# Nonlinear Dimensionality Reduction by Manifold Unfolding

by

Pooyan Khajehpour Tadavani

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2013

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Pooyan Khajehpour Tadavani

## Abstract

Every second, an enormous volume of data is being gathered from various sources and stored in huge data banks. Most of the time, monitoring a data source requires several parallel measurements, which form a high-dimensional sample vector. Due to the curse of dimensionality, applying machine learning methods, that is, studying and analyzing high-dimensional data, could be difficult. The essential task of dimensionality reduction is to faithfully represent a given set of high-dimensional data samples with a few variables. The goal of this thesis is to develop and propose new techniques for handling high-dimensional data, in order to address contemporary demand in machine learning applications.

Most prominent nonlinear dimensionality reduction methods do not explicitly provide a way to handle out-of-samples. The starting point of this thesis is a nonlinear technique, called Embedding by Affine Transformations (EAT), which reduces the dimensionality of out-of-sample data as well. In this method, a convex optimization is solved for estimating a transformation between the high-dimensional input space and the low-dimensional embedding space. To the best of our knowledge, EAT is the only distance-preserving method for nonlinear dimensionality reduction capable of handling out-of-samples.

The second method that we propose is TesseraMap. This method is a scalable extension of EAT. Conceptually, TesseraMap partitions the underlying manifold of data into a set of tesserae and then unfolds it by constructing a tessellation in a low-dimensional subspace of the embedding space. Crucially, the desired tessellation is obtained through solving a small semidefinite program; therefore, this method can efficiently handle tens of thousands of data points in a short time.

The final outcome of this thesis is a novel method in dimensionality reduction called Isometric Patch Alignment (IPA). Intuitively speaking, IPA first considers a number of overlapping flat patches, which cover the underlying manifold of the high-dimensional input data. Then, IPA rearranges the patches and stitches the neighbors together on their overlapping parts. We prove that stitching two neighboring patches aligns them together; thereby, IPA unfolds the underlying manifold of data. Although this method and TesseraMap have similar approaches, IPA is more scalable; it embeds one million data points in only a few minutes. More importantly, unlike EAT and TesseraMap, which unfold the underlying manifold by stretching it, IPA constructs the unfolded manifold through patch alignment. We show this novel approach is advantageous in many cases. In addition, compared to the other well-known dimensionality reduction methods, IPA has several important characteristics; for example, it is noise tolerant, it handles non-uniform samples, and it can embed non-convex manifolds properly.

In addition to these three dimensionality reduction methods, we propose a method for subspace clustering called Low-dimensional Localized Clustering (LDLC). In subspace clustering, data is partitioned into clusters, such that the points of each cluster lie close to a low-dimensional subspace. The unique property of LDLC is that it produces localized clusters on the underlying manifold of data. By conducting several experiments, we show this property is an asset in many machine learning tasks. This method can also be used for local dimensionality reduction. Moreover, LDLC is a suitable tool for forming the tesserae in TesseraMap, and also for creating the patches in IPA.

**Acknowledgements**

First of all, I would like to express my gratitude to my supervisor, Professor Ali Ghodsi. I appreciate all his support and advice throughout my doctorate program. Indeed, among the best moments during the Ph.D. studies, I will never forget our friendly discussions about philosophy and politics. I would also like to thank the members of my defense committee, Professor Wayne Oldford, Professor Forbes Burkowski, and Professor Pascal Poupart for their detailed review and helpful comments. I am especially grateful to Professor Poupart and also, Professor Shai Ben-David, for the short periods of time when I was researching under their supervision.

In addition, I extend my appreciation to my remarkable friend and collaborator, Dr. Babak Alipanahi, for all his valuable help and also, for the brainstorming sessions which resulted in a number of research papers and finally this thesis. I would also like to thank the members of the Computational Statistics Research Group, especially Dr. Ahmed Farahat and Dr. Mehrdad Gangeh, for their constructive feedbacks in our discussions.

I sincerely appreciate Professor Yuying Li, for her kind support and follow-ups, both as the graduate director of the school of Computer Science and as the associate dean of graduate studies in the faculty of Mathematics. Moreover, I would like to thank the graduate studies coordinator, Ms. Margaret Towell, who always has helpful solutions for the graduate students' problems. I am also grateful to Ms. Cait Glasson and Ms. Nicole Keshav, who kindly accepted to proofread my thesis and improve its writing. In addition, I appreciate for the financial support, the David R. Cheriton Graduate Scholarship, kindly provided by Professor David Ross Cheriton, and also the Ontario Graduate Scholarship (OGS), provided by the Ontario Ministry of Training, Colleges and Universities.

I am grateful to my beloved parents, Iraj Khajehpour and Sharareh Darvish, because they have always believed in me, because of all the sacrifices they made for their children, and because of their pure love. Also, many thanks to my darling sister, Moozhan, who always brings joy to my heart and pushes me forward in life. And finally, Avid, for her patience, for her kindness, for her sincere support, and for all the energy she has been giving to me during my doctoral studies.

## Dedication

This thesis is dedicated to my dearest parents, Sharareh and Iraj, and also my beloved sister, Moozhan, for their endless support and constant encouragement. In addition, I would like to dedicate this work to my grandmother, Ms. Kowkab Ashraghi, to express my gratitude for her eternal love.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

Data banks have been growing larger in parallel with the recent advancements in information technology. In addition to the large number of data samples that are being collected every second from wide varieties of information sources, another aspect of this growth is the dimensionality of data. An informative sample of data usually consists of several different measurements, which form a high-dimensional vector. Hence, a data bank consisting of these kinds of samples can usually be represented by a high-dimensional matrix.

The need to analyze large volumes of high-dimensional data is increasing. Despite the best efforts of industrial and academic research, the *'curse of dimensionality'* continues to severely challenge machine learning and data mining algorithms. The term *curse* was probably first used by Bellman [5]. This problem typically arises when the dimensionality of data increases, which quickly expands the volume of the input space, such that the available data samples become sparse. Consequently, obtaining a statistically reliable result for any machine learning task needs a number of data samples which usually grows exponentially with the dimensionality. Removing the curse is only possible through reducing the number of dimensions by which a dataset is presented. Thus, there has been sustained interest in dimensionality reduction techniques insightful enough to discover meaningful data patterns, in order to represent the input data appropriately in low-dimensional spaces.

In a high-dimensional dataset, many various dimensions represent a set of data features, which are typically indirect measurements of a single underlying source. In many applications, one does not have access to the underlying source of information; therefore, there is an attempt to gather information from as many different sources as possible which are believed to be related to the desired source. However, the gathered pieces of information can be highly correlated, redundant, or even irrelevant.

Considering a real-world example, suppose our task is to estimate the ages of authors by reading their articles. Since it is unlikely that an author's age would be mentioned, we do not have direct access to the source of information. We can, instead, form a data vector for each author, which consists of the frequency of the words used in the article. Now, we need to learn an estimator function in a space of more than 10,000 dimensions[1]. To learn an accurate estimator we need an adequate number of training samples to cover the input space. Unfortunately, covering the input space is not feasible; acquiring supervised samples is expensive, and even if we had them for free, there is no learning algorithm that can handle such a large number of input samples. The initial observation is that there are many words that contain no information, e.g., the articles, or the conjunction words. Another observation is that many words are synonyms, so having both of them as the dimensions of an article is redundant. Finally, and most importantly, many words are conceptually related. That is, having one of them increases that probability that we will see the other one in the text, e.g., school and teaching, or restaurant and food.

There are two approaches for reducing the dimensionality of data: *feature selection* (also known as variable selection) and *feature extraction*. A feature selection method selects a few dimensions of the input dataset as the most informative features of its data and then, represents the data samples only by those selected features. Feature extraction algorithms represent the input data by few dimensions as well; however, these methods do not select the original dimensions and, instead, transform them to fewer dimensions. For example, in the aforementioned age-estimator task, a feature selection algorithm chooses only a subset of the words and represents the authors by the frequency of those words in their articles. In contrast, a feature extraction algorithm may produce new features by combining the frequency of subsets of the words, e.g., the value assigned to an extracted feature can be a weighted sum of the frequency of these words: school, teacher, and student.

In feature selection, since the redundant dimensions of data are removed, there will usually be a loss of information. This is particularly important when there are highly correlated dimensions which measure a common hidden source, but none of them is enough to accurately explain that source. In contrast, feature extraction can potentially exploit all of the dimensions with the minimum possible information loss. The new computed dimensions seldom have clear meanings by themselves; this is because they are not easily interpretable based on the original features. Nevertheless, the outcome is low-dimensional; therefore, the relations between the data samples can be more evident and hence, data patterns will possibly be revealed. This property is generally considered to be an asset for all of the pattern recognition methods.

[1]Usually the number of different words in a language is more than 10,000.

## 1.1 Problem Statement

The focus of this thesis is mainly on dimensionality reduction by feature extraction. As will be discussed in Section 1.3, existing dimensionality reduction methods unfortunately suffer from certain types of shortcomings. The goal of this research, therefore, is to resolve those problems by designing and proposing novel methods for dimensionality reduction which address the critical requirements of contemporary machine learning applications.

The implicit assumption behind dimensionality reduction is the existence of relationships between features of high-dimensional data, which cause dependency. When there is a dependency between dimensions of a high-dimensional dataset, the data points usually do not span the entire high-dimensional space and, rather, form manifolds in the space. A (smooth) manifold is a topological object which, in small scale, resembles a low-dimensional Euclidean space. That is, a small neighborhood of each point on a manifold forms an affine subspace of a lower dimensionality.

In fact, a common assumption in dimensionality reduction is that the data points lie on, or close to, a smooth manifold in the high-dimensional input space. Hence, the task of dimensionality reduction can be interpreted as computing a low-dimensional *embedding*[2] of the underlying manifold, in which the topological properties are preserved. The embedding of the underlying manifold can be seen as unfolding it in a low-dimensional space, such that the pattern of the input data is preserved over the manifold, according to some criteria. The most intuitive way to describe the pattern of the data points on a manifold is by representing their pairwise distances. Therefore, distance preservation is a popular criterium for preserving the data patterns in embedding.

Based on the discussion above, we define the task of dimensionality reduction as follows: A set of $n$ data points $\{\mathbf{x}_i\}_{i=1}^n$ in the high-dimensional Euclidean space $\mathbb{R}^d$ and also, a target dimensionality $r$ (where $r \ll d$) are given as input. Provided that the data samples are close to an unknown low-dimensional manifold $\mathcal{M}$ in the input space, the goal is to represent the input dataset in an $r$-dimensional space by unfolding the underlying manifold $\mathcal{M}$, such that local pairwise distances are preserved.

[2] In 1936, Hassler Whitney proved [72] that any smooth compact $r$-dimensional manifold can be embedded in a $(2r+1)$ Euclidean space; therefore, from a theoretical point of view, dimensionality reduction by way of embedding is possible, assuming that the data points are close to a low-dimensional manifold.

## 1.2 The Road Map

This thesis consists of my research on dimensionality reduction and has six chapters. Except for the first and last, each chapter proposes a new method. These four chapters are set in a chronological order; while the first section reflects my initial ideas in this field, the reader may find the last chapter more mature and complex.

In the first step, we briefly survey the existing methods; the following section discusses the popular dimensionality reduction methods, and compares them to each other. Then, Chapter 2 explains that the nonlinear dimensionality reduction methods do not provide an explicit way for handling out-of-samples. The first proposed method in this thesis is *Embedding by Affine Transformations (EAT)*, which is a nonlinear dimensionality reduction method that is nevertheless able to map the out-of-sample points. This method achieves its nonlinear power through utilizing kernels. How this problem can be cast as a standard convex optimization will be shown in detail.

In Chapter 3, the concept of EAT will be extended, but with a major focus on scalability. The most crucial drawback shared by all of the existing dimensionality reduction methods (including EAT) is the lack of scalability. Chapter 3 suggests *tessellation* as a novel idea to address this problem, and proposes *TesseraMap*, which is the second contribution of this research. TesseraMap is a nonlinear dimensionality reduction method which scales well with the input size. This method is able to map out-of-samples as well, and has a convex optimization formula.

In the fourth chapter, a local dimensionality reduction method is proposed. In local dimensionality reduction, instead of searching for a global solution, the points are divided into clusters, such that each cluster is represented by few dimensions separately. The proposed method, which is called *Low-dimensional Localized Clustering (LDLC)* is, in fact, a clustering algorithm. The main idea behind this algorithm was first motivated by the need for a suitable tessellation in TesseraMap. This method partitions the points into clusters, which are both localized and low-rank. As explained at the end of the next section, although current local methods are able to produce low-rank clusters, none of them attempts to generate localized clusters on the underlying manifold of data. This may lead to undesirable errors in machine learning tasks, such as classification. Chapter 4 provides an efficient algorithm designed for LDLC, which rapidly converges. In addition, the most popular techniques will be compared, in practice, for some classification tasks.

The most significant contribution of this research is *Isometric Patch Alignment (IPA)*, which is discussed in Chapter 5. At first glance, this dimensionality reduction method can be seen as an extension of TesseraMap; however, there is a major difference between

them. Both EAT and TesseraMap stretch the underlying manifold to unfold it. As will be explained, although this approach employs the data variance, which is the closest relaxation to the rank function in optimization, it has known flaws and there are cases for which it fails. In contrast, IPA proposes a novel idea of patch alignment for unfolding the underlying manifold. While the outcome of IPA is provably low-rank, it does not have the flaws of the stretching approach. More importantly, the computational complexity of IPA does not depend on the input size and, hence, it is capable of handling millions of data samples. IPA has many other advantages: besides an ability to handle noisy data, bad sampling, and non-convex manifolds, it also has a convex optimization. This thesis ends with Chapter 6, which contains the conclusions, in addition to some suggestions about future work.

## 1.3 Background

Dimensionality reduction is one of the most popular approaches to circumvent the problem of the curse of dimensionality. A dimensionality reduction method reduces the dimensionality of data with minimum information loss, which decreases the number of data samples needed for a learning task and consequently, minimizes the learning cost. Overcoming the curse of dimensionality, however, is not the only benefit of dimensionality reduction; a typical method is usually beneficial in many ways:

- *Complexity reduction:* the computational complexities of many machine learning algorithms depends on the dimensionality of the input data. By reducing the dimensionality, the algorithms run faster, and also can accept more data samples.
- *Noise cancellation:* noise is distributed in all dimensions, and can be independent of data. By reducing the dimensionality of data, part of the noise effect is canceled, which improves the signal-to-noise ratio.
- *Data compression:* by reducing the dimensionality of a data bank, the volume of the bank is reduced, which is an advantage in data storing and transferring.
- *Visualization:* observing data in a few dimensions reveals the pattern of data which can be used in designing and analyzing learning algorithms.

In the literature of machine learning, there are many practical applications for dimensionality reduction. In general, dimensionality reduction can be used to raise the classification power [46], for example, in [23] Isomap and LLE are used for image classification, or

in [48] Kernel PCA is employed for microarray data classification. Also, [47] used neural networks for feature extraction and classification of hyperspectral images. In addition, dimensionality reduction can be used in text mining applications; in [8] there are many examples in which dimensionality reduction is used for text classification. Moreover, reducing the dimensionality of data can be helpful in indexing [31]. Finally, there are many recognition tasks in which dimensionality reduction techniques are applied for preprocessing, for example, in [49] for image recognition, and also in [4] for speech recognition.

Many dimensionality reduction algorithms have been proposed, balancing various motivations, tradeoffs, and limitations. In this section, the most relevant methods will be studied. First, the notation used in this thesis is introduced.

### 1.3.1 Notation

The matrices are shown by bold capital letters, e.g., $\mathbf{X}$ and $\mathbf{\Delta}$. The vectors are also bold but in lowercase letters, e.g., $\mathbf{x}$ and $\boldsymbol{\delta}$. The scalars are not bold and they are in lowercase letters, e.g., $n$ and $\rho$. The sets are shown with Euler script capital letters, e.g., $\mathcal{A}$ and $\mathcal{B}$.

The input data is given by a $d \times n$ matrix $\mathbf{X}$, where each column $\mathbf{x}_i, i \in \{1, 2, \ldots, n\}$, represents one data point. Here, $d$ denotes the original dimensionality and $n$ is the number of data points. These points, after unfolding their underlying manifold, are represented by matrix $\mathbf{Y}$, which, like $\mathbf{X}$, has $n$ columns, but the number of rows depends on the unfolding method. The final output is an $r \times n$ matrix denoted by $\mathbf{Z}$, which is usually obtained by projecting $\mathbf{Y}$ into an $r$-dimensional subspace, where $r$ is the target dimensionality. Slightly abusing the notation, we sometimes use a matrix of points as a set, e.g., $\mathbf{x}_i \in \mathbf{X}$.

The rest of the notation is as follows:

| | |
|---|---|
| $\mathbb{R}^d$ | the space of real $d$-dimensional vectors. |
| $\mathbb{R}^{r \times d}$ | the space of real $r \times d$ matrices. |
| $\mathcal{S}_+^r$ | the set of symmetric positive semidefinite $r \times r$ matrices. |
| $\mathbf{I}_r$ | $r \times r$ identity matrix. |
| $\mathbf{1}_r$ | all-one vector of size $r$. |
| $\mathbf{0}_r$ | all-zero vector of size $r$. |
| $\mathbf{e}_i$ | the $i^{\text{th}}$ column of the identity matrix. |
| $\lvert\mathcal{C}\rvert$ | the cardinality of set $\mathcal{C}$. |
| $\lVert\mathbf{x}\rVert$ | the 2-norm of the vector $\mathbf{x}$, which is computed as $\sqrt{\mathbf{x}^\top \mathbf{x}}$. |

In addition, for a matrix $\mathbf{A}$:

| | |
|---|---|
| $\mathbf{A}_{ij}$ | the $(i,j)^{\text{th}}$ entry of $\mathbf{A}$. |
| $\mathbf{A}^\top$ | the transpose of $\mathbf{A}$. |
| $\mathbf{Tr}\{\mathbf{A}\}$ | the trace of $\mathbf{A}$. |
| $\mathbf{Rank}(\mathbf{A})$ | the rank of $\mathbf{A}$. |
| $\mathbf{vec}(\mathbf{A})$ | the vectorizaion operator which stacks all the columns to form one column. |
| $\mathbf{A}^\dagger$ | the pseudo-inverse of $\mathbf{A}$. |
| $\|\mathbf{A}\|_F$ | the Frobenius norm of $\mathbf{A}$, which is computed from $\sqrt{\mathbf{Tr}\{\mathbf{A}^\top\mathbf{A}\}}$. |

### 1.3.2 Linear Methods

Perhaps the most popular techniques for dimensionality reduction are Multi-dimensional Scaling (MDS), and Principal Component Analysis (PCA). These are, in fact, the oldest methods as well. The earliest work on MDS was done by Young and Householder [74] in 1938, and then completed by Torgerson [64] in 1952. Also, PCA was first proposed by Pearson [45] in 1901. The input of MDS is an $n \times n$ matrix representing the pairwise similarities between $n$ data points, and the goal is to find the coordinates of these points in a space of the target dimensionality $r$. If this similarity matrix defines a metric on the points, the algorithm is called metric MDS. The non-metric MDS is out of the scope of this review. The most widely used measure of similarity between two vectors $\mathbf{z}_i$ and $\mathbf{z}_j$ is their scalar product $\mathbf{z}_i^\top \mathbf{z}_j$. Classical metric MDS (cMDS) employs the scalar product as similarity [36], and thus, it is defined by the following optimization:

$$\min_{\mathbf{Z}\in\mathbb{R}^{r\times n}} \left\|\mathbf{G} - \mathbf{Z}^\top\mathbf{Z}\right\|_F, \tag{1.1}$$

where $\mathbf{G}$ is the given Gram matrix of the points[3]. The optimization problem in Eq.1.1 has a closed-form solution based on the *Singular Value Decomposition (SVD)* of the input similarity matrix. The SVD of an $n \times m$ matrix $\mathbf{A}$ is

$$\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^\top, \tag{1.2}$$

where $\mathbf{U}$ is $n\times n$ and $\mathbf{V}$ is $m\times m$ and both of them are orthonomal matrices (i.e., $\mathbf{U}^\top\mathbf{U} = \mathbf{I}_n$ and $\mathbf{V}^\top\mathbf{V} = \mathbf{I}_m$). The matrix $\mathbf{S}$ is an $n \times m$ diagonal matrix, whose diagonal values are the singular values of $\mathbf{A}$. It can be shown [58] that the minimum squared error of the rank $r$ approximation of $\mathbf{A}$ is obtained by:

[3] The Gram matrix of $n$ data points $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ is defined by $\mathbf{G}_{ij} = \mathbf{x}_i^\top \mathbf{x}_j$ or in the matrix form it is defined as $\mathbf{G} = \mathbf{X}^\top\mathbf{X}$.

$$\min_{\mathbf{B}\in\mathbb{R}^{n\times m},\ \mathbf{Rank}(\mathbf{B})=r} \|\mathbf{A}-\mathbf{B}\|_F = \left\|\mathbf{A}-\widetilde{\mathbf{A}}_r\right\|_F, \tag{1.3}$$

where $\widetilde{\mathbf{A}}_r = \mathbf{U}_r\mathbf{S}_r\mathbf{V}_r^\top$, in which $\mathbf{U}_r$ is an $n\times r$ orthonormal matrix formed by $r$ columns of $\mathbf{U}$ which are associated with the largest $r$ singular values of $\mathbf{A}$, and the $m\times r$ orthonormal matrix $\mathbf{V}_r$ is formed similarly from the columns of $\mathbf{V}$. In the same way, $\mathbf{S}_r$ is an $r\times r$ diagonal matrix, whose diagonal elements are the $r$ largest singular values of $\mathbf{A}$. The order of diagonal elements in $\mathbf{S}_r$ should be the same as the order of the columns in $\mathbf{U}_r$ and $\mathbf{V}_r$.

The optimization problem in Eq.1.1 can be solved by computing the rank $r$ approximation of $\mathbf{G}$. Since the Gram matrix of the points is real symmetric, its SVD can be obtained by computing *Eigenvalue Decomposition (EVD)*. That is, the input Gram matrix is decomposed as $\mathbf{G} = \mathbf{U}\mathbf{S}\mathbf{U}^\top$. Based on Eq.1.3, the optimal $\mathbf{Z}$ for the cMDS problem defined in Eq.1.1 can be computed as:

$$\mathbf{Z}^\top\mathbf{Z} = \widetilde{\mathbf{G}}_r = \mathbf{U}_r\mathbf{S}_r\mathbf{U}_r^\top \implies \mathbf{Z} = \mathbf{S}_r^{\frac{1}{2}}\mathbf{U}_r^\top. \tag{1.4}$$

Note that cMDS does not have a unique solution and any rotation of $\mathbf{Z}$ can be considered as an answer with the same similarity matrix. In addition to the aforementioned application, MDS can be used to compute a low-dimensional representation for the points with a given dissimilarity matrix. In many applications, input is an $n\times n$ Euclidean distance matrix (EDM) $\mathbf{D}$ consisting of the pairwise squared Euclidean distances between the points, which represents the pairwise dissimilarities between them. In this case, the goal of the algorithm is to find the best approximation $\mathbf{Z}$, such that $\forall i,j : \|\mathbf{z}_i-\mathbf{z}_j\|^2 \simeq \mathbf{D}_{ij}$. Thus, the solution is obtained by solving the following optimization:

$$\min_{\mathbf{Z}\in\mathbb{R}^{r\times n}} \|\mathbf{D}_\mathbf{Z}-\mathbf{D}\|_F, \tag{1.5}$$

where $\mathbf{D}_\mathbf{Z}$ is the $n\times n$ EDM of $\mathbf{Z}$. One way to solve this optimization is to transform it to the form of Eq.1.1 [16]; it is only required to calculate $\mathbf{G}$ and $\mathbf{G}_\mathbf{Z}$ based on $\mathbf{D}$ and $\mathbf{D}_\mathbf{Z}$ respectively. The following equation shows the relation between the Gram matrix of a set of $n$ centered points and their EDM matrix:

$$\mathbf{G} = -\frac{1}{2}\mathbf{H}_n\mathbf{D}\mathbf{H}_n \quad \text{where} \quad \mathbf{H}_n = \mathbf{I}_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top. \tag{1.6}$$

In Eq.1.6, $\mathbf{H}_n$ is called the centering matrix of size $n$, and this process is known as *double centering*. Based on this equation, it is possible to transform the distance matrices to the Gram matrices, and then compute $\mathbf{Z}$ using Eq.1.4.

Another classical method for dimensionality reduction is PCA, which is closely related to cMDS. Although these two methods have different goals, their solutions are, in fact, very similar. The input to PCA is a set of high-dimensional observations; therefore, unlike MDS, the coordinates of the input points are known (denoted by $\mathbf{X}$). The goal of PCA is to find $r$ independent *principal directions* $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_r$ forming the columns of a $d \times r$ orthonomal matrix $\mathbf{P}$, by which the input data is linearly mapped to its low-dimensional representation, i.e., $\mathbf{Z} = \mathbf{P}^\top \mathbf{X}$.

To find the principal directions, different objectives have been defined for PCA; however, the final mappings are all the same. The most well-known objective of PCA is to search for a set of principal directions that captures the maximum variation of data. Since the directions are independent, the captured variation is the sum of data variances over all of the principal directions. Translating the data points does not alter these variances; hence, for convenience, it is assumed that the input data is centered, i.e. $\mathbf{X}\mathbf{1}_n = \mathbf{0}_d$. The following optimization maximizes the captured variation:

$$\max_{\mathbf{P} \in \mathbb{R}^{d \times r},\, \mathbf{P}^\top \mathbf{P} = \mathbf{I}_r} \mathbf{Tr}\left\{\mathbf{P}^\top \mathbf{X}\mathbf{X}^\top \mathbf{P}\right\}. \tag{1.7}$$

In Eq.1.7, the constraint $\mathbf{P}^\top \mathbf{P} = \mathbf{I}_r$ guarantees the independence of the principal directions. The trace term, in fact, shows the sum of the variances in all of the $r$ principal directions. The inner part of the trace in Eq.1.7 can be replaced by the covariance matrix of the input points, i.e. $\mathbf{C}_\mathbf{X} = \frac{1}{n}\mathbf{X}\mathbf{X}^\top$. Suppose applying EVD to this matrix results in $\mathbf{C}_\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{U}^\top$. It is easy to see that the optimal answer for $\mathbf{P}$ should be a set of $r$ eigenvectors of $\mathbf{C}_\mathbf{X}$ that are associated to the $r$ greatest eigenvalues of $\mathbf{C}_\mathbf{X}$, i.e. $\mathbf{P} = \mathbf{U}_r$.

Another application of PCA is to compute a set of coordinates for the points in an $r$-dimensional subspace[4], such that the reconstruction error is minimized. The objective function, based on this approach, can be formulated as follows:

$$\min_{\mathbf{Z} \in \mathbb{R}^{r \times n},\, \mathbf{P}^\top \mathbf{P} = \mathbf{I}_r} \|\mathbf{X} - \mathbf{P}\mathbf{Z}\|_F. \tag{1.8}$$

In Eq.1.8, it is easy to show that $\mathbf{Z} = \mathbf{P}^\top \mathbf{X}$ and then, the optimal solution for $\mathbf{P}$ is the same as the answer of Eq.1.7. Finally, in another approach, PCA can be defined as computing a linear projection $\mathbf{P}$, which preserves the pairwise distances between the points

[4] An $r$-dimensional subspace of a $d$-dimensional Euclidean space is defined as $\{\mathbf{U}\boldsymbol{\alpha} + \boldsymbol{\beta} \,|\, \boldsymbol{\alpha} \in \mathbb{R}^r\}$, where $\boldsymbol{\beta} \in \mathbb{R}^d$ is an arbitrary point of the subspace and the $d \times r$ orthonormal matrix $\mathbf{U}$ indicates a basis for this subspace. If $\boldsymbol{\beta} = \mathbf{0}_d$ the subspace is linear and otherwise, it is affine.

in the $r$-dimensional representation $\mathbf{Z} = \mathbf{P}^\top \mathbf{X}$ as much as possible. It can be formulated by the following optimizations:

$$\min_{\mathbf{Z}=\mathbf{P}^\top\mathbf{X},\, \mathbf{P}^\top\mathbf{P}=\mathbf{I}_r} \|\mathbf{D}_\mathbf{X} - \mathbf{D}_\mathbf{Z}\|_F \implies \min_{\mathbf{Z}=\mathbf{P}^\top\mathbf{X},\, \mathbf{P}^\top\mathbf{P}=\mathbf{I}_r} \left\|\mathbf{X}^\top\mathbf{X} - \mathbf{Z}^\top\mathbf{Z}\right\|_F . \tag{1.9}$$

The minimizations of Eq.1.9 are equivalent based on the earlier discussion regarding cMDS [16]. The solution of this problem reveals the close relation between cMDS and PCA. Suppose $\mathbf{X}$ is decomposed by SVD as $\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^\top$. Consequently, the input Gram matrix $\mathbf{G} = \mathbf{X}^\top\mathbf{X}$ becomes $\mathbf{G} = \mathbf{V}\mathbf{S}^\top\mathbf{S}\mathbf{V}^\top$. From the solution of cMDS in Eq.1.4, the low-dimensional representation $\mathbf{Z}$ in Eq.1.9 can be obtained by $\mathbf{Z} = \mathbf{S}_r\mathbf{V}_r^\top$. On the other hand, from the solution of PCA in Eq.1.7, $\mathbf{P}$ should be equal to $\mathbf{U}_r$. Therefore, $\mathbf{Z} = \mathbf{P}^\top\mathbf{X} = \mathbf{S}_r\mathbf{V}_r^\top$, which is exactly the same answer as cMDS.

The computational complexity of cMDS mainly depends on the computation of EVD, which for small $r$ is of $\mathsf{O}(rn^2)$; however, for large values of $r$ the practical complexity is of $\mathsf{O}(n^3)$. Thus, computing MDS for large datasets is not feasible. In contrast, PCA can be computed in $\mathsf{O}(r \min\{n^2, d^2\})$, which is a great advantage when $d \ll n$.

Another linear method for dimensionality reduction is *Locality Preserving Projection (LPP)* [43]. While both PCA and cMDS attempt to preserve the global structure of data, LPP assumes an underlying manifold for the input data, and attempts to preserve the pairwise neighborhoods. First, it computes an $n \times n$ adjacency matrix $\mathbf{W}$ for the input points, where simply $\mathbf{W}_{ij} = 1$ if $\mathbf{x}_i$ and $\mathbf{x}_j$ are neighbors and zero otherwise. The neighbors can be detected by applying $k$-nearest neighbor or $\epsilon$-ball algorithms. The goal of LPP is to find a projection $\mathbf{P}$, such that in the low-dimensional representation $\mathbf{Z} = \mathbf{P}^\top\mathbf{X}$ the neighborhoods are preserved. The objective can be represented as follows:

$$\min_{\mathbf{Z}=\mathbf{P}^\top\mathbf{X},\, \mathbf{P}\in\mathbb{R}^{d\times r}} \sum_{1\le i,j\le n} \|\mathbf{z}_i - \mathbf{z}_j\|^2 \mathbf{W}_{ij}. \tag{1.10}$$

In this optimization, the non-neighboring pairs of points have no effect, but when $\mathbf{W}_{ij} = 1$, to minimize Eq.1.10, the distance $\|\mathbf{z}_i - \mathbf{z}_j\|$ is required to be small. Therefore, the minimization attempts to preserve the neighborhood between $\mathbf{z}_i$ and $\mathbf{z}_j$, and so for all other neighboring pairs as well. Note that to avoid the trivial solution $\mathbf{Z} = 0$, a constraint should be added. Using matrix notation, the optimization in Eq.1.10 can be simplified to the following constrained minimization:

$$\min_{\mathbf{P}^\top\mathbf{X}\mathbf{D}\mathbf{X}^\top\mathbf{P}=\mathbf{I}_r} \mathbf{Tr}\left\{\mathbf{P}^\top\mathbf{X}\mathbf{L}\mathbf{X}^\top\mathbf{P}\right\}, \tag{1.11}$$

where $\mathbf{D}$ is an $n \times n$ diagonal matrix whose diagonal values are $\mathbf{D}_{ii} = \sum_j \mathbf{W}_{ij}$, and the $n \times n$ matrix $\mathbf{L}$ is the *Laplacian* of $\mathbf{W}$, defined by $\mathbf{L} = \mathbf{D} - \mathbf{W}$. The constraint $\mathbf{Z}\mathbf{D}\mathbf{Z}^\top = \mathbf{I}_r$ gives special importance to the data points with more neighbors (i.e. with higher $\mathbf{D}_{ii}$), while it makes the dimensions of $\mathbf{Z}$ independent. Interestingly, Eq.1.11 has a closed-form solution. By adding the constraint as a *Lagrangian* term in Eq.1.11, the projection matrix $\mathbf{P}$ can be calculated from the following *Generalized Eigenvalue* problem:

$$\mathbf{X}\mathbf{L}\mathbf{X}^\top\mathbf{P} = \mathbf{X}\mathbf{D}\mathbf{X}^\top\mathbf{P}\mathbf{S}, \tag{1.12}$$

where $\mathbf{S}$ is an $r \times r$ diagonal matrix. To minimize the objective function of LPP, matrix $\mathbf{P}$ should consist of $r$ generalized eigenvectors, which are associated to the smallest $r$ generalized eigenvalues obtained from Eq.1.12. Compared to PCA, LPP does not preserve the distances, however, it is more discriminative; that is, it separates the non-similar groups of data. Therefore, the outcome of LPP is usually better for visualization or classification tasks. Regarding the computational complexity, LPP is of $\mathsf{O}(n^3)$ and hence, solving the optimization of LPP, in practice, is more time-consuming than cMDS and PCA.

In fact, most linear dimensionality reduction methods have relatively low computational costs. Moreover, they provide transformations that can then be used to map new points into the same low-dimensional subspace of $\mathbf{Z}$ and consequently, they can easily handle out-of-sample examples. However, their effectiveness is substantially limited by the linearity of the subspace they reveal and thus, when the underlying manifold is complex, they fail to preserve the pattern of data in the resulting low-dimensional representation.

### 1.3.3 Nonlinear Methods

In order to resolve the problem of dimensionality reduction in nonlinear cases, many nonlinear techniques have been proposed. One of the pioneers of nonlinear dimensionality reduction is *Locally Linear Embedding (LLE)* [50]. This method, preserves the local geometry of the underlying manifold while unfolding it. LLE, like LPP, first finds the set of neighbors of each point in the input high-dimensional space. In fact, finding local neighborhoods is a common practice in most of the dimensionality reduction algorithms that work based on manifold discovery. Suppose $\mathcal{N}_i$ is the set of the neighbors of $\mathbf{x}_i$. LLE requires each point, in the low-dimensional representation $\mathbf{Z}$, to be described by the set of its neighbors, in the same way it is described in the input data $\mathbf{X}$:

$$\mathbf{x}_i \simeq \sum_{\mathbf{x}_j \in \mathcal{N}_i} \mathbf{W}_{ij}\mathbf{x}_j \quad \Longrightarrow \quad \mathbf{z}_i \simeq \sum_{\mathbf{x}_j \in \mathcal{N}_i} \mathbf{W}_{ij}\mathbf{z}_j. \tag{1.13}$$

The first step is to find the weights $\mathbf{W}_{ij}$ by which each point is linearly described by its neighbors in the input space. The weights should be constrained to $\sum_i \mathbf{W}_{ij} = 1$. Then in the second step, these weights are used to compute the low-dimensional representation $\mathbf{Z}$. It can be shown [50] that the optimization of LLE can be formulated as follows:

$$\min_{\mathbf{Z}\in\mathbb{R}^{r\times n},\, \mathbf{Z}\mathbf{Z}^\top=\mathbf{I}_r} \mathbf{Tr}\left\{\mathbf{Z}(\mathbf{I}_n-\mathbf{W})^\top(\mathbf{I}_n-\mathbf{W})\mathbf{Z}^\top\right\}. \tag{1.14}$$

The constraint $\mathbf{Z}\mathbf{Z}^\top = \mathbf{I}_r$ is added to avoid degenerate solutions. This optimization problem is very similar to that of PCA in Eq.1.7; however, in LLE it is minimized based on matrix $\mathbf{Z}$, instead of being maximized based on $\mathbf{P}$. Therefore, similar to PCA, the problem is solved by EVD, but in contrast to PCA, this time the rows of $\mathbf{Z}$ are the eigenvectors of $(\mathbf{I}_n-\mathbf{W})^\top(\mathbf{I}_n-\mathbf{W})$ which are associated to the smallest eigenvalues. Note that since the first eigenvalue is always zero, the first eigenvector is ignored[5], and the second to $(r+1)^{\text{th}}$ eigenvectors are considered for making the low-dimensional representation $\mathbf{Z}$.

LLE is a powerful method which is capable of unfolding nonlinear and complex manifolds. It should be noted that it does not preserve the distances and may change their scale over the embedding. Computationally, LLE is as complex as cMDS, i.e. $\mathsf{O}\left(rn^2\right)$; however, since $\mathbf{W}$ is sparse, each eigenvector can be computed in sub-quadratic time of $n$, which makes LLE comparably faster.

*Local Tangent Space Alignment (LTSA)* [76] is another nonlinear method, which attempts to preserve the local geometry of the underlying manifold. For each data point and its closest neighbors, LTSA defines a *tangent subspace* of dimensionality $r$. That is, for the neighborhood around each point $\mathbf{x}_i$, LTSA searches for a center point $\mathbf{c}_i$, and also a $d\times r$ orthonormal matrix $\mathbf{Q}_i$ representing the basis of the tangent subspace. Meanwhile it computes the coordinates of the points in the tangent spaces indicated by $(\mathbf{Q}_i, \mathbf{c}_i)$:

$$\forall \mathbf{x}_i\in\mathbf{X}: \quad \min_{\mathbf{c}_i,\,\mathbf{Q}_i,\,\mathbf{\Theta}_i} \left\|\mathbf{X}_i-\mathbf{Q}_i\mathbf{\Theta}_i-\mathbf{c}_i\mathbf{1}_{n_i}^\top\right\|_F. \tag{1.15}$$

In this optimization $\mathbf{X}_i$ is a $d\times n_i$ matrix consisting of $n_i$ neighbor points of $\mathbf{x}_i$. The $r\times n_i$ matrix $\mathbf{\Theta}_i$ contains the coordinates of the points in the tangent space. This is in fact equal to applying PCA, locally to the neighborhood of each point; $\mathbf{c}_i$ is the mean of the points in $\mathbf{X}_i$ and the matrices $\mathbf{Q}_i$ and $\mathbf{\Theta}_i$ can be calculated in closed-form. The main idea in LTSA is to align the tangent subspaces such that they construct an unfolded manifold

[5] It can be shown that skipping the first eigenvector, in fact, enforces the constraint $\mathbf{Z}\mathbf{1}_n = \mathbf{0}_n$, which removes an extra degree of freedom from the solution.

and then, to compute the global coordinates of the points, $\mathbf{Z}$, on this manifold. To preserve the local geometry, LTSA assumes that the local coordinates of the points on the tangent subspaces are affinely related to their global coordinates. It can be formulated as:

$$\min_{\mathbf{Z},\mathbf{L}_i} \sum_i \|\mathbf{Z}_i\mathbf{H}_{n_i} - \mathbf{L}_i\mathbf{\Theta}_i\|_F^2 \,, \tag{1.16}$$

where $\mathbf{L}_i$ represents the transformation between the local coordinates $\mathbf{\Theta}_i$ to the global coordinates $\mathbf{Z}_i$, and $\mathbf{H}_{n_i}$ is the centering matrix of size $n_i$. Note that $\mathbf{Z}$ is the union of all $\mathbf{Z}_i$. To avoid degenerate solutions, the constraint $\mathbf{Z}\mathbf{Z}^\top = \mathbf{I}_r$ should be added, by which LTSA enforces an independency between the dimensions in the low-dimensional space.

It can be shown [76] that the final optimization is similar to that of LLE; i.e. $\mathbf{Z}$ is obtained by computing EVD for an $n \times n$ sparse matrix and taking the $r$ eigenvectors corresponding to the second to $(r+1)^{\text{th}}$ smallest eigenvalues. Like LLE, most of the computational burden is due to computing EVD; however, taking advantage of higher sparsity, LTSA can be computed relatively more quickly than LLE.

Another approach for respecting the pattern of data during dimensionality reduction, is to preserve the *geodesic distances* between the data points. This is in contrast with methods like PCA and cMDS, that preserve the pairwise Euclidean distances. The geodesic distance between two points is defined as their shortest distance on their underlying manifold. One approach to estimate the pairwise geodesic distances is to form a neighborhood graph over the data points, and then calculate the pairwise shortest distances for this graph. The neighborhood graph is usually created by connecting each point to its $k$-nearest neighbors in the input space, where the weight of each connection is indicated by the Euclidean distance between its two end points. In the second step, an all-pair shortest path algorithm should be used; since the neighborhood graph is highly sparse, the most efficient choice is *Johnson's algorithm* [28]. This algorithm employs *Dijkstra's algorithm* [17] as an inner subroutine, and if it is implemented by *Fibonacci Heap* [14], it will be able to compute all of the pairwise shortest paths in $\mathsf{O}\left(n^2(\log(n)+k)\right)$.

The first algorithm which proposed this approach was Isomap [61]. In this algorithm, first the pairwise geodesic distances are calculated to form an $n \times n$ squared distance matrix $\mathbf{D}_{\text{Geo}}$. Then cMDS is applied to this dissimilarity matrix to compute the low-dimensional coordinates $\mathbf{Z}$. The entire process can formulated as follows:

$$\mathbf{G} = -\frac{1}{2}\mathbf{H}_n\mathbf{D}_{\text{Geo}}\mathbf{H}_n \;,\; \mathbf{G} = \mathbf{U}\mathbf{S}^2\mathbf{U}^\top \implies \mathbf{Z} = \mathbf{S}_r\mathbf{U}_r^\top. \tag{1.17}$$

One implicit outcome of embedding by Isomap is that the local distances are preserved. There are other attempts to preserve the local distances, e.g., *Distance Preserving Projection (DPP)* [73], none of which are as powerful as Isomap. In DPP, the minimum spanning tree of the neighborhood graph is computed and then, the points are embedded such that the local distances on this graph are preserved by way of triangulation. DPP is an extension to an early work on two-dimensional embedding by Lee *et al.* [38], which has the same triangulation approach.

In fact, there are many methods which are basically extensions of other existing methods in dimensionality reduction. For example, *Linear Local Tangent Space Alignment* [75] is a linear approximation of LTSA, or LPP has been extended for nonlinear cases by *Orthogonal Neighborhood Preserving Projections (ONPP)* [34], and also *Hessian Eigenmaps* [18], which is a method inspired by, and closely related to, two well-known methods: *Laplacian Eigenmaps* [3], and LLE.

Although the quality of embedding in nonlinear dimensionality reduction methods is usually satisfactory, they share one deficiency: they provide an embedding only for the given input dataset, with no straightforward extension for out-of-sample points. This shortcoming makes them unsuitable for supervised machine learning tasks such as classification and regression.

### 1.3.4 Kernel Methods

As discussed in the previous sections, linear dimensionality reduction methods provide mappings, which can be used to embed out-of-samples; however, they are not able to handle nonlinear manifolds. In contrast, nonlinear methods provide high quality results for complex datasets, yet with no direct extension for mapping out-of-sample points to the low-dimensional embedding space. The gap between the nonlinear techniques and the linear methods can be filled by *Kernel methods*.

Intuitively speaking, kernel methods first map the input data to a very high-dimensional feature space, where each dimension is supposed to represent one feature of the data. Suppose function $\phi : \mathbb{R}^d \mapsto \mathbb{F}$ is used to map the data points from the input space $\mathbb{R}^d$ to the feature space $\mathbb{F}$. The problem is then solved in the feature space. In the kernel methods, interestingly, the coordinates of the points in the feature space are not explicitly required. Rather, a method should be formulated based on the inner products of the points in the feature space and therefore, only a kernel function $\kappa : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$ is needed. The kernel function returns the inner products in $\mathbb{F}$ as $\forall \mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^d : \ \kappa(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$. In this way, the method is *Kernelized*.

When a linear method is kernelized, it computes a linear mapping with a set of desired properties, from the feature space to the output space. Since the mapping to the feature space is usually nonlinear, combining the linear method with a kernel grants nonlinear capabilities to the method in solving the problem. The computed linear mapping combined with the mapping to the feature space can then be used for mapping any out-of-sample point. That is, the kernelized linear method still provides a direct mapping, although this mapping is not necessarily linear.

Most linear dimensionality reduction methods can be kernelized. For example, PCA has been extended to *Kernel Principal Component Analysis (KPCA)* [25]. KPCA can be seen as applying PCA to $\phi(\mathbf{X}) = [\phi(\mathbf{x}_1), \phi(\mathbf{x}_2), \ldots, \phi(\mathbf{x}_n)]$, the coordinates of the input points in the feature space, which can be formulated as:

$$\max_{\mathbf{P}^\top \mathbf{P} = \mathbf{I}_r} \mathbf{Tr}\left\{\mathbf{P}^\top \phi(\mathbf{X}) \phi(\mathbf{X})^\top \mathbf{P}\right\}. \tag{1.18}$$

To solve this optimization problem, without loss of generality, it is possible to assume that the columns of the projection matrix $\mathbf{P}$ can be re-expressed as linear combinations of the points in the feature space $\mathbb{F}$ [53], i.e. $\mathbf{P} = \phi(\mathbf{X})\mathbf{\Omega}$. This is a direct result of the *Representor Theorem* [54]. Thus, Eq.1.18 can be reformulated as:

$$\max_{\mathbf{\Omega}^\top \phi(\mathbf{X})^\top \phi(\mathbf{X}) \mathbf{\Omega} = \mathbf{I}_r} \mathbf{Tr}\left\{\mathbf{\Omega}^\top \phi(\mathbf{X})^\top \phi(\mathbf{X}) \phi(\mathbf{X})^\top \phi(\mathbf{X}) \mathbf{\Omega}\right\} \implies \max_{\mathbf{\Omega}^\top \mathbf{K} \mathbf{\Omega} = \mathbf{I}_r} \mathbf{Tr}\left\{\mathbf{\Omega}^\top \mathbf{K} \mathbf{K} \mathbf{\Omega}\right\}, \tag{1.19}$$

where $\mathbf{K} = \phi(\mathbf{X})^\top \phi(\mathbf{X})$ is called *Kernel Matrix*, and its elements are simply computed as $\mathbf{K}_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$. It is clear that only having $\mathbf{K}$ is enough to solve Eq.1.19, and knowing $\phi(\mathbf{X})$ is not necessary. In fact, usually in the kernel methods, the explicit form of the mapping $\phi(\cdot)$ is unknown; instead, only the kernel function $\kappa(\cdot, \cdot)$ is defined to represent the pairwise similarity between the points in the feature space. Any function can be considered as a kernel function, as long as it meets the condition of *Mercer's Theorem* [42]:

$$\exists \phi : \kappa(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})^\top \phi(\mathbf{y}) \quad \iff \quad \forall g : \int \kappa(\mathbf{x}, \mathbf{y}) g(\mathbf{x}) g(\mathbf{y}) \, d\mathbf{x} \, d\mathbf{y} \geq 0. \tag{1.20}$$

The optimization of Eq.1.19 is an eigenvalue problem (considering that $\mathbf{Rank}(\mathbf{K}) \geq r$) and can be computed in $\mathsf{O}(n^3)$. Note that it is necessary to be certain that $\phi(\mathbf{X})$ is centered, i.e. $\overline{\phi}(\mathbf{x}_i) = \phi(\mathbf{x}_i) - \frac{1}{n}\phi(\mathbf{X})\mathbf{1}_n$. Centering is basically a translation in the feature

space, and should be applied to any new out-of-sample point as well. However, since $\phi(\cdot)$ is unknown, the kernel function $\kappa(\cdot,\cdot)$ should be modified for centering:

$$\begin{aligned}\overline{\kappa}(\mathbf{x}_i,\mathbf{x}_j) = \overline{\phi}(\mathbf{x}_i)^\top\,\overline{\phi}(\mathbf{x}_j) &= \left(\phi(\mathbf{x}_i) - \frac{1}{n}\phi(\mathbf{X})\mathbf{1}_n\right)^\top \left(\phi(\mathbf{x}_j) - \frac{1}{n}\phi(\mathbf{X})\mathbf{1}_n\right) \\ &= \kappa(\mathbf{x}_i,\mathbf{x}_j) + \frac{1}{n^2}\mathbf{1}_n^\top\mathbf{K}\mathbf{1}_n - \frac{1}{n}\left(\kappa(\mathbf{x}_i,\mathbf{X}) + \kappa(\mathbf{x}_j,\mathbf{X})\right)\mathbf{1}_n,\end{aligned} \tag{1.21}$$

where $\kappa(\mathbf{x}_i,\mathbf{X})$ is a $1\times n$ vector of the form $[\kappa(\mathbf{x}_i,\mathbf{x}_1),\,\kappa(\mathbf{x}_i,\mathbf{x}_2),\,\ldots,\,\kappa(\mathbf{x}_i,\mathbf{x}_n)]$. Note that $\mathbf{x}_i$ and $\mathbf{x}_j$ can be any out-of-sample point. Inheriting from PCA, KPCA provides a linear transformation for embedding, by which all of the pairwise distances defined in the feature space $\mathbb{F}$, are preserved. Therefore, KPCA relies on the strength of its kernel function to unfold the underlying manifold of data and to reveal low-dimensional structures in the feature space. Unfortunately, common kernels such as RBF kernels and polynomial kernels generally perform poorly in manifold learning problems, which is perhaps what motivated the development of nonlinear algorithms, such as LLE and Isomap.

Another method that can be kernelized is LPP. Following the same approach from KPCA, which in the literature is called the *Kernel Trick*, Eq.1.11 can be kernelized as:

$$\min_{\mathbf{\Omega}^\top\mathbf{K}\mathbf{D}\mathbf{K}\mathbf{\Omega}=\mathbf{I}_r} \mathbf{\Omega}^\top\mathbf{K}\mathbf{L}\mathbf{K}\mathbf{\Omega}. \tag{1.22}$$

Similar to LPP, the transformation matrix $\mathbf{\Omega}$ can be computed by solving a generalized eigenvalue problem. Adding the nonlinear power of kernels to LPP improves the quality of embedding in kernel LPP [43]. Note that unlike KPCA which preserves the pairwise distances which are defined in the feature space, kernel LPP preserves the neighborhood of the points, which are defined in the original input space.

Another direction of research is motivated by the fact that many prominent nonlinear dimensionality reduction algorithms can be formulated as Kernel PCA [25]. It has been realized that the difference among many of these, apparently different, algorithms lies mainly in the choice of kernel. In fact, the properties and behavior of each algorithm are encoded in its kernel. For example, in Eq.1.17, the matrix $\mathbf{G}$, which is calculated based on $\mathbf{D}_{\text{Geo}}$, plays the role of a kernel matrix. Also, the outcome of $\lambda\mathbf{I}_n - (\mathbf{W}-\mathbf{I}_n)^\top(\mathbf{W}-\mathbf{I}_n)$ in Eq.1.14 can be considered as another kernel matrix. Except kernel PCA with a closed-form kernel, all of these algorithms are based on data-driven kernels, and therefore, have no straightforward extension for out-of-samples. However, in [6], based on the *Nyström method*, a unified framework for a number of nonlinear methods has been suggested, which can be used for embedding new out-of-samples.

### 1.3.5 Using Semidefinite Programming

Weinberger and Saul in 2004 showed that the crucial problem of choosing an appropriate kernel for dimensionality reduction can be cast as an instance of semidefinite programming (SDP) [69]. A semidefinite programming problem is a convex optimization which can be formulated (in the standard form) as [12]:

$$\min_{\mathbf{X}\in\mathbf{S}^n} \mathbf{Tr}\left\{\mathbf{CX}\right\} \quad \text{s.t.} \quad \mathbf{X} \succeq 0 \quad \text{and} \quad \forall i = 1 \ldots m : \mathbf{Tr}\left\{\mathbf{A}_i\mathbf{X}\right\} = b_i, \tag{1.23}$$

and in the dual form:

$$\max_{\mathbf{y}\in\mathbb{R}^m} \mathbf{b}^\top\mathbf{y} \quad \text{s.t.} \quad \sum_{i=1}^{m} y_i\mathbf{A}_i \preceq \mathbf{C}. \tag{1.24}$$

Weinberger's approach tackles the problem of dimensionality reduction by kernel PCA. Unlike classical kernel PCA, however, this approach avoids using a predefined closed-form or data-driven kernel. Instead it defines a set of desired properties for the low-dimensional representation and then estimates a kernel matrix optimally using SDP. Following this approach, many similar solutions have been proposed [55, 57, 71, 56] for dimensionality reduction. For example, in [69, 55] the kernel is constructed for preserving the local pairwise distances, or in [56] it is estimated just for preserving the neighborhoods.

These algorithms usually provide a faithful embedding for the input dataset; however, they also usually suffer from three problems. The first problem with all of these algorithms is the computational complexity of SDP. Most widely used SDP solvers (e.g., Sedumi [59], CSDP [9], SDPT3 [63]) are based on the Primal-Dual interior point method. Even after exploiting the sparsity[6] of the constraint matrix, each iteration of SDP takes $\mathsf{O}\left(n^3 + m^3\right)$ time and $\mathsf{O}\left(n^2 + m^2\right)$ space, where $n$ is the size of the SDP matrix (i.e. the number of points) and $m$ is the number of constraints [10]. Therefore, only problems of limited size can be directly solved using current SDP-based algorithms.

Due to this limitation, some large-scale variations of the SDP-based methods (e.g., Landmark SDE [70] and Fast MVU [71]) have been proposed. These techniques are relatively fast and scalable to large datasets. However, they reduce the size of SDP by way of approximation, which results in suboptimal solutions that require post-processing by a local gradient descent search method.

[6] It is important to note that for dense constraint matrices the computational complexity of each step is of $\mathsf{O}\left(mn^3 + m^3 + m^2n^2\right)$ [10].

Another shortcoming of the existing SDP-based methods is that they provide a low-dimensional representation only for the given input data, with no straightforward extension for out-of-sample points. Considering the computational load of these methods, it is not practical to include any new point in the input dataset, and rerun the algorithm. Therefore, this shortcoming makes difficulties in many supervised machine learning tasks.

The last problem with using SDP is regarding optimizing or constraining the rank of kernel matrices; since the rank function is not convex, there is no direct way to minimize the rank of a kernel matrix in an optimization. Thus, applying PCA to the estimated kernel deforms the structure of the manifold in the low-dimensional embedding and consequently, some information will be lost.

The closest convex envelope that approximates the rank of a matrix is its trace [20]. In practice, however, trace minimization results in folding the underlying manifold of data in a low-dimensional subspace. To circumvent the rank optimization problem in dimensionality reduction, the optimization is generally relaxed by another convex objective function, for example, the total variance of the data points in the final embedding, as employed by the MVU-inspired methods. By maximizing the variance, one attempts to unfold the underlying manifold of data, thereby minimizing the rank of the embedding. In some cases, however, this approach does not result in the desired low-rank solutions [55].

### 1.3.6 Subspace Clustering

Since a manifold locally resembles a low-dimensional subspace, it can be modeled by a configuration of some affine subspaces. To this end, *Subspace Clustering* algorithms are introduced which partition the data points into clusters, such that the points of each cluster lie on, or close to, a low-dimensional affine subspace. This is why these methods are also known as *Local Dimensionality Reduction*. Modeling a manifold by a number of subspaces can be beneficial for many unsupervised machine learning tasks, such as data compression, data visualization, image segmentation, and noise cancelation. In addition, this technique can be applied to semi-supervised and even some supervised problems, such as semi-supervised classification, image recognition, and piecewise regression.

Among the subspace clustering methods, *Mixture of Probabilistic PCA (MPPCA)* [62] and *Mixture of Factor Analyzers (MFA)* [26] are the most popular. MPPCA is an extension to the probabilistic view of PCA (PPCA), and MFA is an extension of *Factor Analysis (FA)* [2]. From the probabilistic point of view, dimensionality reduction can be interpreted as finding a set of *Latent Variables* that more likely produce the given input data. Usually the following relation between the observations and the latent variables is assumed:

$$\mathbf{x} = \mathbf{P}\mathbf{z} + \boldsymbol{\mu} + \boldsymbol{\epsilon}, \tag{1.25}$$

where $\boldsymbol{\epsilon}$ represents the noise, $\mathbf{x}$ is the high-dimensional observation (generating the input data), $\mathbf{z}$ is a low-dimensional latent variable, and the combination of the transformation matrix $\mathbf{P}$ and the translation vector $\boldsymbol{\mu}$ indicates the parameters of an affine relationship between $\mathbf{x}$ and $\mathbf{z}$. Note that $\mathbf{x}$, $\mathbf{z}$, and $\boldsymbol{\epsilon}$ are all random variables. The latent variable is assumed to be $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_r)$ and for the noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Psi})$, in which $\boldsymbol{\Psi}$ is diagonal. $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Psi})$ is the following Gaussian distribution:

$$p(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Psi}) = (2\pi)^{-\frac{d}{2}} |\boldsymbol{\Psi}|^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Psi}^{-1}(\mathbf{x}-\boldsymbol{\mu})}. \tag{1.26}$$

It is easy to show that, under the aforementioned assumptions, the observation variable will have a Gaussian distribution as well, such that $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Psi} + \mathbf{P}\mathbf{P}^\top)$. Both PPCA and FA use Eq.1.25 to relate the random variables to each other, however, in PPCA, the noise is assumed to be isotropic, i.e. $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma\mathbf{I}_d)$. This leads to an important difference between them: PCA has a closed-form solution, but FA does not. MFA and MPPCA assume a mixture of latent variables for generating the observation variable $\mathbf{x}$. Since there is no closed-form solution for their maximum likelihood optimizations, the posterior probability, $p(\mathbf{z}|\mathbf{x})$, in these methods is calculated by *Expectation-Maximization (EM)*. The calculated values of the posteriors are then used for soft partitioning of the input points. Considering that the optimization is not convex, the common practice is to run EM several times with different random initializations.

In addition to the probabilistic methods, there are methods for subspace clustering that have been proposed based on geometric perspectives. For example *Vector Quantization Principal Component Analysis (VQPCA)* [30], in which the goal is to partition the points to clusters such that the reconstruction distances are minimized. That is, the sum of distances between the points and their affine subspaces should be minimized. For the given number of clusters $c$, target dimensionality $r$, and the input data points in matrix $\mathbf{X} \in \mathbb{R}^{d \times n}$, the problem can be formulated as follows:

$$\min_{(\boldsymbol{\mu}_i, \mathbf{P}_i, \mathcal{C}_i)_{1 \le i \le c}} \quad \sum_{1 \le i \le c} \sum_{\mathbf{x}_j \in \mathcal{C}_i} \left( \|\mathbf{x}_j - \boldsymbol{\mu}_i\|^2 - \left\|\mathbf{P}_i^\top (\mathbf{x}_j - \boldsymbol{\mu}_i)\right\|^2 \right), \tag{1.27}$$

where $(\boldsymbol{\mu}_i, \mathbf{P}_i, \mathcal{C}_i)$ represents the center point, orthonormal basis, and members of cluster $\mathcal{C}_i$. If all $\mathcal{C}_i$ are known, the other variables can be calculated using PCA, and if the subspace parameters are known, each point should be simply assigned to the closest subspace. Thus,

an iterative scheme can be used for solving Eq.1.27. However, due to non-convexity it may result in local minima and therefore, the algorithm should be rerun with several random initializations in order to find the global minimum.

There are other methods proposed for subspace clustering, such as *K-Subspace Clustering* [68] and *Generalized Principal Component Analysis (GPCA)* [67]. Usually these methods, like VQPCA, result in a firm partitioning of the points. In general, the firm partitioning methods are more successful than the mixture methods (MPPCA and MFA) in forming low-rank clusters, although they tend to generate clusters that are not as localized as those of the mixture methods.

All of these methods are different instances of the family of subspace clustering algorithms (for more detail see [66] and [44]). The algorithms of this family all find low-dimensional subspaces successfully, but fail to generate compact and localized clusters in many cases. This is mainly because these methods do not respect the topology of the underlying manifold.

# Chapter 2

# Learning Affine Transformations for Nonlinear Dimensionality Reduction

In this chapter, we propose a novel dimensionality reduction method, called *Embedding by Affine Transformations (EAT)*. Although the proposed method is a nonlinear technique, it has the advantage of linear techniques as well. A linear dimensionality reduction method, based on its input dataset, estimates a linear transformation, that projects the high-dimensional data points to a low-dimensional subspace. This transformation can then be used to reduce the dimensionality of a new set of points, by projecting them into the same subspace. Thus, the linear methods are capable of handling out-of-samples. The effectiveness of these methods, however, is limited by the linearity of the resulting transformation. In contrast, nonlinear techniques have been proposed which are able to produce high quality results in nonlinear cases. Their solutions, however, are provided only for the given input dataset and hence, they have no straightforward extension for new out-of-sample points.

Interestingly, kernel methods, for example KPCA, have both of the advantages; they can handle nonlinear cases by mapping the input data to a very high-dimensional feature space, and also they are able to map out-of-samples by computing a linear transformation from the feature space to the embedding space. It is important to note that most nonlinear dimensionality reduction methods can be formulated as KPCA, yet they have data-driven kernels and therefore, they do not provide a mapping to be used for out-of-sample points.

In most kernel methods the choice of kernel is crucial. For example, KPCA preserves the pairwise distances which are measured in the feature space; therefore, its embedding completely depends on the choice of kernel for unfolding the underlying manifold. Un-

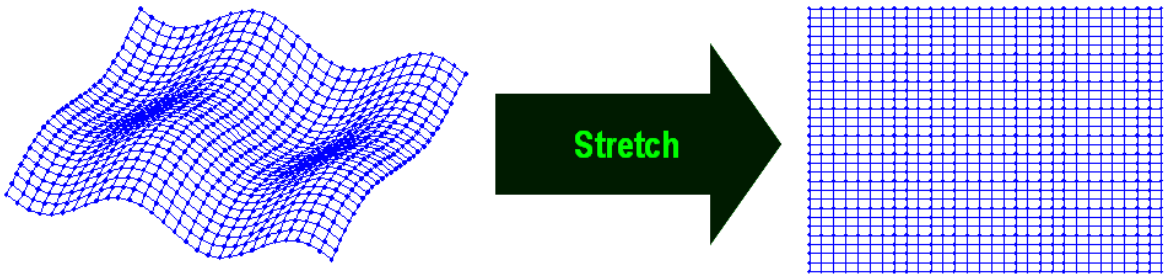


Figure 2.1: A two-dimensional manifold is represented in a higher-dimensional space. Stretching out the manifold allows it to be correctly embedded in a two-dimensional space.

fortunately, common kernels, such as Gaussian or polynomial kernels, generally perform poorly in manifold unfolding.

The proposed method, EAT, similarly to KPCA, maps the input points into a high-dimensional feature space. The similarity ends here, however, as we explicitly search for a linear transformation that unfolds the underlying manifold, while preserving only the local distances measured in the input space. In contrast to KPCA, the proposed method does not expect the kernel to reveal the underlying structure of the data in the feature space. The kernel simply helps to make use of the *blessing of dimensionality*. That is, when the dimensionality of a dataset exceeds its quantity (i.e., the number of samples), a linear transformation can span the entire embedding space. Therefore, it is possible to find a linear transformation that preserves the distances between the neighbors, and also pulls the non-neighbors apart, consequently, flattening the manifold. This idea for unfolding the underlying manifold is depicted in Fig.2.1.

The search for the desired linear transformation can be formulated as an instance of semidefinite programming, which is convex and therefore, always converges to a global optimum. As mentioned, the resulting transformation can be used to map out-of-sample points into the low-dimensional embedding space.

After this brief introduction, which intuitively explains the main idea behind EAT, in Section 2.1, the mathematical details of the proposed method are presented in four parts. First, we will explain how local distances can be preserved. Then, in the second part, the stretching of the underlying manifold is formulated. In the third part, EAT is kernelized, and finally, in the fourth part, an algorithm is proposed for EAT. In Section 2.2, some experimental results are shown, and at the end, in Section 2.3, different aspects of the proposed method are discussed.

## 2.1 The Proposed Method

We would like to find a linear transformation that preserves the distances between neighboring points, while pulling non-neighbor points as far apart as possible. In a high-dimensional space, this transformation looks locally like an affine transformation, consisting of a rotation plus a translation, which leads to a local isometry; however, for non-neighboring points, it acts as a scaling.

Consider an input dataset consisting of $n$ points in a $d$-dimensional space. The input is denoted by a $d \times n$ matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_n]$, where each column $\mathbf{x}_i$ represents a high-dimensional point in $\mathbb{R}^d$. We wish to compute a $d \times d$ transformation matrix $\mathbf{W}$, which unfolds the underlying manifold of the data points, and embeds it into a low-dimensional subspace, such that the input points are mapped into $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \dots \mathbf{y}_n]$ by:

$$\mathbf{y} = \mathbf{W}^\top \mathbf{x}. \tag{2.1}$$

This mapping does not change the dimensionality of the data, that is, $\mathbf{Y}$ has the same dimensionality as $\mathbf{X}$. Rather, the transformed data points in $\mathbf{Y}$ are expected to lie on, or close to, a low-dimensional linear subspace. Therefore, to reduce the dimensionality of $\mathbf{Y}$, one may simply apply PCA to obtain the orthonormal basis of the linear subspace. First, we must define two disjoint sets of pairs:

$$\begin{aligned} \mathcal{S} =& \{(i,j) |\ \ \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ are neighbors}\} \\ \mathcal{O} =& \{(i,j) |\ \ \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ are non-neighbors}\} \end{aligned}$$

The first set consists of the neighboring pairs of points in the original space, for which their pairwise distances should be preserved. These pairs can be identified, for example, by computing a neighborhood graph using the $k$-nearest neighbor (KNN) algorithm. The second set is the set of the non-neighboring points, which we would like to pull as far apart as possible. This set can simply include all of the pairs that are not in the first set.

### 2.1.1 Preserving Local Distances

Assume for all $(i, j)$ in the given set $\mathcal{S}$, the target distances are known as $\tau_{ij}$. We specify the following cost function to preserve these distances:

$$\sum_{(i,j)\in\mathcal{S}} \left(\|\mathbf{y}_i - \mathbf{y}_j\|^2 - \tau_{ij}^2\right)^2, \tag{2.2}$$

and then, we normalize it to obtain[1]:

$$Err = \sum_{(i,j)\in\mathcal{S}} \left(\|\frac{\mathbf{y}_i - \mathbf{y}_j}{\tau_{ij}}\|^2 - 1\right)^2. \tag{2.3}$$

By substituting Eq.2.1 into Eq.2.3, we have:

$$Err = \sum_{(i,j)\in\mathcal{S}} \left((\frac{\mathbf{x}_i - \mathbf{x}_j}{\tau_{ij}})^\top \mathbf{W}\mathbf{W}^\top (\frac{\mathbf{x}_i - \mathbf{x}_j}{\tau_{ij}}) - 1\right)^2 = \sum_{(i,j)\in\mathcal{S}} \left(\boldsymbol{\delta}_{ij}^\top \mathbf{A}\boldsymbol{\delta}_{ij} - 1\right)^2, \tag{2.4}$$

where $\boldsymbol{\delta}_{ij} = \frac{\mathbf{x}_i - \mathbf{x}_j}{\tau_{ij}}$ and $\mathbf{A} = \mathbf{W}\mathbf{W}^\top$ is clearly a positive semidefinite (PSD) matrix. It can be verified that:

$$\boldsymbol{\delta}_{ij}^\top \mathbf{A}\boldsymbol{\delta}_{ij} = \mathbf{vec}(\mathbf{A})^\top \mathbf{vec}(\boldsymbol{\delta}_{ij}\boldsymbol{\delta}_{ij}^\top) = \mathbf{vec}(\mathbf{A})^\top \mathbf{vec}(\boldsymbol{\Delta}_{ij}), \tag{2.5}$$

where $\mathbf{vec}(\cdot)$ simply rearranges a matrix into a vector by concatenating its columns, and $\boldsymbol{\Delta}_{ij} = \boldsymbol{\delta}_{ij}\boldsymbol{\delta}_{ij}^\top$. For a symmetric matrix $\mathbf{A}$ we know:

$$\mathbf{vec}(\mathbf{A}) = \mathbf{D}_d \mathbf{vech}(\mathbf{A}) \tag{2.6}$$

where $\mathbf{vech}(\cdot)$ is the half-vectorization operator, and $\mathbf{D}_d$ is a unique $d^2 \times \frac{d(d+1)}{2}$ duplication matrix. Similar to the $\mathbf{vec}(\cdot)$ operator, the half-vectorization operator rearranges a matrix into a vector by concatenating its columns; however, it stacks the columns from the principal diagonal downwards in a column vector. In other words, a symmetric matrix of size $d$ will be rearranged to a column vector of size $d^2$ by the $\mathbf{vec}(\cdot)$ operator, whereas $\mathbf{vech}(\cdot)$ will stack it into a column vector of size $\frac{d(d+1)}{2}$. This can significantly reduce the

[1] Eq.2.2 corresponds to the assumption that noise is additive, i.e. $\|\mathbf{y}_i - \mathbf{y}_j\|^2 = \tau_{ij}^2 + \varepsilon_{ij}$ where $\varepsilon_{ij}$ is the additive noise. Then clearly $\varepsilon_{ij} = \left(\|\mathbf{y}_i - \mathbf{y}_j\|^2 - \tau_{ij}^2\right)^2$. In contrast, Eq.2.3 captures a multiplicative error, i.e $\|\mathbf{y}_i - \mathbf{y}_j\|^2 = \tau_{ij}^2 + \varepsilon_{ij} \times \tau_{ij}^2$, hence $\varepsilon_{ij} = \left(\|\frac{\mathbf{y}_i - \mathbf{y}_j}{\tau_{ij}}\|^2 - 1\right)^2$.

number of unknown variables, especially when $d$ is large. $\mathbf{D}_d$ is a unique constant matrix. For example, for a $2 \times 2$ symmetric matrix $\mathbf{A}$ we have:

$$\mathbf{vec}(\mathbf{A}) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{vech}(\mathbf{A}).$$

Since both $\mathbf{A}$ and $\mathbf{\Delta}_{ij}$ are symmetric matrices, we can rewrite Eq.2.5 using $\mathbf{vech}(\cdot)$ and reduce the size of the problem:

$$\boldsymbol{\delta}_{ij}^\top \mathbf{A} \boldsymbol{\delta}_{ij} = \mathbf{vech}(\mathbf{A})^\top \mathbf{D}_d^\top \mathbf{D}_d \mathbf{vech}(\mathbf{\Delta}_{ij}) = \mathbf{vech}(\mathbf{A})^\top \boldsymbol{\xi}_{ij}, \tag{2.7}$$

where $\boldsymbol{\xi}_{ij} = \mathbf{D}_d^\top \mathbf{D}_d \mathbf{vech}(\mathbf{\Delta}_{ij})$. Using Eq.2.7, we can reformulate Eq.2.4 as:

$$Err = \sum_{(i,j)\in\mathcal{S}} (\mathbf{vech}(\mathbf{A})^\top \boldsymbol{\xi}_{ij} - 1)^2 = \mathbf{vech}(\mathbf{A})^\top \mathbf{Q} \mathbf{vech}(\mathbf{A}) - 2\mathbf{vech}(\mathbf{A})^\top \mathbf{p} + |\mathcal{S}|, \tag{2.8}$$

where $\mathbf{Q} = \sum_{(i,j)\in\mathcal{S}} \boldsymbol{\xi}_{ij} \boldsymbol{\xi}_{ij}^\top$ and $\mathbf{p} = \sum_{(i,j)\in\mathcal{S}} \boldsymbol{\xi}_{ij}$, and $|\mathcal{S}|$ in Eq.2.8 denotes the number of elements in $\mathcal{S}$, which is constant and can be dropped from the optimization. Now, we can decompose the matrix $\mathbf{Q}$ using EVD to obtain:

$$\mathbf{Q} = \mathbf{U} \mathbf{\Lambda} \mathbf{U}^\top. \tag{2.9}$$

If $\mathbf{Rank}(\mathbf{Q}) = r$, then $\mathbf{U}$ is a $\frac{d(d+1)}{2} \times r$ matrix with $r$ orthonormal basis vectors (i.e. $\mathbf{U}^\top \mathbf{U} = \mathbf{I}$). We denote the null space of $\mathbf{Q}$ by $\overline{\mathbf{U}}$. Any vector of size $\frac{d(d+1)}{2}$, including vector $\mathbf{vech}(\mathbf{A})$, can be represented using the space and the null space of $\mathbf{Q}$:

$$\mathbf{vech}(\mathbf{A}) = \mathbf{U}\boldsymbol{\alpha} + \overline{\mathbf{U}}\boldsymbol{\beta}. \tag{2.10}$$

In Eq.2.10, $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are vectors of size $r$ and $\left(\frac{d(d+1)}{2} - r\right)$ respectively. Since $\mathbf{Q}$ is the summation of $\boldsymbol{\xi}_{ij}\boldsymbol{\xi}_{ij}^\top$, and $\mathbf{p}$ is the summation of $\boldsymbol{\xi}_{ij}$, it is easy to verify that $\mathbf{p}$ is in the space of $\mathbf{Q}$ and therefore $\overline{\mathbf{U}}^\top \mathbf{p} = 0$. By substituting Eq.2.9 and Eq.2.10 in Eq.2.8, the cost function can be expressed as:

$$\mathbf{vech}(\mathbf{A})^\top \left(\mathbf{Q}\mathbf{vech}(\mathbf{A}) - 2\mathbf{p}\right) = \boldsymbol{\alpha}^\top \left(\boldsymbol{\Lambda}\boldsymbol{\alpha} - 2\mathbf{U}^\top\mathbf{p}\right). \tag{2.11}$$

The only unknown variable in this equation is $\boldsymbol{\alpha}$. Hence, by taking the derivative with respect to $\boldsymbol{\alpha}$, and setting the result to zero, the objective function in Eq.2.11 is minimized[2], and the unknown variable $\boldsymbol{\alpha}$ can be obtained in closed-form as:

$$\boldsymbol{\alpha} = \boldsymbol{\Lambda}^{-1}\mathbf{U}^\top\mathbf{p}. \tag{2.12}$$

Interestingly, Eq.2.11 does not depend on $\boldsymbol{\beta}$. That is, the transformation $\mathbf{A}$, which preserves the distances in $\mathcal{S}$ (local distances), is not unique. In fact, by choosing different values for $\boldsymbol{\beta}$, Eq.2.10 defines a family of transformations, all of which preserve the local distances. In this family, we search for the one that is positive semidefinite and increases the distances of the pairs in set $\mathcal{O}$ as much as possible. Now, we show how the freedom of vector $\boldsymbol{\beta}$ can be exploited to search for a transformation that satisfies these conditions.

### 2.1.2 Stretching Non-local Distances

We define the following objective function which, when optimized, attempts to maximize the squared distances between the non-neighbor points. That is, it attempts to stretch the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ if $(i,j) \in \mathcal{O}$.

$$Str = \sum_{(i,j)\in\mathcal{O}} \frac{||\mathbf{y}_i - \mathbf{y}_j||^2}{\tau_{ij}^2}. \tag{2.13}$$

Similar to the cost function $Err$ defined in Eq.2.4 in the previous part, we have:

$$\begin{aligned} Str &= \sum_{(i,j)\in\mathcal{O}} \left( (\frac{\mathbf{x}_i - \mathbf{x}_j}{\tau_{ij}})^\top \mathbf{W}\mathbf{W}^\top (\frac{\mathbf{x}_i - \mathbf{x}_j}{\tau_{ij}}) \right) \\ &= \sum_{(i,j)\in\mathcal{O}} \boldsymbol{\delta}_{ij}^\top \mathbf{A} \boldsymbol{\delta}_{ij} = \sum_{(i,j)\in\mathcal{O}} \mathbf{vech}(\mathbf{A})^\top \boldsymbol{\xi}_{ij} = \mathbf{vech}(\mathbf{A})^\top \mathbf{s}, \end{aligned} \tag{2.14}$$

where $\mathbf{s} = \sum_{(i,j)\in\mathcal{O}} \boldsymbol{\xi}_{ij}$. Then, the optimization problem is:

[2] Since the matrix $\mathbf{Q}$ in the quadratic objective function Eq.2.11 is PSD, the obtained critical point for $\boldsymbol{\alpha}$ is a global minimum.

$$\max_{\mathbf{A}\succeq 0} \quad \mathbf{vech}(\mathbf{A})^\top \mathbf{s}. \tag{2.15}$$

Recall that $\mathbf{vech}(\mathbf{A}) = \mathbf{U}\boldsymbol{\alpha} + \overline{\mathbf{U}}\boldsymbol{\beta}$, and $\boldsymbol{\alpha}$ is already determined from Eq.2.12. So the problem[3] can be simplified as:

$$\max_{\mathbf{A}\succeq 0} \quad \boldsymbol{\beta}^\top \overline{\mathbf{U}}^\top \mathbf{s}. \tag{2.16}$$

Clearly if $\mathbf{Q}$ is full-rank, then the matrix $\overline{\mathbf{U}}$ (i.e. the null space of $\mathbf{Q}$) does not exist and therefore, it is not possible to stretch the non-local distances. However, it can be shown that if the dimensionality of the data is more than its quantity, $\mathbf{Q}$ is always rank-deficient, and $\overline{\mathbf{U}}$ exists. The rank of $\mathbf{Q}$ is at most $|\mathcal{S}|$, which is because $\mathbf{Q}$ is defined in Eq.2.8 as a summation of $|\mathcal{S}|$ rank-one matrices. Clearly, the maximum of $|\mathcal{S}|$ is the maximum possible number of pairs i.e. $\frac{n\times(n-1)}{2}$; however, the size of $\mathbf{Q}$ is $\frac{d\times(d+1)}{2}$.

When $d \geq n$ matrix $\mathbf{Q}$ is rank-deficient. To be certain that $\mathbf{Q}$ is rank-deficient, one can project the points into a high-dimensional feature space by a possibly nonlinear mapping $\phi(\cdot)$. We employ the well-known kernel trick [53], using a kernel function $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ that computes the inner products between the high-dimensional feature vectors without explicitly constructing them.

### 2.1.3 Kernelizing the Proposed Method

Now, we show how to extend EAT to nonlinear mappings of data. Conceptually, the points are mapped into a high-dimensional feature space $\mathbb{F}$ by some nonlinear mapping $\phi(\cdot)$, and then the desired transformation is computed in that space. This can be done implicitly through the use of kernels.

Based on the representor theorem [54], the columns of the linear transformation $\mathbf{W}$ can always be re-expressed as linear combinations of the data points in the feature space, i.e. $\mathbf{W} = \mathbf{X}\boldsymbol{\Omega}$. Therefore, we can rewrite each pairwise squared distance as:

[3] The primal form of the SDP defined in Eq.2.16 is: $\min_{\mathbf{A}\succeq 0,\ \mathbf{Q}\mathbf{vech}(\mathbf{A})=\mathbf{p}} -\mathbf{vech}(\mathbf{A})^\top \mathbf{s}$.

$$\begin{aligned}\|\mathbf{y}_i - \mathbf{y}_j\|^2 &= (\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{W}\mathbf{W}^\top (\mathbf{x}_i - \mathbf{x}_j) \\ &= (\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{X}\boldsymbol{\Omega}\boldsymbol{\Omega}^\top \mathbf{X}^\top (\mathbf{x}_i - \mathbf{x}_j) \\ &= (\mathbf{x}_i^\top \mathbf{X} - \mathbf{x}_j^\top \mathbf{X})\boldsymbol{\Omega}\boldsymbol{\Omega}^\top (\mathbf{X}^\top \mathbf{x}_i - \mathbf{X}^T \mathbf{x}_j) \\ &= (\mathbf{X}^\top \mathbf{x}_i - \mathbf{X}^\top \mathbf{x}_j)^\top \mathbf{A}_\phi (\mathbf{X}^\top \mathbf{x}_i - \mathbf{X}^\top \mathbf{x}_j),\end{aligned} \tag{2.17}$$

where $\mathbf{A}_\phi = \boldsymbol{\Omega}\boldsymbol{\Omega}^\top$ is an $n \times n$ PSD matrix. We have now expressed the distances in terms of a matrix to be estimated (i.e. $\mathbf{A}_\phi$) and the inner products between the data points which can be computed via a kernel function, i.e. $\mathbf{x}_i^\top \mathbf{x}_j = \kappa(\mathbf{x}_i, \mathbf{x}_j)$.

$$\begin{aligned}\|\mathbf{y}_i - \mathbf{y}_j\|^2 &= \left(\kappa(\mathbf{X}, \mathbf{x}_i) - \kappa(\mathbf{X}, \mathbf{x}_j)\right)^\top \mathbf{A}_\phi \left(\kappa(\mathbf{X}, \mathbf{x}_i) - \kappa(\mathbf{X}, \mathbf{x}_j)\right) \\ &= (\mathbf{k}_i - \mathbf{k}_j)^\top \mathbf{A}_\phi (\mathbf{k}_i - \mathbf{k}_j),\end{aligned} \tag{2.18}$$

where $\mathbf{k}_i = \kappa(\mathbf{X}, \mathbf{x}_i) = \mathbf{X}^\top \mathbf{x}_i$ is the $i_{th}$ column of the kernel matrix $\mathbf{K} = \mathbf{X}^\top \mathbf{X}$. The optimization of $\mathbf{A}_\phi$ then proceeds just as in the non-kernelized version presented earlier, by substituting $\mathbf{X}$ and $\mathbf{W}$ with $\mathbf{K}$ and $\boldsymbol{\Omega}$ respectively. That is, we now have $\boldsymbol{\delta}_{ij} = \frac{\mathbf{k}_i - \mathbf{k}_j}{\tau_{ij}}$, and so all of the other variables are computed based on $\boldsymbol{\delta}_{ij}$.

### 2.1.4 The Algorithm

The learning procedure of Embedding by Affine Transformations (EAT) is summarized in Alg.1. Following it, Alg.2 explains how out-of-samples are mapped into the embedding space. In these algorithms, we suppose that all input data points are stacked into the columns of a $d \times n$ matrix $\mathbf{X}$. Likewise, all projected data points form the columns of matrix $\mathbf{Y}$, and the $r \times n$ matrix $\mathbf{Z}$ denotes the low-dimensional representation of the data.

In the last line of Alg.1, the columns of $\mathbf{P}$ are the eigenvectors of $\mathbf{Y}\mathbf{Y}^\top$ corresponding to the top $r$ eigenvalues which are calculated by PCA. Note that to apply PCA, $\mathbf{Y}$ must be centered. It is easy to see that if $\mathbf{X}$ is centered (or equivalently $\mathbf{K}$ is double centered), $\mathbf{Y}$ will be centered as well. This is the reason we center the input data at the beginning. In addition, in the kernelized version, the kernel matrix $\mathbf{K}$ should be centered, which is done by double centering $\mathbf{H}_n\mathbf{K}\mathbf{H}_n$, where $\mathbf{H}_n = \mathbf{I}_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top$.

After the learning phase of EAT, we have the desired transformation $\mathbf{W}$ for unfolding the latent structure of the data. We also have $\mathbf{P}$ from PCA, which is used to reduce the

**Algorithm 1** EAT - Learning

**Input:** $\mathbf{X}$, and $r$

1: Center the input: $\mathbf{X} \leftarrow \mathbf{X} - \frac{1}{n}\mathbf{X}\mathbf{1}_n\mathbf{1}_n^\top$

2: Compute the pairwise distances $\forall 1 \leq i, j \leq n : \tau_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|$

3: Based on $\{\tau_{ij}\}$, create the neighborhood graph and form the sets $\mathcal{S}$ and $\mathcal{O}$

4: Choose a kernel function $\kappa(\cdot, \cdot)$ and compute the kernel matrix $\mathbf{K}$

5: Apply double centering to the kernel: $\mathbf{K} \leftarrow \mathbf{H}_n\mathbf{K}\mathbf{H}_n$

6: Calculate the matrix $\mathbf{Q}$, and the vectors $\mathbf{p}$ and $\mathbf{s}$, from Eq.2.8 and Eq.2.14

7: Compute $\mathbf{U}$ and $\mathbf{\Lambda}$ by performing EVD on $\mathbf{Q}$ such that $\mathbf{Q} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$

8: Let $\boldsymbol{\alpha} = \mathbf{\Lambda}^{-1}\mathbf{U}^\top\mathbf{p}$

9: Solve the SDP problem $\max_{\mathbf{A} \succeq 0} \boldsymbol{\beta}^\top\overline{\mathbf{U}}^\top\mathbf{s}$, where $\mathbf{vech}(\mathbf{A}) = \mathbf{U}\boldsymbol{\alpha} + \overline{\mathbf{U}}\boldsymbol{\beta}$

10: Decompose $\mathbf{A} = \mathbf{W}\mathbf{W}^\top$ (or in the kernelized version $\mathbf{A}_\phi = \mathbf{\Omega}\mathbf{\Omega}^\top$)

11: Compute the coordinates on the unfolded manifold $\mathbf{Y} = \mathbf{W}^\top\mathbf{X} = \mathbf{\Omega}^\top\mathbf{K}$

12: Apply PCA to $\mathbf{Y}$ and obtain the low-dimensional representation $\mathbf{Z} = \mathbf{P}^\top\mathbf{Y}$

13: Return the solution $\mathbf{Z}$, and the transformation matrices $\mathbf{W}$ (or $\mathbf{\Omega}$) and $\mathbf{P}$

**Algorithm 2** EAT - Embedding

**Input:** $\mathbf{x}$, $\mathbf{W}$ (or $\mathbf{\Omega}$), and $\mathbf{P}$

1: Compute $\mathbf{k}_\mathbf{x} = \kappa(\mathbf{X}, \mathbf{x})$ (Only in the kernelized version)

2: Apply the centering to the new point: $\mathbf{x} \leftarrow \mathbf{x} - \mathbf{X}\mathbf{1}_n$ or $\mathbf{k}_\mathbf{x} = \mathbf{H}_n\left(\mathbf{k}_x - \frac{1}{n}\mathbf{K}\mathbf{1}_n\right)$

3: Let $\mathbf{y} = \mathbf{W}^\top\mathbf{x} = \mathbf{\Omega}^\top\mathbf{K}_\mathbf{x}$

4: Return $\mathbf{z} = \mathbf{P}^\top\mathbf{y}$

dimensionality of the unfolded data. As a result, we can embed any new point $\mathbf{x}$ by using Alg.2. It is important to note that the new points should be centered as well. In addition, this centering must be the same as the centering transformation used in the learning phase, i.e. $\mathbf{x} \leftarrow \mathbf{x} - \mathbf{X}\mathbf{1}_n$ or $\mathbf{k}_\mathbf{x} = \mathbf{H}_n \left(\mathbf{k}_x - \frac{1}{n}\mathbf{K}\mathbf{1}_n\right)$ in the kernelized version.

## 2.2 Experimental Results

In order to evaluate the performance of the proposed method, we conducted several experiments on synthetic and real datasets. To emphasize the difference between the transformation computed by EAT and the one that PCA provides, we designed a simple experiment on a synthetic dataset. In this experiment, we considered a three-dimensional V-shaped dataset illustrated in Fig.2.2 (top-left). We uniformly sampled 1000 data points from this manifold, and divided it into two subsets: an input dataset consisting of 28 well-sampled[4] points, and a test set of 972 points. EAT was applied to the input dataset, and then the learned transformation was used to project the test set. The result is depicted in Fig.2.2 (top-right). This image illustrates $\mathbf{Y} = \mathbf{W}^\top\mathbf{X}$, which is the result of EAT in three dimensions. It shows that the third dimension carries no information. Thus, the input data was unfolded before applying PCA to reduce the dimensionality to two.

In addition, the bottom row of Fig.2.2 shows the results of PCA and KPCA, when applied to the entire dataset. PCA computes a global distance preserving transformation, and captures the directions of the maximum variation in data. Clearly, in this example, the direction with the maximum variation is not the one that unfolds the dataset. This is the key difference between the functionality of PCA and EAT. Kernel PCA does not provide a satisfactory embedding either. Fig.2.2 shows the result that is generated by an RBF kernel; we experimented on KPCA with a variety of popular kernels, but none were able to reveal a faithful embedding of the V-shaped dataset.

Unlike kernel PCA, EAT does not expect the kernel to reveal the underlying structure of data. When the dimensionality of data is higher than its quantity, a linear transformation can span the entire space. This means that we can always find the transformation $\mathbf{W}$ to flatten the underlying manifold. When the original dimensionality of data is high ($d > n$, e.g., for images), EAT does not need a kernel in principal; however, using a linear kernel reduces the computational complexity of the method[5]. In all of the following experiments,

[4]The original data is down-sampled by clustering to obtain 28 points.

[5] In the kernelized version, $\mathbf{W}$ is $n \times n$ but in the original version it is $d \times d$. Thus, computing $\mathbf{W}$ in the kernelized form is less complex when $d > n$.

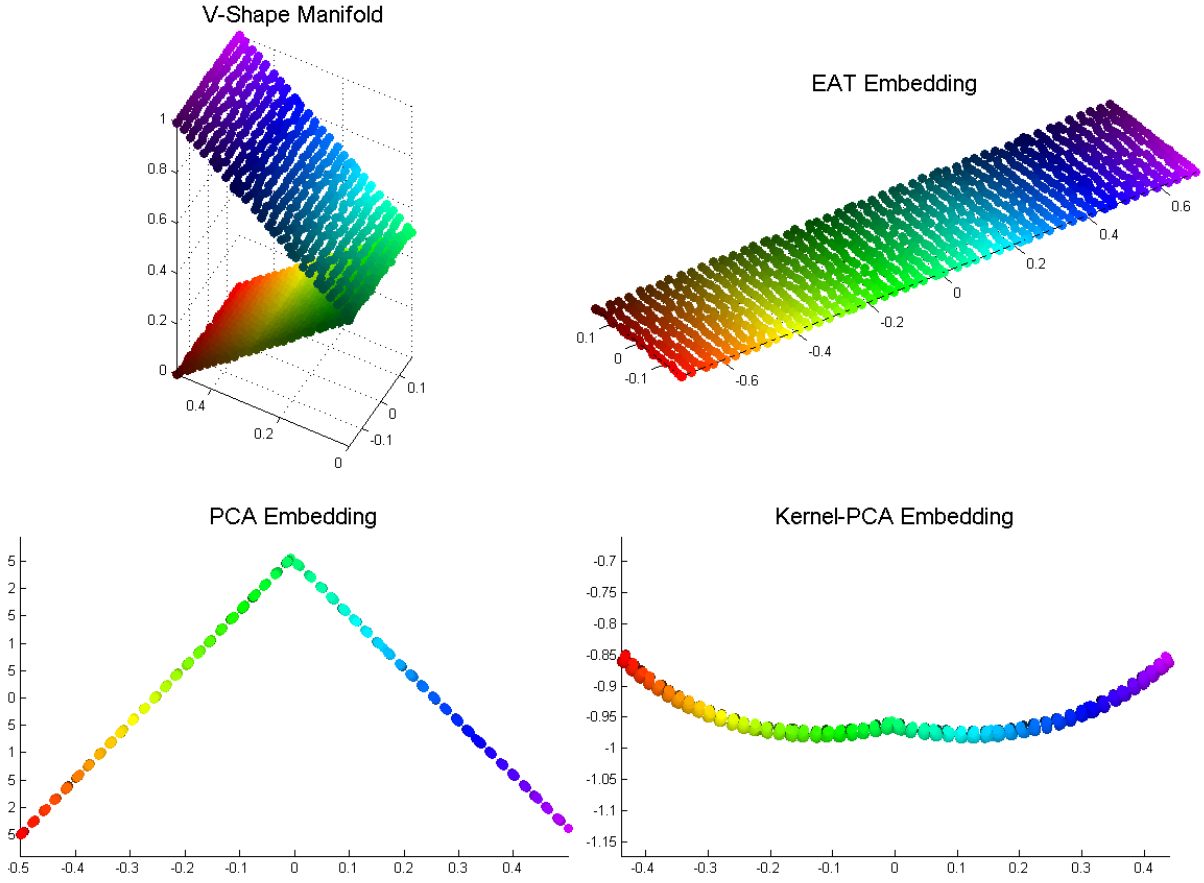


Figure 2.2: A V-shaped manifold and the results of EAT, PCA, and kernel PCA.

we used a linear kernel when the original dimensionality of data is high (e.g., for images), and RBF in all other cases. In general EAT is not sensitive to the type of kernel. We will discuss the effect of kernel type and its parameters later in this section.

The next experiment is on a Swiss-roll manifold, depicted in Fig.2.3 (bottom-left). Although Swiss-roll is a three-dimensional dataset, it tends to be one of the most challenging datasets due to its complex global structure. We sampled 50 points for our training set (in the same way that we did for the first experiment) and 950 points as an out-of-sample test set. The results of Maximum Variance Unfolding (MVU), Isomap, and EAT[6] are presented

[6] In general, Kernel PCA fails to unfold Swiss-roll. LLE generally produces a good embedding, but not on small datasets (e.g., our training set). For this reason we do not demonstrate their results.

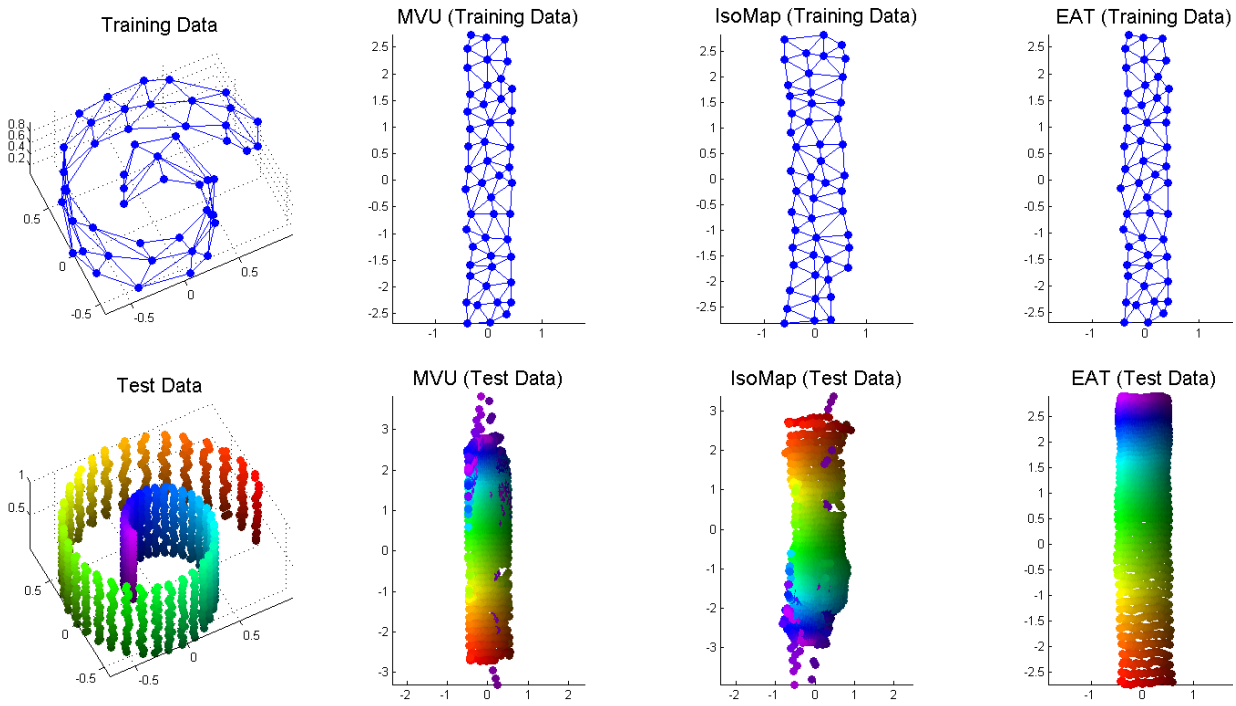


Figure 2.3: A Swiss-roll manifold, and the results of different dimensionality reduction methods: MVU, Isomap, and EAT. The top row demonstrates the results on the training set, and the bottom row shows the results of the out-of-sample test set.

in the first row of Fig.2.3. The second row shows the projection of the out-of-sample points into a two-dimensional embedding space. EAT computes a transformation that maps the new data points into the low-dimensional space. MVU and Isomap, however, do not provide any direct way to handle out-of-sample examples. A common approach for resolving this problem is to learn a non-parametric model between the low-dimensional embedding space and the high-dimensional input space.

In this approach, a high-dimensional test data point $\mathbf{x}$ is mapped to the low-dimensional space in three steps: (i) the $k$ nearest neighbors of $\mathbf{x}$ among the training inputs (in the original space) are identified; (ii) the linear weights that best reconstruct $\mathbf{x}$ from its neighbors, subject to a sum-to-one constraint, are computed; (iii) the low-dimensional representation of $\mathbf{x}$ is computed as the weighted sum (with weights computed in the previous step) of the embedded points corresponding to those $k$ neighbors of $\mathbf{x}$ in the original space. In all of our experiments, the out-of-sample embedding has been conducted using this non-parametric

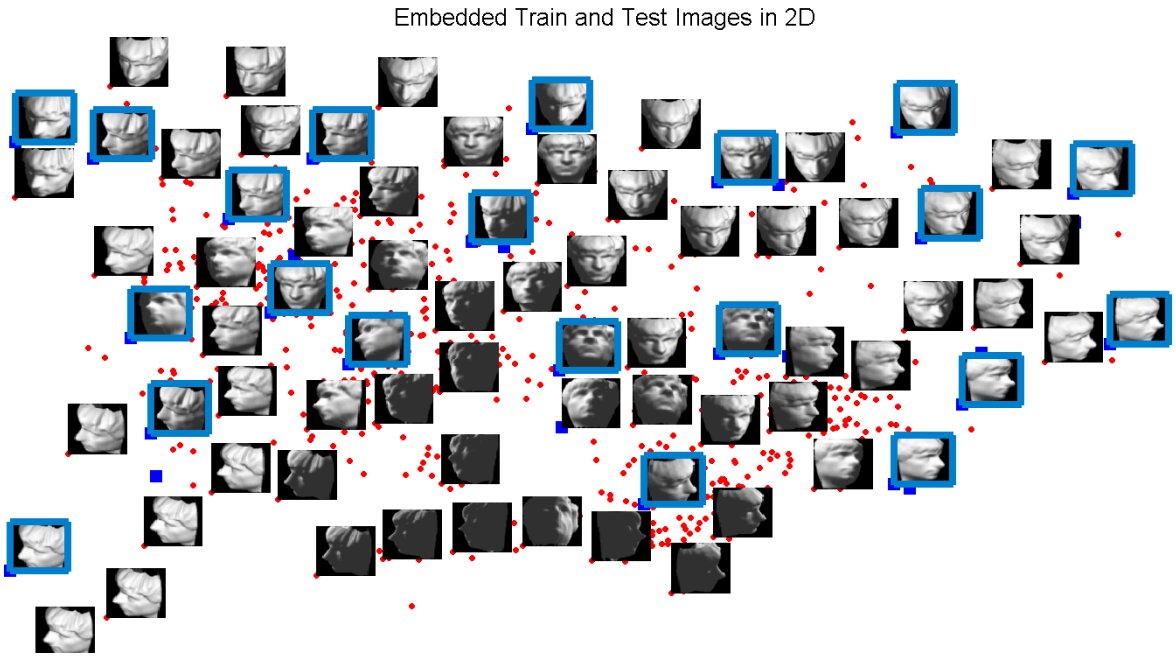


Figure 2.4: The visualization of synthetic face images in two dimensions by EAT, using a linear kernel.

model except for EAT, PCA, and Kernel PCA, which provide parametric models. It is clear that the out-of-sample estimates of MVU and Isomap are not faithful to the Swiss-roll shape, especially along its border.

Now we illustrate the performance of the proposed method on real datasets. Fig.2.4 shows the result of EAT when applied to a dataset of face images. This dataset consists of 698 images, from which we randomly selected 35 images for the learning phase of EAT and used the rest as test data. Training points are indicated with a solid blue border.

The images in this experiment have three degrees of freedom: pan, tilt, and brightness. In Fig.2.4, the horizontal and vertical axes appear to represent the pan and tilt, respectively. Interestingly, while there are no low-intensity images among the training samples, darker out-of-sample points appear to have been organized together in the embedding. These darker images still maintain the correct trends in the variation of pan and tilt across the embedding. In this example, EAT was used with a linear kernel.

In another experiment, we conducted dimensionality reduction on a subset of the Olivetti images dataset [51]. Face images of three different persons were used as the training set, and images of a fourth person were tested as the out-of-sample examples. The results

of PCA, KPCA, EAT, LLE, Isomap and MVU are illustrated in Fig.2.5. Different persons in the training data are indicated by red squares, green triangles and purple diamonds. PCA and Kernel PCA did not provide interpretable results even for the training set. The other methods, however, separated the different people along different chains. Each chain shows a smooth change between the side view and the frontal view of an individual.

The key difference between the algorithms is the way they embedded the images of the new person (represented by blue circles). MVU, LLE, Isomap, PCA, and Kernel PCA all superimposed these images onto the images of the most similar individual in the training set, and by this, they clearly lost a part of the information. This is because we used a non-parametric model for embedding the out-of-samples examples for these methods. EAT, however, embedded the images of the new person as a separate cluster (chain), and maintained a smooth gradient between the frontal and side views.

Finally, we attempted to unfold a globe map, shown in Fig.2.6 (top-left), into a faithful two-dimensional representation. Since a complete globe is a closed surface and thus cannot be unfolded, our experiment was on a half-globe. A regular mesh was drawn over the half-globe, shown in Fig.2.6 (top-right), and 181 samples were taken for learning. EAT was used to unfold the sampled mesh and find its transformation; the result is depicted in Fig.2.6 (bottom-right).

Note that it is not possible to unfold a globe into a two-dimensional space while preserving the original local distances; in fact, the transformation with the minimum preservation error is the identity function. So rather than preserving the local distances, we defined Euclidean distances based on the latitude and longitude of the training points along the surface of the globe; then the two-dimensional embedding became feasible. This is an interesting aspect of EAT: it does not need to operate on the original distances of the data, but can instead be supplied with arbitrary distance values (as long as they are compliant with the desired dimensionality reduction of the data).

We used a test set consisting of 30,000 points as specified by their three-dimensional coordinates with respect to the center of the globe. For this experiment we used an RBF kernel with $\sigma = 0.3$. Applying the output transformation of EAT resulted in a two-dimensional embedding shown in Fig.2.6 (bottom-left); color was used to denote elevation in these images. Note that the pattern of the globe has not changed during the embedding process, which demonstrates that the representation of EAT is faithful. However, the two-dimensional embedding of the test points is distorted at the sides, which is due to the lack of information from the training samples in these areas.

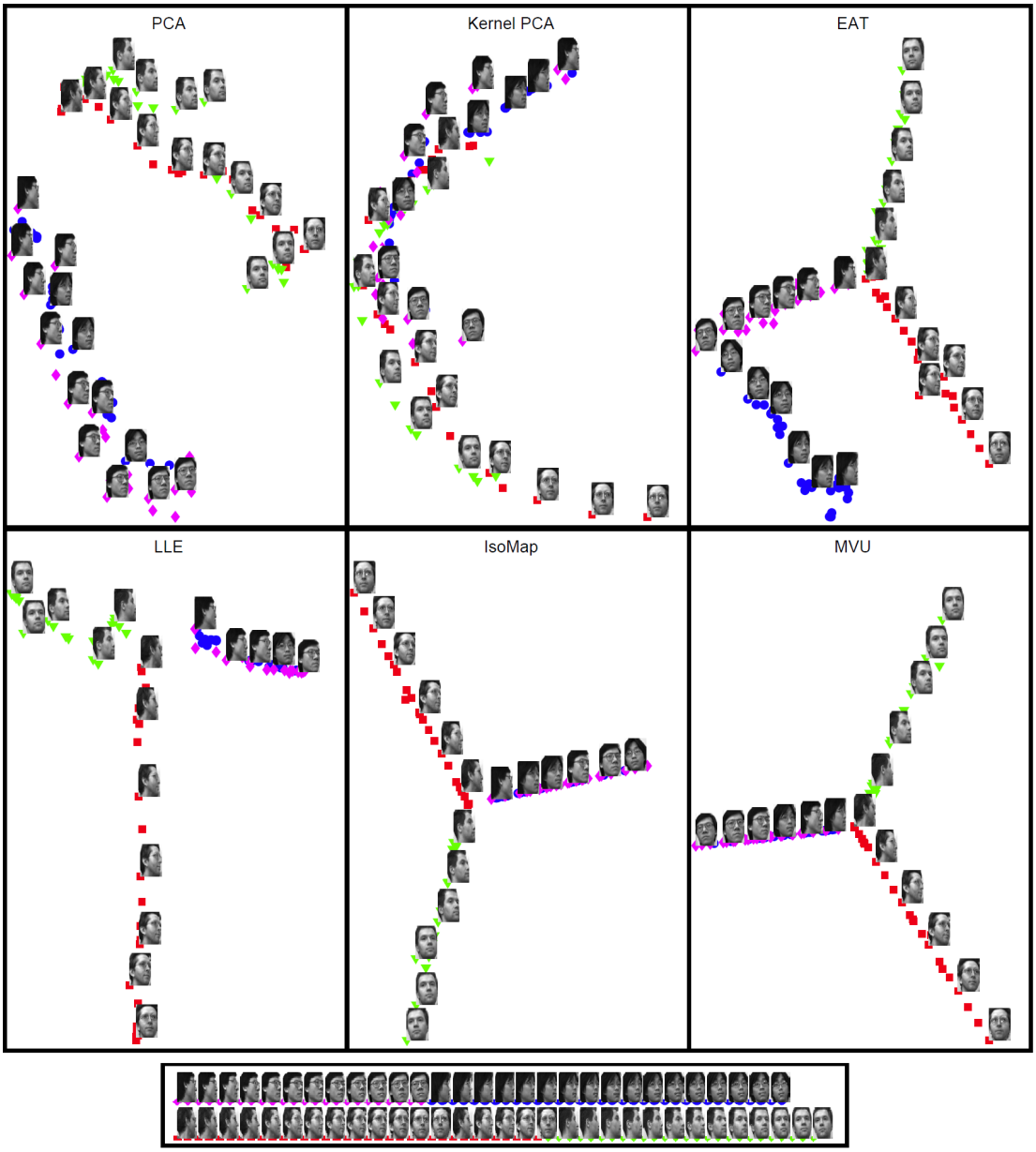


Figure 2.5: The results of PCA, KPCA, EAT, LLE, Isomap and MVU on the dataset of face images. Each color represents the pictures of one of four individuals. The blue circles show the test data (pictures of the fourth individual). The bottom panel shows half of the pictures used in the experiment.

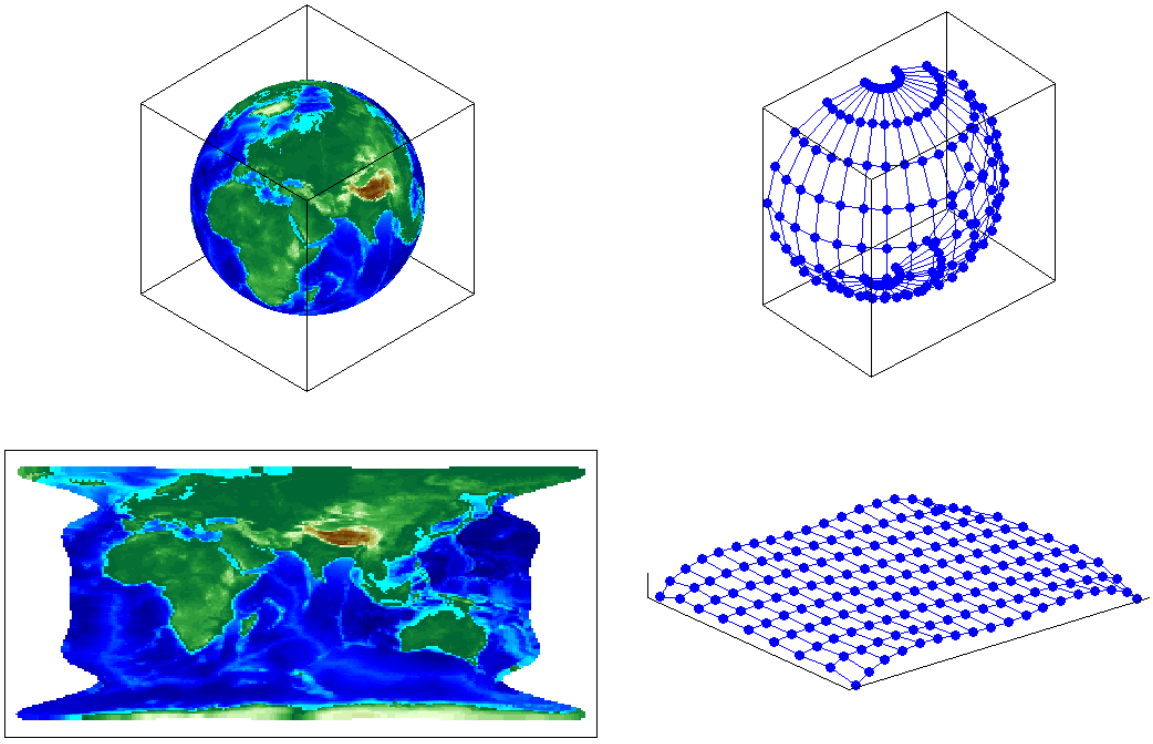

Figure 2.6: Unfolding a half-globe into a two-dimensional map by EAT. A half-sphere mesh is used for training. Color is used to denote elevation. The out-of-sample test set comprises 30,000 points from the surface of Earth.

### 2.2.1 The Effect of Type and Parameters of Kernels

Intuitively speaking, the number of basis vectors corresponding to a particular kernel matrix is equal to the rank of that matrix; in a full-rank kernel matrix (i.e. $\mathbf{Rank}(\mathbf{K}) = n$), the number of basis vectors is equal to the number of data points and thus, a linear transformation from the feature space $\mathbb{F}$ can span the entire $n$-dimensional embedding space. That is, it is always possible to find a transformation $\mathbf{W}$ to unfold the underlying manifold of the data, as far as the training data points are concerned.

For an extreme example one may consider the identity matrix as a kernel. Since it is a full-rank kernel, it guarantees that we can find a perfect map for any training dataset; but it will fail to map out-of-sample points correctly, because it cannot measure the similarity between the out-of-sample points and the training examples. In other words, using a full-

rank kernel is a sufficient condition in order to faithfully embed the training points. But if the correlation between the kernel and the data is weak (an extreme case is using the identity matrix as a kernel), EAT will not perform well for the out-of-sample points.

In this experiment, the effect of different kernel functions on embedding by EAT is studied. We used a family of polynomial kernel functions[7] with different degrees, and also a family of RBF kernels[8] with different $\sigma$ values. The original manifold was a three-dimensional Swiss-roll, shown in Fig.2.3 (bottom-left), from which we uniformly sampled 102 training points, and 1,000 out-of-sample points.

We define a variable $\rho = \frac{\mathbf{Rank(K)}}{n}$; clearly for full-rank matrices $\rho = 1$ and $\rho < 1$ shows a rank-deficiency. For each kernel function applied on the training set, we have calculated this variable. The effects of using polynomials are illustrated in Fig.2.7. We used polynomial functions of degrees one to ten in separate columns. Each column has two plots; the top one shows the embedding of the training set, and the related result of out-of-sample mapping is in the bottom panel. We indicate the degree of the polynomial at the top, and $\rho$ in the middle of each column.

All of the polynomials with degree greater than one have provided the same proper embeddings in both of the learning phase and the out-of-sample mapping. The degree-one polynomial, which is in fact a linear kernel, was not able to help EAT in this example; the manifold was not unfolded in the first column, and so the out-of-sample embedding also failed. This experiment interestingly shows that even with rank-deficient kernels (i.e. small values of $\rho$, for example here from 0.1), unfolding is possible.

The effect of using different RBF kernels is illustrated in Fig.2.8. We used the RBF function with different $\sigma$ values. Since the magnitude of $\sigma$ should be proportional to the scale of the manifold, it is multiplied by the mean of the local distances, to be normalized. We chose ten different $\sigma$ values from a wide range $\frac{1}{3}$ to 50. For each $\sigma$ value, the result is shown in one column.

When $\sigma$ is large (more than 10), $\rho$ is small and therefore, EAT has not been able to find the desired transformation in the feature space that would unfold the manifold of the training data in this experiment (the two rightmost columns in Fig.2.8). On the other hand, when $\sigma$ is very small (less than one), although the kernels are full-rank ($\rho = 1$) and consequently, the embedding of the training set is faithful, EAT has failed to embed the out-of-samples correctly (the leftmost columns in Fig.2.8). In fact, small values of $\sigma$ form kernel matrices that are very close to the identity matrix, and thus over-fitting occurs.

[7] i.e., $\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^\top \mathbf{x}_j + 1)^p$.

[8] i.e., $\kappa(\mathbf{x}_i, \mathbf{x}_j) = e^{\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}}$.

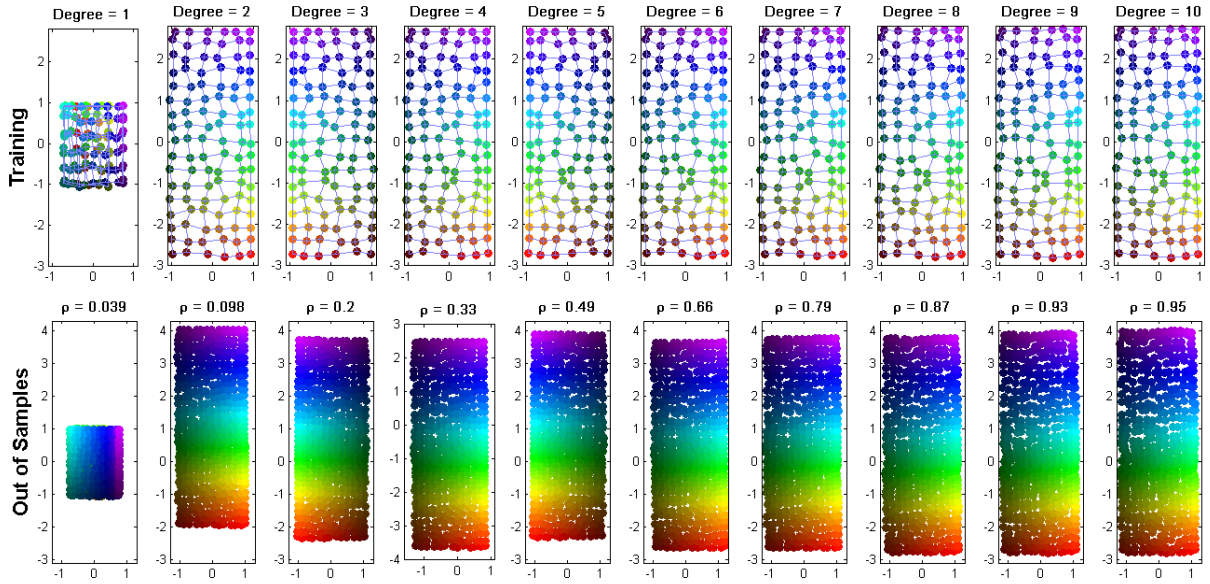


Figure 2.7: The effect of using polynomial kernels of different degrees in EAT for embedding a three-dimensional Swiss-roll manifold into a two-dimensional space, in the training phase (top) and in out-of-sample mapping (bottom).

Experiments with a wide variety of other kernels on different datasets shows similar results. Based on our experiments and this study, we suggest that an RBF kernel can be used for any dataset. The value of $\sigma$ should be selected close to the mean of local distances. In this way, the kernel matrix is close to full-rank ($\rho \approx 1$), which guarantees a proper embedding for the training set. On the other hand, the resulting kernel is still able to measure the similarity between nonidentical data points to avoid the over-fitting problem. When the dimensionality of the original data is more than or equal to the number of data points, there is no need for a kernel, but one may use a simple linear kernel (or any other desired kernel) to reduce the computational complexity.

## 2.3 Conclusion and Discussion

We presented Embedding by Affine Transformations (EAT), a novel dimensionality reduction method, which, unlike other prominent methods, can easily embed out-of-sample examples. This method learns a mapping from a high-dimensional feature space to a low-dimensional embedding space. EAT is performed in two steps; first, the input data

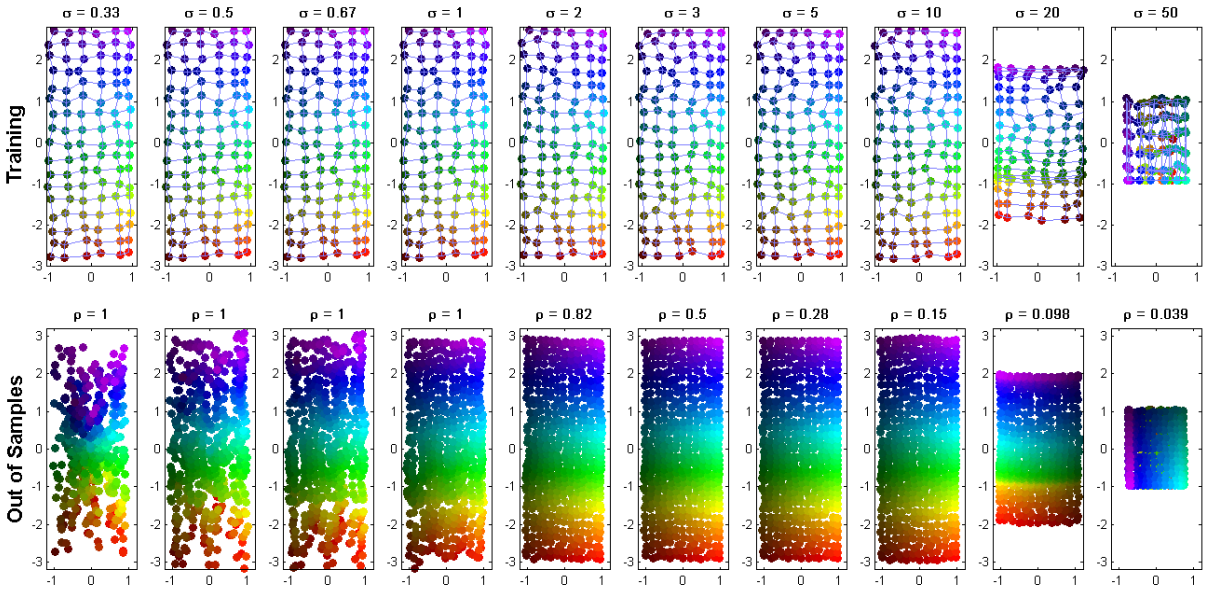


Figure 2.8: The effect of using RBF kernels with different $\sigma$ values in EAT for embedding a three-dimensional Swiss-roll manifold into a two-dimensional space, in the training phase (top) and out-of-sample mapping (bottom).

is projected into the high-dimensional feature space, and then, a linear transformation is learned that maps the data points from the feature space into the low-dimensional embedding space. By unfolding the underlying manifold of the input data in the embedding space, EAT decreases the dimensionality of the data. The search for this transformation is cast as an instance of semidefinite programming (SDP), which is convex and always converges to the global optimum. Our experimental results on real and synthetic datasets demonstrated that EAT produces a robust and faithful embedding even for very small datasets. They also showed that EAT is successful in projecting out-of-sample examples.

Solving the SDP problem is computationally intensive, which can make it inefficient to learn EAT on large datasets. Thus, one approach for handling large datasets with EAT would involve down-sampling the input data, by selecting a small subset as the training input and then, embedding the rest of the data as test examples.

To form the SDP of Eq.2.16, first $\mathbf{Q}$ is decomposed and its null-space is computed. Due to the large size of $\mathbf{Q}$, this step is the most time-consuming part of EAT. However, by comparing Eq.2.15 and Eq.2.16, it is clear that using the null-space of $\mathbf{Q}$ to represent the SDP in the dual form, significantly reduces the number of unknown variables.

As discussed, EAT first finds a family of projections that produce the minimum possible error in distance preservation, and then, in the SDP problem, it chooses one of them that stretches non-neighboring distances the most. It is important to note that it was possible to use the neighboring distances as constraints, however, in case no feasible solution for a given set of $\{\tau_{ij}|(i,j) \in \mathcal{S}\}$ exists, the SDP could not be solved. One might resolve this issue by using slack variables to relax the constraints; however, the resulting SDP problem would depend on penalty terms assigned to the slack variables, and therefore, it would need parameter tuning. In the current setting, we have considered this problem, and if no feasible solution exists, we use only solutions which result in a minimum error. Therefore, without any parameter tuning, EAT preserves the local distances as much as possible.

Another feature of EAT is that it treats the distances between the data points in three different ways. One can preserve a subset of the distances (set $\mathcal{S}$), stretch another subset (set $\mathcal{O}$) and leave the third set, whose pairs are unspecified in $\mathcal{S}$ and $\mathcal{O}$. This is in contrast with methods like MVU that preserve local distances but stretch any non-local pairs. This property means that EAT could be useful for semi-supervised tasks where only partial information about similarity and dissimilarity of the input points is known.

# Chapter 3

# TesseraMap: Unfolding Manifolds by Tessellation of Linear Subspaces

In this chapter, we propose a novel method for dimensionality reduction, which we call *TesseraMap*. The proposed method can be seen as an extension to EAT which scales well with the number of data points. Similar to EAT, this nonlinear technique employs semidefinite programming to compute a transformation between the input dataset and its low-dimensional representation, which can be used for mapping out-of-samples.

As discussed in Chapter 1, Ham *et al.* showed that many popular nonlinear dimensionality reduction algorithms can be formulated as KPCA [25]. This implies that the way a method treats high-dimensional data is encoded in its kernel and, in fact, the differences between the embeddings generated by these methods are caused by their data-driven kernels. This perspective, in addition to considering that all kernel matrices are positive semidefinite, is perhaps what motivated the development of methods in which an appropriate kernel for embedding is estimated by solving an instance of semidefinite programming (SDP), for example MVU [69].

Semidefinite programming has recently been the center of attention in the literature of nonlinear dimensionality reduction. However, due to the computational complexity of the SDP solvers, the effectiveness of the SDP-based algorithms is still limited, therefore, only small-size problems can be solved using the current methods. A few relatively scalable versions, for example, Landmark MVU [70], Fast MVU [71], and Colored MVU [57], have been proposed for the current SDP-based methods; however, their quality of embedding on large datasets is not satisfactory. There are two reasons why these methods are not able to provide faithful embeddings:

- To reduce the size of the SDP problem, they factorize the kernel as $\mathbf{K} = \mathbf{U}\mathbf{S}\mathbf{U}^\top$, where $\mathbf{U}$ is known, and then they approximate the desired kernel $\mathbf{K}$ with a small positive semidefinite matrix $\mathbf{S}$. Since this approximation is usually done by heuristic methods, the loss of information is inevitable and therefore, the final embedding can be highly corrupted.
- To compensate for the information loss of approximation, they employ local search methods, such as gradient descent, as a post-processing step. Unfortunately, this approach does not guarantee the optimality of the final result. In fact, there is a trade-off between the time spent on the post-processing, and the required quality for the final embedding.

Moreover, since the kernels of these methods are data-driven, to add a new point the kernel matrix must be completely recalculated. This is why the existing SDP-based methods are unable to provide a straightforward extension for embedding out-of-sample points. Unfortunately, this is an undesirable shortcoming when one employs these methods in supervised machine learning tasks, which are trained based on training sets and then might be used on new test sets.

In Chapter 2, we proposed Embedding by Affine Transformations (EAT), which is solved based on a semidefinite program. EAT provides a mapping to unfold the underlying manifold of the input data, which can be used for out-of-samples as well. Unfortunately, EAT suffers from lack of scalability. Now, we show that in EAT, the search space of the SDP problem can be restricted to a small face of the semidefinite cone. The formulation of this method significantly reduces the size of the SDP problem. For this reason, TesseraMap is fast and scalable, and it allows us to solve large problems in dimensionality reduction. Crucially, the proposed method is not an approximation and thus, it provides an exact solution without the need for post-processing by local gradient descent. This is in contrast with the other large-scale SDP-based methods. In addition, inheriting from EAT, this algorithm is able to provide a mapping that allows out-of-sample points to be sensibly mapped into the embedding space.

The rest of this chapter is organized as follows. There are two main sections, followed by a short discussion at the end. In Section 3.1, first we slightly reformulate EAT from the previous chapter, and simplify it to be employed as a kernel method. Then, we continue by intuitively explaining the ideas behind TesseraMap. We mathematically describe how the computational complexity of the SDP problem in EAT can be reduced, and develop the TesseraMap algorithm. In Section 3.2, we illustrate some experimental results on TesseraMap and comparisons conducted on the other popular methods. Finally, in Section 3.3 we discuss different aspects of TesseraMap.

## 3.1 Learning a Mapping for Unfolding

In TesseraMap, we use the kernelized version of EAT; we seek a linear mapping between a high-dimensional feature space and the embedding space, whereby the underlying manifold of the input data is unfolded, and thus the dimensionality of the data is reduced. The desired transformation should be faithful to the local structure of the manifold. That is, it preserves the distances between the neighboring points (i.e. the pairs indicated by set $\mathcal{S}$). Moreover, for unfolding, the non-neighboring points (the pairs of set $\mathcal{O}$) should be pulled apart, as already discussed in EAT. Suppose a $d \times n$ matrix $\mathbf{X}$ as the input dataset, which will be embedded in $\mathbf{Y}$. We explained that the transformation can be formulated as:

$$\mathbf{Y} = \mathbf{W}^\top \phi(\mathbf{X}) = \mathbf{\Omega}^\top \phi(\mathbf{X})^\top \phi(\mathbf{X}) = \mathbf{\Omega}^\top \mathbf{K}, \tag{3.1}$$

where $\mathbf{K}$ is a kernel matrix of the input, and the goal is to find the linear transformation $\mathbf{\Omega}$. From Eq.3.1, it is clear that if matrix $\mathbf{K}$ is full-rank, obtaining any result for $\mathbf{Y}$ is possible. If the mapping $\phi : \mathbb{R}^d \mapsto \mathbb{F}$ produces an adequate number of independent features for the data points, the kernel matrix $\mathbf{K} = \phi(\mathbf{X})^\top \phi(\mathbf{X})$ becomes full-rank. This can always be achieved, for example, by choosing a kernel function whose $\phi(\cdot)$ maps the data points to an infinite-dimensional feature space, e.g., exponential kernel functions.

The distances between the neighboring points should be preserved in $\mathbf{Y}$; more formally, they should satisfy $\forall (i,j) \in \mathcal{S} : \ \|\mathbf{y}_i - \mathbf{y}_j\| = \tau_{ij}$, where $\tau_{ij}$ is given (simply can be $\|\mathbf{x}_i - \mathbf{x}_j\|$). The distance between two embedded points can be expanded as follows:

$$\begin{aligned} \|\mathbf{y}_i - \mathbf{y}_j\|^2 =& \|\mathbf{\Omega}^\top \kappa(\mathbf{X}, \mathbf{x}_i) - \mathbf{\Omega}^\top \kappa(\mathbf{X}, \mathbf{x}_j)\|^2 = (\mathbf{k}_i - \mathbf{k}_j)^\top \mathbf{\Omega}\mathbf{\Omega}^\top (\mathbf{k}_i - \mathbf{k}_j) \\ =& (\mathbf{k}_i - \mathbf{k}_j)^\top \mathbf{A} (\mathbf{k}_i - \mathbf{k}_j) = ((\mathbf{k}_i - \mathbf{k}_j) \otimes (\mathbf{k}_i - \mathbf{k}_j))^\top \mathbf{vec}(\mathbf{A}), \end{aligned} \tag{3.2}$$

where $\otimes$ is the Kronecker product operator, $\mathbf{k}_i = \kappa(\mathbf{X}, \mathbf{x}_i)$ is the $i^{\text{th}}$ column of the kernel matrix $\mathbf{K} = \kappa(\mathbf{X}, \mathbf{X})$, and $\mathbf{A} = \mathbf{\Omega}\mathbf{\Omega}^\top \in \mathcal{S}_+^n$ is an unknown positive semidefinite matrix. To preserve the distances in the embedding space, we define the following normalized error, and attempt to minimize it:

$$\sum_{(i,j)\in\mathcal{S}} \left( \frac{\|\mathbf{y}_i - \mathbf{y}_j\|^2}{\tau_{ij}^2} - 1 \right)^2 = \sum_{(i,j)\in\mathcal{S}} \left( \boldsymbol{\xi}_{ij}^\top \mathbf{vec}(\mathbf{A}) - 1 \right)^2 = \|\mathbf{\Xi}_{\mathcal{S}}^\top \mathbf{vec}(\mathbf{A}) - \mathbf{1}_{|\mathcal{S}|}\|^2, \tag{3.3}$$

where, similar to EAT, $\boldsymbol{\xi}_{ij} = \frac{1}{\tau_{ij}^2}((\mathbf{k}_i - \mathbf{k}_j) \otimes (\mathbf{k}_i - \mathbf{k}_j))$. For all $(i,j) \in \mathcal{S}$, the vectors $\boldsymbol{\xi}_{ij}$ form the columns of an $n^2 \times |\mathcal{S}|$ matrix $\boldsymbol{\Xi}_{\mathcal{S}}$. By taking the derivative of Eq.3.3 with respect to $\mathbf{vec}(\mathbf{A})$, and setting it to zero, the following constraint for $\mathbf{A}$ is obtained:

$$\boldsymbol{\Xi}_{\mathcal{S}}\boldsymbol{\Xi}_{\mathcal{S}}^{\top}\mathbf{vec}(\mathbf{A}) = \boldsymbol{\Xi}_{\mathcal{S}}\mathbf{1}_{|\mathcal{S}|}. \tag{3.4}$$

Following the notation of EAT[1], we can rewrite the constraint as $\mathbf{Q}\mathbf{vec}(\mathbf{A}) = \mathbf{p}$, where $\mathbf{Q} = \boldsymbol{\Xi}_{\mathcal{S}}\boldsymbol{\Xi}_{\mathcal{S}}^{\top}$ and $\mathbf{p} = \boldsymbol{\Xi}_{\mathcal{S}}\mathbf{1}_{|\mathcal{S}|}$. Although $\mathbf{Q}$ is $n^2 \times n^2$, which means we have $n^2$ constraints, its rank does not exceed $|\mathcal{S}|$. Considering that $|\mathcal{S}| \in \mathsf{O}(n)$, the number of the constraints can be reduced[2] to $\mathsf{O}(n)$. Next we define the following objective function which, when optimized, attempts to maximize the relative stretching of the distances between the non-neighboring points specified in set $\mathcal{O}$:

$$\sum_{(i,j)\in\mathcal{O}} \frac{1}{\tau_{ij}^2}\|\mathbf{y}_i - \mathbf{y}_j\|^2 = \sum_{(i,j)\in\mathcal{O}} \boldsymbol{\xi}_{ij}^{\top}\mathbf{vec}(\mathbf{A}) = \mathbf{s}^{\top}\mathbf{vec}(\mathbf{A}), \tag{3.5}$$

where $\mathbf{s} = \sum_{(i,j)\in\mathcal{O}} \boldsymbol{\xi}_{ij} = \boldsymbol{\Xi}_{\mathcal{O}}\mathbf{1}_{|\mathcal{O}|}$, in which the columns of the $n^2 \times |\mathcal{O}|$ matrix $\boldsymbol{\Xi}_{\mathcal{O}}$ are the vectors $\boldsymbol{\xi}_{ij}$, when $(i,j) \in \mathcal{O}$. The problem of finding the desired transformation can be cast as an instance of semidefinite programming as follows:

$$\max_{\mathbf{A}\succeq 0} \quad \mathbf{s}^{\top}\mathbf{vec}(\mathbf{A}) \quad \text{subject to} \quad \mathbf{Q}^{\top}\mathbf{vec}(\mathbf{A}) = \mathbf{p}. \tag{3.6}$$

To obtain the desired transformation $\boldsymbol{\Omega}$, the solution of the convex optimization problem Eq.3.6 should be decomposed as $\mathbf{A} = \boldsymbol{\Omega}\boldsymbol{\Omega}^{\top}$. In this way, the transformation $\boldsymbol{\Omega}$ is computed that when applied to the data, preserves the local distances and pulls the non-neighbor points as far apart as possible.

Unfortunately, the constraint matrix $\mathbf{Q}$ in the optimization problem Eq.3.6 is not sparse. Therefore, the computational complexity of the SDP problem is in $\mathsf{O}(n^4)$ [10]. This means that only limited size problems can be directly solved by EAT.

[1] In this chapter, for the sake of simplicity, we omit the use of $\mathbf{vech}(\cdot)$ operator. However, the argument is still valid for $\mathbf{vech}(\cdot)$, if the duplication matrix $\mathbf{D}_n$ is applied. Note that due to using $\mathbf{vec}(\cdot)$ instead of half-vectorization, the sizes of $\mathbf{Q}, \mathbf{b}$ and $\mathbf{s}$ have changed.

[2] Many of the constraints defined in Eq.3.4, are redundant. To reduce the number of the constraints to $|\mathcal{S}|$, one can simply multiply both sides by $\boldsymbol{\Xi}_{\mathcal{S}}^{\top}$, and solve the problem with the new constraint set $\boldsymbol{\Xi}_{\mathcal{S}}^{\top}\boldsymbol{\Xi}_{\mathcal{S}}\boldsymbol{\Xi}_{\mathcal{S}}^{\top}\mathbf{vec}(\mathbf{A}) = \boldsymbol{\Xi}_{\mathcal{S}}^{\top}\boldsymbol{\Xi}_{\mathcal{S}}\mathbf{1}_{|\mathcal{S}|}$.

To overcome this difficulty, we propose a novel SDP formulation by employing *semidefinite facial reduction* [11], which dramatically reduces the complexity of the semidefinite problem. In this formulation, we restrict the search space of the SDP problem to a small face of the semidefinite cone. This is done by decomposing the large unknown semidefinite variable $\mathbf{A}$ to $\mathbf{VBV}^\top$, where $\mathbf{V}$ is known and $\mathbf{B}$ is a small unknown positive semidefinite matrix. We will show that such a decomposition can be done on the Gram matrix of the points in their low-dimensional representation, crucially with no approximation.

### 3.1.1 The Proposed Method

TesseraMap is motivated by observing that for any given dataset, the appealing local structure of its underlying manifold lies beyond mere pairwise neighborhood relationships between the data points. In fact, in most cases, it is more efficient to preserve the structure of a reasonably large chunk of data as a whole, rather than dividing it into small (possibly pairwise) neighborhoods and preserving them, like other dimensionality reduction methods. Based on this observation, we first perform a clustering on the input dataset, so that it is conceptually reduced to a set of small clusters. In this way, one can imagine the underlying manifold of the data is partitioned into a set of regions. We call each of these regions a *tessera*[3]. The underlying manifold can then be considered as a mosaic, composed of many tesserae. Each tessera represents a local region of the manifold, and the global structure of the manifold can simply be described as the overall configuration, or tiling of the tesserae.

Secondly, we unfold the underlying manifold on a low-dimensional linear subspace, by reconstructing its mosaic as a tessellation of that linear subspace. This is done by considering a neighborhood graph for the tesserae and computing a transformation, whereby non-neighbor tesserae are pulled apart, while the distances between the neighboring tesserae are kept fixed. This yields a low-dimensional embedding of the underlying manifold whose local structure has been preserved.

Note that the constraints in Eq.3.6 preserve the local structure of the manifold by fixing the pairwise Euclidean distances between the neighboring points. However, the desired transformation treats the tesserae as rigid parts and only preserves the pairwise distances between two neighboring tesserae. This also dramatically reduces the computational complexity of the SDP problem.

Suppose that the input data points $\{\mathbf{x}_i\}_{i=1}^n$ are clustered into $c$ tesserae. The set $\mathcal{C}_\ell$ holds the indices of the $\ell^{\text{th}}$ tessera, and without loss of generality we can assume that

[3] A tessera (plural: tesserae) is an individual tile in a mosaic.

$\mathcal{C}_1 = \{1, 2, \ldots, n_1\}$, $\mathcal{C}_2 = \{n_1 + 1, \ldots, n_1 + n_2\}$, and so on, where $n_\ell = |\mathcal{C}_\ell|$. It follows that the input data matrix $\mathbf{X}$ can be represented as $\mathbf{X} = [\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_c]$ and consequently, its low-dimensional representation is $\mathbf{Y} = [\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_c]$, where $\mathbf{X}_\ell$ is the matrix whose columns represent the points of the $\ell^{\text{th}}$ tessera, and $\mathbf{Y}_\ell$ is an unknown matrix with the same number of columns, representing the coordinates of the points in the embedded tessera.

Let $\mathbf{D}_\ell$ be the $n_\ell \times n_\ell$ Euclidean Distance Matrix (EDM) of the points in $\mathbf{X}_\ell$. Also, we suppose that the points in each tessera are close to a low-dimensional affine subspace[4]; therefore, we can treat the tesserae as rigid parts and would like to preserve their structures. That is, we would like to preserve all of the pairwise distances between the points in each tessera. Thus we need to find $\mathbf{Y}_\ell$ such that its EDM is equal to $\mathbf{D}_\ell$ for $1 \leq \ell \leq c$. As is well-known (for example see [52]), the Gram matrix of a set of centered points whose EDM is $\mathbf{D}_\ell$, is given by double centering:

$$\mathbf{G}_\ell = -\frac{1}{2}\mathbf{H}_{n_\ell}\mathbf{D}_\ell\mathbf{H}_{n_\ell}, \tag{3.7}$$

where $\mathbf{H}_{n_\ell} = \mathbf{I}_{n_\ell} - \frac{1}{n_\ell}\mathbf{1}_{n_\ell}\mathbf{1}_{n_\ell}^\top$ is the centering matrix. It is clear that $\mathbf{G}_\ell$ is an $n_\ell \times n_\ell$ positive semidefinite matrix and therefore, can be decomposed into $\mathbf{G}_\ell = \mathbf{P}_\ell^\top\mathbf{P}_\ell$. If $\mathbf{Rank}(\mathbf{G}_\ell) = d_\ell$, then $\mathbf{P}_\ell \in \mathbb{R}^{d_\ell \times n_\ell}$ is a matrix of centered points, whose EDM is $\mathbf{D}_\ell$. We require $\mathbf{Y}_\ell$ to share the same EDM with $\mathbf{P}_\ell$, which means that there is an isometry between $\mathbf{Y}_\ell$ and $\mathbf{P}_\ell$. This isometry can be formulated by an orthonormal transformation (i.e. rotation and reflection), and a translation. Note that the pairwise Euclidean distances in each tessera are invariant to orthonormal transformations and translations.

$$\exists \mathbf{R}_\ell, \mathbf{t}_\ell : \quad \mathbf{Y}_\ell = \mathbf{R}_\ell\mathbf{P}_\ell + \mathbf{t}_\ell\mathbf{1}_{n_\ell}^\top = \begin{bmatrix}\mathbf{R}_\ell & \mathbf{t}_\ell\end{bmatrix}\begin{bmatrix}\mathbf{P}_\ell \\ \mathbf{1}_{n_\ell}^\top\end{bmatrix}, \tag{3.8}$$

where $\mathbf{R}_\ell$ consists of $d_\ell$ orthonormal columns (i.e. $\mathbf{R}_\ell^T\mathbf{R}_\ell = \mathbf{I}_{d_\ell}$), and $\mathbf{t}_\ell$ is the translation vector. The last term in Eq.3.8 is in fact the homogeneous coordinates of the centered points $\mathbf{P}_\ell$. We represent this known $(d_\ell + 1) \times n_\ell$ matrix by $\tilde{\mathbf{P}}_\ell$. Let $\mathbf{S}_\ell = \begin{bmatrix}\mathbf{R}_\ell & \mathbf{t}_\ell\end{bmatrix}$ be an unknown matrix of the transformations; then $\mathbf{Y}_\ell = \mathbf{S}_\ell\tilde{\mathbf{P}}_\ell$, and therefore, we can rewrite $\mathbf{Y}$ based on $\tilde{\mathbf{P}}_\ell$ and $\mathbf{S}_\ell$ of all tesserae:

[4] Note that it is always possible to provide such a tessellation for a given dataset if the size of its tesserae is small enough.

$$\mathbf{Y} = \begin{bmatrix} \mathbf{S}_1 & \mathbf{S}_2 & \dots & \mathbf{S}_c \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{P}}_1 & 0 & \dots & 0 \\ 0 & \tilde{\mathbf{P}}_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \tilde{\mathbf{P}}_c \end{bmatrix} = \mathbf{SU}, \tag{3.9}$$

where $\mathbf{U} \in \mathbb{R}^{\gamma \times n}$ is a known block-diagonal matrix, and $\mathbf{S}$ is an unknown matrix with $\gamma = c + \sum d_\ell$ columns. Finally, the Gram matrix of the embedded points will be equal to

$$\mathbf{G} = \mathbf{Y}^\top \mathbf{Y} = \mathbf{U}^\top \mathbf{S}^\top \mathbf{S} \mathbf{U} = \mathbf{U}^\top \mathbf{B} \mathbf{U}. \tag{3.10}$$

Here $\mathbf{B} = (\mathbf{S}^\top \mathbf{S}) \in \mathcal{S}_+^\gamma$ is an unknown symmetric positive semidefinite matrix. In practice $\gamma$, the size of the unknown matrix $\mathbf{B}$, is significantly less than $n$, because $d_\ell$'s are usually very small[5]. The matrix $\mathbf{B}$ has the following structure:

$$\mathbf{B} = \mathbf{S}^\top \mathbf{S} = \begin{bmatrix} \mathbf{S}_1^\top \mathbf{S}_1 & \mathbf{S}_1^\top \mathbf{S}_2 & \dots & \mathbf{S}_1^\top \mathbf{S}_c \\ \mathbf{S}_2^\top \mathbf{S}_1 & \mathbf{S}_2^\top \mathbf{S}_2 & \dots & \mathbf{S}_2^\top \mathbf{S}_c \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{S}_c^\top \mathbf{S}_1 & \mathbf{S}_c^\top \mathbf{S}_2 & \dots & \mathbf{S}_c^\top \mathbf{S}_c \end{bmatrix}, \tag{3.11}$$

so, on the diagonal of $\mathbf{B}$ we have the following blocks:

$$\mathbf{S}_\ell^\top \mathbf{S}_\ell = \begin{bmatrix} \mathbf{R}_\ell^\top \mathbf{R}_\ell & \mathbf{R}_\ell^\top \mathbf{t}_\ell \\ \mathbf{t}_\ell^\top \mathbf{R}_\ell & \mathbf{t}_\ell^\top \mathbf{t}_\ell \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{d_\ell} & \mathbf{R}_\ell^\top \mathbf{t}_\ell \\ \mathbf{t}_\ell^\top \mathbf{R}_\ell & \|\mathbf{t}_\ell\|^2 \end{bmatrix}. \tag{3.12}$$

Restricting the diagonal blocks of the unknown matrix $\mathbf{B}$ in the form specified by Eq.3.12 guarantees the orthonormality of $\mathbf{R}_1, \mathbf{R}_2, \dots, \mathbf{R}_c$. Therefore, having constraints of this form is sufficient to be certain that the pairwise distances within each tessera are preserved in the final embedding. By combining Eq.3.1 and Eq.3.10 we can obtain $\mathbf{A}$ as

$$\begin{aligned} \mathbf{G} = \mathbf{Y}^\top \mathbf{Y} = \mathbf{K}\boldsymbol{\Omega}\boldsymbol{\Omega}^\top \mathbf{K} = \mathbf{K}\mathbf{A}\mathbf{K} = \mathbf{U}^\top \mathbf{B} \mathbf{U} \\ \implies \quad \mathbf{A} = (\mathbf{K}^\dagger \mathbf{U}^\top) \mathbf{B} (\mathbf{K}^\dagger \mathbf{U}^\top)^\top = \mathbf{V}\mathbf{B}\mathbf{V}^\top, \end{aligned} \tag{3.13}$$

[5] If the points of a tessera do not form a low-rank matrix, we first project them into a low-dimensional subspace using PCA, and then compute EDM on the result. In this way, we guarantee that $\gamma \ll n$.

where $\mathbf{K}^\dagger$ is the pseudo-inverse[6] of $\mathbf{K}$, $\mathbf{V} = \mathbf{K}^\dagger \mathbf{U}^\top \in \mathbb{R}^{n\times\gamma}$ is known and can be calculated, and $\mathbf{B} \in \mathcal{S}_+^\gamma$ is unknown, where $\gamma \ll n$. Now, the optimization problem can be solved based on the new smaller unknown variable $\mathbf{B}$ and therefore, the computational complexity will be dramatically reduced.

The set $\{\mathbf{V}\mathbf{B}\mathbf{V}^\top : \mathbf{B} \in \mathcal{S}_+^\gamma\}$ is formally called a *face* of the semidefinite cone $\mathcal{S}_+^n$; so the substitution $\mathbf{A} = \mathbf{V}\mathbf{B}\mathbf{V}^\top$ is called a *semidefinite facial reduction* [11]. The idea is that by such a substitution, the search space in an SDP problem can be restricted to a small face of the semidefinite cone. This can be applied to reformulate the SDP problem in Eq.3.6, by substitution $\mathbf{A} = \mathbf{V}\mathbf{B}\mathbf{V}^\top$. Clearly, the new objective function can be reexpressed as:

$$\mathbf{s}^\top \mathbf{vec}(\mathbf{A}) = \mathbf{s}^\top \mathbf{vec}(\mathbf{V}\mathbf{B}\mathbf{V}^\top) = \mathbf{s}^\top (\mathbf{V}\otimes\mathbf{V})\mathbf{vec}(\mathbf{B}) = \mathbf{s}_{\text{new}}^\top \mathbf{vec}(\mathbf{B}), \tag{3.14}$$

where $\mathbf{s}_{\text{new}}^\top = \mathbf{s}^\top(\mathbf{V}\otimes\mathbf{V})$. To form the new set of constraints we first need to be certain that the aforementioned blocks on the diagonal of $\mathbf{B}$ are the identity matrices as shown in Eq.3.12. More formally, suppose $h_\ell = \sum_{i=1}^{\ell-1} d_i + \ell$; then we can represent the $\ell^{\text{th}}$ diagonal block of $\mathbf{B}$ by $\mathbf{B}_\ell = \mathbf{B}_{(\{h_\ell,\dots,h_\ell+d_\ell-1\},\{h_\ell,\dots,h_\ell+d_\ell-1\})}$. This block should be equal to the identity matrix of size $d_\ell$ and therefore, the matrix $\mathbf{B}$ should look like

$$\mathbf{B} = \begin{bmatrix} \mathbf{B}_1 & . & . & . & \dots \\ . & \|\mathbf{t}_1\|^2 & . & . & \\ . & . & \mathbf{B}_2 & . & \\ . & . & . & \|\mathbf{t}_2\|^2 & \\ \vdots & & & & \ddots \end{bmatrix}, \tag{3.15}$$

where for each tessera $\mathbf{B}_\ell = \mathbf{I}_{d_\ell}$, and $\|\mathbf{t}_\ell\|^2$ is an unknown scalar which is unconstrained. Secondly, we need to preserve the distance between neighboring tesserae. That is, we need to preserve the distance between two neighboring points if they belong to different tesserae. Using the notation of Eq.3.4, we can form a new matrix $\breve{\mathbf{\Xi}}_{\mathcal{S}}$ whose columns are

$$\{\boldsymbol{\xi}_{ij} : \quad (i,j) \in \mathcal{S} \ \text{ where } i \text{ and } j \text{ are not in the same tessera}\}. \tag{3.16}$$

[6] When $\mathbf{K}$ is full-rank, matrix inversion can be used to obtain $\mathbf{A}$. But in the case that $\mathbf{K}$ is rank-deficient, pseudo-inverse $\mathbf{K}^\dagger$ should be used. It is easy to see that the part of $\mathbf{A}$ which is in the null-space of $\mathbf{K}$ does not affect the optimization (based on the representor theorem). Therefore, without loss of generality, we can assume that the optimal answer for $\mathbf{A}$ is in the space spanned by $\mathbf{K}$, and thus, by $\mathbf{K}^\dagger$. Note that if $\mathbf{K}$ is rank-deficient, there is no guarantee to have $\mathbf{G}$ in the space spanned by $\mathbf{K}$. In other words, it is possible that no linear mapping from the feature space generates the desired gram matrix $\mathbf{G}$ of the tessellation. However, to minimize the difference between $\mathbf{K}\mathbf{A}\mathbf{K}$ and $\mathbf{G}$, the PSD matrix $\mathbf{A}$ should be computed as $\mathbf{K}^\dagger\mathbf{G}\mathbf{K}^\dagger$. Note that, in this case, the within-tesserae distances can slightly be distorted.

Note that this is equivalent to taking the old matrix $\boldsymbol{\Xi}_{\mathcal{S}}$ and removing the columns $\boldsymbol{\xi}_{ij}$, if $i$ and $j$ are in the same tessera. Then, the reduced constraint matrix $\breve{\mathbf{Q}}$ and vector $\breve{\mathbf{p}}$ are computed based on the new reduced $\breve{\boldsymbol{\Xi}}_{\mathcal{S}}$. Clearly this significantly reduces the rank of $\mathbf{Q}$. The new constraint matrix and vector can be represented as:

$$\breve{\mathbf{Q}}^{\top}\mathbf{vec}(\mathbf{A}) = \breve{\mathbf{Q}}^{\top}(\mathbf{V}\otimes\mathbf{V})\mathbf{vec}(\mathbf{B}) = \mathbf{Q}_{\text{new}}^{\top}\mathbf{vec}(\mathbf{B}) = \mathbf{p}_{\text{new}} = \breve{\mathbf{p}} = \breve{\boldsymbol{\Xi}}_{\mathcal{S}}\mathbf{1}. \tag{3.17}$$

Finally, the reduced optimization problem can be formed as:

$$\max_{\mathbf{B}\succeq 0} \quad \mathbf{s}_{\text{new}}^{\top}\mathbf{vec}(\mathbf{B}) \quad \text{subject to} \quad \begin{cases} \mathbf{Q}_{\text{new}}^{\top}\mathbf{vec}(\mathbf{B}) = \mathbf{p}_{\text{new}} \\ \forall \ell \in \{1,\dots,c\} : \mathbf{B}_{\ell} = \mathbf{I}_{d_{\ell}} \end{cases} \tag{3.18}$$

The size of the new semidefinite variable is $\gamma \times \gamma$. Let $r$ be the maximum rank[7] of the tesserae; then it is clear that $\gamma \in \mathsf{O}(rc)$. Considering the fact that the number of constraints in TesseraMap is definitely less than in EAT (which was in $\mathsf{O}(n)$), the complexity of solving the SDP problem in Eq.3.18 will be indeed lower than $\mathsf{O}(n^2r^2c^2)$. This indicates that TesseraMap is at least $\left(\frac{n}{rc}\right)^2$ times faster than EAT.

The scalability of TesseraMap can be improved further by controlling the number of constraints. For each tessera, the constraint $\mathbf{B}_{\ell} = \mathbf{I}_{d_{\ell}}$ guarantees that TesseraMap preserves the within-tessera distances in that tessera. Therefore, the number of constraints for preserving the within-tessera distances for all $c$ tesserae is bounded above by $\mathsf{O}(cr^2)$, i.e., the number of elements on the diagonal blocks of matrix $\mathbf{B}$. Note that in TesseraMap we have two different types of constraints: the within-tessera constraints and the between-tessera ones. Consequently, the total number of constraints is bounded below by $\Omega(cr^2)$ because of the number of within-tessera constraints.

The number of between-tessera constraints can increase the total number of constraints, which is not desirable for TesseraMap. In fact, this is where one should control the number of constraints; if the number of between-tessera constraints is bounded above by $\mathsf{O}(cr^2)$, the order of the total number of constraints does not increase and so, will be minimized. Considering that the number[8] of neighboring pairs of tesserae is in $\mathsf{O}(c)$, in order to have

[7] It is possible to set an upper bound $r$ for the rank of the tesserae. That is, if the rank of a tessera is higher than this upper bound, the points of the tessera should be projected to an $r$-dimensional space by applying PCA. The upper bound can be set to the value of the target dimensionality.

[8] Assuming that the underlying manifold is low-dimensional, the number of neighboring tesserae for each tessera can be considered in $\mathsf{O}(1)$.

at most $\mathsf{O}(cr^2)$ between-tessera constraints, for each two neighboring tesserae, we should consider at most[9] $r^2$ neighboring points. In this way, the total number of constraints will be in $\mathsf{O}(cr^2)$, and so, the overall complexity of solving SDP will be in $\mathsf{O}(c^4r^6)$, which interestingly does not depend on $n$ and thus, is highly scalable. In our experiments we realized that a dataset consisting of $n$ data points can usually be represented by $c \in \mathsf{O}(\sqrt{n})$ tesserae. Hence, considering that $r$ is generally a small constant, the total complexity is roughly of $\mathsf{O}(n^2)$, which is $n^2$ times faster than EAT.

### 3.1.2 The Effect of Full Rank Kernels

Recall that $\mathbf{s}$ and $\mathbf{Q}$ are computed from the vectors $\boldsymbol{\xi}_{ij} = \frac{1}{\tau_{ij}^2}(\mathbf{k}_i - \mathbf{k}_j) \otimes (\mathbf{k}_i - \mathbf{k}_j)$. It can be seen that the term $\boldsymbol{\xi}_{ij}^\top (V \otimes V)$ in both vector $\mathbf{s}_{\text{new}}$ and matrix $\mathbf{Q}_{\text{new}}$ makes them independent of the choice of kernel, as long as the kernel matrix is full-rank and thus invertible ($\mathbf{K}^{-1} = \mathbf{K}^\dagger \Rightarrow \mathbf{K}\mathbf{K}^\dagger = \mathbf{K}^\dagger\mathbf{K} = \mathbf{I}$):

$$\begin{aligned}
\boldsymbol{\xi}_{ij}^\top(\mathbf{V} \otimes \mathbf{V}) &= \frac{1}{\tau_{ij}^2}\left((\mathbf{k}_i - \mathbf{k}_j) \otimes (\mathbf{k}_i - \mathbf{k}_j)\right)^\top (\mathbf{K}^\dagger\mathbf{U}^\top \otimes \mathbf{K}^\dagger\mathbf{U}^\top) \\
&= \frac{1}{\tau_{ij}^2}((\mathbf{k}_i - \mathbf{k}_j)^\top\mathbf{K}^\dagger\mathbf{U}^\top) \otimes ((\mathbf{k}_i - \mathbf{k}_j)^\top\mathbf{K}^\dagger\mathbf{U}^\top) \\
&= \frac{1}{\tau_{ij}^2}((\mathbf{e}_i - \mathbf{e}_j)^\top\mathbf{U}^\top) \otimes ((\mathbf{e}_i - \mathbf{e}_j)^\top\mathbf{U}^\top) \\
&= \frac{1}{\tau_{ij}^2}(\mathbf{u}_i - \mathbf{u}_j)^\top \otimes (\mathbf{u}_i - \mathbf{u}_j)^\top
\end{aligned} \tag{3.19}$$

where $\mathbf{e}_i$ is the $i^{\text{th}}$ column of the identity matrix $\mathbf{I}_n$ and $\mathbf{u}_i = \mathbf{U}\mathbf{e}_i$ is the $i^{\text{th}}$ column of $\mathbf{U}$. Crucially Eq.3.19 implies that the matrix $\mathbf{B}$, obtained from the optimization problem Eq.3.18, is independent of the choice of kernel. However, this is not the case for the transformation $\boldsymbol{\Omega}$. In Eq.3.13 we showed how to decompose the matrix $\mathbf{A}$ as $\mathbf{V}\mathbf{B}\mathbf{V}^\top$ where $\mathbf{V} = \mathbf{K}^\dagger\mathbf{U}^\top$. Since $\mathbf{A} = \boldsymbol{\Omega}\boldsymbol{\Omega}^\top$, the transformation $\boldsymbol{\Omega}$ can be expressed as

$$\boldsymbol{\Omega} = \mathbf{V}\mathbf{B}^{\frac{1}{2}} = \mathbf{K}^\dagger\mathbf{U}^\top\mathbf{B}^{\frac{1}{2}}. \tag{3.20}$$

[9] Having many distance constraints between two low-rank tesserae may remove degrees of freedom between them, and fix them together, which is undesirable in the process of unfolding. Therefore, it is sensible to limit the number of these constraints, for example by considering only the $r$ nearest points between each two neighboring tesserae.

Now we can find the embedding of the training points, $\mathbf{X}$, by:

$$\mathbf{Y} = \mathbf{\Omega}^\top \mathbf{K} = (\mathbf{B}^{\frac{1}{2}})^\top \mathbf{U}\mathbf{K}^\dagger \mathbf{K}. \tag{3.21}$$

If $\mathbf{K}$ is full-rank, $\mathbf{Y} = (\mathbf{B}^{\frac{1}{2}})^\top \mathbf{U}$, and since $\mathbf{B}$ is independent of the kernel function, the embedded training set, $\mathbf{Y}$, is also independent of the kernel function. In other words, as long as the kernel $\mathbf{K}$ is full-rank, the result of embedding for the training set is the same. Note that different kernels result in different transformations $\mathbf{\Omega} = \mathbf{K}^\dagger \mathbf{U}^\top \mathbf{B}^{\frac{1}{2}}$. That is, Eq.3.20 defines a family of transformations between the high-dimensional feature space and the embedding space; for the full-rank kernels, all members of this family yield exactly the same result when applied to the training set. This is not, however, true when it comes to out-of-sample points; the out-of-sample mapping is affected by the choice of kernel.

$$\mathbf{y} = \mathbf{\Omega}^\top \phi(\mathbf{X})^\top \phi(\mathbf{x}_t) = (\mathbf{B}^{\frac{1}{2}})^\top \mathbf{U}\mathbf{K}^\dagger \kappa(\mathbf{X}, \mathbf{x}_t). \tag{3.22}$$

It is worth mentioning that if only the embedding of the training set is concerned, it is possible to assume that a full-rank kernel has been used, and based on Eq.3.19, simplify the computations. In this way, the computed constraint matrix $\mathbf{Q}_{\text{new}}$ will be sparse, giving rise to reduction of the complexity down to $\mathsf{O}\left(c^3 r^6\right)$, or roughly $\mathsf{O}\left(n^{\frac{3}{2}}\right)$. However, note that in this case, overfitting occurs in TesseraMap, and therefore, it is not possible to map out-of-samples properly.

### 3.1.3 The Algorithm

The learning phase of TesseraMap is summarized in Alg.3. At the end of this algorithm, we use PCA to reduce the dimensionality of the embedded data to the desired dimensionality. In other words, we compute a transformation matrix whose columns are the eigenvectors of $\mathbf{Y}\mathbf{Y}^\top$ (corresponding to the top eigenvalues), and use it to project $\mathbf{Y}$ and reduce its dimensionality. Note that similar to EAT, the kernel matrix of the input data should be double centered (line 4).

After the learning phase, the computed transformation can be used to map any new out-of-sample point, as it was used in EAT. Note that before using Eq.3.22, the similarity vector $\kappa(\mathbf{X}, \mathbf{x}_t)$ should be centered. Also, the same PCA projection used in the learning phase should be applied to reduce the dimensionality of the mapped out-of-sample point.

**Algorithm 3** TesseraMap - Learning

1: Compute the pairwise distances $\forall 1 \leq i, j \leq n : \tau_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|$

2: Based on $\{\tau_{ij}\}$, compute a neighborhood graph and form the sets $\mathcal{S}$ and $\mathcal{O}$

3: Choose a kernel function $\kappa(\cdot, \cdot)$ and compute the kernel matrix $\mathbf{K}$

4: Apply double centering to the kernel: $\mathbf{K} \leftarrow \mathbf{H}_n \mathbf{K} \mathbf{H}_n$

5: Perform clustering on $\mathbf{X}$ and find the tesserae $\{\mathcal{C}_\ell\}_{\ell=1}^{c}$,
for each tessera $\mathcal{C}_\ell$:

5-A: Calculate the EDM of $\mathbf{X}_\ell$ ($\mathbf{D}_\ell$)

5-B: Calculate the Gram matrix $\mathbf{G}_\ell$ from Eq.3.7

5-C: Obtain the centered points $\mathbf{P}_\ell$

5-D: Form the homogenous coordinates in $\tilde{\mathbf{P}}_\ell$

6: Compute $\mathbf{U}$ from Eq.3.9

7: Calculate $\mathbf{Q}_{\text{new}}$, $\mathbf{p}_{\text{new}}$ and $\mathbf{s}_{\text{new}}$ from Eq.3.17 and Eq.3.14

8: Solve the optimization

$$\max_{\mathbf{B} \succeq 0} \quad \mathbf{s}_{\text{new}}^{\top} \mathbf{vec}(\mathbf{B}) \quad \text{subject to} \quad \begin{cases} \mathbf{Q}_{\text{new}}^{\top} \mathbf{vec}(\mathbf{B}) = \mathbf{p}_{\text{new}} \\ \forall \ell \in \{1, \ldots, c\} : \mathbf{B}_\ell = \mathbf{I}_{d_\ell} \end{cases}$$

9: Compute the mapped $\mathbf{X}$ in the embedding space $\mathbf{Y} = (\mathbf{B}^{\frac{1}{2}})^{\top} \mathbf{U} \mathbf{K}^{\dagger} \mathbf{K}$

10: Apply PCA to $\mathbf{Y}$ and return the obtained low-dimensional embedding

## 3.2 Experimental Results

We have conducted some experiments with TesseraMap using different synthetic and real-world datasets to evaluate its performance. TesseraMap is implemented in MATLAB, and uses SeDuMi [59] as its SDP solver. In all of the experiments, we used affinity propagation clustering (APC) [22] to cluster the given dataset. The input to APC is a set of similarities between the data points, which is usually the negative of the Euclidean distances between them, and also a preference value. In all the experiments, we set the preference value as the median of the similarities. APC does not require the user to predetermine the number of clusters. Instead the number of clusters will be implicitly determined by the preference value. This removes the need for tuning the clustering parameters.

| Dataset | Train Size | Test Size | TesseraMap | $c$ | Fast MVU |
|---|---|---|---|---|---|
| Dual Spiral | 350 | 5,000 | 57 | - | - |
| Swiss-roll | 30,000 | 20,000 | 118 | 108 | 174 |
| World Map | 15,000 | 30,000 | 32 | 84 | 128 |
| Frey Faces | 965 | 1,000 | 4 | 36 | 59 |
| MNIST Digits | 3000 | 20,975 | 6 | 19 | - |

Table 3.1: The number of samples and run times of TesseraMap and Fast MVU (in seconds).

In addition to some qualitative comparisons with popular methods, we compared the efficiency of our method with Fast MVU [71] (with the default parameters), which is capable of embedding large datasets. The run times of both methods are listed in Table 3.1. In the experiments of this paper, we used linear kernels and RBF kernels $\kappa(\mathbf{x}_i, \mathbf{x}_j) = e^{\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}}$.

### 3.2.1 Embedding Synthetic Data I - *Dual Spiral Manifold*

The first dataset we considered was a three-dimensional dual spiral manifold, depicted in the left panel of Fig.3.2. It is known that some of the dimensionality reduction methods are not able to handle non-convex manifolds [37]. Therefore, in order to evaluate TesseraMap in the context of complex manifolds, we made the manifold non-convex, by placing a hole in its center. We compared the TesseraMap result with that of many popular dimensionality reduction algorithms, in the training phase and also in out-of-sample embedding.

First, we uniformly sampled 350 training points, and formed a neighborhood network, which is shown in Fig.3.1 (top-left). Then we applied TesseraMap, MVU, PCA, Kernel PCA, LLE, Isomap, LLTSA, LTSA, LPP, and Kernel LPP to this training set using the same neighborhood network. For TesseraMap, Kernel PCA, and Kernel LPP, we used an RBF kernel with $\sigma = 0.5$. Since the training dataset was small, the TesseraMap algorithm could be applied directly and without performing clustering on the data. This means that we simply used the new formulation Eq.3.6 to solve EAT, which does not require EVD of $\mathbf{Q}$, and is consequently less complex.

It can be seen that the embedding of TesseraMap is completely conformal for the given training set, and having a hole on the manifold has not affected the quality of the result. As expected, MVU has unfolded the manifold as well, and returned a similar result. In contrast, due to the complex nonlinear structure of the manifold, all of the linear methods (PCA, LLTSA, LPP) have failed to unfold it and therefore, their resulted embeddings are not faithful. Even the kernelized versions of PCA and LPP have not provided a proper

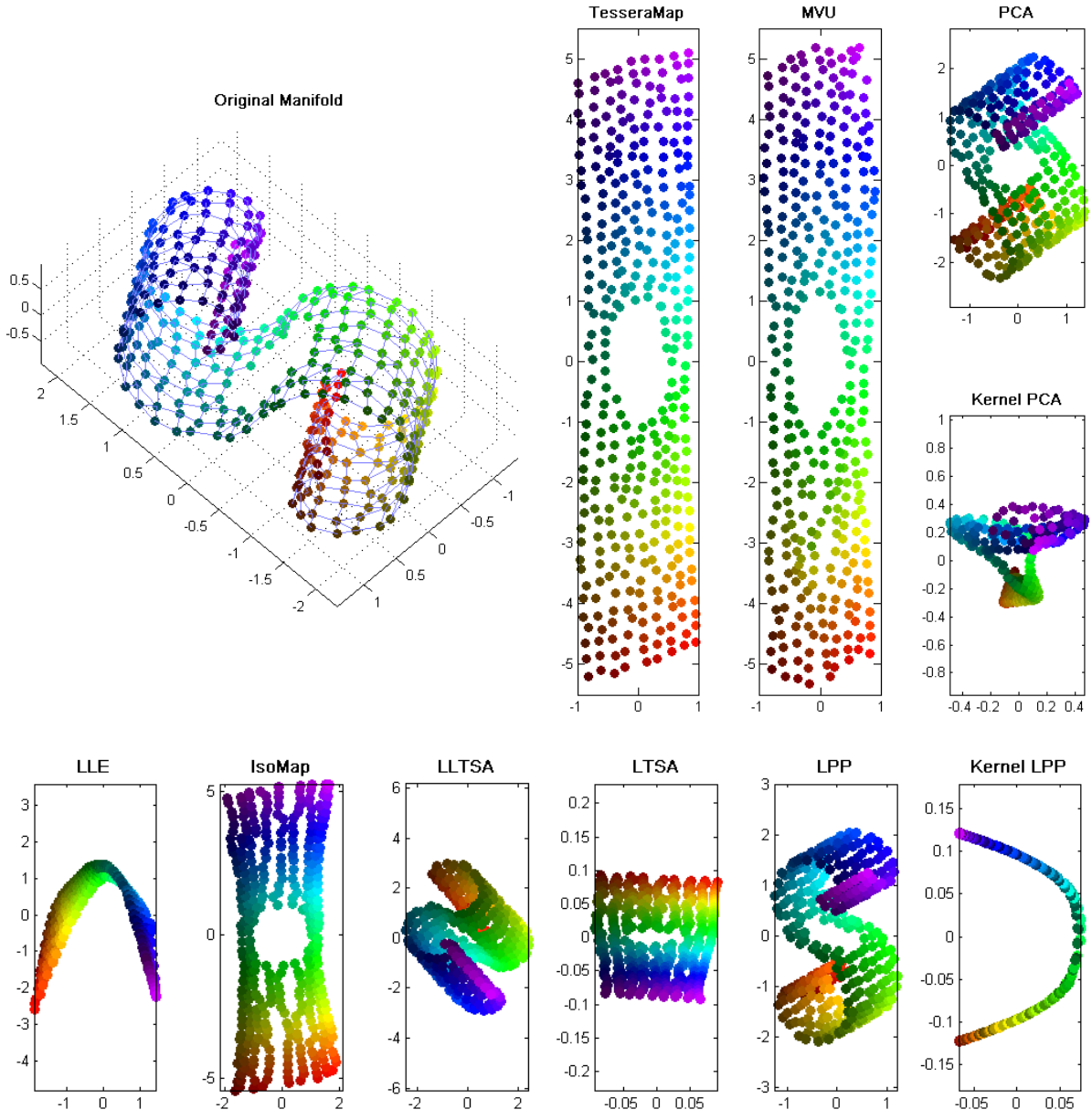


Figure 3.1: A set of 350 training points, uniformly sampled from a three-dimensional dual spiral manifold with a large hole in its center. The results of embedding by TesseraMap, MVU, PCA, Kernel PCA, LLE, Isomap, LLTSA, LTSA, LPP, and Kernel LPP on this training set are shown to compare their quality.

embedding, which is in fact due to their sensitivity to the choice of kernel; the RBF kernel apparently does not unfold the given manifold. It should be noted that we tried many different kernel functions, in order to improve the quality of the results of KPCA and Kernel LPP; however, we were not able to find any closed-form kernel capable of unfolding this manifold. Although LLE is a powerful method in the face of nonlinearity, it is known that it has difficulties when confronted with non-convex manifolds. In this case, LLE has collapsed segments of the manifold in the embedding space, as shown in Fig.3.1 (bottom-left). Isomap has provided a relatively better result, although computing correct geodesic distances on a non-convex manifold is not possible; thus, its embedding has been affected so that the width of the unfolded manifold is 4 instead of 2 (note that Isomap is supposed to preserve the distances on the manifold). Finally, we observe that the embedding of LTSA is also meaningful, although it has not preserved the scales.

In the next step, we uniformly sampled 5,000 out-of-sample points from the same manifold, depicted in Fig.3.2 (top-left), and applied the resulting mappings of TesseraMap, PCA, KPCA, LPP and Kernel LPP to this set of points. The results are depicted in Fig.3.2. Since none of these methods, except TesseraMap, have been able to unfold the manifold (in the previous experiment), their results for the out-of-sample set cannot be appropriate. In contrast, the mapping of TesseraMap was able to map out-of-sample points with the same quality as the training set. Furthermore, in the resulting embedding the hole can be seen clearly on the unfolded manifold, which demonstrates that TesseraMap preserves the local pattern of data in face of complex non-convex manifolds.

### 3.2.2 Embedding Synthetic Data II - *Swiss-roll Manifold*

In this experiment, we considered a large dataset containing 30,000 training points randomly sampled from a Swiss-roll. We also randomly sampled 20,000 test points from this manifold. We applied TesseraMap to the training dataset and used APC to cluster the data, which found $c = 108$ clusters in this training set. The size of the reduced matrix, $\mathbf{B}$, was nearly one percent of the original matrix $\mathbf{A}$. We also applied Fast MVU to the same training dataset. The unfolded manifolds are depicted in Fig.3.3, which illustrates the superiority of the TesseraMap embedding.

The learning phase in TesseraMap took 118 seconds, and Fast MVU solved the problem in 174 seconds. Nevertheless, it is clear from the outcome of Fast MVU in Fig.3.3 (bottom-right), that it was not successful. The original manifold is evidently a rolled rectangle, whose width is 20, however, Fast MVU has not been able to capture its rectangular shape, and also, the width of its embedding is between 2 to 8. This experiment explains that despite its longer run time, Fast MVU may produce comparatively low quality embeddings.

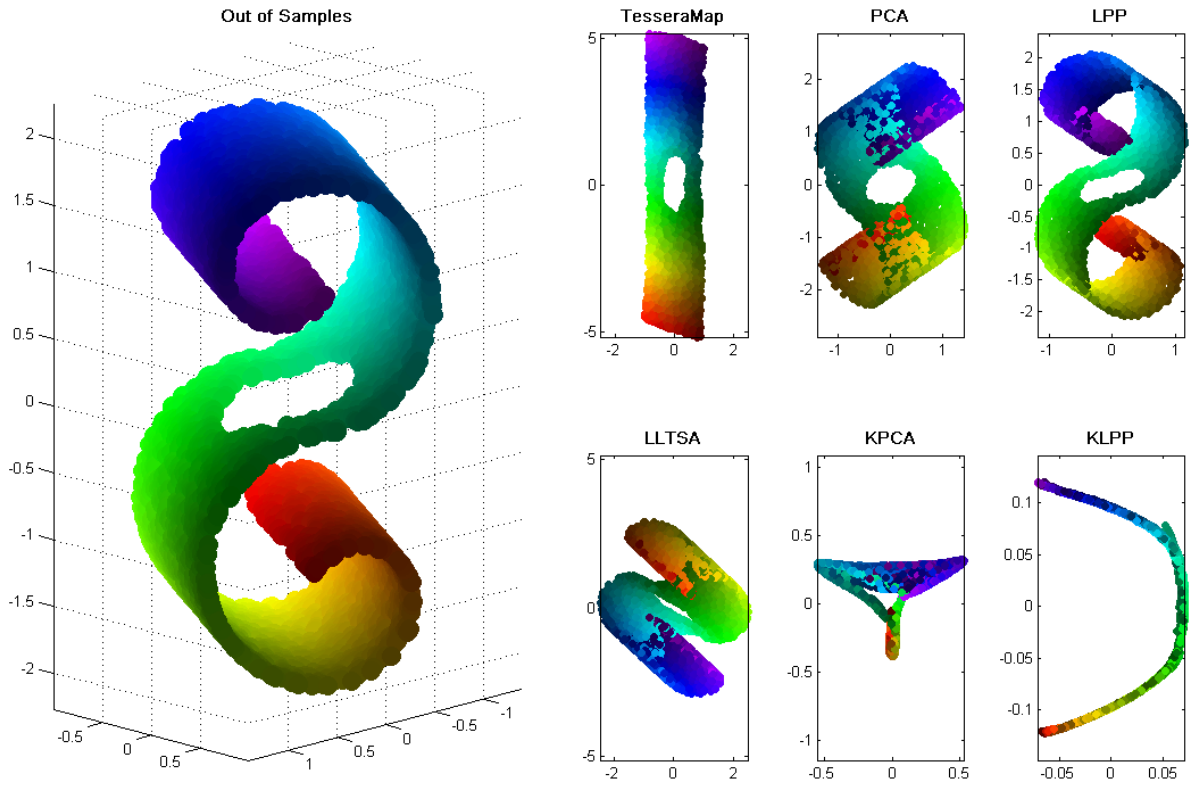


Figure 3.2: A set of 5,000 test points uniformly sampled from the manifold, and the results of applying the mappings of TesseraMap, PCA, LPP, LLTSA, Kernel PCA, and Kernel LPP that were learned in the training phase.

### 3.2.3 Embedding Real World Data I - *Map of Cities*

The first real dataset we considered was a large map of randomly selected cities from Europe, Asia, and Africa. We downloaded the *World Cities* dataset[10], which includes the latitude and longitude of nearly 2.7 million cities. We randomly selected 45,000 cities, and mapped their spherical coordinates to the three-dimensional Cartesian coordinates. We put the coordinates of 15,000 cities in the training dataset and the remaining 30,000 in the out-of-sample test dataset. We used an RBF kernel with $\sigma = 1$ and applied TesseraMap to the three-dimensional dataset; APC found 84 tesserae in the training set and SeDuMi solved the SDP in 32 seconds. After training, we used the resulting transformation to

[10]Can be downloaded from http://geolite.maxmind.com/download.

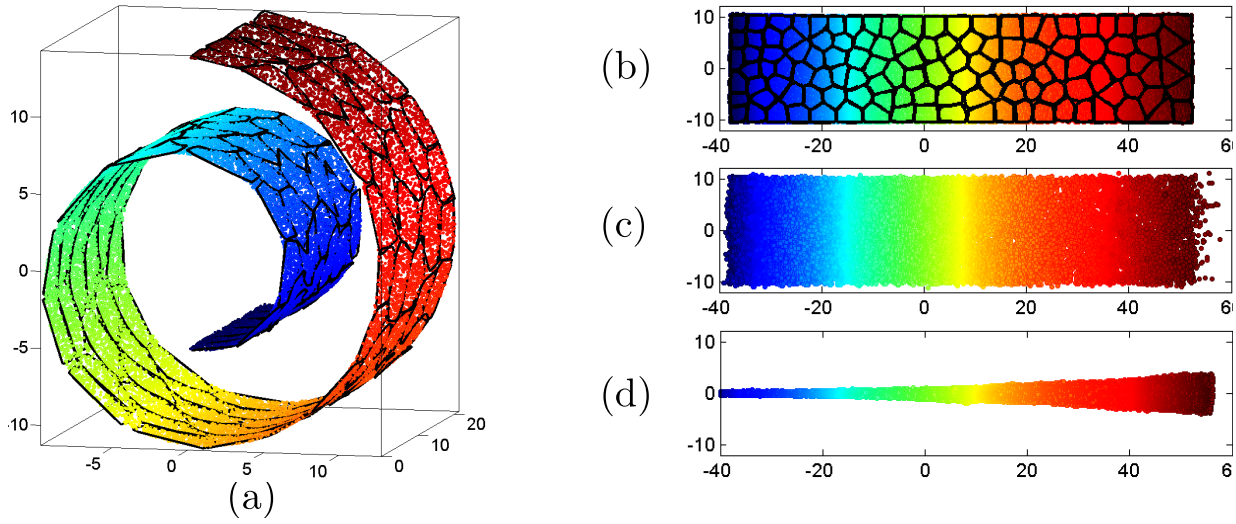


Figure 3.3: (a) 30,000 points sampled from the manifold with the corresponding tesserae; (b) embedding of the training set by TesseraMap (tesselation of the linear subspace); (c) embedding of 20,000 new out-of-sample points by Tessermap; (d) embedding by Fast MVU.

map the 30,000 test points. Fast MVU was also applied to the same dataset; the original three-dimensional dataset together with the two-dimensional embeddings are depicted in Fig.3.4. It can be seen that TesseraMap has produced a better result on both training and test sets, and has been faithful to the original map. Alternatively, the two-dimensional map generated by Fast MVU is visibly distorted, although it required 96 seconds more than TesseraMap, to solve the problem.

### 3.2.4 Embedding Real World Data II - *Face Images*

In this experiment, we used 1965 $28 \times 20$ pixel images of Brendan Frey's face taken from a short video. These images show his face in different moods: happy, angry, frowning, and so on. We randomly split the images into 965 training and 1,000 test points, and ran TesseraMap on this dataset. Since the original dimensionality of the data was high enough, we simply used a linear kernel in TesseraMap. APC found $c = 36$ tesserae and the SDP was solved in 4 seconds. We also applied Fast MVU to the training dataset; the embeddings are shown in Fig.3.5. It can be observed that in the TesseraMap mapping of the training set, images with similar facial expressions are bundled together; the same statement is true for the out-of-sample test images, whereas Fast MVU has failed to provide an interpretable structure in 59 seconds.

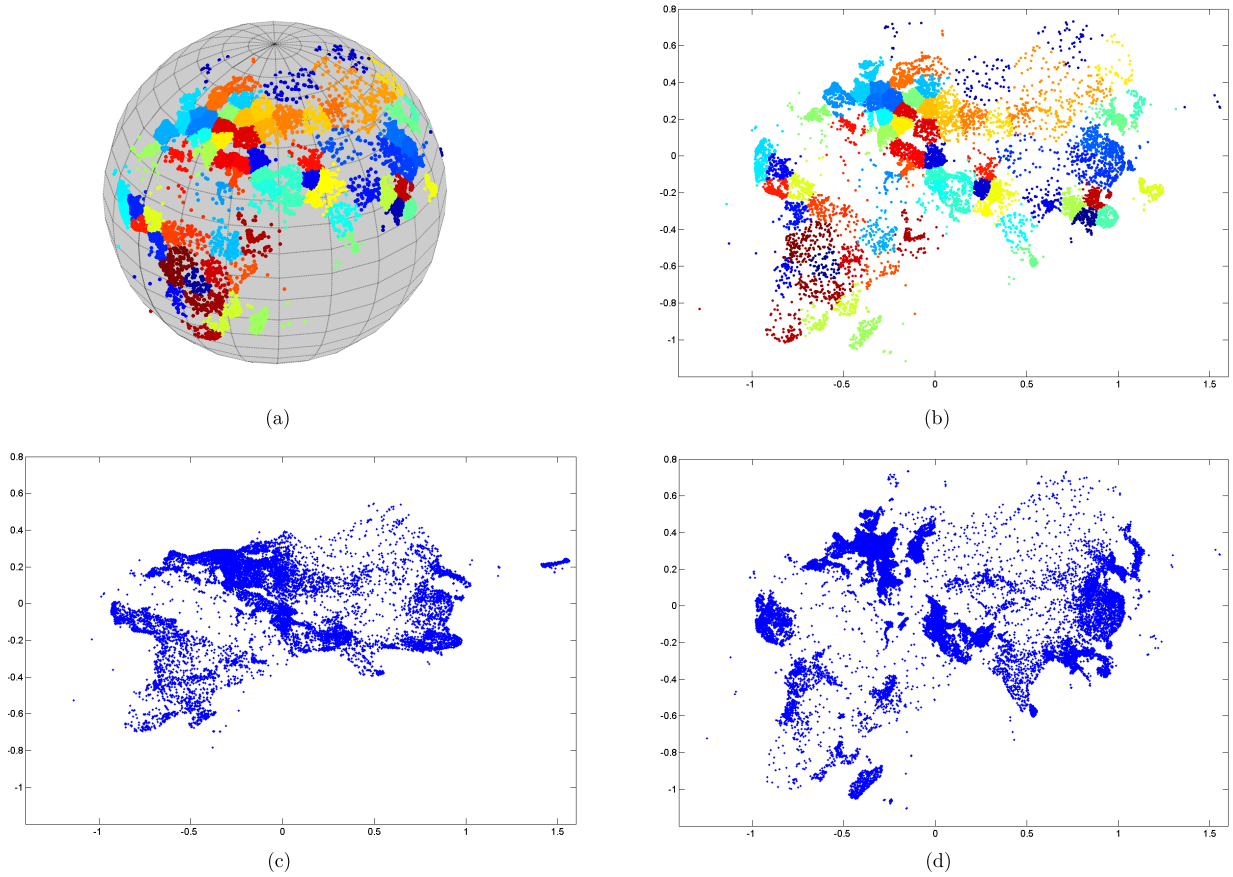

Figure 3.4: (a) 15,000 cities sampled from Africa, Asia, and Europe with the corresponding tesserae; (b) TesseraMap embedding of the training set; (c) embedding by Fast MVU; (d) TesseraMap embedding of the 30,000 new out-of-sample cities.

### 3.2.5 Embedding Real World Data III - *MNIST Digits*

A challenging real world dataset was chosen in our last experiment. This set is called MNIST [35], and contains many handwritten samples of single digits (0 to 9). Each sample is a 28 by 28 greyscale image.

It is generally reasonable to assume that in a dataset containing different categories of high-dimensional data, the points of each category lie on or close to an individual manifold in the space. Consequently, the dataset can be modeled by the union of these manifolds.

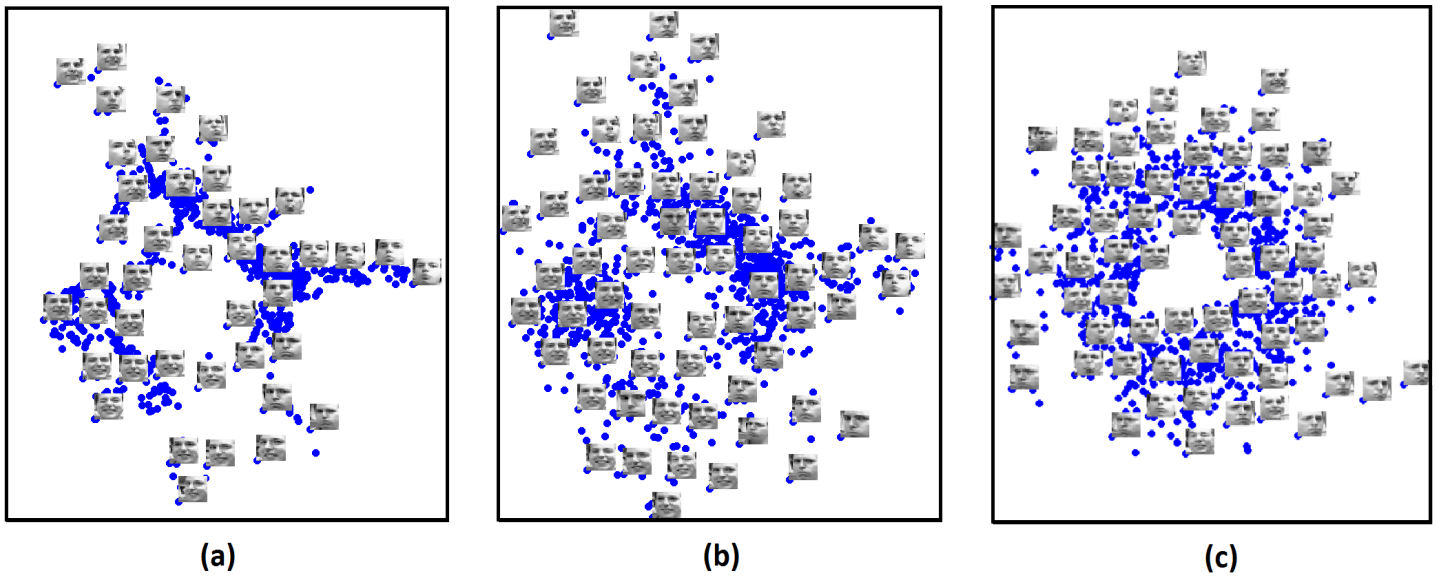


Figure 3.5: (a) Embedding of 965 training images by TesseraMap and (b) embedding of the 1,000 new out-of-sample images; (c) embedding by Fast MVU.

In addition, for any two disjoint categories of data, their underlying manifolds do not intersect. That is, there is a gap between their manifolds, from which no point is sampled. Crucially, by only considering the local patterns of data, it is not possible to relate two disjoint manifolds together - because there is no point sampled from the gap between them, and so the local patterns are not defined there to relate the manifolds.

As mentioned in Chapter 1, most dimensionality reduction methods supposedly preserve the local patterns (e.g., LLE, LTSA, and MVU). A common practice in dimensionality reduction is to create a neighborhood graph for the sampled points, and define the local patterns based on this graph. Generally, this neighborhood graph is required to be connected; therefore, the disjoint manifolds are attached to their closest neighbors by few, long-distance links. In fact, this is how disjoint manifolds are handled in most dimensionality reduction methods. However, this approach results in placing the disjoint manifolds, almost arbitrarily with respect to each other, in the low-dimensional representation of the dataset. Unfortunately, this issue is inevitable, because most of these methods consider a single underlying manifold for the entire dataset, and attempt to unfold it. Note that it is always possible to assume that the data points are sampled only from one underlying manifold, although this apparently non-convex manifold cannot be low-dimensional.

MNIST is a union of manifolds; the images of each digit form a manifold in the original high-dimensional space. It is indeed possible to unfold each of these manifolds separately, and return each embedding individually (as TessseraMap has done in Fig.3.6), but based on the above discussion, embedding all of these submanifolds together is rather meaningless, and does not necessarily represent the true local pattern of the data.

Considering the aforementioned issue, we selected only a subset of digits (3, 6, and 9) which form neighboring manifolds in the original input space. From each digit, we took 1000 random samples to form a training set of size 3000 and dimensionality 784. The training set was then clustered into 19 tesserae by APC. Since the dimensionality of the data was not higher than the number of the data points, an RBF kernel was used. It is important to note that this experiment has been performed completely unsupervised.

The result of this embedding is presented in Fig.3.8 (top) where the samples of each digit are shown with a different color. In the embedding of the training set, the variation of images apparently has been considered, while the samples of different digits have been clearly separated[11]. The separation boundary of each digit is clear, and only a few samples are misplaced - which is absolutely natural due to the fact that the images of two different handwritten digits can be very similar. For the test set, we used all of the 20,975 image

[11] Since TesseraMap is an unsupervised method that preserves all of the local distances, we do not expect to see that all of the digits become separated in the embedding.

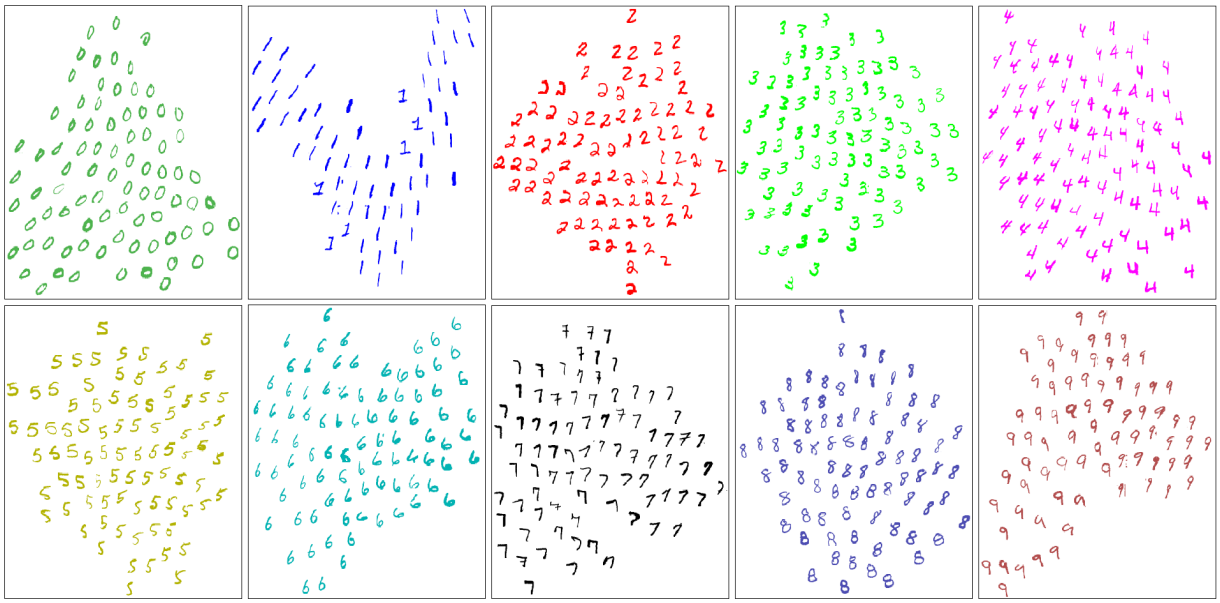

Figure 3.6: The visualization of all ten digits from the MNIST dataset, each one separately embedded by TesseraMap

samples of these three digits. The embedding of the test set is depicted in Fig.3.8 (bottom). In addition, in Fig.3.7, a three-dimensional visualization of the training set is depicted. As expected, three disjoint but neighboring manifolds are visualized in this figure. This experiment demonstrates that TesseraMap can be employed in visualization tasks, even when the input dataset consists of more than one category of data.

## 3.3 Conclusion and Discussion

We proposed TesseraMap, a novel algorithm for nonlinear dimensionality reduction which is able to overcome some shortcomings shared by many other methods. In particular, it is efficient and scalable to large datasets, and it is able to map out-of-sample points into the embedding space. This algorithm reduces the input dataset to a tessellation, and uses a semidefinite facial reduction technique, to unfold the global structure into a low-dimensional subspace while preserving the local geometry of the tesserae. Experimental results on synthetic and real-world datasets demonstrate that TesseraMap is computationally efficient for dimensionality reduction, and can produce high-quality embeddings for both training data and out-of-sample points.

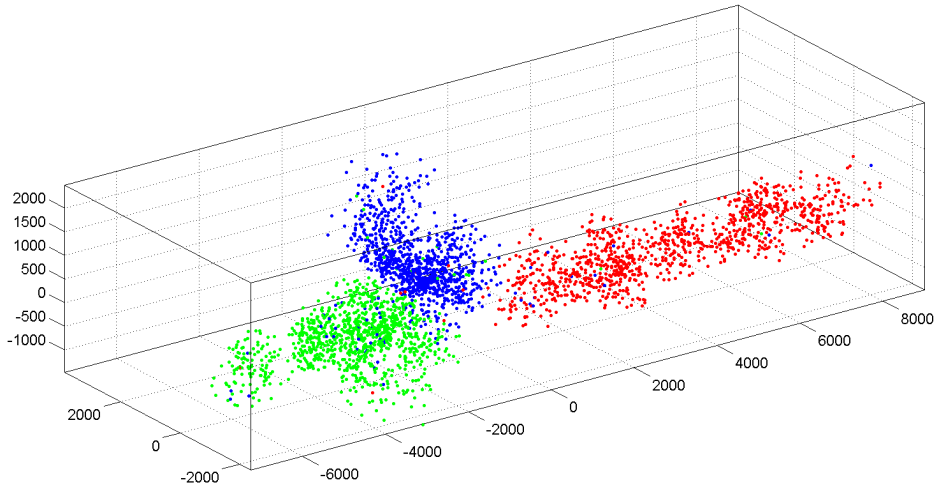


Figure 3.7: The embedding of the digits 3,6, and 9, from the MNIST dataset into three dimensions by TesseraMap.

An implicit assumption in TesseraMap is the existence of a low-dimensional underlying manifold for the input dataset. This is a common assumption in most dimensionality reduction techniques, and if violated, there will be a non-negligible loss of information in the results. Moreover, real-world datasets may consist of more than one manifold. In these cases, TesseraMap should be applied to each manifold separately. This is a common issue for all dimensionality reduction techniques which depend on the neighborhood graph of the input point, e.g., MVU, Isomap, and LLE.

Another assumption in TesseraMap, and also in the other well-known methods, is regarding the smoothness of the underlying manifold. When the underlying manifold is smooth, if the input data points are clustered into small tesserae, the points of each tessera will be close to a low-dimensional affine subspace, and so, form a low-rank subset of the data points. Consequently, the sum of all $d_\ell$, the rank of tesserae, will be small, which significantly reduces the size of the SDP problem.

As mentioned earlier, if the rank of each tessera is small, TesseraMap can directly be employed to solve the problem and find the tessellation of the embedding subspace. Since for unfolding, the tesserae are pulled apart from each other, the rank of the tessellation is expected to be as low as the maximum rank of the tesserae. In contrast, if the maximum rank is high, the user should preset a target dimensionality $r$, and project the points of

each tessera to an $r$-dimensional subspace using PCA. This is because PCA is fast and produces the lowest reconstruction error; however, it is possible to employ any other distance preserving method, instead of PCA, in this respect. In the dimensionality reduction tasks, the target dimensionality is usually very small. However, it is important to note that if the given $r$ is high, such that $rc \simeq n$, the size of the SDP problem does not decrease.

We employed APC for forming the tesserae, however it is possible to use any other clustering method. Specially when the underlying manifold is smooth and the size of the input data is relatively large, using fast clustering methods such as $k$-means is beneficial. Unfortunately, none of these methods guarantees production of low-rank clusters. However, based on the definition of a smooth manifold, any small localized subset of the manifold can approximately be considered as an affine subspace of dimensionality close to the intrinsic dimensionality of the manifold.

It is also important to consider that if the kernel matrix is not full-rank, there is no guarantee that the transformation $\mathbf{\Omega}$ forms a proper tessellation in the embedding subspace. However, being full-rank is a sufficient condition, not a necessary one. This is specifically important when the intrinsic dimensionality of the underlying manifold is low, so a few basis vectors are required to span the low-dimensional subspace of the unfolded manifold. That is why EAT, with low-degree polynomial kernels, was able to unfold the three-dimensional Swiss-roll manifold in Chapter 2.

We showed that full-rank kernels produce the same embedding for the training set. Suppose the embedding of the training set with the full-rank matrices is $\mathbf{Y}_{\text{tr}}$. The embedding of the training set with a rank-deficient kernel $\mathbf{K}$ is in fact the reconstruction of $\mathbf{Y}_{\text{tr}}$ based on the basis vectors of $\mathbf{K}$, i.e. $\mathbf{Y}_{\text{tr}}\mathbf{K}^{\dagger}\mathbf{K}$. Therefore, we would suggest to embed the training set with an identity kernel matrix, which results in a sparse constraint matrix, and consequently reduces the complexity of the SDP.

The above argument is also true for EAT. We should mention that the main difference between the way EAT searches for the transformation $\mathbf{\Omega}$ and the way TesseraMap does is that in EAT, $\mathbf{Q}$ is decomposed by EVD, which due to the large size of $\mathbf{Q}$ would take a long time. However, the optimization of EAT is formulated in the dual form, with a small number of unknown variables, using $\mathbf{vech}(\cdot)$ operator. In TesseraMap, we wrote the problem in the primal form and omitted the EVD step on $\mathbf{Q}$. Moreover, to reduce the number of the constraints in TesseraMap, we would suggest applying SVD on $\mathbf{\Xi}_{|\mathcal{S}|}$ to simplify the constraint $\mathbf{\Xi}_{\mathcal{S}}\mathbf{\Xi}_{\mathcal{S}}^{\top}\mathbf{vec}(\mathbf{A}) = \mathbf{\Xi}_{\mathcal{S}}\mathbf{1}_{|\mathcal{S}|}$. Note that TesseraMap is an extension for EAT, and their major difference lies in the scalability of these methods.

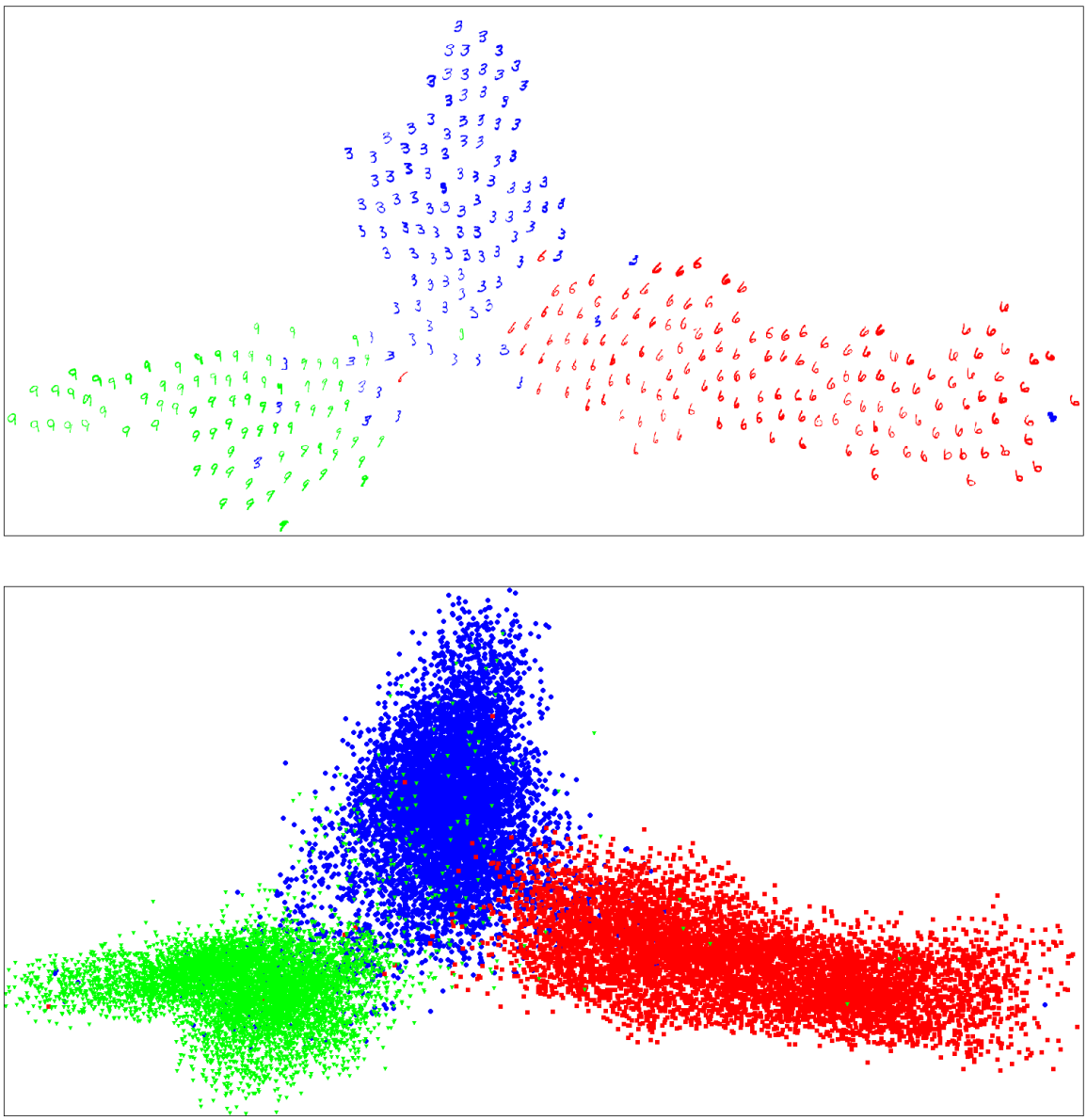

Figure 3.8: The embedding of the digits 3,6, and 9, from the MNIST dataset into two dimensions by TesseraMap; (top) the training images; (bottom) the out-of-sample points

# Chapter 4

# Low-dimensional Localized Clustering

In this chapter, we propose *Low-dimensional Localized Clustering (LDLC)*, a novel method for subspace clustering. The proposed method partitions the input data into clusters, such that the points of each cluster are close to a low-dimensional subspace[1]. In addition, the resulting clusters are localized on their underlying manifold. As will be explained, these two important properties make LDLC a suitable candidate for clustering high-dimensional data in TesseraMap.

In the world of high-dimensional data, it is generally reasonable to assume that the data points are sampled from manifolds in a space, where each underlying manifold models one category of the data. If two categories of data are disjoint, their corresponding manifolds are separated, but if they share some points, their manifolds intersect one another and may form a complex object in the space.

Based on the definition of manifold, the neighborhood of each point on its underlying manifold is homeomorphic to a low-dimensional Euclidean space. Most nonlinear dimensionality reduction methods exploit this intuition, and by preserving local structures, these methods attempt to represent the underlying manifold of data in a low-dimensional space. Nevertheless, there is another approach for dimensionality reduction that does not explore the structure of the underlying manifolds. It is based on the assumption that a manifold can be modeled approximately by a union of affine subspaces. Consequently in this approach, the high-dimensional input data is partitioned to clusters, such that the points of

[1] Here, by using the term 'subspace' we generally mean 'affine subspace'.

each cluster are close to a low-dimensional subspace. As mentioned in Chapter 1, this approach, in the analysis of high-dimensional data, is called subspace clustering. Essentially a subspace clustering algorithm can be seen as a local dimensionality reduction method; however, it is also a clustering method, useful in various machine learning tasks.

As mentioned in the previous chapter, TesseraMap requires a clustering of data in which the ranks of clusters[2] are preferably low. Although subspace clustering seems to be a suitable solution for obtaining the desired clusters in TesseraMap, unfortunately, we realized that it has dubious functionality as a clustering solution. When a clustering method is used, one might naturally expect to see a compact and localized set of points in each cluster, however, existing subspace clustering methods do not guarantee such a result. They may produce clusters consisting of disjoint subsets of points. Moreover, since in subspace clustering the underlying manifolds are not considered, it is possible to observe clusters whose points belong to disconnected parts of the manifolds. By considering the role of subspace clustering in dimensionality reduction, however, it is expected that each cluster gathers a subset of points localized in the low-dimensional representation of data.

Clusters consisting of subsets of points from different underlying manifolds can be problematic, for example, in semi-supervised classification. For semi-supervised classification, it is possible to cluster all of the training points, then use the supervised data to label each cluster. If a cluster were to contain disjoint parts of multiple manifolds, its points would most likely have different labels which would, in turn, cause a noticeable error in the final classification. Moreover, in most unsupervised applications, users generally prefer that each cluster represents a compact subset of points from only one category of the data. For example, in visualization, showing disconnected parts or different categories of the data close to each other in one cluster is usually undesirable.

An example of non-localized clustering is depicted in Fig.4.1, where the image samples of two handwritten digits are visualized in three dimensions. The samples of digit 1 form the blue banana-shaped manifold and the red manifold of digit 3 is disc-shaped. In Fig.4.1 (middle) we show a non-localized cluster by green points. Although this cluster is low-rank, it consists of points from both manifolds, which is undesirable. We visualized these points in two dimensions, and show their corresponding images in Fig.4.1 (right). It is clear that the green cluster contains two categories of the data; therefore, it is not suitable for classification. In addition, these two digits are mixed when we visualize them in two-dimensions, while their corresponding manifolds are separated.

[2] We define the rank of a cluster as the maximum number of affinely independent points in the cluster. This definition, in fact, describes the affine rank of the cluster, although we simply refer to it by the term 'rank'. The rank of a cluster can also be seen as the rank of its points after applying centering to them (i.e., subtracting their mean).

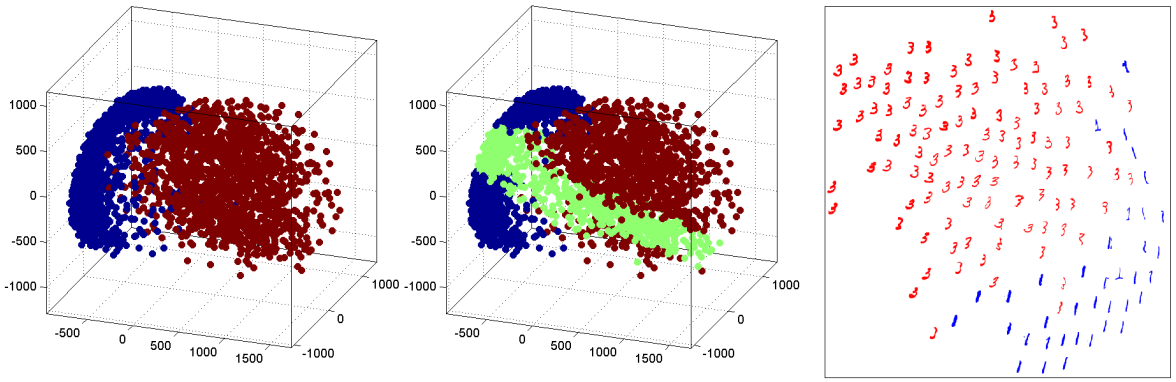


Figure 4.1: (left) The three-dimensional visualization of samples from two handwritten digits, 1 and 3, with the blue and red manifolds respectively; (middle) the two-dimensional subspace of the most similar images of both digits is shown with the green points; (right) the two-dimensional visualization of the images corresponding to the samples of the green subspace, which shows that it consists of two different categories of the data.

We propose LDLC as a solution for this problem. Unlike existing methods, LDLC respects the topology of the underlying manifolds and assigns the data points to low-dimensional subspaces such that each subspace contains a localized subset of the data points on the underlying manifolds. As discussed earlier, this is a valuable property in many pattern recognition tasks, and will be illustrated with some experiments.

In Section 4.1, we first define the notion of the reconstruction error, which is the main objective to be minimized in subspace clustering. Then, we establish a connection between clustering based on the reconstruction error, and $k$-means clustering. We will discuss how minimizing the reconstruction error, as the only objective function, may result in non-localized clusters. We also show how using the geodesic distance instead of the usual Euclidean distance can resolve the problem. Finally, at the end of the next section, we briefly review the most popular subspace clustering methods.

In Section 4.2, we propose our objective function and explain how it can be optimized. Then, an efficient implementation of LDLC is presented and discussed. Finally, we validate our method through various experiments on both synthetic and real datasets, and compare LDLC to the other related methods in Section 4.3.

## 4.1 Subspace Clustering

In this section, we mathematically define the problem of subspace clustering, and show its relation to other relevant methods. Consider an input dataset $\mathbf{X}$, consisting of $n$ points in a $d$-dimensional space. The goal is to partition these points into $c$ clusters such that the rank of each cluster is as close as possible to a target dimensionality $r \ll d$. We assume $c$ (the number of the clusters) and $r$ are both given by the user. The data points are denoted by $d$-dimensional vectors $\{\mathbf{x}_i\}_{i=1}^n$, which are in fact the columns of $\mathbf{X}$. Also, we denote the clusters by $c$ disjoint sets of indices $\{\mathcal{C}_\ell\}_{\ell=1}^c$. That is, if point $\mathbf{x}_i$ belongs to the $\ell^{\text{th}}$ cluster, we simply write $i \in \mathcal{C}_\ell$.

Suppose set $\{\widetilde{\mathbf{x}}_{i,\ell} |\ i \in \mathcal{C}_\ell\}$ represents a rank-$r$ estimation of the points in cluster $\mathcal{C}_\ell$. Since it is of rank $r$, each estimated point can be shown as $\widetilde{\mathbf{x}}_{i,\ell} = \mathbf{U}_\ell \mathbf{y}_{i,\ell} + \mathbf{b}_\ell$, where the $d \times r$ matrix $\mathbf{U}_\ell$ indicates an orthonormal basis (i.e. $\mathbf{U}_\ell^\top \mathbf{U}_\ell = \mathbf{I}_r$), and $\mathbf{b}_\ell \in \mathbb{R}^d$ is a translation vector, and together they indicate an $r$-dimensional affine subspace corresponding to the cluster $\mathcal{C}_\ell$. The $r$-dimensional vector $\mathbf{y}_{i,\ell}$ contains the coordinates of point $\mathbf{x}_i$ with respect to the subspace of cluster $\mathcal{C}_\ell$. We refer to $\mathbf{y}_{i,\ell}$ as *the embedding of point* $\mathbf{x}_i$ *in cluster* $\mathcal{C}_\ell$. The error of estimation $\widetilde{\mathbf{x}}_{i,\ell}$ for this point can be defined as:

$$e_{i,\ell}^2 = \|\mathbf{x}_i - \widetilde{\mathbf{x}}_{i,\ell}\|^2 = \|\mathbf{x}_i - \mathbf{U}_\ell \mathbf{y}_{i,\ell} - \mathbf{b}_\ell\|^2. \tag{4.1}$$

The error of estimation defined in Eq.4.1 is also called the reconstruction error. Similarly, the reconstruction error for the cluster $\mathcal{C}_\ell$ is defined as:

$$e_\ell^2 = \sum_{i \in \mathcal{C}_\ell} e_{i,\ell}^2 = \sum_{i \in \mathcal{C}_\ell} \|\mathbf{x}_i - \mathbf{U}_\ell \mathbf{y}_{i,\ell} - \mathbf{b}_\ell\|^2. \tag{4.2}$$

For rank-$r$ clusters, the estimation is exact, and so the reconstruction error is zero. In fact, the rank of a cluster is close to $r$, if for all of its points, the reconstruction error is small. Following the same approach as PCA [29], we can compute the embedding of the points, the basis, and the translation vector, such that the reconstruction error for the cluster $\mathcal{C}_\ell$ is minimized. First, we find the translation vector $\mathbf{b}_\ell$:

$$\frac{\partial e_\ell^2}{\partial \mathbf{b}_\ell} = \sum_{i \in \mathcal{C}_\ell} 2\mathbf{b}_\ell - 2\left(\mathbf{x}_i - \mathbf{U}_\ell \mathbf{y}_{i,\ell}\right) = 0 \implies \mathbf{b}_\ell = \frac{1}{|\mathcal{C}_\ell|} \sum_{i \in \mathcal{C}_\ell} \mathbf{x}_i - \mathbf{U}_\ell \mathbf{y}_{i,\ell} = \overline{\mathbf{x}}_\ell - \mathbf{U}_\ell \overline{\mathbf{y}}_\ell, \tag{4.3}$$

$$\text{where } \overline{\mathbf{x}}_\ell = \frac{1}{|\mathcal{C}_\ell|} \sum_{i \in \mathcal{C}_\ell} \mathbf{x}_i, \text{ and } \overline{\mathbf{y}}_\ell = \frac{1}{|\mathcal{C}_\ell|} \sum_{i \in \mathcal{C}_\ell} \mathbf{y}_{i,\ell}.$$

Without loss of generality we assume that the embedding of the points in each cluster are centered (i.e. $\overline{\mathbf{y}}_\ell = \mathbf{0}_r$). Consequently, we will have $\mathbf{b}_\ell = \overline{\mathbf{x}}_\ell$. That is, the subspace corresponding to the cluster $\mathcal{C}_\ell$ passes through the mean of its points. To compute the embedding of the points such that the error is minimized, we replace $\mathbf{b}_\ell$ in Eq.4.2. Since $\mathbf{U}_\ell^\top \mathbf{U}_\ell = \mathbf{I}_r$, we will have:

$$\mathbf{y}_{i,\ell} = \mathbf{U}_\ell^\top \left(\mathbf{x}_i - \overline{\mathbf{x}}_\ell\right). \tag{4.4}$$

Note that as expected, the embedded points are centered. Now we combine Eq.4.1 and Eq.4.4, and by replacing $\mathbf{b}_\ell$, we can reformulate the reconstruction error for the point $\mathbf{x}_i$ with respect to the cluster $\mathcal{C}_\ell$ in the following form:

$$\begin{aligned} e_{i,\ell}^2 &= \left\|\mathbf{x}_i - \mathbf{U}_\ell\mathbf{U}_\ell^\top\left(\mathbf{x}_i - \overline{\mathbf{x}}_\ell\right) - \overline{\mathbf{x}}_\ell\right\|^2 = \left\| \left(\mathbf{I}_d - \mathbf{U}_\ell\mathbf{U}_\ell^\top\right)\left(\mathbf{x}_i - \overline{\mathbf{x}}_\ell\right)\right\|^2 \\ &= \left(\mathbf{x}_i - \overline{\mathbf{x}}_\ell\right)^\top \left(\mathbf{I}_d - \mathbf{U}_\ell\mathbf{U}_\ell^\top\right)\left(\mathbf{x}_i - \overline{\mathbf{x}}_\ell\right) \\ &= \left(\mathbf{x}_i - \overline{\mathbf{x}}_\ell\right)^\top \left(\mathbf{x}_i - \overline{\mathbf{x}}_\ell\right) - \left(\mathbf{x}_i - \overline{\mathbf{x}}_\ell\right)^\top \left(\mathbf{U}_\ell\mathbf{U}_\ell^\top\right)\left(\mathbf{x}_i - \overline{\mathbf{x}}_\ell\right) = \left\|\mathbf{x}_i - \overline{\mathbf{x}}_\ell\right\|^2 - \left\|\mathbf{y}_{i,\ell}\right\|^2. \end{aligned} \tag{4.5}$$

To compute $\mathbf{U}_\ell$ such that the reconstruction error is minimized, based on Eq.4.5, we rewrite Eq.4.2 in the following form:

$$\begin{aligned} e_\ell^2 &= \sum_{i\in\mathcal{C}_\ell} \left(\mathbf{x}_i - \overline{\mathbf{x}}_\ell\right)^\top \left(\mathbf{I}_d - \mathbf{U}_\ell\mathbf{U}_\ell^\top\right)\left(\mathbf{x}_i - \overline{\mathbf{x}}_\ell\right) \\ &= \sum_{i\in\mathcal{C}_\ell} \mathbf{Tr}\left\{\left(\mathbf{I}_d - \mathbf{U}_\ell\mathbf{U}_\ell^\top\right)\left(\mathbf{x}_i - \overline{\mathbf{x}}_\ell\right)\left(\mathbf{x}_i - \overline{\mathbf{x}}_\ell\right)^\top\right\} \\ &= \mathbf{Tr}\left\{\left(\mathbf{I}_d - \mathbf{U}_\ell\mathbf{U}_\ell^\top\right)\sum_{i\in\mathcal{C}_\ell}\left(\mathbf{x}_i - \overline{\mathbf{x}}_\ell\right)\left(\mathbf{x}_i - \overline{\mathbf{x}}_\ell\right)^\top\right\} = \mathbf{Tr}\left\{\mathbf{C}_\ell\right\} - \mathbf{Tr}\left\{\mathbf{U}_\ell^\top\mathbf{C}_\ell\mathbf{U}_\ell\right\} \end{aligned} \tag{4.6}$$

where $\mathbf{C}_\ell = \sum_{i\in\mathcal{C}_\ell}\left(\mathbf{x}_i - \overline{\mathbf{x}}_\ell\right)\left(\mathbf{x}_i - \overline{\mathbf{x}}_\ell\right)^\top$.

It is clear that to minimize the error $e_\ell$, the term $\mathbf{Tr}\left\{\mathbf{U}_\ell^\top\mathbf{C}_\ell\mathbf{U}_\ell\right\}$ should be maximized. This optimization can be solved by EVD; that is, the columns of $\mathbf{U}_\ell$ are the eigenvectors of $\mathbf{C}_\ell$, corresponding to its top $r$ eigenvalues. Now, given the cluster $\mathcal{C}_\ell$, we can find its subspace by computing $\mathbf{U}_\ell$ and $\mathbf{b}_\ell$, and then the embedding of each point in its subspace is obtained by Eq.4.4.

In Fig.4.2, an underlying manifold is shown, which contains the cluster $\mathcal{C}_\ell$. Suppose that the target dimensionality $r$ is two. The two-dimensional subspace of this cluster is represented by the center point $\overline{\mathbf{x}}_\ell$, and two orthonormal vectors $\mathbf{u}_{\ell,1}$ and $\mathbf{u}_{\ell,2}$ (the columns of $\mathbf{U}_\ell$). In addition, point $\mathbf{x}_i$ - which does not necessarily belong to the cluster $\mathcal{C}_\ell$ - and its embedding in this cluster, $\mathbf{y}_{i,\ell}$, are shown.

The total reconstruction error is defined by

$$\phi_{rec}\left(\mathcal{C},\mathcal{S}\right)=\sum_{\ell=1}^{c}e_\ell^2=\sum_{\ell=1}^{c}\sum_{i\in\mathcal{C}_\ell}\left\|\mathbf{x}_i-\overline{\mathbf{x}}_\ell\right\|^2-\sum_{\ell=1}^{c}\sum_{i\in\mathcal{C}_\ell}\left\|\mathbf{y}_{i,\ell}\right\|^2. \tag{4.7}$$

The total reconstruction error defined in Eq.4.7 only depends on two parameters:

I- The collection $\mathcal{C}=(\mathcal{C}_1,\mathcal{C}_2,\ldots,\mathcal{C}_c)$ consisting of $c$ cluster sets. This determines which point belongs to which cluster.

II- The collection $\mathcal{S}=((\mathbf{b}_1,\mathbf{U}_1),(\mathbf{b}_2,\mathbf{U}_2),\ldots,(\mathbf{b}_c,\mathbf{U}_c))$ consisting of $c$ pairs of $(\mathbf{b}_\ell,\mathbf{U}_\ell)$ to represent all of the $r$-dimensional subspaces of the clusters.

Therefore, the goal of subspace clustering can be defined as forming subsets of the input data (i.e. $\mathcal{C}$) and computing their corresponding subspaces (i.e. $\mathcal{S}$) such that they produce the minimum possible reconstruction error $\phi_{rec}\left(\mathcal{C},\mathcal{S}\right)$.

### 4.1.1 Connections to $k$-means and VQPCA

As shown in Fig.4.2, three different distances can be defined between the point $\mathbf{x}_i$ and the cluster $\mathcal{C}_\ell$, namely Euclidean distance, in-space distance, and null-space distance. The relation between these three distances, is shown by Eq.4.5 as $\left\|\mathbf{x}_i-\overline{\mathbf{x}}_\ell\right\|^2-\left\|\mathbf{y}_{i,\ell}\right\|^2=e_{i,\ell}^2$. The first term, $\left\|\mathbf{x}_i-\overline{\mathbf{x}}_\ell\right\|$, is the Euclidean distance between $\mathbf{x}_i$ and the mean of the points in the cluster $\mathcal{C}_\ell$, measured in the $d$-dimensional space. The second term, $\left\|\mathbf{y}_{i,\ell}\right\|$, represents the in-space distance, which is also the distance between $\mathbf{x}_i$ and $\overline{\mathbf{x}}_\ell$, but measured in the $r$-dimensional subspace of the cluster $\mathcal{C}_\ell$. Finally, the null-space distance or the reconstruction error $e_{i,\ell}$, is another distance between these two points which is measured in the null-space of $\mathbf{U}_\ell$. There are two essential observations about the total reconstruction error in Eq.4.7:

I- If $r$ is set to zero, Eq.4.4 and consequently the second term in Eq.4.7 will vanish. So, the remaining term of the total reconstruction error, will be the sum of the Euclidean distances between the points and the cluster centers. In this case, $\phi_{rec}\left(\mathcal{C},\mathcal{S}\right)$ in Eq.4.7 represents the objective function of the well-known *k-means* clustering algorithm [40].

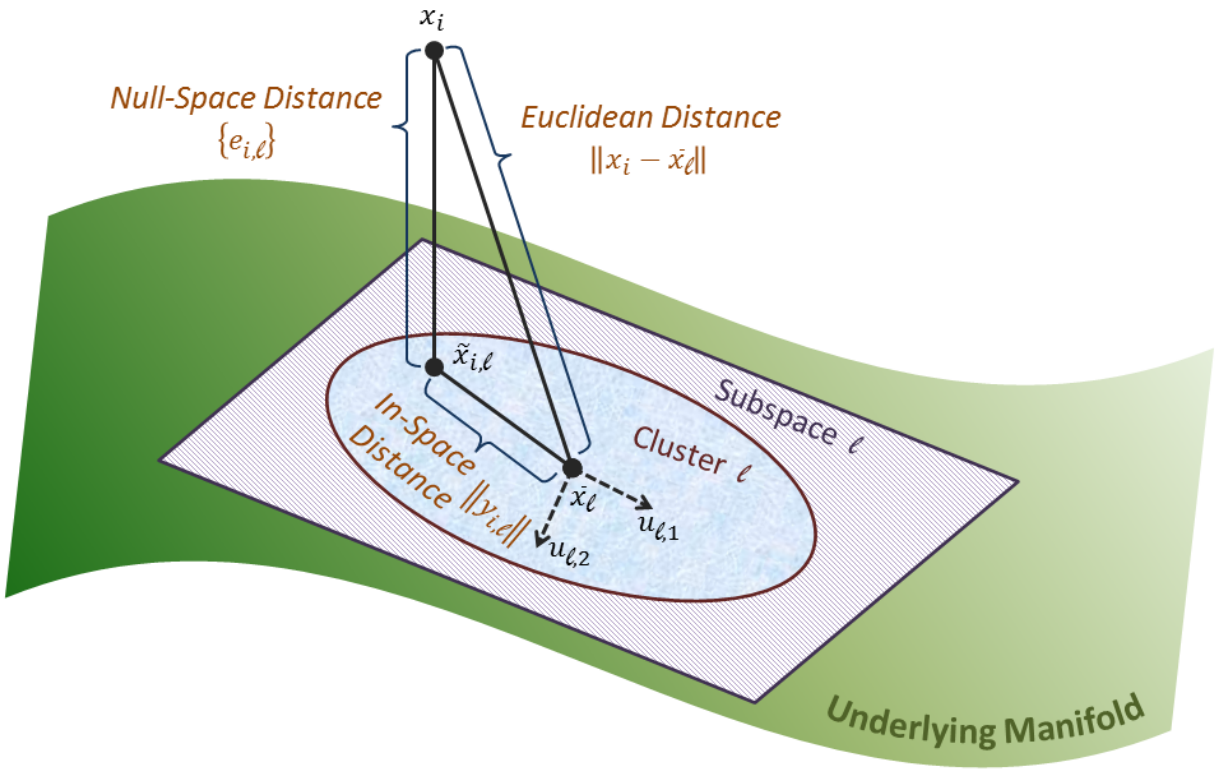


Figure 4.2: A part of a manifold which contains the cluster $\ell$ and its computed subspace. For the arbitrary point $\mathbf{x}_i$, its embedding is shown and the Euclidean, in-space, and null-space distances are denoted.

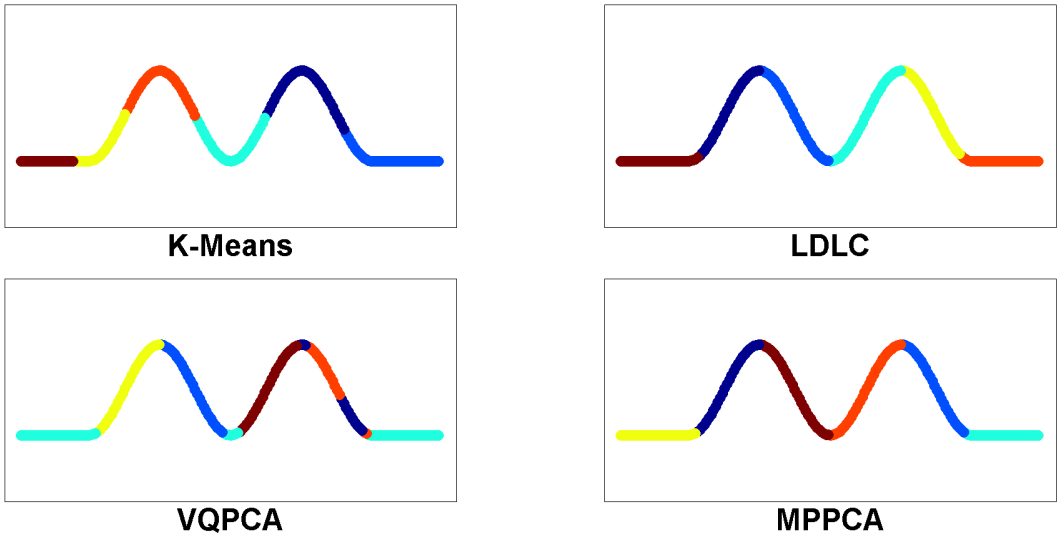


Figure 4.3: The results of different clustering algorithms on a sinusoidal shape toy example. The goal is to partition this two-dimensional dataset into $c = 6$ clusters of rank $r = 1$.

II- For $r > 0$, the reconstruction error, $\phi_{rec}(\mathcal{C}, \mathcal{S})$, will also be the objective function of $k$-means clustering, but now it is done based on the null-space distances instead of the Euclidean distances.

These observations connect the total reconstruction error in Eq.4.7 to the objective function of $k$-means. This resemblance suggests an iterative approach, similar to that of $k$-means (i.e. the Lloyd algorithm [39]), can be utilized to minimize $\phi_{rec}(\mathcal{C}, \mathcal{S})$. That is, in fact, the main idea behind VQPCA [30] for local dimensionality reduction. Unfortunately, both $k$-means and VQPCA suffer from the same shortcoming: their clusters are not necessarily localized, and it is possible that some clusters consist of more than one disjoint part of the underlying manifolds of the data. This is illustrated in Fig.4.3 and Fig.4.4, where two simple two-dimensional datasets are shown, and the goal is to partition the data to clusters of rank one (i.e. $r = 1$). We set $c = 6$ for Fig.4.3, and in Fig.4.4 we have $c = 4$. It is clear that both $k$-means and VQPCA result in clusters consisting of disconnected parts on the underlying manifolds of the datasets.

The main problem is that both of these methods do not consider the topological shape of the manifolds. For $k$-means, however, this can be solved by using geodesic distance in the place of Euclidean distance. This substitution can be seen as mapping the data points into a space in which the pairwise Euclidean distances between the points is the

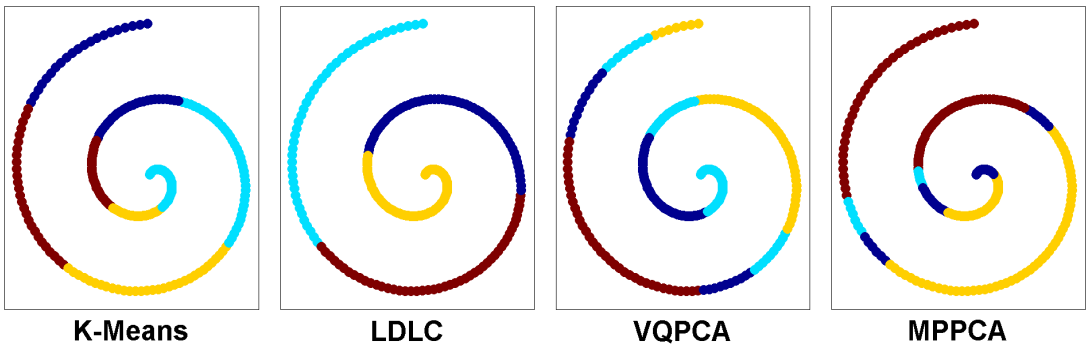


Figure 4.4: The results of different clustering algorithms on a spiral toy example. The goal is to partition this two-dimensional dataset into $c = 4$ clusters of rank close to $r = 1$.

same as their geodesic distances in the original space (e.g., the ideal embedding for Isomap [60]), and then running $k$-means. In this new space, inheriting from $k$-means, each of the resulting clusters covers a convex subset of the manifold, which is localized and compact.

It is important to note that $k$-means computes the mean of each cluster, and so it needs to have the coordinates of the points. But we only have the geodesic distances between them and not their corresponding coordinates on their underlying manifold. Therefore, we first compute the geodesic distances and then employ *$k$-medoids*, which only requires the pairwise distances between the points, instead of $k$-means to find the medoid of each cluster. The objective function of the $k$-medoids algorithm on the geodesic distances is

$$\phi_{geo}\left(\mathcal{C}, \mathcal{M}\right) = \sum_{\ell=1}^{c} \sum_{i \in \mathcal{C}_\ell} d_{geo}^2\left(\mathbf{x}_i, \mathbf{m}_\ell\right), \tag{4.8}$$

where $d_{geo}\left(\mathbf{x}_i, \mathbf{m}_l\right)$ is the geodesic distance between the point $\mathbf{x}_i$ and the cluster medoid $\mathbf{m}_l$. The medoids, which form the set $\mathcal{M} = (\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_c)$, and also the collection $\mathcal{C}$ are the unknown parameters. In this setting, inheriting from $k$-means, each cluster is a connected part on its underlying manifold, yet like $k$-means, this algorithm does not consider the rank of the clusters; consequently the reconstruction error of the clusters can be very high.

There are other methods, like VQPCA, which are able to partition the data such that the rank of each part is minimized. As discussed in the introduction chapter, Mixture of Probabilistic PCA (MPPCA) [62], Mixture of Factor Analyzers (MFA) [26], GPCA, Generalized Principal Component Analysis (GPCA) [67], and $k$-Subspace Clustering [68] are among them. The differences between these methods mostly lie in the models they

assume for the clusters (or the latent variables). They generally attempt to solve non-convex optimizations, to compute the parameters of their models, and to this end, they usually employ iterative approaches, like Expectation Maximization (EM).

Unfortunately, none of these methods exploits the global topology of the data, and therefore, they may produce non-localized clusters. This is illustrated in Fig.4.4 for MP-PCA, where all of the clusters consist of disconnected parts of the manifold. Note that it is claimed in [62] that the clusters produces by MPPCA are more localized compared to the other methods, which is mainly due to the normal distribution model that it assumes for the low-dimensional latent variables.

## 4.2 The Proposed Method

We propose a convex combination of the total reconstruction error in Eq.4.7 and the objective function of the $k$-medoids algorithm applied on the geodesic distances in Eq.4.8 as the desired objective function:

$$\psi\left(\mathcal{C},\mathcal{M},\mathcal{S}\right)=\left(1-\rho\right)\,\phi_{rec}\left(\mathcal{C},\mathcal{S}\right)+\rho\,\phi_{geo}\left(\mathcal{C},\mathcal{M}\right). \tag{4.9}$$

Different values of $\rho\in[0,1]$ combine the properties of VQPCA and $k$-medoids applied to the geodesic distances. When $\rho$ approaches zero, Eq.4.9 behaves more like VQPCA. On the contrary, when $\rho$ is close to one, the result will be more associated with the $k$-medoids clustering. The first term pushes the clusters toward $r$-dimensional affine subspaces, while the second term urges them to create more localized structures on the underlying manifolds. Alternatively this objective function can be interpreted by observing its close connection to $k$-means. First recall that the squared Euclidean distance of a point from the mean of a cluster can be rewritten as a summation of their squared null-space distance and squared in-space distance:

$$\|\mathbf{x}_i-\overline{\mathbf{x}}_\ell\|^2=e_{i,\ell}^2+\|\mathbf{y}_{i,\ell}\|^2. \tag{4.10}$$

In this equation, the first term, $e_{i,\ell}$, emphasizes low reconstruction error, while the second term, $\|\mathbf{y}_{i,\ell}\|$, tries to preserve locality by holding neighboring points together in each subspace. In $k$-means, these two tasks have the same importance. So first, by considering a convex combination of these two terms, we extend $k$-means such that it is possible to choose different weights for these tasks:

$$(1-\rho)\ e_{i,\ell}^2 + \rho\ \|\mathbf{y}_{i,\ell}\|^2. \tag{4.11}$$

The value of $\rho$ moves the focus of clustering between these two tasks. Note that in small scale, the second term, or in-space distance, is equal to geodesic distance. But if the point $\mathbf{x}_i$ is far from the cluster $\mathcal{C}_\ell$, the in-space distance $\|\mathbf{y}_{i,\ell}\|$ can be less than the geodesic distance between the point and the cluster center. Therefore in large scale, this term does not respect the global topology of the manifolds. To address this issue in the second step, we approximate $\|\mathbf{y}_{i,\ell}\|$ in Eq.4.11 by the geodesic distance between the point $\mathbf{x}_i$ and the cluster medoid $\mathbf{m}_\ell$:

$$(1-\rho)\ e_{i,\ell}^2 + \rho\ d_{geo}^2\left(\mathbf{x}_i, \mathbf{m}_\ell\right). \tag{4.12}$$

Although this is a rough approximation, the second term still tries to preserve locality, but instead of the cluster mean, the points are gathered around the cluster medoid $\mathbf{m}_\ell$. Note that the proposed objective function in Eq.4.9 is just the summation of Eq.4.12 over all clusters. This interpretation indicates the close connection between the $k$-means clustering and the proposed method. We call this method Low-dimensional Localized Clustering (LDLC), which is implemented in Alg.4.

In Alg.4, first, $c$ random points are drawn as the initial medoids of the clusters; this will form $\mathcal{M}$. In the second step, we initially cluster the points and form $\mathcal{C}$, based only on the first set of the random medoids. Note that this clustering is done based on the geodesic distances on the underlying manifolds:

$$\mathcal{C}_\ell = \{\ i \mid\ 1 \le i \le n,\ \ell = \arg\min_{\ell'}\ d_{geo}^2\left(\mathbf{x}_i, \mathbf{m}_{\ell'}\right)\}. \tag{4.13}$$

After initialization, the algorithm enters an iterative loop starting at line 3. In each iteration, first we assume that the clusters are fixed, and calculate the subspaces such that they minimize the objective function. This optimization step is fairly simple: in Eq.4.9 if $\mathcal{C}$ is kept fixed, the objective function can be separated into two independent parts, where only the first part relies on the subspace parameters. Therefore, it can be optimized separately based on $\mathcal{S}$:

$$\mathcal{S} \longleftarrow \arg\min_{\mathcal{S}}\ \phi_{rec}\left(\mathcal{C}, \mathcal{S}\right). \tag{4.14}$$

As discussed earlier, PCA is used to compute $\mathbf{b}_\ell$ and $\mathbf{U}_\ell$ for all of the clusters. Therefore, $\mathbf{b}_\ell$ is the mean of the points in the cluster $\mathcal{C}_\ell$, and the columns of $\mathbf{U}_\ell$ are the first $r$

**Algorithm 4** Low-dimensional Localized Clustering

**Input:** $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}, r, c$ **Parameter:** $\rho$

***1*** **-** *Initialize the cluster medoids* $\mathcal{M}$
Assign $c$ random point to the cluster medoids $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_c$

***2*** **-** *Initialize the clusters* $\mathcal{C}$
Assign each point to the cluster with the closest medoid on the manifolds

***3*** **-** *Based on the current clusters, calculate the subspaces*
$\mathcal{S} \longleftarrow \arg\min_{\mathcal{S}} \psi(\mathcal{C}, \mathcal{M}, \mathcal{S})$

***4*** **-** *Based on the current clusters, calculate the medoids*
$\mathcal{M} \longleftarrow \arg\min_{\mathcal{M}} \psi(\mathcal{C}, \mathcal{M}, \mathcal{S})$

***5*** **-** *Based on the current parameters, reassign the points*
$\mathcal{C} \longleftarrow \arg\min_{\mathcal{C}} \psi(\mathcal{C}, \mathcal{M}, \mathcal{S})$

***6*** **-** *If any cluster has been changed go to line 4*

***7*** **-** *Check the objective function*
While there is a noticeable improvement go back to line 3

***8*** **-** *Return the final clusters* $\mathcal{C} = (\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_c)$

eigenvectors of the covariance matrix of the points of $\mathcal{C}_\ell$. Thus, the optimization of line 3 has a closed-form solution. To obtain the medoids in line 4, we again separate the objective function, and this time we use only the second part:

$$\mathcal{M} \longleftarrow \arg\min_{\mathcal{M}} \phi_{geo}(\mathcal{C}, \mathcal{M}), \tag{4.15}$$

which can simply be minimized by a linear search for obtaining the optimal medoids. This search for each individual cluster can be represented by:

$$\mathbf{m}_\ell = \arg \min_{\mathbf{x}_i:\ i \in \mathcal{C}_\ell} \sum_{j \in \mathcal{C}_\ell} d^2_{geo}(\mathbf{x}_j, \mathbf{x}_i). \tag{4.16}$$

In the fifth line, based on the computed parameters (i.e. $\mathcal{S}$ and $\mathcal{M}$), the points are reassigned to the clusters such that the objective function is minimized again. It simply means that we assign each point to the cluster which causes the least error:

$$\mathcal{C}_\ell = \{\, i \mid \; 1 \leq i \leq n,\; \ell = \arg\min_{\ell'} \, (1-\rho)\; e^2_{i,\ell'} + \rho\; d^2_{geo}\,(\mathbf{x}_i, \mathbf{m}_{\ell'})\}. \tag{4.17}$$

In fact, this is the only step in which coefficient $\rho$ is effective; the closer its value to one, the more localized the clusters are. After reassigning the points to the clusters, we check if there exists a point whose cluster label has been changed. As long as such a point exists, the algorithm iterates between lines 4 and 5.

When the clusters become stable, the algorithm proceeds to the seventh line to calculate the objective function based on the latest parameters. Since at each step the parameters are obtained by minimizing the objective function, it is clear that the value of the objective function cannot increase. If the improvement in the objective function is not significant, the algorithm stops, otherwise it returns to the third line and recalculates the subspaces.

It is possible to merge line 3 and 4, and remove line 6. In this case, the algorithm can be implemented with one simple iterative loop, but since computing the eigenvectors at line 3 has the most computational complexity (in order of $\mathsf{O}\left(\frac{n^2 \cdot d}{c}\right)$), it is more efficient to implement the algorithm with two nested loops. This reduces the number of times that EVD is needed, and therefore, reduces the computational cost significantly.

Clearly the objective function is improved at each step, and since it cannot take negative values, it eventually converges. In practice, this convergence happens very fast - however, the algorithm may fall into local minima. To find the global minimum, similar to $k$-means, VQPCA, and all EM methods, the algorithm needs to be run several times with different random initial values and then, the best (minimum) answer should be taken.

Fig.4.3 and Fig.4.4 show the results of the proposed algorithm, applied to two toy examples. Evidently, the clusters are close to one-dimensional subspaces (especially in Fig.4.3 where they are almost line-segments), and also the points in each cluster form a connected component on the manifolds. In other words, LDLC produces low-dimensional localized clusters of data.

## 4.3 Experimental Results

In the first experiment, we studied the effect of $\rho$ in LDLC. For this purpose, we chose a three-dimensional twin-peaks manifold (shown in Fig.4.5), and sampled $n = 2000$ random points uniformly from this manifold. Since this synthetic dataset can be visualized easily, it is possible to verify for which values of $\rho$ the clusters are localized on the manifold. We

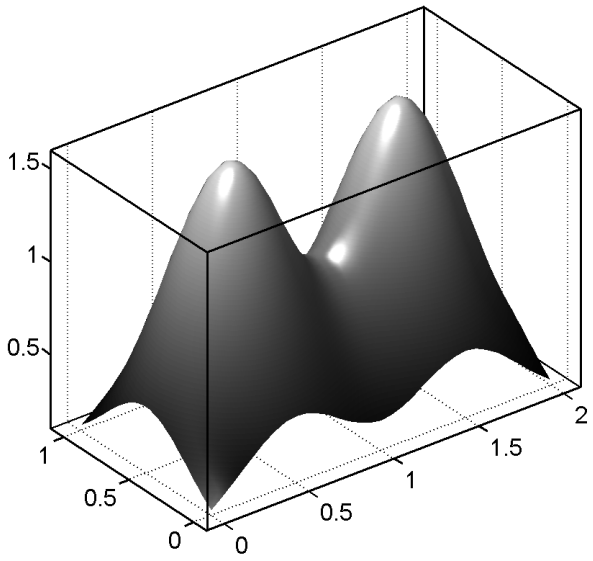


Figure 4.5: The three-dimensional twin-peaks manifold used in the first experiment.

ran LDLC to partition the data points into $c = 6$ clusters. The geodesic distances were calculated based on the $k$-nearest neighbor graph of the data points. It is worth mentioning that LDLC is not sensitive to the choice of $k$, and most of the time there is a wide range of values for $k$ providing reasonable estimates of the geodesic distances, and all of them similarly preserve locality. In this experiment, we observed that values in the range of $4 \leq k \leq 20$ produce similar results, and so we set the number of neighbors $k = 8$.

For each value of $\rho$, we initialized the method with 100 different random seeds of medoids. Each run took less than a minute on one core of a 3.2GHz CPU. As shown in Fig.4.6, for $\rho = 0$ the method behaves similarly to VQPCA, and therefore, despite the lowest reconstruction error, the clusters are not localized. By increasing $\rho$, the clusters become more localized on the manifold. The reconstruction error, however, will grow as $\rho$ increases, as depicted in Fig.4.6 (bottom-right). It is crucial to note that for very small values of $\rho$, it is possible to observe non-localized clusters. In our experiments we observed that the value of $\rho = 0.01$ usually serves the best (i.e. achieves the minimum reconstruction error while the clusters are still localized). The result of MPPCA (again after 100 initializations) is also shown in Fig.4.6. It is clear that MPPCA has one non-localized cluster (indicated by the brown color). This is while LDLC with $\rho = 0.01$ produces completely localized clusters with slightly lower reconstruction error. It is also worth pointing out that MPPCA generally runs slower than LDLC.

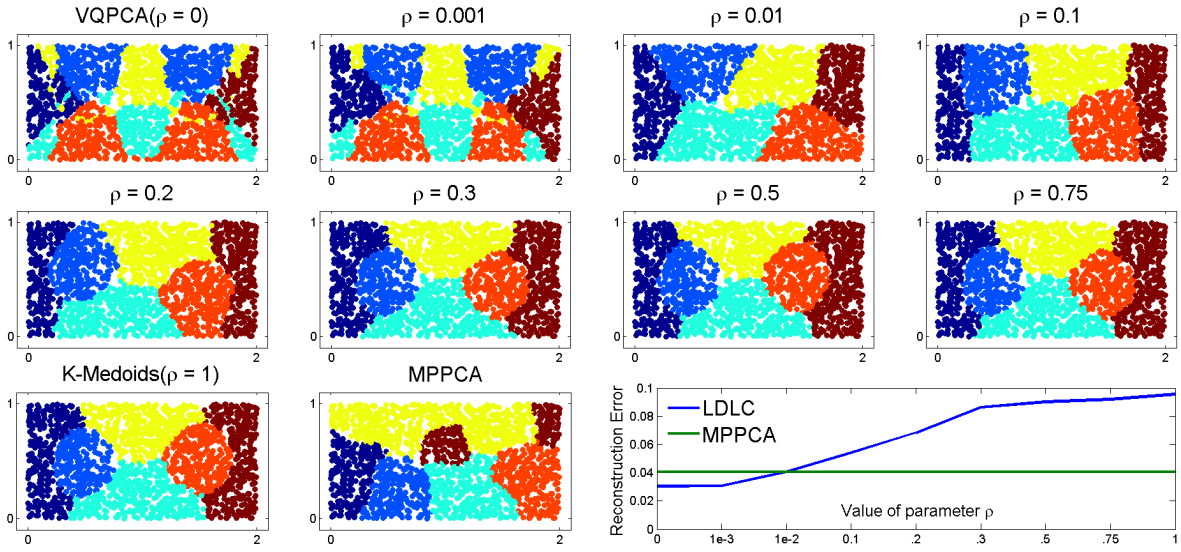


Figure 4.6: The result of clustering by LDLC applied to the twin-peaks manifold for different values of $\rho$. The goal is to return $c = 6$ clusters of rank close to $r = 2$. For the sake of visualization, results are shown from the top view. The result of MPPCA is also shown in the bottom row. The bottom-right plot shows the reconstruction error of MPPCA compared to that LDLC for different values of $\rho$.

### 4.3.1 Olive Oil Dataset - The Effect of $c$

The next experiment was conducted on a real dataset. This set contains the measurements of eight fatty acids in the Italian olive oils [27]. We applied the proposed method with $\rho = 0.01$, and compared it to $k$-means, $k$-medoids applied to the geodesic distances, MPPCA, and VQPCA. In this experiment we studied the relation between the number of clusters, $c$, and the reconstruction error. For each input parameter $c$, we randomly initialized all of the algorithms 100 times to obtain more stable results. The geodesic distances were calculated based on $k = 10$, and the target dimensionality $r$ was set to two.

The scatter plots in Fig.4.7 show the reconstruction errors of each method on all 100 random seeds for the numbers of clusters varying from 1 to 20. In addition, the line plots in this figure indicate the mean of the reconstruction errors of the methods for different numbers of clusters. Clearly, the mean of the error values decreases when the number of clusters increases, and as expected VQPCA has the minimum error in the majority of the cases. Interestingly, the error of LDLC is almost as low as the error of VQPCA. This

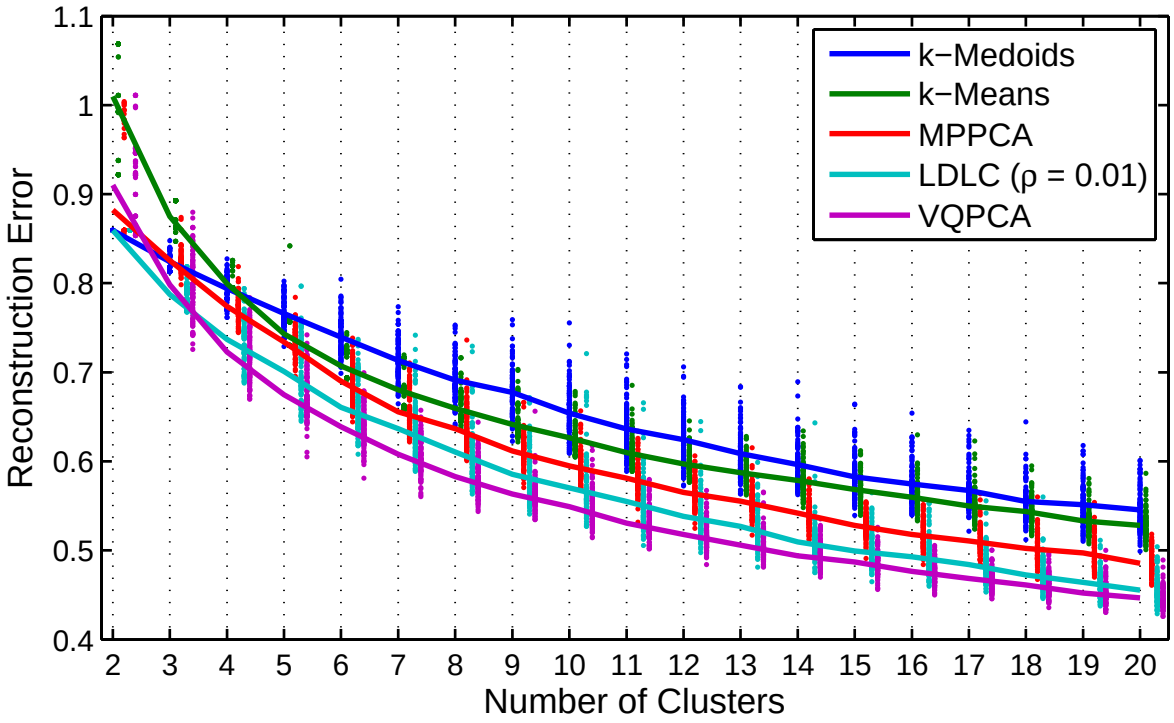


Figure 4.7: Studying the reconstruction error versus the number of clusters, $c$, on the olive oil dataset. Results of five different clustering methods are compared. The target dimensionality is $r = 2$. The scatter plots show the error of all random seeds, while the line plots indicate the mean of the errors.

illustrates that although LDLC has a constraint to enforce the locality of the clusters, its reconstruction error is still low. Another interesting observation in this experiment was that LDLC generally converged with fewer number of iterations than VQPCA, which is due to the nested loops design of LDLC in Alg.4.

### 4.3.2 Wine Quality Dataset - The Effect of $r$

In this experiment, we studied the relation between the target dimensionality $r$ and the reconstruction error. For this purpose another real dataset was used, which consists of 1599 Portuguese red wines, for each of which eleven different chemical characteristics have been measured [15]. Again $\rho$ was set to 0.01, and the comparison was done based on 100

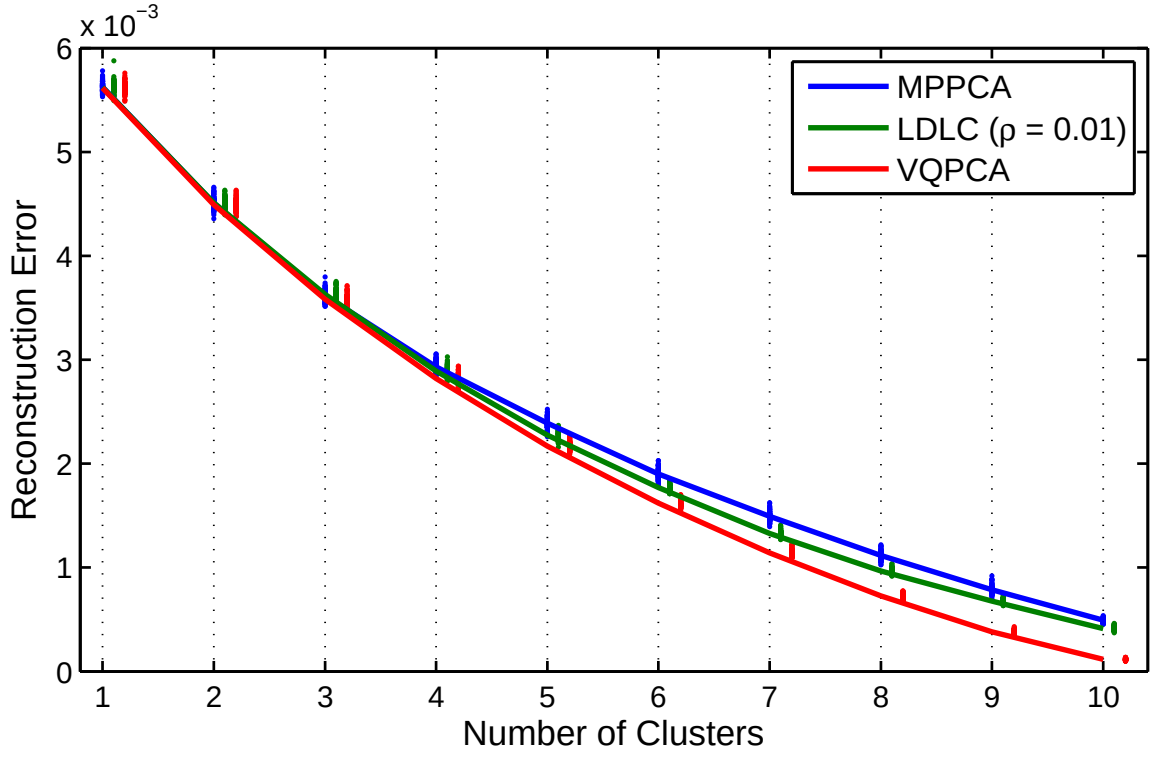


Figure 4.8: Studying the reconstruction error versus the target dimensionality, $r$, on the wine quality dataset. MPPCA, VQPCA and LDLC, are used to partition the points into $c = 10$ clusters. The scatter plots show the error of all random seeds, while the line plots indicate the mean of the errors.

initializations. Due to the large number of samples in this dataset, a wide range of values for $k$ had similar results. We chose $k = 8$ in calculating the geodesic distances. The number of clusters was set to ten. Fig.4.8 shows the reconstruction error of each method for the target dimensionality $r = 1 \dots 10$.

As expected, the errors decrease as the target dimensionality increases. Note that when the target dimensionality is close to the original dimensionality $d = 11$, the performance of VQPCA is superior. In fact, we always expect to see the minimum error for VQPCA. However, when $r$ is small, the search space is more complex, and it is more likely that VQPCA falls into local minima, while LDLC and MPPCA converge relatively faster and to lower errors. This is illustrated in Fig.4.9. As shown, when the target dimensionality is high, LDLC outperforms MPPCA, both in run time and reconstruction error.

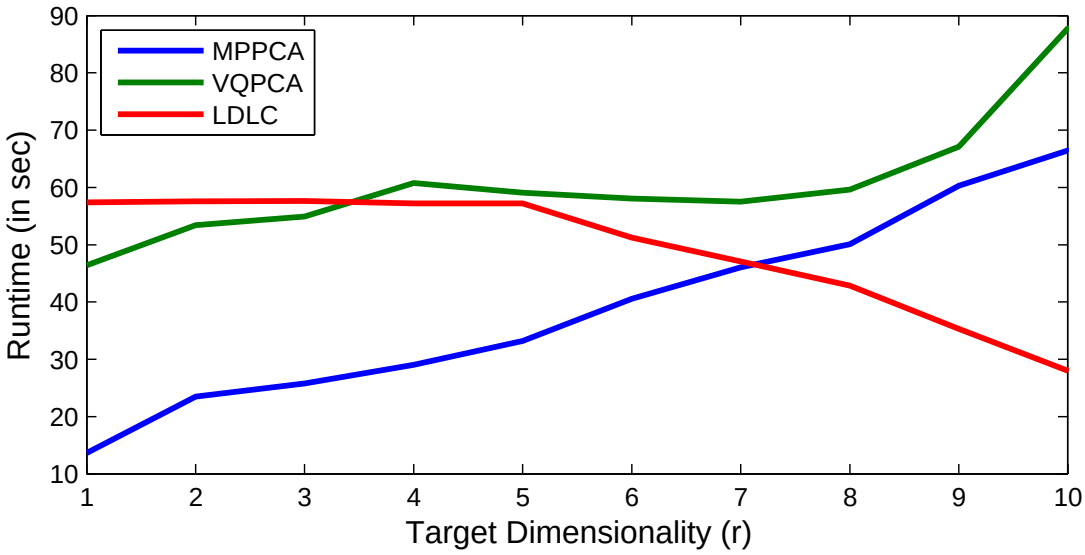


Figure 4.9: Comparing the run time (in seconds) of the different methods versus the target dimensionality on the wine quality dataset. The number of data points is $n = 1599$, and the number of clusters is $c = 10$. We ran each method with 100 random initializations.

### 4.3.3 Wine Type Dataset - Classification

In many cases, for each class of data there exists an underlying manifold from which the samples of that class are taken. Therefore, each class can be approximated by a number of affine subspaces. This enables subspace clustering algorithms to classify the data points in a semi-supervised manner. That is, both labeled and unlabeled samples are used for computing the subspaces. Then the label of each subspace is determined by majority voting on its labeled samples, and consequently the data will be classified. The locality of the clusters on their underlying manifolds, however, is crucial in this task.

In this experiment, we examined data classification as a potential application of LDLC. We show that clusters consisting of disconnected parts of different manifolds may generate a noticeable error in classification. We clustered 178 wine samples which belong to three different cultivars in Italy. This is a real dataset [21] whose samples are labeled vectors of 13 measurements. First, we normalized these measurements, and then we applied LDLC, MPPCA, and VQPCA, each with 100 random seeds. The parameters were $\rho = 0.01, c = 3, r = 2$, and $k = 4$. To evaluate the quality of the clustering results, we have considered two criteria. The first one is *Purity* [41], which is simply the percentage of the points correctly classified (as explained earlier). This measure is mathematically defined as:

$$\text{Purity}(\mathcal{C}, \mathcal{L}) = \frac{1}{n} \sum_{\ell=1}^{c} \max_{g=1}^{c'} |\mathcal{C}_\ell \cap \mathcal{L}_g| \,, \tag{4.18}$$

where collection $\mathcal{L} = \{\mathcal{L}_g\}_{g=1}^{c'}$ indicates the classes of the data, such that $\mathcal{L}_g$ contains the indices of the points of the $g^{\text{th}}$ class. Note that the number of the classes, $c'$, is not necessarily equal to the number of clusters, $c$. The value of purity for bad clusterings is close to zero, while for perfect clustering (i.e. each cluster only consists of points from one class), purity is one. The second measure is *Normalized Mutual Information (NMI)* [41]. This measure is defined as:

$$\text{NMI}(\mathcal{C}, \mathcal{L}) = \frac{2\, I(\mathcal{C}; \mathcal{L})}{H(\mathcal{C}) + H(\mathcal{L})} \tag{4.19}$$

where

$$I(\mathcal{C}; \mathcal{L}) = \sum_{\ell=1}^{c} \sum_{g=1}^{c'} \frac{|\mathcal{C}_\ell \cap \mathcal{L}_g|}{n} \log \left( \frac{n\, |\mathcal{C}_\ell \cap \mathcal{L}_g|}{|\mathcal{L}_g|\ |\mathcal{C}_\ell|} \right), \tag{4.20}$$

is the mutual information between the classes of the data and its clusters. The denominator stands for normalization, and is defined based on entropy:

$$H(\mathcal{C}) = -\sum_{\ell=1}^{c} \frac{|\mathcal{C}_\ell|}{n} \log \left( \frac{|\mathcal{C}_\ell|}{n} \right). \tag{4.21}$$

The value of NMI is normalized between zero and one; if a clustering is randomly done, and does not provide information about the classes, NMI will be zero. Similar to purity, NMI gets its highest value for a perfect clustering. The results of comparisons between LDLC, MPPCA and VQPCA, based on these criteria, are shown in Table 4.1.

As expected, VQPCA has the minimum reconstruction error, however, with the worst classification error (lowest purity). The reconstruction error of LDLC is slightly better than that of MPPCA, however the classification error of MPPCA is much larger than that of LDLC. Moreover, based on NMI it is clear that the clustering done by LDLC provides more information about the classes of the data. Based on this experiment, we can conclude that being localized is an important property for clustering, especially if it is used for classification.

| | LDLC | MPPCA | VQPCA |
|---|---|---|---|
| $\phi_{rec}(\mathcal{C}, \mathcal{S})$ | 2.8722 | 2.8793 | 2.8295 |
| Purity | 0.9045 | 0.8202 | 0.5618 |
| NMI | 0.7222 | 0.6465 | 0.2779 |

Table 4.1: The evaluation of the clustering results for the wine type dataset, and the reconstruction error of LDLC, MPPCA and VQPCA.

| | LDLC | MPPCA | VQPCA |
|---|---|---|---|
| $\phi_{rec}(\mathcal{C}, \mathcal{S})$ | 51.2 | 51.4 | 51.3 |
| Purity | 0.972 | 0.956 | 0.908 |
| NMI | 0.7257 | 0.6898 | 0.6390 |

Table 4.2: The evaluation of the clustering results for the MNIST dataset, and the reconstruction error of LDLC, MPPCA and VQPCA.

### 4.3.4 MNIST Dataset - Image Recognition

As shown in the previous experiment, the proposed method can be used for classification; hence, we applied it to images of MNIST [35] for handwritten digit recognition. For each digit in {0, 3, 6, 9} we drew 750 random $28 \times 28$ image samples, which formed an input dataset of size 3000 and dimensionality 784. The target dimensionality was set to ten (as suggested in [62]), and we partitioned the images into eight clusters (roughly two subspaces for each digit). Note that this time the number of clusters are more than the number of classes. Again the number of initialization seeds was 100 and $\rho = 0.01$. The reconstruction error, purity and NMI of each method are shown in Table 4.2. This time, not only does LDLC outperform the other methods in the quality of clustering (i.e. purity and NMI), but it also has the best reconstruction error. This is evidenced that having localized clusters is an important and useful property in semi-supervised and supervised tasks.

The number of the images of each digit that LDLC has assigned to each of the eight clusters are shown in Fig.4.10. Interestingly, each digit is represented by two localized subspaces. The maximum classification error was 5.94% in the sixth cluster, and the minimum was 1.44% in the second cluster. In fact, there are some similar images of the digit 0 and digit 6, which cause the error in the sixth cluster.

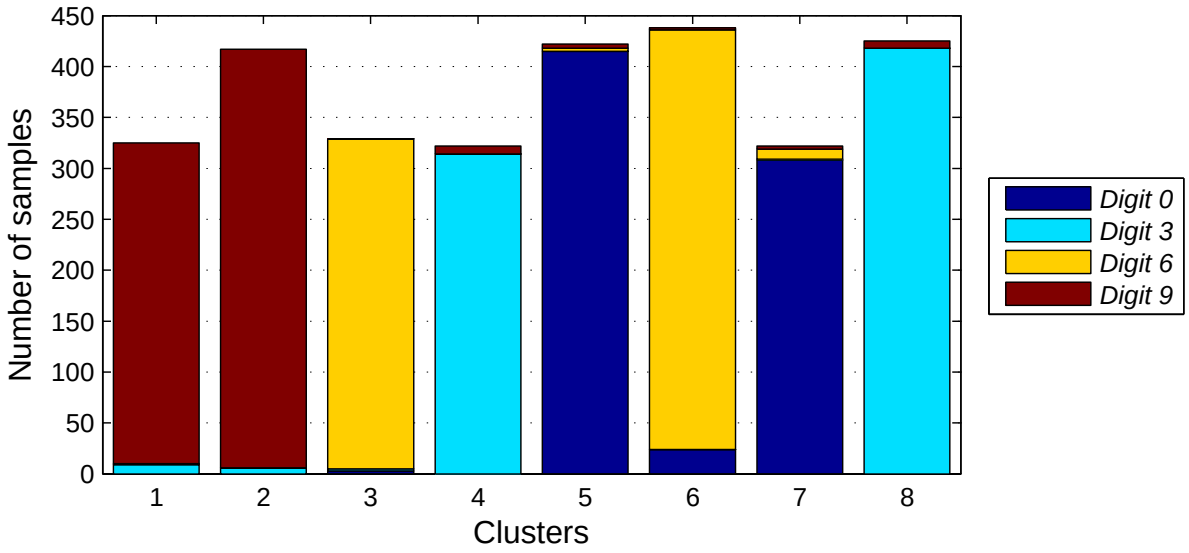


Figure 4.10: The result of digit recognition on the MNIST dataset by the proposed method. The stacked bar plot shows the number of the images from each digit in each of $c = 8$ clusters of rank $r = 10$.

### 4.3.5 Frey Face Images - Locality Checking

In this experiment, we studied the number of non-localized clusters while the total number of clusters $c$ was increasing. We ran this experiment on 500 face images of Brendan Frey. We applied clustering to these $20 \times 28$ images for $c = 2 \dots 10$. In addition, we used three different values for $\rho$ to observe how it affects the locality of the clusters. The target dimensionality is two and $k = 6$. Again a wide range of values for $k$ was acceptable, and produced similar results. After clustering, we applied the *Breadth-First Search (BFS)* algorithm on the connectivity graph defined base on the $k$-nearest neighbors, and counted the number of connected components for each cluster. In this way, we obtained the number of non-localized clusters for each method, which are shown in Table 4.3.

Clearly, for $\rho = 0.1$ the clusters are localized, but for smaller values of $\rho$, non-localized clusters appear for $7 \leq c \leq 9$. In VQPCA there are almost always a number of non-localized clusters. In LDLC, however, the number of non-localized clusters in each partitioning is at most one. Finally, we can see that MPPCA always has more non-localized clusters than LDLC, although it is claimed that this method tends to generate localized clusters [62].

| $c$ | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| MPPCA | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 2 |
| VQPCA | 1 | 2 | 0 | 1 | 2 | 2 | 0 | 5 | 5 |
| $\rho = 0.01$ | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| $\rho = 0.05$ | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| $\rho = 0.10$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Table 4.3: Studying the number of non-localized clusters for different methods applied to the Frey face image dataset.

### 4.3.6 Comparison to APC Used in TesseraMap

As explained in the introduction of this chapter, one motivation to design LDLC was the requirement that TesseraMap has for a suitable clustering method. In the previous chapter we used APC, mainly because it is robust, and does not need the number of clusters as a parameter. Our experiments with TesseraMap showed that APC can fulfill the requirement for clustering, however, it does not necessarily produce low-rank clusters. For this section, we have conducted two experiments, by which we are going to illustrate how LDLC can outperform APC, and why it is a more suitable choice for clustering in TesseraMap.

First, we considered a three-dimensional Swiss-roll, and uniformly sampled $n = 2000$ data points from this manifold. We applied APC to these points, which resulted in 20 clusters. After that, we set $r = 2$ and $c = 20$, and ran LDLC, MPPCA, VQPCA, $k$-means, and $k$-medoids. The average run time (in seconds) for each method is shown in Table 4.4. As expected $k$-medoids and $k$-means have the shortest run time, which is due to their low computational complexity. After these methods, LDLC took the shortest time for clustering. In contrast, APC needed the longest time for converging; it is roughly one hundred times slower than LCLC. The reconstruction errors, $\phi_{rec}(\mathcal{C}, \mathcal{S})$, of each method are also presented in Table 4.4. Since the only goal of VQPCA is to minimize the cluster ranks, it has produced the lowest reconstruction error. After that, LDLC has the minimum reconstruction error, which is smaller than one third of the reconstruction error in APC.

We used the obtained clusters for TesseraMap, to reduce the dimensionality of the data to $r = 2$. The results are shown in Fig. 4.11. The importance of localized clusters is evident in this experiment. Despite their low reconstruction error, both MPPCA and VQPCA failed to produce localized clusters, and therefore, TesseraMap was not able to unfold the Swiss-roll manifold using their clusters.

| | LDLC | APC | $k$-means | $k$-medoids | MPPCA | VQPCA |
|---|---|---|---|---|---|---|
| Run time | 0.5458 | 50.8374 | 0.0535 | **0.0128** | 8.7535 | 1.0328 |
| $\phi_{rec}(\mathcal{C}, \mathcal{S})$ | 0.0168 | 0.0612 | 0.0532 | 0.0680 | 0.0248 | **0.0072** |
| 2D Loss | **0.0001** | 0.0002 | 0.0005 | 0.0007 | 0.0127 | 0.1942 |
| Error | **0.0006** | 0.0043 | 0.0071 | 0.0104 | 0.0133 | 0.5979 |
| Stretch | **7.3329** | 7.0606 | 7.0098 | 6.9612 | 4.0455 | 0.7245 |

Table 4.4: The evaluation of different clustering methods on Swiss-roll.

We applied PCA to the result of TesseraMap for each method, and then calculated the variance of the data captured in the first two dimensions, and normalized it by the total variance. In this way we measured the amount of variance that has been lost to reduce the dimensionality from three to two. These values are indicated by 2D Loss in Table 4.4. The smallest loss belongs to LDLC which is 0.0001 of the total variance. This value confirms that the combination of TesseraMap and LDLC has been successful in unfolding the manifold in two dimensions. In contrast, using VQPCA has caused almost 20% loss of the total variance. In other words, one fifth of the variance has been captured in the third dimension, and so unfolding has failed in this case.

In addition, we measured the normalized preservation error and stretch - which have been defined in the previous chapter - for each clustering method (shown as Error and Stretch respectively in Table 4.4). Note that the measurements were done in the two-dimensional representations. Clearly LDLC has the best results (the lowest Error with the highest Stretch). It is interesting that APC has achieved the second best, which confirms the value of this method for TesseraMap.

A similar experiment was conducted on an MNIST dataset, consisting of $n = 1000$ random images from each digit in $\{3, 6, 9\}$. APC partitioned these images into 19 clusters, and hence, we applied the other methods with the same number of clusters, and this time we set $r = 5$. The average run time of each method is presented in Table 4.5. As expected based on the previous experiment, $k$-medoids is still the fastest. However, this time, due to the high dimensionality of the input data ($d = 784$), the run time of the other methods (except APC) has been significantly increased. Again, after $k$-means and $k$-medoids, LDLC has the minimum run time. Since the labels are available for this dataset, we can evaluate the quality of clustering. Confirming our previous experiments, based on the criteria of Purity and NMI, LDLC has the highest quality in clustering. Regarding the reconstruction error, it is clear from Table 4.5 that although VQPCA has still produced the lowest-rank clusters,

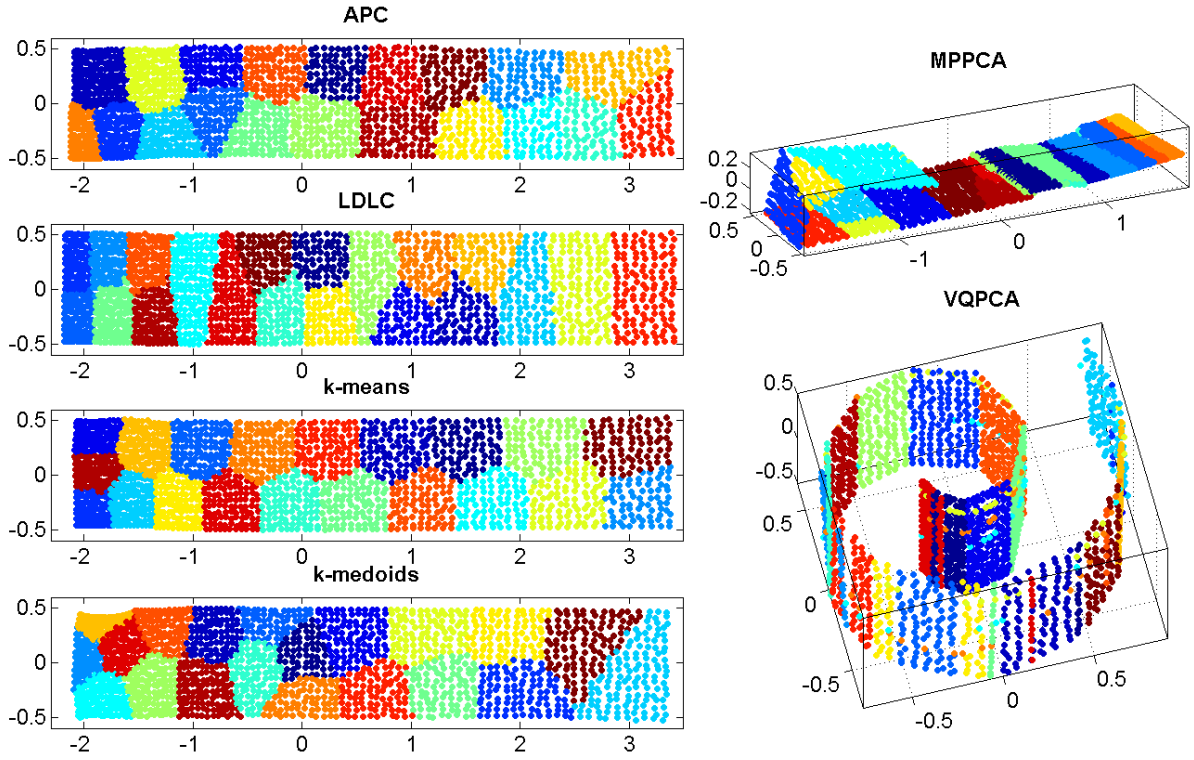


Figure 4.11: The results of TesseraMap on a three-dimensional Swiss-roll manifold, using different clustering methods to partition the data into $c = 20$ clusters.

LDLC has the second lowest reconstruction error, slightly greater than that of VQPCA. We used the obtained clusters for applying TesseraMap. The resulting preservation error, stretch, variance loss, and final rank of embedding are shown in Table 4.5. Variance loss is calculated for $r = 5$ dimensions (i.e. it shows the sum of variance lost in the dimensions higher than five, normalized by the total variance). Evidently, from the final rank of embedding and also the variance loss, LDLC was more successful in aiding TesseraMap in reducing the dimensionality of data. The change of variance loss with respect to the number of dimensions is also depicted in Fig.4.12.

Based on this figure, we can divide the methods into two groups; the first group includes LDLC, VQPCA, and MPPCA, which reduce the dimensionality of data significantly faster than the methods in the second group: APC, $k$-medoids, and $k$-means. This observation was naturally expected; the methods in the first group attempt to produce low-rank clusters, while the methods of the second group do not consider the rank.

| | LDLC | APC | $k$-means | $k$-medoids | MPPCA | VQPCA |
|---|---|---|---|---|---|---|
| Run time | 3.7784 | 19.2738 | 3.3733 | **0.0419** | 11.1600 | 4.4502 |
| Purity | **0.9850** | 0.9577 | 0.9740 | 0.9753 | 0.9547 | 0.9630 |
| NMI | **0.5111** | 0.4677 | 0.4872 | 0.4985 | 0.4964 | 0.4921 |
| $\phi_{rec}(\mathcal{C}, \mathcal{S})$ | 17.4234 | 18.3383 | 17.7434 | 18.4624 | 17.4607 | **17.3032** |
| 5D Loss | **0.0312** | 0.0888 | 0.0501 | 0.1544 | 0.0715 | 0.0652 |
| Rank | **11** | 56 | 30 | 81 | 12 | **11** |
| Error | 0.7726 | 0.8305 | 0.8060 | 0.7778 | 0.7906 | **0.7584** |
| Stretch | **3.4705** | 1.9493 | 2.8490 | 1.1596 | 1.5899 | 1.8359 |

Table 4.5: The evaluation of different clustering methods on MNIST.

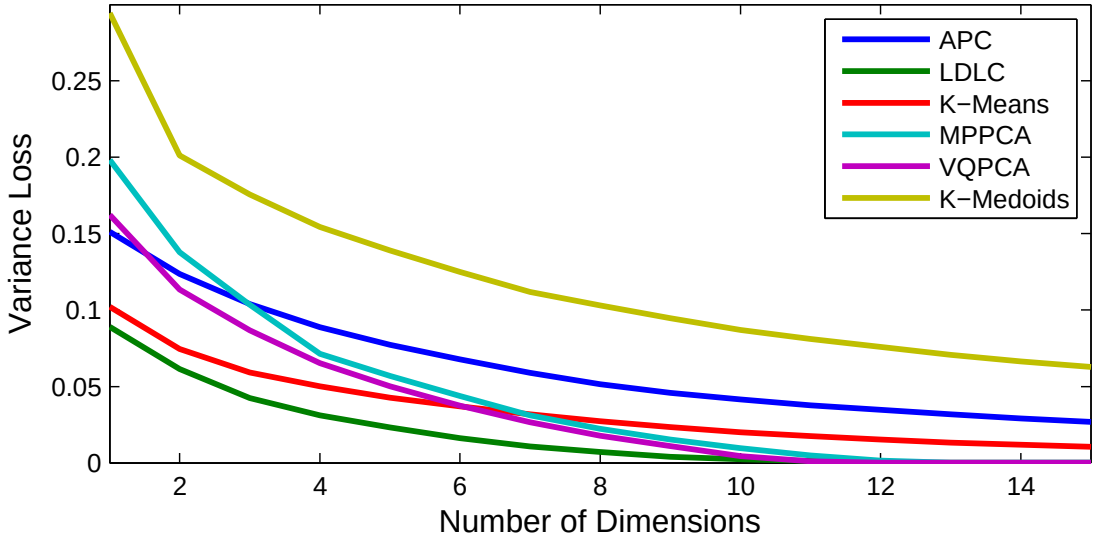


Figure 4.12: Change of variance loss with respect to the number of dimensions for TesseraMap using different clustering methods.

Combining the result of this experiment with that of the previous one on Swiss-roll, we conclude that having the property of being both low-rank and localized is beneficial for the clustering used in TesseraMap. These experiments confirm that LDLC produces clusters with both properties and is, hence, suitable for TesseraMap.

## 4.4 Conclusion and Discussion

In this chapter we suggested a new approach for clustering, proposing a novel method called Low-dimensional Localized Clustering (LDLC). This method clusters the data points into low-rank affine subspaces such that the points in the same cluster are localized on their underlying manifold. Our proposed method belongs to the family of subspace clustering algorithms. This is why we described the importance and applications of subspace clustering in Chapter 1, and briefly reviewed some of the most popular algorithms.

By applying clustering to data, one usually expects to see compact and localized subsets of the data points as the output clusters, although most conventional methods do not guarantee this beneficial property. This is because they do not take into account locality on the manifolds and consequently, the topology of the manifolds will not be preserved. LDLC, however, explicitly enforces this property in clustering by utilizing geodesic distances.

The proposed method exploits $k$-medoids on geodesic distances to respect the topology of the underlying manifolds. To the best of our knowledge, this is the first attempt to combine the concepts of low-rank subspaces and locality on the manifolds. It is important to note that the subspaces are calculated based on the coordinates of the points in the original input space, and the locality of each cluster is defined based on the coordinates of its points on the underlying manifolds. That is why there is no direct way to add this property to the conventional subspace clustering methods.

We showed that LDLC can also be interpreted as an extension of the $k$-means clustering algorithm; that is, the user can set the focus of clustering between producing low-rank clusters and forming localized clusters. In addition, LDLC inherits the simplicity of $k$-means; the objective function of LDLC can be optimized in three straightforward steps. Unfortunately, there is no simple optimization problem when one combines geodesic distances in other methods.

We implemented an algorithm to minimize the objective function of LDLC. Since in each step the objective function is minimized, the algorithm always converges. In addition, due to the nested design of the algorithm, it rapidly converges in few iterations. It has been experimentally demonstrated that the algorithm is also fast in comparison to competing

methods. The most time-consuming step is to compute the first $r$ principal components of each cluster, which takes $\mathsf{O}\left(\frac{n^2 \cdot d}{c}\right)$ in total. Consequently, one run of LDLC for a large dataset (for example, $n \approx 10,000$) will not take more than 10 seconds on a today's conventional personal computer.

There are two parameters used in the proposed method: $\rho$ and $k$. The values used for the parameter $\rho$ are generally low (e.g., $\rho \leq 0.1$), although the values close to zero are risky and may produce non-localized clusters. We explained in the last experiment how it is possible to check whether the clusters are localized. Therefore, in addition to cross-validation, it is possible to add a half-interval search for $\rho$ in the algorithm, such that it always guarantees that each cluster forms a single connected component on the $k$-nearest neighborhood graph of the points. Moreover, based on our experimental results, a very small number of non-localized clusters may exist in the output of LDLC, which can be detected and separated to form new localized clusters on the same subspace.

The second parameter is $k$, which is used to create the $k$-nearest neighborhood graph. Several methods for choosing a proper $k$ for an input dataset exist, and many studies have been done on this subject. Interestingly, LDLC is not sensitive to the choice of $k$, and in the experiments we observed that there is always a wide range of values which provide similar results. This is because we only utilize geodesic distances to make the clusters localized and hence, the accuracy of these distances is not crucial for this task. However, note that the parameter $k$ should be chosen such that it roughly reveals the global topology of the manifolds; therefore, choosing small values (e.g., $k \leq 3$) or large values which connect most of the points to each other (e.g., $k \geq \frac{n}{10}$) might not reflect the shape of the manifolds, so misleading the algorithm.

Finally, our experiments showed that LDLC is able to maintain balance between the reconstruction error and the locality of the output clusters, such that the total reconstruction error is less than or comparable to that of the best methods available, such as VQPCA, while the clusters remain localized. In addition, some applications of LDLC, such as semi-supervised classification, image recognition and naturally, dimensionality reduction, were examined in the experiments.

# Chapter 5

# Dimensionality Reduction by Isometric Patch Alignment (IPA)

In this chapter, we introduce another novel technique for nonlinear dimensionality reduction, called *Isometric Patch Alignment (IPA)*. This method can be seen as an extension to TesseraMap, with scalability being improved. Nevertheless, IPA employs a new idea for unfolding the underlying manifold of data, which greatly enhances its capability in reducing the dimensionality of data, compared not only with TesseraMap, but also with other distance-preserving methods.

Many dimensionality reduction algorithms have been proposed to overcome the curse of dimensionality; the vast majority of these methods, however, suffer from lack of scalability. The computational complexity in most popular methods, for example, Locally Linear Embedding (LLE) [50], Hessian LLE (hLLE) [18], Laplacian Eigenmap (LE) [3], Local Tangent Space Alignment (LTSA) [76], and t-Distributed Stochastic Neighbor Embedding (t-SNE) [65], grows at least with the quadratic order of the input size, which consequently limits the number of data points to $n \sim 50,000$ on a personal computer. This order of complexity is cubic for the traditional methods, i.e. Principal Component Analysis (PCA) [29] and classical Multidimensional Scaling (cMDS) [16], and also for Kernel PCA (KPCA) [53] and for Isomap [61]. It is even higher for more sophisticated methods, such as Maximum Variance Unfolding (MVU) [69], Minimum Volume Embedding (MVE) [55], and Structure Preserving Embedding (SPE) [56], where embedding datasets containing more than 2000 points is almost infeasible. We have previously addressed this shortcoming by proposing TesseraMap, which is at least an order of $\mathsf{O}(n)$ times faster than the other distance preserving methods (i.e. Isomap, MVU, and MVE).

As discussed in Chapter 3, TesseraMap achieves its peak efficiency when the kernel matrix is full-rank; if the kernel matrix is rank-deficient, sparsity in the constraint matrix cannot be exploited, and also the pseudo-inverse of the kernel should be computed. Moreover, the number of between-tesserae constraints should be controlled, as they can potentially increase the computational complexity of solving the SDP.

We propose IPA as a solution to the aforementioned challenges. Similar to TesseraMap, IPA reduces the dimensionality of data by unfolding its underlying manifold while preserving the local distances. In addition, both of these methods provide solutions based on convex optimizations, in situations that many well-known methods cannot handle; for example in the presence of noise or non-uniform samples, or when the underlying manifold of data is non-convex.

Similar to the notion of tesserae in TesseraMap, IPA preserves *patches* of data; we initially partition the input data into preferably low-rank clusters (e.g., by LDLC), and then, for each of them, a low-dimensional representation is separately computed, which is called a patch. In this way, the input data is conceptually reduced to a set of patches. However, there is an essential difference between the patches and the tesserae: in IPA, the neighboring patches overlap each other and thus, they share some of their points, but in TesseraMap all tesserae are disjoint. We mathematically prove that stitching two neighboring patches on an adequate number of boundary points will align them in a low-dimensional subspace. Based on this intuition, stitching all neighboring patches will construct an unfolded instance of the underlying manifold, and consequently places the data points in a subspace with a dimensionality as low as the intrinsic dimensionality of the underlying manifold. In addition to the notion of patch, TesseraMap and IPA are different in two other aspects:

**Scalability:** Both IPA and TesseraMap have a depth-one divide and conquer approach: the large volume of input data is first divided into relatively smaller subsets, which are then embedded separately. This dramatically reduces the total computational cost of embedding. Moreover, the computational complexity of realizing a tessellation in TesseraMap, or of stitching the patches in IPA, is independent of the size of input, which allows them to scale well for the input size. For TesseraMap, this independence is attainable if, first, a full-rank kernel is used, and second, the number of between-tesserae constraints is controlled. The computational complexity of IPA, however, is always independent of the number of samples, allowing it to handle one million data points in only a few minutes on a conventional personal computer.

**Producing low-rank results:** A common practice among various dimensionality reduction methods is to learn a kernel matrix in which a set of desired properties for the final embedding is encoded, and then to apply EVD to this matrix for obtaining a low-dimensional representation of the points [25]. In practice, however, those desired properties will be corrupted by EVD. This is because many methods, such as LLE, LE, hLLE, and Isomap, do not directly consider the target dimensionality in the learning of the kernel matrix and, in fact, only a few methods, such as MVU and MVE, attempt to reduce the final rank explicitly. Considering that the rank function is not convex, numerical search approaches, such as those used in MVE, do not guarantee optimality and would take a long time to return a stable result. The closest convex objective function that approximates the rank of a matrix is its trace [20]; however, in practice, minimizing the trace of a kernel matrix may fold the underlying manifold of data, which is not acceptable in most applications, namely visualization. Therefore, to circumvent the rank optimization problem in dimensionality reduction, the objective function is generally relaxed by other convex functions, such as total variance (as employed by TesseraMap). Relaxing the objective function, however, coincides with counter-examples, e.g., MVU increases the dimensionality of star-shaped datasets. Our solution in IPA is to learn a kernel matrix for rearranging the patches, such that the neighbors can be stitched together. We mathematically prove that our approach produces a low-rank embedding of the underlying manifold, and therefore, applying EVD does not distort it.

Setting aside these two important differences, IPA shares a set of advantages with its predecessor, TesseraMap, compared to the other well-known methods. We briefly explain each of these advantages separately:

**Distance preservation:** The underlying manifold of data can be described by local pairwise distances. IPA is an isometric method, and therefore, it preserves local distances. Distance preservation is a valuable property in many complex tasks where the Euclidean distance between two points is meaningful. For example, unlike IPA, commonly used methods such as LLE, LTSA, SPE, and t-SNE, fail in sensor network applications and also in finding molecular structures.

**Convex optimization:** Unlike many well-known methods, such as the large-scale variations of MVU, Landmark MVU (lMVU)[70] and Fast MVU (fMVU)[71], and also MVE and t-SNE, we implement IPA without any numerical approximation. To find a global embedding, all of the patches are rearranged in a common space, such that the border points of the neighboring patches are matched accordingly for stitching. We show that this

rearrangement can be computed by solving a relatively small semidefinite program, which is convex and hence, always converges rapidly to a unique solution.

**Noise tolerance:** The quality of embedding in most dimensionality reduction methods is highly affected by noisy points, for example, in MVU and Isomap. In IPA, however, the noisy points are embedded in their own patches, and therefore, the effect of noise is diminished in finding the global embedding. This reduces the sensitivity of IPA to the effect of both noise and outliers.

**Handling non-convex manifolds:** If the underlying manifold of data is non-convex, for example, when the manifold contains a hole, the quality of embedding is affected in many popular methods, such as Isomap and MVU[18] [37]. This is mainly due to the global embedding approach in these methods. In contrast, IPA embeds the patches locally and thus, non-convexity of the manifold does not affect its final outcome.

**Low sensitivity to data distribution:** In real world applications the sampling of data is rarely uniform. Non-uniform sampling can create holes in the manifold, which, as mentioned earlier, is undesirable for dimensionality reduction. Most popular methods are sensitive to the distribution of data, even if they do not require a convex manifold (e.g., LTSA, and LLE). A change of distribution over the same manifold may result in different embeddings for many methods, such as PCA, Isomap, LTSA, and LLE. However, IPA considers the local distribution of data separately over each patch, which is more likely to be uniform, and during stitching the global distribution is unseen, thus having no effect.

These beneficial properties make IPA suitable for real-world datasets, especially when the input data is naturally clustered (for example in a protein structure determination task). Our experimental results demonstrate the capabilities of IPA for dimensionality reduction and for data visualization both qualitatively and quantitatively, compared to the other well-known techniques.

The rest of this chapter is organized as follows. First, in Section 5.1 we develop the proposed method, and then we prove its functionality. This section eventually refines IPA to a simple algorithmic representation. After that, in Section 5.2 we will present some experimental results to demonstrate the quality of embedding by IPA on both synthetic and real-world datasets.

## 5.1 The Proposed Method

Analogous to the other proposed methods in this thesis, in IPA we assume that the input high-dimensional data points lie on, or close to, a smooth manifold with a low intrinsic dimensionality. First, the input data points are clustered. Intuitively speaking, clustering divides the underlying manifold of the data into several regions, each of which is represented by the points of one cluster. In the next step, we expand each cluster so that it includes some of the points of its neighboring clusters. In this way, the neighboring regions now overlap around their borders, and therefore, they share some of the data points. Then, for each region we consider a patch, which is, in fact, an isometric embedding of that region in a low-dimensional Euclidean space. Each patch contains a low-dimensional representation of the points of its corresponding cluster. IPA considers a common space for the patches and rearranges them such that the embedded data points which are shared between the neighboring patches, match accordingly, and then it stitches the neighboring patches together. As can be imagined, stitching two patches on their overlapping parts will align them. Thereby, IPA aligns all of the patches, and consequently constructs an unfolded instance of the underlying manifold, which results in reducing the dimensionality of the data. These steps are depicted in Fig. 5.1 for two neighboring patches.

Any simple clustering algorithm such as $k$-means can be applied for partitioning, especially if the manifold is smooth. For example 20,000 data points of the dataset shown in Fig. 5.1 (top-left) are partitioned into 25 clusters using $k$-means. Since it is expected that an isometric embedding of each region forms a low-dimensional patch, it is preferred to have clusters whose ranks are as close as possible to the intrinsic dimensionality of the manifold. Therefore, for complex manifolds, employing subspace clustering algorithms can be beneficial. To have more localized clusters on the underlying manifold of data, with low reconstruction errors, using LDLC is recommended.

As mentioned earlier, to stitch the neighboring patches, IPA requires them to overlap around their boundaries. Sometimes the input data is naturally divided into overlapping clusters (e.g., in the structure of protein), but generally, we need to expand the clusters to be certain that the neighboring clusters have some points in common. Most clustering algorithms assign a membership value to the points with respect to each cluster, which can be used for sharing the points between the neighboring clusters. For example in $k$-means, a point is shared with another cluster if the distance of that point to its cluster center and its distance to the center of the other cluster are approximately equal. This is shown for two clusters in Fig. 5.1 (top-right), where the blue points are the shared points between the red cluster and the green cluster. Similarly, if LDLC is used for partitioning, the Geodesic distance of the points to the medoids can be used to find the shared points. Also,

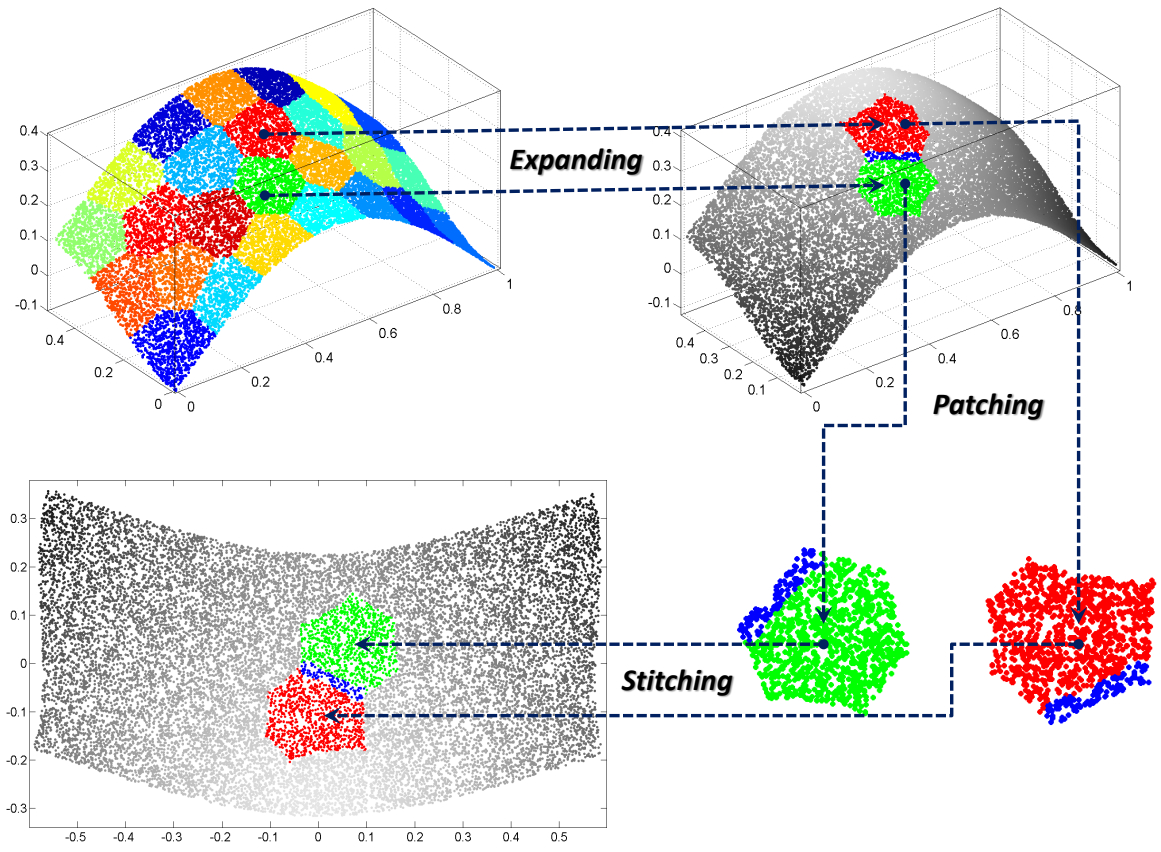


Figure 5.1: The four steps of applying IPA to a simple manifold: the input data is first clustered (top-left), then the clusters are expanded to overlap (top-right), each expanded cluster is separately embedded into a low-dimensional space to form a patch (bottom-right), and finally, IPA rearranges the patches and stitches them together to construct the unfolded manifold (bottom-left).

another approach for expanding a cluster could involve finding the neighboring points of its members (e.g., by computing the $k$-nearest neighbors), and adding them to that cluster.

The rest of this section is as follows. First, we describe how the patches should be formed from the expanded clusters. Then, we mathematically explain how IPA rearranges the patches by solving a convex optimization. Following that, we prove that stitching the neighboring patches on an adequate number of overlapping points will align them. Finally, we present an implementation of IPA as an algorithm.

### 5.1.1 Creating the Patches

Let us begin by introducing the mathematical notations used in IPA. Suppose the input data $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{n}, \mathbf{x}_i \in \mathbb{R}^d$ is partitioned into $c$ preferably low-rank clusters. After expanding, we refer to these clusters as $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_c$. We denote the set of shared points between cluster $\mathcal{C}_i$ and cluster $\mathcal{C}_j$ by $\mathcal{C}_{i\cap j} = \mathcal{C}_i \cap \mathcal{C}_j$, and if this set is not empty, we call them neighbors. For example in Fig. 5.1 (top-right), if we refer to the red cluster as $\mathcal{C}_i$ and to the green cluster as $\mathcal{C}_j$, then $\mathcal{C}_{i\cap j}$ is shown by the blue strip shared between them. Based on this neighborhood notion, we create a connectivity graph $\mathcal{G}$ over $c$ vertices with the edge set of $\mathcal{E}$, such that $(i,j) \in \mathcal{E} \Leftrightarrow \mathcal{C}_{i\cap j} \neq \emptyset$. In IPA we assume that $\mathcal{G}$ is connected; otherwise, IPA should be applied to each of its connected components separately. For each $(i,j) \in \mathcal{E}$, the number of shared points between $\mathcal{C}_i$ and $\mathcal{C}_j$ is indicated by $n_{i,j} = |\mathcal{C}_{i\cap j}|$. Also, for each cluster $\mathcal{C}_i$, the number of its neighboring clusters is indicated by $d_i$.

Since the underlying manifold is smooth, the region of each cluster can be approximated by an $r$-dimensional patch, where $r$ is the target[1] dimensionality. Therefore, in the second step, we map the points of each cluster $\mathcal{C}_i$ into an $r$-dimensional Euclidean space by a distance preserving function $\mathbf{f}_i : \mathbb{R}^d \mapsto \mathbb{R}^r$, to form its corresponding patch[2] $\mathbf{P}_i = \mathbf{f}_i(\mathcal{C}_i)$. If PCA is used for this mapping, $\mathbf{f}_i$ will be an affine transformation. For example, in Fig. 5.1 (bottom-right), the two-dimensional patches of the red cluster and the green cluster are shown, which are computed using PCA. In general, it is possible to employ more sophisticated methods, such as Isomap or MVU for $\mathbf{f}_i$. This, particularly, could be beneficial for large clusters, or for instances when some clusters are not low-rank.

[1] IPA attempts to reduce the dimensionality of input data to $r$. The target dimensionality can be given by a user, or may be set based on the ranks of clusters. For a survey of methods estimating the intrinsic dimensionality, see Chapter 3 of [36].

[2] A patch is a low-dimensional isometric embedding of one region, from the underlying manifold; however, for convenience, we denote the patches by their data points, and therefore, we simply refer to the patches by $\mathbf{P}_i, i \in \{1, 2, \ldots, c\}$.

It should be noted that $\mathbf{P}_i$ is an $r \times |\mathcal{C}_i|$ matrix in which each column represents one point of $\mathcal{C}_i$. Although both $\mathcal{C}_i$ and $\mathbf{P}_i$ refer to the same set of points, the cluster $\mathcal{C}_i$ denotes a subset of the input data points in the $d$-dimensional space, while the patch $\mathbf{P}_i$ contains an $r$-dimensional representation of those points.

For each pair of neighboring clusters $\mathcal{C}_i$ and $\mathcal{C}_j$, their shared points in $\mathcal{C}_{i\cap j}$ are mapped to patch $\mathbf{P}_i$ by $\mathbf{f}_i$, and then separately to patch $\mathbf{P}_j$ by $\mathbf{f}_j$; thus, they may have different coordinates in their patches. We define $\mathbf{P}_{i,j} = \mathbf{f}_i(\mathcal{C}_{i\cap j})$ as an $r \times n_{i,j}$ matrix consisting of the coordinates of the shared points between $\mathcal{C}_i$ and $\mathcal{C}_j$ in the patch $\mathbf{P}_i$, and $\mathbf{P}_{j,i} = \mathbf{f}_j(\mathcal{C}_{i\cap j})$, denoting their coordinates in the patch $\mathbf{P}_j$. Note that $\mathbf{P}_{j,i}$ and $\mathbf{P}_{i,j}$ are generally different matrices, but they have the same size of $r \times n_{i,j}$.

### 5.1.2 Rearranging the Patches

IPA receives low-dimensional patches and, to construct the unfolded manifold, rearranges all of them in a common space, which we call the *unfolding space*, and in that space, stitches the neighbors together. Suppose that the unfolding space is $\mathbb{R}^p$. The dimensionality of the unfolding space, $p$, will be discussed shortly; for now, we assume only that $p \geq r$. First, we mathematically define rearranging:

**Definition 1.** Rearranging patch $\mathbf{P}_i$:
*Transferring $\mathbf{P}_i$ as a solid block to the unfolding space $\mathbb{R}^p$ by an isometry consisting of a $p \times r$ orthonormal matrix $\mathbf{R}_i$ followed by a $p$-dimensional translation vector $\mathbf{t}_i$.*

By rearranging the patch $\mathbf{P}_i$, each data point $\mathbf{x}$ in the cluster $\mathcal{C}_i$ will be mapped to a $p$-dimensional point $\mathbf{y} = \mathbf{R}_i\mathbf{f}_i(\mathbf{x}) + \mathbf{t}_i$. It is important to note that if a data point is shared among two or more patches, it will be mapped to more than one point, and may have different coordinates in the unfolding space. Since rearranging preserves the pairwise distances, the pattern of the data remains intact inside each patch. Ideally, if after rearranging, the overlapping areas of the neighboring patches precisely match, the pattern of the data will be preserved along their borders as well. In that case, each shared point is mapped to some points with the same coordinate in the unfolding space. In contrast, if a shared point has different coordinates in that space, the pattern of the data will be distorted. To evaluate this distortion, we define the following error term:

**Definition 2.** The matching error:
*For two neighboring patches $\mathbf{P}_i$ and $\mathbf{P}_j$, the matching error is defined as follows:*

$$e_{i,j}^2 = \left\| \left( \mathbf{R}_i\mathbf{P}_{i,j} + \mathbf{t}_i\mathbf{1}_{n_{i,j}}^\top \right) - \left( \mathbf{R}_j\mathbf{P}_{j,i} + \mathbf{t}_j\mathbf{1}_{n_{i,j}}^\top \right) \right\|_F^2, \tag{5.1}$$

*where* $\|.\|_F$ *is the Frobenius norm and* $\mathbf{1}_n$ *is a vector consisting of* $n$ *ones. Based on this, the total normalized matching error is defined as:*

$$e^2 = \sum_{(i,j)\in\mathcal{E}} \frac{1}{n_{i,j}} e_{i,j}^2. \tag{5.2}$$

If the shared points precisely match, the matching error will be zero, and we would expect to see no distortion in the pattern of the data. In practice, however, exact match is not possible; therefore, IPA attempts to reduce the distortion by stitching the patches:

**Definition 3.** Stitching:
*Rearranging the patches such that the sum of the matching errors of the neighboring ones is minimized, and then unifying the coordinates of the shared points in the unfolding space, by mapping each point* $\mathbf{x}$*, that is shared among* $m_{\mathbf{x}}$ *patches, to the mean of its coordinates:*

$$\mathbf{y} = \frac{1}{m_{\mathbf{x}}} \sum_{\forall i:\ \mathbf{x}\in\mathcal{C}_i} \mathbf{R}_i \mathbf{f}_i(\mathbf{x}) + \mathbf{t}_i, \tag{5.3}$$

In stitching, we would like to compute $c$ orthonormal matrices $\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_c$, and also $c$ translation vectors $\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_c$ such that the total normalized matching error defined in Eq.5.2 is minimized. Suppose $\mathbf{T} = [\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_c]$ represents an unknown $p \times c$ translation matrix, and $\mathbf{R} = [\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_c]$ to be an unknown $p \times rc$ matrix, whose blocks are the orthonormal $\mathbf{R}_i$ matrices. To facilitate matrix operations on the unknown matrices, we employ selection vectors and matrices. We define the selection vector $\mathbf{e}_i$ such that $\mathbf{t}_i = \mathbf{T}\mathbf{e}_i$. That is, $\mathbf{e}_i$ selects the $i^{th}$ translation vector $\mathbf{t}_i$ from the matrix $\mathbf{T}$. This $c \times 1$ selection vector is the $i^{th}$ column of the identity matrix $\mathbf{I}_c$, whose $i^{th}$ element is one and zero otherwise. By extending the notion of selection, we define the $rc \times r$ selection matrix $\mathbf{E}_i$ to select the matrix $\mathbf{R}_i$ from the unknown matrix $\mathbf{R}$ by: $\mathbf{R}_i = \mathbf{R}\mathbf{E}_i$. This matrix consists of $c$ blocks of size $r \times r$; its $i^{th}$ block is the identity matrix $\mathbf{I}_r$, and the rest are zeros. It can also be seen as putting together the columns $(ri - r + 1), (ri - r + 2), \ldots, (ri)$ of the identity matrix $\mathbf{I}_{rc}$. Using the selection vectors and matrices, we can simplify Eq.5.1 into:

$$
\begin{aligned}
e_{i,j}^2 &= \left\| \left( \mathbf{R}\mathbf{E}_i\mathbf{P}_{i,j} + \mathbf{T}\mathbf{e}_i\mathbf{1}_{n_{i,j}}^\top \right) - \left( \mathbf{R}\mathbf{E}_j\mathbf{P}_{j,i} + \mathbf{T}\mathbf{e}_j\mathbf{1}_{n_{i,j}}^\top \right) \right\|_F^2 \\
&= \left\| \mathbf{R}\left(\mathbf{E}_i\mathbf{P}_{i,j} - \mathbf{E}_j\mathbf{P}_{j,i}\right) + \mathbf{T}\left(\mathbf{e}_i - \mathbf{e}_j\right)\mathbf{1}_{n_{i,j}}^\top \right\|_F^2 \\
&= \mathbf{Tr}\left\{ \left(\mathbf{E}_i\mathbf{P}_{i,j} - \mathbf{E}_j\mathbf{P}_{j,i}\right)^\top \mathbf{R}^\top\mathbf{R}\left(\mathbf{E}_i\mathbf{P}_{i,j} - \mathbf{E}_j\mathbf{P}_{j,i}\right) \right\} \\
&\quad + \mathbf{Tr}\left\{ \mathbf{1}_{n_{i,j}}\left(\mathbf{e}_i - \mathbf{e}_j\right)^\top \mathbf{T}^\top\mathbf{T}\left(\mathbf{e}_i - \mathbf{e}_j\right)\mathbf{1}_{n_{i,j}}^\top \right\} \\
&\quad + 2\ \mathbf{Tr}\left\{ \mathbf{1}_{n_{i,j}}\left(\mathbf{e}_i - \mathbf{e}_j\right)^\top \mathbf{T}^\top\mathbf{R}\left(\mathbf{E}_i\mathbf{P}_{i,j} - \mathbf{E}_j\mathbf{P}_{j,i}\right) \right\},
\end{aligned}
\tag{5.4}
$$

and by substituting $\overline{\mathbf{p}}_{i,j} = \frac{1}{n_{i,j}}\mathbf{P}_{i,j}\mathbf{1}_{n_{i,j}}$ (the mean of $\mathbf{P}_{i,j}$) in Eq.5.4 we have:

$$
\begin{aligned}
e_{i,j}^2 &= \mathbf{Tr}\left\{ \mathbf{R}^\top\mathbf{R}\left(\mathbf{E}_i\mathbf{P}_{i,j} - \mathbf{E}_j\mathbf{P}_{j,i}\right)\left(\mathbf{E}_i\mathbf{P}_{i,j} - \mathbf{E}_j\mathbf{P}_{j,i}\right)^\top \right\} \\
&\quad + n_{i,j}\ \mathbf{Tr}\left\{ \mathbf{T}^\top\mathbf{T}\left(\mathbf{e}_i - \mathbf{e}_j\right)\left(\mathbf{e}_i - \mathbf{e}_j\right)^\top \right\} \\
&\quad + 2\ n_{i,j}\ \mathbf{Tr}\left\{ \mathbf{T}^\top\mathbf{R}\left(\mathbf{E}_i\overline{\mathbf{p}}_{i,j} - \mathbf{E}_j\overline{\mathbf{p}}_{j,i}\right)\left(\mathbf{e}_i - \mathbf{e}_j\right)^\top \right\}.
\end{aligned}
\tag{5.5}
$$

Now we can rewrite the total normalized matching error as:

$$
\begin{aligned}
e^2 &= \mathbf{Tr}\left\{ \mathbf{R}^\top\mathbf{R}\sum_{(i,j)\in\mathcal{E}} \frac{1}{n_{i,j}}\left(\mathbf{E}_i\mathbf{P}_{i,j} - \mathbf{E}_j\mathbf{P}_{j,i}\right)\left(\mathbf{E}_i\mathbf{P}_{i,j} - \mathbf{E}_j\mathbf{P}_{j,i}\right)^\top \right\} \\
&\quad + \mathbf{Tr}\left\{ \mathbf{T}^\top\mathbf{T}\sum_{(i,j)\in\mathcal{E}} \left(\mathbf{e}_i - \mathbf{e}_j\right)\left(\mathbf{e}_i - \mathbf{e}_j\right)^\top \right\} \\
&\quad + 2\ \mathbf{Tr}\left\{ \mathbf{T}^\top\mathbf{R}\sum_{(i,j)\in\mathcal{E}} \left(\mathbf{E}_i\overline{\mathbf{p}}_{i,j} - \mathbf{E}_j\overline{\mathbf{p}}_{j,i}\right)\left(\mathbf{e}_i - \mathbf{e}_j\right)^\top \right\} \\
&= \mathbf{Tr}\left\{\mathbf{R}^\top\mathbf{R}\mathbf{L}_\mathcal{X}\right\} + \mathbf{Tr}\left\{\mathbf{T}^\top\mathbf{T}\mathbf{L}_\mathcal{G}\right\} + 2\ \mathbf{Tr}\left\{\mathbf{T}^\top\mathbf{R}\mathbf{Z}\right\}.
\end{aligned}
\tag{5.6}
$$

The $rc \times rc$ symmetric matrix $\mathbf{L}_\mathcal{X}$, the $c \times c$ symmetric matrix $\mathbf{L}_\mathcal{G}$, and the $rc \times c$ matrix $\mathbf{Z}$ are computed from:

$$
\begin{aligned}
\mathbf{L}_{\mathcal{X}} &= \sum_{(i,j)\in\mathcal{E}} \frac{1}{n_{i,j}} \left(\mathbf{E}_i\mathbf{P}_{i,j} - \mathbf{E}_j\mathbf{P}_{j,i}\right)\left(\mathbf{E}_i\mathbf{P}_{i,j} - \mathbf{E}_j\mathbf{P}_{j,i}\right)^{\top} \\
\mathbf{L}_{\mathcal{G}} &= \sum_{(i,j)\in\mathcal{E}} \left(\mathbf{e}_i - \mathbf{e}_j\right)\left(\mathbf{e}_i - \mathbf{e}_j\right)^{\top} \\
\mathbf{Z} &= \sum_{(i,j)\in\mathcal{E}} \left(\mathbf{E}_i\overline{\mathbf{p}}_{i,j} - \mathbf{E}_j\overline{\mathbf{p}}_{j,i}\right)\left(\mathbf{e}_i - \mathbf{e}_j\right)^{\top}.
\end{aligned}
\tag{5.7}
$$

Note that the positive semidefinite matrix $\mathbf{L}_{\mathcal{G}}$ in Eq.5.7 has the following form:

$$
\mathbf{L}_{\mathcal{G}} = \begin{cases} d_i, & i = j; \\ -1, & (i,j) \in E; \\ 0, & \text{otherwise}, \end{cases}
\tag{5.8}
$$

and is in fact the Laplacian matrix of the connectivity graph $\mathcal{G}$. In addition, from Eq.5.7 it is clear that $\mathbf{L}_{\mathcal{G}}\mathbf{1}_c = \mathbf{0}$. Since $\mathcal{G}$ is connected, only one eigenvalue of its Laplacian is zero, which is related to the direction of $\mathbf{1}_c$. In other words, the rank of $\mathbf{L}_{\mathcal{G}}$ is $(c-1)$, and its null-space is spanned by vector $\mathbf{1}_c$.

Stitching the patches in order to construct the manifold in the unfolding space amounts to minimizing the total matching error in Eq.5.6. First, we partially optimize the error by taking its derivative with respect to $\mathbf{T}$ and setting it to zero:

$$
\frac{\partial e^2}{\partial \mathbf{T}} = 2\ \mathbf{T}\mathbf{L}_{\mathcal{G}} + 2\ \mathbf{R}\mathbf{Z} = \mathbf{0},
\tag{5.9}
$$

and we obtain $\mathbf{T}\mathbf{L}_{\mathcal{G}} = -\mathbf{R}\mathbf{Z}$. Note that $\mathbf{L}_{\mathcal{G}} \succeq \mathbf{0}$, and Eq.5.6 is a convex quadratic function of $\mathbf{T}$. Therefore, the resulting critical point in Eq.5.9 is indeed a global minimum. Adding a constant vector to all of the translation vectors in $\mathbf{T}$ does not alter the amount of error. To remove this extra degree of freedom in the optimization, we restrict the translation vectors by centering them with $\mathbf{T}\mathbf{1}_c = \mathbf{0}$. Therefore, we will have:

$$
\begin{aligned}
&\mathbf{T}\left[\mathbf{L}_{\mathcal{G}}\ \mathbf{1}_c\right] = -\left[\mathbf{R}\mathbf{Z}\ \mathbf{0}\right] \quad\Longrightarrow\quad \mathbf{T}\left[\mathbf{L}_{\mathcal{G}}\ \mathbf{1}_c\right]\begin{bmatrix}\mathbf{L}_{\mathcal{G}}^{\top} \\ \mathbf{1}_c^{\top}\end{bmatrix} = -\left[\mathbf{R}\mathbf{Z}\ \mathbf{0}\right]\begin{bmatrix}\mathbf{L}_{\mathcal{G}}^{\top} \\ \mathbf{1}_c^{\top}\end{bmatrix} \\
&\Longrightarrow\quad \mathbf{T}\left(\mathbf{L}_{\mathcal{G}}\mathbf{L}_{\mathcal{G}}^{\top} + \mathbf{1}_c\mathbf{1}_c^{\top}\right) = -\mathbf{R}\mathbf{Z}\mathbf{L}_{\mathcal{G}}^{\top} \quad\Longrightarrow\quad \mathbf{T} = -\mathbf{R}\mathbf{Z}\mathbf{L}_{\mathcal{G}}^{\dagger}
\end{aligned}
\tag{5.10}
$$

where $\mathbf{L}_{\mathcal{G}}^{\dagger} = \mathbf{L}_{\mathcal{G}}^{\top}\left(\mathbf{L}_{\mathcal{G}}\mathbf{L}_{\mathcal{G}}^{\top} + \mathbf{1}_c\mathbf{1}_c^{\top}\right)^{-1}$ is the pseudo-inverse of $\mathbf{L}_{\mathcal{G}}$. Note that $\mathbf{L}_{\mathcal{G}}$ is not full-rank, but since its null-space is spanned only by $\mathbf{1}_c$, the summation of $\mathbf{L}_{\mathcal{G}}\mathbf{L}_{\mathcal{G}}^{\top} + \mathbf{1}_c\mathbf{1}_c^{\top}$ becomes full-rank, and consequently, the inverse always exists.

From Eq.5.10 it is clear that $\mathbf{T}$ and $\mathbf{R}$ are linearly related. This leaves us to find the unknown matrix $\mathbf{R}$. By substituting $\mathbf{T}$ in the total matching error of Eq.5.6 we will have:

$$\begin{aligned} e^2 &= \mathbf{Tr}\left\{\mathbf{R}^{\top}\mathbf{R}\mathbf{L}_{\mathcal{X}}\right\} + \mathbf{Tr}\left\{\mathbf{L}_{\mathcal{G}}^{\dagger}\mathbf{Z}^{\top}\mathbf{R}^{\top}\mathbf{R}\mathbf{Z}\mathbf{L}_{\mathcal{G}}^{\dagger}\mathbf{L}_{\mathcal{G}}\right\} - 2\ \mathbf{Tr}\left\{\mathbf{L}_{\mathcal{G}}^{\dagger}\mathbf{Z}^{\top}\mathbf{R}^{\top}\mathbf{R}\mathbf{Z}\right\} \\ &= \mathbf{Tr}\left\{\mathbf{R}^{\top}\mathbf{R}\mathbf{L}_{\mathcal{X}}\right\} + \mathbf{Tr}\left\{\mathbf{R}^{\top}\mathbf{R}\mathbf{Z}\mathbf{L}_{\mathcal{G}}^{\dagger}\mathbf{L}_{\mathcal{G}}\mathbf{L}_{\mathcal{G}}^{\dagger}\mathbf{Z}^{\top}\right\} - 2\ \mathbf{Tr}\left\{\mathbf{R}^{\top}\mathbf{R}\mathbf{Z}\mathbf{L}_{\mathcal{G}}^{\dagger}\mathbf{Z}^{\top}\right\} \\ &= \mathbf{Tr}\left\{\mathbf{R}^{\top}\mathbf{R}\mathbf{L}_{\mathcal{X}}\right\} - \mathbf{Tr}\left\{\mathbf{R}^{\top}\mathbf{R}\mathbf{Z}\mathbf{L}_{\mathcal{G}}^{\dagger}\mathbf{Z}^{\top}\right\} \\ &= \mathbf{Tr}\left\{\mathbf{R}^{\top}\mathbf{R}\left(\mathbf{L}_{\mathcal{X}} - \mathbf{Z}\mathbf{L}_{\mathcal{G}}^{\dagger}\mathbf{Z}^{\top}\right)\right\} = \mathbf{Tr}\left\{\mathbf{R}^{\top}\mathbf{R}\mathbf{K}\right\} = \mathbf{Tr}\left\{\mathbf{A}\mathbf{K}\right\}, \end{aligned} \tag{5.11}$$

where we used the fact that $\mathbf{L}_{\mathcal{G}}^{\dagger} = \mathbf{L}_{\mathcal{G}}^{\dagger}\mathbf{L}_{\mathcal{G}}\mathbf{L}_{\mathcal{G}}^{\dagger}$. In addition, we have the $rc \times rc$ positive semidefinite[3] matrix $\mathbf{K} = \mathbf{L}_{\mathcal{X}} - \mathbf{Z}\mathbf{L}_{\mathcal{G}}^{\dagger}\mathbf{Z}^{\top}$, which is calculated for the input data.

In Eq.5.11, we introduced an unknown $rc \times rc$ positive semidefinite matrix $\mathbf{A} = \mathbf{R}^{\top}\mathbf{R}$. Suppose $\mathbf{A}$ is partitioned into $c$ by $c$ blocks of $r \times r$ matrices, such that the block $(i, j)$ contains $\mathbf{R}_i^{\top}\mathbf{R}_j$. In other words, we have $\mathbf{E}_i^{\top}\mathbf{A}\mathbf{E}_j = \mathbf{R}_i^{\top}\mathbf{R}_j$ (remember that $\mathbf{R}_i = \mathbf{R}\mathbf{E}_i$). To impose the orthonomality constraint on $\mathbf{R}_i$, it suffices to set the block $(i, i)$ to the identity matrix $\mathbf{I}_r$, or equivalently $\mathbf{E}_i^{\top}\mathbf{A}\mathbf{E}_i = \mathbf{I}_r$. That is, $\mathbf{A}$ has $c$ blocks of $r \times r$ identity matrices on its main diagonal to guarantee that each $\mathbf{R}_i$ is an orthonormal matrix. Note that there is no constraint on the non-diagonal blocks of $\mathbf{A}$. To calculate the isometry maps that minimize the total normalized matching error, one can solve the following standard semidefinite programming problem:

$$\min_{\mathbf{A}\succeq\mathbf{0}}\ \mathbf{Tr}\left\{\mathbf{A}\mathbf{K}\right\}\ \ s.t.\quad \mathbf{A} = \begin{pmatrix} \mathbf{I}_r & & & ? \\ & \mathbf{I}_r & & \\ & & \ddots & \\ ? & & & \mathbf{I}_r \end{pmatrix} \tag{5.12}$$

[3] From Eq.5.4 we can derive that $e_{i,j}^2 = \|\mathbf{R}\mathbf{B}_{i,j}\|_F^2$, where $\mathbf{B}_{i,j} = \mathbf{E}_i\mathbf{P}_{i,j} - \mathbf{E}_j\mathbf{P}_{j,i} - \mathbf{Z}\mathbf{L}_{\mathcal{G}}^{\dagger}\left(\mathbf{e}_i - \mathbf{e}_j\right)\mathbf{1}_{n_{i,j}}^{\top}$. Therefore, we can rewrite the total normalized error as $e^2 = \sum_{(i,j)\in\mathcal{E}}\frac{1}{n_{i,j}}\mathbf{Tr}\left\{\mathbf{A}\mathbf{B}_{i,j}\mathbf{B}_{i,j}^{\top}\right\}$. Eventually we will have $\mathbf{K} = \sum_{(i,j)\in\mathcal{E}}\frac{1}{n_{i,j}}\mathbf{B}_{i,j}\mathbf{B}_{i,j}^{\top}$, which is a sum of positive semidefinite matrices, and so it should be positive semidefinite as well.

After solving this SDP optimization, the matrix of mappings, $\mathbf{R}$, can be calculated by decomposing $\mathbf{A}$. Since $\mathbf{A}$ is $rc \times rc$, the resulting $\mathbf{R}$ will have $rc$ rows and columns. This means that in order to achieve the minimum error in rearranging the patches, they are mapped to an $rc$-dimensional space; therefore setting $p = rc$ provides enough degrees of freedom to construct the unfolded manifold. This is expected because each patch uses $r$ orthonormal vectors as its basis, thus the maximum possible affine rank of the configuration of all $c$ patches will be $rc$ (considering that $\mathbf{T}$ is linearly related to $\mathbf{R}$, the translations do not increase the rank).

One might ask why IPA does not construct the unfolded manifold in an $r$-dimensional space. It is important to note that in many cases, $r$ dimensions do not provide enough degrees of freedom for rearranging the patches, such that the topology of the underlying manifold is preserved in the constructed manifold. For example, a circle can be approximated by many one-dimensional patches; however, rearranging these patches in a one-dimensional space results in a line segment, which does not have the topological structure of the circle. Therefore, at least two dimensions are needed for a proper rearrangement in this example. This is why IPA might need more than $r$ dimensions in the unfolding space; however, as discussed, $p = rc$ is always enough. After stitching, Eq.5.3 can be used to obtain the coordinates of any point $\mathbf{x}$, linearly based on $\mathbf{R}$:

$$\begin{aligned}\mathbf{y} &= \frac{1}{m_{\mathbf{x}}} \sum_{\forall i:\ \mathbf{x}\in\mathcal{C}_i} \mathbf{R}_i\mathbf{f}_i(\mathbf{x}) + \mathbf{t}_i = \frac{1}{m_{\mathbf{x}}} \sum_{\forall i:\ \mathbf{x}\in\mathcal{C}_i} \mathbf{R}\mathbf{E}_i\mathbf{f}_i(\mathbf{x}) + \mathbf{T}\mathbf{e}_i \\ &= \frac{\mathbf{R}}{m_{\mathbf{x}}} \sum_{\forall i:\ \mathbf{x}\in\mathcal{C}_i} \mathbf{E}_i\mathbf{f}_i(\mathbf{x}) - \mathbf{Z}\mathbf{L}_{\mathcal{G}}^{\dagger}\mathbf{e}_i = \mathbf{R}\,\mathbf{f}(\mathbf{x}),\end{aligned} \tag{5.13}$$

or in the matrix form

$$\mathbf{Y} = \mathbf{R}\,\mathbf{F}(\mathbf{X}), \tag{5.14}$$

where $\mathbf{f}(\mathbf{x}) = \frac{1}{m_{\mathbf{x}}} \sum_{\forall i:\ \mathbf{x}\in\mathcal{C}_i} \mathbf{E}_i\mathbf{f}_i(\mathbf{x}) - \mathbf{Z}\mathbf{L}_{\mathcal{G}}^{\dagger}\mathbf{e}_i$ and so $\mathbf{F}(\mathbf{X}) = [\mathbf{f}(\mathbf{x}_1), \ldots, \mathbf{f}(\mathbf{x}_n)]$ can be calculated for all of the $n$ input data points. In Eq.5.14, $\mathbf{Y}$, which contains the coordinates of the points after unfolding, is $rc$-dimensional. Therefore, if we want to represent the data into a lower-dimensional space, we should apply PCA at the end.

### 5.1.3 Patch Alignment by Stitching

In this part, we explain why the rank of the embedded data points on the constructed manifold (i.e. $\mathbf{Y}$) is much lower than their dimensionality, $rc$, and is, in fact, close to the target dimensionality, $r$. We proceed by considering the effect of stitching on two neighboring patches.

By attaching two patches, one may form a new object whose rank is higher than the dimensionality of both patches. For example, two one-dimensional patches (i.e. line segments), which are attached on a single shared point, can form a V-shaped (or more precisely, an X-shaped) object which is obviously of rank two. Similarly, two attached planar patches can form an object of rank four, three, or two. In general, attaching two rank-$r$ patches together, forms an object whose rank is at least $r$ and at most $2r$. However, the following theorem proves that under a certain condition, stitching aligns the patches, such that the overall rank does not increase.

**Theorem 1.** *If the cross-covariance matrix of the shared points between two $r$-dimensional patches is full-rank, stitching them in an arbitrarily higher-dimensional space aligns them in an $r$-dimensional subspace.*

*Proof.* Consider two neighboring $r$-dimensional patches $\mathbf{P}_i$ and $\mathbf{P}_j$. To stitch them, we should minimize their matching error:

$$e_{i,j}^2 = \left\| \mathbf{R}_i \mathbf{P}_{i,j} - \mathbf{R}_j \mathbf{P}_{j,i} + (\mathbf{t}_i - \mathbf{t}_j) \mathbf{1}_{n_{i,j}}^\top \right\|_F^2 . \tag{5.15}$$

First, we compute $(\mathbf{t}_i - \mathbf{t}_j)$ by taking the derivative[4] and setting it to zero:

$$\frac{\partial e_{i,j}^2}{\partial (\mathbf{t}_i - \mathbf{t}_j)} = 0 \quad \Rightarrow \quad \mathbf{t}_i - \mathbf{t}_j = \mathbf{R}_j \overline{\mathbf{p}}_{j,i} - \mathbf{R}_i \overline{\mathbf{p}}_{i,j}. \tag{5.16}$$

Now, by substituting Eq.5.16 in Eq.5.15 we have:

$$\begin{aligned} e_{i,j}^2 =& \left\| \mathbf{R}_i \mathbf{P}_{i,j} - \mathbf{R}_j \mathbf{P}_{j,i} + \left( \mathbf{R}_j \overline{\mathbf{p}}_{j,i} - \mathbf{R}_i \overline{\mathbf{p}}_{i,j} \right) \mathbf{1}_{n_{i,j}}^\top \right\|_F^2 \\ =& \left\| \mathbf{R}_i \overline{\mathbf{P}}_{i,j} - \mathbf{R}_j \overline{\mathbf{P}}_{j,i} \right\|_F^2 \\ =& \mathbf{Tr} \left\{ \overline{\mathbf{P}}_{i,j}^\top \overline{\mathbf{P}}_{i,j} + \overline{\mathbf{P}}_{j,i}^\top \overline{\mathbf{P}}_{j,i} \right\} - 2 \, \mathbf{Tr} \left\{ \overline{\mathbf{P}}_{i,j}^\top \mathbf{R}_i^\top \mathbf{R}_j \overline{\mathbf{P}}_{j,i} \right\}, \end{aligned} \tag{5.17}$$

[4] Note that $e_{i,j}^2$ is a convex quadratic function of $(\mathbf{t}_i - \mathbf{t}_j)$, so the critical point is a global minimum.

where $\overline{\mathbf{P}}_{i,j} = \mathbf{P}_{i,j}\left(\mathbf{I}_{n_{i,j}} - \frac{1}{n_{i,j}}\mathbf{1}_{n_{i,j}}\mathbf{1}_{n_{i,j}}^{\top}\right)$ is the centered version of the shared points between the two patches in $\mathbf{P}_i$. Since the first term is constant, to minimize the error it is only necessary to maximize the second term, which can be rewritten as:

$$\begin{aligned}\mathbf{Tr}\left\{\overline{\mathbf{P}}_{i,j}^{\top}\mathbf{R}_i^{\top}\mathbf{R}_j\overline{\mathbf{P}}_{j,i}\right\} &= \mathbf{Tr}\left\{\mathbf{R}_i^{\top}\mathbf{R}_j\mathbf{M}\right\} = \mathbf{Tr}\left\{\mathbf{R}_i^{\top}\mathbf{R}_j\mathbf{U}\mathbf{S}\mathbf{V}^{\top}\right\} \\ &= \mathbf{Tr}\left\{(\mathbf{R}_i\mathbf{V})^{\top}(\mathbf{R}_j\mathbf{U})\mathbf{S}\right\} = \mathbf{Tr}\left\{\mathbf{Q}_i^{\top}\mathbf{Q}_j\mathbf{S}\right\} \\ &= \sum_{\alpha=1}^{r}\mathbf{q}_{i_\alpha}^{\top}\mathbf{q}_{j_\alpha}\, s_\alpha,\end{aligned} \tag{5.18}$$

where $\mathbf{M} = \overline{\mathbf{P}}_{j,i}\overline{\mathbf{P}}_{i,j}^{\top}$ is the cross-covariance matrix of the shared points. SVD has been applied to $\mathbf{M}$ for obtaining the $r \times r$ orthonormal matrices $\mathbf{U}$ and $\mathbf{V}$ and also for the diagonal matrix $\mathbf{S}$. Clearly in Eq.5.18, $\mathbf{Q}_i = \mathbf{R}_i\mathbf{V}$ and $\mathbf{Q}_j = \mathbf{R}_j\mathbf{U}$ are $p \times r$ orthonormal matrices; we represent their columns by $\mathbf{q}_{i_\alpha}$ and $\mathbf{q}_{j_\alpha}$ respectively. Note that in this proof, we have no assumption regarding $p$ except that $p \geq r$.

Given that $\mathbf{M}$ is full-rank[5], all the diagonal elements of $\mathbf{S}$ are greater than zero ($s_\alpha > 0$). Since $\mathbf{q}_{i_\alpha}$ and $\mathbf{q}_{j_\alpha}$ are normal vectors, it is easy to show that $\mathbf{q}_{i_\alpha}^{\top}\mathbf{q}_{j_\alpha}\, s_\alpha \leq s_\alpha$. The equality only happens if $\forall\alpha \in \{1, 2, \ldots, r\} :\ \mathbf{q}_{i_\alpha} = \mathbf{q}_{j_\alpha}$, which indicates that Eq.5.18 is maximized when $\mathbf{Q}_i = \mathbf{Q}_j$. Therefore, we denote both of them by a single matrix $\mathbf{Q}$ consisting of $r$ orthonormal columns. Since $\mathbf{U}$ and $\mathbf{V}$ are orthonormal square matrices we have:

$$\mathbf{R}_i = \mathbf{Q}\mathbf{V}^{\top} \quad \text{and} \quad \mathbf{R}_j = \mathbf{Q}\mathbf{U}^{\top}; \tag{5.19}$$

Suppose $\mathbf{P}_i$ and $\mathbf{P}_j$ are stitched together. Using Eq.5.14, we can put all of their points in the following $p$-dimensional matrix:

$$\mathbf{Y}_{i\cup j} = \mathbf{R}\,\mathbf{F}(\mathcal{C}_i \cup \mathcal{C}_j) = [\mathbf{R}_i\ \mathbf{R}_j]\ \mathbf{F}(\mathcal{C}_i \cup \mathcal{C}_j) = \mathbf{Q}\left[\mathbf{V}^{\top}\ \mathbf{U}^{\top}\right]\ \mathbf{F}(\mathcal{C}_i \cup \mathcal{C}_j) = \mathbf{Q}\,\mathbf{\Delta}, \tag{5.20}$$

in which it is clear that the rearranged points lie on an $r$-dimensional subspace, which is spanned by the $r$ columns of the orthonormal matrix $\mathbf{Q}$. In other words, the attached patches are now aligned in an $r$-dimensional subspace. □

[5] For each two neighboring patches IPA requires at least $r+1$ points in common. In fact, it is sufficient to consider $r+1$ shared points whose coordinates are affinely independent in both patches.

Intuitively speaking, when the patches of a manifold are rearranged such that the neighbors can be stitched together, the expectation is to see them aligned in a subspace of dimensionality close to that of the patches. Therefore, the final configuration can be imagined as the underlying manifold of the input data, unfolded in a low-dimensional subspace. This is the main reason why IPA reduces the dimensionality of data.

### 5.1.4 The Algorithm

A simple pseudo-code for the proposed method is presented in Algorithm 5. As previously discussed, first the input data should be partitioned into $c$ clusters, where $c$ is given as input. This can be done either by subspace clustering or by any other clustering method. The target dimensionality $r$ can be given as an input, or can be set based on the maximum rank of the clusters. In the next step, all of the clusters are expanded to guarantee that the neighboring clusters have enough shared points. After that, the neighboring patches are detected, and the connectivity graph $\mathcal{G}$ and its Laplacian matrix $\mathbf{L}_{\mathcal{G}}$ are formed. Then, the expanded clusters are embedded to an $r$-dimensional space to form the patches. This embedding is usually done by PCA; however, it is also possible to use any other distance-preserving dimensionality reduction method, such as Isomap or MVU, to improve the embedding quality, although with a higher computational cost.

After calculating the coordinates of the embedded shared points ($\mathbf{P}_{i,j}$), all of the known variables ($\mathbf{Z}, \mathbf{L}_{\mathcal{X}}$, and $\mathbf{K}$) are computed to set up the SDP optimization of Eq.5.12. The SDP problem can be solved by any standard solver to obtain $\mathbf{A}$ and, subsequently, $\mathbf{R}$, by decomposing it. Finally, the $rc$-dimensional coordinates of the points in the unfolding space will be calculated, and then, by applying PCA, the unnecessary dimensions will be removed to represent the data in a low-dimensional space.

## 5.2 Experimental Results

To examine the different aspects of the proposed method, we conducted several experiments on IPA and compared its results to those of other prominent methods. These experiments can be divided into three parts: we start by studying the properties of IPA by applying it to synthetic datasets, then we show the quality of IPA for embedding real-world image sets, and finally as an interesting application, we will explain how IPA can be used for protein structure determination. All of the implementations were in Matlab, and the codes were executed on one core of a 3.2 GHz processor.

**Algorithm 5** Isometric Patch Alignment (IPA)

**Inputs:** $\mathbf{X}, c$ and $r$

// Partitioning $\mathbf{X}$ into low-rank clusters
$\{\mathcal{C}_1, .., \mathcal{C}_c\} \leftarrow Clustering(\mathbf{X}, c, r)$
**for all** $i \in \{1, 2, \ldots, c\}$ **do**
  $\mathcal{C}_i \leftarrow Expand(\mathcal{C}_i)$  // Expanding the clusters
  Compute the embedding function $\mathbf{f}_i$
**end for**
$\mathcal{E} \leftarrow \emptyset$
**for all** $i, j \in \{1, 2, \ldots, c\}$ **do**
  $\mathcal{C}_{i \cap j} \leftarrow \mathcal{C}_i \cap \mathcal{C}_j$
  **if** $\mathcal{C}_{i \cap j} \neq \emptyset$ **then**
    Add $(i, j)$ to $\mathcal{E}$  // Forming the neighborhood graph
    $\mathbf{P}_{i,j} \leftarrow \mathbf{f}_i(\mathcal{C}_{i \cap j})$
  **end if**
**end for**
Compute the Laplacian matrix $\mathbf{L}_{\mathcal{G}}$ from Eq.5.8
$\mathbf{L}_{\mathcal{G}}^{\dagger} \leftarrow \mathbf{L}_{\mathcal{G}}^{\top} \left(\mathbf{L}_{\mathcal{G}} \mathbf{L}_{\mathcal{G}}^{\top} + \mathbf{1}_c \mathbf{1}_c^{\top}\right)^{-1}$  // Computing pseudo-inverse
Compute the matrices $\mathbf{Z}$ and $\mathbf{L}_{\mathcal{X}}$ from Eq.5.7
$\mathbf{K} \leftarrow \mathbf{L}_{\mathcal{X}} - \mathbf{Z} \mathbf{L}_{\mathcal{G}}^{\dagger} \mathbf{Z}^{\top}$
Calculate $\mathbf{F}(\mathbf{X})$ from Eq.5.13
Solve the SDP problem in Eq.5.12 to obtain $\mathbf{A}$
Apply eigen-decomposition to $\mathbf{A}$ to find $\mathbf{R}$
$\mathbf{Y} \leftarrow \mathbf{R}\, \mathbf{F}(\mathbf{X})$  // Unfolding $\mathbf{X}$
Return $PCA(\mathbf{Y})$  // Removing unnecessary dimensions

### 5.2.1 Synthetic Datasets

As mentioned earlier in the introduction, using IPA has many advantages. Now we illustrate them separately via the results of our experiments on some synthetic datasets. Since the ground truth is usually known for the synthetic datasets, we decided to perform our comparisons based on them, rather than real-world datasets.

#### Handling non-convex manifolds

The first experiment was conducted on a modified Swiss-roll. Although Swiss-roll is a three-dimensional toy example, it tends to be one of the most challenging synthetic datasets due to its complex global structure. In addition, the complexity and challenge of this experiment has been augmented by considering a non-convex holed Swiss-roll. The hole has a rectangular shape, as depicted in Fig.5.2 (top-left). For this experiment, to have a better visualization of the data pattern, we sampled 700 data points from a regular grid over the manifold.

Non-convex datasets are, in fact, quite common [19], either because of the structure of the dataset itself or due to scarcity of samples. This can severely hamper the use of most existing methods (argued in [18]). For example, as shown in the right side of Fig.5.2, in this experiment we see that MVU, Isomap, and tSNE, have been unable to capture the rectangular shape of the hole, and have instead resulted in circular/ellipitcal holes. Although LTSA and LLE have found the rectangular hole, they have changed the scales, thereby distorting the dataset in Fig.5.2 (bottom-left). In contrast, IPA has been apparently successful in both finding the rectangular hole, and in preserving the pattern of the data over this non-convex manifold.

#### Superior quality of embedding

Despite the observable qualitative difference between the embeddings in Fig.5.2, we conducted a quantitative study on this experiment. We used two well-known criteria to study the quality of embedding for each method.

The first criterion is the *k-nearest neighbor intersection*, in which for each data point, two sets of $k$-nearest neighbors are formed; one set contains the neighbors in the input space and the other set in the embedding space. Then, these sets are compared together [13]; the more points they have in common, the better the quality of the embedding is. Mathematically speaking, it can be computed by

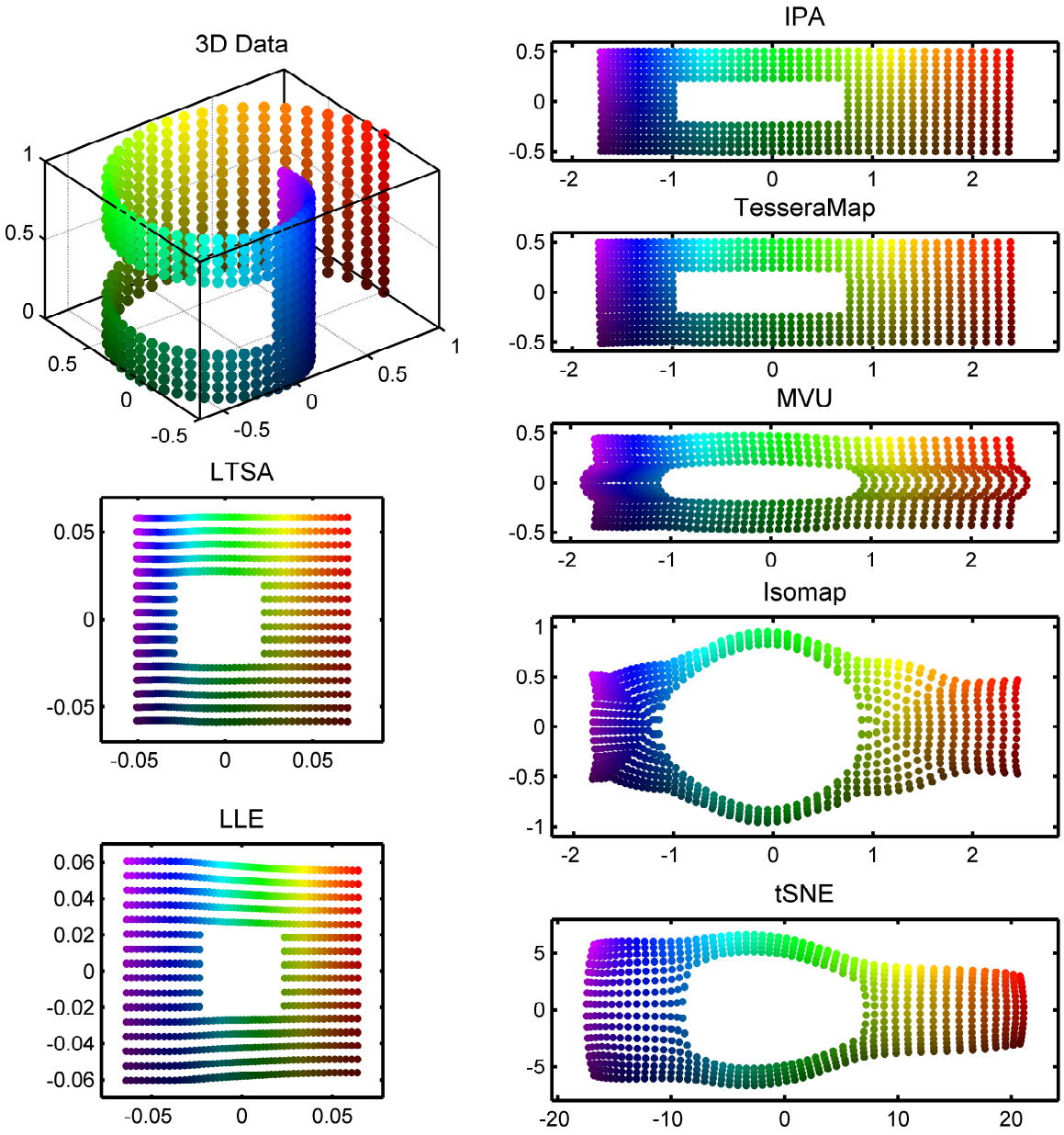


Figure 5.2: Different embeddings of a three-dimensional non-convex manifold.

$$\psi_{KNN} \;=\; 1 \;-\; \frac{1}{kn} \sum_{i=1}^{n} |\mathcal{N}_{\mathbf{x}_i} \cap \mathcal{N}_{\mathbf{y}_i}|, \tag{5.21}$$

where $\mathcal{N}_{\mathbf{x}_i}$ is the set of indices of the $k$-nearest neighbors of $\mathbf{x}_i$ in the input space, and in a similar way, $\mathcal{N}_{\mathbf{y}_i}$ is the $k$-nearest neighbor set of $\mathbf{y}_i$, where $\mathbf{y}_i$ is the low-dimensional representation of $\mathbf{x}_i$ in the unfolding space. To form an error measure, $\psi_{KNN}$ is normalized between zero and one, where zero indicates that all of the $k$-nearest neighbor sets are the same, and so the neighborhood is completely preserved during the embedding.

The other criterion is the *mean relative rank error (MRRE)* [36], that is based on the notion of proximity rank. To compute the proximity rank of the points with respect to point $\mathbf{x}_i$, first all $n-1$ points in $\{\mathbf{x}_j\}_{j\neq i}$ are sorted in ascending order based on their distance to $\mathbf{x}_i$ (i.e. $\|\mathbf{x}_i - \mathbf{x}_j\|$). Then, the rank of the $j^{\text{th}}$ input data point is computed as the position $\mathbf{x}_j$ in this ordered list, which is denoted by $\mathfrak{r}_{\mathbf{x}_i}(j)$. In a similar way, $\mathfrak{r}_{\mathbf{y}_i}(j)$ is computed based on the distances in the unfolding space. There are two different error measures can be defined for this criterion:

$$\begin{aligned} \text{MRRE}_{\mathbf{X}} \;&\triangleq\; \frac{1}{\beta} \sum_{i=1}^{n} \sum_{j\in\mathcal{N}_{\mathbf{x}_i}} \frac{|\mathfrak{r}_{\mathbf{x}_i}(j) - \mathfrak{r}_{\mathbf{y}_i}(j)|}{\mathfrak{r}_{\mathbf{x}_i}(j)} \\ \text{MRRE}_{\mathbf{Y}} \;&\triangleq\; \frac{1}{\beta} \sum_{i=1}^{n} \sum_{j\in\mathcal{N}_{\mathbf{y}_i}} \frac{|\mathfrak{r}_{\mathbf{y}_i}(j) - \mathfrak{r}_{\mathbf{x}_i}(j)|}{\mathfrak{r}_{\mathbf{y}_i}(j)}, \end{aligned} \tag{5.22}$$

where $\beta$ is used for scaling between zero and one, and given by:

$$\beta \;=\; n \sum_{\alpha=1}^{k} \frac{|n+1-2\,\alpha|}{\alpha}. \tag{5.23}$$

We assessed the quality of embedding for all the compared methods in Fig.5.2 by the aforementioned measures. The results are depicted in Fig.5.3. As shown, the errors of IPA and TesseraMap are significantly less than the errors of the other methods. In other words, the neighborhood of each point is better preserved by the proposed methods in embedding. Although based on MRRE, IPA and TesseraMap have similar qualities, based on the KNN intersection measure, the quality of embedding by IPA is slightly better than that of TesseraMap. It shows that the approach of IPA in using the overlapping patches can improve the quality, compared with the tessellation approach in TesseraMap.

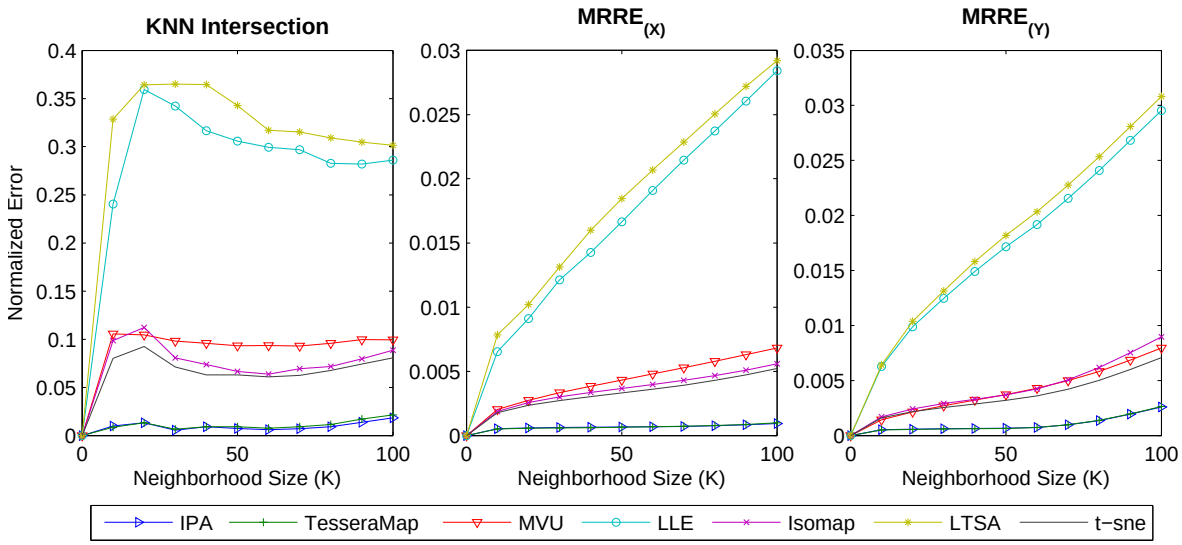


Figure 5.3: Evaluating the quality of embedding for different methods applied to the modified Swiss-roll. The KNN Intersection and mean relative rank errors are measured for different values of neighborhood size: $1 \leq k \leq 100$.

Based on this experiment, we can sort the quality of embedding in the following order: IPA (best), TesseraMap, tSNE, Isomap, MVU, LLE, and LTSA (worst). Since LLE and LTSA change the scales, they have the worst results in this experiment. The errors of Isomap, MVU and tSNE are very similar; although they have better results compared to LLE and LTSA, the magnitude of error in these methods is apparently higher than that of IPA and TesseraMap. It is important to note that for large neighborhood sizes preserving the neighborhood while the manifold is being unfolded, is not possible. This is the reason why for $k \geq 50$ the errors increase for IPA.

### Scalability

To illustrate the scalability of IPA, we sampled 1,000,000 points randomly from the same manifold (i.e. the holed Swiss-roll depicted in Fig.5.2). To the best of our knowledge, no dimensionality reduction method is able to handle this number of data points. Most dimensionality reduction methods (e.g., Isomap, LTSA, LLE) need to compute EVD for a matrix of size $n \times n$ which in practice is in $\mathcal{O}(n^3)$. In IPA the most complex step is to solve a SDP of size $rc \times rc$ with $cr^2$ constraints. Since our constraints are sparse and

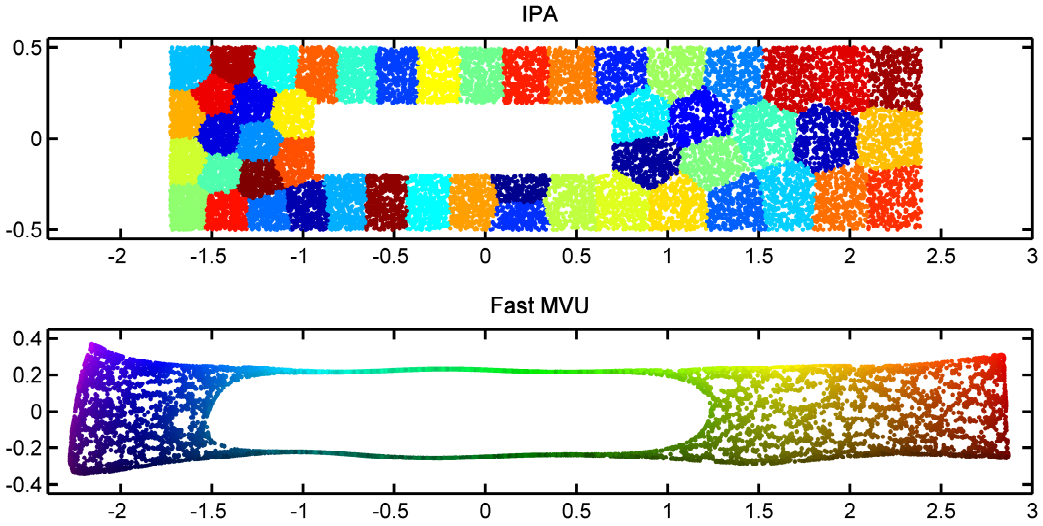


Figure 5.4: Embedding 1,000,000 points from the modified Swiss-roll manifold by IPA (top) and 10,000 points by Fast MVU (bottom).

rank-one, each step of optimization will take $\mathcal{O}(c^3r^6)$. Considering that in dimensionality reduction $r$ is usually small and also $n \gg c$, this implies that IPA is dramatically faster and more scalable than other prominent methods. Comparing the best case of TesseraMap, where the kernel matrix is full-rank and the number of the between-tesserae constraints are controlled, this method and IPA share the same order of computational complexity. However, the number of constraints and the size of the SDP matrix in TesseraMap are always greater than in IPA, which in practice, makes IPA even faster than TesseraMap.

We used $k$-means clustering to partition the points into 50 clusters, and embedded them by PCA into $r = 2$ dimensional patches. The final embedding is shown in Fig.5.4 (top), in which the hole has a rectangular shape and the Swiss-roll is perfectly unfolded. The overall running time was 202 seconds (182 seconds for clustering plus 20 seconds for running IPA).

Since it was not possible to run the other algorithms for this size of data, we chose Fast MVU, the scalable version of MVU, for comparison. However, even for Fast MVU handling more than 50,000 points is infeasible, and hence, we only considered 10,000 points, with 30 landmarks. Fast MVU took 31 seconds for computing an initial solution, and then used 10,000 iterations of fine tuning in 95 seconds. It is worth mentioning that the total running time of IPA for this dataset with 30 clusters (instead of landmarks) would not be greater than five seconds and also, it was about 14 seconds for TesseraMap. The result of

Fast MVU is shown in Fig.5.4 (bottom); on its two-dimensional representation of the given input manifold, the pattern of the data is clearly distorted. Although the postprocessing step in Fast MVU was the most time-consuming part of the embedding, it has not been able to fine-tune the final embedding properly.

### Noise Tolerance

To study the effect of noise, we chose a simple manifold which is depicted in Fig.5.5 (top-left), and drew 1500 random samples from it. First we applied IPA, TesseraMap, MVU, Isomap, LTSA, and LLE to the original manifold. The results are shown in the top row of Fig.5.5. Since the manifold is simple, all of the obtained results are acceptable. Then we added a Gaussian noise with the variance of 0.001. The noisy data points are shown in Fig.5.5 (middle-left). All of the aforementioned methods were again applied to the noisy dataset. The resulting embeddings are shown in the middle row of Fig.5.5. It is clear that MVU and LTSA are highly sensitive to noise; LLE has been affected too, but still its embedding is acceptable. In contrast, IPA, TesseraMap and Isomap have been able to cancel the effect of noise, and result in the expected embedding (i.e. the results shown in the first row). The main reason that MVU is not able to tolerate this small amount of noise is that the noise makes some short-cuts on the surface of the manifold, which forces MVU to bend the manifold on itself. If a large neighborhood size ($k > 50$ ) were considered for LTSA, the effect of noise might have been reduced; however, the set of neighboring points of each point would not form a low-rank (here two-dimensional) affine subspace, and the embedding could ruin the pattern of the data.

### Low sensitivity to data distribution

We used the non-noisy manifold of the previous experiment, to investigate the effect of non-uniform sampling on the embedding. This time, we drew 1350 points from one half of the manifold (purple side), and 150 points from the other half (red side), as shown in Fig.5.5 (bottom-left). That is, the density of the points in the first half of the manifold is 9 times the density of the other half. Unlike the result of the previous experiment, this time the embedding of MVU is acceptable, but the results of Isomap, LTSA and LLE have been distorted. The embeddings are depicted in the bottom row of Fig.5.5. This simple experiment illustrates that the embeddings of Isomap, LTSA, and LLE depend on the distribution of data and therefore, different sampling from the same manifold may change their results. In contrast, IPA, TesseraMap, and MVU are robust with respect to sampling and data distribution.

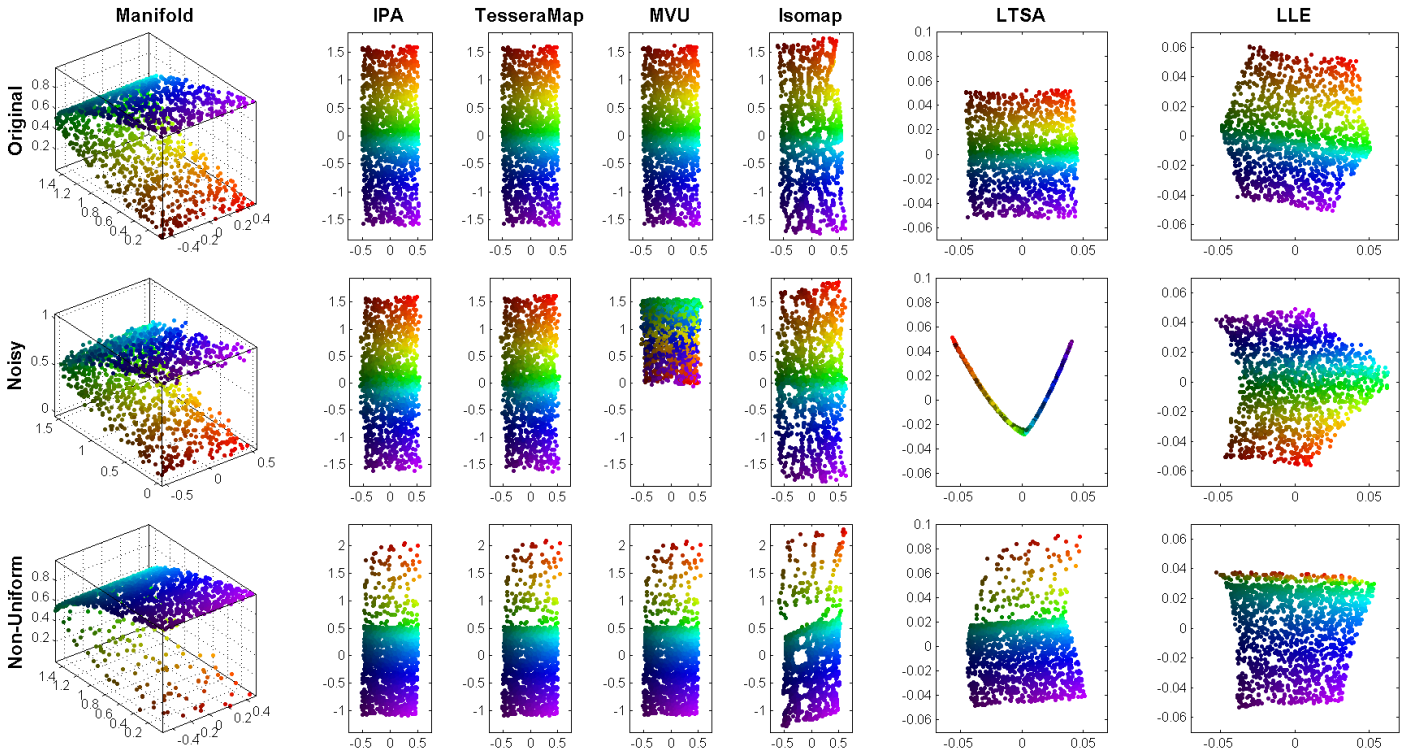


Figure 5.5: The effect of noise and non-uniform sampling on embedding.

#### Rank minimization

The third toy example that we used, was a star-shaped object consisting of four twisted rectangular segments joining in its center in a three-dimensional space (depicted in Fig.5.6). It has been shown ([55]) that the methods which unfold by maximizing the variance (for example, MVU), fail to embed this simple dataset into low dimensions. Applying MVU to this set resulted in a three-dimensional star, which is shown in Fig.5.6 (right). Thus, applying PCA to obtain the final embedding in two dimensions will change the length of the segments and so the distances cannot be preserved.

The result of TesseraMap is depicted in Fig.5.6 (middle). Since TesseraMap unfolds by maximizing the variance, it has the same issue as MVU and its embedding is not of rank two. However, TesseraMap uses the idea of tessellation, and since attaching each two neighboring tesserae is similar to attaching the patches in IPA, this method tends to align the tesserae. Therefore, in this embedding, the third eigenvalue of the covariance matrix of the points is smaller than that of MVU. That is, in TesseraMap, the third dimension contains less information, compared to MVU. It is clear that IPA has perfectly embedded the star in the target dimensionality ($r = 2$), and has been able to preserve the length of the segments. This experiment simply shows a major difference between the approaches for minimizing the rank.

### 5.2.2 Real World Image Sets

After inspecting the characteristics of IPA on the synthetic datasets, and comparing it to TesseraMap and also the other methods, we conducted some experiments on the real-world image datasets. We studied the quality of embedding on two real-world datasets. Since real data usually has a complex structure, LDLC is applied for clustering the data points into low-dimensional patches.

#### Visualizing the Frey face images

The first dataset consisted of 1965 face images in different moods. These images were partitioned into 30 patches of dimensionality $r = 2$. LDLC was applied with 10 random seeds, which in total took 23 seconds, and then IPA embedded the images in 5 seconds. The resulting embedding is shown in Fig.5.7, where the orientation of the faces is totally captured, while the images showing similar moods are gathered together.

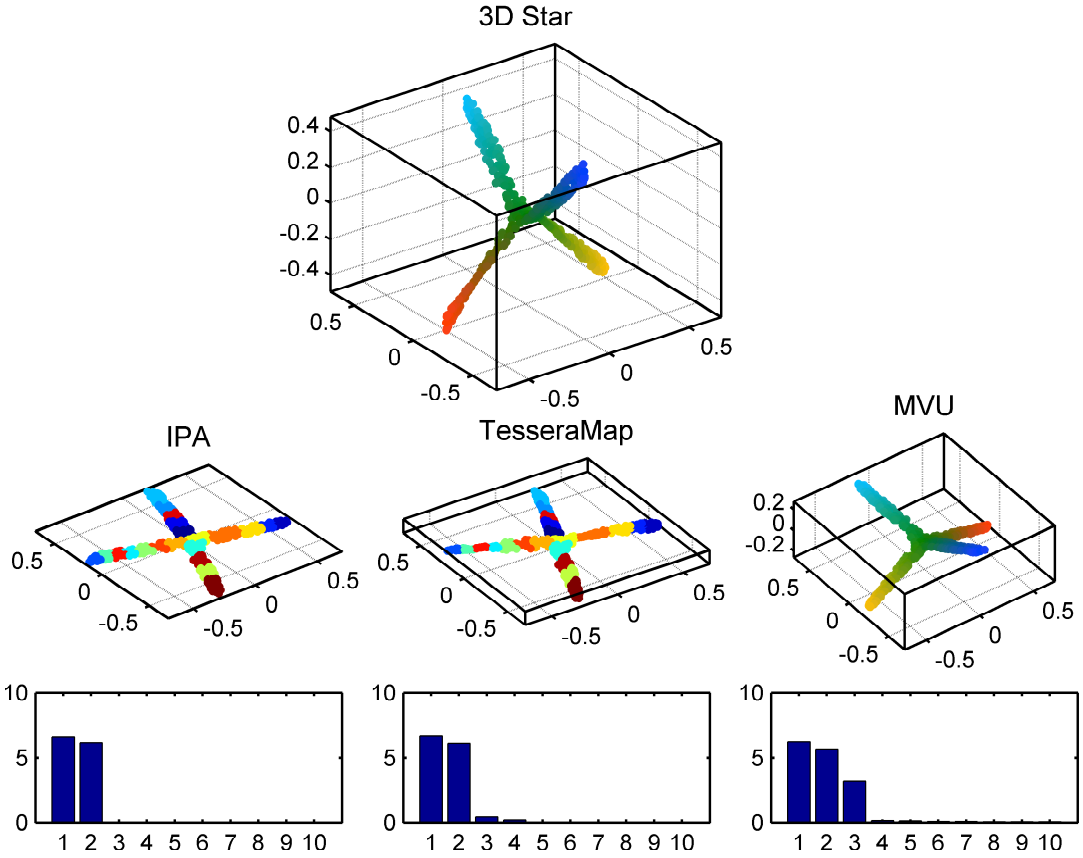


Figure 5.6: A three-dimensional star-shaped object and its embedding by IPA, TesseraMap and MVU, with their eigenvalue spectra.

To study the effect of target dimensionality on the rank of final embedding, we repeated this experiment for $r = 2, 3, \ldots, 7$, and for each value of $r$, we computed the eigenvalue spectrum of the embedded points separately. For each spectrum, we calculated the percentage of variance covered in each dimension. The results are depicted in Fig.5.8. For example, when the target dimensionality was set to two, the final embedding was of rank four, and PCA was able to capture 80% of the total variance, or when $r = 8$, the rank of the final embedding was 14, and embedding in seven dimensions could cover at least 90% of the data variation. In Fig.5.8, it can be seen that the rank of the final embedding is always close to the target dimensionality, which in fact validates Theorem 1 in practice.

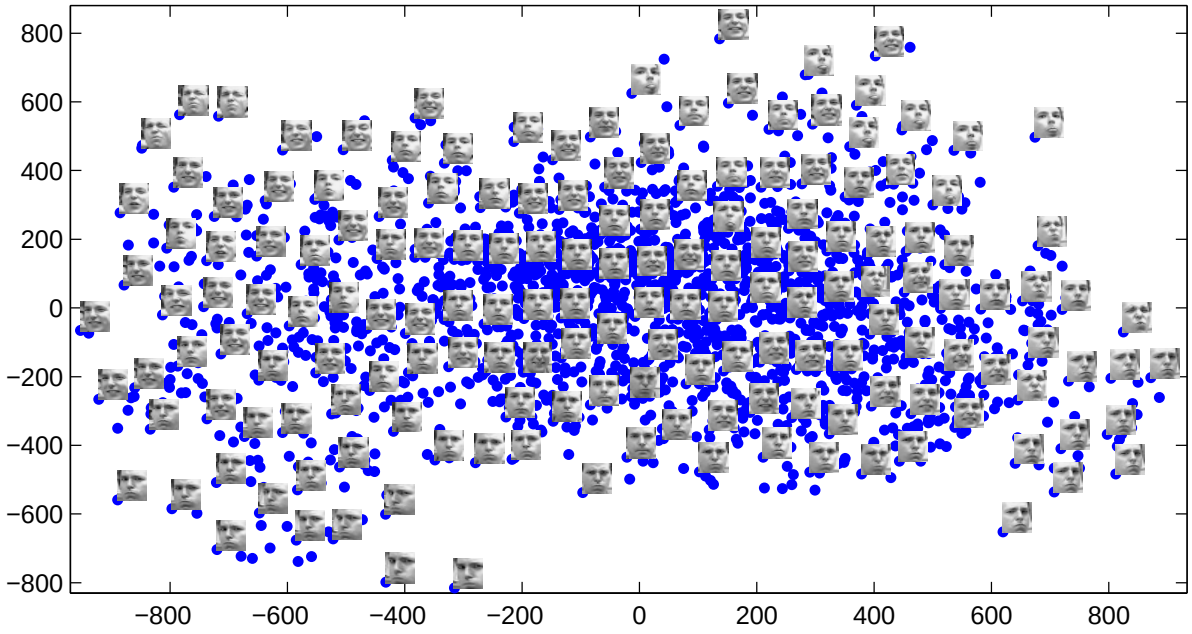


Figure 5.7: Embedding Frey face images into two dimensions by IPA.

### Visualizing the MNIST Dataset

For the last experiment on the images, to challenge IPA on a complex dataset, we considered MNIST, which contains many handwritten samples of all digits. As discussed in Chapter 4, for this dataset, instead of one underlying manifold, there exists a union of manifolds from which the images are sampled. In Chapter 3 we showed the embedding of these manifolds separately, and also for digits 3, 6, and 9 together. Like TesseraMap, IPA preserves the local distances; thus, for the same reason that we explained for TesseraMap, we do not expect to observe that IPA separates all of the digit classes in its embedding. However, since TesseraMap was able to embed the underlying manifolds of digits 3, 6, and 9 all together, we conducted the same experiment on IPA.

From those three digits, we took 1500 samples. These points were partitioned into 30 patches for IPA, and then embedded by different methods. The first two dimensions of the embedding were used for visualization in Fig.5.9 where the samples of each digit are shown with a different color. In addition, some of the images are shown in Fig.5.10. In the embedding of IPA, the variation of the images is considered, while the samples of different digits are clearly separated.

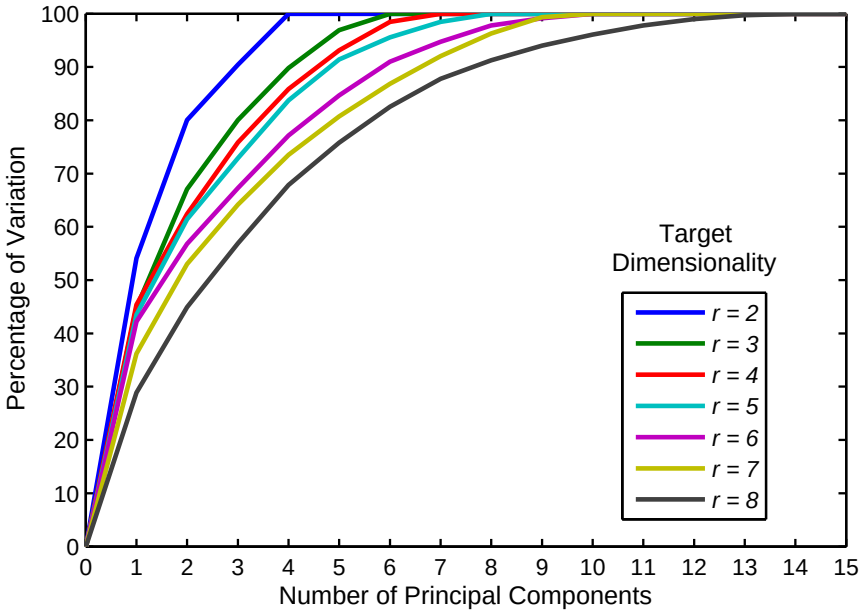


Figure 5.8: The resulting eigenvalue spectra of the embedded Frey images for different target dimensionalities.

In this experiment, it is clear from the results that the underlying manifold of these three digits are connected to each other; however, tSNE has separated them in the embedding. Therefore, its embedding does not represent the true pattern of the original data. To study the quality of the different embeddings in this experiment, we computed the MRRE and KNN Intersect measures for each method. Moreover, to see the effect of $r$, we repeat the experiment for $r \in \{2, 3, 5, 10\}$. It is important to note that the intrinsic dimensionality of these manifolds is more than 50, and consequently, the task of dimensionality reduction on this set is really challenging. The results are shown in Fig.5.11.

As shown, when the target dimensionality is low (i.e. $r = 2$ or $r = 3$), since the loss of information is high - especially for IPA in applying PCA to form the patches - the neighborhoods are not preserved completely. But, as the target dimensionality increases toward the intrinsic dimensionality of the data, IPA outperforms the other methods.

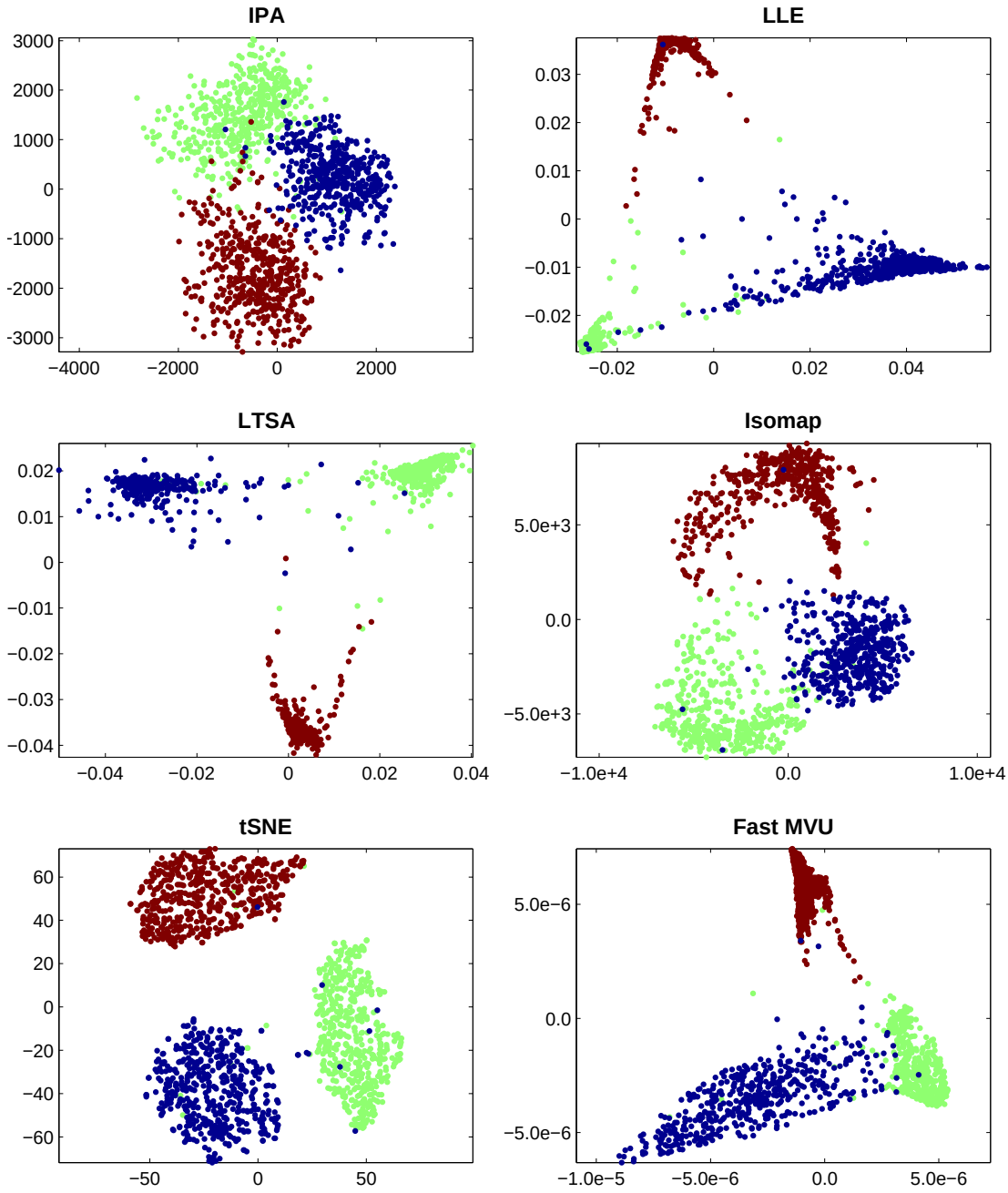


Figure 5.9: The embedding of three digits 3-6-9 from the MNIST dataset in two dimensions by different methods. From each of the digits 500 data points have been used as the input.

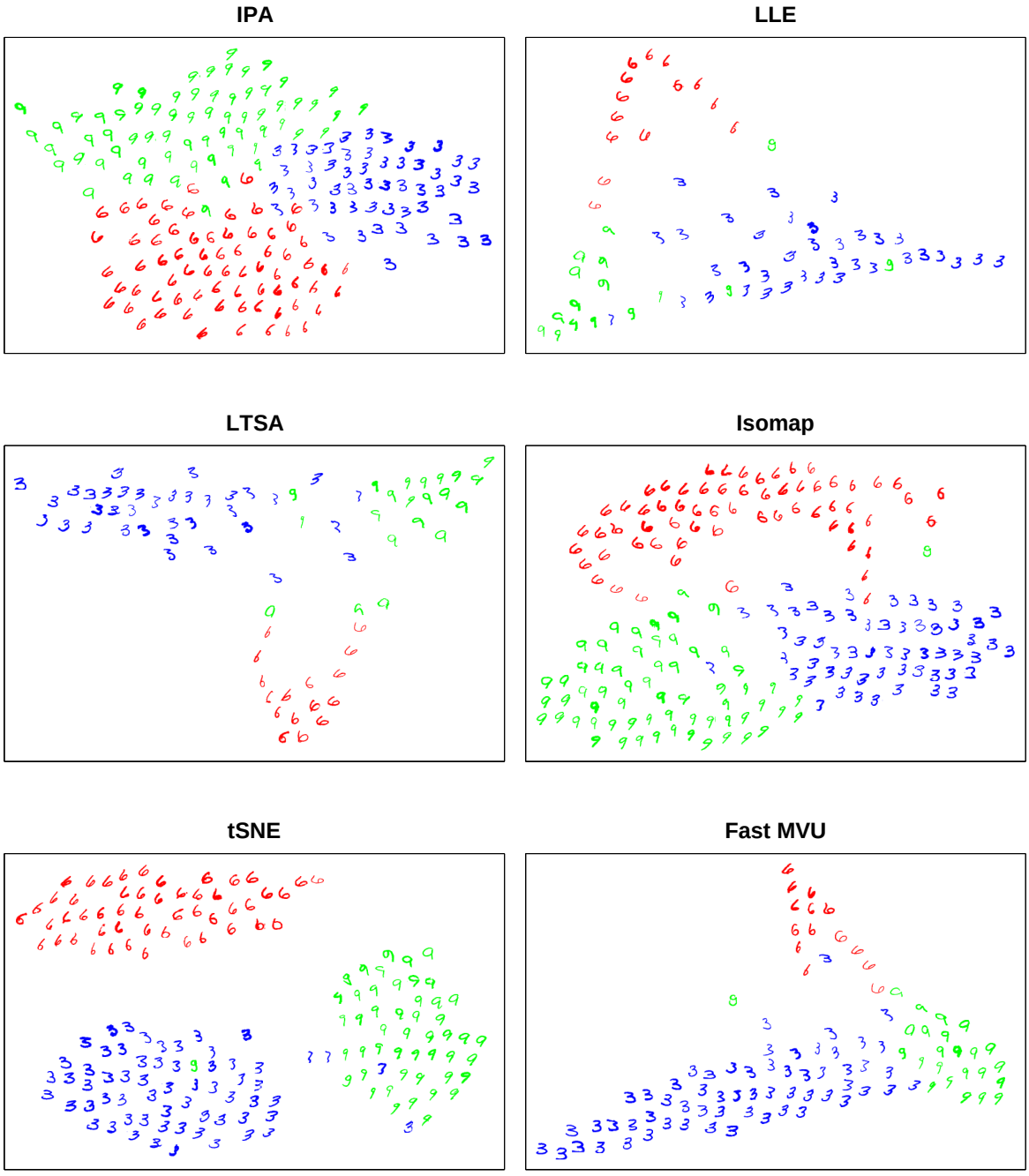


Figure 5.10: The visualization of the images of 3,6, and 9 from the MNIST dataset in two dimensions. For each different method some sample images of the digits are shown.

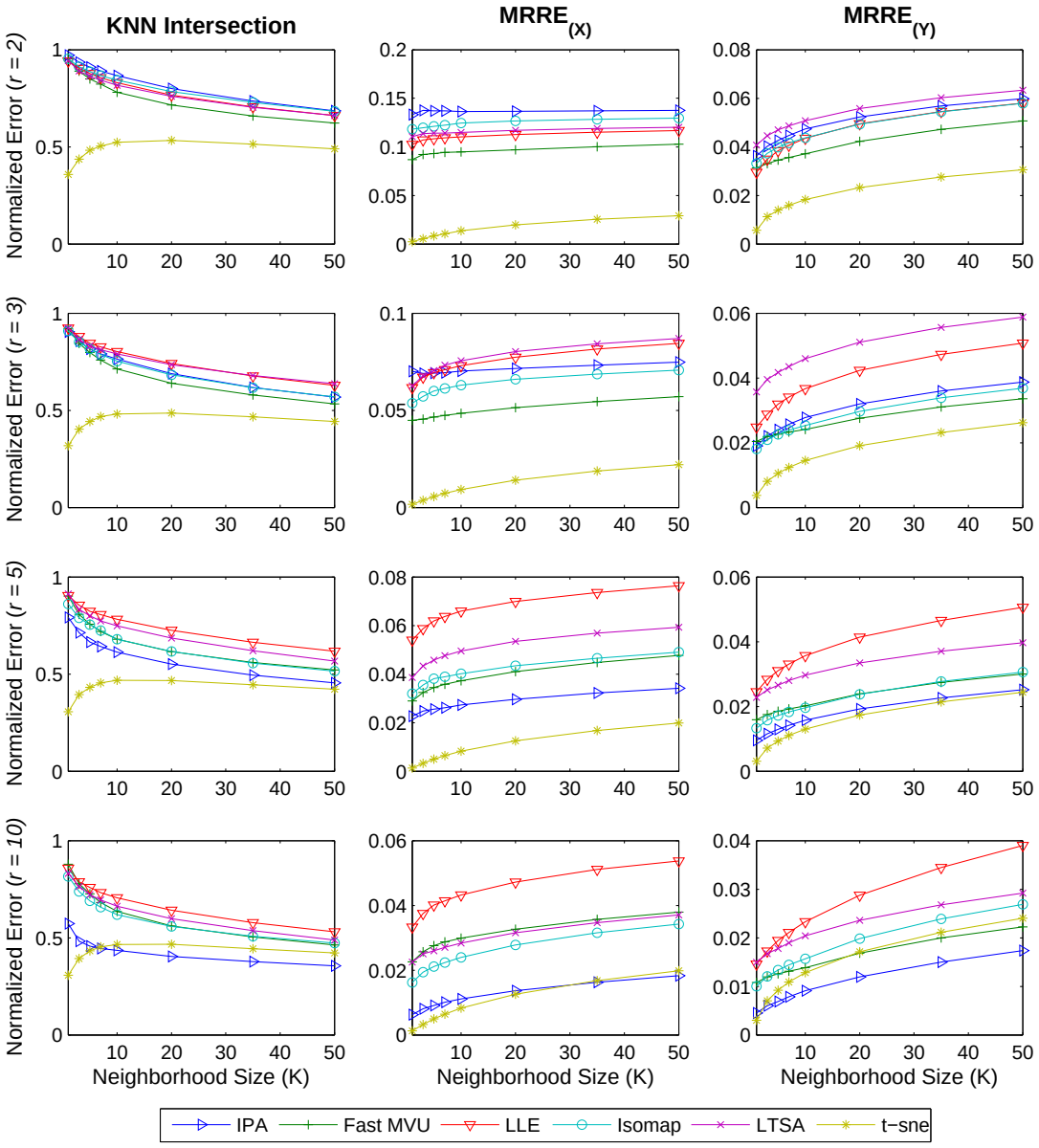


Figure 5.11: Measuring the quality of embedding for different methods applied to MNIST, for the target dimensionality $r = 2$, 3, 5, and 10. First column of the plots shows the $k$-nearest neighbor intersection error, and the second and third columns show the mean relative rank error of the original space and the embedding space respectively. The plots in each row represent the measurements for one target dimensionality.

### 5.2.3 Protein Structure Determination

Computing the three-dimensional structure of biomacromolecules such as proteins is one of the most important and challenging problems in structural bioinformatics. In protein Nuclear Magnetic Resonance (NMR), the structure of a protein is calculated from a set of experimentally-determined short-range upper bounds between proximal hydrogen atoms and the domain knowledge about proteins, such as bond angles and bond lengths [24]. Theoretically, any distance preserving dimensionality reduction method can be employed to compute the three-dimensional coordinates of the atoms in a protein. However, since the size of protein molecules, i.e., the number of atoms, is usually in the range of a few thousands, the scalability of the employed method is crucial. Therefore, employing current distance preserving methods, such as MVU, for determining the three-dimensional structure of a large protein molecule is usually very time-consuming.

Here, we observe that IPA is naturally suitable for the protein structure determination problem. Based on the domain knowledge, all the pairwise distances between the atoms within certain substructures of protein molecules, such as peptide planes, tetrahedral carbons, and aromatic rings are completely known. IPA considers these substructures as the low-dimensional patches of data and treats them as rigid parts. Consequently, each protein molecule is virtually considered as a set of interconnecting patches [1]. The neighboring patches have one or more atoms in common, for example, two consecutive peptide planes can be modeled as two patches that share the corresponding alpha carbon. Moreover, based on the short-range upper bounds between the hydrogen atoms, the distances between hydrogen atoms are constrained.

Interestingly, not only can IPA be employed for the protein structure determination problem, but it can also significantly reduce the problem size. Imagine the planar indole functional group in the Tryptophan side chain that contains 16 atoms; this group is, in fact, a two-dimensional patch and hence, can be represented by two columns of a matrix $\mathbf{R}_i$. That is, the size of the PSD variable is only increased by two, instead of 16.

We tested IPA on several protein datasets made from the protein structures deposited in the Protein Data Bank [7]. These datasets were made as follows: (*i*) the downloaded structures were parsed and the ground-truth coordinates were recorded, (*ii*) all hydrogen atom pairs closer than 6 Å were enumerated, and to make the test more realistic half of the pairs were randomly discarded, and (*iii*) upper bounds were formed by adding 10% multiplicative noise. We assume that IPA has access to all the protein conformation-independent distances between atoms as the domain knowledge. Based on that, to solve the protein structure, we first form the patches of the input protein molecule and then determine their shared points.

Some of the solved structures together with the reference structures from PDB are shown in Fig. 5.12. As can be seen in this figure, for each case, the two structures are very close, which shows that IPA was able to mitigate the effect of the added noise to the upper bounds. It is important to note that many other methods, such as LTSA, LLE, and tSNE, are not suitable to be used in this application.

## 5.3 Conclusion and Discussion

We proposed Isometric Patch Alignment (IPA) as a novel dimensionality reduction method with a low computational cost. In IPA, the input data is first partitioned into a number of clusters, from which a set of overlapping low-dimensional patches is created. Then all of the patches are rearranged such that the shared points between the neighboring ones are matched accordingly. We showed that the rearrangement can be computed by solving a small semidefinite program which has a unique solution.

In addition, we proved that stitching two neighboring patches on an adequate number of common points aligns them in a low-dimensional subspace, which implies that rearranging the patches of the underlying manifold unfolds it. Finally, by performing a wide range of experiments, we demonstrated the superiority of IPA to conventional methods in different applications, including in manifold learning, in data visualization, and also in the protein structure determination problem.

In the partitioning step, although low-rank clusters are preferred, IPA does not necessarily require the clusters to be low-rank. In fact, any clustering method can be used for partitioning in IPA. If the points of a cluster lie near a low-dimensional affine subspace we simply embed them by PCA and compute a low-dimensional patch for that cluster. However, if the points of a cluster cannot be approximated by any low-dimensional flat, we should use a nonlinear distance-preserving dimensionality reduction method, such as MVU or Isomap, for embedding. Generally, if the intrinsic dimensionality of a cluster is not low, no such dimensionality reduction method is able to embed the cluster to a low-dimensional patch while the local distances are preserved. In that case, the cluster should be partitioned further into two or more new clusters; otherwise, due to preserving the local pattern of the data within the cluster, the dimensionality of its patch cannot be low, which may increase the dimensionality of the final embedding by IPA.

In Alg.5, the target dimensionality, $r$, is a given input. For convenience, we assumed that $r$ is constant for all of the patches. However, it is possible to consider a different dimensionality for each patch. That is, one may consider an intrinsic dimensionality $r_i$ for

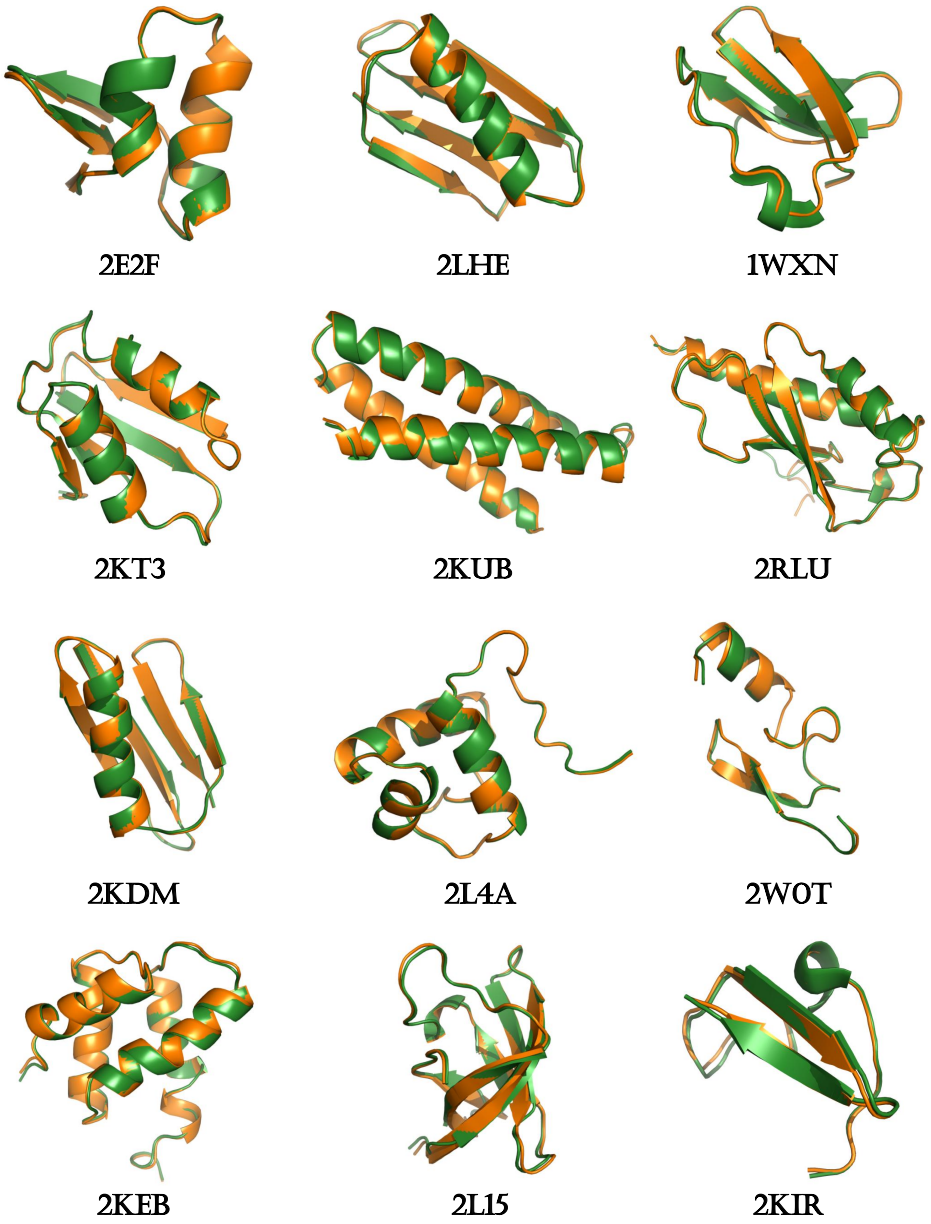


Figure 5.12: Superimposition of protein structures determined by IPA in orange and reference structures from PDB in green.

cluster $\mathcal{C}_i$; therefore, its points would be embedded into an $r_i \times |\mathcal{C}_i|$ patch $\mathbf{P}_i$. In that case, the entire argument of this chapter is still valid, with small modifications, namely on the definition of the selection matrix $\mathbf{E}_i$ and also on the constraint matrix.

During rearrangement, the pattern of data is completely preserved inside the patches; however, it can be distorted along the border of the patches, where they are stitched together. For a non-flat cluster, this distortion is more evident if the size of the cluster is larger. This is the main reason why a divide and conquer approach, based on only a small number of clusters (in an extreme case, by dividing the manifold into two pieces), would fail, and hence, it is suggested that more clusters be used for the complex manifolds.

Theorem 1 explains that stitching two neighboring patches aligns them. Using this theorem, it is evident that if the neighborhood graph $\mathcal{G}$ is a tree, the unfolded manifold will be constructed exactly in an $r$-dimensional subspace of the unfolding space. Nevertheless, if $\mathcal{G}$ contains cycles, more than $r$ dimensions are needed to represent the unfolded manifold. Based on our experiments, we observed that in practice, the maximum dimensionality needed to represent the unfolded manifold is $2r$.

Interestingly, from Eq.5.14 we can simply compute the Gram matrix of the embedded points as $\mathbf{G} = \mathbf{Y}^\top \mathbf{Y} = \mathbf{F}(\mathbf{X})^\top \mathbf{A} \mathbf{F}(\mathbf{X})$. It shows that the Gram matrix $\mathbf{G}$ is linearly related to the PSD matrix $\mathbf{A}$. This gives flexibility in modifying the optimization problem of IPA; for example, one can add the trace of $\mathbf{G}$ (i.e. the objective of MVU) in the cost function. In addition, it is possible to add new constraints, such as upper-bounds on the distances, or even to add angle constraints.

Last, it should be mentioned that one can use IPA for mapping out-of-samples. For a given out-of-sample test point $\mathbf{x}$, it is enough to decide which clusters or patches it belongs to. Then, based on Eq.5.13, the new coordinates of the point $\mathbf{x}$ are obtained from $\mathbf{y} = \frac{1}{m_\mathbf{x}} \sum_{\forall i:\ \mathbf{x} \in \mathcal{C}_i} \mathbf{R}_i \mathbf{f}_i(\mathbf{x}) + \mathbf{t}_i$, or simply by assuming a single cluster for each test point we have $\mathbf{x} \in \mathcal{C}_i \ \Rightarrow \ \mathbf{y} = \mathbf{R}_i \mathbf{f}_i(\mathbf{x}) + \mathbf{t}_i$. Therefore, IPA, similar to EAT and TesseraMap, is capable of handling out-of-samples.

# Chapter 6

# Conclusion and Discussion

We have proposed three novel methods for dimensionality reduction and also developed a new technique for subspace clustering. In all of these methods, we modeled the high-dimensional input data from a geometric point of view. That is, we assumed that there exists an underlying manifold from which the input data points have been sampled. We provided three solutions for unfolding the underlying manifold, thereby reducing the dimensionality, in which the local distances are preserved. All of these solutions were formulated as instances of semidefinite programming. Despite similar approaches, our proposed methods have distinct motivations and address different shortcomings in dimensionality reduction.

Our first proposed method in dimensionality reduction is Embedding by Affine Transformations (EAT). The capabilities of linear and nonlinear methods are combined in EAT. This method computes a linear transformation for unfolding, which can be used to map out-of-samples. We explained that EAT can be kernelized, such that, unlike the other kernel methods, unfolding does not depend on the choice of kernel. That is, EAT is able to nonlinearly map the new samples.

The second method that we proposed for dimensionality reduction is TesseraMap. TesseraMap shares the same approach with EAT in solving the problem; however, it does so with a dramatically enhanced scalability. While the high computational cost of solving SDP in EAT limits the number of input points to, at most, 1000 samples, in TesseraMap datasets consisting of 50,000 points can be handled in less than ten minutes. The scalability of TesseraMap is not only an improvement compared with EAT; it is generally an asset in dimensionality reduction, considering that most prominent methods, especially the distance preserving techniques, cannot embed more than 10,000 points.

Both EAT and TesseraMap compute a linear mapping that, when applied, pulls the non-neighboring points apart, thereby unfolding the underlying manifold. Although this approach is computationally efficient for relaxing rank optimization, it does not guarantee the generation of an embedding with the lowest possible rank. Hence, we proposed Isometric Patch Alignment (IPA), which constructs an unfolded instance of the underlying manifold by partitioning the manifold to a number of overlapping flat patches and aligning them together. We showed that in some cases, the rank of the final embedding by IPA can be lower than that of TesseraMap. Note that IPA and TesseraMap employ similar techniques for reducing the computational complexity of their convex optimizations, although in practice, the ratio of the number of data points that IPA can embed, with respect to TesseraMap in the same amount of time, is about 100.

In addition to these three methods in dimensionality reduction, we have proposed a novel technique in subspace clustering, called Low Dimensional Localized Clustering (LDLC). This method finds clusters of the high-dimensional input points, such that each cluster not only can be modeled by a low-dimensional affine subspace, but can also form a localized subset of the underlying manifold of the data. The clustering result of LDLC is an invaluable asset for both IPA and TesseraMap. In addition, we showed that this method has the potential for many applications in machine learning.

We tested all the proposed methods in practice, and ran several experiments on both synthetic and real-world datasets to study their functionalities. Many comparisons to other prominent methods have been conducted, and the results have been analyzed qualitatively and quantitatively. We believe the final outcome of the research for this thesis, IPA, is superior in both efficiency and quality of embedding compared with all other existing dimensionality reduction methods.

## 6.1 Future Work

We proposed four methods for handling high-dimensional data in machine learning, each of which can be extended further or in other directions. In addition, new questions arise with potential answers which may motivate new areas of research. We list some of these directions and questions:

- What kind of kernel function would work well with EAT and TesseraMap? Although we explained that having full-rank kernel matrices is beneficial for embedding, we showed some examples in which rank-deficient kernel matrices could work as well.

One might expect to observe less over-fitting by using rank-deficient kernels. Therefore, perhaps by using a mixture of kernels, it might be possible to maintain a balance between over-fitting and unfolding.

- Can we sensibly define an embedding error for dimensionality reduction? Is there a bound for this error on the embeddings produced by the proposed dimensionality reduction methods? Conducting this kind of analysis even on a simple manifold, for example a Swiss-roll, can be beneficial.

- How can we run these methods on a set of parallel processors? We expect that the iterative nature of LDLC is suitable for parallel processing. Also, the clustered form of IPA and TesseraMap can be an advantage in designing a parallel algorithm for dimensionality reduction.

- Is it possible to propose an algorithm for LDLC using convex optimizations?

- Is there a way to partition input data into low-rank clusters, such that the points of each cluster are sampled from a convex subset of the underlying manifold? It is beneficial to have an algorithm which, similar to $k$-means, partitions the input space into convex regions, and meanwhile, minimizes the ranks of the output clusters.

- Is it possible to extend or modify IPA from a probabilistic point of view, rather than from a geometric point of view? For example, one might start by considering different latent variables and by mixing them, find a low-dimensional representation for the data.

- Is there a bound for the rank of embedding produced by EAT, IPA, or TesseraMap? Suppose a manifold is given by its mathematical formulation. How large is the rank of embedding produced by these methods, considering infinitely many samples from the manifold?

- Is there a method to reduce the distortion in the data pattern along with the border of two neighboring patches in IPA? In IPA, we map each shared point to the mean of its coordinates in the different patches. A nonlinear combination of these coordinates might possibly improve the results. That is, one could exploit other methods for stitching the patches, which may result in new dimensionality reduction methods.

# Copyright Permissions

Parts of Chapter 2 and Chapter 4 are reprinted from the following published papers:

[32] - Pooyan Khajehpour Tadavani and Ali Ghodsi. Learning an affine transformation for nonlinear dimensionality reduction. In Jose Luis Balcazar, Francesco Bonchi, Aristides Gionis, and Michele Sebag, editors, Machine Learning and Knowledge Discovery in Databases, volume 6322 of Lecture Notes in Computer Science, pages 19-34. Copyright © Springer-Verlag London Limited 2012. Reprinted with kind permission from Springer Science and Business Media. The Springer Licence is included on page 131.

[33] - Pooyan Khajehpour Tadavani and Ali Ghodsi. Low-dimensional Localized Clustering (LDLC). In Data Mining (ICDM), 2012 IEEE 12th International Conference on, pages 936-941, Copyright © IEEE 2012. The IEEE Thesis/Dissertation Reuse statement is included on page 134.

**SPRINGER LICENSE**
**TERMS AND CONDITIONS**

Sep 23, 2013

This is a License Agreement between pooyan khajehpour tadavani ("You") and Springer ("Springer") provided by Copyright Clearance Center ("CCC"). The license consists of your order details, the terms and conditions provided by Springer, and the payment terms and conditions.

**All payments must be made in full to CCC. For payment instructions, please see information listed at the bottom of this form.**

| | |
|---|---|
| License Number | ********** |
| License date | Mar 05, 2013 |
| Licensed content publisher | Springer |
| Licensed content publication | Springer eBook |
| Licensed content title | Learning an Affine Transformation for Non-linear Dimensionality Reduction |
| Licensed content author | Pooyan Khajehpour Tadavani |
| Licensed content date | Aug 17, 2010 |
| Type of Use | Thesis/Dissertation |
| Portion | Full text |
| Number of copies | 10 |
| Author of this Springer article | Yes and you are the sole author of the new work |
| Order reference number | |
| Title of your thesis / dissertation | Nonlinear Dimensionality Reduction by Manifold Unfolding |
| Expected completion date | May 2013 |
| Estimated size(pages) | 200 |
| Total | 0.00 CAD |
| Terms and Conditions | |

Introduction
The publisher for this copyrighted material is Springer Science + Business Media. By clicking "accept" in connection with completing this licensing transaction, you agree that the following terms and conditions apply to this transaction (along with the Billing and Payment terms and conditions established by Copyright Clearance Center, Inc. ("CCC"), at the time that you opened your Rightslink account and that are available at any time at http://myaccount.copyright.com).

Limited License
With reference to your request to reprint in your thesis material on which Springer Science and Business Media control the copyright, permission is granted, free of charge, for the use indicated in your enquiry.

Licenses are for one-time use only with a maximum distribution equal to the number that you identified in the licensing process.

No Transfer of License
This license is personal to you and may not be sublicensed, assigned, or transferred by you to any other person without Springer Science + Business Media's written permission.

No Amendment Except in Writing
This license may not be amended except in a writing signed by both parties (or, in the case of Springer Science + Business Media, by CCC on Springer Science + Business Media's behalf).

Objection to Contrary Terms
Springer Science + Business Media hereby objects to any terms contained in any purchase order, acknowledgment, check endorsement or other writing prepared by you, which terms are inconsistent with these terms and conditions or CCC's Billing and Payment terms and conditions. These terms and conditions, together with CCC's Billing and Payment terms and conditions (which are incorporated herein), comprise the entire agreement between you and Springer Science + Business Media (and CCC) concerning this licensing transaction. In the event of any conflict between your obligations established by these terms and conditions and those established by CCC's Billing and Payment terms and conditions, these terms and conditions shall control.

Jurisdiction
All disputes that may arise in connection with this present License, or the breach thereof, shall be settled exclusively by arbitration, to be held in The Netherlands, in accordance with Dutch law, and to be conducted under the Rules of the 'Netherlands Arbitrage Instituut' (Netherlands Institute of Arbitration).***OR:***

**All disputes that may arise in connection with this present License, or the breach thereof, shall be settled exclusively by arbitration, to be held in the Federal Republic of Germany, in accordance with German law.**

**Other terms and conditions:**

**v1.3**

**If you would like to pay for this license now, please remit this license along with your payment made payable to "COPYRIGHT CLEARANCE CENTER" otherwise you will be invoiced within 48 hours of the license date. Payment should be in the form of a check or money order referencing your account number and this invoice number 500970667.**
**Once you receive your invoice for this order, you may pay your invoice by credit card. Please follow instructions provided at that time.**

**Make Payment To:**
**Copyright Clearance Center**
**Dept 001**
**P.O. Box 843006**
**Boston, MA 02284-3006**

**For suggestions or comments regarding this order, contact RightsLink Customer Support:customercare@copyright.com or +1-877-622-5543 (toll free in the US) or +1-978-646-2777.**

**Gratis licenses (referencing $0 in the Total field) are free. Please retain this printable license for your reference. No payment is required.**

# References


[1] Babak Alipanahi, Nathan Krislock, Ali Ghodsi, Henry Wolkowicz, Logan Donaldson, and Ming Li. Determining protein structures from NOESY distance constraints by semidefinite programming. *Journal of computational biology : a journal of computational molecular cell biology*, 20(4):296–310, April 2013.

[2] DJ Bartholomew. Latent variable models and factor analysis, 1987.

[3] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in neural information processing systems*, 14:585–591, 2001.

[4] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

[5] R.E. Bellman. *Adaptive control processes: a guided tour*. Rand Corporation Research studies. Princeton University Press, 1961.

[6] Yoshua Bengio, Jean-François Paiement, Pascal Vincent, Olivier Delalleau, Nicolas Le Roux, and Marie Ouimet. Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering. *Advances in neural information processing systems*, 16:177–184, 2004.

[7] F. C. Bernstein, T. F. Koetzle, G. J. Williams, E. F. Meyer, M. D. Brice, J. R. Rodgers, O. Kennard, T. Shimanouchi, and M. Tasumi. The protein data bank: a computer-based archival file for macromolecular structures. *Journal of molecular biology*, 112(3):535–542, May 1977.

[8] M.W. Berry. *Survey of Text Mining I: Clustering, Classification, and Retrieval*. Number v. 1. Springer, 2004.

[9] Brian Borchers. Csdp, ac library for semidefinite programming. *Optimization Methods and Software*, 11(1-4):613–623, 1999.

[10] Brian Borchers and Joseph G. Young. Implementation of a primal-dual method for SDP on a shared memory parallel architecture. *Comput. Optim. Appl.*, 37(3):355–369, 2007.

[11] Jonathan M Borwein and Henry Wolkowicz. Facial reduction for a cone-convex programming problem. *Journal of the Australian Mathematical Society. Series A: Pure mathematics*, 30:369–380, 1981.

[12] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[13] Lisha Chen and Andreas Buja. Local multidimensional scaling for nonlinear dimension reduction, graph drawing, and proximity analysis. *Journal of the American Statistical Association*, 104(485):209–219, 2009.

[14] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[15] Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. Modeling wine preferences by data mining from physicochemical properties. *Decis. Support Syst.*, 47:547–553, November 2009.

[16] T. Cox and M. Cox. *Multidimensional Scaling*. Chapman Hall, Boca Raton, 2nd edition, 2001.

[17] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[18] David L. Donoho and C. Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Sciences*, 100(10):5591–5596, 2003.

[19] David L. Donoho and Carrie Grimes. Image manifolds which are isometric to euclidean space. *Journal of Mathematical Imaging and Vision*, 23:5–24, 2005.

[20] M. Fazel, H. Hindi, and S.P. Boyd. A rank minimization heuristic with application to minimum order system approximation. In *American Control Conference, 2001. Proceedings of the 2001*, volume 6, pages 4734 –4739 vol.6, 2001.

[21] M. Forina, S. Lanteri, and C. Armanino. Parvus–an extendible package for data exploration, classification and correlation. 1991.

[22] Brendan J. Frey and Delbert Dueck. Clustering by passing messages between data points. *Science*, 315:972–976, 2007.

[23] Xin Geng, De-Chuan Zhan, and Zhi-Hua Zhou. Supervised nonlinear dimensionality reduction for visualization and classification. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 35(6):1098–1107, 2005.

[24] P. Güntert. Structure calculation of biological macromolecules from NMR data. *Quarterly reviews of biophysics*, 31(2):145–237, 1998.

[25] J. Ham, D. Lee, S. Mika, and Schölkopf B. A kernel view of the dimensionality reduction of manifolds. In *International Conference on Machine Learning*, 2004.

[26] Geoffrey E. Hinton, Peter Dayan, and Michael Revow. Modeling the manifolds of images of handwritten digits. *IEEE Transactions on Neural Networks*, 8:65–74, 1997.

[27] Philip K. Hopke and Dsir L. Massart. Reference data sets for chemometrical methods testing. *Chemometrics and Intelligent Laboratory Systems*, 19(1):35 – 41, 1993.

[28] Donald B. Johnson. Efficient algorithms for shortest paths in sparse networks. *J. ACM*, 24(1):1–13, January 1977.

[29] I. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 1986.

[30] Nandakishore Kambhatla and Todd K. Leen. Dimension reduction by local principal component analysis. *Neural Comput.*, 9:1493–1516, October 1997.

[31] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. Locally adaptive dimensionality reduction for indexing large time series databases. *SIGMOD Rec.*, 30(2):151–162, May 2001.

[32] Pooyan Khajehpour Tadavani and Ali Ghodsi. Learning an affine transformation for non-linear dimensionality reduction. In Jose Luis Balcazar, Francesco Bonchi, Aristides Gionis, and Michele Sebag, editors, *Machine Learning and Knowledge Discovery in Databases*, volume 6322 of *Lecture Notes in Computer Science*, pages 19–34. Springer Berlin Heidelberg, 2010.

[33] Pooyan Khajehpour Tadavani and Ali Ghodsi. Low-dimensional localized clustering (ldlc). In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 936–941, 2012.

[34] E. Kokiopoulou and Y. Saad. Orthogonal neighborhood preserving projections. In *ICDM '05: Proceedings of the Fifth IEEE International Conference on Data Mining*, pages 234–241, Washington, DC, USA, 2005. IEEE Computer Society.

[35] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

[36] John A. Lee and Michel. Verleysen. *Nonlinear dimensionality reduction*. Springer, New York; London, 2007.

[37] John Aldo Lee and Michel Verleysen. Nonlinear dimensionality reduction of data manifolds with essential loops. *Neurocomput.*, 67:29–53, 2005.

[38] R.C.T. Lee, J.R. Slagle, and H. Blum. A triangulation method for the sequential mapping of points from n-space to two-space. *Computers, IEEE Transactions on*, C-26(3):288 –292, march 1977.

[39] S. Lloyd. Least squares quantization in pcm. *Information Theory, IEEE Transactions on*, 28(2):129 – 137, 1982.

[40] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[41] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*, volume 1. Cambridge University Press Cambridge, 2008.

[42] James Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character*, 209:415–446, 1909.

[43] P. Niyogi. Locality preserving projections. *Advances in neural information processing systems*, 16:153–160, 2004.

[44] Lance Parsons, Ehtesham Haque, and Huan Liu. Subspace clustering for high dimensional data: a review. *SIGKDD Explor. Newsl.*, 6:90–105, June 2004.

[45] Karl Pearson. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[46] Frank Plastria, Steven Bruyne, and Emilio Carrizosa. Dimensionality reduction for classification. In Changjie Tang, CharlesX. Ling, Xiaofang Zhou, NickJ. Cercone, and Xue Li, editors, *Advanced Data Mining and Applications*, volume 5139 of *Lecture Notes in Computer Science*, pages 411–418. Springer Berlin Heidelberg, 2008.

[47] A. Plaza, P. Martinez, J. Plaza, and R. Perez. Dimensionality reduction and classification of hyperspectral image data using sequences of extended morphological transformations. *Geoscience and Remote Sensing, IEEE Transactions on*, 43(3):466–479, 2005.

[48] Nathalie Pochet, Frank De Smet, Johan A. K. Suykens, and Bart L. R. De Moor. Systematic benchmarking of microarray data classification: assessing the role of nonlinearity and dimensionality reduction. *Bioinformatics*, 20(17):3185–3195, 2004.

[49] Ronald Poppe. A survey on vision-based human action recognition. *Image and Vision Computing*, 28(6):976 – 990, 2010.

[50] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

[51] F.S. Samaria and AC Harter. Parameterisation of a stochastic model for human face identification. In *Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop on*, pages 138–142. IEEE, 1994.

[52] I. J. Schoenberg. Remarks to Maurice Fréchet's article "Sur la définition axiomatique d'une classe d'espace distanciés vectoriellement applicable sur l'espace de Hilbert". *Ann. of Math. (2)*, 36(3):724–732, 1935.

[53] B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, Cambridge, Massachusetts, 2002.

[54] Bernhard Schölkopf, Ralf Herbrich, and Alex J. Smola. A generalized representer theorem. In *Proceedings of the 14th Annual Conference on Computational Learning Theory and and 5th European Conference on Computational Learning Theory*, COLT '01/EuroCOLT '01, pages 416–426, London, UK, UK, 2001. Springer-Verlag.

[55] B. Shaw and T. Jebara. Minimum volume embedding. In Marina Meila and Xiaotong Shen, editors, *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics March 21-24, 2007, San Juan, Puerto Rico*, volume 2 of JMLR: W&CP, pages 460–467, March 2007.

[56] Blake Shaw and Tony Jebara. Structure preserving embedding. In Léon Bottou and Michael Littman, editors, *Proceedings of the 26th International Conference on Machine Learning*, pages 937–944, Montreal, June 2009. Omnipress.

[57] Le Song, Alex J. Smola, Karsten M. Borgwardt, and Arthur Gretton. Colored maximum variance unfolding. In *NIPS*, 2007.

[58] Gilbert W Stewart. On the early history of the singular value decomposition. *SIAM review*, 35(4):551–566, 1993.

[59] Jos F Sturm. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization methods and software*, 11(1-4):625–653, 1999.

[60] J. Tenenbaum. Mapping a manifold of perceptual observations. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *Advances in Neural Information Processing Systems 10*, volume 10, pages 682–687. MIT Press, 1998.

[61] Joshua B. Tenenbaum, Vin Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, December 2000.

[62] Michael E. Tipping and Christopher M. Bishop. Mixtures of probabilistic principal component analysers. *Neural Computation*, 11(2):443–482, 1999.

[63] Kim-Chuan Toh, Michael J Todd, and Reha H Tütüncü. Sdpt3a matlab software package for semidefinite programming, version 1.3. *Optimization Methods and Software*, 11(1-4):545–581, 1999.

[64] WarrenS. Torgerson. Multidimensional scaling: I. theory and method. *Psychometrika*, 17(4):401–419, 1952.

[65] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.

[66] R. Vidal. Subspace clustering. *Signal Processing Magazine, IEEE*, 28(2):52 –68, march 2011.

[67] R. Vidal, Yi Ma, and S. Sastry. Generalized principal component analysis (gpca). In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–621 – I–628 vol.1, june 2003.

[68] Dingding Wang, Chris H. Q. Ding, and Tao Li. K-subspace clustering. In *ECML/PKDD (2)*, pages 506–521, 2009.

[69] K. Q. Weinberger and L. K. Saul. Unsupervised learning of image manifolds by semidefinite programming. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR-04)*, volume II, pages 988–995, 2004.

[70] Kilian Q. Weinberger, Benjamin D. Packer, and Lawrence K. Saul. Nonlinear dimensionality reduction by semidefinite programming and kernel matrix factorization. In *Proceeding of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 381–388, 2005.

[71] K.Q. Weinberger, F. Sha, Q. Zhu, and L.K. Saul. Graph laplacian regularization for large-scale semidefinite programming. *Advances in neural information processing systems*, 19:1489, 2007.

[72] H. Whitney. Differentiable manifolds. *Ann. Math*, 37(3):645–680, 1936.

[73] Li Yang. Distance-preserving projection of high-dimensional data for nonlinear dimensionality reduction. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 26(9):1243–1246, 2004.

[74] Gale Young and A.S. Householder. Discussion of a set of points in terms of their mutual distances. *Psychometrika*, 3(1):19–22, 1938.

[75] Tianhao Zhang, Jie Yang, Deli Zhao, and Xinliang Ge. Linear local tangent space alignment and application to face recognition. *Neurocomputing*, 70(7-9):1547 – 1553, 2007.

[76] Zhenyue Zhang and Hongyuan Zha. Principal manifolds and nonlinear dimension reduction via local tangent space alignment. *SIAM Journal of Scientific Computing*, 26:313–338, 2002.