Neurocomputing 336 (2019) 46–55

Contents lists available at ScienceDirect

Neurocomputing

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

A preliminary geometric structure simplification for Principal Component Analysis

Huamao Gu[a], Tong Lin[b], Xun Wang[a,*]

[a] School of Computer Science and Information Engineering, Zhejiang Gongshang University, Hangzhou 310018, China
[b] State Key Laboratory of Machine Perception, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China

ARTICLE INFO

Article history:
Received 13 July 2017
Revised 7 May 2018
Accepted 22 May 2018
Available online 2 November 2018

Keywords:
Data preprocessing
PCA
Geometric structure

ABSTRACT

Real world data are commonly geometrically nonlinear and thus are not easy to be processed by the traditional linear methods. Many existing techniques for nonlinear dimensionality reduction need careful parameter tuning and cannot be applied to real data stably and consistently. In this article we propose an efficient data preprocessing algorithm, called Curve Straightening Transformation (CST), to flatten the nonlinear geometric structure of data. Then Principal Component Analysis (PCA) and other linear projection methods are adequate to perform the dimensionality reduction task in most cases. In this aspect, the proposed CST algorithm can be regarded as a geometric preprocessing step tailored for PCA. The comprehensive experiments on both artificial and real datasets demonstrate that the proposed preprocessing algorithm is able to simplify the nonlinear geometric structures, and the flattened data are suitable for further dimensionality reduction by linear methods such as PCA.

1. Introduction

The processing of high-dimensional data has been a key issue that can arise in almost every field. Aside from the general difficulties lying in both low-dimensional and high-dimensional data, such as noise [1], missing values [2], and inconsistency [3], the complexity of high-dimensional data mainly stems from two aspects. One is the high dimensionality which may incur heavy computational burdens. Another is the nonlinear geometric structure lying in the high dimensional space, which may hinder the discovery of useful rules to describe the patterns for later processing. These two complexities often mix together and thus increase the difficulties of developing efficient data analysis methods.

Most currently available solutions are nonlinear dimensionality reduction techniques, called manifold learning, to tackle these two complexities as a whole. However, the performances of these manifold learning methods are far from satisfactory. One explanation is that these methods often rely on rather strong assumptions in the input data. For instance, the Isomap [4] algorithm requires that the data manifold is connected and convex. Other locality preserving methods, including Maximum Variance Unfolding (MVU) [5], Locally Linear Embedding (LLE) [6], Laplacian Eigenmaps (LE) [7], Local Tangent Space Alignment (LTSA) [8], and Riemannian Manifold Learning (RML) [9], often impose strong requirements on the density of the input data. Some experiments [10] have already confirmed the unstability of these algorithms.

Another difficulty to use these nonlinear methods is parameter tuning. For example, in kernel based techniques like Kernel PCA (KPCA) [11] and Diffusion Maps (DM) [12], the best kernel function usually depends on the data in hand and there lacks efficient research to find the optimal kernels. On the other hand, the techniques based on neural networks (such as Autoencoder [13]) have lots of parameters to be finely tuned; and the parameter tuning often requires a painstaking examination by human, and is usually time-consuming.

To circumvent these difficulties, we propose a new idea from the aspect of data preprocessing to decrease the data nonlinearity by geometric transformations. Fig. 1 illustrates this preprocessing idea on the Swiss Roll data. The nonlinear geometric structure of the example dataset becomes simpler after the proposed geometric transformation, as shown at Fig. 1. In this way, one can use the simple but efficient Principal Component Analysis (PCA) [14] for feature extraction in most cases, thus avoiding the selection of complicated manifold learning methods and excluding the tunning of parameters in the existing nonlinear methods.

2. The proposed algorithm

We firstly introduce the basic principle of the proposed algorithm to show how it works and why it is effective. Then, we give

* Corresponding author.
E-mail addresses: lintong@pku.edu.cn (T. Lin), wx@mail.zjgsu.edu.cn (X. Wang).

https://doi.org/10.1016/j.neucom.2018.05.119
0925-2312/© 2018 Elsevier B.V. All rights reserved.

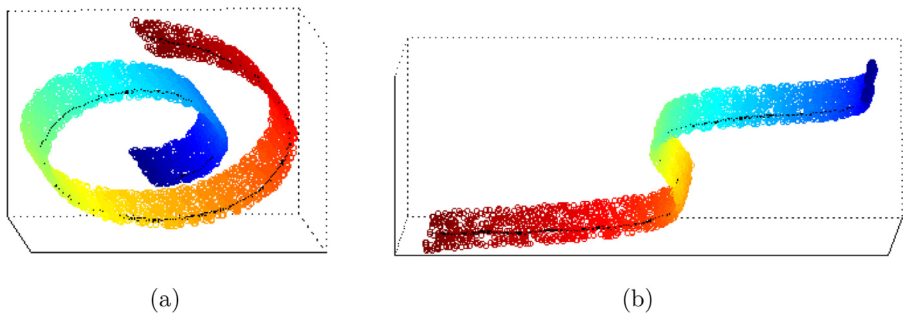

Fig. 1. The original (a) and preprocessed (b) Swiss roll data.

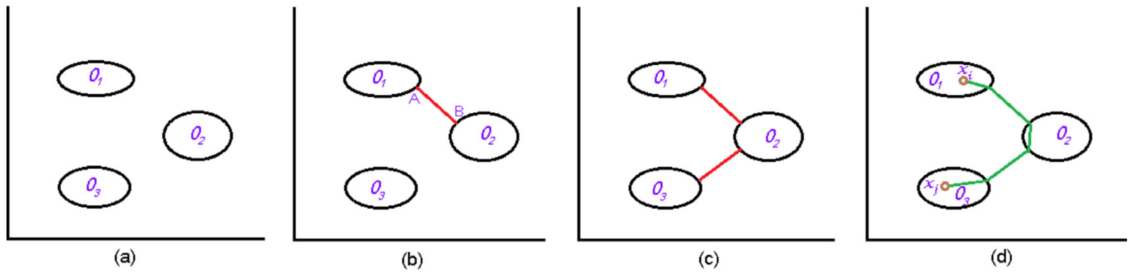

Fig. 2. The conceptual examples for discrete region (a), bridge (b), shortest connected region (c) and connection path (d).

two geometric facts that we apply to the implementation of our algorithm. And we present the detailed implementation at the last subsection.

2.1. Basic principle

This paper aims at proposing a novel idea to simplify the geometric structure of data into a much simpler and less nonlinear structure. Followed by linear dimensionality reduction methods like PCA, the simplified structure can be used for clustering, classification, and other tasks. The preprocessing algorithm is named as Curve Straightening Transformation (CST). To better describe the CST, let us introduce four definitions as follows. The corresponding examples are shown in Fig. 2.

Definition 1 (Discrete region)**.** In a n dimensional Euclidean space, a set $X = \bigcup_{i=1}^{c} O_i$ is called a *discrete region* if each $O_i \subset \mathbb{R}^n$ is a connected subset and $O_i \cap O_j = \varnothing$ $(1 \le i, j \le c)$. And each O_i is called a *primitive subregion* accordingly.

Definition 2 (Bridge)**.** The *bridge* between two primitive subregions O_i and O_j is the line segment $\overline{AB}$ $(A \in O_i, B \in O_j)$ which is the shortest edge connecting two subregions.

Definition 3 (Shortest connected region)**.** If a discrete region is fully connected by $c - 1$ bridges and the sum of the lengths of these bridges reaches minimum, then it is called a *shortest connected region*.

Definition 4 (Connection path)**.** The shortest curve connecting two points in the shortest connected region is called a *connection path*. And its length is called the *connection distance* accordingly.

As a connection path is the shortest path running through the data region between two given points, it serves as an outline to represent the geometric structure (curvedness or straightness) of the underlying region. A short connection path reflects the geometric structure of a local region, while a long connection path provides a signature of the principal geometric structure of the global region. We call such a long connection path the *principal connection path*.

With the aid of the above concepts, the proposed CST algorithm is composed of following steps. (1) The nonlinear geometric structure of the input data is represented by a principal connection path (see Fig. 1(a)). (2) The shortest connected region is mapped to this curve. (3) Some geometric rules are designed to straighten the curve. (4) The operation of straightening also drives the region to unfold accordingly. (5) A new region with simpler geometric structure is finally constructed (see Fig. 1(b)).

2.2. Prerequisite claims

The computation procedure of the CST algorithm are based on some geometric intuitions. Before describing the algorithmic details, two geometric conclusions are stated as follows. The equations in these conclusions are used somewhere in the CST algorithm.

- Let $\mathbf{L} : x = vE$ be an arbitrary line passing through the origin in $\mathbb{R}^n$, where v is a real number and E is a unit n-dimensional column vector. Then, the projection of point x_0 on $\mathbf{L}$ can be expressed by

$$x^* = EE^T x_0 \tag{1}$$

Here are some explanations. Suppose the projection of point x_0 on $\mathbf{L}$ is $x^* = v_0 E$. The inner product of vector $(x_0 - v_0 E)$ and E should then be 0, that is, $E^t(x_0 - v_0 E) = 0$. So it holds $v_0 = E^T x_0$. Therefore, $x^* = Ev_0 = EE^T x_0$.

- Let $\mathbf{S} : E^t x = 0$ be a $(n-1)$-dimensional hyperplane embedded in $\mathbb{R}^n$, where E is a unit n-dimensional column vector. Then, the projection point of point x_0 on $\mathbf{S}$ can be expressed by

$$x^* = x_0 - EE^T x_0 \tag{2}$$

Here are some explanations. The projection point x^* of point x_0 on $\mathbf{S}$ should satisfy: $x_0 - x^* = \beta E$ (β is a real number) and $E^T x^* = 0$. The former equation can be transformed into $x^* = x_0 - \beta E$. Multiply both sides by E^T, and it gets $E^T x_0 = \beta E^T E$. Since E is a unit vector, then $\beta = E^T x_0$. Finally, it gets $x* = x_0 - EE^T x_0$.

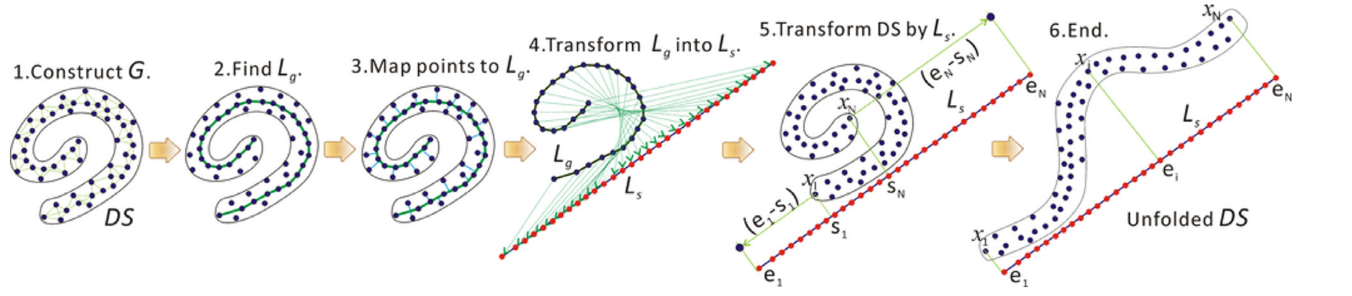

Fig. 3. The schematic diagram of CST steps. Each subgraph corresponds to one step of CST successively. *DS* used in the figure is a 2-dimensional dataset that is unfolded along one direction.

2.3. Algorithm implementation

The data of an application are always located on certain region. And the region is invariably expressed via its sampled data points that serve as the input to the CST algorithm. For a specific dataset, a nearest neighbor graph can be built to represent the discrete region, its connection components corresponding to the primitive subregions. Naturally, the shortest line segment between the points of two connection components is regarded as a bridge. This way, the shortest connected region of that dataset is constructed. Then, the connection path between two data points is approximated by the shortest path accessible on the connected nearest neighbor graph (i.e., the shortest connected region). Algorithm 1 presents the complete procedure of the CST approach.

```
Algorithm 1: The CST Algorithm.
   Input: $D = \{X_1, X_2, \dots, X_n\}$: $m \times n$ dataset;
          $K$: the number of neighbours;
          $U$: the number of unfolding directions.
   Output: $D'$: the unfolded $m \times n$ $D$.
 1 Initialize $D' = \{X'_1, X'_2, \dots, X'_n\} = \{X_1, X_2, \dots, X_n\}$;
 2 for $u = 1$ to $U$ do
 3     Construct a $K$ Nearest Neighbour (K-NN) graph $G$ from $D$;
 4     while $G$ not connected do
 5         Link a connection component to its nearest one;
 6     end
 7     Choose a random point $r$ and obtain a principal
       connection path via $r$, denoted as $L_g = [g_1, g_2, \dots, g_q]$;
 8     Calculate the length of $L_g$, denoted as $c$;
 9     Compute the connection distances to $g_1, g_q$, denoted as
       $[b_1, b_2, \dots, b_n]$ and $[a_1, a_2, \dots, a_n]$;
10     Calculate the mapping values $[d_1, d_2, \dots, d_n]$ via the law
       of cosines, i.e., $d_i = ({b_i}^2 + c^2 - {a_i}^2)/2c$;
11     Obtain the first principal component $E$ of $D$ via NIPALS;
12     Move each point $X'_i \in D'$ by $X'_i = X'_i + (e_i - s_i)$ (where
       $e_i = d_i E$ and $s_i = EE^T X_i$);
13     Project $D$ to the hyperplane $E^T X = 0$;
14 end
15 Return $D'$;
```

And an illustrative diagram is shown in Fig. 3 for explaining these steps.

Here are some explanations on the procedure in more details. The graph G constructed at Line 3 corresponds to the discrete region. The while loop at Line 4 tries to build a shortest connected region based on the graph G. In Lines 7–9, we find the principal connection path L_g by running the Dijkstra [15] algorithm twice. The first run is to find g_1, the other end of the longest connection path starting from r. The second run is to obtain the principal connection path L_g and the distances $[b_1, b_2, \dots, b_n]$. And finally, we need to run the Dijkstra algorithm one more time to compute $[a_1, a_2, \dots, a_n]$, respectively.

Mapping values $[d_1, d_2, \dots, d_n]$ at Line 10 indicate the mapping locations of D on L_g. The vector E calculated via NIPALS [16] at Line 11 is the direction on which the projection variance of D gets its maximum. The geometric transformation of the dataset D' at Line 12 is based on a line $L_s : x = \nu E$ along which the D' is to be unfolded. The $s_i = EE^T X_i$ (Eq. (1)) is the projection of $X_i \in D$ before the transformation, and the $e_i = Ed_i$ is the projection of X_i after the transformation. That is why we give X'_i such a movement. Line 13 projects D to the hyperplane $E^T X = 0$ by $D = D - E(E^T D)$ (Eq. (2)), resulting in a new D for the next iteration.

Note that the nearest neighbor graph built in the CST algorithm is only a common basic version, for this is not the focus of the paper. Some researchers have been trying some strategies to build more reasonable neighbor graphs [9,17]). The calculation of $[d_1, d_2, \dots, d_n]$ at Line 10 is the core part of the algorithm. We employ a variation of the Law of Cosine to construct the mapping because the three points (X_i, g_1 and g_q) connected by connection paths are regarded as a triangle in the curved space. In a triangle consisting of edges a, b, and c, it holds $a^2 = b^2 + c^2 - 2bc\cos(\alpha)$ according to the Law of Cosines, where α is the included angle of b and c. Then, we have $b\cos(\alpha) = (b^2 + c^2 - a^2)/2c$. We take $b\cos(\alpha)$ as the mapping value (i.e., d_i in Algorithm 1) to build a new mapping pattern. And this is our new idea.

As the algorithm shows, it selects U orthogonal directions for unfolding, which also means the number of loops the algorithm should run. Theoretically, the larger the U, the better the preprocessing effect. However, the triangles in the graph that we apply the Cosine theorem to are not strict Euclidean triangles. The loops may accumulate some errors at each loop. If U is too large, the unfolding effect at the latter loops may be weak. According to our preliminary tests, U had better be less than 5. considering both the computation burden and unfolding effect, the values 1–3 are good choice for most data. As far as K is concerned, it is actually the parameter K of the K-NN algorithm for constructing a K-nearest neighbour graph. There is no particular knowledge for selecting K. Common range of K is about 8–12 based on the common cognition, and 1–2 for some special purposes. For many cases, there is no big difference with varied K between 8 and 12.

3. Algorithm analysis

3.1. Algorithm complexity

In this subsection, we provide an analysis of the computational complexity of each step as a function of the number of samples n, the input dimension m, the number of unfolding directions U, and the other related factors if necessary.

The first step is to construct the neighbor graph G. For a given point x, computing the distances needs $O(mn)$ calculation, and sorting to get its K-NNs takes $O(Kn)$. Thus, for all n points, finding K-NNs is $O(n^2(m+K))$. To insure a connected G, what follows is to check the connectivity of G, which requires $O(Kn)$. If G is not connected, calculating the distances between the points of two connected components (let the numbers of elements in the two sets be n_1 and n_2) is $O(n_1n_2m)$ and finding the two nearest points takes $O(n_1n_2)$.

The next step is to find the principal connection path $L_g = [g_1, g_2, \ldots, g_q]$. On the neighbor graph G, it spends $O(n_2)$ to find the endpoint g_1 by searching from a randomly selected point x_r via the Dijkstra algorithm. In the same way, using the Dijkstra to find $L_g = [g_1, g_2, \ldots, g_q]$ and all the connection paths starting from g_1 needs same computation. Again, finding all the connection paths starting from g_q via the Dijkstra algorithm costs equal computation. With the computational results of the previous steps, the third step is very simple. The complexity of calculating the mapping positions of all data points on L_g is $O(n)$.

The major task in transforming L_g into unfolding line L_s is to find the unfolding direction E, which is done via NIPALS algorithm. As we know, each iteration of NIPALS needs $O(mn)$ and the maximum number of iterations is a constant. Therefore, its complexity is still $O(mn)$. Besides, marking the transferred mapping point $e_i = Ed_i$ on L_s for each $X_i \in D$ requires also $O(mn)$ calculation. During the course of transforming D' by L_s, projecting each point $X_i \in D$ to L_s and gaining its projection point s_i takes $O(mn)$. And then moving each point $X_i' \in D'$ by $X_i' = X_i' + (e_i - s_i)$ requires also same calculation. Lastly, if required, projecting D to the hyperplane defined by E spends still $O(mn)$.

In summary, the computation of unfolding data along one direction is $O(n^2(m+K))$. Considering the number of unfolding directions U, the final computational complexity of CST should be $O(Un^2(m+K))$.

3.2. Algorithm properties

To further investigate the properties of the proposed algorithm, we have the following results.

CLAIM 1. Suppose that the mapping positions of two data points P_1, P_2 on L_g are indicated by d_1 and d_2. Then it holds

$$\lim_{P_2 \to P_1} (d_2) = d_1.$$

Proof. Let b_1, a_1 and b_2, a_2 be the connection distances between two nearby data points P_1, P_2 and the two endpoints g_1, g_q of the principal connection path L_g. Since P_1, P_2 stay close, it is reasonable to approximate their connection distance by the Euclidean distance (denoted as s). We may assume $b_2 = b_1 + b_e$. It can be verified based on the definition of connection distance that $b_2 < b_1 + s$ and $b_1 < b_2 + s$. Then it follows that $b_1 + b_e < b_1 + s$ and $b_1 < b_1 + b_e + s$. So, it holds that $b_e < s$ and $-s < b_e$, that is, $|b_e| < s$. Hence

$$\lim_{s \to 0} (b_e) = 0$$

Therefore, it holds that

$$\lim_{s \to 0} (b_2) = b_1$$

Similarly, it is true that

$$\lim_{s \to 0} (a_2) = a_1$$

As

$$\lim_{\substack{b_2 \to b_1 \\ a_2 \to a_1}} (d_2) = \lim_{\substack{b_2 \to b_1 \\ a_2 \to a_1}} \frac{b_2^2 + c^2 - a_2^2}{2c} = \frac{b_1^2 + c^2 - a_1^2}{2c} = d_1$$

we get that

$$\lim_{s \to 0} (d_2) = d_1 \Rightarrow \lim_{P_2 \to P_1} (d_2) = d_1. \quad \Box$$

□

Clearly the claim implies that two points nearby will stay still nearby after unfolding, because their moving displacements along unfolding direction are quite close. Therefore, the CST algorithm is able to preserve the locality relations in the dataset. Namely, the *CST is a locality preserving technique*. Moreover, two far points are sure to keep far if their mapping points are far on L_g. Consequently, the *CST has the globality preserving property to a certain degree*.

Naturally, the geometric transformations in the CST are dependent on the selected principal connection path, but they will make no obvious difference if the selected curve is long enough to reflect the geometric structure of the dataset. For a point (denoted as B) in the nearest neighbor graph, if there exists another point (denoted as A) such that the shortest path starting from point A to point B is not contained in any other shortest paths starting from point A, then point B is looked as an edge point. The principal connection path sought by applying the Dijkstra twice is a path starting from one edge point to another farthest one thus being always able to reflect the principal geometric structure of the data. And this has ensured the effectiveness of the CST algorithm.

4. Experiments

In this section a serial of comprehensive experiments are performed to examine the CST. The dataset is unfolded by the CST and then reduced into lower dimensions via PCA. Therefore, our approach is indicated as CST+PCA in the experiments as follows. Furthermore, many notable dimensionality reduction techniques are selected to compete, including PCA, Isomap, LLE, LE, DM, LTSA, MVU, Hessian LLE (HLLE) [18], KPCA, Sammon mapping [19]. To make a full comparison of these techniques, five artificial datasets and five real datasets are taken as testing data. The artificial datasets focus on multi-region and irregular sampling cases which are common in real-world applications. And the choice of real datasets lays more stress on comprehensiveness, considering the variety in dimensions, number of samples, and domains. Furthermore, to better compare the performances quantitatively, all the selected real datasets contain label information. And it is reasonable to assume that the data points with different labels are on different subregions. Note that the label information is only used after the datasets are reduced in dimensions. Therefore, all the techniques used in the experiments are unsupervised. In next subsection, we introduce some quantitative evaluation items.

4.1. Experimental setting

All the artificial datasets are mapped from a three-dimensional (3D) space onto a two-dimensional (2D) plane. Since the data with three dimensions or less can be directly visualized on the corresponding coordinate systems, and the artificial datasets are all 3D data, the performances of different techniques on these datasets can be directly compared. The dimensions of the real datasets are typically greater than three, however. To get more diversity, we transform the real datasets to the spaces of randomly-chosen fewer dimensions. And we set up three types of quantitative evaluation measures to better compare the dimensionality reduction results.

The first type of evaluation measure is the preserving rate of nearest neighbors (*NR*). This measure is mainly used to inspect on how much the lower dimensional representation preserves the locality (i.e., nearest neighbors) of the original dataset. The calculation is very simple. Two nearest neighbor graphs are constructed on the original and dimensionality reduced data sets via

K-NN algorithm with same K value respectively. And then the preserved neighbors of each point are counted. Based on this concept, we are able to give the definitions of this measure. Suppose that $T(a, b)$ is a function returning the number of elements of the intersection of sets a and b. And let n be the number of data points of the dataset and K be the value of the parameter K of K-NN. The NR is then defined as

$$NR = \frac{1}{K * n} \sum_{i=1}^{n} T(S_i, R_i) \tag{3}$$

Where S_i and R_i are the nearest neighbor sets of the data point i in the two graphs constructed on the original and dimensionality reduced data sets.

The second measure is the generalization error (GE). The generalization errors of the K-NN classifiers are able to reflect the local clustering of the same-class points because the classifiers test the neighborhoods between the data points of a same class. Generally, low GE value (which means good local clustering) often implies good potential for classification and clustering. Here, the generalization errors of the 1-NN and 12-NN classifiers are measured using leave-one-out validation. To K-NN classifiers, $k = 1$ is the most critical argument. Only when a data point is the same class as its closest one, 1-NN classifiers produce correctly classified results. This value is often used in the experiments. By contrast, it is easy to know that $k = 12$ is the loosest argument in the usual range (8–12) based on the similar analysis. We use both $k = 1$ and $k = 12$ for more comprehensive measurements (compared with using only $k = 1$). Unlike previous evaluation items, the smaller, the better for GE.

To make the experiments more convincing and trustable, we employed the Support Vector Machine (SVM) and Back Propagation (BP) Neural Network for classification on the original and dimensionality reduced datasets. And we adopted One-Vs-The-Rest strategy to reduce the problem of multiclass classification to multiple binary classification problems. The training and classification functions of SVM are directly from the Statistics and Machine Learning Toolbox of Matlab 2014a. We choose least squares as the method to find the separating hyperplane for function svmtrain. Other arguments of svmtrain adopt the default values. The source code of BP Neural Network is downloaded from Phil Brierley [20]. And we used only default values for the experiments. The real datasets are divided into two subsets, with 90% of samples for training and the other 10% for testing. The measures are obtained by averaging of 10-fold cross validation.

Amidst the techniques used in the experiments, the CST+PCA, Isomap, LLE, LE, LTSA, MVU and HLLE are dependent on nearest neighbor graphs. And the graphs are all constructed via K-NN algorithm with $K = 10$ in the experiments. The other parameter of the CST+PCA is $U = 1$. And the other parameter of LE is $\sigma = 1$. The two parameters of DM are set to $t = 10$ and $\sigma = 1$. The KPCA using linear kernel equals to traditional PCA, and the KPCA using Gaussian kernel resembles to Diffusion Maps [12]. Thus, a polynomial kernel with $K(x, y) = (x.y + 1)^5$ is provided to the KPCA in the experiments. PCA and Sammon have no parameters for setting. We ran the experiments for the parameter settings described above, and recorded the corresponding measurement items. For the NR and GE, we only present the recorded result of each technique. And for the SVM recognition rates, we take a mean accuracy of ten runs of a technique on each real dataset.

4.2. Experiments on artificial datasets

Amidst the five artificial datasets, Changing-Swiss-roll and 3D-Clusters are generated by drtoolbox [21]. Suspending-peaks are adapted from the Twin-peaks dataset generated by drtoolbox. Other two datasets, Overlapped-hemispheres and Nested-C, are generated by ourselves. The numbers of data points of these datasets range from 1600 to 2000. The 3D datasets are reduced to 2D by the selected techniques and then plotted on a plane. On the 2D plotting, we firstly inspect the separation between different data patches(or clusters). And inside each patch, we further check if the locality is still kept. To this end, we turn to observe whether the data patches are unfolded well. In the followings, we compare the results of each dataset in detail.

(1) Overlapped-hemispheres data in Fig. 4. This dataset consists of one small hemisphere covered by another bigger one. The CST+PCA and MVU were able to separate the two hemispheres with each hemisphere unfolded well and their size ratio kept, but the shapes in the MVU low-dimensional representation were not so smooth and regular as those in the CST+PCA representation. The LLE failed to keep the size ratio and the unfolding was not as good as the CST+PCA and MVU. The Isomap, LTSA, HLLE and LE were also able to separate two hemispheres, but they failed to unfold at least one part. The PCA, DM, Sammon and KPCA projected the small hemisphere into the bigger one. (2) Nested-C data in Fig. 5. This dataset contains two C-shape patches with one nested into the other. The CST+PCA was able to separate the two patches with each one unfolded well and their size ratio kept. The Isomap, LLE and LE were also able to separate the two parts, but they failed to unfold at least one. The two patches unfolded by the MVU nearly touched together and lost their size ratio. The PCA, DM and Sammon gained the similar incorrect low-dimensional representation. The KPCA mixed two patches into a fan-like minor sector. The LTSA and HLLE produced a cross-like shape.

(3) Suspending-peaks data in Fig. 6. This dataset consists of four peaks with one pair suspending under and the other pair floating over a grid-like base frame. The CST+PCA, LTSA, Sammon and HLLE were able to correctly visualize the four peaks as well as the base frame. The PCA and DM projected the dataset from an improper direction, thus unable to display the base frame correctly. The MVU and LLE only unfolded the dataset partly, and some components were not able to be identified. The LE, Isomap and KPCA only produced some intertwining lines.

(4) 3D-Clusters data in Fig. 7. The Sammon and CST+PCA were able to separate the six clusters with each cluster deployed well. The Sammon outperformed the CST+PCA as far as unfolding is concerned. The Isomap made two clusters shrunk to short lines. The HLLE, LE and LTSA failed to unfold each cluster although they separate the six clusters. The MVU failed to unfold two clusters with one cluster overlapped by another one. The PCA and DM made two pairs of clusters overlapped. The LLE and KPCA performed poorly on both unfolding and separation.

(5) Changing-Swiss-roll data in Fig. 8. This dataset is sampled from a Swiss roll with irregular sampling rates. It is very challenging for most dimensionality reduction techniques. The CST+PCA was able to unfold the Swiss roll correctly despite the imperfect shape. The Isomap produced a curve-like shape. The Swiss roll was made overlapped heavily by the rest techniques.

In general, the CST+PCA performs well on all the five artificial datasets in terms of both correct separation and good unfolding. Considering correct separation alone, all techniques can be ranked as: CST+PCA (5), Isomap (4), HLLE (3), LTSA (3), LE (3), Sammon (2), MVU (2), LLE (2), PCA (0), DM (0), KPCA (0). The number following each technique indicates on how many datasets the technique is able to do correct separation. Therefore, even not taking unfolding into account, the CST+PCA is still among the best techniques.

4.3. Experiments on real datasets

Artificial data, after all, are just a very simple simulation of real data. The performances of techniques on real data tend to be more valuable. For this reason, we selected five real datasets

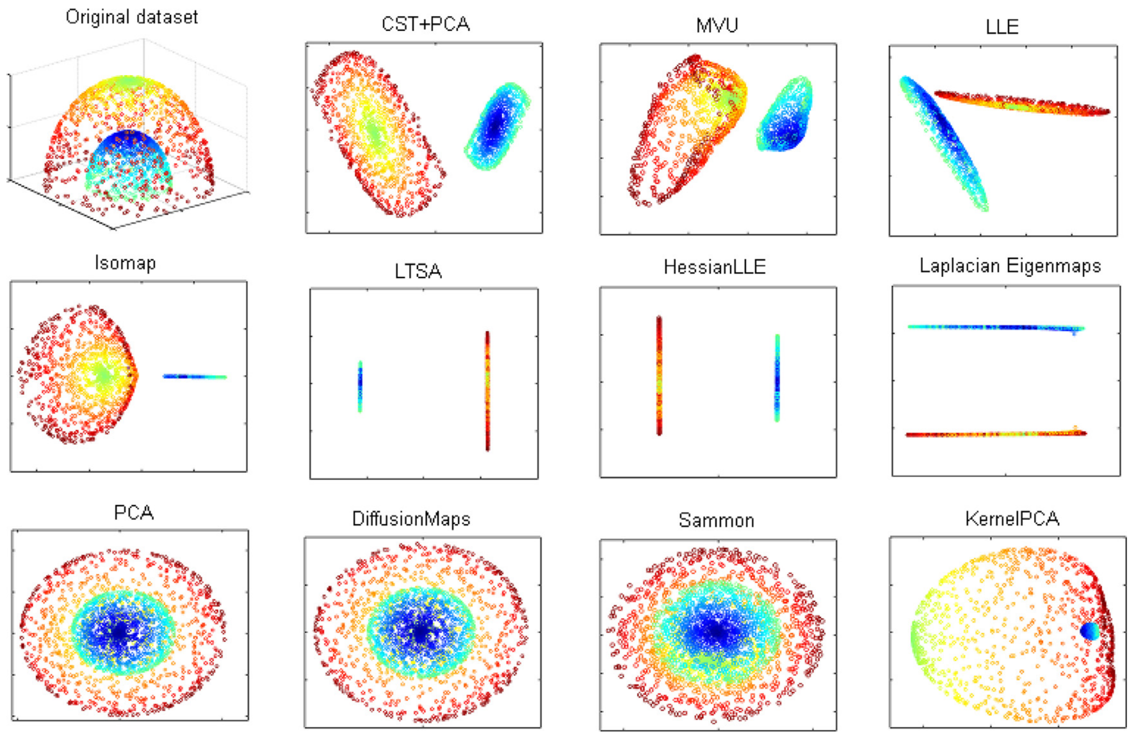

Fig. 4. Two-dimensional plotting results of the Overlapped-hemispheres data.

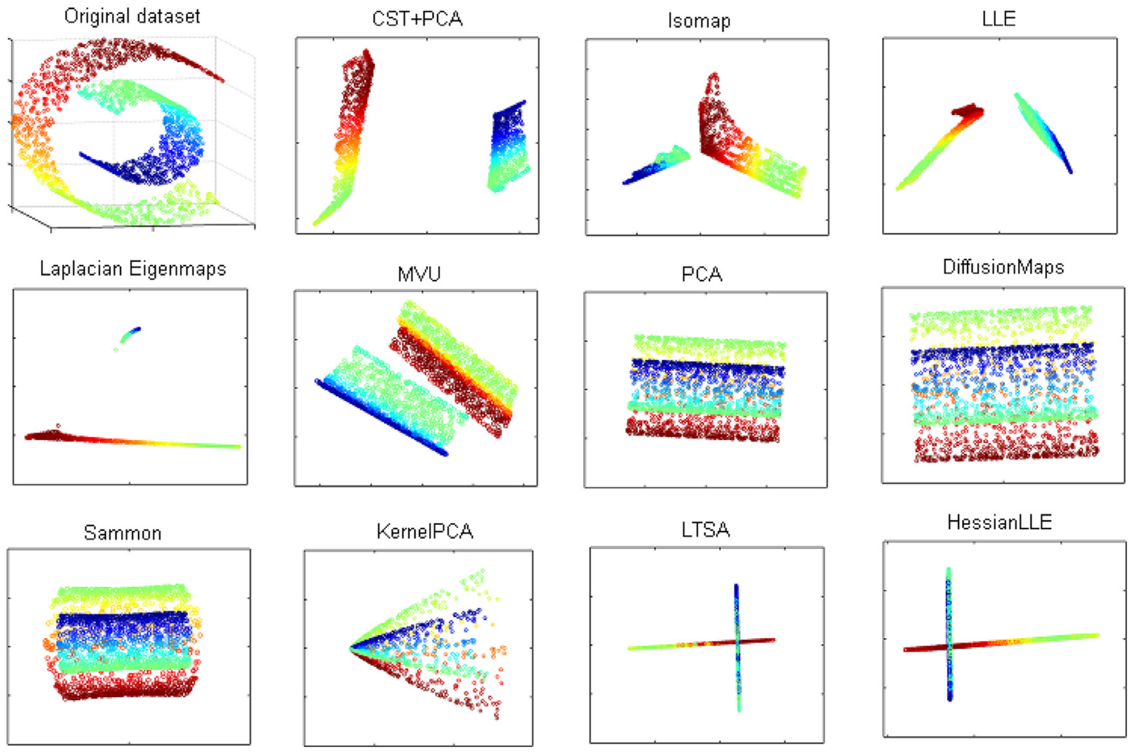

Fig. 5. Two-dimensional plotting results of the Nested-C data.

coming from different domains: (1) the ORL face dataset [22], (2) the Libras Movement dataset, (3) the Climate Model Simulation Crashes dataset, (4) the Ecoli dataset, and (5) the Semeion Handwritten Digit dataset. The latter four datasets are downloaded from the UCI Machine Learning Repository [23]. All the datasets are normalized before dimensionality reduction. All the five measures are shown on the tables. Therefore, the experimental result of a dataset corresponds to one table. In the following, we compare the performances of different techniques on each dataset. Note that the Ori, CPCA, ISO, and Sam are the abbreviations for Original, CST+PCA, Isomap, and Sammon, respectively, in Tables 1–5. And all the values in these tables are percentage.

(1) The ORL data in Table 1. The ORL dataset used in the experiments is a face recognition dataset that contains 400 gray scale images of 32×32 pixels that depict 40 faces under various conditions (i.e., the dataset contains 10 images per face). In Table 1, the

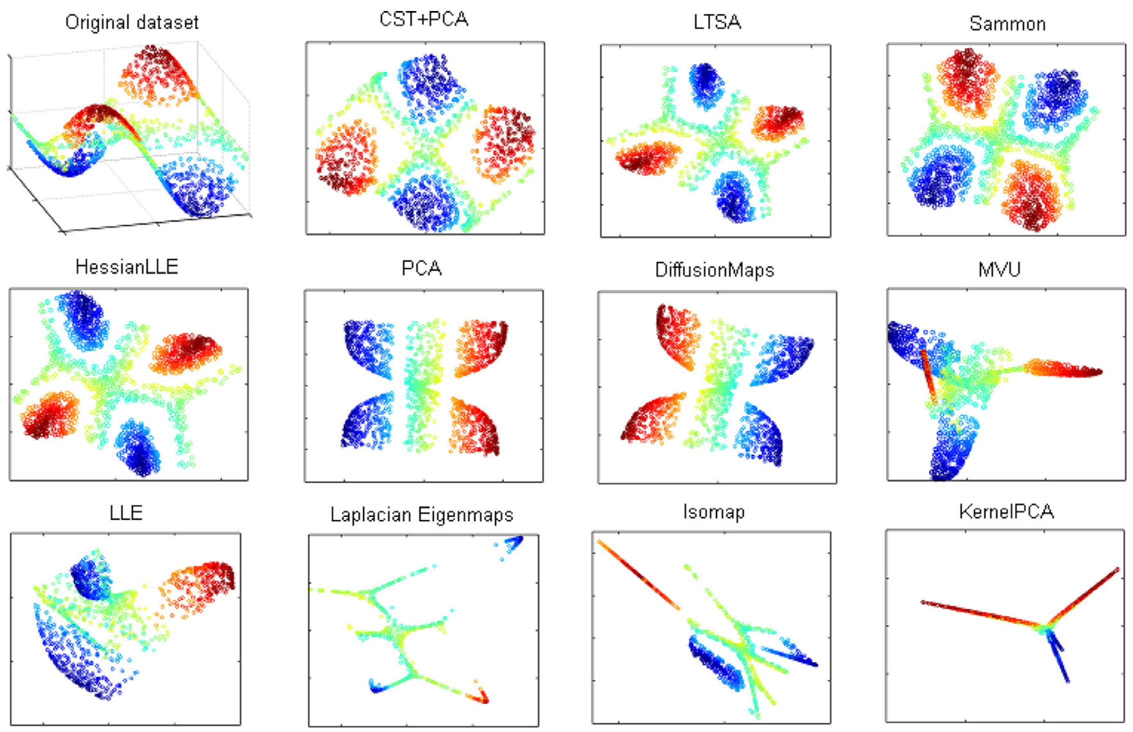

Fig. 6. Two-dimensional plotting results of the Suspending-peaks data.

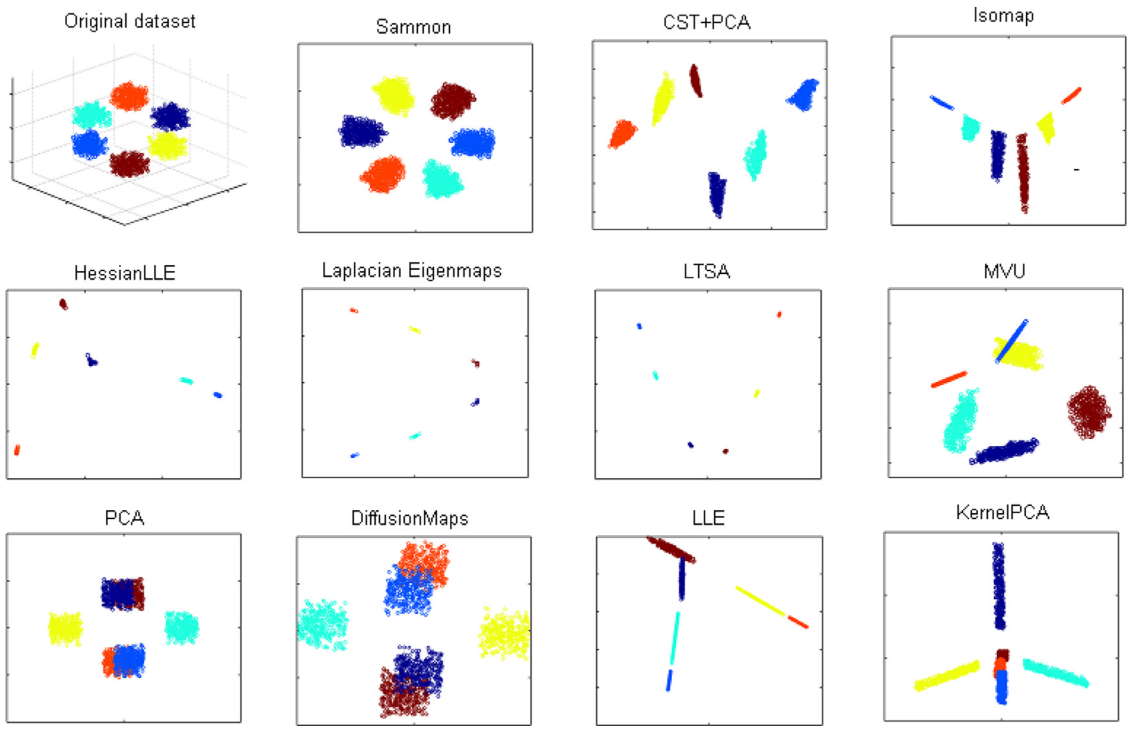

Fig. 7. Two-dimensional plotting results of the 3D-Clusters data.

Table 1
The evaluation of 10 D representation of the ORL data. The best values are in the bold style.

Item	Ori	CPCA	PCA	ISO	LLE	LE	DM	LTSA	MVU	HLLE	KPCA	Sam
NR	100	67.75	71.06	45.27	46.33	55.15	40.29	23.13	46.88	22.10	37.29	**73.77**
$GE_{(k=1)}$	5.25	11.50	9.25	29.50	14.25	32.50	30.57	42.50	32.50	70.00	38.50	**8.50**
$GE_{(k=12)}$	31.00	50.75	43.00	51.75	43.00	50.50	62.25	71.00	64.75	75.75	66.00	**33.50**
SVM	88.00	**70.83**	63.33	65.00	58.33	63.33	57.50	20.00	67.96	48.67	57.33	65.67
BP	57.50	**51.85**	47.50	35.73	39.22	40.54	51.50	35.88	20.65	46.25	48.69	43.76

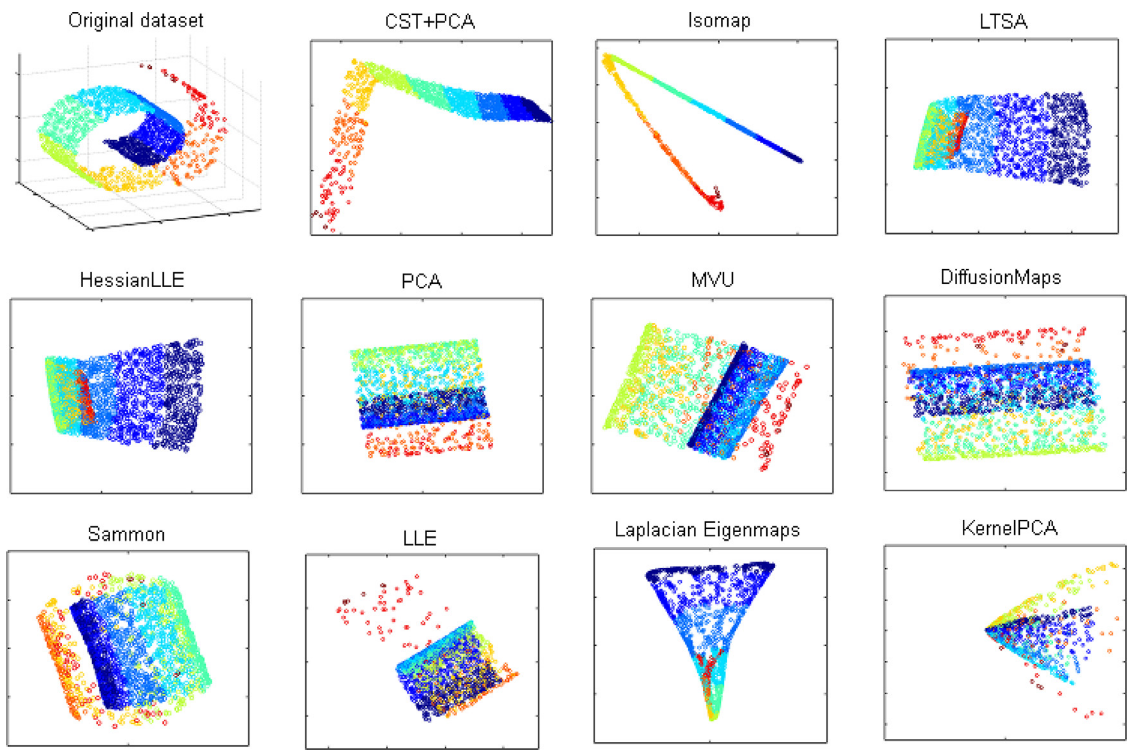

Fig. 8. Two-dimensional plotting results of the Changing-Swiss-roll data.

Table 2
The evaluation of 4D representation of the Libras Movement data. The best values are in the bold style.

Item	Ori	CPCA	PCA	ISO	LLE	LE	DM	LTSA	MVU	HLLE	KPCA	Sam
NR	100	**78.68**	78.52	62.62	29.47	59.05	40.86	57.22	24.94	22.52	42.73	60.02
$GE_{(k=1)}$	13.05	**20.00**	**20.00**	25.83	46.66	36.11	37.50	22.22	50.00	70.00	36.66	31.63
$GE_{(k=12)}$	39.72	39.44	**36.94**	39.44	63.89	48.33	50.00	50.00	64.16	75.00	54.44	43.43
SVM	64.44	**61.11**	50.00	56.67	52.96	59.63	52.96	52.59	41.67	47.78	47.78	53.75
BP	72.22	**64.14**	52.00	63.89	52.77	25.00	44.44	27.78	46.53	63.33	58.63	27.78

Table 3
The evaluation of 3D representation of the Climate Model Simulation Crashes data. The best values are in the bold style.

Item	Ori	CPCA	PCA	ISO	LLE	LE	DM	LTSA	MVU	HLLE	KPCA	Sam
NR	100	**57.59**	8.13	8.41	7.80	14.15	7.41	8.04	8.37	7.39	5.96	8.04
$GE_{(k=1)}$	11.48	14.26	16.11	14.07	**13.33**	15.37	15.74	15.74	**13.33**	15.56	15.19	15.37
$GE_{(k=12)}$	7.04	9.44	9.63	8.89	8.70	8.70	8.70	**8.52**	**8.52**	**8.52**	**8.52**	**8.52**
SVM	86.67	**74.07**	61.11	60.74	56.30	61.48	64.44	26.67	56.05	25.93	69.15	63.44
BP	96.27	**98.15**	98.00	79.63	**98.15**	96.30	96.30	94.44	95.66	97.15	50.00	83.33

Table 4
The evaluation of 2D representation of the Ecoli data. The best values are in the bold style.

Item	Ori	CPCA	PCA	ISO	LLE	LE	DM	LTSA	MVU	HLLE	KPCA	Sam
NR	100	**74.46**	38.53	25.18	22.67	37.07	14.53	16.84	10.98	12.31	20.91	42.80
$GE_{(k=1)}$	18.24	25.40	26.38	25.41	44.96	**19.87**	35.18	38.76	42.67	40.72	40.72	20.85
$GE_{(k=12)}$	13.03	17.26	16.94	19.87	38.11	**15.64**	28.01	37.46	38.11	38.76	39.74	16.61
SVM	83.90	**85.56**	74.44	61.84	60.61	83.06	61.84	64.94	67.96	46.38	42.42	81.39
BP	100	90.00	86.68	70.00	76.67	76.67	83.33	46.67	66.32	**96.67**	16.67	83.33

Table 5
The evaluation of 8D representation of the Semeion Handwritten Digit data. The best values are in the bold style.

Item	Ori	CPCA	PCA	ISO	LLE	LE	DM	LTSA	MVU	HLLE	KPCA	Sam
NR	100	**59.44**	55.77	46.03	33.61	43.34	20.63	3.58	35.29	7.74	20.24	57.59
$GE_{(k=1)}$	7.78	10.42	**9.42**	11.93	16.94	11.17	35.46	69.05	12.55	54.48	27.93	9.66
$GE_{(k=12)}$	9.92	14.24	**8.29**	10.80	17.71	13.11	36.40	70.24	14.57	50.22	27.87	8.53
SVM	75.68	**69.81**	65.83	67.77	61.39	47.99	57.95	60.47	56.05	47.78	65.83	67.84
BP	81.76	**74.59**	70.82	74.22	70.82	67.04	72.14	61.58	53.85	50.57	71.45	64.53

NR value of CST+PCA, 67.75%, was only a slightly less than the best one, 73.77%, ranked third behind Sammon and PCA. This means the CST+PCA is among the best in preserving the local structure. The $GE(k=1)$ of CST+PCA ranked same as *NR*, which gives the CST+PCA a good probability for pattern recognition. And the best SVM and BP recognition rates demonstrated that CST+PCA is good for later pattern recognition. The $GE(k=12)$ seems not relevant to other items very much. Therefore, the $GE(k=1)$ may be more meaningful for evaluating the performance.

(2) The Libras Movement data in Table 2. The Libras Movement dataset contains 15 classes of 24 instances each. Each class references to a hand movement type in LIBRAS (Portuguese name 'Lingua BRAsileira de Sinais', official brazilian signal language). And each movement is represented by 90 features. As the *NR* values in Table 2 shows, the CST+PCA presented the best $GE(k=1)$ value and the best preserving rate of nearest neighbors. And the CST+PCA had also the best SVM and BP recognition rates. Moreover, the $GE(k=12)$ was only less than that of PCA in Table 2. Although PCA did well on the first three testing items, the CST+PCA still a little outperformed PCA in SVM and BP recognition. This shows that the CST is able to improve the performance of PCA.

(3) The Climate Model Simulation Crashes data in Table 3. This dataset contains records of simulation crashes encountered during climate model uncertainty quantification (UQ) ensembles. Each ensemble comprises 18 climate model input parameter values. Three separate Latin hypercube ensembles were conducted, each containing 180 ensemble members. The goal is to use classification to predict simulation outcomes (fail or succeed) from input parameter values. Surprisingly, the CST+PCA had much better *NR* value than any other techniques. And consequently, the CST+PCA had also the best SVM recognition rate. Moreover, the BP recognition rate and generalization errors of the CST+PCA were as good as or better than those of the other techniques as well. Therefore, the advantages of the CST+PCA are noticeable on keeping local structure and improving the recognition performances.

(4) The Ecoli data in Table 4. This dataset is used for predicting protein localization sites. The dataset contains 8 classes of 336 instances where each class refers to a localization site depicted with 7 attributes. And we removed three classes which contain only less than 5 instances. According to the information in Table 4, we could achieve similar analytical results as in the Climate Model Simulation Crashes data. The preserving rate of nearest neighbors obtained from CST+PCA was much higher than those obtained from other techniques. And the SVM recognition rate of the CST+PCA was still better than the other techniques. This also demonstrates that the CST+PCA is more widely applicable than the other techniques.

(5) The Semeion Handwritten Digit data in Table 5. The Semeion Handwritten Digit dataset is a dataset of 1593 scanned handwritten digits from around 80 persons. Each digit is stretched in a rectangular box 16×16 in a binary scale. The CST+PCA showed no advantages with respect to the $GE(k=1)$ and $GE(k=12)$ measurements. As the Table 5 shows, however, the CST+PCA achieved best values in *NR*, SVM and BP tests. Although there were no big gaps between the best values and the their successors, it is at least able to say that the CST+PCA is among the best techniques regarding this dataset.

In the five-real-dataset experiments, the CST+PCA occupied most of the best pattern recognition. And the CST+PCA also outperformed all other techniques in the *NR* evaluation on three datasets. To visually and numerically compare the techniques, we assign an integer to a technique for each evaluation according to the ranking of performance. For example, a technique ranking 1 (or 2) ,which means best, gains value 1 (or 2), and so on. The larger the value, the worse the performance. Finally, we average the ranking values for each technique over the five-real-dataset experiments. According to this rule, the techniques can be ranked as: CST+PCA(2.32), Sammon (3.72), PCA(4.04), Isomap(4.96), LE (4.96), LLE(6.40), DM(6.68), MVU(7.32), KPCA(7.40), LTSA(8.08), HLLE(8.72). The value following each technique is the averaged ranking value. The ranking clearly shows the improvement of PCA via the CST preprocessing.

Combining the two groups of experiments, it is clear that the CST tends to unfold the data so that the geometric structure of the data becomes simpler. This preprocessing lays a good foundation for later PCA processing. Therefore, with the CST support, the PCA is still able to produce good results even for the datasets that those nonlinear techniques perform poorly on. Besides, we can see from the experiments that the CST is applicable to different datasets, showing strong robustness.

5. Conclusions and discussions

Based on the CST principle, we are able to know that the CST is at least much suitable for such kind of multi-class data: (1) each class of points is clustered together. Therefore, one class of points corresponds to one cluster. (2) These clusters are located in such a chaotic way that no matter what hyperplane is used as the projecting plane, there are always some clusters overlapped. And of course, we believe the CST is not only limited to such kind of data. The Experiments on both artificial and real datasets showed the good performance of the CST towards different types of data.

As a matter of fact, the geometric structures of the dataset becomes simpler after the CST unfolding, even the PCA can be good enough for the following dimensionality reduction on most datasets. This is the main contribution of this article. Existing nonlinear techniques may either require strong assumptions on the input data or need careful parameter tuning. By contrast, the proposed CST algorithm does not depend on strong assumptions. Also the parameter setting for the CST is not burdensome.

In conclusion, by the novel idea of geometric structure simplification, we introduced a data preprocessing method for PCA (or other linear projection methods) from a geometric perspective. Comprehensive experiments demonstrate that the CST algorithm can unfold the nonlinear geometric structures into a simpler structure. In this way, the original complicated dataset could become more tractable for linear dimensionality reduction methods. In the future we would investigate how to incorporate label information in this preprocessing algorithm for better discriminant capability in classification problems.

Acknowledgments

This work was supported in part by grants from Natural Science Foundation of Zhejiang Province of China (Grant no. LY15F020005), National Key Technology Research and Development Program of the Ministry of Science and Technology of China (Grant no. 2014BAK14B01), and National Science Foundation of China (Grant nos. 61170098, 61379075, 6110037,61672460).

References

[1] B. Sluban, D. Gamberger, N. Lavrac, Ensemble-based noise detection: noise ranking and visual performance evaluation, Data Min. Knowl. Discov. 28 (2) (2014) 265–303.
[2] P.G. Clark, J.W. Grzymala-Busse, W. Rzasa, Emining incomplete data with singleton, subset and concept probabilistic approximations, Inf. Sci. 280 (2014) 368–384.
[3] S. Zhang, Q. Chen, Q. Yang, Acquiring knowledge from inconsistent data sources through weighting, Data Knowl. Eng. 69 (8) (2010) 779–799.
[4] J.B. Tenenbaum, V. de Silva, J.C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (5500) (2000) 2319–2323.
[5] K.Q. Weinberger, L.K. Saul, Unsupervised learning of image manifolds by semidefinite programming, Int. J. Comput. Vis. 70 (1) (2006) 77–90.
[6] S.T. Roweis, L.K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326.

[7] M. Belkin, P. Niyogi, Laplacian eigenmaps for dimensionality reduction and data representation, Neural Comput. 15 (6) (2003) 1373-139.
[8] Z. Zhang, H. Zha, Principal manifolds and nonlinear dimensionality reduction via tangent space alignment, J. Shanghai Univ. (English Ed.) 8 (4) (2004) 406–424.
[9] T. Lin, H. Zha, Riemannian manifold learning, IEEE Trans. Pattern Anal. Mach. Intell. 30 (5) (2008) 796–809.
[10] L.J.P. van der Maaten, E.O. Postma, H.J. van den Herik, Dimensionality Reduction: A Comparative Review, Tilburg University, 2009. Technical Report, TiCC-TR 2009-005.
[11] H. Yin, W. Huang, Adaptive nonlinear manifolds and their applications to pattern recognition, Inf. Sci. 180 (14) (2010) 2649–2662.
[12] P. Nieminen, I. Polonen, T. Sipola, Research literature clustering using diffusion maps, J. Inf. 7 (4) (2013) 874–886.
[13] G. Hinton, R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.
[14] C.C. Chen, Z.-Y. Chen, C.-Y. Wu, An unsupervised approach for person name bipolarization using principal component analysis, IEEE Trans. Knowl. Data Eng. 24 (11) (2012) 1963–1976.
[15] S. Wang, S. Yang, Z. Liu, L. Jiao, Saliency generation from complex scene via digraph and Bayesian inference, Neurocomputing 170 (2015) 176–186.
[16] H. Wold, Path Models with Latent Variables: the NIPALS Approach, Academic Press, 1975.
[17] Z. Zhang, J. Wang, H. Zha, Adaptive manifold learning, IEEE Trans. Pattern Anal. Mach. Intell. 34 (2) (2012) 253–265.
[18] D.L. Donoho, C. Grimes, Hessian eigenmaps: locally linear embedding techniques for high-dimensional data, Proc. Natl. Acad. Sci. USA 100 (10) (2003) 5591–5596.
[19] J.W. Sammon, A nonlinear mapping for data structure analysis, IEEE Trans. Comput. 18 (5) (1969) 401–409.
[20] P. Brierley, Matlab neural network backprop code, Retrieved January 15, 2018 from http://www.philbrierley.com/code.html.
[21] L.J.P. van der Maaten, Matlab Toolbox for Dimensionality Reduction, Retrieved January 7, 2015 from http://homepage.tudelft.nl/19j49/Matlab_Toolbox_for_Dimensionality-_Reduction.html.
[22] D. Cai, Four Face Databases in Matlab Format, Retrieved February 8, 2015 from http://www.cad.zju.edu.cn/home/dengcai/Data/FaceData.html.
[23] K. Bache, M. Lichman, UCI Machine Learning Repository, Retrieved February 8, 2015 from http://archive.ics.uci.edu/ml.

Hua-Mao Gu is an associate professor of School of Computer Science and Information Engineering, Zhejiang Gongshang University. He studied computer science at Zhejiang University and obtained his Ph.D. there. Prior to that, he graduated from Nanjing University of Science and Technology, and received his Masters degree in pattern recognition. His current research interest focuses on dimensionality reduction and manifold learning.

Tong Lin received the Ph.D. degree in applied mathematics from Peking University, Beijing, China, in 2001. In 1999 and 2000, he was a visiting student at Microsoft Research Asia, Beijing. In 2002, he joined the State Key Laboratory of Machine Perception at Peking University, where he is currently an associate professor. From 2004 to 2005, he was an exchange scholar at Seoul National University, Korea. His research interests include wavelet analysis, computer vision, pattern recognition, and machine learning.

Xun Wang is currently a professor at the School of Computer Science and Information Engineering, Zhejiang Gongshang University, China. He received his B.Sc. in mechanics, Ph.D. degrees in computer science, all from Zhejiang University, Hangzhou, China, in 1990 and 2006, respectively. His current research interests include mobile graphics computing, image/video processing, pattern recognition, intelligent information processing and visualization. In recent years, He has published over 80 papers in high-quality journals and conferences. He holds 11 authorized invention patents and 5 provincial and ministerial level scientific and technological progress awards. He is a member of the IEEE and ACM, and a senior member of CCF.