Embedding New Data Points for Manifold Learning Via Coordinate Propagation.

Conference Paper *in* Knowledge and Information Systems · January 2007

DOI: 10.1007/s10115-008-0161-3 · Source: DBLP

Shiming Xiang
Institute of Automation, Academy of Sciences, China
135 PUBLICATIONS **2,978** CITATIONS
SEE PROFILE

Feiping Nie
University of Texas at Arlington
301 PUBLICATIONS **7,227** CITATIONS
SEE PROFILE

Changshui Zhang
Tsinghua University
358 PUBLICATIONS **8,914** CITATIONS
SEE PROFILE

Embedding New Data Points for Manifold Learning via Coordinate Propagation

Shiming Xiang[1], Feiping Nie[1], Yangqiu Song[1], Changshui Zhang[1], and Chunxia Zhang[2]

[1] State Key Laboratory of Intelligent Technology and Systems,
Department of Automation, Tsinghua University, Beijing 100084, China
{xsm, zcs}@mail.tsinghua.edu.cn, {nfp03, songyq99}@mails.tsinghua.edu.cn
[2] School of Computer Science, Software School,
Beijing Institute of Technology, Beijing 100081, China
cxzhang@bit.edu.cn

Abstract. In recent years, a series of manifold learning algorithms have been proposed for nonlinear dimensionality reduction (NLDR). Most of them can run in a batch mode for a set of given data points, but lack a mechanism to deal with new data points. Here we propose an extension approach, i.e., embedding new data points into the previously-learned manifold. The core idea of our approach is to propagate the known coordinates to each of the new data points. We first formulate this task as a quadratic programming, and then develop an iterative algorithm for coordinate propagation. Smoothing splines are used to yield an initial coordinate for each new data point, according to their local geometrical relations. Experimental results illustrate the validity of our approach.

1 Introduction

Recently, some manifold learning algorithms have been proposed for nonlinear dimensionality reduction (NLDR). Typical algorithms include Isomap [1], local linear embedding (LLE) [2], Laplacian eigenmap [3], local tangent space alignment (LTSA) [4], charting [5], Hessian LLE (HLLE) [6], semi-definite embedding [7], conformal eigenmap [8], spline embedding (SE) [9], etc. Real performances on many data sets show that they are effective methods to discover the underlying structure hidden in the high-dimensional data set.

All the manifold learning algorithms are initially developed to obtain a low-dimensional embedding for a set of given data points. The problem in general is formulated as an optimization problem, in which the low-dimensional coordinates of all the given data points need to be solved. Matrix eigen-decomposition or other mathematical optimization tools (for instance, the semidefinite programming [7]) are used to obtain the final results, i.e., the intrinsic embedding coordinates. Accordingly, the algorithms run in a batch mode, once for all the input data points. When new data points arrive, one needs to rerun the algorithm with all data points. In applications, however, rerunning the algorithm may become impractical as more and more data points are collected sequentially. On the other hand, *rerunning* means the previous results are simply discarded. This may be very wasteful in computation.

Our interest here is to embed the new data points into the previously-learned results. This is also known as out-of-sample problem. In literature, out-of-sample extensions for LLE, Isomap, Laplacian Eigenmap are given by Bengio et al., using kernel tricks [10]. This problem is further formulated as an incremental learning problem, and extensions for LLE and Isomap are given by several researchers [11, 12]. These approaches are suitable for once dealing with one new data point. For more than one new data points, however, we need to embed them one by one.

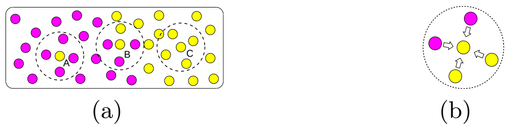


Fig. 1. (a) The task of embedding new data points; (b) coordinate propagation in a neighborhood.

Let us use Fig. 1 to explain our motivation. We are given two sets of data points which are well sampled from a manifold embedded in a high-dimensional Euclidean space. The low-dimensional embedding coordinates of one data set are also given, saying in Fig. 1(a), the data points with (dark) purple color. Under these conditions, our task is to embed the new data points (yellow points). In this work setting, the coordinates of the neighbors of a new data point may be known, partly known or even all unknown (Fig. 1(a)).

Our idea to solve this problem is to propagate the known coordinates to the new data points. To this end, we first consider this problem in view of coordinate reconstruction in the intrinsic space and formulate it as a quadratic programming. In this way, we can get a global embedding for the new data points. Then, we develop an iterative algorithm through a regularization framework. Through iterations, each new data point gradually obtains a coordinate (Fig. 1(b)). We call this process *coordinate propagation*. We also use smoothing splines to generate an initial coordinate for each new data point to speed up the precess.

2 Model and Algorithm

2.1 Problem Formulation

The NLDR problem. Given a set of data points $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \subset \mathbb{R}^m$, which lie on a manifold M embedded in a m-dimensional Euclidean space. The goal is to invert an underlying generative model $\mathbf{x} = f(\mathbf{y})$ to find their low-dimensional parameters (embedding coordinates) $\mathcal{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n\} \subset \mathbb{R}^d$ with $d < m$. In this form, NLDR is also known as manifold learning.

The out-of-sample extension problem. Given a set of data points $\mathcal{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_l, \mathbf{x}_{l+1}, \cdots, \mathbf{x}_n\} \subset \mathbb{R}^m$ and the low-dimensional embedding coordinates $\mathcal{Y}_L =$

$\{\mathbf{y}_1^0, \cdots, \mathbf{y}_l^0\} \subset \mathbb{R}^d$ learned from the first l data points. The goal is to obtain the low-dimensional coordinates $\mathcal{Y}_U = \{\mathbf{y}_{l+1}, \cdots, \mathbf{y}_n\}$ of the rest $n - l$ data points, according to their relevances to the first l data points.

2.2 Model

A large family of nonlinear manifold learning algorithms can be viewed as the approaches based on minimizing the low-dimensional coordinate reconstruction error. The algorithms in this family include LLE, Laplacian eigenmap, LTSA, and spline embedding (SE). The optimization problem can be uniformly formulated as follows:

$$\begin{array}{ll} min & tr(Y^T M Y) \\ s.t. & Y^T C Y = I \end{array} \tag{1}$$

where tr is a trace operator, M is a $n \times n$ matrix which is calculated according to the corresponding geometrical preserving criterion, C is a $n \times n$ matrix used to constrain Y to avoid degenerate solutions, and $Y = [\mathbf{y}_1, \cdots, \mathbf{y}_n]^T$ is a $n \times d$ matrix to be solved, in which $\mathbf{y}_i$ is a d-dimensional embedding coordinate of $\mathbf{x}_i$ $(i = 1, \cdots, n)$. That is, each row of Y corresponds to a low-dimensional coordinate of a data point in $\mathcal{X}$.

Specifically, in LLE, $M = (I - W)^T (I - W)$; in Laplacian eigenmap, $M = D - W$; in LTSA, $M = S^T W^T W S$, and in SE, $M = S^T B S$. In these algorithms, C is a $n \times n$ identity matrix. Problem (1) can be easily solved via matrix eigen-decomposition.

Now we use problem (1) to solve the out-of-sample problem. Introducing the known low-dimensional embedding coordinates of the first l data points in $\mathcal{X}$, naturally we can obtain a linearly constrained optimization problem:

$$\begin{array}{ll} min & tr(Y^T M Y) \\ s.t. & \mathbf{y}_i = \mathbf{y}_i^0, \quad i = 1, 2, \cdots, l \end{array} \tag{2}$$

It seems that problem (2) is very complex since the variable to be optimized is a matrix which has $d \times n$ unknown components. Directly solving it may be very expensive due to different constraints from problem (1).

Now we rewrite Y in terms of column vectors, and denote it by $Y = [\tilde{\mathbf{y}}_1, \cdots, \tilde{\mathbf{y}}_d]$, in which $\tilde{\mathbf{y}}_i \in \mathbb{R}^n$ $(i = 1, \cdots, d)$ is the i-th coordinate component vector of all the n data points. Then

$$tr(Y^T M Y) = \sum\nolimits_{i=1}^{d} \tilde{\mathbf{y}}_i^T M \tilde{\mathbf{y}}_i \tag{3}$$

We can see that $\tilde{\mathbf{y}}_1, \cdots, \tilde{\mathbf{y}}_d$ are decoupled from each other in Eq. (3). We further write out the d coordinate components of $\mathbf{y}_i^0$ and let $\mathbf{y}_i^0 = [f_i^0(1), \cdots, f_i^0(d)]^T$, $i = 1, \cdots, l$. Based on the Lagrange multiplier method, problem (2) can be converted, equivalently, into the following d subproblems, each of which is used to optimize a coordinate component vector :

$$\begin{cases} min & \mathbf{y}^T M \mathbf{y} \\ s.t. & y_i = f_i^0(1), \quad i = 1, 2, \cdots, l \\ \cdots & \\ min & \mathbf{y}^T M \mathbf{y} \\ s.t. & y_i = f_i^0(d), \quad i = 1, 2, \cdots, l \end{cases} \tag{4}$$

where $\mathbf{y} = [y_1, \cdots, y_n]^T$ is a n-dimensional vector to be solved. Note that each subproblem is a convex quadratic programming (QP) since M is a positive semidefinite matrix[3]. Therefore, we can easily solve them.

To reduce the number of variables to be solved, we write M as follows [13]:

$$M = \begin{pmatrix} M_{ll} & M_{lu} \\ M_{ul} & M_{uu} \end{pmatrix} \tag{5}$$

where M_{ll} is a $l \times l$ sub-block, M_{lu} is a $l \times (n-l)$ sub-block, M_{ul} is a $(n-l) \times l$ sub-block, and M_{uu} is a $(n-l) \times (n-l)$ sub-block. Let $\mathbf{y}_l = [y_1, \cdots, y_l]^T$ and $\mathbf{y}_u = [y_{l+1}, \cdots, y_n]^T$. Then

$$\mathbf{y}^T M \mathbf{y} = \mathbf{y}_u^T \cdot M_{uu} \cdot \mathbf{y}_u + \mathbf{y}_l^T (M_{lu} + M_{ul}^T) \mathbf{y}_u + \mathbf{y}_l^T \cdot M_{ll} \cdot \mathbf{y}_l$$

Note that $\mathbf{y}_l$ is known in each subproblem. Substituting it into the corresponding objective function, problem (4) can be further reduced to the following QP problems:

$$\begin{cases} min & \mathbf{y}_u^T \cdot M_{uu} \cdot \mathbf{y}_u + \mathbf{h}_1^T \cdot \mathbf{y}_u \\ \cdots & \\ min & \mathbf{y}_u^T \cdot M_{uu} \cdot \mathbf{y}_u + \mathbf{h}_d^T \cdot \mathbf{y}_u \end{cases} \tag{6}$$

where $\mathbf{h}_i$ $(\in \mathbb{R}^{n-l}, i = 1, \cdots, d)$ is calculated according to $(M_{ul} + M_{lu}^T)\mathbf{y}_l$. Now each QP subproblem in (6) has only $n-l$ variables to be solved. Meanwhile, M_{uu} is also positive semidefinition[4]. Thus, each QP in (6) is a convex QP, which has a global optimum.

Finally, we can combine the d global optima of problem (6) together into $n-l$ d-dimensional coordinates. In this way, we achieve a low-dimensional global embedding for $n-l$ new data points.

2.3 Iterative Algorithm for Coordinate Propagation

In this subsection, we develop an iterative algorithm for solving the out-of-sample extension problem. The iterative algorithm can reduce the computational complexity and need much less computation resources. In an iterative framework, it would be possible for us to embed a very large number of new data points.

[3] This can be easily justified in LLE, LTSA, and SE. In Laplacian eigenmap, M is a Laplacian matrix, which is also positive semidefinition. Actually, for any vector $\mathbf{x} = [x_1, \cdots, x_n]^T \in \mathbb{R}^n$, $\mathbf{x}^T M \mathbf{x} = \mathbf{x}^T (D - W) \mathbf{x} = \frac{1}{2} \sum_{i,j} (x_i - x_j)^2 w_{ij}$. Since each component w_{ij} in W is nonnegative, then $\mathbf{x}^T M \mathbf{x} \geq 0$.

[4] For any $\mathbf{x} = [0, \cdots, 0, x_{l+1}, \cdots, x_n]^T$, since M is positive semidefinition, we have $\mathbf{x}^T M \mathbf{x} = [x_{l+1}, \cdots, x_n] \cdot M_{uu} \cdot [x_{l+1}, \cdots, x_n]^T \geq 0$. This indicates that M_{uu} is also positive semidefinition.

Here we consider one of the subproblems in problem (4) since they have the same form. For convenience, we omit the superscripts and the subscripts in the constraints, and rewrite the problem as follows:

$$\begin{array}{ll} min & \mathbf{y}^T M\mathbf{y} \\ s.t. & y_i = f_i, \quad i = 1, 2, \cdots, l \end{array} \tag{7}$$

Converting the hard constraints in (7) into soft constraints and introducing a predicting term for the new data points, we have the following regularization representation:

$$min \quad \mathbf{y}^T M\mathbf{y} + \mu_1 \sum\nolimits_{i=1}^{l} (y_i - f_i)^2 + \mu_2 \sum\nolimits_{i=l+1}^{n} (y_i - g_i)^2 \tag{8}$$

where $\mu_1 > 0$ and $\mu_2 > 0$ are regularization parameters, and g_i is a predicted value for y_i, $i = l+1, \cdots, n$. Here, g_i will be evaluated from the neighbors of $\mathbf{x}_i$. We use spline interpolation to solve this problem (in Section 3).

In (8), the first term is the smoothness constraint, which means that the embedding coordinates should not change too much between neighboring data points. The second term is the fitting constraint, which means that the estimated function y should not change too much from the given values. The third term is the predicting constraint, which means that function y should not bias too much from the predicted values. The trade-off among these three constraints are stipulated by μ_1 and μ_2. We can see that the second term is equivalent to the hard constraints in (7) in case of $\mu_1 \to \infty$.

Let us assume for the moment that each g_i is known. Differentiating the objective function with respect to $\mathbf{y} \triangleq [\mathbf{y}_l^T, \mathbf{y}_u^T]^T$, we have

$$\begin{cases} M_{ll}\mathbf{y}_l + \mu_1\mathbf{y}_l + M_{lu}\mathbf{y}_u - \mu_1\mathbf{f}_l = 0 \\ M_{uu}\mathbf{y}_u + \mu_2\mathbf{y}_u + M_{ul}\mathbf{y}_l - \mu_2\mathbf{g}_u = 0 \end{cases} \tag{9}$$

where $\mathbf{f}_l = [f_1, \cdots, f_l]^T \in \mathbb{R}^l$ and $\mathbf{g}_u = [g_{l+1}, \cdots, g_n]^T \in \mathbb{R}^{n-l}$. For an out-of-sample extension problem, we need not solve $\mathbf{y}_l$ since $\mathbf{y}_l = \mathbf{f}_l$ is known. Thus we only need to consider $\mathbf{y}_u$. Here we can see that the second equation in (9) is equivalent to the subproblem in (6) when $\mu_2 = 0$. Now it can be transformed into

$$\mathbf{y}_u = \frac{1}{1+\mu_2} S\mathbf{y}_u - \frac{1}{1+\mu_2} M_{ul}\mathbf{f}_l + \frac{\mu_2}{1+\mu_2}\mathbf{g}_u$$

where $S = I - M_{uu}$ and I is a $(n-l) \times (n-l)$ identity matrix. Let us introduce two new variables: $\alpha = \frac{1}{1+\mu_2}$ and $\beta = 1 - \alpha$. Then we can get an iteration equation:

$$\mathbf{y}_u^{(t+1)} = \alpha S\mathbf{y}_u^{(t)} - \alpha M_{ul}\mathbf{f}_l + \beta\mathbf{g}_u \tag{10}$$

Some remarks. (1). According to the theories of linear algebra, the sequence $\{\mathbf{y}_u^{(t)}\}$ generated by Eq. (10) is convergent if and only if the spectral radius of αS is less than one, i.e., $\rho(\alpha S) < 1$. Note that the spectral radius of a matrix is less than any kinds of operator norms[5]. Here we can take $\alpha = 1/(||S||_1 + 1)$. Thus

[5] $||A||_1 = \max\limits_j \{\sum_{i=1}^m |a_{ij}|, j = 1, \cdots, n\}$, $||A||_\infty = \max\limits_i \{\sum_{j=1}^n |a_{ij}|, i = 1, \cdots, m\}$, etc.

$\rho(\alpha S) < 1$. (2). In Eq. (10), the first term is the contribution from the new data points, while the second term is the contribution from the previously learned data points on the manifold. Note that these contributions are decreased since $\alpha < 1$ holds. If we only consider these two terms, then the sequence will converge to $\mathbf{y}_u^* = -\alpha(I - \alpha S)^{-1} M_{ul} f_l$. This may be far from the optimization optimum $-M_{uu}^{-1} M_{ul} f_l$, especially when α is a small number. To avoid attenuations, we introduce a compensation term, i.e., the third term in Eq. (10). Here we call it a prediction to the new data points in the iterative framework, and its contribution to $\mathbf{y}_u$ is stipulated by the positive parameter β, which is a parameter in $(0, 1)$. Therefore, it is necessary for us to get a good prediction to $\mathbf{g}_u$. We will evaluate it along the manifold via spline interpolation on the neighbors.

The steps of the iterative algorithm can be summarized as follows:

(1) Provide an initial coordinate component vector $\mathbf{g}_u^{(0)}$ (in Section 3); Let $i = 0$.

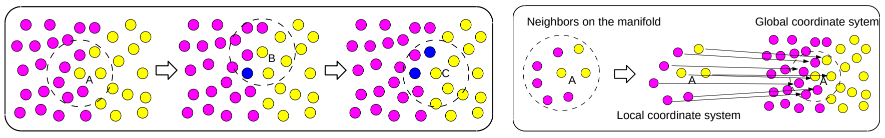
(2) Let $\mathbf{g}_u$=$\mathbf{g}_u^{(i)}$ and run the iteration equation (10) to obtain a $\mathbf{y}_u^*$.

(3) Justify if it is convergence.

(4) Predict a $\mathbf{g}_u^{(i+1)}$ according to $\mathbf{y}_u^*$ and $\mathbf{f}_l$ (in Section 3).

(5) $i = i + 1$, go to step (2).

Through the iterations, each new data point gradually receives a value (here it is a coordinate component). To construct d coordinate component vectors, we need to perform the iterative algorithm d times. In this way, the known coordinates are finally propagated to the new data points.



(a) (b)

Fig. 2. Coordinate propagation. (a) The first three steps of coordinate prediction; (b) Two steps of mapping the neighbors on the manifold to the global coordinate system.

In computation, we set the maximum iteration times to 100 when performing the iteration Equation (10). Now a task to be solved is to provide an initial $\mathbf{g}_u^{(0)}$ and generate a $\mathbf{g}_u^{(i+1)}$ in step (4). Details will be introduced in Section 3.

3 Predicting Coordinates via Smoothing Splines

We first discuss how to provide a $\mathbf{g}_u^{(0)}$. Fig. 2(a) is used to explain our idea. There we first select to predict the new data point "A" since it has the maximum number of neighbors with known coordinates. After "A" is treated, then we select "B". After "A" and "B" have been treated, "C" is one of the candidates in next time.

In the above process, a basic task can be summarized as follows. Given a new data point $\mathbf{x} \in \mathbb{R}^m$ and its k neighbors $\{\mathbf{x}_1, \cdots, \mathbf{x}_r, \mathbf{x}_{r+1}, \cdots, \mathbf{x}_k\}$. We assume that the low-dimensional coordinates of the first r data points are known. The task is to generate a coordinate for the center point $\mathbf{x}$.

Our method includes two steps (Fig. 2(b)). (1). Construct a local coordinate system to represent $\mathbf{x}$ and its k neighbors, and calculate $r+1$ local coordinates $\mathbf{t}, \mathbf{t}_1, \cdots, \mathbf{t}_r \in \mathbb{R}^d$. (2). Construct a function $g : \mathbb{R}^d \to \mathbb{R}$ through which we can get a value $f = g(\mathbf{t})$ for $\mathbf{x}$. Here f can be considered as a low-dimensional coordinate component. Furthermore, g should meet the following conditions:

$$f_j = g(\mathbf{t}_j), \quad j = 1, 2, \cdots, r \tag{11}$$

where f_j is a known coordinate component of $\mathbf{x}_j$ $(j = 1, \cdots, r)$. Actually, we use g to map the local coordinate $\mathbf{t}$ of $\mathbf{x}$ into the global coordinate system with lower dimensionality, in which the original data points are represented.

Suppose that the n data points in $\mathcal{X}$ are densely sampled from the manifold, then the tangent space of the manifold M at $\mathbf{x} \in \mathcal{X}$ can be well estimated from the neighbors of $\mathbf{x}$ [14]. We use this subspace to define the local coordinates [4, 6]. To be robustness, we simultaneously coordinatize the $k+1$ data points. Note that the computation for the rest data points $\mathbf{x}_{r+1}, \cdots, \mathbf{x}_k$ is also necessary for us to generate $\mathbf{g}_u^{(i+1)}$. After the local coordinates are evaluated, then we need to map them into the global coordinate system. Fig. 2(b) shows the above process.

To satisfy the conditions in (11), spline regression method is used to construct the function g. The spline we use is developed from the Sobolev space, and has the following form [15, 16]:

$$g(\mathbf{t}) = \sum\nolimits_{j=1}^{r} \alpha_j \, \phi_j(\mathbf{t}) + \sum\nolimits_{i=1}^{p} \beta_i \, p_i(\mathbf{t}) \tag{12}$$

where the first term is a linear combination of r Green's functions $\phi_j(\mathbf{t})$ $(j = 1, \cdots, r)$, and the second term is a polynomial in which all $p_i(\mathbf{t})$, $i = 1, \cdots, p$, constitute a base of a polynomial space. Here we take the one-degree polynomial space as an example to explain $p_i(\mathbf{t})$. Let $\mathbf{t} = [t_1, t_2]^T$ in the case of $d = 2$, we have $p_1(\mathbf{t}) = 1$, $p_2(\mathbf{t}) = t_1$ and $p_3(\mathbf{t}) = t_2$. In this case, p is equal to 3.

In addition, the Green's function $\phi_j(\mathbf{t})$ is a general radical basis function [15, 16]. For instances, in the case of $d = 2$, $\phi_j(\mathbf{t}) = (||\mathbf{t} - \mathbf{t}_j||)^2 \cdot \log(||\mathbf{t} - \mathbf{t}_j||)$; in the case of $d = 3$, $\phi_j(\mathbf{t}) = ||\mathbf{t} - \mathbf{t}_j||$.

To avoid degeneracy, we add the following conditionally positive definition constraints [17]:

$$\sum\nolimits_{j=1}^{r} \alpha_j \cdot p_i(\mathbf{t}_j) = 0, \quad i = 1, \cdots, p \tag{13}$$

Now substituting the interpolation conditions (11) into Eq. (12) and Eq. (13), we can get a linear system for solving the coefficients $\alpha \in \mathbb{R}^r$ and $\beta \in \mathbb{R}^p$:

$$\begin{pmatrix} K & P \\ P^T & 0 \end{pmatrix} \cdot \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ 0 \end{pmatrix} \tag{14}$$

where K is a $r \times r$ symmetrical matrix with elements $K_{ij} = \phi(||\mathbf{t}_i - \mathbf{t}_j||)$, P is a $r \times p$ matrix with elements $P_{ij} = p_i(\mathbf{t}_j)$, and $\mathbf{f} = [f_1, \cdots, f_r]^T \in \mathbb{R}^r$.

We point out that g is a smooth function and $f_j = g(\mathbf{t}_j)$ holds for all j, $j = 1, \cdots, r$. Faithfully satisfying the given conditions in (11) is necessary for us to rely on it to interpolate a new point $\mathbf{x}$.

To avoid error accumulation, the above performance is only used to yield a coordinate component for the center point $\mathbf{x}$ and not used to map the rest new data points $\mathbf{x}_{r+1}, \cdots, \mathbf{x}_k$. To get the d coordinate components of $\mathbf{x}$, we need to construct d splines. That is, we need to solve Eq. (14) d times. Note that in each time the coefficient matrix keeps unchanged since it is only related to the r local coordinates $\mathbf{t}_j$ ($j = 1, \cdots, r$). Finally, according to the steps as illustrated in Fig. 2, each new data point can get an initial coordinate.

To predict a new vector $\mathbf{g}_u^{(i+1)}$ during the iterations, we only need to set $r = k$ and perform the above algorithm again for each new data point.

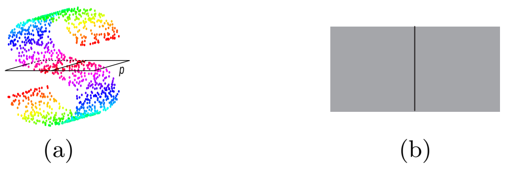


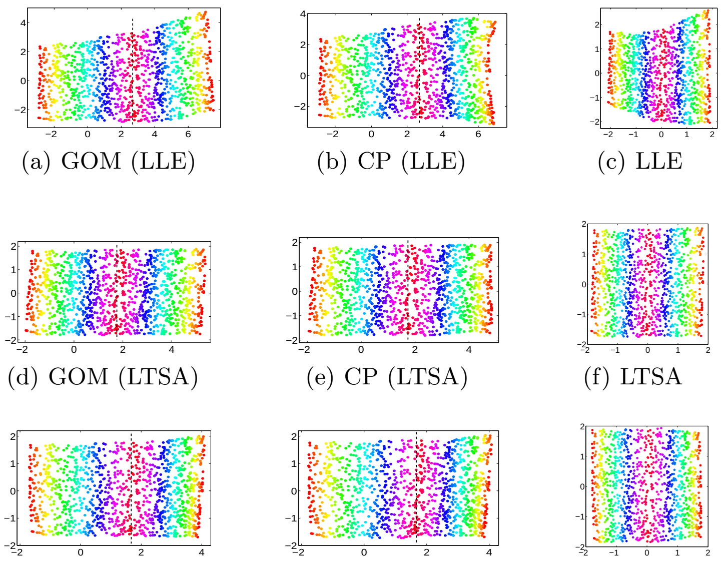
Fig. 3. (a): The 1200 data points sampled from a S-surface; (b): The intrinsic structure of the data points in (a), i.e., a 2-dimensional rectangle

4 Experimental Results

We evaluated the algorithm on several data sets including toy data points and real-world images. Here we give some results obtained by the global optimization model (GOM) and the coordinate propagation (CP). These experiments can give us a straightforward explanation on the data and the learned results. In addition, the computation complexity is also analyzed in this section.

Fig. 3(a) illustrates 1200 data points sampled from a S-surface. Among these data points, 600 data points below the horizontal plane "p" are treated as original data points, and the rest 600 data points above the plane "p" are treated as new data points. The intrinsic manifold shape hidden in these data points is a rectangle (Fig. 3(b)). The sub-rectangle located in the left of the center line in Fig. 3(b) can be viewed as the 2-dimensional (2D) parameter domain of the original data pints. Our goal is to use the new data points to extend it to the right sub-rectangle.

We use LLE, LTSA and SE to learn the original data points. The results with $k = 12$ nearest neighbors are shown in the left region of the dash line in Fig. 4(a)/4(b), Fig. 4(d)/4(e), and Fig 4(g)/4(h), resepctively. Naturally, the learned structure is only a part of the whole manifold shape. The intrinsic manifold structure hidden in these 600 original data points is a small rectangle.


(a) GOM (LLE) (b) CP (LLE) (c) LLE

(d) GOM (LTSA) (e) CP (LTSA) (f) LTSA

(g) GOM (SE) (h) CP (SE) (i) SE

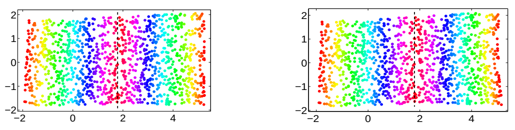
Fig. 4. The 2-dimensional embedding results of 1200 data points as illustrated in Fig. 3(a). The center lines are drawn manually

The new data points are embedded into the right region by GOM and CP. In Fig. 4(a) and Fig. 4(b), the M matrix in Eq. (5) is calculated according to LLE with $k = 12$, i.e., $M = (I - W)^T(I - W)$. In Fig. 4(d) and Fig. 4(e), the M matrix is calculated according to LTSA with $k = 12$, i.e., $M = S^T W^T W S$; In Fig 4(g) and Fig. 4(h), the M matrix is calculated according to SE with $k = 12$, i.e., $M = S^T B S$. From the ranges of the learned coordinates, we can see that the learned manifold shape is extended along the right direction by the new data points.

For comparison, Fig. 4(c), Fig. 4(f) and Fig. 4(i) show the 2D embedding results of all the 1200 data points directly by LLE, LTSA and SE. As can be seen, the results are confined into a square region, not extended to be a long rectangle, which is the real low-dimensional structure hidden in the data points.

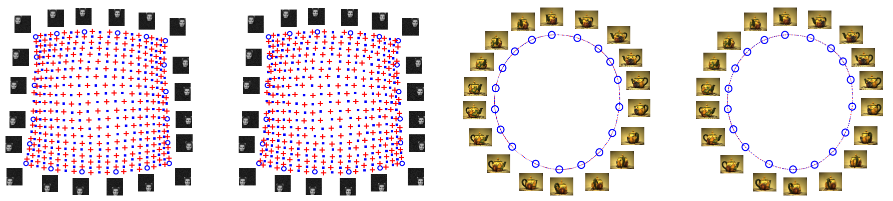
Fig. 5 shows the results by GOM and CP, which use a combination of SE and LLE. That is, the original data points are learned by SE to get their low-dimensional coordinates, but the M matrix in Eq. (5) is calculated via LLE. Compared with the results purely based on LLE (see Fig. 4(a) and Fig. 4(b)), here the shape of the manifold is better preserved.

Fig. 6 shows two experiments on image data points. In Fig. 6(a) and Fig. 6(b), a face moves on a noised background image from the top-left corner to the bottom-right corner. The data set includes 441 images, each of which includes 116×106 grayscale pixels. The manifold is embedded in $\mathbb{R}^{12296}$. Among these images, 221 images are first learned by SE with $k = 8$. The learned results



(a) GOM (b) CP

Fig. 5. The embedding results by GOM and CP. The original data points are learned by SE, while the M matrix in Eq. (5) for GOM and CP is calculated via LLE.



(a) GOM (b) CP (c) GOM (d) CP

Fig. 6. The embedding results by GOM and CP on two image data sets. The original data points are learned by SE. The M matrix in Eq. (5) is calculated according to SE. Representative images of some new data points are shown at the side of the corresponding circle points.

are shown with (red) filled squares. The rest data points are treated as new data points. From Fig. 6(a) and Fig. 6(b), we can see that they are faithfully embedded into the previously learned structure by GOM and CP. Here, the M matrix in Eq. (5) is calculated according to SE.

In Fig. 6(c) and Fig. 6(d), 400 color images are treated, which are taken from a teapot via different viewpoints aligning in a circle. The size of the images is 76 $\times$ 101. The manifold is embedded in $\mathbb{R}^{23028}$. Among the images, 200 images are first learned by SE with $k = 5$, and the results are illustrated with (red)filled circles. The rest 200 images are treated as new data points. They are embedded into the right positions by GOM and CP, using SE to calculate the M matrix in Eq. (5).

Computation complexity. Both GOM and CP require to calculate the M matrix in Eq. (5). Differently, in GOM we need to solve d QP problems. The computation complexity is $O(d(n-l)^3)$. In CP, we need to perform the singular value decompositions (SVD) of $n-l$ matrices in $\mathbb{R}^{(k+1)\times(k+1)}$ when computing $k+1$ local coordinates in tangent space [4, 6] for each of $n-l$ new data points. We also need to solve $d \times (n-l)$ linear systems formulated as Eq. (14). The computation complexity of SVD is $O((k+1)^3)$, while that of the linear system is near to $O((k+d+1)^2)$ (using Gauss-Seidel iteration). In addition, the computation complexity in Eq. (12) is near to $O(k+d+1)$. Thus, totally the complexity in each a coordinate propagation is about $O((n-l)[(k+1)^3)+$

$(k+d+1)^2 + k + d + 1])$. Compared with $O(d(n-l)^3)$ in GOM method, the computation complexity in CP is only linear to the number of new data points.

In most experiments, the real performance of CP is convergent when iteration counter i is equal to one. That is, we only need to provide $\mathbf{g}_u^{(0)}$ and $\mathbf{g}_u^{(1)}$ once and the convergence is achieved during iterating Eq. (10). This is reasonable since the data points are assumed to be well sampled in manifold learning. Thus we can get a good prediction to each new data point along the manifold via spine interpolation.

5 Related Work on Semi-Supervised Manifold Learning

A parallel work related to out-of-sample extension for manifold learning is the semi-supervised manifold learning [13, 18, 19]. In a semi-supervised framework, the coordinates of some landmark points are provided to constrain the shape to be learned. The landmark points are usually provided according to prior knowledge about the manifold shape or simply given by hand. The goal is to achieve good embedding results via a small number of landmark points. In generally, it is formulated as a transductive learning problem. Intrinsically, the corresponding algorithm runs in a batch mode. In contrast, out-of-sample extension starts with a known manifold shape which is learned from the original data points, and focuses on how to embed the new data points, saying, in a dynamic setting which are collected sequentially. During embedding, the coordinates of previously learned data points can maintain unchanged.

6 Conclusion

We have introduced an approach to out-of-sample extension for NLDR. We developed the global optimization model and gave the coordinate propagation algorithm. Promising experimental results have been presented for 3D surface data and high-dimensional image data, demonstrating that the framework has the potential to embed effectively new data points to the previously learned manifold, and has the potential to use the new points to extend an incomplete manifold shape to a full manifold shape. In the future, we would like to automatically introduce the edge information about the manifold shape from the data points so as to obtain a low-dimensional embedding with better shape-preserving.

Acknowledgements

This work is supported by the Projection (60475001) of the National Nature Science Foundation of China. The anonymous reviewers have helped to improve the representation of this paper.

References

1. Tenenbaum, J.B., de Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science. **290** (2000) 2319-2323
2. Roweis, S.T., Saul, L. K.: Nonlinear dimensionality reduction by locally linear embedding. Science. **290** (2000) 2323-2326
3. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural Computation. **15**(6) (2003), 1373-1396
4. Zhang, Z.Y., Zha, H. Y.: Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. SIAM Journal on Scientific Computing. **26**(1) (2004) 313-338
5. Brand, M.: Charting a manifold. Advances in Neural Information Processing Systems 15. Cambridge, MA: MIT Press, (2003) 985-992
6. Donoho, D.L., Grimes, C. E.: Hessian eigenmaps: locally linear embedding techniques for highdimensional data. Proceedings of the National Academy of Arts and Sciences. **100** (2003) 5591-5596
7. Weinberger, K.Q., Sha, F., Saul, L.K.: Learning a kernel matrix for nonlinear dimensionality reduction. Proceedings of International Conference on Machine learning. Banff, Canada (2004) 888-905
8. Sha, F., Saul L. K.: Analysis and extension of spectral methods for nonlinear dimensionality reduction. International Conference on Machine learning. Bonn, Germany (2005) 784-791
9. Xiang, S.M., Nie, F.P., Zhang, C. S., Zhang, C.X.: Spline embedding for nonlinear dimensionality reduction. European conference on Machine Learning, Berlin, Germany (2006) 825-832
10. Bengio, Y., Paiement, J., Vincent, P.: Out-of-sample extensions for LLE, Isomap, MDS, Eigenmaps and spectral clustering. Advances in Neural Information Processing Systems 16. Cambridge, MA: MIT Press (2004).
11. Law, M., Jain, A.K.: Incremental nonlinear dimensionality reduction by manifold learning. IEEE Transactions on Pattern Analysis and Machine Intelligence. **28**(3) (2006) 377-391
12. Kouropteva, O., Okun, O., Pietikäinen, M.: Incremental locally linear embedding. *Pattern Recognition*, **38**(10) (2005) 1764-1767
13. Yang, X., Fu, H.Y., Zha, H.Y., Barlow, J.: Semi-supervised dimensionality reduction. International Conference on Machine Learning. Pittsburgh, USA. (2006)
14. Min, W., Lu, K., He, X. F.: Locality pursuit embedding. Pattern recognition. **37**(4) (2004) 781-788
15. Duchon, J.: Splines minimizing rotation-invariant semi-norms in Sobolev spaces. In: Schempp, W., Zeller, K. (eds): Constructive Theory of Functions of Several Variables. Berlin: Springer-Verlag. (1977) 85-100
16. Wahba, G.: Spline models for observsatonal data. CBMS-NSF Regional Conference Series in Applied Mathematics. SIAM Press. (1990)
17. Yoon, J.: Spectral approximation orders of radial basis function interpolation on the Sobolev space. SIAM Journal on Mathematical Analysis, **33**(4):946-958
18. Ham, J., Lee, L., Saul, L.: Semisupervised alignment of manifolds International Workshop on Artificial Intelligence and Statistics. Barbados, West Indies (2004) 120-127
19. Gong, H.F., Pan, C.H., Yang, Q., Lu, H.Q., Ma, S.D.: A Semi-supervised framework for mapping data to the intrinsic manifold. International Conference on Computer Vision. Beijing, China (2005) 98-105