Information Sciences 345 (2016) 325–339

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Efficient isometric multi-manifold learning based on the self-organizing method

Mingyu Fan[a,*], Xiaoqin Zhang[a], Hong Qiao[b,c], Bo Zhang[d]

[a] College of Mathematics & Information Science, Wenzhou University, Wenzhou 325035, China
[b] State Key Lab of Management Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
[c] CAS Centre for Excellence in Brain Science and Intelligence Technology (CEBSIT), Shanghai 200031, China
[d] LSEC and Institute of Applied Mathematics, AMSS, Chinese Academy of Sciences, Beijing 100190, China

ARTICLE INFO

Article history:
Received 9 March 2015
Revised 30 October 2015
Accepted 27 January 2016
Available online 4 February 2016

Keywords:
Isomap
Nonlinear dimensionality reduction
Manifold learning
Pattern analysis
Multi-manifold embedding

ABSTRACT

Geodesic distance, as an essential measurement for data similarity, has been successfully used in manifold learning. However, many geodesic based isometric manifold learning algorithms, such as the isometric feature mapping (Isomap) and GeoNLM, fail to work on data that distribute on clusters or multiple manifolds. This limits their applications because practical data sets generally distribute on multiple manifolds. In this paper, we propose a new isometric multi-manifold learning method called Multi-manifold Proximity Embedding (MPE) which can be efficiently optimized using the gradient descent method or the self-organizing method. Compared with the previous methods, the proposed method has two steps which can isometrically learn data distributed on several manifolds and is more accurate in preserving both the intra-manifold and the inter-manifold geodesic distances. The effectiveness of the proposed method in recovering the nonlinear data structure and clustering is demonstrated through experiments on both synthetically and real data sets.

1. Introduction

Challenges, known as “the curse of dimensionality”, are usually confronted when scientists are conducting high dimensional data analysis. Dimensionality reduction is a promising tool to circumvent this problem. Linear dimensionality reduction methods, such as Principal Component Analysis (PCA) and Multidimensional Scaling (MDS), often fail to discover the nonlinear intrinsic structures of data. To address this issue, two Nonlinear Dimensionality Reduction (NLDR) methods, isometric feature mapping (Isomap) [32] and Local Linear Embedding (LLE) [27], were proposed in 2000. Since then, many important NLDR methods have been proposed (see, e.g. [4,31,38] for a good survey of many popular NLDR algorithms including their Matlab codes). NLDR or manifold learning has now become a fast growing research activity and proved very useful in many fields and applications, such as image retrieval [15,36], video annotation [35], clustering [10,39,42], and data visualization [3,12,21,26,34].

Most of the manifold learning algorithms implicitly assume that data points uniformly lie on a single manifold. However, in many practical applications, data points always lie on multiple clusters or multiple manifolds. For instance, in face recognition the face images of each person form its own manifold in the feature space. In motion segmentation in computer vision, moving objects trace different low-dimensional trajectories. In video sequence analysis, frames of different

* Corresponding author. Tel.: +86 013616641250.
E-mail address: fanmingyu@amss.ac.cn, fanmingyu@wzu.edu.cn (M. Fan).

http://dx.doi.org/10.1016/j.ins.2016.01.069
0020-0255/

video shots form different trajectories in the low-dimensional space. Therefore, for many tasks such as multi-class recognition and data visualization, most manifold learning algorithms can only learn each class separately and fail to preserve the inter-class relationship, which restricts their application areas. To overcome this difficulty, many multi-manifold learning algorithms have been proposed, such as linear multiple learning algorithms for data classification [11,16,20], multi-manifold learning algorithms for video analysis [18,19] and for dynamic visual tracking [3,24,40], and multi-class data analysis algorithms [17,37,49]. These works have shown the advantage of multi-manifold learning algorithms over the previous manifold learning algorithms in practical applications. However, it is still a challenging problem to isometrically learn the intrinsic nonlinear data structure for multi-manifold data since it is difficult to faithfully preserve the intra- and inter-manifold distances simultaneously.

In this paper, we propose a Multi-manifold Proximity Embedding (MPE) algorithm to learn multi-manifold data in the unsupervised case. First, we propose two efficient Isometric Proximity Embedding (IPE) algorithms for data distributing on a single manifold, based on the gradient descent method and the self-organizing method. Then, a general framework is introduced for Isometric Multi-Manifold Learning (IMML), based on which the MPE algorithm is proposed for multi-manifold data. MPE is an efficient extension of IPE to multi-manifold data. The proposed MPE has two steps to realize IMML. In the first step, it discovers the data manifolds and then reduce the dimensionality of each manifold separately. Meanwhile, a skeleton representing the global structure of the whole data set is built and then mapped into the low-dimensional space. In the second step, by referring to the low-dimensional representation of the skeleton, the embedding of each manifold is relocated into a global coordinate system. Compared with the previous unsupervised IMML algorithms, our MPE algorithm is not only computationally efficient but also accurately preserves both the intra-manifold and the inter-manifold geodesic distances and therefore is more effective in recovering the nonlinear data structures and clustering.

This paper extends and improves upon our previous work [6] substantially, as indicated below.

(1) A general framework for IMML is introduced in this paper. The framework not only provides the motivation and explanations of the previous methods but can also be regarded as a general platform to design new IMML methods, such as the MPE method in this paper.
(2) With the help of the framework, the MPE method is presented in this paper which is able to preserve the geodesic distance more accurately than our previous method in [6]. Further, the MPE method is based on two efficient optimization algorithms: the gradient descent method and the self-organizing method, so it is computationally efficient and also applicable to large-scale data sets.
(3) More experiments are conducted on both synthetic and image data sets. In particular, clustering results on low-dimensional embeddings of image data are provided to show the effectiveness of the proposed method.

The rest of the paper is organized as follows. In Section 2, previous IMML algorithms are briefly reviewed. In Section 3, a general framework is first introduced for designing IMML algorithms and then used to propose a new IMML algorithm, that is, the MPE algorithm. In Section 4, the effectiveness of the proposed MPE method is demonstrated through experiments on both synthetical and image data sets. Comparisons of our MPE algorithm with several previous IMML algorithms are also made. Concluding remarks are provided in Section 5.

2. Isometric manifold learning and related IMML algorithms

Isomap was first introduced in [32] and is based on the classical MDS method. Its main idea is to preserve the geodesic distance on the data manifold in finding the low-dimensional embedding manifold. Another isometric manifold learning method, namely the GeoNLM [48], has been shown more robust to short circuit edges than Isomap. In these two methods, the geodesic distance between two data points is approximated by the shortest path on a constructed graph using Dijkstra's or Floyd–Warshall's algorithm. If the data points distribute on several clusters or manifolds, neither the k-NN method nor the ε-NN method can construct a good quality neighborhood graph. Therefore, both Isomap and GeoNLM fail to work on this kind of data, which limits their applications. Much work has been done to extend the two methods to multi-manifold learning, which will be briefly reviewed as follows.

Wu and Chan [41] proposed a split-augment approach to construct a neighborhood graph. Their method can be regarded as a variation of the k-NN method, which is simple to implement and has the same computational complexity as the k-NN method. However, since there is only one edge connecting every two graph components, the geodesics across the components are poorly approximated.

Yang [44–47] introduced four methods to construct a connected neighborhood graph: the k minimum spanning trees (k-MST) method [44], the minimum-k-spanning trees (Min-k-ST) method [45], the k-edge-connected (k-EC) method [46] and the k-vertices-connected (k-VC) method [47]. These methods have the following advantages over the k-NN method. First, the local neighborhood relationship is affected by the global distribution of the data points. This is beneficial for adaptively preserving the global geometric metrics. Secondly, these methods can guarantee that the constructed neighborhood graph is totally connected. Thirdly, compared with the k-NN method using the same neighborhood size k, the neighborhood graph constructed by Yang's methods contains more edges. These properties ensure the good quality of the neighborhood graphs. However, these methods are computationally expensive (see [50,51]).

Table 1
Main notations.

k, ε	Parameters used to identify neighborhood, where k is an integer and ε is a positive real number
$\mathcal{X}$	$\mathcal{X} = \{x_1, \ldots, x_N\}$, set of the high-dimensional inputs, with x_i the ith input in $\mathbb{R}^D$, $i = 1, 2, \ldots, N$
$\mathcal{Y}$	$\mathcal{Y} = \{y_1, \ldots, y_N\}$, set of the low-dimensional representations, with y_i the representation in $\mathbb{R}^d$ of x_i after dimensionality reduction, $i = 1, 2, \ldots, N$, $d \ll D$
X	$X = [x_1, \ldots, x_N]$, data matrix of $D \times N$
Y	$Y = [y_1, \ldots, y_N]$, data matrix of $d \times N$
$\mathcal{X}^m$	$\mathcal{X}^m = \{x_1^m, \ldots, x_{l_m}^m\}$, set of the mth data manifold, $m = 1, \ldots, C$, where $x_i^m \in \mathcal{X}$ is the ith data point in the mth data manifold, $i = 1, \ldots, l_m$
$\mathcal{Y}^m$	$\mathcal{Y}^m = \{y_1^m, \ldots, y_{l_m}^m\}$, $m = 1, \ldots, C$, where $y_i^m \in \mathcal{Y}$ is the low-dimensional representation of x_i^m, $i = 1 \ldots, l_m$
$\mathcal{S}_\mathcal{X}$	Set of the selected data points from $\mathcal{X}$ which forms a structure supporting set of $\mathcal{X}$
$\mathcal{S}_\mathcal{X}^m$	$\mathcal{S}_\mathcal{X}^m = \mathcal{S}_\mathcal{X} \cap \mathcal{X}^m$, the structure supporting set of $\mathcal{X}^m$
$\mathcal{S}_\mathcal{Y}$	Set of the low-dimensional representation of $\mathcal{S}_\mathcal{X}$
$x_{n(i)}^m$	$x_{n(i)}^m \in \mathcal{X}^m$, an ending vertex of the ith shortest edge between $\mathcal{X}^m$ and $\mathcal{X}^n$

In order to resolve video analysis problems, such as rush editing and dynamic texture representations, two unsupervised Isomap-based MML algorithms are proposed in [18,19]. These two algorithms apply a new multi-layer graph construction method.

In [23], Meng et al. proposed a decomposition-composition method (D-C Isomap) which extends Isomap to multi-cluster learning. The D-C Isomap algorithm has two steps. First, the algorithm discovers the data clusters and reduces the dimensionality of the clusters separately, and meanwhile, the algorithm builds up a skeleton of the whole data set and preserves the skeleton in a low-dimensional space. Secondly, by referring to the low-dimensional representation of the skeleton, the embeddings of the clusters are relocated to a global coordinate system. However, the data points in the supporting set (the skeleton) possibly lie on a subspace with dimensionality much lower than the dimension of the manifolds. This means that the embedding of each cluster or manifold, whose intrinsic dimensionality is higher than that of the skeleton, cannot be properly relocated by referring to it.

In this paper, we propose an isometric multi-manifold learning framework which is similar to the ideas of split-and-augment used in [41] and decomposition-and-composition used in [23]. In the framework, the intra-manifold and inter-manifold distances are preserved separately and thus the bad geodesic distance approximations (the inter-manifold geodesic distance) do not pollute the good approximations (the intra-manifold geodesic distance) in dimensionality reduction. As a result, the proposed method is more accurate in nonlinear data structure recovery and clustering.

3. Multiple-manifold Proximity Embedding

For convenience a list of main notations used in this paper is given in Table 1. Throughout this paper, all data points are in the form of column vectors. Data sets, as well as data manifolds, are denoted by capital curlicue letters. Matrices are represented by normal capital letters, and data vectors are expressed by lowercase letters.

In Section 3.1, we first introduce an Isometric Proximity Embedding (IPE) method for data distributing on a single manifold, whose objective function resembles with GeoNLM in [48], and then introduce two efficient optimization algorithms for IPE to derive two efficient IPE algorithms: the gradient descent IPE algorithm and the self-organizing IPE algorithm. A general framework for IMML is presented in Section 3.2. Based on the framework and IPE, our MPE method is proposed in Section 3.3. In Section 3.4, the computational complexity of the proposed MPE method is analyzed.

3.1. Isometric Proximity Embedding

The main concern of Isomap is that it needs to eigen-decompose a square similarity matrix of size $N \times N$, which requires $O(N^3)$ computational time. This is generally computational prohibitive for medium or large size data sets whose size $N \geq 5000$. An alternative approach to preserve the geodesic distance is to directly minimize the mismatches between the distance of the output data and the geodesic distance of the input data. An error function known as the stress function has been defined and utilized in [30] to preserve the input distance in the low-dimensional space, which is given as

$$S(Y) = \sum_{i \neq j} \frac{\left(\|y_i - y_j\| - d_G(x_i, x_j)\right)^2}{d_G(x_i, x_j)}, \tag{1}$$

where $d_G(x_i, x_j)$ is the geodesic distance between x_i and x_j. One can find the low-dimensional embedding that best preserves the intrinsic distance by minimizing the stress function. For medium size data sets, the objective function (1) can be efficiently optimized using the gradient descend method. At each iteration, the low-dimensional representation Y can be optimized as

$$Y \leftarrow Y - \lambda \frac{\partial S(Y)}{\partial Y},$$

where λ is a learning rate parameter that decreases during the process and

$$\frac{\partial S(Y)}{\partial Y} = \left[\frac{\partial S(Y)}{\partial y_1}, \ldots, \frac{\partial S(Y)}{\partial y_N} \right], \quad \frac{\partial S(Y)}{\partial y_i} = \sum_{j \neq i} \left(\frac{\|y_i - y_j\| - d_G(x_i, x_j)}{d_G(x_i, x_j)} \cdot \frac{y_i - y_j}{\|y_i - y_j\|} \right), \quad i = 1, \ldots, N.$$

The gradient descent method for IPE is presented in Algorithm 1. As for the computational complexity, Dijkstra's method needs $O(N^2 \log N)$ computational time to compute pairwise graph distances $d_G(x_i, x_j)$ at the first step. The computation time for the graph distances is acceptable since this step only runs once. As can be seen, without the calculation of the graph distances, the computational complexity of the gradient descent method is $O(CN^2)$ since the evaluation of pairwise distance ($\|y_i - y_j\|$) consumes $O(N)$ computational time at each step and $O(N^2)$ computational time at each cycle.

Algorithm 1 IPE: Gradient descent algorithm.

Input: Sparse geodesic distance matrix D_G, initial learning rate λ and decay rate δ
Output: Low-dimensional representations Y
Initialization: Random matrix $Y \in \mathbb{R}^{d \times N}$
for C cycles **do**
  for $i = 1, \ldots, N$ **do**
    $\frac{\partial S(Y)}{\partial y_i} = \sum_{j \neq i} \left(\frac{\|y_i - y_j\| - d_G(x_i, x_j)}{d_G(x_i, x_j)} \cdot \frac{y_i - y_j}{\|y_i - y_j\|} \right)$
    $y_i \leftarrow y_i - \lambda \frac{\partial S(Y)}{\partial y_i}$,
  end for
  Decrease the learning rate $\lambda = \delta\lambda$
end for

In contrast to the gradient descent method, the self-organizing algorithm attempts to bring each cost term $(\|y_i - y_j\| - d_G(x_i, x_j))^2 / d_G(x_i, x_j)$ to zero rapidly at each iteration. Therefore, starting with an initial random configuration, the method iteratively refines the low-dimensional embedding by repeatedly selecting a cost term $(\|y_i - y_j\| - d_G(x_i, x_j))^2 / d_G(x_i, x_j)$ and adjusting the associated low-dimensional coordinates y_i and y_j so that their Euclidean distance matches closely with their input geodesic distance $d_G(x_i, x_j)$. The adjustment is proportional to $\left| \|y_i - y_j\| - d_G(x_i, x_j) \right| / \|y_i - y_j\|$. The self-organizing algorithm of IPE is presented in Algorithm 2. Except for the calculation of the graph distances which needs $O(N^2 \log N)$ computational time, the self-organizing method has $O(kCSN)$ computational complexity since it needs $O(kN)$ computational time at each step and thus consumes $O(kSN)$ computational time at each cycle. This means that the self-organizing IPE algorithm can deal with large-scale datasets.

Algorithm 2 IPE: Self-organizing algorithm.

Input: Sparse geodesic distance matrix D_G, error threshold ε, initial learning rate λ and decay rate δ
Output: Low-dimensional representations Y
Initialization: Random matrix $Y \in \mathbb{R}^{d \times N}$
for C cycles **do**
  for S steps **do**
    for $i = 1, \ldots, N$ **do**
      select randomly a data pair, x_i and x_j for example
      if $\left| \|y_i - y_j\| - d_G(x_i, x_j) \right| > \varepsilon$ **then**
        $y_i \leftarrow y_i - \lambda \frac{1}{2} \frac{\|y_i - y_j\| - d_G(x_i - x_j)}{\|y_i - y_j\| + \varepsilon} (y_i - y_j)$,
        $y_j \leftarrow y_j - \lambda \frac{1}{2} \frac{\|y_i - y_j\| - d_G(x_i - x_j)}{\|y_i - y_j\| + \varepsilon} (y_j - y_i)$,
      end if
    end for
  end for
  Decrease the learning rate $\lambda = \delta\lambda$
end for

To verify the effectiveness of the proposed method, we compare the self-organizing IPE algorithm with Isomap on the benchmark Swissroll data set, which consists of 1800 data points on the manifold and is presented in Fig. 1(a). The neighborhood size for both methods is set as $k = 10$. The convergence curve of the self-organizing IPE algorithm is shown in

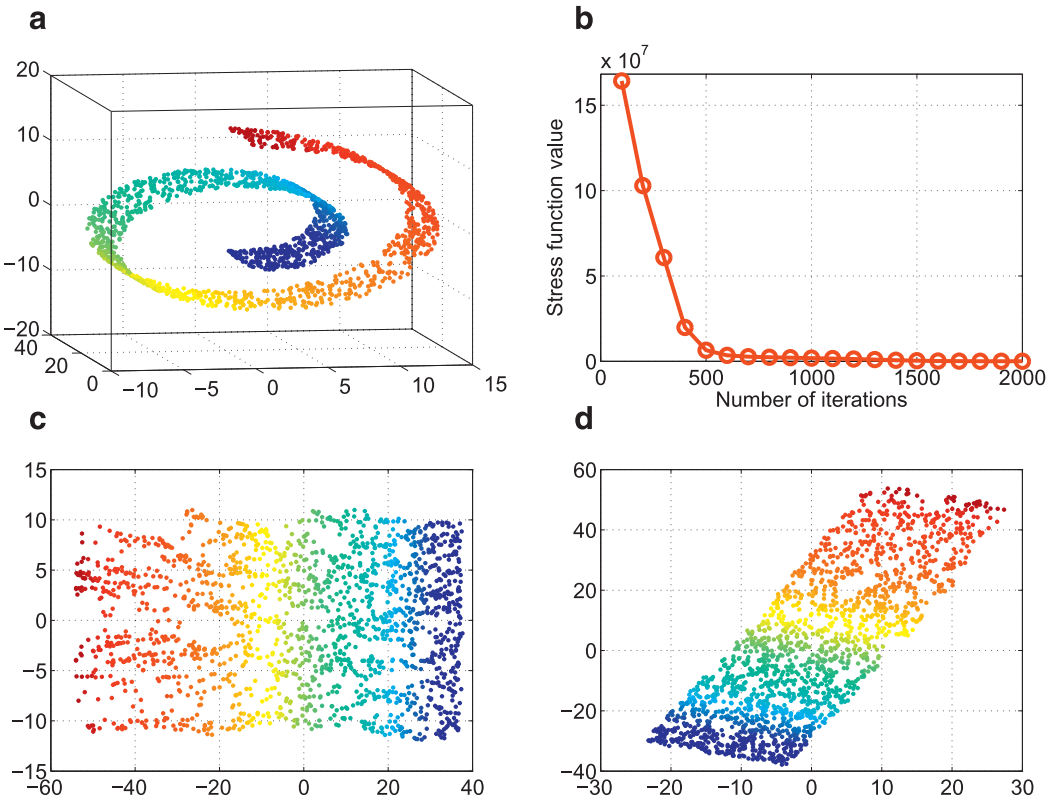

Fig. 1. The self-organizing IPE algorithm and Isomap are applied to the Swissroll data set, which consists of 1800 data points on a single manifold. The neighborhood size for both Isomap and IPE is set as $k = 10$. (a) The 3D scatter plot of the Swissroll data set, (b) the convergence of the self-organizing IPE algorithm with the number of iterations/cycles (C), (c) the low-dimensional embedding obtained by Isomap with residual variance 2.65×10^{-4}, (d) the low-dimensional embedding obtained by the self-organizing IPE algorithm with residual variance 1.98×10^{-4}.

Fig. 1(b). As can be seen, the stress function curve drops to the stable results by less than 1000 iterations/cycles (S is set as 10). The low-dimensional embeddings obtained by Isomap and the self-organizing IPE algorithm are shown in Fig. 1(c) and Fig. 1(d), respectively. The shape of the low-dimensional embedding result obtained by the self-organizing IPE algorithm seems not a rectangle. This is due to the differences of the plotting scales on the x-axle and the y-axle. The numerical results indicate that the self-organizing IPE algorithm gives a better performance (the residual variance is define as in [32], which is 1.98×10^{-4} for the self-organizing IPE algorithm and 2.65×10^{-4} for Isomap) in preserving the geodesic distance. Because of the effectiveness of self-organizing method, it is used as the default method for IPE in the rest of the paper.

3.2. The framework for IMML

Many previous methods extended Isomap to MML by revising the neighborhood graph construction step of the Isomap algorithm [18,19,41,44–47]. However, the shortest paths across clusters or data manifolds are not good approximations to the geodesic distances. Therefore, the preservation of the intra-manifold geodesic distance will be badly affected when the intra-manifold and inter-manifold distances need to be preserved simultaneously [33]. In order to overcome this difficulty, we propose the following general framework for IMML, which not only provides the motivation and explanations of the previous methods but can also be used as a platform to design new IMML methods.

Step I: The decomposition process

1. *Clustering the whole data set* by using clustering methods such as k-NN, ε-NN, K-means, Isodata and other methods introduced in [7,43]. Even if the manifolds overlay with each other, they can still be identified and clustered [13].
2. *Estimating the intrinsic parameters of the data manifolds.* For each data manifold, its intrinsic dimensionality can be estimated by using intrinsic dimension estimation methods, such as the MLE method [22] and the incising ball method [5]. Assume that d_m is the intrinsic dimension of the mth data manifold. Let $d = \max_m d_m$. Then $\mathbb{R}^d$ is the Euclidean space with the lowest dimensionality that can contain all the low-dimensional embeddings of the data manifolds. [29] introduces a method to automatically generate the neighborhood size k or ε for Isomap on a single data manifold. For convenience, the appropriate dimension d and the neighborhood sizes can also be given manually.

3. *Learning the data manifolds individually.* Each data manifold can be learned by IPE, Isomap, or other isometric manifold learning algorithms. However, the low-dimensional embeddings of the data manifolds are centralized to the original point separately. So, in this step, their mutual distances are not kept.

Step II: The composition process

1. *Preserving the supporting structure, $\mathcal{S}_{\mathcal{X}}$, of the whole data set in the low-dimensional space.* The structure supporting set $\mathcal{S}_{\mathcal{X}}$ should be carefully designed so that it can represent the global structure of $\mathcal{X}$. Traditional manifold learning algorithms can be applied to $\mathcal{S}_{\mathcal{X}}$ to get its low-dimensional representation $\mathcal{S}_{\mathcal{Y}}$.
2. *Transforming $\mathcal{Y}^m$s (obtained at Step I.3) into the global coordinate system.* Construct a Euclidean transformation from $\mathcal{Y}^m$ into the coordinate system of $\mathcal{S}_{\mathcal{Y}}$ for $m = 1, \ldots, C$. When $\{\mathcal{Y}^m, m = 1, \ldots, C\}$ are transformed (or relocated), the annexation of the results is the final output.

The idea of a decomposition-composition procedure was first used by Wu and Chan [41] in their split-augment process and then well developed and used in [18,19,23]. However, the above framework aims to solve a more general problem. Step I.1 permits that the designed learning algorithm has a good ability to identify data manifolds. Step I.2 gives a guideline on learning manifolds with different intrinsic dimensions and neighborhood sizes. Step I.3 learns each data manifold individually so that the intra-manifold relationship can be faithfully preserved. Step II.1 is the most flexible part of the procedure which allows us to design new IMML algorithms. A well-designed supporting structure set $\mathcal{S}_{\mathcal{X}}$ can better represent the inter-manifold relationship.

3.3. Multi-manifold Proximity Embedding

Based on the proposed framework, we design a new IMML algorithm which is referred to as Multi-manifold Proximity Embedding (MPE) for multi-manifold data.

3.3.1. Using the k-EG method to construct the neighborhood graph and identify the data manifolds

The variation of the k-NN method, called k-Edge connected Graph (k-EG) method [6], is applied to identify the manifolds and the inter-manifold edges. The k-EG method inherits the computational advantage of the k-NN method and is able to cluster data into manifolds. Besides, it ensures to construct a totally connected neighborhood graph. A brief description of the k-EG method is given as follows, where the output of the k-EG method is the graph $G = (\mathcal{X}, \mathcal{E})$ whose vertices are the data pints in $\mathcal{X}$ and whose edge set is $\mathcal{E}$.

Step 1. Identifying the k nearest-neighbors $\{x_{i1}, \ldots, x_{ik}\}$ of x_i and letting $\mathcal{E} = \mathcal{E} \bigcup \{(x_i, x_{i1}), \ldots, (x_i, x_{ik})\}$, for $i = 1, \ldots, N$. Since the k-NN method cannot guarantee a connected graph, then several disconnected graph components may be obtained at this step.

Step 2. Identifying the graph components by checking the connectivity among the data points in the k-NN graph [2]. Let C be the number of the graph components/manifolds discovered and let $\mathcal{X}^m$ denote the mth graph component/manifold, where $m = 1, \ldots, C$.

Step 3. Identifying the k shortest inter-manifold data pairs, $\{(x^m_{n(i)}, x^n_{m(i)}), i = 1, \ldots, k\}$, between $\mathcal{X}^m$ and $\mathcal{X}^n$, and connecting these data pairs by edges, for $m, n = 1, \ldots, C$. Then the k-EG graph is constructed on the data set $\mathcal{X}$.

By the k-EG method, $\mathcal{X}$ is clustered into C data manifolds $\{\mathcal{X}^m, m = 1, \ldots, C\}$ and the k-shortest edges between $\mathcal{X}^m$ and $\mathcal{X}^n$ are identified as $\{(x^m_{n(i)}, x^n_{m(i)}), i = 1, \ldots, k\}$, where $x^m_{n(i)} \in \mathcal{X}^m$ is an ending vertex of the ith shortest edge between the manifolds $\mathcal{X}^m$ and $\mathcal{X}^n$. The result is a connected graph on the data.

3.3.2. Learning data manifolds individually

The intrinsic dimensionality of each data manifold is set manually as d in our algorithm.

Since a connected graph has been constructed by the k-EG method, one can calculate the graph distance between data points by Floyd's or Dijkstra's algorithm. Let $D_G = (d_G(x_i, x_j))$ denote the $N \times N$ graph distance matrix of $\mathcal{X}$. For each data manifold $\mathcal{X}^m$, the geodesic distance matrix for $\mathcal{X}^m$ can be extracted from the total distance matrix D_G. Assume that $D^m_G = (d_G(x^m_i, x^m_j))$ for the data manifold $\mathcal{X}^m$. Then the low-dimensional embedding $\mathcal{Y}^m$ (before being relocated into a global coordinate system) can be obtained for $\mathcal{X}^m$ by using the IPE method.

3.3.3. Preserving the skeleton of the data manifold $\mathcal{X}$

Similar to M-Isomap [6], the structure supporting set $\mathcal{S}_{\mathcal{X}}$ consists of both the k-nearest and the furthest inter-manifold data points, where the nearest inter-manifold points $\{x^m_{n(i)}, x^n_{m(i)}, i = 1, \ldots, k, m \neq n\}$ are obtained at Step 3 of the k-EG method, and the furthest inter-manifold data points are computed as

$$\{fx^m_n, fx^n_m\} = \arg\max_{i,j} d_G(x^m_i, x^n_j), \quad fx^m_n \in \mathcal{X}^m, \ fx^n_m \in \mathcal{X}^n, \tag{2}$$

where fx_m^n denotes an ending vertex of the furthest inter-manifold data pair between the manifolds $\mathcal{X}^n$ and $\mathcal{X}^m$. Then the structure supporting subset in $\mathcal{X}^m$ is given by

$$\mathcal{S}_{\mathcal{X}}^m = \bigcup_{n=1}^{C} \{x_{n(1)}^m, \ldots, x_{n(k)}^m, fx_n^m\}, \qquad m = 1, \ldots, C.$$

The structure supporting set of $\mathcal{X}$ is thus obtained as $\mathcal{S}_{\mathcal{X}} = \bigcup_{m=1}^{C} \mathcal{S}_{\mathcal{X}}^m$, which is considered as the global skeleton of $\mathcal{X}$.

The reason why we construct the structure supporting set $\mathcal{S}_{\mathcal{X}}$ in this way is that the nearest inter-manifold edges can provide a good measure of the inter-manifold distance and the furthest points fx_n^m with the nearest inter-manifold points $x_{n(1)}^m, \ldots, x_{n(k)}^m$ can be considered as the skeleton of $\mathcal{X}^m$.

On the data manifold $\mathcal{X}$, the skeleton $\mathcal{S}_{\mathcal{X}}$ formulates a sparse graph. Assume that D_S is the graph distance matrix of $\mathcal{S}_{\mathcal{X}}$, which can be extracted from the total graph distance matrix D_G for $\mathcal{X}$. By applying the IPE algorithm on D_S, the low-dimensional representation $\mathcal{S}_{\mathcal{Y}}$ of $\mathcal{S}_{\mathcal{X}}$ can be obtained. It is assumed that $\mathcal{S}_{\mathcal{Y}}^m \subset \mathcal{S}_{\mathcal{Y}}$ is the low-dimensional representation of $\mathcal{S}_{\mathcal{X}}^m$.

3.3.4. Euclidean transformations

We now introduce a method to construct a Euclidean transformation from $\mathcal{Y}^m$ to the coordinate system of $\mathcal{S}_{\mathcal{Y}}$.

The advantage of using Euclidean transformations to relocate the embeddings is that the intra-manifold geodesic distances will be faithfully preserved and the inter-manifold geodestic distances will be kept in the least squares sense. Without loss of generality, the associated low-dimensional representation in $\mathcal{Y}^m$ of $\mathcal{S}_X^m$ is expressed as $\{y_i^m, i = 1, \ldots, s_m\}$ and the associated low-dimensional representation in $\mathcal{S}_{\mathcal{Y}}$ of $\mathcal{S}_X^m$ is expressed as $\{sy_i^m, i = 1, \ldots, s_m\}$, where s_m is the cardinality of $\mathcal{S}_{\mathcal{X}}^m$. Our aim is to find a Euclidean transformation from y_i^m to sy_i^m.

Take the mth data manifold as an example. Assume that the Euclidean transformation from y_i^m to sy_i^m is given by

$$sy_i^m = \mathcal{A}_m y_i^m + \beta_m, \qquad i = 1, \ldots, s_m, \tag{3}$$

where $\mathcal{A}_m$ is an orthonormal matrix and β_m is a translation vector. Note that the Eq. (3) do not always hold because of the inexact approximation and preservation of the inter-manifold geodesic distances. Instead, we minimize the following cost function to find an optimal Euclidean transformation:

$$C(\mathcal{A}_m, \beta_m) = \sum_{i=1}^{s_m} \|\mathcal{A}_m y_i^m + \beta_m - sy_i^m\|^2, \quad \text{s.t.} \quad (\mathcal{A}_m)^T \mathcal{A}_m = I.$$

This problem can be efficiently solved by alternative iterations with respect to $\mathcal{A}_m$ and β_m, and the detailed procedure is summarized in Algorithm 3.

Algorithm 3 Euclidean Transformations.

1: **Input:** $RY^m = [sy_1^m, \ldots, sy_{s_m}^m]$ and $Y_s^m = [y_1^m, \ldots, y_{s_m}^m]$
2: **Output:** $\mathcal{A}_m$, β_m
3: **Initialization:** $\mathcal{A}_m$ and β_m are initialized using the least squares estimator ($\hat{\mathcal{A}}_m$, $\hat{\beta}_m$ in Eq. (5))
4: **while** not converging **do**
5: Fixing β_m, solve the optimization problem

$$\mathcal{A}_m = \arg\min_{\mathcal{A}_m} C(\mathcal{A}_m, \beta_m), \quad \text{s.t.} \quad (\mathcal{A}_m)^T \mathcal{A}_m = I$$

6: Fixing $\mathcal{A}_m$, solve the optimization problem

$$\beta_m = \arg\min_{\beta_m} C(\mathcal{A}_m, \beta_m)$$

7: **end while**

In order to reduce the number of iterations in Algorithm 3, the initial guesses of $\mathcal{A}_m$ and β_m are taken to be their least squares estimators, respectively. Let $RY^m = [sy_1^m, \ldots, sy_{s_m}^m]$ and $Y_s^m = [y_1^m, \ldots, y_{s_m}^m]$. The Euclidean transformation (3) can be rewritten in the matrix form:

$$RY^m = \mathcal{A}_m Y_s^m + \beta_m e^T = (\mathcal{A}_m \quad \beta_m) \begin{pmatrix} Y_s^m \\ e^T \end{pmatrix} \tag{4}$$

where e is a vector with all elements being 1. Eq. (4) can be solved using the least-square method, and the solution is given by

$$(\hat{\mathcal{A}}_m \quad \hat{\beta}_m) = RY^m \begin{pmatrix} Y_s^m \\ e^T \end{pmatrix}^T \left(\begin{pmatrix} Y_s^m \\ e^T \end{pmatrix} \begin{pmatrix} Y_s^m \\ e^T \end{pmatrix}^T + \lambda I \right)^{-1} \tag{5}$$

where I is the identity matrix and λ is a regularization parameter in the singular case ($\lambda = 0.005$ in all experiments). Then the initial guesses of $\mathcal{A}_m$ and β_m are taken to be $\hat{\mathcal{A}}_m$ and $\hat{\beta}_m$. Note that the least-square estimator $\hat{\mathcal{A}}_m$ is generally not an orthonormal matrix.

When β_m is fixed, we need to solve the following constrained least-square problem.

$$\min_{\mathcal{A}_m} \|RY^m - \mathcal{A}_m Y_s^m - \beta_m e^T\|^2, \quad \text{s.t.} \quad (\mathcal{A}_m)^T \mathcal{A}_m = I. \tag{6}$$

It can be seen that

$$\begin{aligned}
\min_{\mathcal{A}_m} \|RY^m - \mathcal{A}_m Y_s^m - \beta_m e^T\|^2 &= \min_{\mathcal{A}_m} \operatorname{tr}\left(RY^m - \mathcal{A}_m Y_s^m - \beta_m e^T\right)^T \left(RY^m - \mathcal{A}_m Y_s^m - \beta_m e^T\right) \\
&= \min_{\mathcal{A}_m} \operatorname{tr}\left(-2\left(RY^m - \beta_m e^T\right)^T \mathcal{A}_m Y_s^m + \left(RY^m - \beta_m e^T\right)^T \left(RY^m - \beta_m e^T\right) + (\mathcal{A}_m Y_s^m)^T (\mathcal{A}_m Y_s^m)\right) \\
&\Leftrightarrow \max_{\mathcal{A}_m} \operatorname{tr}\left(\left(RY^m - \beta_m e^T\right)^T \mathcal{A}_m Y_s^m\right) \\
&= \max_{\mathcal{A}_m} \operatorname{tr}\left(\mathcal{A}_m Y_s^m \left(RY^m - \beta_m e^T\right)^T\right)
\end{aligned}$$

Therefore, finding the solution of (6) is equivalent to solving the following problem:

$$\max_{\mathcal{A}_m} \operatorname{tr}(\mathcal{A}_m \Theta_m), \quad \text{s.t.} \quad (\mathcal{A}_m)^T \mathcal{A}_m = I.$$

where $\Theta_m = Y_s^m \left(RY^m - \beta_m e^T\right)^T$. The solution of the above problem is given as

$$\mathcal{A}_m = U_m (V_m)^T,$$

where U_m and V_m are the left and right singular values of the SVD factorization of Θ_m [8], that is, $\Theta_m = U_m \Gamma_m (V_m)^T$ and the diagonal elements of Γ_m are forced to be nonnegative.

When $\mathcal{A}_m$ is fixed, β_m can be obtained by minimizing the cost functional

$$C(\beta_m) = \sum_{i=1}^{s_m} \|\mathcal{A}_m y_i^m + \beta_m - s y_i^m\|^2.$$

Solving the equation $\partial C(\beta_m)/\partial \beta_m = 0$ gives

$$\beta_m = \frac{1}{s_m} \sum_{i=1}^{s_m} (s y_i^m - \mathcal{A}_m y_i^m)$$

The transformation parameters $\mathcal{A}_m$ and β_m are computed and updated alternatively until convergence or the number of maximum iterations is achieved. As demonstrated in our experiments, usually at most five iterations will be sufficient to get a satisfactory result.

When $\mathcal{A}_m$ and β_m are obtained, the low-dimensional representations $\mathcal{Y}_m$ can be transformed into a global coordinate system of $\mathcal{S}_y$, $m = 1, \ldots, C$. By summarizing the above discussion, the MPE algorithm is given in Algorithm 4.

Algorithm 4 The MPE algorithm.

Input:	$\mathcal{X} = \{x_i, i = 1, \ldots, N\}$, initial neighborhood size k and intrinsic dimensionality d.
Step I.1	*Performing the k-EG algorithm on* $\mathcal{X}$ to get a totally connected graph, with the labels of the data manifolds $\{\mathcal{X}^m, m = 1, \ldots, C\}$ and the set of the inter-manifold edges $\{(x_{n(i)}^m, x_{m(i)}^n, i = 1, \ldots, k\}$ between $\mathcal{X}^m$ and $\mathcal{X}^n$.
Step I.2	*Computing the graph distances on the k-EG graph* by Dijkstra's algorithm. Denote by D_G the graph distance matrix for $\mathcal{X}$. The distance matrix for $\mathcal{X}^m$ can be extracted from D_G.
Step I.3	*Performing IPE on* $\mathcal{X}^m$, $m = 1, \ldots, M$. Apply IPE on the graph distance matrix for $\mathcal{X}^m$ to get the low-dimensional embedding $\mathcal{Y}^m$ of $\mathcal{X}^m$.
Step II.1	*Preserving the skeleton of the data set* $\mathcal{X}$. The furthest inter-manifold points $\{fx_n^m\}_{m \neq n}^M$ can be obtained by solving Eq. (2). Apply IPE on D_S to get the low-dimensional embedding $\mathcal{S}_y$ and hence $\mathcal{S}_y^m$.
Step II.2	*Constructing Euclidean transformations from* $\mathcal{Y}^m$ *to the coordinate system of* $\mathcal{S}_y$, $m = 1, \ldots, C$. Using the Euclidean transformation (cf. Section 3.3.4), $\mathcal{Y}^m$ is transformed into its new location.
Output	$\mathcal{Y} = \bigcup_{m=1}^C \mathcal{Y}^m$ is the final output.

3.4. Computational complexity of the MPE algorithm

In the MPE algorithm based on the self-organizing method, the k-EG method needs $O((k+1)N^2)$ time to construct a totally connected graph and to identify the manifolds. It needs $O(N^2 \ln N)$ time to compute the shortest path on $\mathcal{X}$ and $O(C \times S \times k \sum_{m=1}^{C} l_m) = O(kCSN)$ time to implement IPE on each distance matrix of the data manifolds. The time complexity of finding fx_m^n and fx_n^m is $O(\sum_{m<n}^{C} l_m l_n)$. Performing IPE on D_S needs $O(C \times S \times k \sum_{m=1}^{C} s_m)$ computational time. The time complexity of finding the least-square solution and processing the SVD factorization for C data manifolds is $O(Cd^3)$. Finally, it needs $O(d^2N)$ computational time to transform Y^ms into the global coordinate system.

Thus, the total time complexity of MPE is $O\left((k+1)N^2 + N^2 \ln N + kCS\left(N + \sum_{m=1}^{C} s_m\right) + \sum_{m<n}^{C} l_m l_n + Cd^3 + d^2N\right)$. For a large data set with $N \gg C$ and $N \gg d$, the overall time complexity of MPE is $O\left((k+1+\ln N)N^2 + \sum_{m<n}^{C} l_m l_n + kCSN\right)$.

4. Experiments

In the experiments, comparisons are made among IPE, Isomap [32], Extended Isomap [41], k-MST Isomap [44], D-C Isomap [23], M-Isomap [6], and the proposed MPE method on both synthetic 3D data sets and real world data sets. IPE and Isomap are single manifold learning algorithms and k-MST Isomap, D-C Isomap, M-Isomap and MPE are all IMML methods. For the comparing methods, we use the same neighborhood size k and dimensionality d as input parameters. IPE and Isomap use the classic k-NN graph construction method and therefore cannot guarantee a connected graph on multi-manifold data. Further, IPE and Isomap have to learn each graph component separately. k-MST Isomap utilizes the k-MST graph construction method [44] to build graphs.

4.1. 3-D data sets

To give a quantitative evaluation of the compared algorithms, we use the residual variance as an error measure, which is defined as $1 - R^2(D_G, D)$, where R represents the correlation coefficient, $D_G = \left(d_G(x_i, x_j)\right)$ is a matrix of order N with $d_G(x_i, x_j)$ being the length of the geodesic distance between the data points x_i and x_j and $D = \left(\|y_i - y_j\|\right) \in \mathbb{R}^{N \times N}$ with $\|y_i - y_j\|$ being the Euclidean distances between the projected data points y_i and y_j.

We also define the intra-manifold and inter-manifold residual variances, respectively, as

$$1 - R^2((D_G)_w, D_w), \qquad 1 - R^2((D_G)_b, D_b),$$

which can give a better evaluation on the performance of the algorithms in keeping the intra-manifold and inter-manifold geodesic distances. The matrices $(D_G)_w$, D_w, $(D_G)_b$ and D_b are defined as follows. Let Case1 represent the case where the data points x_i and x_j are in the same data manifold and let Case2 denote the case where the data points x_i and x_j are in different data manifolds. Then we have

$$[(D_G)_w]_{ij} = \begin{cases} d_G(x_i, x_j) & \text{Case1} \\ 0 & \text{Case2} \end{cases}, \quad [D_w]_{ij} = \begin{cases} \|y_i - y_j\| & \text{Case1} \\ 0 & \text{Case2} \end{cases}$$

and

$$[(D_G)_b]_{ij} = \begin{cases} 0 & \text{Case1} \\ d_G(x_i, x_j) & \text{Case2} \end{cases}, \quad [D_b]_{ij} = \begin{cases} 0 & \text{Case1} \\ \|y_i - y_j\| & \text{Case2} \end{cases}.$$

Fig. 2 (a) presents a data set with $N = 1600$ data points lying on a separated Swiss roll manifold. Each data manifold is intrinsically a rectangular region with 800 data points. Fig. 2(b)–(h) present the low-dimensional embeddings obtained by the comparing algorithms. As can be seen, both IPE and Isomap do not consider the inter-manifold relationship so the two embeddings of the rectangles overlap each other. The embeddings obtained by Extended Isomap and MST Isomap bend outwards. D-C Isomap fails to learn the inter-manifold relationship on the data set because it cannot find sufficient reference points to preserve the inter-manifold distances. Both M-Isomap and MPE can faithfully preserve the geodesic distance between the two manifolds.

The intra-manifold residual variances, the inter-manifold residual variances, and the total residual variances of these comparing algorithms on the two-rectangle Swiss roll manifold data set are shown in Fig. 3(a)–(c), respectively. As can be seen, IPE, Isomap, D-C Isomap, M-Isomap and MPE are comparably good in keeping the intra-manifold distances. In Fig. 3(b), M-Isomap and MPE show better results in preserving the inter-manifold distances. In Fig. 3(c), the MPE algorithm has a better performance in keeping both the intra-manifold and the inter-manifold distances. All the comparing algorithms deteriorate when the neighborhood size is larger than 10. This is because the short circuit edges emerge on the graphs of the rectangles which connect intrinsically the far away points using edges directly.

Fig. 4 (a) shows a two-manifold data set with $N = 1600$ data points lying on a sphere and a rectangle. Each data cluster consists of 800 data points. The corresponding low-dimensional embeddings obtained by the comparing algorithms are shown in Fig. 4(b)–(h). Single manifold learning algorithms, IPE and Isomap, still fail to separate the embeddings of the two manifolds. D-C Isomap cannot learn this data set because it cannot find sufficient reference points when the number of the manifolds is 2. The embedding obtained by E-Isomap shrinks badly. The M-Isomap and MPE algorithms give a better performance in preserving both the intra-manifold and the inter-manifold distances.

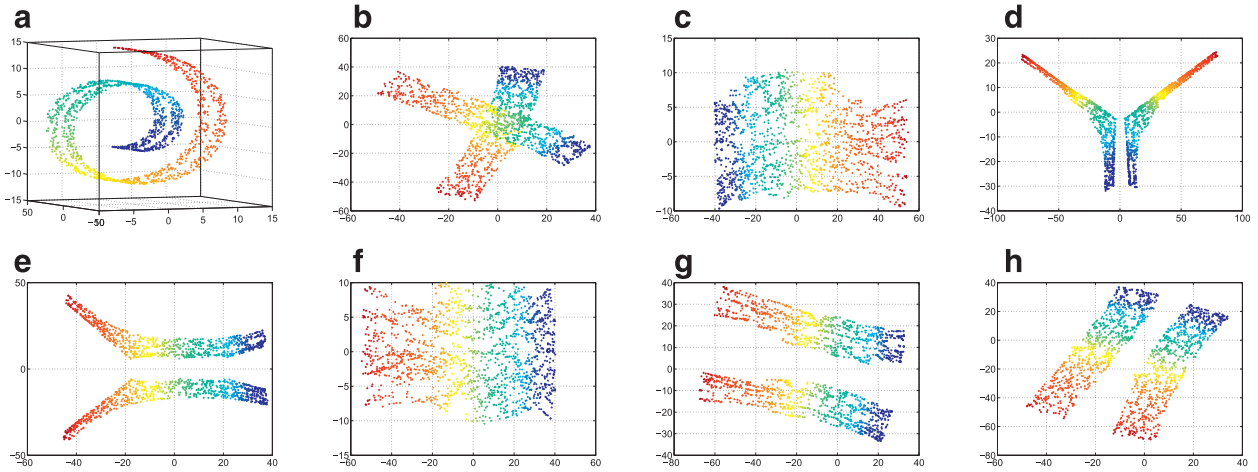

Fig. 2. Experiments on the two-rectangle Swiss roll manifold data set. The neighborhood size $k = 10$ for all the comparing algorithms. (a) 3D scatter plot of the data set. (b) The embedding obtained by IPE. (c) The embedding obtained by Isomap. (d) The embedding obtained by Extended Isomap. (e) The embedding obtained by MST Isomap. (f) The embedding obtained by D-C Isomap. (g) The embedding obtained by M-Isomap. (h) The embedding obtained by MPE.

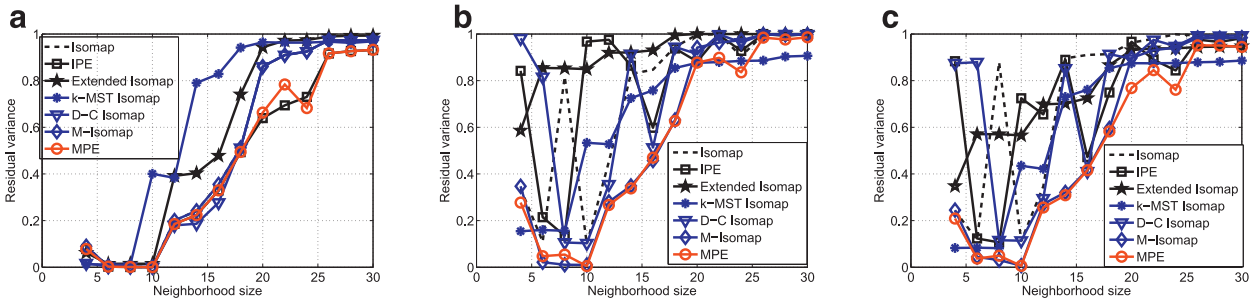

Fig. 3. Residual variances of the comparing algorithms on the two-rectangle Swiss roll manifold data set. (a) The intra-manifold residual variance. (b) The inter-manifold residual variance. (c) The total residual variance.

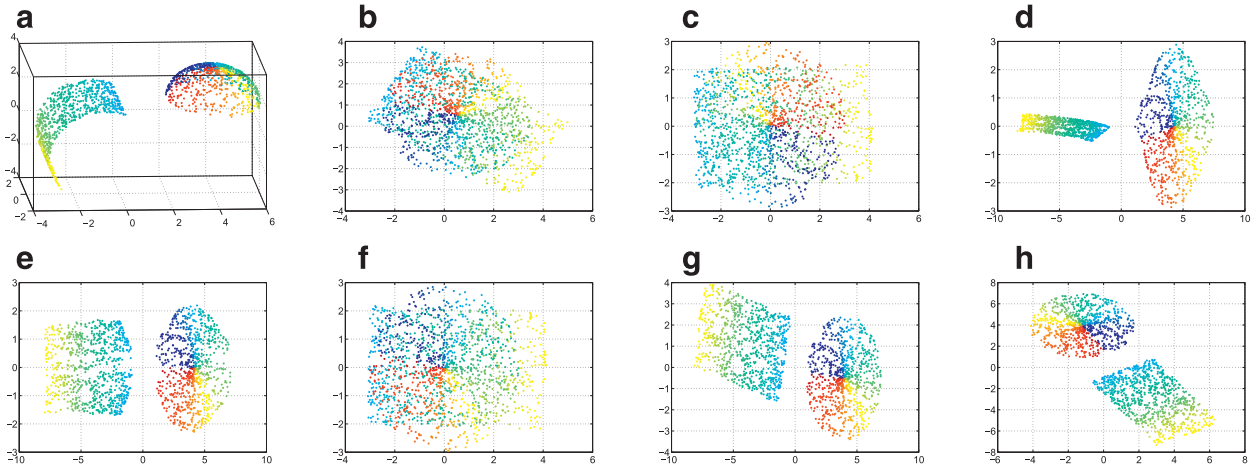

Fig. 4. Experiments on the sphere-and-rectangle manifold data set. The neighborhood size $k = 10$ for all the comparing algorithms. (a) 3D scatter plot of the data set. (b) The embedding obtained by IPE. (c) The embedding obtained by Isomap. (d) The embedding obtained by extended Isomap. (e) The embedding obtained by MST Isomap. (f) The embedding obtained by D-C Isomap. (g) The embedding obtained by M-Isomap. (h) The embedding obtained by MPE.

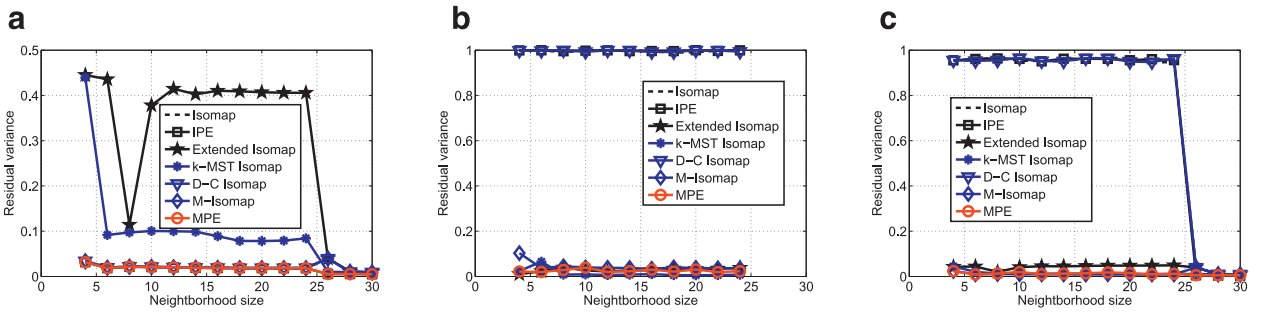

Fig. 5. Residual variances of the compared algorithms on the sphere-and-rectangle manifold data set. (a) The intra-manifold residual variance. (b) The inter-manifold residual variance. (c) The total residual variance.

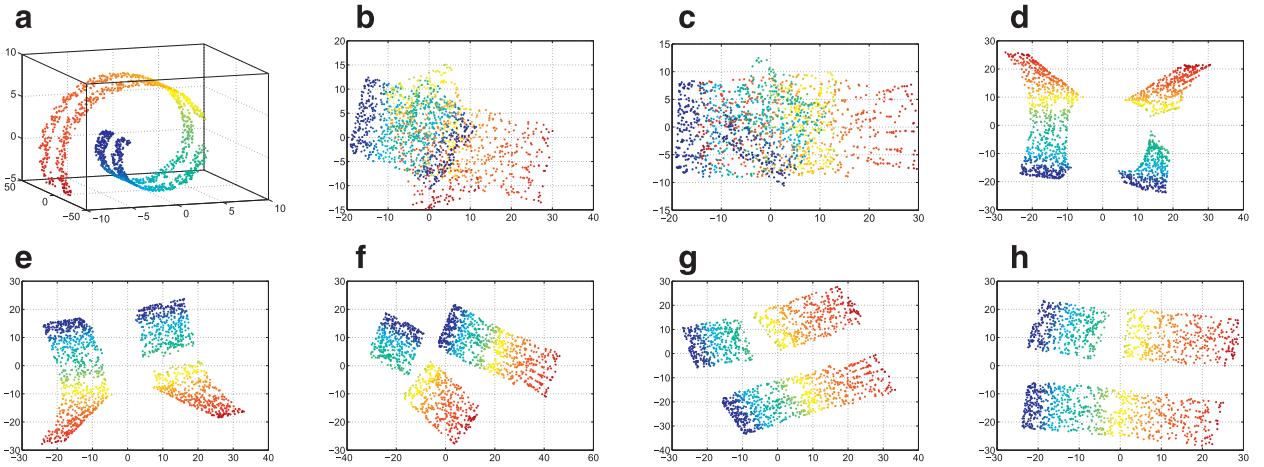

Fig. 6. Experiments on a three-manifold data set, whose data points lie on the Swiss roll manifold. The neighborhood size $k = 10$ for all the comparing algorithms. (a) 3D scatter plot of the data set. (b) The embedding obtained by IPE. (c) The embedding obtained by Isomap. (d) The embedding obtained by Extended Isomap. (e) The embedding obtained by MST Isomap. (f) The embedding obtained by D-C Isomap. (g) The embedding obtained by M-Isomap. (h) The embedding obtained by MPE.

Fig. 5 (a)–(c) present the intra-manifold, the inter-manifold and the total residual variances of these comparing algorithms on the sphere-and-rectangle data set, respectively. As can be seen from Fig. 5(a), E-Isomap and MST Isomap have a bigger error in the intra-manifold distance preservation since they try to preserve the intra-manifold and the inter-manifold distances together in finding the low-dimensional embedding. However, the inter-manifold distances are poorly approximated and should not be mixed up with the intra-manifold distances which are more accurately approximated. The other algorithms, which consider to preserving the intra-manifold and the inter-manifold distances separately, give good results in preserving the intra-manifold distances. Fig. 5(b) shows that IPE, Isomap and D-C Isomap have the highest residual variance curves for the inter-manifold distances. It should be noted that the two manifolds are connected when the neighborhood size k is set larger than 24.

Fig. 6 (a) presents a data set with $N = 1600$ points lying on three rectangles of the Swiss roll manifold. A long rectangle data manifold consists of 800 data points and each of the other two data manifolds consists of 400 data points. Fig. 6(b)–(h) present the corresponding low-dimensional embeddings obtained by the comparing algorithms. As can be seen, the embeddings obtained by E-Isomap and k-MST Isomap bend outward since they preserve the intra-manifold and the inter-manifold distances simultaneously. D-C Isomap can find sufficient reference points on this data set, but the result is not ideal since the distribution of the reference points is uneven. Comparably, M-Isomap and MPE both are able to faithfully preserve both the intra-manifold and the inter-manifold distances.

The intra-manifold residual variances, the inter-manifold residual variances and the total residual variances of these comparing algorithms on the three-manifold data set are presented in Fig. 7(a)–(c), respectively. As can be seen, M-Isomap and MPE have a good performance in the preservation of both the intra-manifold and the inter-manifold distances.

Based on the experiments on these three synthetic 3D data sets, we have several observations which are summarized as follows:

- The IPE and Isomap algorithms learn all data manifolds separately. So the embeddings of the data manifolds are centralized and mixed up with each other.

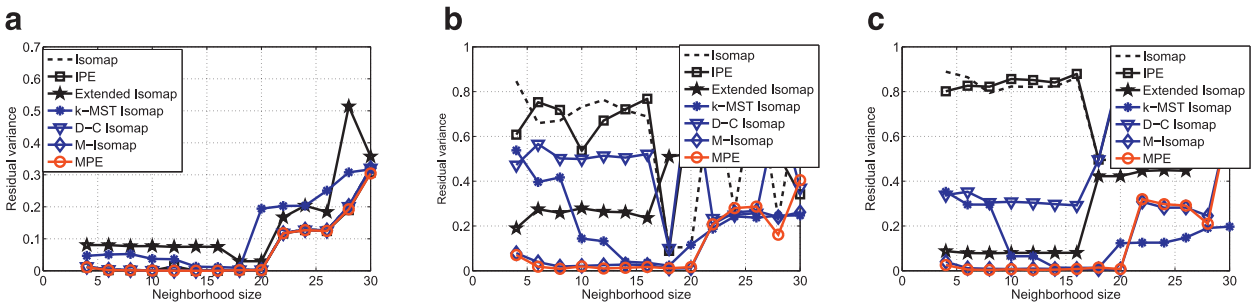

Fig. 7. Residual variances of the compared algorithms on the three-manifold data set. (a) The intra-manifold residual variance. (b) The inter-manifold residual variance. (c) The total residual variance.

- The shape of the embeddings obtained by Extended Isomap and MST-Isomap are always bending outwards. This is because the inter-manifold geodesic distances are poorly approximated, and meanwhile, the algorithms try to keep the good and the bad approximations simultaneously in finding the low-dimensional embeddings.
- Compared with the above algorithms, both M-Isomap and MPE can keep both the intra-manifold and the inter-manifold distances more faithfully when an appropriate neighborhood size is used.

4.2. Real world data sets

In addition to dimensionality reduction, manifold learning has also been considered as a powerful tool for data clustering. In this section, the proposed MPE algorithm is evaluated on image clustering problems for real world data sets.

Three image data sets are used in this paper. The first data set is the Coil20 image library[1] [25], which contains 32 × 32 gray scale images of 20 objects. The second data set is the ORL data set[2] [28] which consists of ten different images of each of 40 distinct persons. For each person, the images were taken at varying times, lighting, facial expression and so on. The third data set is the Extended Yale B face data set[3] [9,14]. It contains 2114 frontal-face images of 30 individuals, and each image is a cropped 64 × 64 gray-scale image under varying lighting conditions. For each individual, there are about 60 images.

We compare the following dimensionality reduction algorithms for data clustering.

- PCA + K-means clustering algorithm (K-means)
- Isomap + K-means
- Extended Isomap + K-means
- *k*-MST Isomap + K-means
- D-C Isomap + K-means
- M-Isomap + K-means
- MPE + K-means

The intrinsic dimensionality d is set to be the same for all the comparing algorithms, and the neighborhood size k for each data set is set as the same for all the manifold learning algorithms. IPE and Isomap give similar clustering results, so the corresponding results are omitted here.

We evaluate the clustering results by comparing the cluster label of each data point with its ground truth label. Two standard clustering metrics, the accuracy (AC) and the normalized mutual information (NMI) metrics [1], are used to measure the clustering performance. For each data set, different ground truth number K of clusters is set for the K-means method (see Tables 2, 3 and 4). And 10 test runs were conducted on different randomly chosen clusters (except for the case when the entire data set is used).

The intrinsic dimensionality d is set as 30, 50 and 40 for the Coil20, ORL and extended YaleB data sets, respectively. Here, the dimensionality is set to be larger than the number of classes of each data set and the same for the comparing algorithms for fair evaluation. The neighborhood size for the comparing manifold learning algorithms is set as 7, 4 and 4 for the data sets, respectively. It is found that a larger neighborhood size leads to a poorer geodesic approximation and thus a lower clustering accuracy. The averaged results and the standard variance of the performance are reported in Tables 2, 3 and 4.

[1] http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php.
[2] http://www.uk.research.att.com/facedatabase.html.
[3] http://vision.ucsd.edu/leekc/ExtYaleDatabase/ExtYaleB.html.

Table 2
Clustering results on the Coil-20 data set, where K denotes the number of clusters.

K	Accuracy (%) + Standard variance (%)						
	PCA	Isomap	E-Isomap	MST Isomap	D-C Isomap	M-Isomap	MPE
2	86.81(2.37)	64.03(2.40)	88.82(2.13)	89.72(2.36)	63.61(2.40)	89.44(2.29)	**93.26(2.40)**
7	69.66(0.35)	58.45(1.19)	79.48(1.71)	73.61(1.42)	57.14(0.53)	72.64(1.06)	**79.94(1.42)**
12	68.63(0.52)	51.46(0.71)	69.69(0.19)	71.12(0.48)	56.04(0.75)	64.28(0.79)	**74.56(0.81)**
17	61.60(0.52)	55.16(0.53)	65.91(0.21)	64.12(0.19)	55.10(0.63)	64.80(0.39)	**72.67(0.21)**
K	Normalized mutual information (%) + Standard variance (%)						
	PCA	Isomap	E-Isomap	MST Isomap	D-C Isomap	M-Isomap	MPE
2	62.21(10.6)	17.36(6.34)	66.19(9.20)	70.95(11.3)	16.92(6.34)	69.46(10.8)	**82.88(10.87)**
7	71.51(0.71)	56.56(1.00)	77.65(1.17)	72.99(1.06)	56.53(0.62)	73.45(0.66)	**79.17(0.89)**
12	75.07(0.40)	58.26(1.01)	77.10(0.14)	76.33(0.27)	58.26(0.85)	72.60(0.41)	**81.11(0.40)**
17	72.93(0.04)	66.80(0.41)	76.58(0.08)	74.38(0.04)	65.42(0.46)	75.21(0.13)	**79.93(0.07)**

Table 3
Clustering results on the ORL data set, where K denotes the number of clusters.

K	Accuracy (%) + Standard variance (%)						
	PCA	Isomap	E-Isomap	MST Isomap	D-C Isomap	M-Isomap	MPE
5	80.00(1.30)	70.00(2.44)	80.00(1.46)	84.40(1.07)	69.20(1.81)	84.40(1.53)	**84.50(1.30)**
15	49.20(0.15)	56.13(0.46)	60.40(0.27)	54.93(0.24)	54.13(0.21)	59.73(0.68)	**64.67(0.28)**
25	49.12(0.08)	51.60(0.04)	53.12(0.03)	50.88(0.01)	53.76(0.13)	56.40(0.05)	**60.64(0.08)**
40	42.25(0.04)	47.70(0.09)	50.10(0.02)	42.00(0.04)	48.50(0.02)	49.00(0.03)	**54.70(0.05)**
K	Normalized mutual information (%) + Standard variance (%)						
	PCA	Isomap	E-Isomap	MST Isomap	D-C Isomap	M-Isomap	MPE
5	77.55(1.00)	62.91(3.62)	79.84(0.83)	80.97(1.09)	63.77(3.30)	79.69(1.65)	**82.42(0.70)**
15	63.82(0.06)	64.66(0.42)	69.49(0.36)	65.58(0.28)	62.69(0.26)	69.53(0.44)	**73.63(0.41)**
25	66.27(0.03)	66.25(0.05)	68.38(0.02)	66.60(0.01)	68.16(0.12)	69.53(0.03)	**73.58(0.05)**
40	65.49(0.01)	67.37(0.01)	69.30(0.00)	64.46(0.01)	67.86(0.01)	69.05(0.01)	**71.40(0.01)**

Table 4
Clustering results on the Yale B extended data set, where K denotes the number of clusters.

K	Accuracy (%) + Standard variance (%)						
	PCA	Isomap	E-Isomap	MST Isomap	D-C Isomap	M-Isomap	MPE
2	50.23(0.00)	100(0.00)	100(0.00)	52.58(0.01)	100(0.00)	99.92(0.00)	100(0.00)
3	35.83(0.00)	78.97(4.47)	80.75(3.40)	46.64(1.28)	77.51(5.60)	80.75(3.40)	**81.91(3.17)**
10	13.66(0.00)	47.91(0.89)	47.27(0.26)	33.47(0.14)	45.49(0.42)	46.17(0.31)	**49.31(0.78)**
20	10.25(0.01)	40.84(0.08)	41.44(0.06)	32.25(0.09)	41.74(0.12)	41.18(0.09)	**42.34(0.06)**
K	Normalized mutual information (%) + Standard variance (%)						
	PCA	Isomap	E-Isomap	MST Isomap	D-C Isomap	M-Isomap	MPE
2	0.01(0.00)	100(0.00)	100(0.00)	0.27(0.00)	100(0.00)	99.42(0.03)	100(0.00)
3	0.27(3.17)	64.73(10.7)	68.21(7.55)	16.73(4.18)	63.50(12.13)	68.21(7.55)	**69.50(9.88)**
10	1.75(0.78)	52.29(0.43)	51.86(0.19)	37.61(0.44)	50.84(0.32)	50.94(0.30)	**52.83(0.53)**
20	5.97(0.03)	55.97(0.02)	55.74(0.00)	47.90(0.12)	55.74(0.03)	54.86(0.06)	**56.16(0.05)**

From these experimental results we have the following interesting observations.

- The inter-manifold distance is important for recovering the intrinsic data structure. Because of this, single manifold learning algorithms fail to preserve the intrinsic inter-manifold distance of data and give an inferior performance than PCA on the Coil20 and ORL data sets.
- IMML methods, which consider to preserve the inter-manifold distance, outperform PCA and the single manifold learning algorithms. As can be seen from Tables 2 and 3, E-Isomap, MST Isomap, M-Isomap and MPE outperform PCA and Isomap in data clustering.
- Compared with M-Isomap, the MPE algorithm based on the stress function can better preserve the inter-manifold distance and is efficient. Therefore, it shows a better performance in data clustering on the evaluated data sets.

For the Extended YaleB data set, the clustering result of PCA and MST Isomap is almost equivalent to random guess. In order to see the reason why these two methods fail, we randomly select two classes and present the two-dimensional

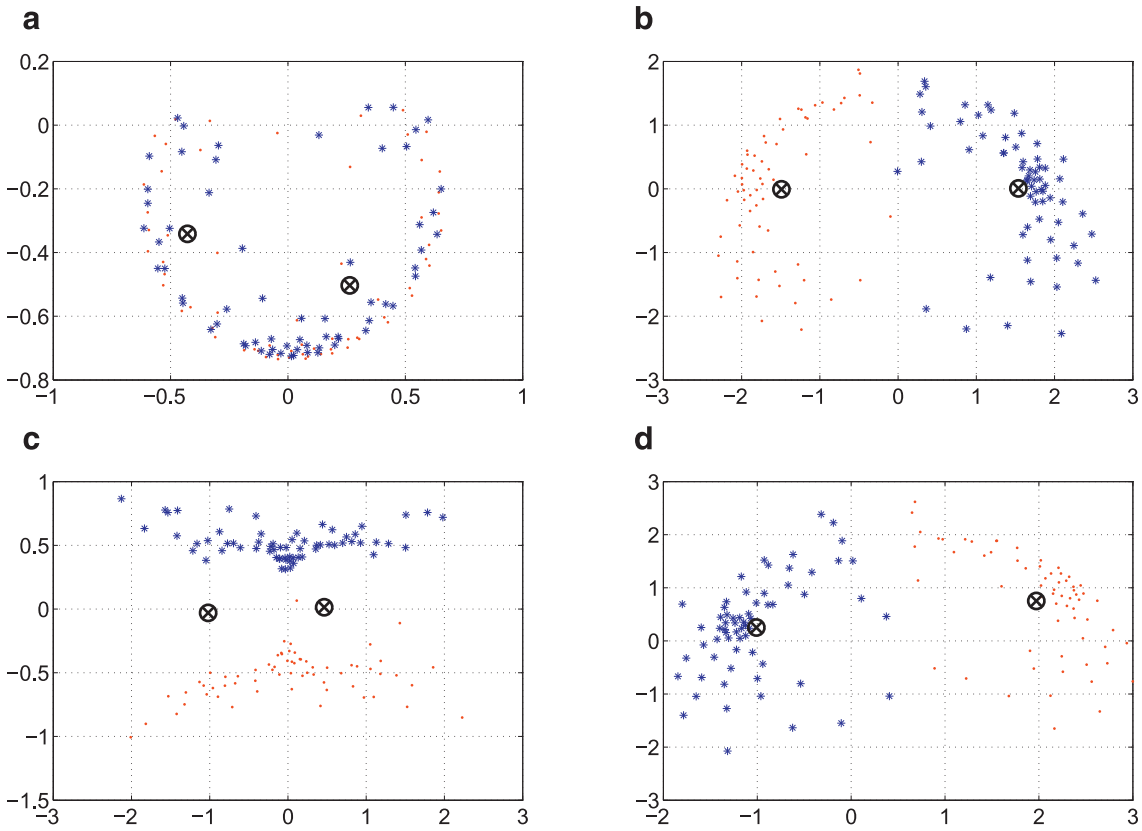

Fig. 8. The two-dimensional embeddings of randomly selected two classes of the extended YaleB data set by (a) PCA, (b) Extended Isomap, (c) MST Isomap, and (d) MPE methods.

embeddings obtained by PCA, E-Isomap, MST Isomap and MPE. The corresponding cluster centers are also shown in the sub-parts of Fig. 8. Points from different classes are represented by blue stars and red dots, respectively. As can be seen, PCA cannot discover the nonlinear structure of the data, so the points are inseparable. The points on the embeddings obtained by all manifold learning algorithms are visually separable. As can be seen, the cluster centers by K-means on the low-dimensional embedding obtained by MST Isomap are mis-located. Compared with E-Isomap, MST Isomap provides more inter-manifold edges, so the intra-manifold distance is more seriously polluted by the bad approximations (the inter-manifold distance) when MST Isomap tries to mix the two kinds of distances and preserve them simultaneously in obtaining the low-dimensional embedding. MPE keeps the intra-manifold and the inter-manifold distances separately, so the intra-manifold distance is not polluted at all.

5. Conclusion

In this paper, we introduced a general framework for isometric multi-manifold learning. The framework can be used to design new MML algorithms which are able to faithfully preserve not only the intra-manifold geodesic distances but also the inter-manifold geodesic distances. Based on the framework, the multi-manifold learning method called MPE was proposed which gives promising results in both recovering the nonlinear data structure and data clustering. Compared with M-Isomap which was introduced in our previous work, MPE is computationally more efficient and more accurate in preserving the geodesic distances. Experiments on both synthetic 3D multi-manifold data sets and images data sets have been conducted to illustrate the efficiency of the proposed MPE algorithm.

Acknowledgment

This work was partly done when the first author (MF) visited National Center for Mathematics and Interdisciplinary Sciences, Chinese Academy of Sciences, Beijing, and he thanks NCMIS for the financial support. The first and second authors (MF and XZ) were partly supported by the NNSF of China Grant (nos. 61203241, 61473212, 61472285, 6141101224, and 61305035) and NSF of Zhejiang Province under LY15F030011. The third author was partly supported by the Strategic Priority Research Program of the CAS (Grant XDB02080003). The fourth author (BZ) was partly supported by the NNSF of

China Grant (nos. 61379093 and 11131006). The authors thank the anonymous reviewers for their invaluable comments and suggestions which helped improve the presentation of the paper.

References

[1] D. Cai, X. He, J. Han, Document clustering using locality preserving indexing, IEEE Trans. Knowl. Data Eng. 17 (12) (2005) 1624–1637.
[2] T.H. Cormen, C. Stein, R.L. Rivest, C.E. Leiserson, Introduction to Algorithms, second ed., McGraw-Hill Higher Education, 2001.
[3] A. Elgammal, C.S. Lee, Tracking people on a torus, IEEE Trans. Pattern Anal. Mach. Intell. 31 (3) (2009) 520–538.
[4] M. Fan, N. Gu, H. Qiao, B. Zhang, Dimensionality reduction: An interpretation from manifold regularization perspective, Inf. Sci. 277 (2014) 694–714.
[5] M. Fan, H. Qiao, B. Zhang, Intrinsic dimension estimation of manifolds by incising balls, Pattern Recognit. 42 (5) (2009) 780–787.
[6] M. Fan, H. Qiao, Z. Zhang, X. Zhang, Isometric multi-manifold learning for feature extraction, in: Proceedings of the Twelfth IEEE International Conference on Data Mining (ICDM), December 2012, pp. 241–250.
[7] M. Filippone, F. Camastra, F. Masulli, S. Rovetta, A survey of kernel and spectral methods for clustering, Pattern Recognit. 41 (2007) 176–190.
[8] G.H. Golub, C.F.V. Loan, Matrix Computations, third ed., Johns Hopkins, 1996.
[9] A.S. Georghiades, P.N. Belhumeur, D.J. Kriegman, From few to many: illumination cone models for face recognition under variable lighting and pose, IEEE Trans. Pattern Anal. Mach. Intell. 23 (6) (2001) 643C660.
[10] D. Gong, X. Zhao, G.G. Medioni, Robust multiple manifold structure learning, in: Proceedings of the Twenty-ninth International Conference on Machine Learning (ICML 2012), Edinburgh, Scotland, June 2012.
[11] A.B. Goldberg, X. Zhu, A. Singh, Z. Xu, R. Nowak, Multi-manifold semi-supervised learning, J. Mach. Learn. Res. Workshop Conf. Proc. 5 (2009) 169–176.
[12] G. Hinton, S. Roweis, Stochastic neighbor embedding, Neural Information Processing Systems, vol. 15, MIT Press, Cambridge, MA, 2002, pp. 833–840.
[13] D. Kushnir, M. Galun, A. Brandt, Fast multiscale clustering and manifold identification, Pattern Recognit. 28 (10) (2006) 1876–1891.
[14] K.C. Lee, J. Ho, D. Kriegman, Acquiring linear subspaces for face recognition under variable lighting, IEEE Trans. Pattern Anal. Mach. Intell. 27 (5) (2005) 684–698.
[15] P. Li, M. Wang, J. Cheng, C. Xu, H. Lu, Spectral hashing with semantically consistent graph for image indexing, IEEE Trans. Multimed. 15 (1) (2013) 141–152.
[16] B. Li, D. Huang, C. Wang, K. Liu, Feature extraction using constrained maximum variance mapping, Pattern Recognit. 41 (2008) 3287–3294.
[17] M. Li, J. Xu, J. Yang, D. Yang, D. Wang, Multiple manifolds analysis and its application to fault diagnosis, Mech. Syst. Signal Process. 23 (8) (2006) 2500–2509.
[18] Y. Liu, Y. Liu, K.C.C. Chan, Dimensionality reduction for heterogeneous dataset in rushes editing, Pattern Recognit. 42 (2) (2009) 229–242.
[19] Y. Liu, Y. Liu, K.C.C. Chan, Nonlinear dimensionality reduction with hybrid distance for trajectory representation of dynamic texture, Signal Process. 90 (8) (2010) 2375–2395.
[20] J. Lu, Y.P. Tan, G. Wang, Discriminative multi-manifold analysis for face recognition from a single training sample per person, in: Proceedings of the IEEE International Conference on Computer Vision, 2011, pp. 1943–1950.
[21] L.V. de Maaten, G. Hinton, Visualizing data using t-SNE, J. Mach. Learn. Res. 9 (2008) 2579–2605.
[22] E. Levina, P.J. Bickel, Maximum Likelihood Estimation of Intrinsic Dimension, in: Advances in Neural Information Processing Systems, 2005, pp. 777–784.
[23] D. Meng, Y. Leung, T. Fung, Z. Xu, Nonlinear dimensionality reduction of data lying on the multicluster manifold, IEEE Trans. Syst. Man Cybern. B 38 (4) (2008) 1111–1122.
[24] J. Nascimento, J.G. Silva, Manifold learning for object tracking with multiple motion dynamics, in: Proceedings of the Eleventh European Conference on Computer Vision, 2010, pp. 172–185.
[25] S.A. Nene, S.K. Nayar, H. Murase, Columbia Object Image Library (COIL-20), Technical Report CUCS-005-96., February 1996. http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php.
[26] H. Qiao, P. Zhang, D. Wang, B. Zhang, An explicit nonlinear mapping for manifold learning, IEEE Trans. Cybern. 43 (1) (2013) 51–63.
[27] S.T. Roweis, L.K. Saul, Nonlinear dimensionality reduction by local linear embedding, Science 290 (2000) 2323–2326.
[28] F.S. Samaria, A.C. Harter, Parameterisation of a stochastic model for human face identiffication, in: Proceedings of the Second IEEE Workshop on Applications of Computer Vision, 1994, pp. 138–142.
[29] O. Samko, A.D. Marshall, P.L. Rosin, Selection of the optimal parameter value for the Isomap algorithm, Pattern Recognit. Lett. 27 (9) (2006) 968–979.
[30] J.W. Sammon, A nonlinear mapping for data structure analysis, IEEE Trans. Comput. 18 (5) (1969) 401–409.
[31] H. Strange, R. Zwiggelaar, Open Problems in Spectral Dimensionality Reduction, Springer, 2014.
[32] J.B. Tenenbaum, V. de Sliva, J.C. Landford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (2000) 2319–2323.
[33] J.B. Tenenbaum, V. de Sliva, J.C. Landford, Response to comments on the Isomap algorithm and topological stability, Sciences 295 (5552) (2002) 7.
[34] J. Venna, S. Kaski, Local multidimensional scaling, Neural Netw. 19 (6) (2006) 889–899.
[35] M. Wang, X. Hua, R. Hong, J. Tang, G. Qi, Y. Song, Unified video annotation via multigraph learning, IEEE Trans. Circuits Syst. Video Technol. 19 (5) (2009) 733–746.
[36] M. Wang, H. Li, D. Tao, K. Lu, X. Wu, Multi-modal graph-based reranking for web image search, IEEE Trans. Image Process. 21 (11) (2012) 4649–4661.
[37] X. Wang, P. Tino, M.A. Fardal, Multiple manifolds learning framework based on hierarchical mixture density model, Mach. Learn. Knowl. Discov. Databases 5212 (2008) 566–581.
[38] J. Wang, Geometric structure of high-dimensional data and dimensionality reduction, High Education Press, Beijing & Springer, New York, 2012.
[39] X. Wang, K. Slavakis, G. Lerman, Multi-manifold modeling in non-euclidean spaces, in: Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, 2015, pp. 1023–1032.
[40] L. Wang, D. Suter, Visual learning and recognition of sequential data manifolds with applications to human movement analysis, Comput. Vis. Image Underst. 110 (2) (2008) 153–172.
[41] Y. Wu, K.L. Chan, An extended Isomap algorithm for learning multi-class manifold, in: Proceedings of the 2004 International Conference on Machine Learning and Cybernetics, vol. 6, 2004, pp. 3429–3433.
[42] B. Xie, M. Wang, D. Tao, Toward the optimization of normalized graph Laplacian, IEEE Trans. Neural Netw. 22 (4) (2011) 660–666.
[43] R. Xu, D. Wunsch II, Survey of clustering algorithms, IEEE Trans. Neural Netw. 16 (3) (2005) 645–678.
[44] L. Yang, K-edge connected neighborhood graph for geodesic distance estimation and nonlinear projection, in: Proceedings of the Seventeenth International Conference on Pattern Recognition (ICPR'04), vol. 1, 2004, pp. 196–199.
[45] L. Yang, Building k edge-disjoint spanning trees of minimum total length for isometric data embedding, IEEE Trans. Pattern Anal. Mach. Intell. 27 (10) (2005) 1680–1683.
[46] L. Yang, Building k edge-connected neighborhood graph for distance-based data projection, Pattern Recognit. Lett. 26 (13) (2005) 2015–2021.
[47] L. Yang, Building k-connected neighborhood graphs for isometric data embedding, IEEE Trans. Pattern Anal. Mach. Intell. 28 (5) (2006) 827–831.
[48] L. Yang, Sammon's nonlinear mapping using geodesic distances, in: Proceedings of the Seventeenth InternationalConference on Pattern Recognition, vol. 2, 2004, pp. 303–306. 23–26.
[49] H. Yin, S.M. Zaki, A self-organizing multi-manifold learning algorithm, Lect. Notes Comput. Sci. 9108 (2015) 389–398.
[50] D. Zhao, L. Yang, Incremental construction of neighborhood graphs for nonlinear dimensionality reduction, Proc. Int. Conf. Pattern Recognit. 28 (5) (2006) 827–831.
[51] D. Zhao, L. Yang, Incremental isometric embedding of high-dimensional data using connected neighborhood graphs, IEEE Trans. Pattern Anal. Mach. Intell. 31 (1) (2009) 86–98.