Neural Comput & Applic
DOI 10.1007/s00521-013-1369-z

ORIGINAL ARTICLE

Learning high-dimensional correspondence via manifold learning and local approximation

Chenping Hou · Feiping Nie · Hua Wang · Dongyun Yi · Changshui Zhang

Received: 14 June 2012 / Accepted: 22 February 2013
© Springer-Verlag London 2013

Abstract The recent years have witnessed a surge of interests of learning high-dimensional correspondence, which is important for both machine learning and neural computation community. Manifold learning–based researches have been considered as one of the most promising directions. In this paper, by analyzing traditional methods, we summarized a new framework for high-dimensional correspondence learning. Within this framework, we also presented a new approach, Local Approximation Maximum Variance Unfolding. Compared with other machine learning–based methods, it could achieve higher accuracy. Besides, we also introduce how to use the proposed framework and methods in a concrete application, cross-system personalization (CSP). Promising experimental results on image alignment and CSP applications are proposed for demonstration.

We thank the National Natural Science Foundation of China, under Grant No. 61005003, 60975038, for their supports.

C. Hou (✉) · D. Yi
Department of Mathematics and Systems Science,
National University of Defense Technology,
Changsha 410073, China
e-mail: hcpnudt@gmail.com

F. Nie · H. Wang
Department of Computer Science and Engineering,
University of Texas, Arlington, TX 76019, USA

C. Zhang
State Key Laboratory of Intelligent Technology and Systems,
Tsinghua National Laboratory for Information Science
and Technology (TNList), Department of Automation,
Tsinghua University, Beijing 100084, China

Keywords High-dimensional correspondence · Manifold learning · Local approximation · Maximum variance unfolding

1 Introduction

In many real applications, one can often observe the same object, such as images or web texts, in many different aspects. These descriptions are highly related in essence though looks different sometimes. For example, the same scene may be captured from different viewpoints [9] and the same action of different objects is recorded separately. Besides, we may record profiles of a unique person when he/she browses multiple web sites. In these scenarios, how to learn the correspondence between different observations and descriptions is very important. Taking facial expression recognition as an instance, if we can align one person's faces to a benchmark data, which contains expressions, for example, happy, sad, anger, etc, we can determine this person's expression directly [27]. Similarly, in cross-system personalization (CSP) [11], it will be much easier to accommodate a system with respect to user's tastes if we can align profile data from different systems.

Due to the fact that these data often have high dimensionality, it is difficult to construct the mapping between two observations directly. For example, it is usually difficult to construct a mapping function between two image sets with resolution 100×100, since they are 100,000 dimensional vectors. Fortunately, from the geometric view, each observation set is often lying on a low-dimensional manifold [15, 23, 26]. We can first reduce the dimensionality of each data set and then learn their correspondences in the embedded low-dimensional spaces.

Many efforts have been devoted to reducing the dimensionality of high-dimensional data. The widely used methods include feature selection [18] and feature extraction (or manifold learning–based dimensionality reduction). Briefly, manifold learning approaches can be categorized into two groups, linear and nonlinear. Traditional approaches, such as principal component analysis (PCA) [7, 8] and linear discriminant analysis (LDA) [2] have been widely used to solve the linear problem. Other linear approaches, such as locality preserving projections (LPP) [6] and its extensions in [24], neighborhood preserving embedding (NPE) [5], are the linearization of their nonlinear counterparts. They have also been widely used due to its inductive nature. Nonlinear approaches assume that the high-dimensional data are nearly lying on a nonlinear low-dimensional manifold. Typical approaches include isomap [15], locally linear embedding (LLE) [13] and its variant in [10], Laplacian eigenmaps (LE) [1], and maximum variance unfolding (MVU) [19, 20]. Although these approaches have achieved prominent performances in many applications, they all try to find low-dimensional embedding for only one high-dimensional representation. In our work, we aim to learn high-dimensional correspondence between different observations.

In the past ten years, some investigations have been dedicated to use manifold learning approaches for high-dimensional correspondence learning, from both the machine learning and computer vision community. Ham et al. [3, 4] have proposed a semi-supervised manifold alignment algorithm in learning high-dimensional correspondence. In their works, some traditional approaches, such as factor analysis, LLE, and LE, are extended. Among them, the extensions of LLE and LE, that is, constraint LLE (CLLE) and constraint LE (CLE), have been widely investigated. In [14], an algorithm based on Gaussian process regression is proposed to learn the shared latent structure. Xiong et al. [22] investigated this problem with loose semi-supervised constraints to align manifolds. More recently, some other researches used procrustes analysis to form a new two-stage algorithm [17]. Verbeek [16] proposed a method to learning nonlinear image manifolds by global alignment of local linear models in the perspective of probability. In [25], Zhai et al. presented a method to learn the explicit corresponding projections from each observation space to the common embedding space. Roscher et al. [12] gave an empirical comparison of graph-based dimensionality reduction algorithms in learning high-dimensional correspondences. In applications, Mehta [11] has extensively investigated these approaches for cross-system personalization. These methods have achieved prominent performances in their related applications. Nevertheless, their performances can also be improved since a more faithful common embedding can be derived by employing another manifold learning approach.

In this paper, by analyzing traditional methods, we provide a common framework to learn high-dimensional correspondences from low-dimensional manifolds. We also plan to leverage a more prominent learning approach for correspondence determination. Motivated by the better performance of MVU [19, 20], we propose a new approach named Local Approximation Maximum Variance Unfolding (LAMVU) for high-dimensional correspondence learning. MVU, which has been extended to its semi-supervised counterpart by combining local approximation technique, is employed to discover the common embedding. Besides, to show the validity, it will be used in a real application task, cross-system personalization (CSP). We enable CSP according to local approximation. Since MVU can derive faithful embedding, LAMVU achieves better results in image alignment and higher accuracy than state-of-the-art learning-based CSP approaches. Plenty of experimental results are provided for demonstration.

The rest of this paper is structured as follows. In Sect. 2, high-dimensional correspondence learning is formulated as a special semi-supervised learning problem. The performances of some manifold learning approaches are also provided. In Sect. 3, we first summarize the framework and then LAMVU is proposed in detail, together with some important discussions. Section 4 is the implementation of our algorithm to enable CSP. Section 5 provides experiments on different kinds of real data sets, followed by the concluding remarks in Sect. 6.

2 Problem formulation and performance comparison

In this section, high-dimensional correspondence learning is reformulated and CLLE as a representative method is introduced. Since the outperformance of LAMVU is due to the fact that MVU could derive faithful embedding, we briefly introduce MVU and compare its performance with that of LLE and LE.

2.1 Problem formulation

For simplicity, we consider the scenario with observations from two aspects, denoted as A and B. Generalization to any arbitrary number can be done as the same strategy mentioned in [11].

Commonly, high-dimensional correspondence learning has two different goals: (1) If we have been told that we have observations of the same objects in both sets, our goal is to match the data. (2) If we only have observations in A (or B), we try to predict the corresponding data in B (or A). This is extremely important for CSP. These two goals can be unified to compute a mapping from one observation set to the other. Since constructing the mapping from B to A is almost the same as computing the mapping from A to B, we only consider the mapping from A to B as an instance.

Mathematically, we assume that the data are stored as vectors $x_i \subseteq \mathbb{R}^n$ in A and $y_i \subseteq \mathbb{R}^m$ in B. Given x_i of an object in A, we need to obtain y_i of the same object in B, that is, we are eager to find a mapping:

$$F_{AB} : \mathbb{R}^n \rightarrow \mathbb{R}^m, \text{ s.t.} F_{AB}(x_i) = y_i. \tag{1}$$

Notice that, if we have N_c data who are known in correspondence in both systems, that is, a data set $\{(x_i, y_i) : i = 1, 2, \ldots, N_c\}$ is available, we can consider this set as training data and this amounts to a standard supervised learning problem. Besides, let X_U and Y_U consist of samples with unknown correspondence. We will use these data since they contain some information that is helpful for learning-based methods. If we regard training set $\{(x_i, y_i) : i = 1, 2, \ldots, N_c\}$ as labeled data and X_U, Y_U as unlabeled data, our problem can be referred to as a special semi-supervised learning problem. Our goal is to predict labels for unlabeled data points. Traditional regression methods are not suitable to handle this problem since the function F_{AB} is vector-valued, while regression problem typically only involves a real valued response variable. Besides, the accuracy of regression model is degraded when N_c is small. For convenience, we collect all data and form the following matrix according to their relationships

$$H = \begin{bmatrix} X_C & X_U & X_D \\ Y_C & Y_D & Y_U \end{bmatrix}, \tag{2}$$

where X_C and Y_C consist of N_c labeled data that are known in one-to-one correspondence in A and B. X_U contains N_A objects whose data only in A are known. Similarly, Y_U are formed by N_B objects whose data only in B are known. X_D represents the corresponding data (in A) of Y_U and Y_D is the predicted examples (in B) whose corresponding data are X_U. Note that X_C and Y_C are stacked by simply connecting the corresponding column of each matrix since they have the same number of columns.

Techniques in manifold learning mainly focus on deriving hidden patterns of the example data. As mentioned in [3], in order to account for correspondences between two high-dimensional data sets, two low-dimensional representations should be equivalent for those examples that are in correspondence. This is mainly due to the fact that the intrinsic structure, such as image variability of objects from different viewpoints or social similarity among people in CSP, can be typically revealed by a low-dimensional manifold structure in the embedded space. Thus, the key idea is to embed different observations into a common low-dimensional space, where two low-dimensional representations will be aligned at the corresponding points. In other words, one needs to find low-dimensional representations of X_C, X_U, Y_C, Y_U and formulate the following matrix L

$$L = \begin{bmatrix} P_C & P_U & P_D \\ Q_C & Q_D & Q_U \end{bmatrix}. \tag{3}$$

Here, $P_C = Q_C$ is the common embedding of X_C and Y_C. P_U is the embedding of X_U and Q_U is the embedding of Y_U. Similarly, P_D and Q_D represent embeddings of X_D and Y_D respectively. In learning-based methods, since $P_C = Q_C$, we aim to construct X_D and Y_D according to the relationship among these low-dimensional embeddings, that is, the elements of L.

Take a representative method CLLE as an example, it first computes the local linear reconstruction weight matrixes W^1 and W^2 on two sets $\{X_C \cup X_U\}$ and $\{Y_C \cup Y_U\}$, respectively. More concretely, we use several nearby points to reconstruct concerning points linearly and assign these weights into a matrix. See more details in [13]. After that, it assumes $P_C = Q_C$ and minimizes the following objective function

$$\begin{aligned} &\min \operatorname{trace}((\hat{P} - W^1\hat{P})^T(\hat{P} - W^1\hat{P}) \\ &\quad + (\hat{Q} - W^2\hat{Q})^T(\hat{Q} - W^2\hat{Q})) \end{aligned} \tag{4}$$

where $\hat{P} = [P_C, P_U]$ and $\hat{Q} = [Q_C, Q_U]$.

Similarly, CLE extends traditional LE and minimizes a function with different similarity matrices W^1 and W^2. Its objective function is

$$\min \operatorname{trace}(\hat{P}L^1\hat{P}^T) + \operatorname{trace}(\hat{Q}L^2\hat{Q}^T) \tag{5}$$

where L^1 and L^2 are two graph Laplacian matrixes determined by W^1 and W^2. The above two problems shown in Eq. (4) and Eq. (5) can be solved by eigen-decomposition of a symmetric matrix. See more details in [3].

2.2 Performance comparison of manifold learning approaches

Since one motivation of our work is the outperformance of MVU in deriving low-dimensional structures, we will first introduce MVU and then compare it with LLE and LE, especially when the sample size is small. This situation is important since it is hard to obtain too many corresponding points in real application. Besides, traditional MVU can only be applied on data of one observation set and it cannot incorporate label information either.

The basic idea of MVU is to maximize the variance of its embedding. It assumes that distances and angles between nearby points are preserved. Low-dimensional coefficients can be considered as several local linear transformations of the input data. There are mainly three stages of MVU. The first step is to construct a neighborhood graph. Then, the constraints that preserve local distances and angles can be formulated as follows

$$\eta_{ij} \parallel p_i - p_j \parallel^2 = \eta_{ij} \parallel x_i - x_j \parallel^2 . \tag{6}$$

Here, p_i is the embedding of x_i. Besides, in order to keep the translation invariable of embeddings and guarantee the identifiable of solution, the outputs are constrained to be centered, that is, $\sum_i p_i = 0$, as in LLE and LE. MVU, which aims to unfold the data by maximizing the variance, that is, $\sum_i \sum_j \|p_i - p_j\|^2$, can be formulated as following optimization problem

$$\begin{aligned} &\max \sum_i \sum_j \|p_i - p_j\|^2 \\ &\text{s.t.}\ \eta_{ij} \parallel p_i - p_j \parallel^2 = \eta_{ij} \parallel x_i - x_j \parallel^2 \cdot \\ &\sum_i p_i = 0. \end{aligned} \tag{7}$$

LLE and LE aim to preserve local similarities in deriving low-dimensional embeddings. They use different methods in characterizing local structures. More concretely, LLE aims to maintain local linear approximating weights while LE tries to keep local similarity defined by different weight matrix. Please see more details in [1] and [13].

To show the effectiveness of MVU in deriving embedding, we compare it with LLE and LE on a toy data set for illustration. The Swiss Roll data [13], which are commonly used to compare the performances of different manifold learning approaches, are employed. Since the former work provides empirical evidence that LLE and LE outperform other approaches, we only compare MVU with them. With two data sets consisting of 500 and 1,000 randomly sampled points, Fig. 1 shows the embeddings derived by LLE, LE, and MVU. Here the parameter k is set as 6 empirically.

As seen from Fig. 1, MVU performs the best in both cases, especially when the sample size is 500. There are folds and overlaps in the embeddings of LLE and LE. Since it is difficult to describe the local geometric structures with low sample density, local approaches, such as LLE and LE, would fail in this situation. On the contrary, MVU aims to unfold the data when local distance is preserved; it still works well and thus effective. This is the motivation why we extend MVU in solving CSP problem.

3 Learning high-dimensional correspondence via manifold learning

In this section, by summarizing a common framework of learning high-dimensional correspondence, we then analyze some traditional approaches within this framework and propose our new method LAMVU. Finally, we analyze the proposed approach briefly.

3.1 Common framework

Recalling the basic idea of traditional high-dimensional correspondence learning methods, we can divide its procedure into two main parts. The first one is to learn the common embeddings of two data sets, with the constraint that low-dimensional representations of ever-known one-to-one corresponding observations are the same. The other step is to assign correspondence based on their low-

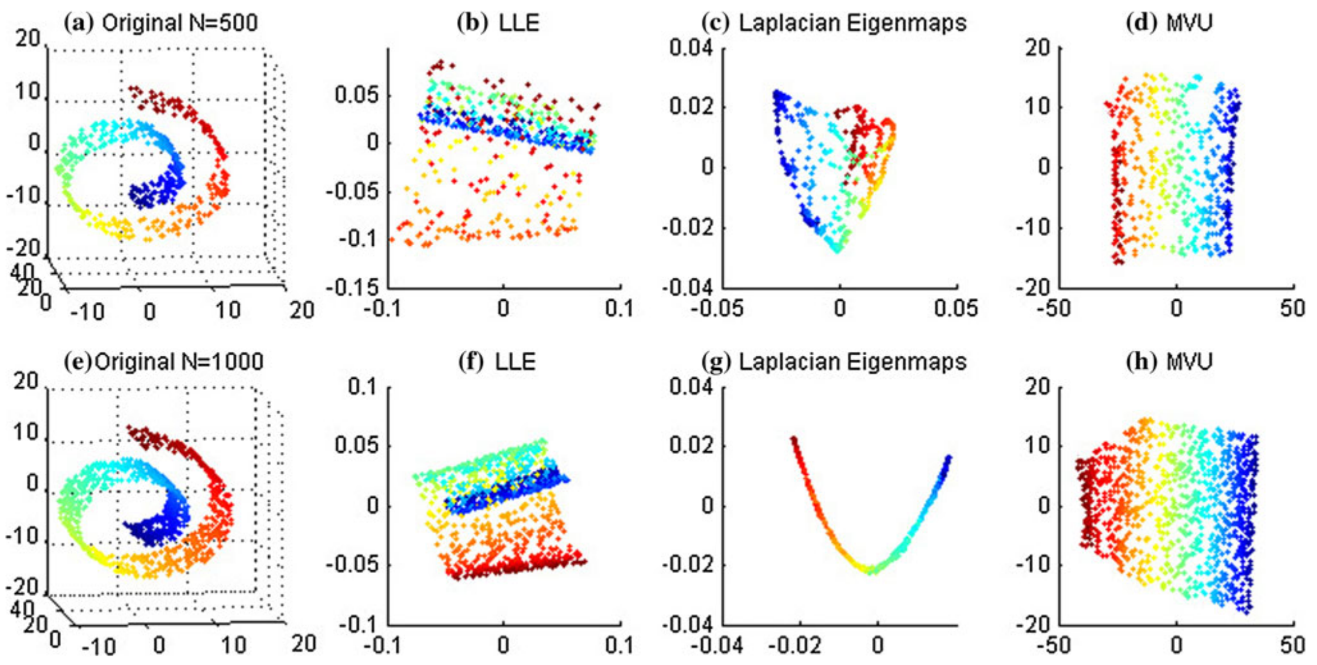

Fig. 1 Performance comparison of LLE, LE, and MVU. **a** Original Swiss Roll data with $N = 500$; **b** LLE embedding with $N = 500$; **c** LE embedding with $N = 500$; **d** MVU embedding with $N = 500$; **e** Original Swiss Roll data with $N = 1000$; **f** LLE embedding with $N = 1{,}000$; **g** LE embedding with $N = 1000$; **h** MVU embedding with $N = 1{,}000$

dimensional embeddings, as the substitution of the corresponding relationship in high-dimensional space.

More concretely, as mentioned above, if we denote the low-dimensional embedding of X_C, Y_C, X_U, Y_U as P_C, Q_C, P_U, Q_U, the first step of our framework amounts to the following optimization problem:

$$\begin{aligned} &\arg \min_{P_C,Q_C,P_U,Q_U} \mathcal{L}(X_C,Y_C,X_U,Y_U,P_C,Q_C,P_U,Q_U) \\ &\text{s.t.}\, P_C = Q_C \end{aligned} \tag{8}$$

Here, $\mathcal{L}$ is the loss function in learning common embeddings. It is commonly designed by combining the objective functions of traditional dimensionality reductions on two data sets.

Take two famous correspondence learning methods CLLE and CLE as examples; the objective function of CLLE is shown in Eq. (4). Thus, within this framework, its objective function can be regarded as

$$\begin{aligned} &\arg \min_{P_C,Q_C,P_U,Q_U} \text{trace}((\hat{P} - W^1\hat{P})^T(\hat{P} - W^1\hat{P}) \\ &+(\hat{Q} - W^2\hat{Q})^T(\hat{Q} - W^2\hat{Q})) \\ &\text{s.t.}\, P_C = Q_C \end{aligned} \tag{9}$$

where $\hat{P} = [P_C, P_U]$ and $\hat{Q} = [Q_C, Q_U] \cdot W^1$ is the local similarity matrix which is determined by X_C and X_U. Similarly, W^2 is computed based on Y_C and Y_U.

As in traditional LE method, CLE uses the Gaussian function to compute the local similarity matrix on two data sets, that is, $\{X_C \cup X_U\}$ and $\{Y_C \cup Y_U\}$, respectively. Denote the computed similarity matrix by W^1 and W^2, CLLE has the following objective function, which can also be unified within the proposed framework

$$\begin{aligned} &\arg \min_{P_C,Q_C,P_U,Q_U} \sum_{p_i,p_j \in \{P_C \cup P_U\}} W^1_{ij}\|p_i - p_j\|^2 \\ &+ \sum_{q_i,q_j \in \{Q_C \cup Q_U\}} W^2_{ij}\|q_i - q_j\|^2 \\ &\text{s.t.}\, P_C = Q_C \end{aligned} \tag{10}$$

Here p_i is low-dimensional embeddings of data $x_i \in \{X_C \cup X_U\}$. q_i is low-dimensional representation of data $y_i \in \{Y_C \cup Y_U\}$ and similarly for others.

As seen from two concrete forms of $\mathcal{L}$ in Eqs. (9) and (10), they are both the sum of objective functions of traditional methods, that is, LLE and LE. One can also extend other dimensionality reduction methods within this framework in the same way.

The second step of our framework is to assign high-dimensional correspondence based on previous derived low-dimensional representations. Mathematically, we want to construct a function f_{AB} as the substitution of F_{AB}. Denote p_x as the embedding of x_i, q_i as the embedding of y_i, we aim to derive the following mapping

$$f_{AB}(p_i) = q_j, \tag{11}$$

and assign that $F_{AB}(x_i) = y_j$.

Formally, we should design another objective function to determine F_{AB} and f_{AB} (they are the same based on above analysis). It has the following form

$$\begin{aligned} &\arg \min_{F_{AB};f_{AB}} \mathcal{G}(X_C,Y_C,X_U,Y_U,P_C,Q_C,P_U,Q_U) \\ &\text{s.t.}\, F_{AB}(x_i) = y_i, \text{ if } x_i \in X_C \text{ and } y_i \in Y_C \\ &\quad f_{AB}(p_i) = q_i, \text{ if } p_i \in P_C \text{ and } q_i \in Q_C \end{aligned} \tag{12}$$

where $\mathcal{G}$ is a loss function which can measure the dissimilarity in determining correspondences. The constraints indicate that the ever-known correspondence relationship should not be changed in this framework.

Note that we have assumed $p_i = p_i$ in Eq. (8). Thus, in the following design of f_{AB}, it is mainly based on the similarity between p_i and q_i. If $p_i = q_i$, we should have $f_{AB}(p_i) = q_i$. One common way in determining f_{AB} is based on a similarity metric of the common low-dimensional embedding. In other words, we assign x_i a correspondence y_j if the embedding of y_j (q_j) is the one that nearest to p_i among $Q_C \cup Q_U$. Let us analyze CLLE and CLE within this framework and show its concrete forms.

CLLE and CLE use the same criterion as mentioned above to determine high-dimensional correspondence. More concretely, they use the following function to determine high-dimensional correspondence

$$\begin{aligned} &F_{AB}(x_i) = y_i, \text{ If } x_i \in X_C \text{ and } y_i \in Y_C \\ &F_{AB}(x_i) = y_l, \text{ If } x_i \in X_U \text{ and } \\ &l = \arg \min_j \|p_i - q_j\|, q_j \in \{Q_C \cup Q_U\} \end{aligned} \tag{13}$$

In next subsection, we will provide new formulations of $\mathcal{L}$ and $\mathcal{G}$, which enable MVU to learn the common embeddings of two data sets simultaneously.

In summary, the above framework can be summarized into the procedure shown in Fig. 2. As seen from Fig. 2, data in the top and bottom tables are in original space and examples in the middle tables are their low-dimensional representations. Examples in two columns of each subfigure are assigned according to their corresponding relationships.

3.2 LAMVU

Considering that learning approaches aim to discover the hidden consensus patterns and MVU can derive these patterns faithfully, we propose a new correspondence learning method, LAMVU, within the proposed framework. We first extend MVU to derive the common embedding of labeled examples. Then, low-dimensional representations of unlabeled data are computed by local

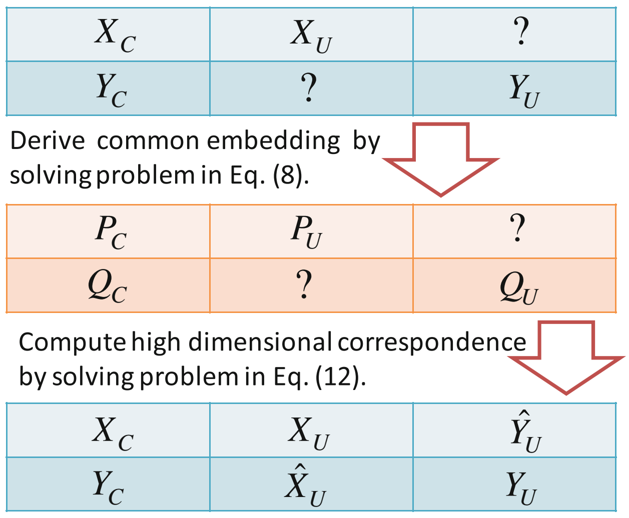
X_C
X_U
?
Y_C
?
Y_U
Derive common embedding by solving problem in Eq. (8).
P_C
P_U
?
Q_C
?
Q_U
Compute high dimensional correspondence by solving problem in Eq. (12).
X_C
X_U
$\hat{Y}_U$
Y_C
$\hat{X}_U$
Y_U

Fig. 2 The proposed framework

approximation. Finally, correspondences of unknown points are assigned or predicted. In brief, the first two steps are the implementation of the first procedure of our framework and the last just aims to determine suitable $\mathcal{G}$. Let us introduce them sequentially under guidance of the proposed framework.

3.2.1 *Derive common embeddings of labeled examples by MVU*

Recalling the basic assumption that both X_C and Y_C are representations of the same hidden structure, we would like to use MVU in deriving this common structure. Nevertheless, as X_C and Y_C are collected from different aspects, there is no numerical relationship between them. For example, one observation may be much larger than that in the other. Besides, they may use different metrics to measure distances. Fortunately, they are all representations of the same structure. More concretely, the local relationship between any two points in X_C should be similar to that of the corresponding points in Y_C. Otherwise, the hidden consensus patterns that they revealed may be different. Manifold learning approach, that is, MVU, should maintain this similarity in deriving the hidden consensus pattern.

To make the distance computed from two data sets comparable and the embedding identifiable to transformation, we would like to use the following two strategies for preprocessing. The first is normalization. We normalize each data set using the following method. For $x_i \in X_C \cup X_U, y_j \in Y_C \cup Y_U$,

$$\begin{aligned} &x_i \leftarrow x_i/s, \text{ where } s = \max\{\|x_j\| | x_j \in X_C \cup X_U\}, \\ &y_i \leftarrow x_i/t, \text{ where } t = \max\{\|y_j\| | y_j \in Y_C \cup Y_U\}. \end{aligned} \quad (14)$$

where $\|\cdot\|$ represents the F-norm of a vector. Note that, through this kind of normalization, the points, which have the largest norms in their own set, both have the norm 1. Thus, the computed distances are comparable in data scale.

The second strategy is centralization. Since two data sets are normalized, we centralize two data sets by their own centers, respectively. That is to say, for $x_i \in X_C \cup X_U, y_j \in Y_C \cup Y_U$,

$$\begin{aligned} &x_i \leftarrow x_i - c, \text{ where } c = \frac{\sum_{x_j \in X_C \cup X_U} x_j}{\#(X_C \cup X_U)}, \\ &y_j \leftarrow y_j - d, \text{ where } d = \frac{\sum_{y_i \in Y_C \cup Y_U} y_i}{\#(Y_C \cup Y_U)}. \end{aligned} \quad (15)$$

where $\#(\cdot)$ means the cardinality of a set.

After this important preprocessing, we take the weighted distance matrix shown in the following as the input of MVU

$$\lambda \text{dis}(X_C) + \mu \text{dis}(Y_C). \quad (16)$$

Here, $\text{dis}(X_C)$ is a matrix whose elements are geodesic distances between points whose representations are column vectors of preprocessed X_C and similarly for $\text{dis}(Y_C)$. As in Isomap [15], these distances are more accurate than the Euclidean distance to characterize the structure of data manifold. They can be approximated by the shortest path distances on the constructed neighborhood graph, whose vertexes combine the points represented by both X_C and X_U (see the introduction about the first step of MVU in Sect. 2.2). The induction of unlabeled points will make the computation of geodesic distances more accurately [15]. λ and μ are two turning weights which can balance the effects of two distance matrixes. For convenience, we take $\lambda = 1/\max(\text{dis}(X_C))$ and $\mu = 1/\max(\text{dis}(Y_C))$. Here, max(-dis($X_C$)) is the maximum value of the elements in $\text{dis}(X_C)$ and similarly for $\max(\text{dis}(Y_C))$.

Take the weighted distance matrix shown in Eq. (16) as the input, we derive the common representations $P_C = Q_C$ of labeled data by employing MVU. It combines the prior knowledge in learning procedure and derives an identical representation for the corresponding data. They also reveal the intrinsic structures of two data sets and maintain their corresponding relationships. For visualization, we will show some two-dimensional embeddings in Sect. 5.

3.2.2 *Learn low-dimensional embedding of unlabeled examples*

In last stage, we only derived the common embedding. As mentioned in above framework, we also need to compute the embedding P_U and Q_U. Considering the fact that any smooth manifold can be linearly approximated in local regions, we employ local approximation technique as in [13] to derive embedding of unlabeled data, that is, X_U and Y_U. As the procedure is similar for both systems, we only take data from the first system as an example.

The first step is to identify local regions for each unlabeled data in X_U. We choose a fixed number, k_D, of nearest points in X_C for each unlabeled data. In other words, neighbors of $X_U\ (i)$, for $i = 1, 2, \ldots, N_A$, are selected only in X_C. Here, $X_U\ (i)$ denotes the data point represented by the ith column of X_U.

The second step is to compute the local combination weights. Assume that W is a $N_C \times N_A$ weight matrix. The ith column of W consists of reconstruction weights for $X_U\ (i)$. W_{ji} represents the contribution of the jth labeled data $X_C(j)$ to the reconstruction of the ith unlabeled data $X_U\ (i)$. $X_C\ (j)$ is the data point shown in the jth column of X_C. If $X_C(j)$ is not a neighbor of $X_U\ (i)$, then $W_{ji} = 0$. Meanwhile, in order to guarantee that the weights are invariant to rotations, rescaling, and translations, the sum of every column vector of W is equal to one, that is, $\sum_{j=1}^{N_C} W_{ji} = 1$, for $i = 1, 2, \ldots, N_A$. Thus, W can be derived by minimizing the following equation, which can be solved by constrained least square regression

$$\begin{aligned} &\arg\min \epsilon(W) = \sum_{i=1}^{N_A} \left\| X_U(i) - \sum_{j=1}^{N_C} W_{ji} X_C(j) \right\|^2 \\ &\sum_{j=1}^{N_C} W_{ji} = 1 \text{ for } i = 1, 2\ldots, N_A \\ &W_{ji} = 0 \text{ If } X_C(j) \text{ is not a neighbor of } X_U(i). \end{aligned} \tag{17}$$

Finally, by using this reconstruction weight matrix, we can calculate the embedding of X_U. These results are shown in the following equation

$$P_U = P_C \times W. \tag{18}$$

Similarly, we can apply the same technique to derive embedding of Y_U, that is, $Q_U = Q_C \times U$. Here, U is a $N_C \times N_B$ weight matrix, which can be calculated in the same way as computing W. All data in the low-dimensional space are listed in the following matrix

$$L = \begin{bmatrix} P_C & P_U = P_C \times W & P_D \\ Q_C & Q_D & Q_U = Q_C \times U \end{bmatrix}. \tag{19}$$

By now, we have finished the first stage of our framework. We will assign correspondences for unlabeled data based on the relationships among the elements of L.

3.2.3 *Assign correspondences for unlabeled examples*

As mentioned above, there are two tasks of learning high-dimensional correspondence, that is, to predict the data when they only known in A or to align the data when they appear in both observations. Considering the second stage of proposed framework, we will show how to achieve these goals, respectively.

We first employ the linear approximation technique to compute the predictions, that is, the approximated mapping results of X_U and Y_U (Eq. 21) and then assign the correspondences based on the metric proposed in our framework.

For illustration, take a particular point $X_U(i)$ as an example, its low-dimensional representation is $P_U(i)$. Since $P_C = Q_C$ and P_U, Q_U are derived by P_C and Q_C, we can linearly approximate $P_U(i)$ by finding its k_A nearest neighbors in the data set formed by the combination of Q_C and Q_U. Assume that ω is an $(N_C + N_B) \times 1$ vector whose elements are the approximating coefficients and computed by the minimization of following equation

$$\begin{aligned} &\arg\min \left\| P_U(i) - \sum_{i=1}^{N_C} \omega(i) Q_C(i) - \sum_{i=1}^{N_B} \omega(i + N_C) Q_U(i) \right\|^2 \\ &\sum_{i=1}^{N_C+N_B} \omega(i) = 1 \\ &\omega(i) = 0, \text{ if } Q_C(i) \text{ is not a neighbor of } P_U(i) \\ &\omega(i + N_C) = 0, \text{ if } Q_U(i) \text{ is not a neighbor of } P_U(i). \end{aligned} \tag{20}$$

Here, ω(i) is the ith element of ω. Weights for points that are beyond the k_A nearest neighbors of $P_U(i)$ are defined as zeros.

Consequently, we can define the prediction $\hat{Y}_D(i)$ of $X_U(i)$ as follows

$$\hat{Y}_D(i) = \sum_{i=1}^{N_C} \omega(i) Y_C(i) + \sum_{i=1}^{N_B} \omega(i + N_C) Y_U(i). \tag{21}$$

We now explain how to achieve the above two goals.

1. To deal with the missing value situation, that is, to establish data of a data which appear only in system A, we directly assign $\hat{Y}_D(i)$ as its correspondence in B.
2. If we have been told that the corresponding data of $X_U(i)$ belong to either Y_C or Y_U, it is straightforward to assign the point which is nearest to $\hat{Y}_D(i)$ as its matching example.

Similarly, we can also compute the prediction $\hat{X}_D$ of Y_U in the same way. The matrix H in original space can be reformulated according to their corresponding relationships

$$H = \begin{bmatrix} X_C & X_U & \hat{X}_D \\ Y_C & \hat{Y}_D & Y_U \end{bmatrix}. \tag{22}$$

In summary, the procedure of LAMVU is in Table 1.

3.3 Discussion

In this subsection, we first compare LAMVU with CLLE and CLE within the proposed framework. Then, some discussions about LAMVU are provided.

As seen from the proposed framework and above methods, we can see that the performance of embedding

Table 1 LAMVU Algorithm

```
Step 1 Derive common embedding of labeled points
Input: Data which are known in correspondence $X_C$, $Y_C$;
Output: Common embedding of $X_C$: $P_C$ and $Y_C$: $Q_C$, such that
    $P_C = Q_C$;
1. Compute geodesic distance matrixes dis($X_C$) and dis($Y_C$) of $X_C$
    and $Y_C$ respectively;
2. Compute the weighted matrix $\lambda$dis($X_C$) + $\mu$ dis($Y_C$);
3. Employ MVU to derive common embedding $P_C$ and $Q_C$, such
    that $P_C = Q_C$;
Step 2 Learn embeddings of unlabeled examples
Input: Labeled data $X_C$, $Y_C$, unlabeled data $X_U$, $Y_U$, common
    embedding $P_C$, $Q_C$;
Output: Embeddings of $X_U$ and $Y_U$, that is, $P_U$, $Q_U$;
1. Compute approximation matrixes $W$ and $U$ by Eq. (17);
2. Assign $P_U = P_C \times W$, $Q_U = Q_C \times U$;
Step 3 Assign correspondences for unlabeled data
Input: Labeled data $X_C$, $Y_C$, unlabeled data $X_U$, $Y_U$, embeddings
    $P_C$, $Q_C$, $P_U$, $Q_U$;
Output: Correspondences of unlabeled examples;
1. For each point in $X_U$ and $Y_U$, compute the local linear
    approximation weights in embedding space by Eq. (20);
2. Compute predictions $\hat{Y}_D, \hat{X}_D$ by Eq. (21);
3. (a) For predicting, assign $\hat{Y}_D$ as prediction of $X_U$, assign $\hat{X}_D$ as
    prediction of $Y_U$;
   (b) For matching, assign the point that is nearest to the
    prediction as its matching point.
```

learning is vital. As mentioned in Sect. 2, since MVU can derive more faithful embedding, LAMVU should outperform CLLE and CLE. More concretely, when CLLE and CLE are employed to solve this problem, they violate the basic assumptions of these methods. In other words, the local similarity that they want to keep cannot be maintained when we add the constrain $P_C = Q_C$. They fail to describe the local properties of data manifold in solving this problem. LAMVU, however, also follows the geometry intuition of MVU, that is, unfolding the manifold while preserving local distances.

The second problem is the similarity and dissimilarity among three methods. As shown in our framework, all the methods follow the same procedure in learning high-dimensional correspondence. Moreover, comparing Eq. (13) with that in formulating Eq. (22), in the second step of our framework, three methods use the same criterion in determining high-dimensional correspondence. Their main differences are the concrete formulations of $\mathcal{L}$ in Eq. (8). As we have known, CLLE and LE are all distance based. Since MVU could use distance in a more direct way, it is no surprise that it performs better than CLLE and CLE.

The third problem is about the computational complexity. Denote N_c as the number of corresponding points, N_x and N_y as the numbers of unlabeled points in A and B, that is, $N_x = \#(X_U)$, $N_y = \#(Y_U)$. As seen from above procedure, the most computational step of CLLE and CLE is the decomposition of the Laplacian matrix. Thus, their computational complexity is $O((N_c + N_x + N_y)^3)$. LAMVU costs the most time in deriving the common embedding by MVU. Essentially, it is formulated as a semi-definite program, which can be solved by other approaches, such as CSDP. Its computational complexity is $O(N_c^3 + \theta^3)$, where θ is the number of constraints on the constructed graph as shown in Eq. (6). Nevertheless, its computational complexity is still high. We can apply the accelerating method proposed in [21].

Finally, we would like to show some shortcomings of the proposed framework and method. Since we have restricted that embedding of corresponding points should be the same, this seems to strict in real applications. We will relax this constraint by assuming that there is a linear transformation between them in the further researches. Besides, LAMVU is not as fast as CLLE and CLE. Meanwhile, as in most learning-based methods, LAMVU assumes that the embedding of two data sets must have the same dimensionality. This may be also too strict in applications. Additionally, all parameters are chosen experientially; we will investigate an effective rule to determine parameters.

4 CSP, a concrete application

In this section, we will briefly explain how to use the methods within our framework in an important application CSP. We will combine the separated personalization information within each system to provide an enhanced personalization experience.

As mentioned in [11], CSP can be regarded as a typical corresponding problem. If we take the user data from two different systems, also denoted as A and B, as the observations from different aspects, personalization of a system amounts to learning the correspondences between them. More concretely, (1) if we have been told that the user u has visited both systems, CSP is to match the data of u in both systems. (2) If the user u is known to only use system A (or B), CSP tries to predict his (or her) data in B (or A).

We would like to explain why we can use the methods within our framework for CSP. When it is applied to personalization, our method can learn the social similarity between people by deriving the common embedding. There are mainly two reasons: (1) There is enough regularity among users that one can learn within a user population. This fact is commonly exploited in social research or collaborative filtering. This regularity is represented by the consensus pattern or common embedding learned by manifold learning approach. (2) When a person uses multiple systems, there is likely to be a lot of consistencies between data of the same person. Intuitively, these consistencies can be regarded as the tastes of a user. Thus, the performance of embedding learning is also vital for CSP. As explained in Sect. 3.3, LAMVU should also outperform CLLE and CLE in CSP.

Considering the above analyses, we can directly use the correspondence learning methods, such as LAMVU, CLLE, and CLE, for CSp. Traditionally, there are two commonly used non-cross-system personalization methods, the *Popular Votes* and *Mean Votes*. In Brief, *Popular Votes* recommends any user with the most popular votes and *Mean Votes* provides the average votes of ever-known labeled points. They are nonpersonalization recommendations. Compared with these methods, the learning-based methods within our framework incorporate their common relationship between two systems. Thus, they should perform better. In next section, we will compare their performances.

Finally, we would like to introduce an evaluation metric which is commonly used to measure the performance of CSP algorithms. We employ modified mean average error (MMAE) as an evaluation metric [11]. It can be computed according to the following formulation

$$\mathrm{MMAE} = \frac{1}{q}\sum_{i=1}^{q}\frac{1}{m(i)}\sum_{j\in\Gamma(i)}|p_j^i - a_j^i|. \tag{23}$$

In Eq. (23), q is the number of evaluation examples. a_j^i is the jth element of the ith real data, that is, p_j^i represents the jth element of the ith predicted data p^i. $\Gamma(i) = \{j|a_j^i > 0\}$ and $m(i)$ is the number of nonzero values provided by the ith user, that is, $m(i) = \#(\Gamma(i))$.

Note that, since the data are often very sparse, the second sum is taken only over known values, that is, $j \in \Gamma(i)$. If the sum is taken over all items, the distance between different points will trend to equal. For example, assume a_1 is a D-dimensional vector whose first element is one and other elements are zeros. a_2 is also a D-dimensional vector whose elements are all zeros. If the sum is taken over all elements, their distance is $1/D$, which is near zero when D is sufficiently large (it is common in CSP). By employing MMAE, however, their distance is one. We can easily separate these two data sets by MMAE since they are different in essence in our applications.

5 Experiments

In this section, we mainly provide two groups of experiments. The first group is to show the image alignment results intuitively. This is important for pattern recognition and computer vision. The second group contains CSP experimental results on several real data sets. They are the teapot data set [3], the hand data set[1], the created Lenna and Earth data sets[2], two MovieLens data sets[3], the Jester data set[4], and the Book Crossing data set[5]. The first four are employed to show the results intuitively, and the last three are commonly used to evaluate the performance of different learning-based CSP approaches. Since it has been shown that CLLE achieves the best performance, we only compare LAMVU with CLLE for visualization. Other two commonly used methods, *Popular Votes* and *Mean Votes*, are also employed for CSP.

5.1 Experiments on the rotated images

As seen from the procedure of our proposed framework in Table 1, the derivation of consensus pattern is vital for high-dimensional correspondence learning. In this section, we propose two examples on two rotated image sets. The first one aims to learn correspondence between images of a rotated teapot data set and a rotated hand data set. They contain 200 and 481 images, respectively. These images have been arranged according to their angles of rotation.

In our experiments, we select 41 training images based on the angels of rotations uniformly. In other words, we select images rotated from left to right with equal interval. The low-dimensional embeddings, derived by CLLE and LAMVU, are shown in Fig. 3. Together with them, we have also selected some typical images. They are drawn according to their embedding. Other parameters are empirically determined. Notice that the horizontal axis represents the data's location in the data set. For example, the x-axis for the first image is 1. The y-axis represents the one-dimensional embedding derived by different methods.

As seen from the results in Fig. 3, the results of LAMVU (Fig. 3c,d) are very similar. The embeddings of CLLE in Fig. 3a, b, however, have large variance with each other. It indicates that it is more accurate in using our method to align high-dimensional data.

In the second experiment, we use the created Lena and earth image data sets. The images of Lena were generated by moving the Lena image of 32×32 pixels on a noisy background of 61×61 pixels. We moved one pixel at a time, and the representing data set is of $61 \times 61 = 3721$ dimensionality. The second data set was 100×100 images of the earth rendered by rotating its azimuthal and elevation angles. It

[1] http://vasc.ri.cmu.edu/idb/html/motion/hand/.

[2] http://www.seas.up-enn.edu/jhham/.

[3] http://www.grouplens.org.

[4] http://goldberg.berkeley.edu/jester-data/.

[5] http://www.informatik.uni-freiburg.de/cziegler/BX/.

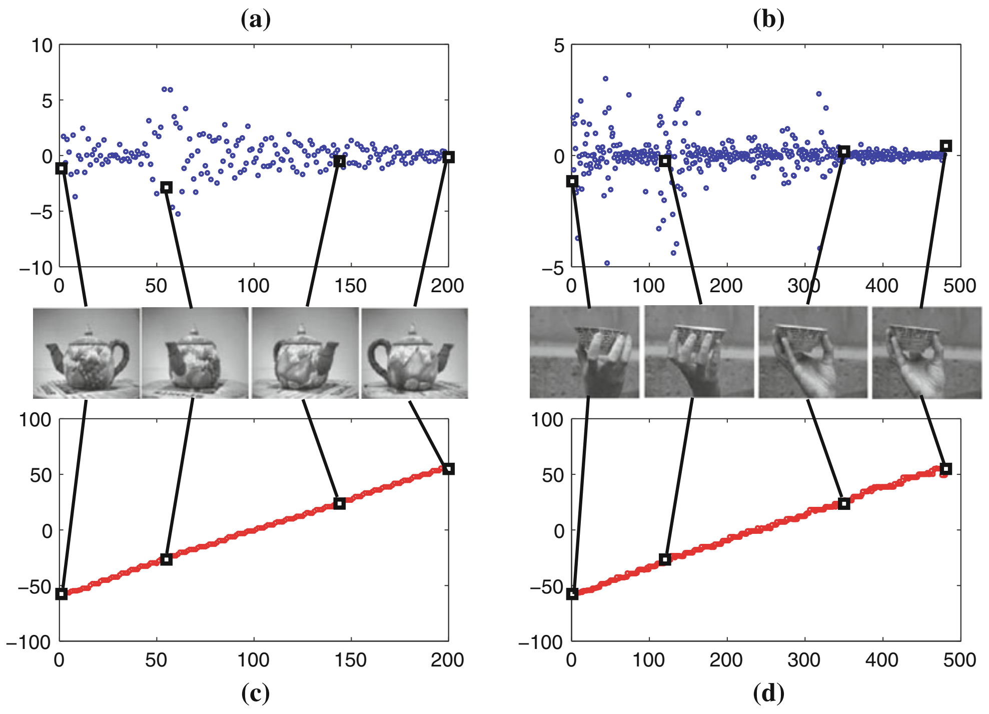

Fig. 3 Aligned real images of Teapot and Hand data sets. **a**, **b** Are embeddings derived by CLLE. **c**, **d** Are embeddings derived by LAMVU. Some typical images are also shown

contains -45 to $45°$ of azimuth and -45 to 45 degree for elevation changes. Both of them consist of 900 points.

In our illustrations, to show the effectiveness of our method, we take some aligned images determined by LAMVU. The correspondence results are shown in Fig. 4. It implies that the up–down moving of the Lena image aligns with the elevation changes and the left-right transfer corresponds to the azimuthal movements. It means that our method could reveal the consensus hidden pattern of two different sets efficiently.

5.2 Experiments on MovieLens

To show the effectiveness of the proposed approach for CSP, we present several results on real data set, that is, the MovieLens data. It contains two subsets with different size. For simplicity, we call them the MovieLens-I and the MovieLens-II.

The MovieLens-I consists of 100,000 ratings for 1682 movies by 944 different users. The rating concerning each movie is referred as one dimension of the data set. For example, the first row of the data file is “196 242 3 881250949.” It means that the 196th user rates the 242nd movie with the score 3 (5-star scale). The final number is a time represented in seconds since the epoch is returned by time. If a user does not rate a movie, the corresponding score is zero. Since every user only rates limited number of movies, these data are very sparse (about 100,000/(1682 $\times$ 944) $\approx$ 0.063). As the experiments in previous papers [11], the users are deemed as the units of analysis, for which there are two different subsects, that is, one’s rating of two sets of items. Those ratings are randomly split into two subsets X (840 $\times$ 944) and Y (842 $\times$ 944).

The MovieLens-II data set consists of approximately 1 million ratings for 3952 movies by 6040 users. It has the same structure as MovieLens-I, except that this data set is more sparse (about 1, 000, 000/(3900 $\times$ 6040) $\approx$ 0.042). In our experiments, we select the ratings provided by the first 1000 users. The same as previous, we randomly split those ratings into two subsets X (2000 $\times$ 1000) and Y (1952 $\times$ 1000).

In principle, we can vary the overlaps between two data sets from no overlap to all items’ overlapping. However, in real-world scenario, items’ overlaps are very small. We choose no overlaps in our experiments. Another important parameter is the size of labeled examples; we vary this parameter ranging from 20 to 200 in our experiments. Ratings of the last 143 users form the unlabeled data sets. Labeled data sets are randomly chosen for every run. In these settings, the data sets are sparse. Therefore, Euclidean distance on pure data is not necessarily effective. We simply apply the cosine distance, which is more suitable to measure the distance between sparse samples.

Some parameters need to set in advance. These parameters are (1) dimensionality of the embedding space, that is, d; (2) neighborhood size in deriving the common

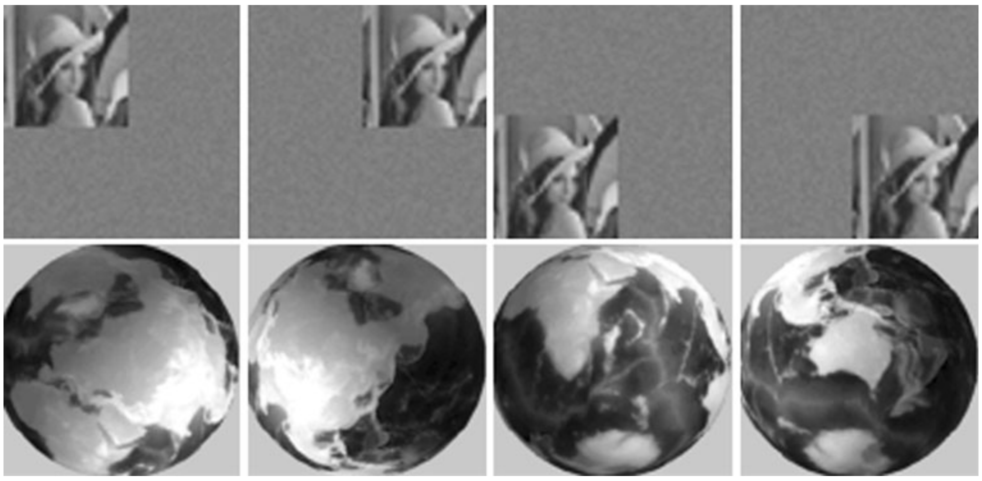
Fig. 4 Aligned real images of the Lena and Earth data sets

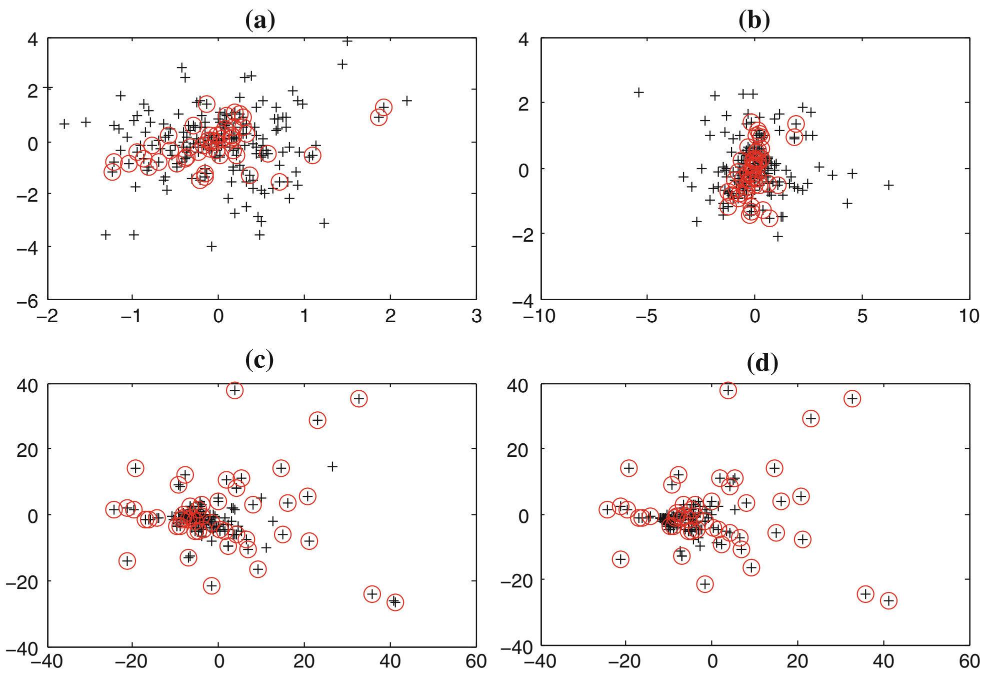

Fig. 5 Two-dimensional embedding derived by CLLE and LAMVU of 50 labeled points and 143 unlabeled samples on the MovieLens-I data set. **a** CLLE on the first data; **b** CLLE on the second data; **c** LAMVU on the first data; **d** LAMVU on the second data

embeddings, that is, k; (3) neighborhood size in learning the embeddings for unlabeled data, that is, k_D; (4) neighborhood size in computing the predictions, that is, k_A. As shown in the following experiments, these parameters have small influence on the performance of LAMVU. Thus, we choose them experientially.

In order to compare LAMVU with CLLE intuitively, we draw two-dimensional embeddings of MovieLens-I, which consist of 50 labeled pairs and 143 unlabeled pairs, in Fig. 5. Representations of labeled pairs have been marked by circles. The parameters are as follows: $d = 2$, $k = 8$, $k_A = 6$ for both methods and $k_D = 6$ for LAMVU.

As seen from Fig. 5, it seems that low-dimensional embeddings, which are derived by our method, have similar shapes. Besides, embeddings of unlabeled points look like to have the same shape for two bottom figures. Therefore, it is more reasonable for LAMVU to use nearest neighbors to determine corresponding data. On the contrary, embeddings derived by CLLE have significant differences. Except for the same embeddings of X_C and Y_C, there are large variances between them. Therefore, LAMVU outperforms CLLE intuitively.

We also compare LAMVU with CLLE, *Popular Votes* and *Mean Votes* quantitatively. With each fixed number but randomly sampled label points, we repeat every algorithms 50 times and compute their average MMAEs. The results are drawn in Fig. 6. Parameters are the same as that in the first experiment except for $d = 15$ in these experiments.

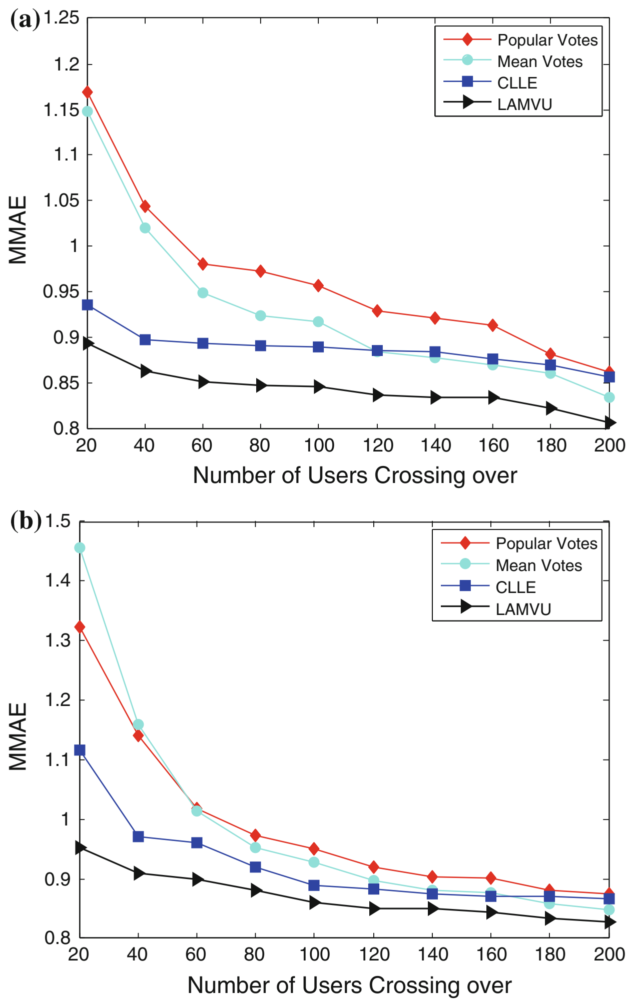

Fig. 6 Precision for different methods on the MovieLens-I and MovieLens-II data set within the manifold alignment framework. The x-axis is the number of users crossing over, that is, N_c. The y-axis represents the mean values after 50 times repeat. **a** Performance comparison on MovieLens-I; **b** Performance comparison on MovieLens-II

As seen from Fig. 6, compared with other three approaches, LAMVU has the smallest MMAEs in all cases. Concretely, it is about 5 % smaller than CLLE and 17–23 % smaller than nonpersonalization methods. More interesting, if we compare the results in Fig. 6b with those in Fig. 6a, it seems that all the methods perform worse in the same situation if the data are more sparse (MovieLens-II). This may be caused by the inaccurate determination of distance for very sparse data, although we have used cosine metric to compute the distance.

Besides, since LAMVU can reveal the intrinsic structure with a small percent of labeled points, a significant advantage is that LAMVU performs better than other approaches even with small N_C. This is very important since labeling points is very expensive, especially in CSP, due to the violation of privacy.

5.3 Experiments on Jester

In order to show the advantage of LAMVU, we also present experimental results on another type of real data set, that is, the Jester data. It consists of 4.1 million continuous ratings (ranging from -10.00 to $+10.00$) of 100 jokes from 73,421 users. It has the same data structure as MovieLens and contains three parts. We randomly choose 983 users' ratings from the first part for evaluation. The users are also deemed as the units of analysis, and those ratings are also randomly split into two subsets X (40×983) and Y (60×983) with no overlaps.

Since the element of this data is decimal, it is difficult to employ the *Popular Votes*; we only compare LAMVU with CLLE and the *Mean Votes*. Similarly, the last 183 samples are unlabeled. The number of labeled pairs varies from 100 to 400. Experiments have been repeated 50 times for each fixed number of points. Mean and standard derivation (number within the bracket) of these MMAEs are shown in Table 2. Parameters are as follows: $d = 15$, $k = 8$, $k_A = 16$ for both methods and $k_D = 16$ for our method. Results with statistical significance are indicated in bold.

As seen from the results, LAMVU outperforms CLLE and the *Mean Votes* in all cases. Our algorithm provides about 11–17 % improvement over CLLE and the *Mean Votes*. Similar to the results shown in Fig. 6, LAMVU achieves encouraging results even when labeled samples are rare. Hence, LAMVU can achieve higher accuracy than CLLE and the *Mean Votes* under pre-given privacy request.

5.4 Experiments on Book Crossing

To show the validity of LAMVU on sparse samples, we choose another benchmark data set, the Book Crossing data. It contains 278,858 users (anonymous but with demographic information) providing 1,149,780 ratings (explicit/implicit) about 271,379 books. We use the book rating information, whose ratings are either explicit, expressed on a scale from 1–10 (higher values denoting higher appreciation), or implicit, expressed by 0. We eliminate the implicit rating and select a subset of users whose have provide more than 10 ratings and a subset of

Table 2 Mean and standard diversity of MMAE on Jester Jokes data

	Mean votes	CLLE	LAMVU
$N_C = 100$	4.3931 (0.1935)	4.2983 (0.1104)	**3.5854** (0.1532)
$N_C = 150$	4.2654 (0.1697)	4.0106 (0.0828)	**3.5825** (0.0464)
$N_C = 200$	4.1426 (0.1189)	4.0053 (0.0172)	**3.5469** (0.0752)
$N_C = 250$	4.0592 (0.1027)	3.9280 (0.0604)	**3.5078** (0.0356)
$N_C = 300$	3.9423 (0.1150)	3.8940 (0.0278)	**3.4510** (0.0588)
$N_C = 350$	3.8196 (0.1045)	3.8628 (0.1112)	**3.4191** (0.0648)
$N_C = 400$	3.7273 (0.1110)	3.7288 (0.0596)	**3.3882** (0.0664)

Table 3 Mean and standard diversity of MMAE on Book Crossing data

	Mean votes	Popular votes	CLLE	LAMVU
$N_C = 100$	5.7327 (0.3535)	5.7684 (0.3497)	3.2165 (0.2068)	**2.8063** (0.1160)
$N_C = 150$	5.1214 (0.1975)	5.3571 (0.1564)	2.7794 (0.1536)	**2.2836** (0.1000)
$N_C = 150$	4.7496 (0.0635)	4.8237 (0.0912)	2.5480 (0.1168)	**1.8853** (0.1046)
$N_C = 150$	4.4727 (0.1114)	4.7621 (0.1522)	2.3706 (0.1563)	**1.7186** (0.1102)
$N_C = 150$	4.2932 (0.0830)	4.5735 (0.1332)	2.3659 (0.1292)	**1.6118** (0.1224)
$N_C = 150$	4.1720 (0.0807)	4.3416 (0.1030)	2.3295 (0.0705)	**1.5289** (0.0878)
$N_C = 150$	4.0395 (0.1018)	4.1539 (0.0517)	2.2732 (0.1396)	**1.4811** (0.0948)

books who have received the most 1000 ratings. The volume of data is 2014. Although we have made some preprocessing, the data are also very sparse (about 30000/(1000 $\times$ 2014)$\approx$ 0.015).

Similarly, we randomly split these data into two sets with X (500 $\times$ 2014) and Y (500 $\times$ 2014) with no overlaps. The last 114 points are fixed as testing samples, and the other settings are the same as previous. With different number of training point, we compare LAMVU with other methods and provide the mean and standard diversity of MMAE for 50 independent runs. The results are listed in Table 3. Results with statistical significance are indicated in bold.

As seen from Table 3, we can draw almost the same conclusion as in previous experiment. LAMVU performs the best, even when the data are sparser. This often occurs in real application since getting voting information is cost consuming.

5.5 Discussions of experiments

We would like to provide some important discussions about LAMVU.

As seen from Tables 2, 3, and Fig. 6, the prediction accuracy becomes higher when more labeled examples are available. It consists with prior knowledge since more information is provided. More importantly, LAMVU does not so heavily depend on the number of labeled pairs, that is, N_C, as other learning-based methods. LAMVU has a significant advantage in achieving high accuracy, especially when the labeled samples are rare.

There are totally four parameters in LAMVU. They have been chosen empirically. Fortunately, it seems that when they are within certain ranges, LAMVU is not very sensitive to these parameters. For example, we have done experiments with different d, that is, the dimensionality of embedding space and k, the size of neighborhood in deriving the consensus low-dimensional embedding.

With each fixed d, we determine other parameters as previous and repeat every experiments 50 times on three data sets. The average MMAEs are shown in Table 4.

As seen from the results in Table 4, when d is within a certain range, LAMVU is robust to d. There are mainly two reasons. (1) The common low-dimensional embeddings of labeled points are derived by kernel matrix factorization and the most d significant eigenvectors are selected. When d is suitable, these eigenvectors are enough to express the structures of the embedding manifolds. The performance of LAMVU does not change too much with the increase of d. (2) Since representations of unlabeled points are linearly constructed by the embeddings of labeled points, they are not sensitive to d either.

We have also evaluated our algorithm with different neighborhood sizes in deriving common low-dimensional embedding, that is, k. There are also 50 experiments for each fixed k. Mean MMAEs are shown in Table 5.

As seen from the results in Table 5, when k is within a certain range, the accuracy of LAMVU has small changes. This is because: (1) The determination of k reflects the extent to which the local similarity should be maintained. When k is within a certain range, we keep similar local information. Thus, the embedding derived by LAMVU does not change heavily. (2) The mean imputation in the first data tends to change distance between any two points. It can also reduce the influence of k.

Table 4 Precision of LAMVU on the Movielens and Jester data sets with different d

	$d = 5$	$d = 10$	$d = 15$	$d = 20$	$d = 25$
MovieLens-I	0.860	0.861	0.860	0.860	0.859
MovieLens-II	0.879	0.877	0.880	0.878	0.878
Jester	3.682	3.643	3.642	3.651	3.702

Table 5 Precision of LAMVU on the MovieLens and Jester data sets with different k

	$k = 6$	$k = 8$	$k = 10$	$k = 12$	$k = 16$
MovieLens-I	0.856	0.862	0.869	0.860	0.860
MovieLens-II	0.873	0.875	0.879	0.878	0.880
Jester	3.597	3.644	3.657	3.657	3.647

6 Conclusion

This paper outlines a framework and novel approach LAMVU to leverage data distributed in different aspects for high-dimensional correspondence learning. It mainly focuses on how to unify previous approaches and increase the accuracy of learning-based approaches. MVU is extended within this framework to derive the common embedding, and local approximation strategy is used to learn the representations of unlabeled examples. Corresponding data of unknown points can be assigned according to the relationships between these representations. Compared with other learning-based methods, LAMVU can achieve higher accuracy, even with few labeled examples. It is of high potential in real applications, such as CSP.

Future work includes how to effectively determine the parameters and the accelerating issue of proposed method.

References

1. Belkin M, Niyogi P (2003) Laplacian eigenmaps for dimensionality reduction and data representation. Neural Comput 15(6):1373–1396
2. Fukunaga K (1990) Introduction to statistical pattern recognition, 2nd edn. Academic Press, Boston
3. Ham JH, Lee DD, Saul LK (2003) Learning high dimensional correspondences from low dimensional manifolds. In: ICML 2003 workshop on the continuum from labeled to unlabeled data in machine learning and data mining, pp 34–41
4. Ham JH, Lee DD, Saul LK (2005) Semisupervised alignment of manifolds. In: Proceedings of the 10th international workshop on artificial intelligence and statistics, pp 120–127
5. He X, Cai D, Yan S, Zhang H (2005) Neighborhood preserving embedding. In: ICCV, pp 1208–1213
6. He X, Niyogi P (2003) Locality preserving projections. In: NIPS
7. Jolliffe IT (2002) Principal component analysis, 2nd edn. Springer, New York
8. Li J, Tao D (2012) On preserving original variables in bayesian pca with application to image analysis. IEEE Trans Image Process 21(12):4830–4843
9. Li M, Xue XB, Zhou ZH (2009) Exploiting multi-modal interactions: a unified framework. In: IJCAI, pp 1120–1125
10. Li X, Lin S, Yan S, Xu D (2008) Discriminant locally linear embedding with high-order tensor data. IEEE Trans Syst Man Cybernet Part B 38(2):342–352
11. Mehta B (2008) Cross system personalization: enabling personalization across multiple systems. Ph.D. thesis, University of Duisburg
12. Roscher R, Schindler F, Förstner W (2010) High dimensional correspondences from low dimensional manifolds—an empirical comparison of graph-based dimensionality reduction algorithms. In: ACCV workshops, no 2, pp 334–343
13. Roweis ST, Saul LK (2000) Nonlinear dimensionality reduction by locally linear embedding. Science 290:2323–2326
14. Shon AP, Grochow K, Hertzmann A, Rao RPN (2006) Learning shared latent structure for image synthesis and robotic imitation. In: In Proceedings of NIPS. MIT Press, pp 1233–1240
15. Tenenbaum JB, de Silva V, Langford JC (2000) A global geometric framework for nonlinear dimensionality reduction. Science 290(5500):2319–2323
16. Verbeek J (2006) Learning nonlinear image manifolds by global alignment of local linear models. IEEE Trans Pattern Anal Mach Intell 28:1236–1250
17. Wang C, Mahadevan S (2008) Manifold alignment using procrustes analysis. In: ICML, pp 1120–1127
18. Wang L (2008) Feature selection with kernel class separability. IEEE Trans Pattern Anal Mach Intell 30(9):1534–1546
19. Weinberger KQ, Packer BD, Saul LK (2005) Nonlinear dimensionality reduction by semidefinite programming and kernel matrix factorization. In: Proceedings of the 10th international workshop on artificial intelligence and statistics, pp 381–388
20. Weinberger KQ, Saul LK (2006) An introduction to nonlinear dimensionality reduction by maximum variance unfolding. In: AAAI
21. Weinberger KQ, Sha F, Zhu Q, Saul LK (2006) Graph laplacian regularization for large-scale semidefinite programming. In: NIPS, pp 1489–1496
22. Xiong L, Wang F, Zhang C (2007) Semi-definite manifold alignment. In: ECML, pp 773–781
23. Yan S, Xu D, Zhang B, Zhang H, Yang Q, Lin S (2007) Graph embedding and extensions: a general framework for dimensionality reduction. IEEE Trans PAMI 29(1):40–51
24. Yang J, Zhang D, Yang Jy, Niu B (2007) Globally maximizing, locally minimizing: unsupervised discriminant projection with applications to face and palm biometrics. IEEE Trans Pattern Anal Mach Intell 29:650–664
25. Zhai D, Li B, Chang H, Shan S, Chen X, Gao W (2010) Manifold alignment via corresponding projections. In: BMVC, pp 1–11
26. Zhan Y, Yin J, Liu X. Nonlinear discriminant clustering based on spectral regularization. Neural Comput Appl
27. Zhang J, Marszalek M, Lazebnik S, Schmid C (2006) Local features and kernels for classification of texture and object categories: a comprehensive study. In: Computer vision and pattern recognition workshop, 2006. CVPRW'06. Conference on, Ieee. pp 13–13