.27
Twin peaks	0.00	31.94	**12.41**	15.91	32.16	40.85	16.67	32.19
	0.03	**15.16**	15.48	16.16	27.02	39.17	36.45	33.30
Toroidal helix	0.00	**0.16**	87.50	5.09	31.33	0.19	**0.16**	84.77
	0.03	0.63	84.06	0.16	32.94	0.29	**0.17**	82.86

Table 5. Running Time (s)

Dataset	σ	LLE	LTSA	Isomap	PSA	MVU	MVE	LOPA
Swiss roll	0.00	0.257	0.444	7.439	1 866.953	652.166	2 414.328	144.110
	0.03	0.261	0.449	6.665	1 491.382	598.826	2 506.421	143.023
Swiss hole	0.00	0.257	0.441	6.861	1 780.307	566.924	2 848.470	152.897
	0.03	0.253	0.447	6.834	2 780.833	610.673	3 499.761	142.318
Punctured sphere	0.00	0.241	0.457	6.830	533.858	160.716	437.917	99.776
	0.03	0.242	0.465	9.443	2 419.072	175.084	639.375	138.883
Twin peaks	0.00	0.258	0.448	6.922	3 631.019	112.332	1 588.542	110.066
	0.03	0.282	0.455	9.341	4 482.946	188.189	1 887.481	110.929
Toroidal helix	0.00	0.206	0.412	6.874	2 710.461	436.081	1 255.400	85.829
	0.03	0.209	0.397	7.185	4 608.337	226.327	1 741.251	141.108

4.2 Real Data: Pose Varying

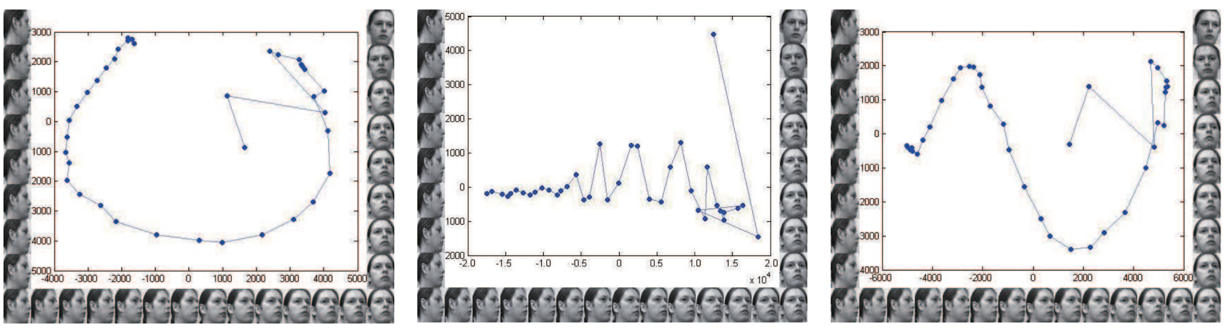
In this test, we compare the ability of recovering the pose transition of facial images and Coil data. Fig.2 compares LOPA with MVU and MVE on 2D embedding of the UMist facial images for one person. Since the inherent structure should be a circular arc, we can see that MVU thoroughly fails to uncover this structure and MVE method yields a sin-like curve. In contrast, LOPA reveals the underlying structure as a circular arc. Fig.3 displays 2D embedding results of the images "duck" and "cat" in Coil data. Each group of Coil images, such as the "duck" and the "cat", was captured at every 5° by rotating the objects. Hence the Coil images should have an inherent circle structure. In comparison, LOPA, MVU, and Isomap perform the best on the two groups of Coil images, while PSA fails to detect the circular structure.

4.3 Real Data: Motion Sequence

We use the UCF-sport dataset to explore the low-dimensional representation of a motion image sequence. The 2D embedding of a basketball video clip with 140 frames is shown in Fig.4. It can be seen that LTSA, Isomap, LOPA, and MVE can unfold the data into a roughly smooth curve, maintaining the sequential property of the motion. Since the ground-truth underlying structure of this dataset is unknown, we also report the quantitative measurements shown in Table 6. From these errors, we can see that LOPA and MVE perform much better than the other algorithms. However LOPA runs much faster than MVE.

Table 6. Runtime (s) and Geometric Measurements on a Basketball Video

Method	Time (s)	$R_{k\text{dist}}$ (%)	$R_{k\text{angl}}$ (%)	$R_{k\text{inner}}$ (%)	$R_{k\text{nn}}$ (%)
PCA	0.359 0	81.10	46.74	82.63	22.94
LLE	0.175 3	100.00	56.90	100.00	36.06
LTSA	**0.150 0**	100.00	44.10	100.00	22.09
Isomap	0.313 0	41.39	45.14	60.70	9.04
PSA	211.751 0	10.61	37.10	50.49	51.21
MVU	9.736 0	49.61	39.26	43.94	9.75
MVE	135.842 9	43.33	**35.07**	56.08	**7.04**
LOPA	2.153 0	**9.71**	36.26	**42.11**	7.99



(a) (b) (c)

Fig.2. 2D embedding of the UMist face images by using (a) LOPA, (b) MVU and (c) MVE. The low-dimensional shape of this dataset should be an arc in the embedding space.

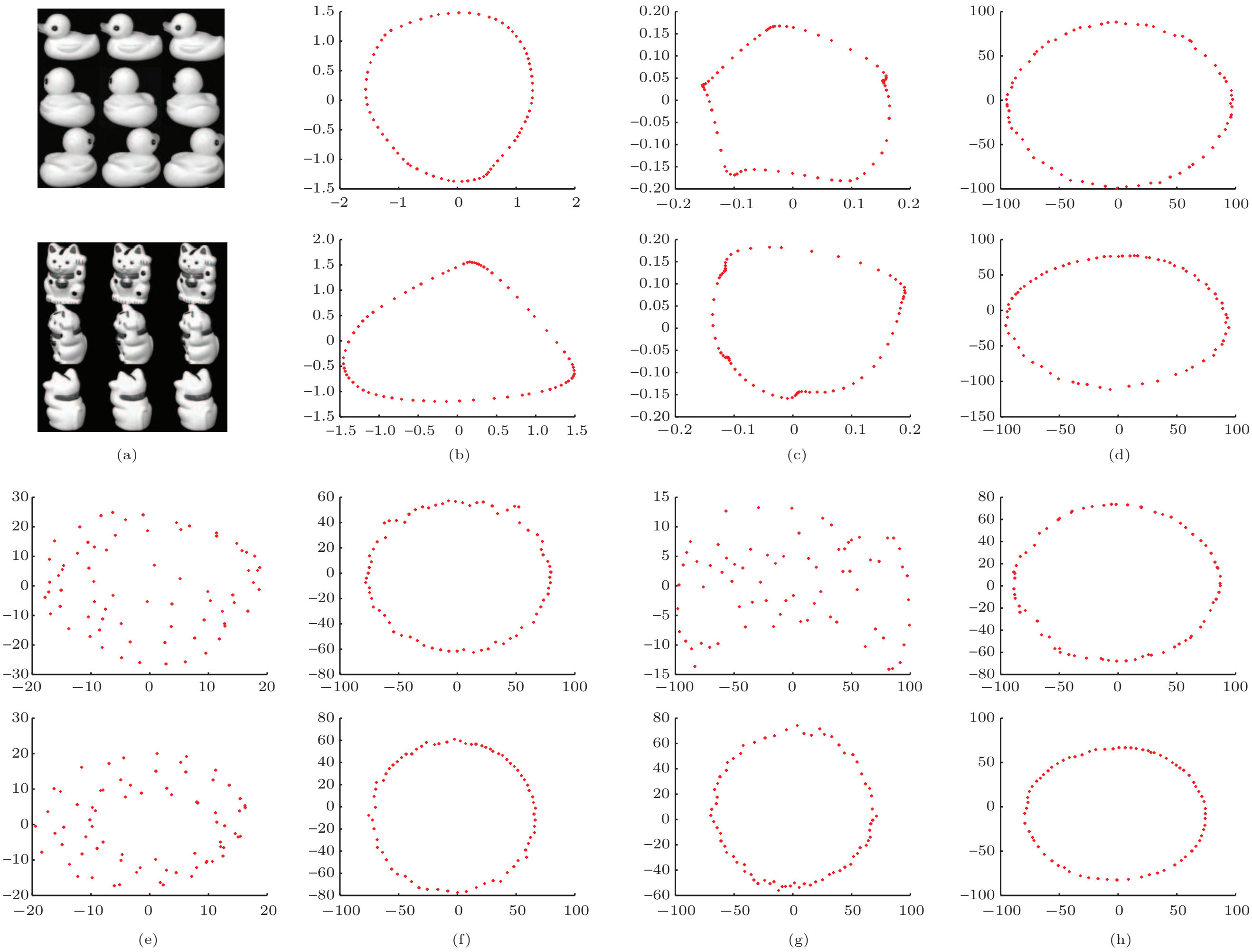


Fig.3. Duck images and cat images in Coil. Note that only part of images are shown in (a). The low-dimensional shapes of the two data sets should be a smooth circle in the embedding space. (a) Input. (b) LLE. (c) LTSA. (d) Isomap. (e) PSA. (f) MVU. (g) MVE. (h) LOPA.

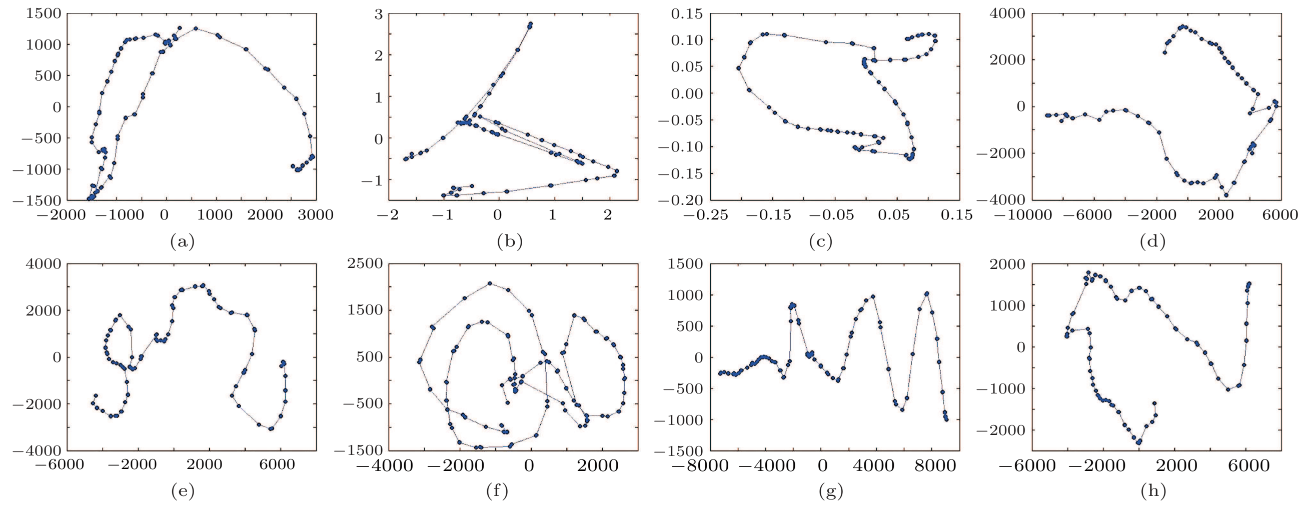


Fig.4. 2D embedding of a basketball video. (a) PCA. (b) LLE. (c) LTSA. (d) Isomap. (e) LOPA. (f) PSA. (g) MVU. (h) MVE. The low-dimensional shape of the dataset should be a continuous curve in the embedding space.

4.4 Real Data: Classification

We test the classification performance using k-nearest neighbor classifier after dimensionality reduction. Five datasets are used for this purpose. Both the MNIST dataset and the USPS dataset are handwritten digits. The ORL dataset consists of 400 facial images of 40 persons under different conditions. The HIVA dataset is a drug discovery dataset with two-class labels. The UCI satellite dataset is an infra-red astronomy database with six classes. Some datasets are too large for algorithms like PSA and MVE, and thus we randomly sample 600∼800 data points from each dataset. Each dataset is preprocessed by using PCA to transform the data into a 100-dimensional space beforehand, and then we run different dimensionality reduction algorithms further to embed the datasets into a very-low dimension.

Table 7 shows the errors of a k-nearest neighbor classifier on embeddings produced by different dimensionality reduction methods. Some parameters are listed in the table. Within the eight unsupervised methods, we can see that PCA performs well on most datasets and takes the first place on two digits data. In comparison with the other unsupervised methods, LOPA achieves the best on the HIVA dataset and the UCI satellite dataset. We argue that the main purpose of manifold learning is to faithfully preserve the original geometric properties of the input data when reducing the dimensionality. Since the class label information is not used, manifold learning may not compete with other discriminant dimensionality reduction methods like Fisher's discriminant analysis (FDA). By contrast, results of FDA are listed in the last column of Table 7 showing the lowest errors when compared with the other unsupervised methods. The survey of [31] has claimed that most of manifold learning methods are even inferior to PCA when using 1-NN classifiers on real datasets.

Fig.5 displays comparison of the 2D embedding results of digits 5∼7 from the MNIST data and from the USPS data. The results indicate that proximity preserving methods like LLE and LTSA often fail to correctly separate the three classes of digits, while PSA yields totally mix-up embeddings on digits. LOPA, together with MVU and MVE, can yield embedding results that are highly separable for different digits. It implies that LOPA can serve as a feature extraction method for digit classification.

5 Conclusions

We proposed a new manifold learning algorithm called Local Orthogonality Preserving Alignment (LOPA). Our algorithm is built upon the neighborhood alignment framework suggested by LTSA, and extends the general linear transformations in LTSA into orthogonal alignments. LOPA overcomes the difficulties in PSA by using the pseudo-inverse trick to avoid multiple incompatible local transformations. Compared with the complicated simulated annealing method used in PSA, we used more efficient SDP relaxation to find the numerical solutions. Experimental results demonstrated that LOPA could produce embedding results comparable to state-of-the-art algorithms like MVU and MVE. Particularly, our method can faithfully preserve distances, angels, inner products, and the neighborhood of the input data. On the other hand, the complexity of LOPA is much lower than that of MVU and MVE because the number of constraints used in LOPA is smaller. Our future work is to investigate efficient numerical methods, to incorporate discriminant information in labels, and to explore some real applications in visualization and classification.

Table 7. Test Errors (%) of k-Nearest-Neighbor (kNN) Classification (Leave-One-Out) on Low-Dimensional Data Representation Produced by Multiple Dimensionality Reduction Methods

Data	Parameters				Test Errors (%)								
	n	D	K_{dr}	K_{n}	PCA	LLE	LTSA	Isomap	PSA	MVU	MVE	LOPA	FDA
USPS	600	10	15	15	**1.67**	2.83	2.16	6.33	12.33	5.67	5.50	8.83	**0.67**
MNIST	600	20	20	20	**1.34**	2.67	45.50	2.00	18.33	2.00	2.00	7.00	**0.50**
ORL	400	8	10	3	4.75	21.00	40.75	19.25	31.75	6.00	**4.50**	5.50	**3.00**
HIVA	800	15	15	3	3.50	3.75	**3.25**	3.63	3.25	3.38	3.37	**3.25**	**3.25**
Satellite	800	12	15	15	13.63	17.00	16.50	15.00	23.25	15.25	14.75	**13.62**	**13.00**

Note: n: number of data points; D: intrinsic dimension estimated by Dr Toolbox (also as embedding dimension); K_{dr}: number of neighbors used in dimensional reduction; K_{n}: number of neighbors used in kNN classifier.

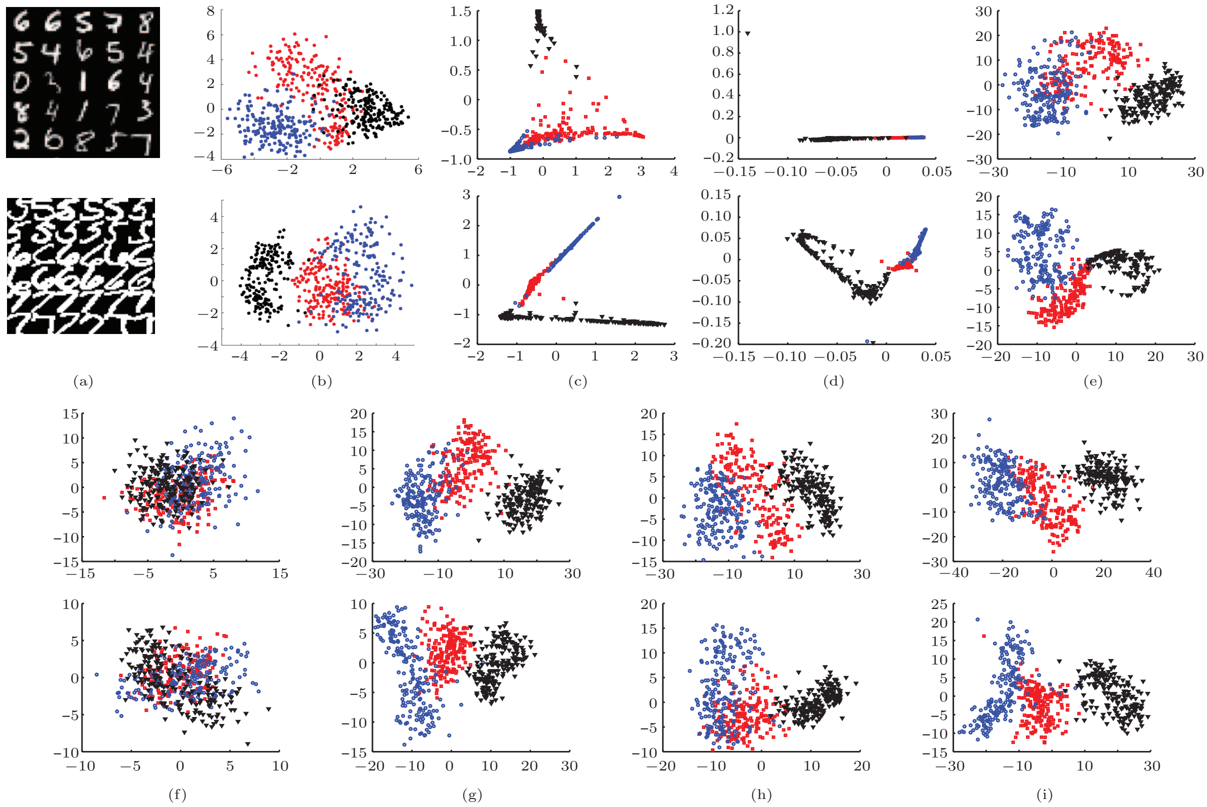


Fig.5. 2D embeddings of selected digits (5∼7). (a) Input. (b) PCA. (c) LLE. (d) LTSA. (e) Isomap. (f) PSA. (g) MVU. (h) MVE. (i) LOPA. Top row of each subfigure: the MNIST data; bottom row of each subfigure: the USPS data. Note only example images are shown in (a) input.

References

[1] Tenenbaum J B, de Silva V, Langford J C. A global geometric framework for nonlinear dimensionality reduction. *Science*, 2000, 290(5500): 2319-2323.

[2] Roweis S T, Saul L K. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 2000, 290(5500): 2323-2326.

[3] Saul L K, Roweis S T. Think globally, fit locally: Unsupervised learning of low dimensional manifolds. *The Journal of Machine Learning Research*, 2003, 4: 119-155.

[4] Seung H S, Lee D D. The manifold ways of perception. *Science*, 2000, 290(5500): 2268-2269.

[5] Donoho D L. High-dimensional data analysis: The curses and blessings of dimensionality. In *Proc. AMS Mathematical Challenges of the 21st Century*, Aug. 2000.

[6] Pei Y, Huang F, Shi F, Zha H. Unsupervised image matching based on manifold alignment. *IEEE Trans. Pattern Analysis and Machine Intelligence* (*TPAMI*), 2012, 34(8): 1658-1664.

[7] Zhang J, Huang H, Wang J. Manifold learning for visualizing and analyzing high-dimensional data. *IEEE Intelligent Systems*, 2010, 25(4): 54-61.

[8] Weinberger K Q, Saul L K. Unsupervised learning of image manifolds by semidefinite programming. *International Journal of Computer Vision*, 2006, 70(1): 77-90.

[9] Weinberger K Q, Sha F, Saul L K. Learning a kernel matrix for nonlinear dimensionality reduction. In *Proc. the 21st Int. Conf. Machine Learning*, July 2004.

[10] Shaw B, Jebara T. Minimum volume embedding. In *Proc. the 11th Int. Conf. Artificial Intelligence and Statistics*, Mar. 2007, pp.460-467.

[11] Lin T, Zha H. Riemannian manifold learning. *TPAMI*, 2008, 30(5): 796-809.

[12] Sha F, Saul L K. Analysis and extension of spectral methods for nonlinear dimensionality reduction. In *Proc. the 22nd Int. Conf. Machine Learning*, Aug. 2005, pp.784-791.

[13] Donoho D L, Grimes C. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proc. the National Academy of Sciences*, 2003, 100(10): 5591-5596.

[14] Belkin M, Niyogi P. Laplacian Eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 2003, 15(6): 1373-1396.

[15] Zhang Z, Zha H. Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. *SIAM Journal of Scientific Computing*, 2004, 26(1): 313-338.

[16] Qiao H, Zhang P, Wang D, Zhang B. An explicit nonlinear mapping for manifold learning. *IEEE Trans. Cybernetics*, 2013, 43(1): 51-63.

[17] Yang L. Alignment of overlapping locally scaled patches for multidimensional scaling and dimensionality reduction. *TPAMI*, 2008, 30(3): 438-450.

[18] Wang R, Shan S, Chen X, Chen J, Gao W. Maximal linear embedding for dimensionality reduction. *TPAMI*, 2011, 33(9): 1776-1792.

[19] Gashler M, Ventura D, Martinez T. Manifold learning by graduated optimization. *IEEE Trans. Systems, Man, and Cybernetics – Part B: Cybernetics*, 2011, 41(6): 1458-1470.

[20] Venna J, Peltonen J, Nybo K, Aidos H, Kaski S. Information retrieval perspective to nonlinear dimensionality reduction for data visualization. *The Journal of Machine Learning Research*, 2010, 11: 451-490.

[21] Goldberg Y, Ritov Y. Local Procrustes for manifold embedding: A measure of embedding quality and embedding algorithms. *Machine Learning*, 2009, 77(1): 1-25.

[22] do Carmo M P. Riemannian Geometry (1st edition). Birkhäuser, Boston, 1992.

[23] Perraul-Joncas D, Meila M. Non-linear dimensionality reduction: Riemannian metric estimation and the problem of geometric discovery. arXiv:1305.7255, 2013.

[24] Kokiopoulou E, Saad Y. Orthogonal neighborhood preserving projections: A projection-based dimensionality reduction technique. *TPAMI*, 2007, 29(12): 2143-2156.

[25] Wen Z, Yin W. A feasible method for optimization with orthogonality constraints. *Mathematical Programming*, 2013, 142(1/2): 397-434.

[26] Borchers B. CSDP, A C library for semidefinite programming. *Optimization Methods and Software*, 1999, 11(1/2/3/4): 613-623.

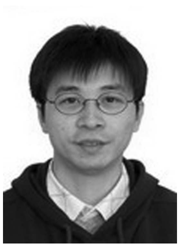
[27] Toh K, Todd M, Tütüncü R. SDPT3 — A Matlab software package for semidefinite programming, version 1.3. *Optimization Methods and Software*, 1999, 11(1-4): 545-581.

[28] Sturm J. Using SeDuMi 1.02, a Matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 1999, 11(1/2/3/4): 625-653.

[29] Vandenberghe L, Boyd S. Semidefinite programming. *SIAM Review*, 1996, 38(1): 49-95.

[30] Golub G H, van Van Loan C F. Matrix Computations (3rd edition). The Johns Hopkins University Press, 1996.

[31] van der Maaten L, Postma E, van den Herik J. Dimensionality reduction: A comparative review. Technical Report, TiCC TR 2009-005, Tilburg Centre for Creative Computing, Tilburg University, Oct. 2009.



Tong Lin received his Ph.D. degree in applied mathematics from Peking University, Beijing, in 2001. In 1999 and 2000, he was a visiting student at the Media Computing Group, Microsoft Research Asia, Beijing. In 2002, he joined the Key Laboratory of Machine Perception at Peking University, where he is currently an associate professor. From 2004 to 2005, he was an exchange scholar at Seoul National University, Korea. From 2007 to 2008, he was an exchange scholar at UCSD Moores Cancer Center, USA. His recent research interests are machine learning algorithms and theories.

Yao Liu is currently pursuing his B.S. degree in machine intelligence at Peking University, Beijing. His research mainly focuses on machine learning, computational learning theory, and reinforcement learning.

Bo Wang received his M.S. degree in intelligent science and technology from Peking University, Beijing, in 2014. In 2011 he received his B.S. degree in mathematics from Nanjing University, Nanjing. Now he is a software engineer at the Industrial Bank, Shanghai.

Li-Wei Wang is a professor of the School of Electronics Engineering and Computer Sciences, Peking University, Beijing. He received his Ph.D. degree in applied mathematics from Peking University, Beijing, in 2005, his B.S. and M.S. degrees in electronic engineering from Tsinghua University, Beijing, in 1999 and 2002 respectively. His research interest is machine learning theory. He was named among "AI's 10 to Watch" in 2010.

Hong-Bin Zha received his B.E. degree in electrical engineering from Hefei University of Technology, Hefei, in 1983, and his M.S. and Ph.D. degrees in electrical engineering from Kyushu University, Fukuoka, in 1987 and 1990, respectively. After working as a research associate at Kyushu Institute of Technology, he joined Kyushu University in 1991 as an associate professor. He was also a visiting professor in Centre for Vision, Speech, and Signal Processing, Surrey University, UK, in 1999. Since 2000, he has been a professor at the Key Laboratory of Machine Perception, Peking University, Beijing. His research interests include computer vision, digital geometry processing, and robotics. He has published more than 300 technical publications in journals, books, and international conference proceedings. He received the Franklin V. Taylor Award from the IEEE Systems, Man, and Cybernetics Society in 1999.