Linear Dimensionality Reduction Method Based on Topological Properties*

Yuqin Yao[a], Hua Meng[a], Yang Gao[b], Zhiguo Long[b,c,*], Tianrui Li[b,c]

[a]*School of Mathematics, Southwest Jiaotong University, Chengdu, 611756, China*
[b]*School of Computing and Artificial Intelligence, Southwest Jiaotong University, Chengdu, 611756, China*
[c]*Manufacturing Industry Chains Collaboration and Information Support Technology Key Laboratory of Sichuan Province, China*

Abstract

Dimensionality reduction is an important data preprocessing technique that has been extensively studied in machine learning and data mining. Locality Preserving Projection (LPP) is a widely used linear unsupervised dimensionality reduction method, which maps high-dimensional data into low-dimensional subspace through linear transformation. Although various variants of LPP have been proposed to tackle different drawbacks of LPP, it is identified in this article that LPP does not possess the important topological property of translation invariance, that is, the linear transformation given by LPP is strongly related to the relative position between the data and the origin of the coordinate system. In this article, we theoretically analyze the reason why this drawback exists in LPP and propose to resolve it by introducing a kind of centralization to the model. Moreover, as topological properties are prominent information to characterize the structure of the data, this article proposes a further improvement of LPP to maintain topological connectivity of data after dimensionality reduction. Experiments on multiple synthetic and real-world datasets show that the new model incorporating topological properties outperforms not only the original LPP model but also several other classic linear or non-linear dimensionality reduction methods.

Keywords: Dimensionality reduction, linear projection, LPP, translation invariance, connectivity

1. Introduction

The era of big data brings us the problems of complexity, diversity and high dimensionality in data, e.g., [1, 2, 3]. If this kind of data are directly used to obtain information, there will be not only troubles of irrelevant attributes, but also increased computational

*Accepted version. Formal version available at: http://dx.doi.org/10.1016/j.ins.2022.12.098
*Corresponding author.
Email address: zhiguolong@swjtu.edu.cn (Zhiguo Long)

complexity and reduced performance. Dimensionality reduction can remove irrelevant information and reduce the complexity of data, and thus has become an important preprocessing step in machine learning and data mining. Traditional dimensionality reduction methods can be divided into two categories: feature extraction [4, 5] and feature selection [6, 7, 8] methods. Feature extraction methods aim to produce new features by mapping the original high-dimensional data into low-dimensional space through algebraic transformation. A good dimensionality reduction method should retain essential characteristics of data as much as possible, and remove redundant information to reveal the underlying structure and pattern of data. In addition, dimensionality reduction is also helpful for data visualization. For applications, dimensionality reduction has achieved great success in face recognition [9, 10], handwritten numeral recognition [11], signature verification [12], disease diagnosis [13] and stock selection [14].

In the past, numerous feature extraction based dimensionality reduction techniques have been studied from different aspects. There are supervised, unsupervised, and semi-supervised ones according to the availability of label information, and there are linear and non-linear ones depending on the corresponding mapping types. For example, Principal Component Analysis (PCA) [15] is a classic unsupervised linear technique; Linear Discriminant Analysis (LDA) [16] is a widely used supervised linear technique, and there are many other supervised linear techniques (e.g., [17]); Locally Linear Embedding (LLE) [18], t-Distributed Stochastic Neighbor Embedding (t-SNE) [19], Isometric Feature Mapping (Isomap) [20] and Laplacian Embedding (LE) [21], and many more (e.g., [22, 23]) are on the other hand non-linear techniques. There are also dimensionality reduction methods based on nonnegative matrix factorization, e.g., [24, 25].

Locality Preserving Projection (LPP) [26] is a well-known feature extraction based unsupervised dimensionality reduction approach, and it is a linear approximation of LE. LPP aims to project the original data through a linear transformation while retaining nearest neighbor connections. Several variants have been proposed to deal with different drawbacks of LPP. For example, supervised information is exploited in some variants, including CLPP [27] and CdLPP [28], which only consider similarities between data points within the same class, and LPDP [29] and DLPP [30], which incorporate inter- and intra-class information; L_2 norm is replaced by L_1 norm to achieve better robustness, e.g., 2D-DLPP-L1 [31] and ILPP-L1 [32], where the latter also proposes to preserve similarities between points and their neighbors in addition to dissimilarities; similarity measure calculation is also improved, e.g., LAPP [33] adaptively measures similarities in new representations obtained by applying LPP iteratively.

Although LPP has been shown to perform well on various datasets, it is surprising to notice that its selected directions for projection are very sensitive to the coordinate system for data points. In fact, by only changing the position of the data points in the coordinate system, i.e. translating the data points as a whole, the projection directions given by LPP might change significantly, though the relative positions of data points are not changed (cf. Figure 1). This means LPP does not possess an important topological property that is *translation invariance*.

In fact, taking topological properties into account for data analysis is a trending topic

known as topological data analysis (TDA) [34]. One of the main differences of TDA approaches from traditional statistical learning ones is that, the former concern more about local or global structural information, such as compactness, connectivity, or algebraic properties like persistent homology [35], whereas the latter are more interested in distribution characteristics of data. Having seen its merits, researchers applied TDA to various fields and tasks, e.g., image processing [36], classification [37], and bioinformatics [38].

This article repairs the topological property of translation invariance by introducing a kind of *centralization* to the LPP model. Moreover, as topological properties concern important information of data (e.g., the number of connected components in data is important structural information), we propose a novel characterization of the latent topological structure of data, viz. *topological connectivity*, which reflects connections of different parts of data, and empirically show that it is useful to retain this kind of topological information when performing dimensionality reduction. Existing works [18, 21] usually consider maintaining connectivity between points and their neighbors, including LPP, LE, and LLE. In other words, they only capture local structures, but ignore higher level structures like connected components (cf. Figure 4). However, these higher level structures can reveal important information of data, e.g., similarity and separation between data points in these structures should also be maintained after dimensionality reduction. Therefore, the approach proposed here not only tries to capture local structures, but also explores these higher level ones by constructing topological connectivity of data.

There are LPP variants that consider maintaining certain kinds of structures of data. For example, 2D-DLPP-L1 [31] proposes to maintain relative positions of image pixels by using matrices instead of vectors for finding projection subspaces. It considers structural information within each sample instead of among data points, which is different from the approach in this article. SSTNTL [39] makes use of supervised information to remove connections between points in different classes, and then reduces dimensionalities of data by using the new connection graph. It constructs topology on data points to help characterize similarities between points, but does not care about topological structural information like connected components.

The contributions of the article are as follows:

- We identify, theoretically analyze, and resolve the problem of the original LPP model that it is *highly sensitive to geometric translation*, i.e., the projection directions change significantly when data are moved around in the coordinate system.

- We propose to retain topological connectivity of data in dimensionality reduction, by exploring connectivity information in variant scales with novel connectivity measures for data, and devise an improved projection model for such purpose.

- We demonstrate the effectiveness of the improved model and its superiority over LPP and several other classic dimensionality reduction methods on multiple synthetic and real-world datasets.

The rest of this paper is arranged as follows. Section 2 briefly introduces the original LPP model. Section 3 analyzes the identified problems of LPP in details and proposes an improved

model (ConLPP) and its corresponding algorithms. Section 4 evaluates the performance of the new method on both synthetic and real-world datasets. Section 5 concludes the paper.

2. Preliminaries

Given a dataset $X = (x_1, x_2, \ldots\ldots, x_m)$, where $x_i \in \mathbb{R}^n$ is a column vector with n feature values. A linear projection can be described by a matrix $A_{n\times d}$ which maps X to $Y = (y_1, y_2, \ldots\ldots, y_m)$, where $y_i \in \mathbb{R}^d$ and $y_i = A^T x_i$.

2.1. Locality Preserving Projection

Locality Preserving Projection (LPP) is an unsupervised dimensionality reduction model, which tries to preserve the neighborhood structure of data. It is a linear approximation of Laplacian Eigenmaps (LE), and can achieve better results than LE on various types of data. The goal of LPP is to select several projection directions $a_i \in \mathbb{R}^n$ $(1 \le i \le d)$ to form a projection matrix $A = (a_1, \ldots, a_d)$, so that the projected data $A^T X$ can satisfy that nearest neighboring points are still neighbors. In order to achieve this goal, LPP constructs $\text{argmin}_a \sum_{i,j}(a^T x_i - a^T x_j)^2 W_{ij}$ as the optimization goal, where W_{ij} is a value characterizing the similarity between x_i and x_j. LPP adds the constraint $a^T X D X^T a = 1$ to avoid the influence of scaling on the projection directions, where D is a diagonal matrix s.t. $D_{ii} = \sum_j W_{ij}$. So the LPP model can be expressed as

$$\text{argmin}_a \sum_{i,j}(a^T x_i - a^T x_j)^2 W_{ij}, \text{ s.t. } a^T X D X^T a = 1. \quad \text{(Model 1)}$$

The following lemma shows that the objective function of Model 1 can be written in matrix-vector form.

Lemma 1 ([26]). *$\sum_{i,j}(a^T x_i - a^T x_j)^2 W_{ij} = a^T X L X^T a$, where $L = D - W$ is a Laplacian matrix and D is a diagonal matrix s.t. $D_{ii} = \sum_j W_{ij}$.*

Model 1 is then equivalently transformed to Model 2 below:

$$\text{argmin}_a \, a^T X L X^T a \quad \text{s.t.} \quad a^T X D X^T a = 1. \qquad \text{(Model 2)}$$

The steps to solve Model 2 of LPP are as follows.

1. Construct the adjacency matrix:
   If x_i and x_j are *neighbors*, then $W_{ij} = \exp(-\frac{\|x_i - x_j\|^2}{t})$ is used to represent the similarity between x_i and x_j; otherwise W_{ij} is set to 0. There are two common ways to determine neighbors:
   (a) ε-neighborhood. For a given parameter $\varepsilon > 0$, if $\|x_i - x_j\|^2 < \varepsilon$, then x_i and x_j are neighbors.
   (b) $k-$nearest neighborhood. For a given positive integer k, if x_i is among the k nearest neighbors of x_j, or x_j is among the k nearest neighbors of x_i, then x_i and x_j are neighbors.
2. Find the d optimal projection directions of Model 2 by calculating the eigenvectors of $(XDX^T)^{-1}XLX^T$ corresponding to the d smallest eigenvalues.

2.2. An Equivalent Model of LPP

It can be easily seen that Model 2 of LPP is also equivalent to the following model.

$$\text{argmin}_a \frac{a^T XLX^T a}{a^T XDX^T a}. \quad \text{s.t.} \quad a^T XDX^T a = 1. \qquad \text{(Model 3)}$$

In order to solve Model 3, we can first solve the following Model 3* and then adjust the length of a to satisfy the constraint $a^T XDX^T a = 1$. That is, for a being a solution to Model 3* and $a^T XDX^T a = h$, let $a' = \frac{1}{\sqrt{h}} a$, then $a'^T XDX^T a' = 1$ and is a solution to Model 3. The correctness of this process is ensured by Proposition 2.

$$\text{argmin}_a \frac{a^T XLX^T a}{a^T XDX^T a}. \qquad \text{(Model 3*)}$$

Proposition 2. *If c is a solution to Model 3*, then there is a solution b to Model 3 s.t. the direction of c is the same as b, and vice versa.*

PROOF. Note that the value of $\frac{c^T XLX^T c}{c^T XDX^T c}$ is only related to the direction of c, not its length, i.e., $\frac{c^T XLX^T c}{c^T XDX^T c} = \frac{(\lambda c)^T XLX^T (\lambda c)}{(\lambda c)^T XDX^T (\lambda c)}$ for any $\lambda \neq 0$. Therefore, for a solution c of Model 3*, we can always find some λ s.t. $(\lambda c)^T XDX^T (\lambda c) = 1$, and thus $b = \lambda c$ is also a solution of Model 3, whereas c and b have the same direction.

On the other hand, if b is a solution to Model 3, then b is also a solution to Model 3*. This is because, if $\frac{b^T XLX^T b}{b^T XDX^T b}$ is not minimal, then $\exists c$ s.t. $\frac{c^T XLX^T c}{c^T XDX^T c} < \frac{b^T XLX^T b}{b^T XDX^T b}$ and hence $\exists \lambda$ s.t. $(\lambda c)^T XDX^T (\lambda c) = 1$ and $\frac{(\lambda c)^T XLX^T (\lambda c)}{(\lambda c)^T XDX^T (\lambda c)} = \frac{c^T XLX^T c}{c^T XDX^T c} < \frac{b^T XLX^T b}{b^T XDX^T b}$. This means λc is a solution to Model 3 and with a smaller objective function value, which contradicts to that b is a solution to Model 3. □

Minimizing $\frac{a^T XLX^T a}{a^T XDX^T a}$ is to balance the two goals: minimizing $a^T XLX^T a$ and maximizing $a^T XDX^T a$. Minimizing $a^T XLX^T a$ is to make the transformed points as close as possible, i.e., to maintain neighbors. In next section, we will see that maximizing $a^T XDX^T a$ is to maximize a weighted sum of squares of the distances between the transformed points and the origin of the coordinate system. This makes the projection directions found by LPP very sensitive to the location of data in the coordinate system, instead of the relative position between data points, which is unfavorable in many scenarios. For example, for self-driving cars, as the car moves, a static obstacle will be at dynamic positions from the point of view of the car, and if LPP is applied for data proprecessing, then this obstacle might not be well recognized during the movement, since the projection direction keeps changing and the resulting representation might lose important information for some of the projection directions. We propose to repair this problem and to consider more topological properties to improve the original LPP model.

3. ConLPP Algorithm

We improve LPP from two aspects, i.e., repairing translation invariance and introducing topological connectivity. In the following, we discuss them in details.

3.1. Translation Invariance

3.1.1. Analysis

In the objective function of Model 3 of LPP, the term $a^T XDX^T a$ is strongly affected by the specific coordinates of data points. This means that for different coordinates of the data points, even when the relative positions of them are the same, LPP might produce significantly different projection directions, i.e., LPP is very sensitive to geometric translations. Take Figure 1 as an example. In this figure, there are five groups of data and each group has two clusters, where the distributions of the data are exactly the same between groups (different groups are simply geometric translations of each other), but their positions in the coordinate system are different. Arrows represent projection directions found by LPP, and the points on an arrow represent the data after projection on that direction. It can be seen from the figure that the projection directions vary significantly across these five groups although the distributions of points in different groups are the same. Notably, the change of directions will significantly affect the resulting representation after dimensionality reduction: some directions might destroy the structure within a group, resulting in a large overlap between data points in the two clusters within a group.

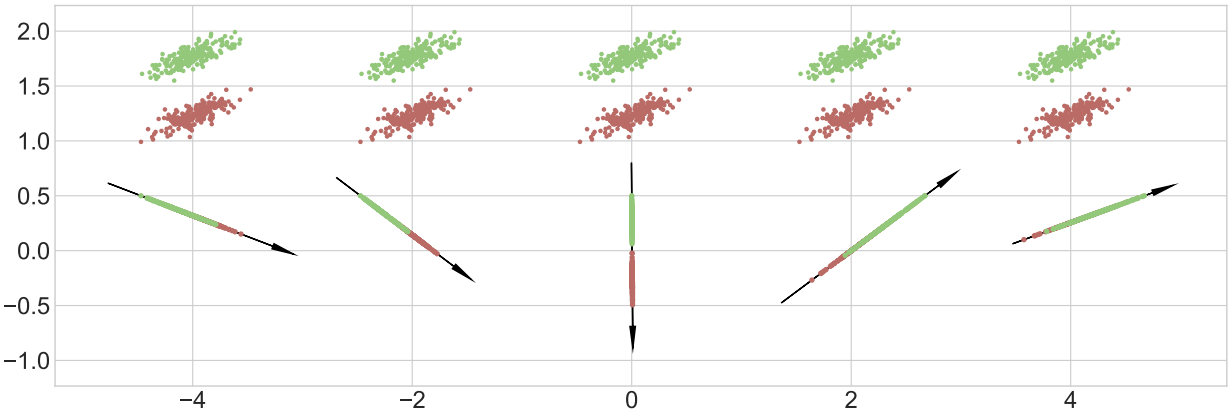


Figure 1: LPP projection results for geometric translations of the same group of data.

In order to analyze why LPP is seriously affected by coordinates, we make the following observation.

Proposition 3. $a^T XDX^T a = \sum (a^T x_i)^2 D_{ii}$.

PROOF. Recall that $X = (x_1, x_2, \cdots, x_m)$, then

$$\begin{aligned}\sum_i (a^T x_i)^2 D_{ii} &= \sum_i \left(a^T x_i\right)\left(a^T x_i\right)^T D_{ii} \\ &= \sum_i a^T \left(x_i D_{ii} x_i^T\right) a \\ &= a^T \left(\sum_i x_i D_{ii} x_i^T\right) a \\ &= a^T \left((x_1 D_{11}, x_2 D_{22}, \cdots, x_m D_{mm})\,(x_1, x_2, \cdots, x_m)^T\right) a \\ &= a^T X D X^T a.\end{aligned} \tag{1}$$

□

Note that $(a^T x_i)^2$ is the square of the distance from the projected coordinates to the origin. Therefore, by Proposition 3, maximizing the term $a^T X D X^T a$ is equivalent to maximize the weighted sum of squares of the distance from the projected coordinates to the origin, where the weights are D_{ii} $(i = 1, \ldots, m)$. This term in Model 3 of LPP tends to keep the data away from the origin after projection. For example, for data with higher density, D_{ii} will be larger and this model will be more likely to keep the data points away from the origin after projection. This is why $\max a^T X D X^T a$ is strongly affected by coordinates.

3.1.2. Improvement for Translation Invariance

In order to reduce the influence of the positions of the data in the coordinate system for LPP, we propose to *centralize* the coordinates of data. Let $x_i' = x_i - \mu$ and $X' = (x_1', x_2' \cdots x_m')$, where $\mu = \sum_i x_i/m$. It is easy to check that $||x_i - x_j|| = ||x_i' - x_j'||$ for any i and j. Then, the nearest neighbors of data points do not change, and if we consider Model 3 on X', we have

$$W'_{ij} = \exp(\frac{-||x_i' - x_j'||^2}{\sigma^2}) = \exp(\frac{-||x_i - x_j||^2}{\sigma^2}) = W_{ij}, \tag{2}$$

$$D'_{ii} = \sum_j W'_{ij} = \sum_j W_{ij} = D_{ii}, \tag{3}$$

$$L' = D' - W' = D - W = L. \tag{4}$$

Therefore, an improved model of Model 3 for LPP can be proposed to repair translation invariance as follows.

$$\arg\min_a \frac{a^T X' L X'^T a}{a^T X' D X'^T a} \quad \text{s.t.} \quad a^T X' D X'^T a = 1. \qquad \text{(Model 4)}$$

The relation between Model 3 of LPP and Model 4 can be seen from the following proposition.

Proposition 4. *$a^T X'LX'^T a$ is equal to $a^T XLX^T a$, but when $\mu \neq 0$, $a^T X'DX'^T a$ is not always equal to $a^T XDX^T a$.*

PROOF. From Lemma 1, we know that $a^T XLX^T a = \sum_{i,j}(a^T x_i - a^T x_j)^2 W_{ij}$, and similarly $a^T X'LX'^T a = \sum_{i,j}(a^T x'_i - a^T x'_j)^2 W'_{ij}$. Note that

$$\begin{aligned} \sum_{i,j}(a^T x'_i - a^T x'_j)^2 W'_{ij} &= \sum_{i,j}(a^T(x_i - \mu) - a^T(x_j - \mu))^2 W_{ij} \\ &= \sum_{i,j}(a^T x_i - a^T x_j)^2 W_{ij}. \end{aligned} \tag{5}$$

Therefore, $\sum a^T X'LX'^T a = \sum a^T XLX^T a$. On the other hand, by Proposition 3, $a^T XDX^T a = \sum(a^T x_i)^2 D_{ii}$, and similarly $a^T X'DX'^T a = \sum_i(a^T(x_i - \mu))^2 D_{ii}$ It is easy to see that generally when $\mu \neq 0$ then $\sum(a^T x_i)^2 D_{ii}$ is not equal to $\sum_i(a^T(x_i - \mu))^2 D_{ii}$.

□

In fact, from the proof, we can see that the term $a^T XDX^T a$ in Model 3 of LPP corresponds to the "weighted distance" of data points to the origin, whereas $a^T X'DX'^T a$ of Model 4 corresponds to the "weighted distance" of data points to the center of them, which is invariant w.r.t. geometric translations of data. In addition, it is also easy to see that $a^T X'LX'^T a = \sum_{i,j}(a^T x'_i - a^T x'_j)^2 W'_{ij}$ is also invariant w.r.t. geometric translations of data. Then we have the following conclusion.

Theorem 5. *For any dataset X, the projection directions found by Model 4 are independent of geometric translations of the data.*

Figure 2 illustrates the projection of the same data as in Figure 1 with Model 4. We can see that the projection directions are not affected by the relative position of data to the origin.

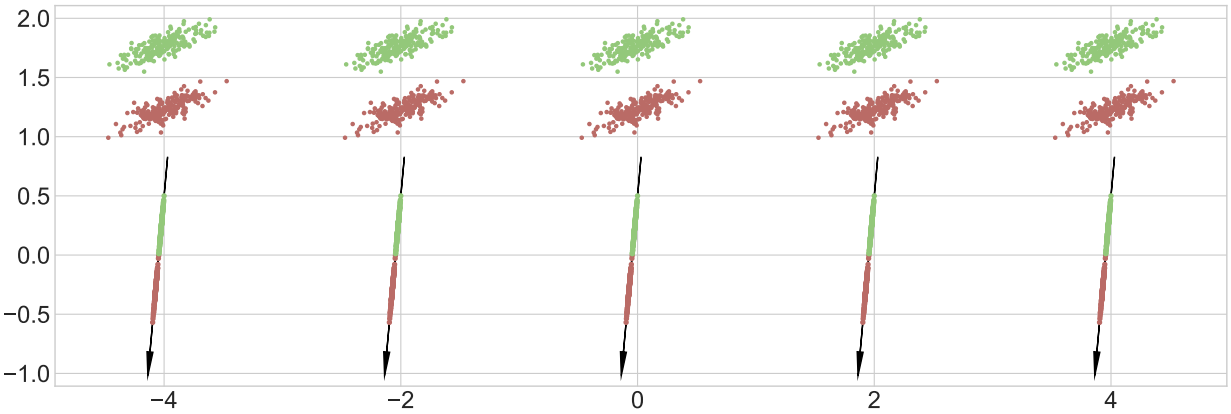


Figure 2: Projection results of the improved Model 4 for geometric translations of the same group of data.

Note that $a^T X' D X'^T a = \sum_i (a^T(x_i - \mu))^2 D_{ii}$. In particular, when $D_{ii} = 1$, $a^T X' D X'^T a$ degenerates to

$$\begin{aligned} \sum_i (a^T(x_i - \mu))^2 &= \sum_i \left(a^T x_i - a^T \mu\right) \left(a^T x_i - a^T \mu\right)^T \\ &= a^T \left(\sum_i \left(x_i - \mu\right) \left(x_i - \mu\right)^T \right) a \end{aligned} \tag{6}$$

In this situation, argmax $\sum_i (a^T(x_i - \mu))^2$ is exactly the goal of PCA, that is, to keep the data as separated as possible. Therefore, argmax $\sum_i (a^T(x_i - \mu))^2 D_{ii}$ can be regarded as a weighted PCA model, s.t. the points with higher density will have higher weights. Therefore, Model 4 is equivalent to finding the projection directions that *locally* make the nearest neighbors of data as close as possible, and *globally* maintain the weighted separation of data as much as possible. The chosen directions balance these two optimization objectives.

3.2. Topological Connectivity

Although Model 4 repairs the translation invariance property of LPP, it ignores the intermediate structures that are between local nearest neighborhoods and global data separation. In fact, data usually have several clusters, and clusters may be connected or separated. However, the original LPP and the revised Model 4 do not care about these structures, and this may result in poor projection results when the data structure is relatively complex. For example, consider Figure 3. The left of the figure shows the projection of the original data by LPP, and the right shows the projection by Model 4. As both LPP and Model 4 only consider to keep local neighboring points close and to maintain global separation of the whole data, but ignore intermediate structures, the dimensionality reduction results of them have obvious overlaps between data from different clusters. The main reason is that the two smaller clusters were taken as a whole for global separation, and the two models did not consider separating them as the main optimization goal. We argue that this kind of intermediate separation information can be captured by *topological connectivity*.

3.2.1. Topological Space and Its Connectivity

Connectivity is a fundamental concept in topology. Before discussing topological connectivity of a dataset, we briefly introduce the concepts related to topological connectivity, and more details can be found in [40].

For a nonempty set X, let $T \subseteq 2^X$ be a family of subsets of X. If T contains $\varnothing$ and X, and is closed with respect to set union operation and finite intersection operation, then T is called a *topology* on X. The elements in T are called *open sets*, and (X, T) is called a *topological space*. A *closed set* in (X, T) is a subset of X whose complement is an open set. Suppose (X, T) is a topological space and $Y \subseteq X$. Then the *subspace topology* T_Y is a topology on Y such that $V \in T_Y$ iff there is a $U \in T$ and $V = U \cap Y$. For example, the n-dimensional Euclidean space $\mathbb{R}^n$, with the topology in which all the *open balls* are open sets, is a topological space, called the *Euclidean topological space*. Here, an open ball $B(x, \delta)$ in $\mathbb{R}^n$ is $B(x, \delta) = \{y \mid d(x, y) < \delta\}$. A subspace topology of the Euclidean topological space

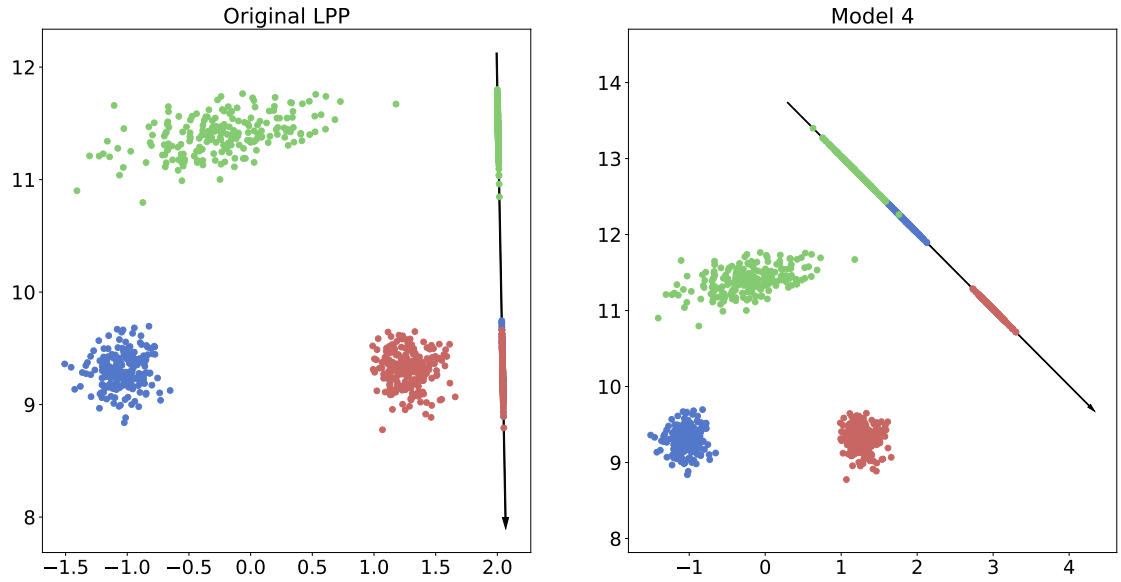


Figure 3: Illustration of the problems of the original LPP model and the improved Model 4.

$\mathbb{R}^2$ could be T_Y for $Y = \{x \mid ||x|| \leq 2\}$. Topology can be used to characterize the relative distance between data points. For instance, if $x \in \mathbb{R}^n$, and $y \in B(x, 1/5), z \notin B(x, 1/3)$ can characterize that y is closer to x than z is.

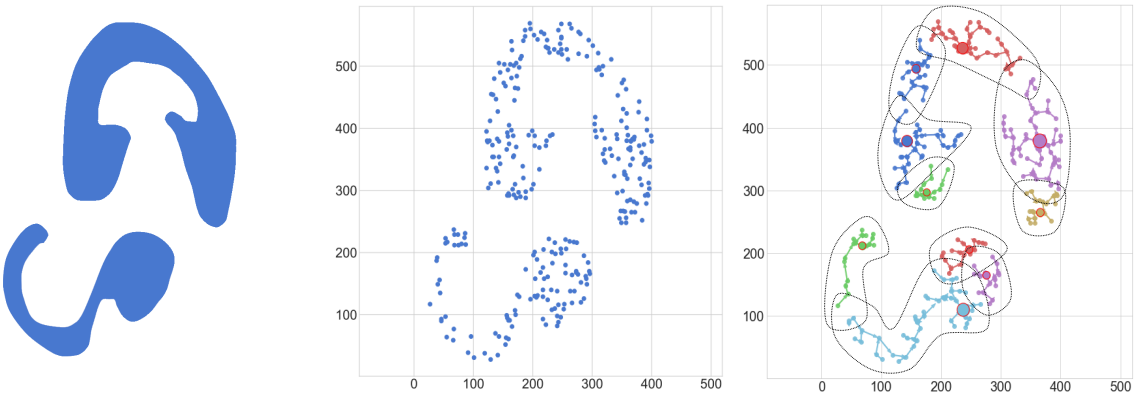
Topology can also characterize connectivity information of data. A topological space is said to be *connected* if and only if there is no nonempty set U in it such that both U and its complement are open sets. Therefore, if a topological space is not connected, then it can be divided into disjoint parts, i.e., *connected components*.

Definition 1. Suppose (X, T) is a topological space, $Y \subseteq X$, and $Y \neq \emptyset$. Then Y is a *connected component* of (X, T) iff (Y, T_Y) is connected and for all Y' s.t. $Y \subset Y' \subseteq X$, $(Y', T_{Y'})$ is not connected.

If (X, T) is connected, then X itself is the unique connected component of (X, T); if (X, T) is not connected, then X can be decomposed into several connected components. When two points $x, y \in X$ are in different connected components, we say x and y are *strongly separated*. Take Figure 4(a) as an example. The two geometries shown in it together form a topological subspace of $\mathbb{R}^2$, which is not connected but contains two connected components.

In practice, *path connectivity* is usually used to characterize connectivity of a set in a topological space.

Definition 2 (Path connectivity). Suppose (X, T) is a topological space, and $x, y \in X$. Then if there is a continuous map f from $[0, 1]$ to X, such that $f(0) = x$ and $f(1) = y$, then f is a *path* from x to y. In this case, x and y are said to be *path connected*. A topological space (X, T) is *path connected* if any two distinct $x, y \in X$ are path connected.



(a) (b) (c)

Figure 4: Illustration of connected components in a topological subspace and connectivity of data points.

In Euclidean space, a path between two points x and y is a continuous curve connecting x and y. Note that a path connected topological space is also connected, but the other way around generally is not true. However, for most of the practical cases, a connected topological space is also path connected, e.g., for a connected component in Figure 4(a), it is both connected and path connected. We also have the following properties for path connectivity.

Lemma 6. *Suppose (X, T) is a topological space, and $x_1, x_2, x_3 \in X$.*

- *If x_1, x_2 and x_2, x_3 are path connected then x_1, x_3 are path connected.*
- *If x_1 and x_2 are path connected, then x_1 and x_2 are in the same connected component.*

To maintain the structure of data during dimensionality reduction, connectivity information of data is an important aspect to consider. Topological connectivity provides a promising candidate to model such information. However, as we usually do not have all of the data points from a topological subspace, but only some samples of them, and sometimes even the whole data subspace itself may be a discrete set. For example, in Figure 4(b), we only have some sample data points from the topological subspace in Figure 4(a), and thus we have to use only these sample data points to discover the structural information of the whole topological subspace.

In the following, we will discuss how to model connectivity for sample data points with the idea of topological connectivity. The general idea is to approximate connectivity of a sample space by exploring a kind of "path connectivity" of data points and constructing connected components with such information. Once connectivity and separation information is available, we can compute similarity and separation measures for data points and thus incorporate more topological information into the dimensionality reduction model.

3.2.2. Connectivity of Data

Suppose $X = (x_1, x_2, \ldots, x_m)$ contains the sample data points. For every $x_i \in X$ we denote by $N_k(x_i)$ the nearest k neighbors of x_i and we require $x_i \in N_k(x_i)$. Intuitively, a connected component in a topological space consists of two kinds of points: inner points and boundary points. To distinguish between these points, we exploit the concept of density, as inner points usually have higher density than boundary points.

Definition 3 (*Density of points*). For each $x_i \in X$, we define the *density* of x_i as follows:

$$\rho(x_i) = \sum_{x_j \in N_k(x_i) \setminus \{x_i\}} \exp(-\frac{d_{ij}^2}{\sigma^2}), \tag{7}$$

where d_{ij} is the Euclidean distance between x_i and x_j.

The density of data points reflects the tightness of the distribution of data points in the local area. The inner-most point of a connected component tends to have the highest density, and this point is called a *core point*.

Definition 4 (*Leader point and core point*). Denote by $R(x_i)$ the *leader point* of $x_i \in X$. Then $R(x_i)$ can be defined as follows. If the density of x_i is greater than that of any $x \in N_k(x_i) \setminus \{x_i\}$ then $R(x_i) = x_i$, otherwise $R(x_i) = x_j$, where x_j is the data point with higher density than x_i in $N_k(x_i)$ that is closest to x_i (if there are multiple, then we select the first met one). We call x_i a *core point* if $R(x_i) = x_i$.

The leading relationship reflects connectivity to some extent. Generally, a point with higher density is considered to be more representative for data. A point is likely to be connected to some of its nearest neighbors, and the most possible connection happens between this point and its nearest neighbor that has higher density (i.e. more representative). Therefore, we consider there is a path between each point and its leader point, and they are in the same connected component of a topological subspace. Thus, a point and its leader point should be in the same connected component. In order to formally characterize connected component with the help of the leading relationship between points, we need to define the *leader set* of a point.

Definition 5 (Leader set). For each $x \in X$, $\mathcal{S}(x)$ is the *leader set* of x, which is recursively defined as follows:

- $R(x) \in \mathcal{S}(x)$;
- If $y \in \mathcal{S}(x)$, then $R(y) \in \mathcal{S}(x)$.

For convenience, we also call a point y in $\mathcal{S}(x)$ a *leader* of x. It is easy to see that if z is a core point then $\mathcal{S}(z) = \{z\}$. In the following, we also identify the set of the points whose leader set contains the core point z, as a *density branch*, inspired by [41], and identify the nearest k neighbors of each point in a density branch as an *expanded density branch*.

Definition 6 (Density branch and expanded density branch). For each core point z, we call the set of points that have z as a leader, i.e., $D(z) = \{x \mid z \in \mathcal{S}(x)\}$, the *density branch* of z. We also define the *expanded density branch* of a core point z as $E(z) = \cup_{x \in D(z)} N_k(x)$.

It can be seen from the definition that there is a 1-1 correspondence between density branches and core points, i.e., a density branch contains exactly one core point and each core point has a density branch.

Different density branches may belong to the same connected component and this happens if they are closely *adjacent*, that is, they have enough *shared nearest neighbors*. The shared nearest neighbors of two core points z_1 and z_2 is the set $\mathrm{SNN}(z_1, z_2) = E(z_1) \cap E(z_2)$.

Definition 7 (Connectivity). Given two core points z_1 and z_2, the density branches $D(z_1)$ and $D(z_2)$ are *connected* if $|\mathrm{SNN}(z_1, z_2)| > \tau \times \min\{|E(z_1)|, |E(z_2)|\}$, where τ is some given threshold parameter. Furthermore, if $D(z_1)$ is connected with $D(z_2)$ and $D(z_2)$ is connected with $D(z_3)$, then we also say $D(z_1)$ is connected with $D(z_3)$. For any two point $x_i \in D(z_1)$ and $x_j \in D(z_2)$, x_i and x_j are said to be *connected* if $D(z_1)$ is connected with $D(z_2)$.

Here, the idea is to consider a point x is connected with its leaders and thus the points in each density branch are connected with each other, and two density branches are connected if they have enough shared neighbors (τ is set as 0.05 in experiments). Figure 4(c) illustrates this idea. In this figure, each point is connected with its leaders and the points with a red circle are core points. Each core point determines a density branch in which each pair of points have a path between them. Expanded density branches are illustrated as regions bounded by dashed lines. For two density branches, if they share enough neighbors and thus have strong connection with each other, then we consider them as connected. Although there is no path between points in two connected density branches, we think they have a high possibility to be connected in the original topological subspace. In this way, we obtain *connected components* of data points, and these components approximate those in the original topological subspace.

Definition 8 (Connected component). A *connected component* C for a dataset X is the maximal set in which each pair of points are connected.

Each density branch has a unique core point, which is the leader of every point in this density branch. On the other hand, a connected component can contain more than one core points. In fact, a connected component is exactly the union of all density branches that are connected with each other. That is, $C = \bigcup_{i=1}^{l} D(z_i)$, where for any $1 \leq i, j \leq l$, $D(z_i)$ and $D(z_j)$ are connected, but for any core point $z \notin \{z_1, z_2, \ldots, z_l\}$, $D(z_i)$ and $D(z)$ are not connected. Roughly speaking, density branches characterize local structural information, while connected components captures higher level structural information.

It is easy to see that the connectivity relation between data points is reflexive, transitive, and symmetric. Therefore, the connectivity relation induces a partition of X into disjoint connected components, as stated in the following proposition.

Proposition 7. *Given k and τ, each dataset X is divided into disjoint connected components.*

For example, as shown in Figure 5, given the values of k and τ, for each x_i, we can calculate $N_k(x_i)$ and $\rho(x_i)$ by definition. Then we can calculate $R(x_i)$ and find all the core points of the dataset. After that, for each core point z_t we can obtain its density branch $D(z_t)$ and expanded density branch $E(z_t)$. According to Definition 8, the connected density branches are combined to obtain connected components $\{C_k\}$. The corresponding process is shown in Algorithm 1. Note that if the data contain outliers, then there are some connected components that contain only few samples. Thus, to remove outliers, only the connected components with more than 2 samples are retained by the algorithm.

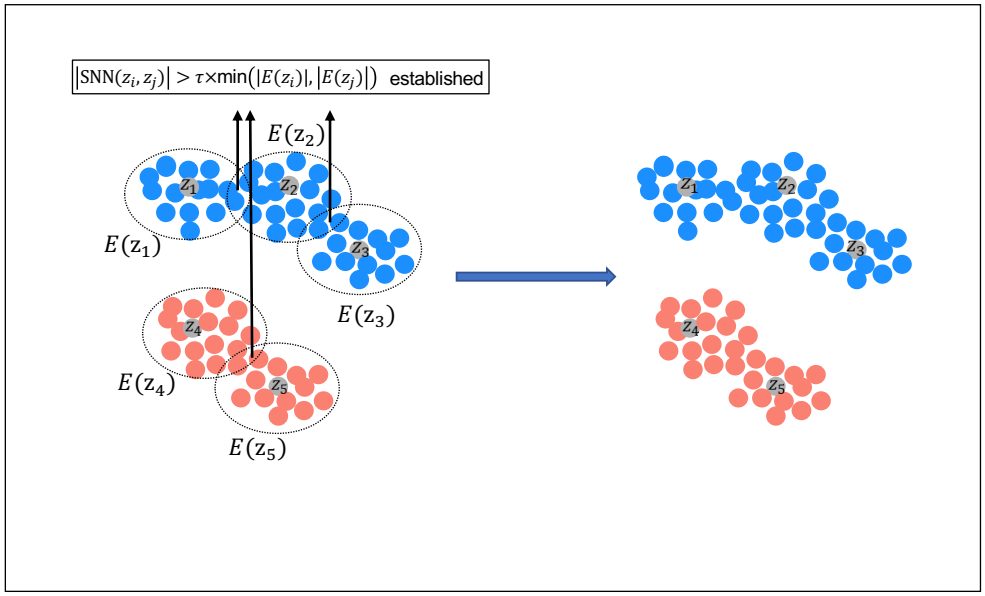


Figure 5: Illustration of connectivity and constructing connected components of data points.

Note that the number of connected components is related to the number of nearest neighbors k and the given threshold τ. Given τ, if $k = 1$, then each data point itself forms a connected component; if $k = m$ (m is the total number of data points), then all the data points together form a unique connected component. The smaller k is, the more connected components there will be; the greater k is, the fewer connected components there will be, and more density branches will be merged together. Moreover, if k is small and two points are in the same density branch, then they should be considered as very similar. Conversely, if k is large and two points belong to different connected components, then they should be considered to be obviously different. Therefore, later we will take the results of different k into consideration to better capture the structure of data.

3.2.3. Similarity Matrix and Separation Matrix

With the connected components information obtained by Algorithm 1, we propose two measures to characterize the similarity and difference between points, respectively.

Algorithm 1: ExploreStructure

Input: Dataset X; number of nearest neighbors k; threshold parameter τ.
Output: Core points; density branches $\{D(z_i)\}$; connected components $\{C_i\}$

```
Calculate N_k(x), ρ(x) and leader point R(x) for each x ∈ X;
Get the core points: Core = {z_1, z_2, ... z_t};
Calculate the density branches: D(z_1), D(z_2), ... D(z_t);
Calculate the expanded density branches: E(z_1), E(z_2), ... E(z_t);
foreach (z_i, z_j) ∈ Core × Core and z_i ≠ z_j do
    G[i, j] ← 0;
    SNN(z_i, z_j) ← E(z_i) ∩ E(z_j);
    if |SNN(z_i, z_j)| > τ × min(|E(z_i)|, |E(z_j)|) then
        G[i, j] ← 1;
Get connected components {C_i} from G;
return Core, {D(z_i)}, {C_i}.
```

If two samples x_i and x_j are in the same density branch then they are very similar, and we denote a *similarity* matrix by $\mathrm{Sim}^{(k)}$ for a given k to describe the similarity of those samples, such that

$$\mathrm{Sim}_{ij}^{(k)} = \begin{cases} \exp(-\frac{\|x_i - x_j\|^2}{\sigma^2}) & \text{if } i \neq j, \text{and } \exists z_t, x_i, x_j \in D(z_t); \\ 0 & \text{otherwise,} \end{cases} \tag{8}$$

where the parameter σ is set as 1% of the square of the largest distance between data points. In fact, the performance of the proposed approach is not sensitive to this parameter, so we will just apply this setting to all of the experiments.

On the other hand, if x_i and x_j are in different connected components then they should be considered as dissimilar and dissimilarity will be characterized by a *separation* matrix. The separation matrix is over core points only, instead of all the data points, as core points are the most representative ones for separation. For core points, the separation matrix $\mathrm{Sep}^{(k)}$ is defined as follows.

1. When there is only one connected component found by Algorithm 1, the separation matrix $\mathrm{Sep}^{(k)}$ is defined as:

$$\mathrm{Sep}^{(k)} = \frac{1}{|\,\mathrm{Core}\,|^2} \sum_{(z_i, z_j) \in \mathrm{Core} \times \mathrm{Core}} (z_i - z_j)\,(z_i - z_j)^T . \tag{9}$$

2. When there are more than one connected components found by Algorithm 1, then we can calculate the separation matrix $\mathrm{Sep}^{(k)}$ over pairs of core points from different connected components. Denote by $P^{(k)}$ the set of all the pairs of z_i, z_j from different connected components, then

$$\mathrm{Sep}^{(k)} = \frac{1}{|P^{(k)}|} \sum_{(z_i, z_j) \in P^{(k)}} (z_i - z_j)\,(z_i - z_j)^T \tag{10}$$

The corresponding algorithm to calculate $\text{Sim}^{(k)}$ and $\text{Sep}^{(k)}$ is shown in Algorithm 2.

Algorithm 2: ComputeMatrices

Input: Dataset X; number of nearest neighbors k; threshold parameter τ.
Output: $\text{Sim}^{(k)}$, $\text{Sep}^{(k)}$

$\text{Core}, \{D(z_i)\}, \{C_i\} \leftarrow \text{ExploreStructure}(X, k, \tau)$
$\text{Sim}^{(k)} \leftarrow (0)_{m \times m}$;
foreach $z \in \text{Core}$ **do**
  foreach $(x_u, x_v) \in D(z) \times D(z)$ *and* $x_u \neq x_v$ **do**
    $\text{Sim}^{(k)}_{uv} \leftarrow \exp(-\frac{\|x_u - x_v\|^2}{\sigma^2})$;
$M \leftarrow \emptyset$;
if $|\{C_i\}| > 1$ **then**
  foreach $(z_u, z_v) \in \text{Core} \times \text{Core}$ **do**
    $C_u \leftarrow$ the connected component containing z_u;
    $C_v \leftarrow$ the connected component containing z_v;
    if $C_u \neq C_v$ **then**
      $M \leftarrow M \cup \{(z_u, z_v)\}$;
else
  $M \leftarrow \text{Core} \times \text{Core}$;
$\text{Sep}^{(k)} = \frac{1}{|M|} \sum_{(z_u, z_v) \in M} (z_u - z_v)(z_u - z_v)^T$;
return $\text{Sim}^{(k)}$, $\text{Sep}^{(k)}$.

3.3. Dimensionality Reduction Algorithm

As we have mentioned before, the value of k will influence the number of connected components and thus influence the structure of data revealed by these components. For example, we increase the value of k from k_1 to k_2 and the results are shown in Figure 6. When $k = k_1$, there are 7 connected components; when k is increased to k_2, the two connected components C_{11} and C_{12} are merged, and the same happens to C_{15} and C_{16}; when k is increased even larger to k_3, more components are merged and a higher level structure of the data is then revealed.

Inspired by this observation, we vary the value of k to explore the structure of data in different scales. Several similarity matrices and separation matrices will be obtained through the process, and we propose to combine these matrices by a weighted sum. As the similarity with a smaller k will be more significant than that with a larger k, we assign larger weight to $\text{Sim}^{(k)}$ for a smaller value of k. On the other hand, we assign larger weight to $\text{Sep}^{(k)}$ for a larger value of k, since the separation with a larger k will be more significant.

In particular, the similarity matrix and the separation matrix will defined as follows.

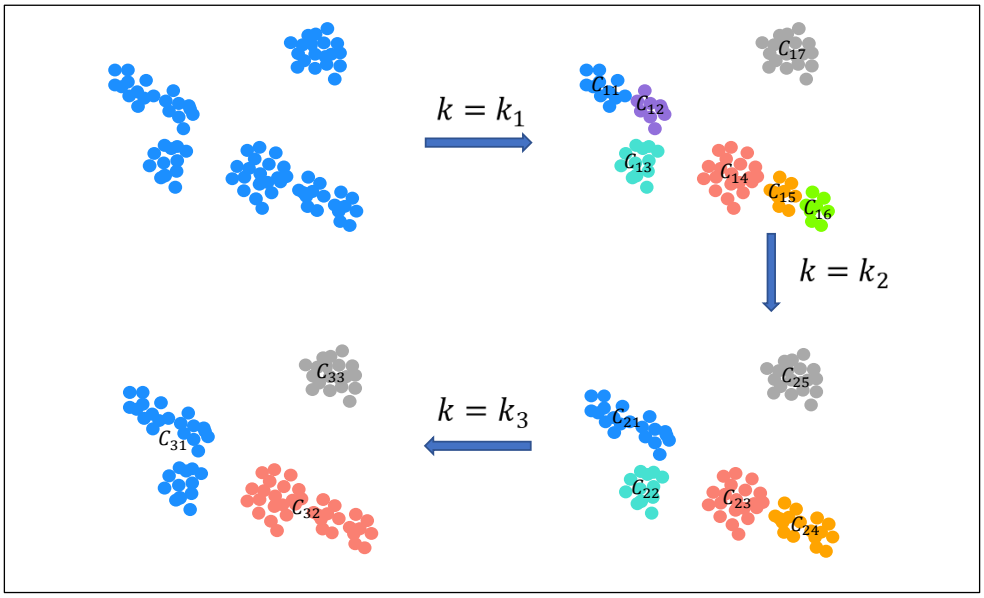


Figure 6: Illustration of the effect of different values of k on constructing connected components.

$$\text{Sim} = \sum_{i=1}^{l} \frac{\exp(\frac{1}{k_i})}{\sum_{i=1}^{l} \exp(\frac{1}{k_i})} \text{Sim}^{(k_i)}; \tag{11}$$

$$\text{Sep} = \sum_{i=1}^{l} \frac{\exp(k_i)}{\sum_{i=1}^{l} \exp(k_i)} \text{Sep}^{(k_i)} \tag{12}$$

The original model of LPP is thus improved by combining the idea in Model 4 and the connectivity measures, as below.

$$\text{argmin}_a \frac{a^T X(L + L^*)X^T a}{a^T \left(XD^*X^T + \text{Sep}\right) a}, \quad \text{s.t.} \quad a^T(XD^*X^T + \text{Sep})a = 1. \tag{ConLPP}$$

where $X = (x_1 - \mu, \ldots, x_m - \mu)$, $L^* = D^* - \text{Sim}$, and D^* is a diagonal matrix and $D^*_{ii} = \sum_{j=1}^{|X|} \text{Sim}_{ij}$. Note that the constraint here is just to change the length of the projection vectors and thus to scale the coordinates after projection (cf. Proposition 2). One can also remove this constraint and use unit projection vectors instead to have orthogonal projections, or any other lengths of interest.

The above optimization problem is a typical Rayleigh quotient problem and can be solved by calculating eigenvectors. Let $S_1 = X(L + L^*)X^T$ and $S_2 = XD^*X^T + \text{Sep}$. Then the solution of the above optimization problem is the eigenvector corresponding to the smallest eigenvalue of $S_2^{-1}S_1$, and this eigenvector is the projection direction for dimensionality reduction. When d directions are needed, the eigenvectors of the d smallest eigenvalues are then used. The corresponding algorithm is shown in Algorithm 3.

Algorithm 3: ConLPP

Input: Dataset $X = (x_1, \ldots, x_m)$; the range of the numbers of nearest neighbors $[k_0, k_1]$; target dimensionality d; threshold parameter τ.

Output: Projection directions $A = (a_1, \ldots, a_d)$.

$X \leftarrow X - \mu$, where μ is the average vector of X;
foreach $k \in [k_0, k_1]$ **do**
  $\text{Sim}^{(k)}, \text{Sep}^{(k)} \leftarrow \text{ComputeMatrices}(X, k, \tau)$;
Calculate Sim and Sep by Equations 11 and 12;
Calculate L, which is consistent with that of LPP;
Calculate $D^* = (D^*_{ii})_{m\times m}$, where $D^*_{ii} = \sum_{j=1}^{|X|} \text{Sim}_{ij}$;
Calculate $L^* = D^* - \text{Sim}$;
Calculate $S_1 = X\left(L + L^*\right)X^T$;
Calculate $S_2 = XD^*X^T + \text{Sep}$;
Calculate the eigenvalues and eigenvectors of $S_2^{-1}S_1$, and then sort them in ascending order;
Select the eigenvectors corresponding to the first d non-zero minimum eigenvalues: $\lambda_1, \lambda_2, \ldots\ldots\ \lambda_d$ to make up $A = (\frac{a_1}{\sqrt{a_1^T S_2 a_1}}, \ldots, \frac{a_d}{\sqrt{a_d^T S_2 a_d}})$;
return A.

3.4. Time Complexity Analysis and Comparison

Suppose the number of data points in a dataset is m, the original dimension of data is n, the number of core points is t, and the number of nearest neighbors k is in range $[k_0, k_1]$, where the length of the range is l.

In Algorithm 1, to get $N_k(x)$, $\rho(x)$, $R(x)$, and core points in lines 1 and 2, one can make use of a KD-tree, and the time complexity is $\mathcal{O}((n+k)m\log m)$. To obtain density branches, one can descendingly sort the points by their densities $\rho(x)$, and go through the points one by one to allocate them to the density branch that their leader point belongs to (if the leader point of a point is itself, then this point is a core point and it will be allocated to a new density branch). So line 3 of Algorithm 1 takes $\mathcal{O}(m\log m + m)$ time. Line 4 expands the density branches by including the k-nearest neighbors of each point in the density branches, so its time complexity is $\mathcal{O}(km)$. For two expanded density branches, one can obtain their intersection by scanning through one of them and checking if each point of it is also in the other one, which takes $\mathcal{O}(m)$ time in the worst case. In total, lines 5-9 take $\mathcal{O}(t^2m)$ time. Line 10 can obtain connected components with a depth first search in time $\mathcal{O}(t^2)$. Therefore, the time complexity of Algorithm 1 is $\mathcal{O}((\log m + k + t^2)m)$.

For Algorithm 2, line 1 exploits Algorithm 1 and has time complexity $\mathcal{O}((\log m+k+t^2)m)$. For lines 3-5, it takes $\mathcal{O}(n)$ time to calculate each $\text{Sim}^{(k)}_{uv}$, and each core point will take $\mathcal{O}(m_s^2)$ steps to obtain all $\text{Sim}^{(k)}_{uv}$, where m_s is the number of points in density branch z_s and $\sum_{s=1}^{t} m_s = m$. So the total time for lines 3-5 is $\mathcal{O}(nm^2)$. Lines 6-14 take $\mathcal{O}(t^2)$ time, and line 15 takes $\mathcal{O}(n^2t^2)$ time. So in total, Algorithm 2 takes $\mathcal{O}((\log m+k+t^2)m+nm^2+n^2t^2)$.

The main algorithm ConLPP, i.e. Algorithm 3, basically has similar steps to LPP, with additional steps for centralizing X and obtaining Sim and Sep in lines 1-4, where these additional steps take $\mathcal{O}(mn+l((\log m+k+t^2)m+nm^2+n^2t^2))$ time. Lines 5-7 take $\mathcal{O}(m^2)$ time to obtain L, D^*, and L^*. Lines 8-9 take $\mathcal{O}(nm^2+n^2m)$ time. Lines 10-11 take $\mathcal{O}(n^3)$ to obtain the final projection directions. In summary, the time complexity of ConLPP is $\mathcal{O}(mn+l((\log m+k+t^2)m+nm^2+n^2t^2)+m^2+nm^2+n^2m+n^3)$. As k and l is usually very small compared to m and n, the time complexity can be simplified to $\mathcal{O}((m+n^2)t^2+nm^2+n^2m+n^3)$, or $\mathcal{O}(nm^2)$ if $n \ll m$. This is comparable to LPP, which has the time complexity of $\mathcal{O}(nm^2+n^2m+n^3)$, and is also comparable to many other dimensionality reduction methods, like t-SNE, LE, and LLE, whose time complexity is $\mathcal{O}(nm^2)$. For example, on a dataset of 2000 points and 649 features, for a target dimensionality of 5, ConLPP takes 3.98s, and LPP takes 0.54s, whereas LLE takes 0.23s, LE takes 45s, and t-SNE takes 178s.

4. Empirical Evaluations

In order to evaluate the proposed model, we conduct experiments on both synthetic and real-world datasets, against seven dimensionality reduction baseline methods, i.e., LLE [18], PCA [15], LDA [16], LE [21], t-SNE [19], LPP [26], ILPP-L1 [32], and LAPP [33], where LLE, LE, and t-SNE are non-linear methods, and LPP, ILPP-L1, and LAPP are LPP based methods. LDA is supervised and is used to demonstrate that ConLPP can capture inherent structures of data. Note that t-SNE, LLE, PCA, and LDA here are from the scikit-learn library and use default parameters; LE, ILPP-L1, and LAPP are implemented according to the corresponding papers. The range of k for ConLPP is set to $[5, 15]$.

4.1. Datasets

There are one synthetic dataset and 14 datasets used in the experiments. The synthetic dataset is used to show that ConLPP model can repair the problem of LPP for translation invariance and can further maintain data connectivity and separation after dimensionality reduction. Real-world datasets are used to illustrate the advantages of ConLPP over other algorithms.

The synthetic dataset is visually illustrated in Figure 7. It is sampled from a multivariate normal distribution, and contains three clusters of points, each of which has 200 points. The details of the 14 real-world datasets are shown in Table 1. Among them, ORL, USPS, COIL-20, FashionMNIST, and CIFAR-10 are image datasets, where the last two are testing subsets of the original datasets.

All datasets are standardized, so that the mean value of a dataset is 0 and the standard deviation is 1. Note that, in this case, the original LPP and Model 4 are equivalent because $\mu = 0$.

4.2. Results on Synthetic Datasets

As illustrated in Figure 2, Model 4 reduces the influence of the position of the origin on LPP and fixes translation invariance. However, from Figure 3, both LPP and Model 4

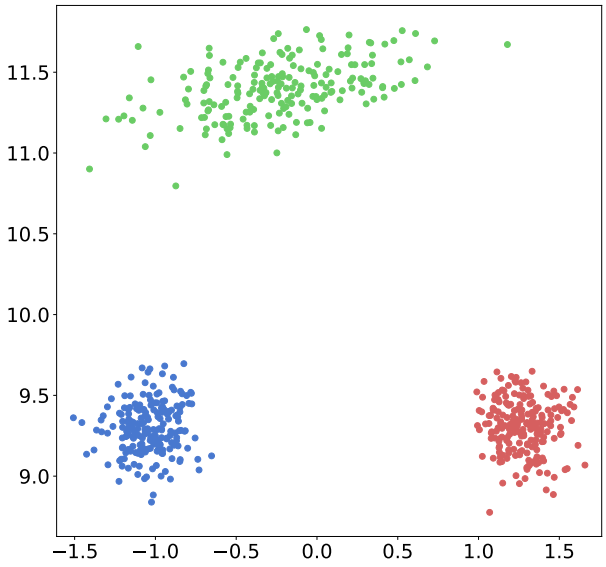


Figure 7: Visualization of the synthetic dataset.

might project clusters that are originally separated into overlapping ones, because they do not consider higher level structural information.

Figure 8 shows the results of LPP (equivalently, Model 4, as data were standardized) and ConLPP into one-dimensional space on the synthetic dataset after standardization. It can be seen that LPP makes the originally separated clusters overlapping after dimensionality reduction, while ConLPP can maintain the separation state of these clusters after projection. The main reason is that the data contain three connected components and ConLPP maintains this structure by introducing characterizations of connectivity information into the optimization model.

4.3. Visualization on Real-World Datasets

Visualization can give a direct and intuitive impression on the performance of maintaining the structure of complex high-dimensional data [42]. Figure 9 is the visualization of the projection of the points for digits 0 to 9 in the dataset "digital" into two-dimensional plane by various algorithms. By comparing ConLPP and LPP, we can find that ConLPP better maintains the separation between different clusters and the closeness of data from the same cluster, e.g., LPP mixes "2", "3", and "8" clusters while ConLPP does not. Note that LDA is a supervised dimensionality reduction method, whereas ConLPP is unsupervised but has a quite similar results as LDA. This means ConLPP can indeed capture inherent structure of the data. The result of t-SNE for visualization is good, but it has several disadvantages which can be seen from later experiments: its performance worsens for higher target dimensionalities, and it cannot give the projection function which means it cannot deal with unseen data points. For the other methods, the advantage of ConLPP is obvious.

Table 1: Details of real-world datasets.

Dataset	#instances	#features	#classes
indianliver	583	10	2
congressEW	435	16	2
vote	435	16	2
sonarEW	208	60	2
ORL	400	1024	40
digital	2000	649	10
pengcolonEW	62	2000	2
segmentation	2100	19	7
mfeat-fou	2000	76	10
mfeat-kar	2000	64	10
USPS	9298	256	10
COIL-20	1440	1024	20
FashionMNIST	10000	784	10
CIFAR-10	10000	3072	10

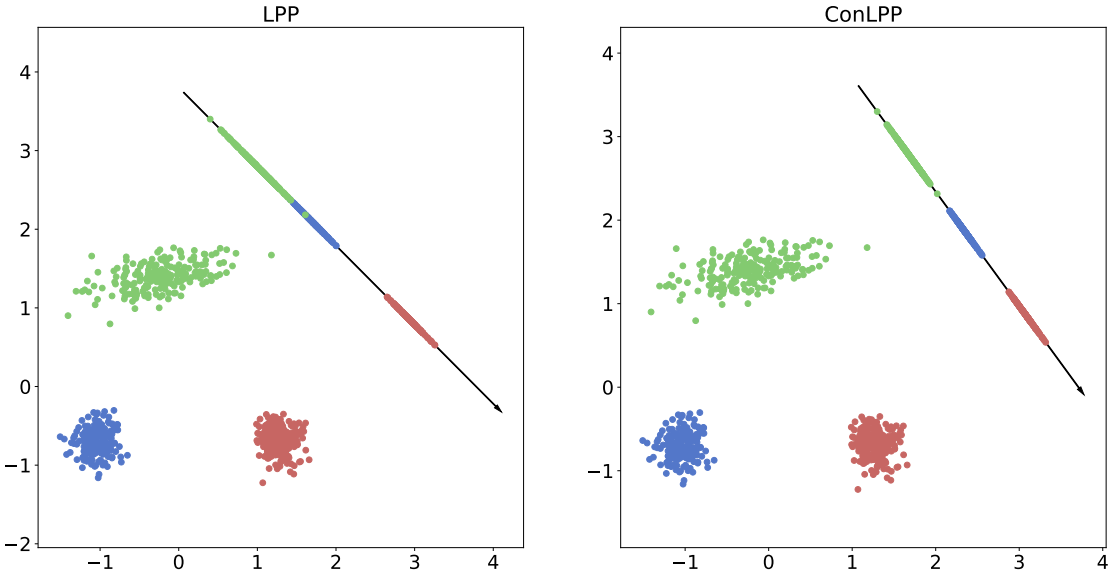


Figure 8: Projection results of the standardized synthetic dataset by LPP and ConLPP, respectively.

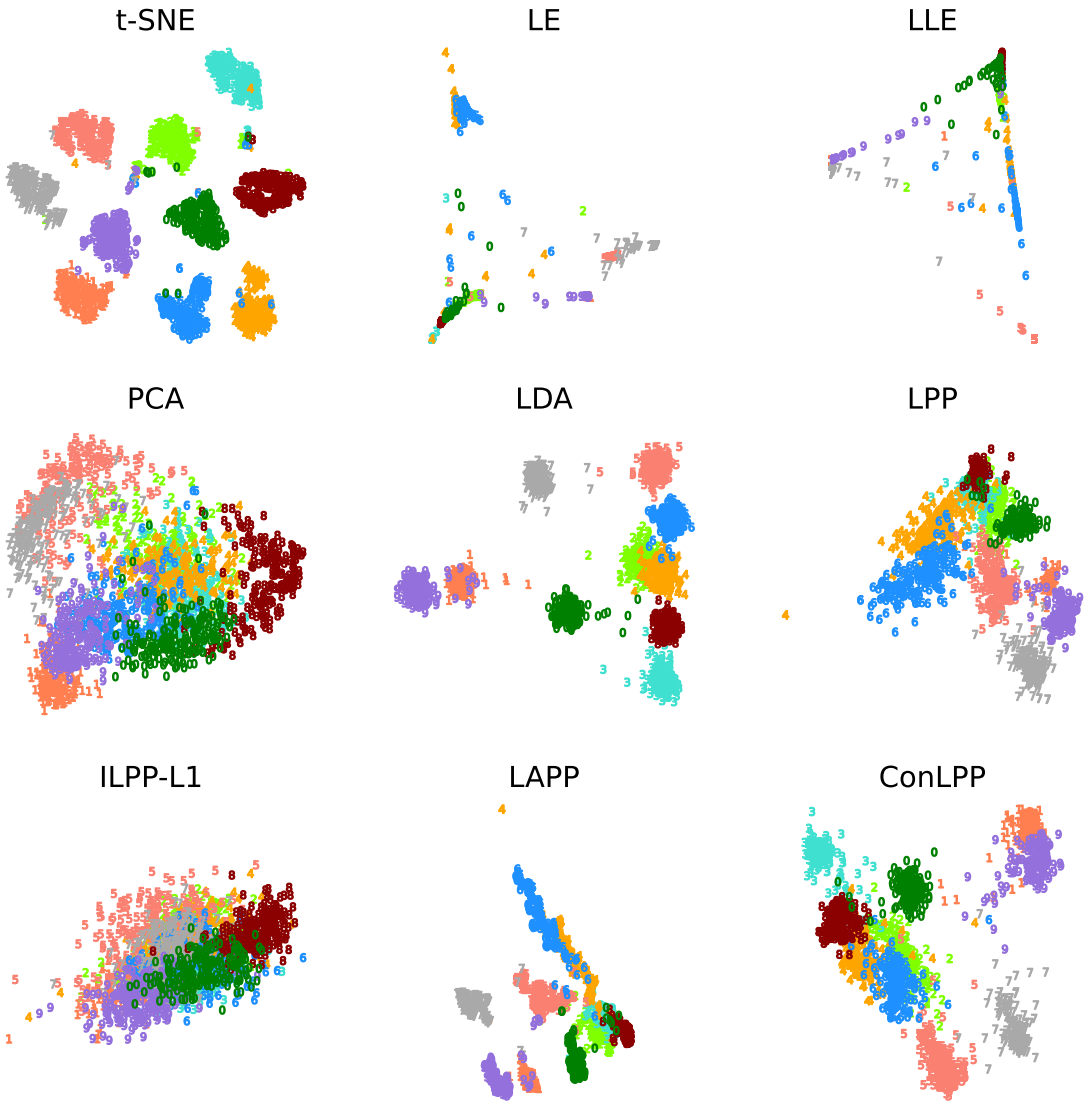


Figure 9: Visualization results of dimensionality reduction methods for mapping data points representing "0"-"9" in the dataset "digital" into two-dimensional space.

Table 2: Results on the real-world datasets in terms of 1NN classification accuracy.

Dataset	t-SNE	LE	LLE	PCA	LPP	ILPP-L1	LAPP	ConLPP
indianliver	66.21(1)	67.40(9)	64.14(8)	65.54(4)	66.22(7)	68.45(4)	66.57(2)	**68.46(4)**
congressEW	92.86(2)	91.48(14)	89.65(7)	92.41(13)	91.73(11)	**94.71(10)**	93.09(15)	93.33(5)
vote	93.09(2)	93.09(7)	89.42(8)	91.94(14)	91.26(8)	94.47(13)	94.03(8)	**94.70(6)**
sonarEW	62.92(2)	65.93(6)	69.70(19)	70.18(4)	63.07(5)	69.23(11)	68.90(14)	**71.08(19)**
ORL	92.24(2)	77.73(18)	86.48(19)	**93.00(19)**	90.74(19)	73.50(18)	60.75(13)	92.49(9)
digital	97.80(2)	98.10(12)	97.70(13)	97.75(19)	97.05(18)	87.30(19)	98.35(19)	**98.45(12)**
pengcolonEW	71.03(2)	71.23(10)	72.82(18)	71.23(4)	72.82(5)	66.27(1)	71.43(13)	**80.75(1)**
segmentation	93.19(13)	88.71(11)	88.62(11)	93.48(13)	93.57(10)	94.00(16)	**94.10(7)**	93.76(10)
mfeat-fou	79.55(3)	74.65(16)	76.20(16)	75.15(19)	74.65(15)	72.75(19)	**81.45(17)**	79.90(17)
mfeat-kar	96.00(4)	94.30(19)	94.9(16)	94.75(18)	91.20(14)	87.65(19)	96.15(15)	**96.45(15)**
USPS	**96.91(2)**	94.49(19)	96.10(16)	96.20(19)	89.85(19)	N/A	83.54(17)	94.67(19)
COIL-20	97.22(4)	89.03(15)	92.5(18)	95.97(18)	93.75(15)	85.77(18)	84.86(15)	**99.72(11)**
FashionMNIST	**81.40(9)**	70.58(15)	74.75(19)	79.40(19)	76.60(19)	N/A	73.14(19)	77.71(19)
CIFAR-10	30.3(14)	21.76(15)	22.69(17)	**30.50(19)**	22.25(18)	N/A	20.31(19)	19.45(19)

4.4. Evaluation with 1NN on Results Obtained After Dimensionality Reduction

In this subsection, we first perform dimensionality reduction on the whole dataset, for each of the 14 real-world datasets, and then use 1NN [43] to conduct classification on the dimensionality reduction results. The motivation is that the performance of 1NN can reflect the quality of dimensionality reduction results, illustrating whether points from the same class are relatively close and points from different classes are relatively far away from each other.

For each algorithm, the best dimensionality for dimensionality reduction on each dataset is determined by first performing dimensionality reduction with target dimensionality from 2 to 19 and calculating the average accuracy of 1NN in 10-fold cross validation for the transformed data, and then selecting the dimensionality with the highest accuracy as the best one. Once the best dimensionality is determined, the corresponding average accuracy and standard deviation over ten folds is recorded. For ILPP-L1, because its results might vary largely for different runs, we perform the aforementioned process ten times for each dimensionality and use the highest score in these ten runs as the score of that dimensionality.

The results are shown in Table 2, where the numbers in braces are the best dimensionalities for corresponding algorithms and "N/A" means the corresponding algorithm cannot finish in the time limit of 10 hours.

From Table 2, we can see that on all of the 14 datasets, the average classification accuracy of the transformed data by ConLPP is significantly higher than LPP, and is the best in 7 out of 14 cases and close to the best in most of the other cases. Note that we select the best result of ILPP-L1 in 10 runs, so it is not surprising it is better than ConLPP on the dataset "congressEW", where ConLPP outperforms all the others except ILPP-L1; for the datasets "segmentation" and "mfeat-fou" where LAPP is the best, we can see that ConLPP is very close to the best, and ConLPP actually outperforms LAPP on most of the other datasets and the advantage is very significantly on several datasets, e.g., ORL, pengcolonEW, USPS, and COIL-20.

We also illustrate the results of each algorithm for different target dimensionalities on several datasets in Figure 10. From Figure 10, we observe that ConLPP is dominating for almost all of the dimensionalities. For ORI and vote, ConLPP shows more stability than the others across different dimensionalities. For mfeat-fou and mfeat-kar, although ConLPP is not the best in lower dimensionalities, it quickly becomes evidently better than or close to the others in higher dimensionalities. These demonstrate the effectiveness of the idea of ConLPP for identifying topological structures.

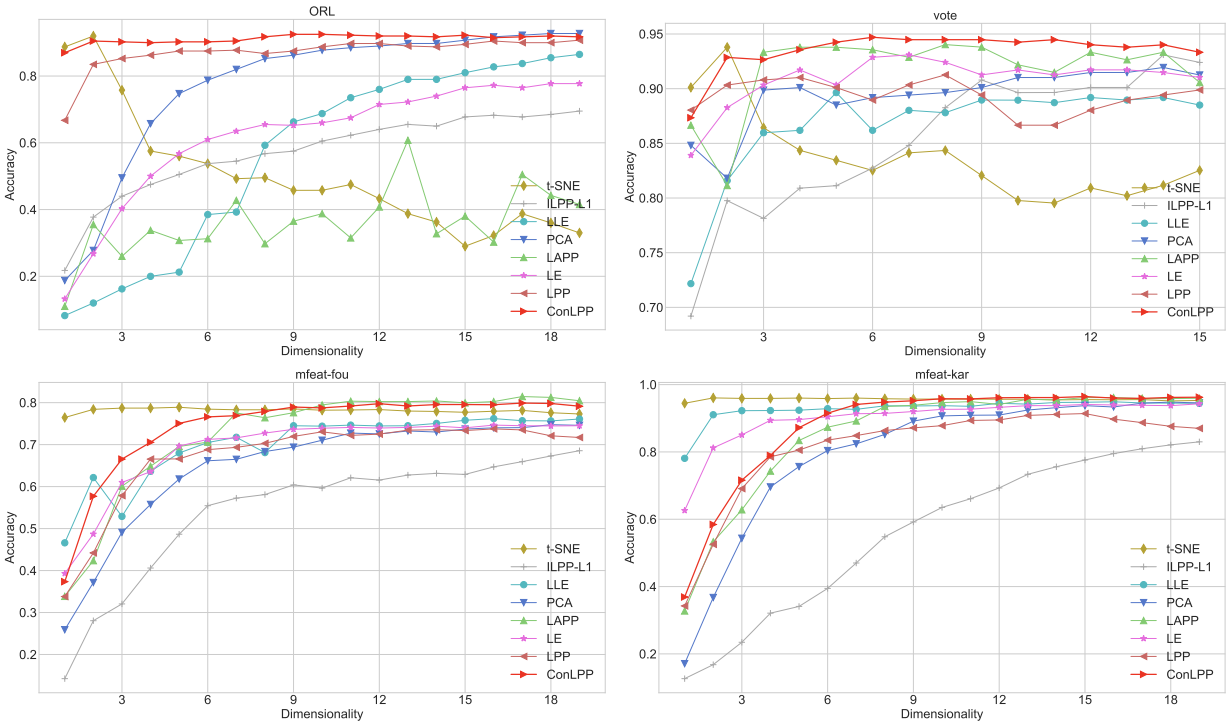


Figure 10: Results on four real-world datasets w.r.t. different target dimensionalities in terms of 1NN classification accuracy.

4.5. Evaluation of Generalization Ability

In this subsection, instead of finding projection directions based on the whole dataset, we first partition the dataset into training set and testing set, and then find projection directions on training set and apply them on testing set. The transformed testing set is then used to perform 1NN classification with 10-fold cross validation. The partition of training set and testing set is also performed 10 times, and the results are averaged. The motivation of the experiments here is to evaluate the generalization ability of the proposed dimensionality reduction model.

Particularly, ConLPP is compared against four linear ones of the baseline methods, i.e., PCA, LPP, ILPP-L1, and LAPP, on the dataset "digital" of handwritten numbers. The other methods used in this article are not applicable as they have to be based on the whole

Table 3: Accuracy results on the dataset "digital" for 1NN classification on transformed testing sets by different dimensionality reduction methods.

Method	20%	40%	60%	80%
PCA	**97.38(19)**	**97.28(19)**	96.72(19)	96.05(19)
LPP	42.06(15)	64.25(9)	93.23(11)	95.43(12)
ILPP-L1	85.49(19)	82.92(19)	81.49(19)	80.27(19)
LAPP	28.31(17)	92.00(15)	97.13(19)	97.49(14)
ConLPP	39.18(8)	87.17(10)	**97.17(11)**	**97.65(11)**

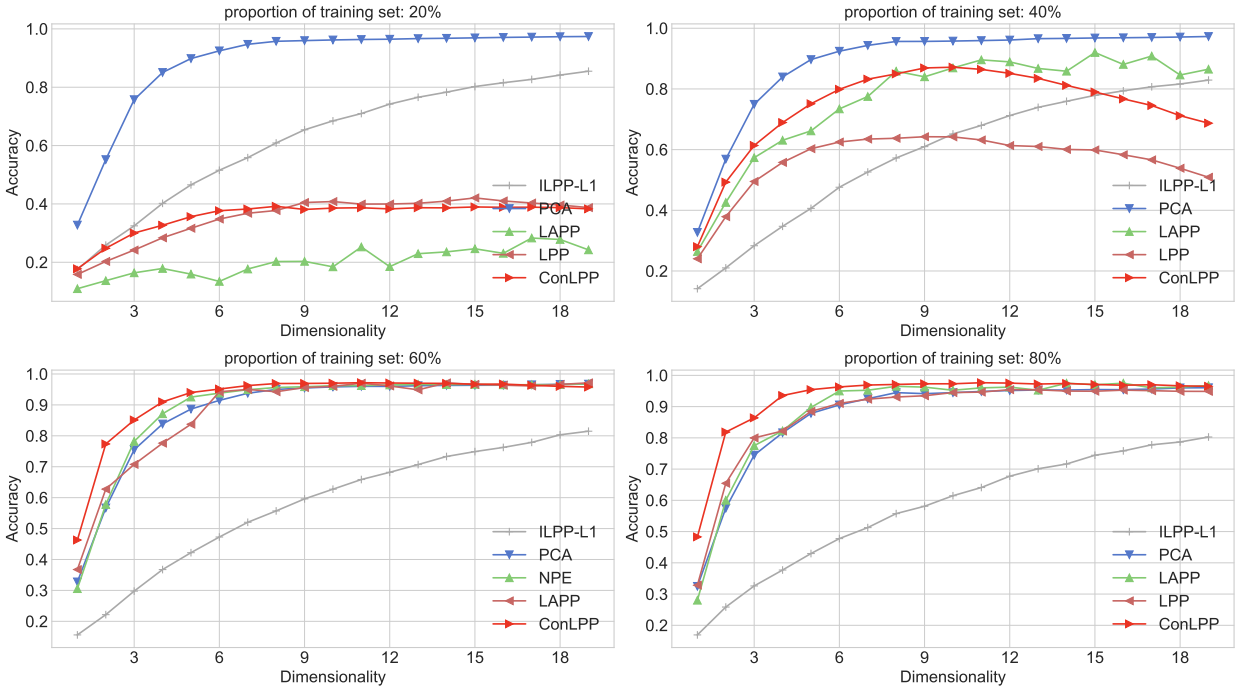


Figure 11: Results on the dataset "digital" w.r.t. different target dimensionalities in terms of 1NN classification accuracy on different portions of transformed testing sets.

dataset. The proportion of training set is set to be 20%, 40%, 60% and 80%, respectively. Table 3 and Figure 11 show the results.

From Table 3, it is easy to see that ConLPP can have better results with larger proportion of training samples and it performs better than LPP, ILPP-L1, LAPP in these cases. It is worth noting that the performance of ConLPP is not ideal when the proportion of training samples is very small (20%). This is expected, as in this case the training samples cannot accurately reflect the structure of the whole dataset, and the confidence of the structure found by ConLPP will be low in this case. This phenomenon becomes clearer in Figure 11. With relatively larger proportion of training samples, ConLPP can better identify the structure of the dataset. Note that for the 40% case, the performance of both LPP and ConLPP becomes worse when the target dimensionality is over 10. The reason is that the

constraint in the model scales the projection coordinates differently in different dimension, which stretches the projected data and affects the performance of 1NN on the transformed data. This can be avoided by discarding the constraint but using unit projection vectors. In summary, ConLPP can have better generalization ability than LPP, ILPP-L1, LAPP, and PCA with relatively larger proportion of training samples (e.g., over 60%), and it is an effective model for dimensionality reduction.

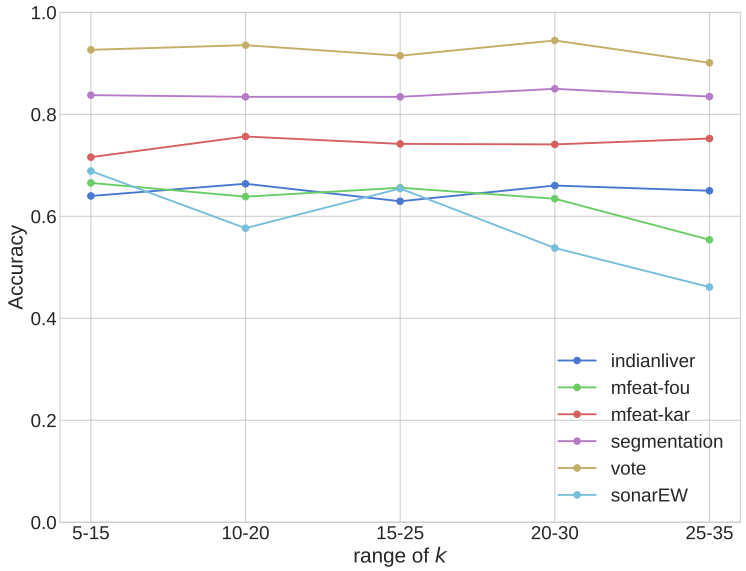


Figure 12: Illustration of the robustness of ConLPP w.r.t. different ranges of k.

4.6. Parameter Analysis

Sensitivity to parameters could limit the applicability of a dimensionality reduction method in practice. In the proposed ConLPP model, we would like to see how the choice of the ranges of k will affect the accuracy of ConLPP. Figure 12 shows the results of 1NN classification on the transformed data of the datasets indianliver, mfeat-fou, mfeat-kar, segmentation and vote, w.r.t. different ranges of k, which are $[5, 15]$, $[10, 20]$, $[15, 25]$, $[20, 30]$, and $[25, 35]$, respectively. The results illustrate that the performance of ConLPP is relatively stable for different choices of parameters, which is favorable for practical applications.

5. Conclusion

This paper proposed a novel model for dimensionality reduction by improving the well-known LPP model to maintain topological properties including translation invariance and topological connectivity. We demonstrated and analysed the translation sensitivity problem of LPP, which, to the best of our knowledge, was not noticed in the literature. We proposed a new model to repair this problem and theoretically proved its effectiveness. We also proposed to take more topological properties into consideration and devised a novel model that considers topological connectivity. Extensive experimental results on synthetic and

real-world datasets demonstrated the effectiveness and superiority of the new model over the original LPP model and several other widely used dimensionality reduction models. In the future, we plan to incorporate more topological properties to the model so that it can discover more complex structures.

Acknowledgment

This work was supported by the National Natural Science Foundation of China [grant number 61806170]; the Humanities and Social Sciences Fund of Ministry of Education [grant number 18XJC72040001]; and the National Key Research and Development Program of China [grant number 2019YFB1706104].

References

[1] S. Yan, D. Xu, B. Zhang, H. Zhang, Q. Yang, S. Lin, Graph embedding and extensions: A general framework for dimensionality reduction, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (1) (2007) 40–51.

[2] R. He, N. Xiong, L. T. Yang, J. H. Park, Using multi-modal semantic association rules to fuse keywords and visual features automatically for web image retrieval, Information Fusion 12 (3) (2011) 223–230.

[3] R. Zhou, X. Wang, J. Wan, N. Xiong, EDM-Fuzzy: An Euclidean distance based multiscale fuzzy entropy technology for diagnosing faults of industrial systems, IEEE Transactions on Industrial Informatics 17 (6) (2021) 4046–4054.

[4] H. Li, T. Jiang, K. Zhang, Efficient and robust feature extraction by maximum margin criterion, IEEE Transactions on Neural Networks 17 (1) (2006) 157–165.

[5] M. Unser, M. Eden, Multiresolution feature extraction and selection for texture segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence 11 (7) (1989) 717–728.

[6] J. Yan, B. Zhang, N. Liu, S. Yan, Q. Cheng, W. Fan, Q. Yang, W. Xi, Z. Chen, Effective and efficient dimensionality reduction for large-scale and streaming data preprocessing, IEEE Transactions on Knowledge and Data Engineering 18 (3) (2006) 320–333.

[7] A. Blum, P. Langley, Selection of relevant features and examples in machine learning, Artificial Intelligence 97 (1-2) (1997) 245–271.

[8] D. Koller, M. Sahami, Toward optimal feature selection, in: International Conference on Machine Learning, 1996, pp. 284–292.

[9] X. He, D. Cai, S. Yan, H. Zhang, Neighborhood preserving embedding, in: IEEE International Conference on Computer Vision, 2005, pp. 1208–1213.

[10] G. D. C. Cavalcanti, T. I. Ren, J. F. Pereira, Weighted modular image principal component analysis for face recognition, Expert Systems with Applications 40 (12) (2013) 4971–4977.

[11] E. Silva Jr, A. Oliveira, W. Santos, C. Mello, C. Zanchettin, G. D. C. Cavalcanti, Feature selection and model design through GA applied to handwritten digit recognition from historical document images, in: International Conference on Frontiers in Handwritten Recognition, 2008, pp. 562–567.

[12] B. Li, D. Zhang, K. Wang, Online signature verification based on null component analysis and principal component analysis, Pattern Analysis and Applications 8 (4) (2006) 345–356.

[13] H. Gunduz, An efficient dimensionality reduction method using filter-based feature selection and variational autoencoders on Parkinson's disease classification, Biomedical Signal Processing and Control 66 (2021) 102452.

[14] J. Han, Z. Ge, Effect of dimensionality reduction on stock selection with cluster analysis in different market situations, Expert Systems with Applications 147 (2020) 113226.

[15] I. T. Jolliffe, Principal Component Analysis, Springer Series in Statistics, Springer, 1986.

[16] R. A. Johnson, D. W. Wichern, Applied Multivariate Statistical Analysis, Vol. 6, Pearson London, 2014.

[17] S. Chen, C. H. Q. Ding, B. Luo, Linear regression based projections for dimensionality reduction, Information Sciences 467 (2018) 74–86.
[18] S. T. Roweis, L. K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326.
[19] L. van der Maaten, G. Hinton, Visualizing data using t-SNE, Journal of Machine Learning Research 9 (86) (2008) 2579–2605.
[20] M. Balasubramanian, E. L. Schwartz, The Isomap algorithm and topological stability, Science 295 (5552) (2002) 7–7.
[21] M. Belkin, P. Niyogi, Laplacian eigenmaps and spectral techniques for embedding and clustering, in: Advances in Neural Information Processing Systems, 2001, pp. 585–591.
[22] Y. Li, Y. Chai, H. Zhou, H. Yin, A novel dimension reduction and dictionary learning framework for high-dimensional data classification, Pattern Recognition 112 (2021) 107793.
[23] J. Chen, L. Liao, W. Zhang, L. Du, Mixture factor analysis with distance metric constraint for dimensionality reduction, Pattern Recognition 121 (2022) 108156.
[24] X. Peng, D. Xu, D. Chen, Robust distribution-based nonnegative matrix factorizations for dimensionality reduction, Information Sciences 552 (2021) 244–260.
[25] Z. Chen, A. Fu, R. H. Deng, X. Liu, Y. Yang, Y. Zhang, Secure and verifiable outsourced data dimension reduction on dynamic data, Information Sciences 573 (2021) 182–193.
[26] X. He, P. Niyogi, Locality preserving projections, in: Advances in Neural Information Processing Systems, 2003, pp. 153–160.
[27] J. Li, J. Pan, S. Chu, Kernel class-wise locality preserving projection, Information Sciences 178 (7) (2008) 1825–1835.
[28] E. R. Silva, G. D. C. Cavalcanti, T. I. Ren, Class-wise feature extraction technique for multimodal data, Neurocomputing 214 (2016) 1001–1010.
[29] J. Gui, W. Jia, L. Zhu, S. Wang, D. Huang, Locality preserving discriminant projections for face and palmprint recognition, Neurocomputing 73 (13-15) (2010) 2696–2707.
[30] W. Yu, X. Teng, C. Liu, Face recognition using discriminant locality preserving projections, Image and Vision Computing 24 (3) (2006) 239–248.
[31] Y. Tang, Z. Zhang, Y. Zhang, F. Li, Robust L1-norm matrixed locality preserving projection for discriminative subspace learning, in: International Joint Conference on Neural Networks, 2016, pp. 4199–4204.
[32] W. Yu, R. Wang, F. Nie, F. Wang, Q. Yu, X. Yang, An improved locality preserving projection with l_1-norm minimization for dimensionality reduction, Neurocomputing 316 (2018) 322–331.
[33] A. Wang, S. Zhao, J. Liu, J. Yang, L. Liu, G. Chen, Locality adaptive preserving projections for linear dimensionality reduction, Expert Systems with Applications 151 (2020) 113352.
[34] L. Wasserman, Topological data analysis, Annual Review of Statistics and Its Application 5 (2018) 501–532.
[35] I. Chevyrev, V. Nanda, H. Oberhauser, Persistence paths and signature features in topological data analysis, IEEE Transactions on Pattern Analysis and Machine Intelligence 42 (1) (2020) 192–202.
[36] H. Elhamdadi, S. Canavan, P. Rosen, AffectiveTDA: Using topological data analysis to improve analysis and explainability in affective computing, IEEE Transactions on Visualization and Computer Graphics 28 (1) (2022) 769–779.
[37] W. Guo, K. Manohar, S. L. Brunton, A. G. Banerjee, Sparse-TDA: Sparse realization of topological data analysis for multi-way classification, IEEE Transactions on Knowledge and Data Engineering 30 (7) (2018) 1403–1408.
[38] T. Feng, J. I. Dávila, Y. Liu, S. Lin, S. Huang, C. Wang, Semi-supervised topological analysis for elucidating hidden structures in high-dimensional transcriptome datasets, IEEE/ACM Transactions on Computational Biology and Bioinformatics 18 (4) (2021) 1620–1631.
[39] A. J. Ma, P. C. Yuen, W. W. W. Zou, J. Lai, Supervised spatio-temporal neighborhood topology learning for action recognition, IEEE Transactions on Circuits and Systems for Video Technology 23 (8) (2013) 1447–1460.

[40] J. R. Munkres, Topology, Pearson, 2000.
[41] D. Cheng, S. Zhang, J. Huang, Dense members of local cores-based density peaks clustering algorithm, Knowledge-Based Systems 193 (2020) 105454.
[42] I. S. Dhillon, D. S. Modha, W. S. Spangler, Class visualization of high-dimensional data with applications, Computational Statistics and Data Analysis 41 (1) (2002) 59–90.
[43] N. Kwak, Principal component analysis based on L1-norm maximization, IEEE Transactions on Pattern Analysis and Machine Intelligence 30 (9) (2008) 1672–1680.