# Dimensionality Reduction

Wei-Lun Chao
weilunchao760414@gmail.com
First edited in December, 2011
Graduate Institute of Communication Engineering, National Taiwan University

## Contents:

## Notations:

General notation:

$a$ : scalar
$\boldsymbol{a}$ : vector
$A$: matrix
$a_i$ : the $i$th entry of $\boldsymbol{a}$
$a_{ij}$ : the entry $(i, j)$ of $A$
$\boldsymbol{a}^{(n)}$ : the $n$th vector $\boldsymbol{a}$ in a dataset
$A^{(n)}$ : the $n$th matrix $A$ in a dataset
$\boldsymbol{b}_k$ : the vector corresponding to the $k$th class in a dataset (or $k$th component in a model)
$B_k$ : the matrix corresponding to the $k$th class in a dataset (or $k$th component in a model)
$\boldsymbol{b}_k^{(i)}$ : the $i$th vector of the $k$th class in a dataset
$|A|$, $|A^{(n)}|$, $|B_k|$: the number of column vectors in $A$, $A^{(n)}$, and $B_k$

Special notation:

* In some conditions, some special notations will be used and desribed at those places.
Ex: $\boldsymbol{b}_k$ denotes a $k$-dimensional vector, and $B_{k\times j}$ denotes a $k \times j$ matrix

# 1. Introduction

In pattern recognition, data mining, and other kinds of data analysis applications, we often face high dimensional data. For example, in face recognition, the size of a training image patch is usually larger than $60 \times 60$, which corresponds to a vector with more than 3600 dimensions. Although as introduced in Section 錯誤! 找不到參照來源。, feature extraction based on the domain knowledge can be performed to explore more important information from the face patch and may result in a lower dimensional vector, usually the remaining dimensionality is still too high for learning. And for training data without explicit background or meaning, where domain knowledge cannot directly be performed, data-driven techniques for reducing the dimensionality of features are of high demand.

In recent face-related research topics, especially for face recognition and facial age estimation, dimensionality reduction (DR) plays a tremendous important role not only because features of these two topics are hard to define and usually of really high dimensionality, but also because they are both multi-class problems. For face recognition, there may be more than a hundred of people in the database and each person has no more than a dozen of images, which results in a multi-class classification problem with limited training set. As introduced in Section 錯誤! 找不到參照來源。, the feature dimensionality does affect the VC dimension, and under the condition of limited training data, the feature dimensionality should also be carefully considered in order to maintain generalization performance. For facial age estimation, each possible age could be seen as a class, so usually the problem is of over 60 classes, which also requires limited feature dimensionality to avoid over-fitting. Outstanding face recognition techniques such as eigenfaces [1], fisherfaces [2], Laplacianfaces [3], and face recognition based on independent component analysis (ICA) [4] exploit different kinds of dimensionality reduction methods to directly reduce the dimensionality of face patches. DR also widely used in facial age estimation. For example, the active appearance models (AAM) used in [5][6][7] is indeed a data-driven DR model, and the aging manifold proposed by Y. Fu [8][9] exploits nonlinear DR methods [10][11] to extract features from face patches. Because DR is so important in face-related topics, we make it a separate tutorial from the one of machine learning.

Compared to the feature extraction process introduced in Section 6 of the machine learning tutorial, the dimensionality reduction methods mentioned in this tutorial are fully data-driven (with some user defined metric or parameters), or it can be named as “data-driven” feature extraction. The tutorial is organized as follows: In

Section 2, the reasons for performing dimensionality reduction and some other fundamental concepts are presented, and an overview is given in Section 3. From Section 4 to Section 6, dimensionality reduction based on linear properties is discussed, and techniques based on nonlinear properties are presented in Section 7 to Section 11. In Section 12, we briefly overview techniques of feature selection and the conclusion is given in Section 13.

## 2. Why Dimensionality Reduction?

High dimensional data is often found in different disciplines when doing data analysis, and each disciplines may have its specific demand to perform dimensionality reduction. In this section, general reasons and requirements of DR are presented, both in practical and theoretical perspectives. Indeed, DR can focus only on the observed data set and ignores the generalization performance, as used in data compression.

### 2.1 Practical Reasons

- **Redundancy reduction and intrinsic structure discovery:** In multimedia researches, the processing data is naturally of high dimensionality, such as digital signals, digital audios, and digital videos. In these cases, each feature usually serves only as a simple measurement, while combining all of them can express a complicated concept or perception. And even if some features are missed, the concept could still be constructed from the remaining features, which means there is partial redundancy and dependency across features. In some applications, the existing redundancy is desired to be removed in order to save the usage of memory and the cost of transmission, such as image, audio, speech, video compression.

- **Intrinsic structure discovery:** Following the idea that there is some redundancy across features, in some other applications, these observed or measured features are assumed coming from a much lower dimensional intrinsic structure. And dimensionality reduction techniques are performed to discover the intrinsic structure or latent variables (the intrinsic parameters or degree of freedom) which can better explain a complicated phenomenon or concept, such as the use of independent component analysis (ICA) in multi-channel electroencephalographic (EEG) data [12], the use of probabilistic latent semantic analysis (PLSA) in text information retrieval [13].

- **Removal of irrelevant and noisy features:** During the feature extraction process, we desire to preserve as many information as possible from the original raw data. These extracted features seems meaningful from our perspective, while may not convey explicit correlations with the ongoing task. For example, the Haar feature is useful in the task of face detection, but not in the task of face recognition and age estimation. Besides, raw data and the feature extraction process may suffer from noise and degrades the performance of learning. Dimensionality reduction techniques have been used to solve these problems, for example, the Adaboost has intrinsic power to select the most relevant features for the ongoing task [14], and the PCA has been used to remove noise in face patches [1].

- **Feature extraction:** In many pattern recognition tasks, especially for the tasks of object recognition and classification, dimensionality reduction is used for feature extraction process. Compared to the reason of intrinsic structure discovery, the usage here doesn't explicitly assume the existing of an intrinsic parameter space, but want to simultaneously lower down the dimensionality of raw data and preserve the most compact representation of information from it. The use of DR techniques in face-related researches is mainly of this reason.

- **Visualization purpose:** One of the best ways for human to understand a concept and the relationship between feature samples is visualization in 2D or 3D representation, so how to preserve the structure in high dimensional feature space into 2D or 3D space is an interesting and challenging problem. The Isomap [15] and LLE [16] which will be introduced later show the ability to locate face patches in a 2D plane and preserve the appearance similarities.

- **Computation and Machine learning perspective:** For applications with real-time usage or with limited computational resources, dimensionality reduction is often required, although some information will get lost. And the last but probably the most important reason, the feature dimensionality has positive correlations with VC dimension and model complexity, so under limited training samples, we also need to limit the feature dimensionality to maintain the generalization performance.

## 2.2 Theoretical Reasons

The theoretical reason of performing dimensionality reduction comes from the perspective of probability and statistics. The Euclidean space have many great properties in low dimensional space, such as the use of Euclidean distance metric for similarity computation, and even human vision system measures distance in a Euclidean way. While in high dimensional spaces, these properties usually become failed and make strange and annoying phenomena, which are joint called the **"curse of dimensionality"** problem. In this subsection, we introduce several aspects of these strange phenomena, and the main reference is the textbook written by J. A. Lee [17].

- **Empty space phenomenon:** This phenomenon was first coined by Bellman [18] in connection with the difficulty of optimization by exhaustive enumeration on product spaces. Considering a Cartesian grid of spacing 1/10 on the unit cube in 10 dimensions, the number of point equals to $10^{10}$, while for a 20-dimensional cube, the number of points further increase to $10^{20}$. From Bellman's interpolation on this phenomenon, in the absence of simplifying assumptions (ex. the structure of distribution), the number of data samples required to estimate a function of several variables to a given accuracy on a given domain grows exponentially with the number of dimensions. More concretely, because the number of training samples is often limited, the high dimensional feature space is inherently sparse, which results in the difficulty of learning from data.

- **Hypervolume of cubes and spheres:** In a *d*-dimensional space, a sphere and the corresponding surrounding cube (all edges equal the sphere diameter) leads to volumes as shown below:

$$V_{sphere}(r) = \frac{\pi^{d/2} r^d}{\Gamma(1 + d/2)} \tag{1}$$

$$V_{cube}(r) = (2r)^d \tag{2}$$

$$\Gamma(n) = (n-1)! \text{ for } n \text{ is a positive integer} \tag{3}$$

, where $r$ is the radius of the sphere and $\Gamma$ is the gamma function. When $d$ increases, the ration $\frac{V_{sphere}(r)}{V_{cube}(r)}$ tends to zero:

$$\lim_{d\to\infty}\frac{V_{sphere}(r)}{V_{cube}(r)}=0 \tag{4}$$

, which means as the dimensionality increases, a cube becomes more spiky, and the spherical body gets smaller and smaller. Specifically, spike regions around the corners of a cube occupy almost all the available volume in high dimensional space.

- **Hypervolume of a thin spherical shell:** From the equations shown above, the relative hypervolume of a thin spherical shell is:

$$\frac{V_{sphere}(r)-V_{sphere}(r(1-\varepsilon))}{V_{sphere}(r)}=\frac{1^d-(1-\varepsilon)^d}{1^d} \tag{1.1}$$

, where $\varepsilon$ is the thickness of the shell ( $\varepsilon \ll$ ). When *d* increase, the ration tens to 1, meaning that the shell contains almost all the volume. This phenomenon implies that neighbor points of a specific point get far away with dimensionality grows.

- **Concentration of norms and distances:** As dimensionality grows, the contrast and discrimination power provided by usual metrics decreases, which is called the concentration phenomenon. Take the Euclidean norm and distance as an example, when *d* increases, random i.i.d (independent and identically distributed) vectors drawing from a specific distribution tends to have the same norm, and the Euclidean distance between any two vectors is approximately constant. In practice, the concentration phenomenon makes the nearest neighbor search more difficult in high-dimensional space.

- **Diagonal of a hypercube:** Consider the hypercube $[-1,1]^d$, any segment from its center to one of its $2^d$ corners (a half-diagonal) can be written as $\boldsymbol{v}=[\pm1,.....,\pm1]^T$.

  The angel between a $\boldsymbol{v}$ and the *i*th coordinate axis $\boldsymbol{e}^i=[0,.....,0,1,0,.....,0]^T$ is computed as:

$$\cos\theta_i=\frac{\boldsymbol{v}^T\boldsymbol{e}^i}{\left\|\boldsymbol{v}^T\right\|\left\|\boldsymbol{e}^i\right\|}=\frac{\pm1}{\sqrt{d}}. \tag{1.2}$$

As *d* grows, the cosine tends to zero, meaning that half-diagonals are nearly orthogonal to all coordinate axes. Then, when performing projection from high

dimensional space to only a 2 or 3 coordinate axes for visualization, samples lying near each diagonal line of the space will be plotted near the origin and mislead our perception.

Despite the difficulty to define a specific metric for the ongoing high-dimensional data, performing dimensionality reduction into a lower dimensional space and using general metrics for analysis seems to be a simpler way.

# 3. Dimensionality Reduction Overview

From the preceding section, we have seen several reasons that performing dimensionality is required, and in this section, overview of the dimensionality reduction techniques is presented, including different strategies, a brief introduction on topology and embedding, and different categorizations.

## 3.1 Strategies of Dimensionality Reduction

According to the reasons and execution methods, techniques of dimensionality reduction can be categorized into two strategies: feature transform and feature selection.

- **Feature transform:** Given a feature set $X = \{\boldsymbol{x}^{(n)} \in R^d\}_{n=1}^N$ with or without the label set, feature transform explores the dependencies among features, and finds a new set of transformed feature vectors $Z = \{\boldsymbol{z}^{(n)} \in R^p\}_{n=1}^N$ that not only has lower dimensionality ( $p \leq d$ ), but preserves the interesting and intrinsic characteristics of the original features. In this tutorial, the original *d*-dimensional space is called the (original) feature space, and the transformed *p*-dimensional space is called the transformed or reduced feature space. There are many different perspectives on what properties or content of the original feature set to be preserved in the transformed feature set, resulting in a variety of feature transform algorithms. In this thesis, feature transform is often called projection. There are two types of feature transform based on their goals, one is just to reduce the dimensionality, and the other one is to retrieve the latent variables and generally called latent variable separation. The second type sets additional constraints on the transformed feature space, such as statistical independence exploited in ICA. To distinguish from the same term used in Section 4.3 of the machine learning

tutorial, in this tutorial, we focus on "data-driven" feature transform, which means the parameters of transformation are learned from data.

- **Feature selection:** Compared to feature transform, feature selection aims at finding relevant features for the ongoing task. Instead of performing transformation to reduce the dimensionality, feature selection directly select a subset of the original features based on some criteria.

Both of these two strategies require the estimation of intrinsic or suitable dimensionality for reduction. Feature selection is usually combined with supervised learning tasks and determines the reduced dimensionality directly based on the learning performance, while feature transform is available for both labeled and unlabeled dataset, and serves as a separated preprocessing step in supervised learning tasks.

### 3.2 Topology and Embedding

This subsection is a brief summary of the Section 1.4 in [17]. From a geometrical point of view, when two or more variables depend on each other, their joint distribution does not span the whole space. Actually, the dependence induces some structure in the distribution, in the form of a geometrical locus that can be seen as a kind of object in the space. And DR, or more specifically the feature transform, aims at giving a new representation of these objects while preserving their structure.

In mathematics, topology studies the properties of objects that are preserved through deformation, twisting, and stretching. Tearing is the only prohibited operation, thereby guaranteeing that the intrinsic structure or connectivity of objects is not altered. For example, a circle is topologically equivalent to an ellipse. A topological object is represented or embedded as a spatial object in the feature space, and topology is used to abstract the intrinsic topological properties or structure, but ignore the detailed from of the objects. Two objects having the same topological properties are said to be homeomorphic.

The neighborhood of a point $\boldsymbol{x} \in R^d$, also called a $\varepsilon$-neighborhood or an infinitesimal open set, is often defined as the open $\varepsilon$-ball $B_\varepsilon(\boldsymbol{x})$ and contains points with Euclidean distance not larger than $\varepsilon$. A topology space is a set for which a topology is specified, and it can be defined using neighborhood and Haussdorf's axioms. A (topological) manifold $M$ is a topological space that is locally Euclidean, which means that around every point of $M$ is a neighborhood that is topologically the

same as the open unit ball in $R^d$. In general, any object that is nearly flat on small scale is a manifold.

An embedding is a representation of a topological object (a manifold, a graph, etc.) in a certain space (ex. $R^d$), in such a way that its topological properties are preserved. More generally, a space $A$ is embedded in another space $B$ when the topological properties of $B$ restricted to $A$ are the same as the properties of $A$. On the other hand, a smooth manifold, also called a differentiable manifold, is a manifold together with its "functional structure". The availability of parametric equations allows us to relate the manifold to its latent variables, and the dimensionality of these latent variables is the dimensionality of the manifold.

A smooth manifold $M$ without boundary is said to be a submanifold of another smooth manifold $N$ if $M \subset N$ and the identity map of $M$ into $N$ is an embedding, and the dimensionality of $M$ can be lower than the dimensionality of $N$. In the following sections, the word manifold typically designates an $m$-dimensional manifold embedded in $R^d$, and feature transform amounts to re-embedding a manifold from a high dimensional space $R^d$ into a lower dimensional space $R^p$. In dimensionality reduction, a manifold is nothing more than the underlying support of a data distribution, which is known only through the observed data.

### 3.3 Categorizations of Dimensionality Reduction

In this subsection, we discuss different ways to categorize dimensionality reduction techniques. These ways are mainly regard the purpose of the technique, its underlying model, and the mathematical criterion to be optimized. Some categorization methods are for both feature transform and feature selection and some for one of them.

- **Traditional vs. generative (feature transform):** Just like the learning strategies mentioned in Section 3.6 of the machine learning tutorial, DR also can be categorized into traditional and generative categories. The traditional methods is similar to the MRE strategy in supervised learning, which makes no assumption on how the original features are generated, but seeks transformation directly from the original features to the latent variables. In contrary, the generative methods assume a generative structure or process from the latent variables to the original features, and usually induce probability models for optimization. In general, the first method is simpler and more preferred in this tutorial.

- **Linear vs. Nonlinear (feature transform):** This categorization may or may not be the easiest to understand. The linear transform means transformation which can be performed by a matrix, and nonlinear is not (regardless the use of feature transform described in Section 錯誤! 找不到參照來源。 of the machine learning tutorial). While as will be seen later, these two terms are better referred to the properties preserved during transformation. For example, PCA tries to preserves the (linear) correlations as much as possible, while Laplacian eigenmap [19] aims at maintain the local neighborhood structure, which is nonlinear from the global view. Even if Laplacian eigenmap is linearized into Local preserving projection (LPP) [20] which can be executed through matrix operations, the properties to preserve is still nonlinear.

- **Continuous vs. discrete (feature transform):** The re-embed lower dimensional space can be either continuous such as LPP, PCA, and LDA or discrete like SOM [21]. Generally, the first category is preferred.

- **Implicit vs. explicit mapping (feature transform):** Given a feature set $X = \{\boldsymbol{x}^{(n)} \in R^d\}_{n=1}^N$, the explicit mapping means the feature transform directly find the reduced feature set $Z = \{\boldsymbol{z}^{(n)} \in R^p\}_{n=1}^N$, and the generalization to a new feature vector is difficult, usually accomplished by other mapping mechanism such as neural networks [22]. In contrary, implicit mapping learns a mapping function from the original dataset, and then this function is performed to determine the reduced feature vectors for both observed samples and new coming samples.

- **Layered vs. standalone embedding (both):** The layered embedding means that if a *p*-dimensional reduced feature set is reached, re-compute the dimensionality reduction with more or fewer reduced dimensionalities is just to add or remove several elements in each *p*-dimensional reduced vector, but values of the remained dimensions are the same. For example, if a vector $\boldsymbol{x}^{(n)} \in R^d$ is reduced to a 3D vector $[1,3,4]^T$, re-compute DR into a 2D or 4D vector will result in $[1,3]^T$ or $[1,3,4,a]^T$, where $a$ is a undefined value. Generally, all feature transform methods that translates the DR into an eigenproblem and assemble eigenvectors to form an embedding share this capability and are called spectral methods. In practice, we can learn layered-embedding DR mapping from

*d*-dimensional to *p*-dimensional, and then successively remove or add in features to modify the reduced dimensionality.

In contrary, methods without this property are called standalone embeddings, and for each reduced dimensionality *p*, the DR mapping should be completely computed again. This categorization can also be used for feature selection, see different kinds of subset selection methods in [23].

- **The type of criterion to be optimized (both):** Last but not the least, the criterion that guides the DR is probably the most important characteristic of a DR technique. For feature transform that based on geometrical considerations and manifold assumptions, preserving the similarities in the neighborhood through distance preservation or topology preservation are the two main criteria. Feature transforms with probabilistic models often optimizes with MLE or MAP, and there are still many different criteria focusing on statistical independency [4] and so forth. The optimization criteria of feature selection are remained to introduce in Section 12.

## 4. Principal Component Analysis (PCA)

Principal component analysis (PCA) is probably the most basic yet important feature transform in feature extraction and dimensionality reduction researches. PCA is widely-used not only because it is easy to train and perform, but also because it has several additional functionalities such noise reduction, ellipse fitting, and solutions for non-full rank eigenproblems, etc. There are several criteria for computing the projection bases (matrix) of PCA, such as maximum variance preserving with decorrelation and minimum mean square reconstruction error. These two criteria are both linear properties, and surprisingly, they lead to the same optimization process and results. In image and video compression researches, PCA is also called the Karhunen-Loeve transform (KL transform).

In Table 1, we present the basic model of PCA as well as these two criteria, and then in Table 2, the optimization method is shown. As will be seen, PCA is a layered, implicit, continuous, and unsupervised transformation, and there are different perspectives to judge whether it is a traditional or generative model. A more probabilistic form of PCA can referred to probabilistic PCA and factor analysis [24][25].

Table 1 Fundamentals of PCA

**Presetting:**

- Training set: $X = \{\boldsymbol{x}^{(n)} \in R^d\}_{n=1}^N$, denote $X$ as a $d \times N$ matrix with each column a sample of feature vector, and each row a feature.
- Denote $C_{xx}$ as the $d \times d$ covariance matrix of $\boldsymbol{x}$ in $X$, where

$$C_{xx} \approx \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{x}^{(n)} - \bar{\boldsymbol{x}})(\boldsymbol{x}^{(n)} - \bar{\boldsymbol{x}})^T = \frac{1}{N}XX^T \text{ with } \bar{\boldsymbol{x}} \approx \frac{1}{N}\sum_{n=1}^{N}x^{(n)}, \text{ and}$$

$$c_{xx(ij)} \approx \frac{1}{N}\sum_{n=1}^{N}(x_i^{(n)} - \bar{x_i})(x_j^{(n)} - \bar{x_j}).$$

**Transform model of PCA:**

- Desire to transform the original *d*-dimensional vector into *p*-dimensional ( $p \le d$ )
- Find a $d \times p$ transformation matrix $W$ **with a constraint** that $W^TW = I^p$, then $\boldsymbol{z} = W^T(\boldsymbol{x} - \bar{\boldsymbol{x}})$, where $\boldsymbol{z} \in R^p$.
- Each column of $W$ is a projection dimension.
- The reconstructed *d*-dimensional vector $\boldsymbol{x} = W\boldsymbol{z} + \bar{\boldsymbol{x}} = WW^T\boldsymbol{x} + \bar{\boldsymbol{x}}$
- Get a transformed set: $Z = \{\boldsymbol{z}^{(n)} \in R^p\}_{n=1}^N$, denote $Z$ as a $p \times N$ matrix.

**The first criterion: Maximum variance preserving with decorrelation**

- The transformed feature set $Z$ is desired to be decorrelated, which means the corresponding covariance matrix $C_{zz}$ should be a diagonal matrix:

$$C_{zz} \approx \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{z}^{(n)} - \bar{\boldsymbol{z}})(\boldsymbol{z}^{(n)} - \bar{\boldsymbol{z}})^T = \frac{1}{N}\sum_{n=1}^{N}(W^T\boldsymbol{x}^{(n)} - W^T\bar{\boldsymbol{x}})(W^T\boldsymbol{x}^{(n)} - W^T\bar{\boldsymbol{x}})^T.$$

- With the above constraint, we are solving the maximum variance preserving function based on $W$ as follows: ( $tr\{\cdot\}$ or $trace\{\cdot\}$ means the trace operator)

$$\begin{aligned}
W^* &= \arg\max_{W^TW=I}\frac{1}{N}\sum_{n=1}^{N}\left\|\boldsymbol{z}^{(n)} - \bar{\boldsymbol{z}}\right\|^2 = \arg\max_{W^TW=I}\frac{1}{N}\sum_{n=1}^{N}\left\|W^T\boldsymbol{x}^{(n)} - W^T\bar{\boldsymbol{x}}\right\|^2 \\
&= \arg\max_{W^TW=I}\frac{1}{N}\sum_{n=1}^{N}(W^T\boldsymbol{x}^{(n)} - W^T\bar{\boldsymbol{x}})^T(W^T\boldsymbol{x}^{(n)} - W^T\bar{\boldsymbol{x}}) \\
&= \arg\max_{W^TW=I}\frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{x}^{(n)} - \bar{\boldsymbol{x}})^TWW^T(\boldsymbol{x}^{(n)} - \bar{\boldsymbol{x}}) \\
&= \arg\max_{W^TW=I}\frac{1}{N}\sum_{n=1}^{N}tr\{(\boldsymbol{x}^{(n)} - \bar{\boldsymbol{x}})^TWW^T(\boldsymbol{x}^{(n)} - \bar{\boldsymbol{x}})\}
\end{aligned}$$

$$= \arg\max_{W^TW=I} \frac{1}{N}\sum_{n=1}^{N} tr\{W^T(\boldsymbol{x}^{(n)}-\bar{\boldsymbol{x}})(\boldsymbol{x}^{(n)}-\bar{\boldsymbol{x}})^T W\}$$
$$= \arg\max_{W^TW=I} tr\{W^T[\frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{x}^{(n)}-\bar{\boldsymbol{x}})(\boldsymbol{x}^{(n)}-\bar{\boldsymbol{x}})^T]W\}$$
$$= \arg\max_{W^TW=I} tr\{W^T C_{xx} W\}$$

- From the above derivation, we found that $C_{zz} = W^T C_{xx} W$, and we want to make $C_{zz}$ diagonal while maximize $trace\{W^T C_{xx}\}$.
- $W^*$ comes from the unit eigenvectors of $C_{xx}$, where $W^* = [\boldsymbol{v}_{C_{xx}}^{(1)}, \boldsymbol{v}_{C_{xx}}^{(2)}, ....., \boldsymbol{v}_{C_{xx}}^{(p)}]$ and $\boldsymbol{v}_{C_{xx}}^{(i)}$ is the *i*th largest eigenvectors of $C_{xx}$. Because $C_{xx}$ is symmetric semi-positive definite, rows of $W^*$ is naturally orthogonal to each other.

**The second criterion: Minimum mean square reconstruction error**

- The function to be minimized is shown as below:

$$W^* = \arg\min_{W^TW=I} \frac{1}{N}\sum_{n=1}^{N}\left\|\boldsymbol{x}^{(n)} - (W\boldsymbol{z}^{(n)} + \bar{\boldsymbol{x}})\right\|^2 = \min_{W^TW=I} \frac{1}{N}\sum_{n=1}^{N}\left\|\boldsymbol{x}^{(n)} - WW^T(\boldsymbol{x}^{(n)}-\bar{\boldsymbol{x}}) - \bar{\boldsymbol{x}}\right\|^2$$
$$= \arg\min_{W^TW=I} \frac{1}{N}\sum_{n=1}^{N}((I^N - WW^T)(\boldsymbol{x}^{(n)}-\bar{\boldsymbol{x}}))^T((I^N - WW^T)(\boldsymbol{x}^{(n)}-\bar{\boldsymbol{x}}))$$
$$= \arg\min_{W^TW=I} \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{x}^{(n)}-\bar{\boldsymbol{x}})^T(\boldsymbol{x}^{(n)}-\bar{\boldsymbol{x}}) - \frac{2}{N}\sum_{n=1}^{N}(\boldsymbol{x}^{(n)}-\bar{\boldsymbol{x}})^T WW^T(\boldsymbol{x}^{(n)}-\bar{\boldsymbol{x}})$$
- $$+ \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{x}^{(n)}-\bar{\boldsymbol{x}})^T W(W^TW)W^T(\boldsymbol{x}^{(n)}-\bar{\boldsymbol{x}})$$
$$= \arg\min_{W^TW=I} \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{x}^{(n)}-\bar{\boldsymbol{x}})^T(\boldsymbol{x}^{(n)}-\bar{\boldsymbol{x}}) - \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{x}^{(n)}-\bar{\boldsymbol{x}})^T WW^T(\boldsymbol{x}^{(n)}-\bar{\boldsymbol{x}})$$
$$\equiv \arg\max_{W^TW=I} \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{x}^{(n)}-\bar{\boldsymbol{x}})^T WW^T(\boldsymbol{x}^{(n)}-\bar{\boldsymbol{x}})$$
$$= ..... = \arg\max_{W} tr\{W^T C_{xx} W\}, \text{ from the derivation of first criterion}$$

- Surprisingly, after several lines of derivations, the function to be optimized equals to the one in criterion 1. Although there is no constraint that $C_{zz}$ in the second criterion should be a diagonal matrix, the constraint that $W^TW = I^p$ will make both two criteria with the same solution $W^*$.

PCA exploits the pairwise 2nd-order statistical properties (correlation) across features, which can be imagined as a less complex DR model compared to other DR techniques preserving higher-order statistics. In many applications, PCA is used as a preprocessing step to roughly reduce the dimensionality without additional assumptions or constraints [2][26]. And as will be mentioned soon, when facing eigenproblem with non-full rank matrices which usually happens in LDA and other DR techniques, PCA and SVD are common solutions to first lower down the dimensionality and then transfer the original null-full rank case into a full-rank case.

Table 2 The optimization methods of PCA

**Presetting and model:**

- The presetting and model is equal to the one in Table 1.
- **Pre-center the data set**, where each $\boldsymbol{x}^{(n)} \leftarrow \boldsymbol{x}^{(n)} - \bar{\boldsymbol{x}}$, and $X \leftarrow (X - \bar{\boldsymbol{x}}\mathbf{1}_d^T)$, where $\mathbf{1}_d$ is a $d \times 1$ all one vector for data centering.

**Learning method 1: eigendecomposition (EVD)**

- $V\Lambda = C_{xx}V$, where $\Lambda$ is a $d \times d$ diagonal matrix, where each diagonal element is the eigenvalues of $C_{xx}$ in the descending order, and $V$ is the $d \times d$ eigenvector matrix. There are at most $N-1$ nonzero eigenvalues, so usually $p \le N-1$.
- $W^* = [\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, ....., \boldsymbol{v}^{(p)}]$, where $\boldsymbol{v}^{(i)}$ is the $i$th largest unit eigenvectors of $C_{xx}$.
- Performing EVD on a symmetric semi-positive definite matrix such as $C_{xx}$ results in $\Lambda = V^T C_{xx} V$ and $V\Lambda V^T = C_{xx}$

**Learning method 2: singular value decomposition (SVD)**

- Perform SVD on $X$, we get $X = V\Sigma U^T$, where $V$ is $d \times d$, $\Sigma$ is $d \times N$ with singular in descending order, and $U$ is $N \times N$. Both $V$ and $U$ are orthonormal matrices.
- $W^* = [\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, ....., \boldsymbol{v}^{(p)}]$, where $\boldsymbol{v}^{(i)}$ is the $i$th column in $V$ with the $i$th largest singular value.
- An interesting relationship between **EVD** on $C_{xx}$ and **SVD** on $X$ is shown as:

$$XX^T = V\Sigma U^T (V\Sigma U^T) = V\Sigma U^T U\Sigma^T V^T = V(\Sigma\Sigma^T)V^T = V(N\Lambda)V^T = NC_{xx}$$

**Learning method 3: Presented in the eigenface paper 0**

- In this paper, the authors recommend to first compute the eigendecomposition of $\frac{1}{N}X^TX \in R^{N\times N}$, where $V^T\Lambda^T = (\frac{1}{N}X^TX)V^T$. Then, multiply both sides with $X$, we get $(XV^T)\Lambda^T = \frac{1}{N}XX^T(XV^T) \rightarrow V\Lambda = \frac{1}{N}XX^TV$ with $V = XV'$

**Comparison:**

- Compare these three methods, Method 1 deals with an $d \times d$ EVD, and Method 3 on an $N \times N$ EVD. In face recognition, $d$ is usually larger than $N$, so Method 3 is used for computing eigenfaces. While in practice, Method 2 is preferred. As the optimization goes, we only need to perform decomposition once and then select the desired number of eigenvectors for DR.

## 5. Linear Discriminant Analysis (LDA)

PCA is an unsupervised learning process, and in supervised learning tasks, we not only want to maintain the variances across features, but seek to preserve the relationships between features and labels. The linear discriminant analysis (LDA) [27][28], also called the Fisher's linear discriminant exploits this idea, taking the discrete label information into consideration and the reduced feature vectors are efficient for discriminant (classification). Table 3 describes the concepts as well as the optimization methods of LDA.

Table 3 Fundamental and optimization of LDA

**Presetting:**

- Training set: $X = \{\boldsymbol{x}^{(n)} \in R^d\}_{n=1}^N, Y = \{y^{(n)} \in L\}_{n=1}^N,\ L = \{l_1, l_2, ....., l_c\}$
- Follow the notation of $X$ and $Z$ used in Table 1.
- Define $X_i$ as the feature set containing all feature samples with label $l_i$, and the number of sample of label $l_i$ in $X_i$ is denoted as $N_i$.
- Define the class mean vector as $\boldsymbol{\mu}_i = \frac{1}{N_i}\sum_{\boldsymbol{x}^{(n)} \in X_i} \boldsymbol{x}^{(n)}$
- Define the total vector means as $\boldsymbol{\mu} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}^{(n)}$
- Define the between-class scatter matrix as $S_B = \sum_{i=1}^{c} N_i(\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T$

- Define within-class scatter matrix as $S_W = \sum_{i=1}^{c} \sum_{\boldsymbol{x}^{(n)} \in X_i} (\boldsymbol{x}^{(n)} - \boldsymbol{\mu}_i)(\boldsymbol{x}^{(n)} - \boldsymbol{\mu}_i)^T$

**Transform model of LDA:**

- Desire to transform the original *d*-dimensional vector into *p*-dimensional ( $p \le d$ )
- Find a $d \times p$ transformation matrix $W$, then $\boldsymbol{z} = W^T \boldsymbol{x}$, where $\boldsymbol{z} \in R^p$. Each column of $W$ is a projection dimension.
- Get a transformed set: $Z = \{\boldsymbol{z}^{(n)} \in R^p\}_{n=1}^{N}$, denote $Z$ as a $p \times N$ matrix.
- LDA doesn't seek to reconstruct the *d*-dimensional vectors.

**The criterion of LDA:**

- The between-class scatter matrix indicates the degree of class separation in the original feature space, and the within-class matrix indicates the distribution tightness of each class.
- The goal we are seeking during LDA is to find a $W$ that maximizes the ration of the transformed between-class scatter $W^T S_B W$ to the transformed within-class scatter $W^T S_W W$ in some sense, meaning that the transformed feature vectors in the same class are tightly clustered, while vectors in different classes are separated as far as possible.
- Determinant and trace are two popular scalar measures of the scatter matrix. In fisherface 0, determinant is used, while in 0, trace is chosen.
- Here we use determinant, and the function to be optimized is shown as below:

$$W^* = \arg\max_W \frac{\left|W^T S_B W\right|}{\left|W^T S_W W\right|}.$$

**Learning method:**

- If $S_W$ is nonsingular and $S_B$ and $S_W$ are both real symmetric positive semi-definite. The optimization problem can be transformed to a generalized eigenvector problem by Rayleigh quotient 0:

$$S_B \boldsymbol{v}^{(i)} = \lambda_i S_W \boldsymbol{v}^{(i)},\ i = 1, 2, ....., d \text{ and } \lambda_i \text{ is in descending order}$$

, and the desired $W = [\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, .....\boldsymbol{v}^{(p)}]$.

- Note that there are at most $c-1$ nonzero eigenvalues in the generalized eigenproblem because the rank of $S_B$ is at most $c-1$. So usually, the reduced dimensionality is chosen to be $p \le c-1$.

**Problems:**

- The rank of $S_W$ is at most $N-c$, and the matrix size is $d \times d$. In several applications such as face recognition, $d$ is larger than $N-c$ and $S_W$ is singular, meaning that Rayleigh quotient can't not be applied.
- There have been many researches focusing on solving the singular problem in LDA [2][29], and we introduce two of them.

**Solutions 1: PCA+LDA**

- Perform PCA with $d \times (N-c)$ $W_{PCA}$ to reduce the original *d*-dimensional feature vectors into ( $N-c$ )-dimensional, and then learn the $(N-c) \times p$ LDA projection matrix $W_{LDA}$.
- When a new *d*-dimensional vector $\boldsymbol{x}$ comes in, the reduced $\boldsymbol{z} = {W_{LDA}}^T {W_{PCA}}^T \boldsymbol{x}$.

**Solutions 2: Null space**

- Since we are searching $W^* = \arg\max_W \frac{\left|W^T S_B W\right|}{\left|W^T S_W W\right|}$, another solution collects the vectors in the null space of $S_W$ to build $W$, which results in zero denominator.

LDA is a layered, implicit, continuous, and supervised transformation. It also preserves the linear 2nd-order statistics of the original feature space. An interesting phenomenon is shown: performing determinant or trace on the scatter matrices result in the same solution if the Rayleigh quotient is available, and performing determinant in the PCA training brings the same solution. The comparison of PCA and LDA can be illustrated by a labeled data set, as shown in Fig. 1. As you can see, PCA finds the projection dimension that preserves the most variance, while LDA finds the one with the most discriminant power.

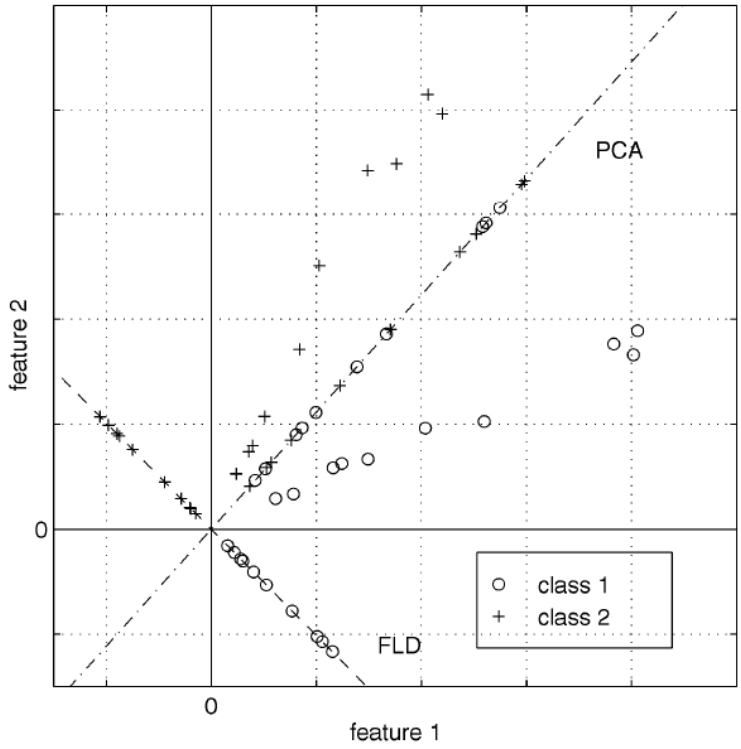


Fig. 1 The comparison between PCA and LDA. Perform DR from 2D into 1D, LDA (FLA) find the dimension with more discriminant power. [2]

## 6. Multidimensional Scaling (MDS)

Besides preserving the 2nd-order statistics such as correlation and scatter measurement between feature vectors, distance is the other criterion to maintain during DR. The multidimensional scaling actually hides a family of methods rather than a single well-defined procedure, and in this section we are mentioning the classical metric MDS. Scaling refers to methods that construct a configuration of points in a target metric space from information about inter-point distances, and MDS is the scaling when the target space is Euclidean. Actually, the classical metric MDS is not a true distance-preserving method but preserves pairwise scalar product (Euclidean inner product) of feature vectors. Moreover, classical metric MDS cannot achieve DR in nonlinear way either (scalar product is linear). Table 4 introduces the idea of metric MDS for data set with feature vectors, and Table 5 further presents the usage for data set with pairwise distance information. Finally, Table 6 shows another form of transferring pairwise distance into scalar product.

Table 4 The metric MDS on data set with feature vectors or scalar products

**Presetting:**

- Follow the notation of $X$ and $Z$ used in Table 1.
- **Pre-center the data set**, where each $\boldsymbol{x}^{(n)}$ is replaced by $\boldsymbol{x}^{(n)} - \bar{\boldsymbol{x}}$.
- The scalar product of two vectors is denoted as:

  $s_X(i,j) = s(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) = \left\langle \boldsymbol{x}^{(i)} \cdot \boldsymbol{x}^{(j)} \right\rangle$.

- Then, the matrix $S$ containing the information of pairwise scalar product is defined as: $S = [s_X(i,j)]_{1 \le i,j \le N} = X^T X$.
- Usually both $X$ and $Z$ are unknown, and only the matrix $S$ (Gram matrix) is given.

**Goal of MDS:**

- Desire to transform the original $d$-dimensional vector into $p$-dimensional ( $p \le d$ )
- Get a transformed set: $Z = \{\boldsymbol{z}^{(n)} \in R^p\}_{n=1}^N$, denote $Z$ as a $p \times N$ matrix.

**The criterion of classical metric MDS:**

- The pairwise scalar product is desired to preserve after DR, where $S \approx Z^T Z$ .
- The criterion of metric MDS is:

  $$Z^* = \arg\min_Z \sum_{i=1}^N \sum_{j=1}^N (s_{ij} - < \boldsymbol{z}^{(i)} \cdot \boldsymbol{z}^{(j)} >)^2 = \arg\min_Z \left\| S - Z^T Z \right\|_{L^2}^2$$

  , where $\|A\|_{L^2}$ is the $L^2$ matrix norm, also called Frobenius matrix norm, $\sqrt{\sum_{i,j} a_{ij}^2}$.

**Learning method:**

- To solve this problem, we first perform eigendecomposition (EVD) on $S$:

  $S = U \Lambda U^T = (U \Lambda^{1/2})(\Lambda^{1/2} U^T) = (\Lambda^{1/2} U^T)^T (\Lambda^{1/2} U^T) \approx Z^T Z$

  , where $U$ is an $N \times N$ orthonormal eigenvector matrix and $\Lambda$ is an $N \times N$ diagonal eigenvalues matrix with at most $d-1$ nonzero eigenvalues. Eigenvalues in $\Lambda$ are sorted in descending order.

- The reduced data set $Z$ is then denoted as $Z = \Psi I_{p \times N} \Lambda^{1/2} U^T$, where $I_{p \times N}$ is a non-square identity matrix to select the first $p$ eigenvectors of $U$ and $\Psi$ is an

arbitrary $p \times p$ orthonormal matrix for rotation.

- From the above process, MDS is actually an explicit feature transform which directly finds the reduced data set $Z$ instead of the transformation matrix $W$.

**Connection between PCA and classical metric MDS:**

- The SVD of $X$ is denoted as $X = V\Sigma U^T$.
- PCA decomposes the covariance matrix $C_{xx}$ (centered $X$) as below:

$$C_{xx} \approx \frac{1}{N} XX^T = \frac{1}{N} V\Sigma U^T U\Sigma^T V = V\frac{\Sigma\Sigma^T}{N} V = V(\Lambda_{PCA})V$$

, and the resulting $Z_{PCA} = (I_{p\times d} V^T)X$.

- On the other hand, metric MDS decomposes the Gram matrix $S$:

$$S = X^T X = U\Sigma^T V^T V\Sigma U^T = U\Sigma^T\Sigma U^T = U\Lambda_{MDS}U^T$$

, and the solution is $Z_{MDS} = \Psi I_{p\times N}\Lambda_{MDS}^{1/2} U^T$.

- Discard $\Psi$ and take several derivations for both $Z_{PCA}$ and $Z_{MDS}$ as below:

$$Z_{MDS} = I_{p\times N}\Lambda_{MDS}^{1/2} U^T = I_{p\times N}(\Sigma^T\Sigma)^{1/2} U^T = I_{p\times d}\Sigma U^T$$
$$Z_{PCA} = (I_{p\times d} V^T)X = I_{p\times d}(V^T V)\Sigma U^T = I_{p\times d}\Sigma U^T$$

, we actually see that both techniques come to the same solution and the criteria are intrinsically the same.

- Comparing them, the metric MDS solves the eigenproblem of an $N \times N$ matrix while PCA of a $d \times d$ matrix.

Table 5 The metric MDS on data set with information of pairwise distances

**Presetting:**

- Follow the notations of $X$, $Z$ and $S$ used in Table 4.
- Instead of given the Gram matrix $S$ or $X$, now we are given a matrix $D_2$ containing the pairwise square (Euclidean) distance between feature vectors of $X$:

$$D_2 = [d_X^2(i,j)]_{1\le i,j\le N}.$$

- In order to perform MDS, $D_2$ should be transformed into the form of $S$.

**From square distance to scalar product:**

- Assume the unknown $X$ is centered (translation doesn't affect distance).

$$d_X^2(i,j) = \left\| \boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)} \right\|^2 = < \boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}, \boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)} > = s_X(i,i) - 2s_X(i,j) + s_X(j,j)$$

$$s_X(i,j) = -\frac{1}{2}(d_X(i,j) - s_X(i,i) - s_X(j,j))$$

- The two scalar products in the right hand side are unknown while can be achieved by the **double centering** of $D_2$, and $S$ can be reached:

$$S = -\frac{1}{2}(D_2 - \frac{1}{N} D_2 \mathbf{1}_N \mathbf{1}_N^T - \frac{1}{N} \mathbf{1}_N \mathbf{1}_N^T D_2 + \frac{1}{N^2} \mathbf{1}_N \mathbf{1}_N^T D_2 \mathbf{1}_N \mathbf{1}_N^T).$$

  This operation subtracts from each entry of $D_2$ the means of the corresponding row and column, and adds back the mean of all entries of $D_2$.

**Unified learning method:**

- If a data set $X$ is given, center it, and then either perform PCA to get $Z$ or perform metric MDS on $S = X^T X$ to get Z.
- If the data set is in the form of $S$, perform metric MDS to Z.
- If the data set is in the form of pairwise distance, compute the squared distance matrix $D_2$, perform double centering to get $S$, and then perform metric MDS to Z.

Table 6 Another form of metric MDS

**Presetting:**

- In the original Isomap paper [15], the process of metric MDS looks different, while it does results in the same solution $Z$.
- Follow the notation in Table 5, and assume the pairwise distance matrix $D$ is given: $D_X = [d_X(i,j)]_{1 \le i,j \le N}$. (not square)

**The criterion of classical metric MDS:**

- $Z^* = \arg\min_Z \left\| \tau(D_X) - \tau(D_Z) \right\|_{L^2}$, where $D_Z$ is the distance matrix generated by reduced feature set $Z$.
- $\tau(\cdot)$ is the operator to convert distances to scalar products:

$$\tau(D) = -HD_2H / 2$$

  , where $H$ is the centering matrix with $h_{ij} = \delta(i,j) - 1/N$ and $D_2 = [d_X^2(i,j)]_{1 \le i,j \le N}$.

- With simple derivations, we show that $\tau(D)$ is exactly the Gram matrix S.

$$\tau(D) = -HD_2H/2 = -(I_{N\times N} - \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^T)D_2(I_{N\times N} - \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^T)/2$$
$$= -(D_2 - \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^T D_2 - D_2\frac{1}{N}\mathbf{1}_N\mathbf{1}_N^T + \frac{1}{N^2}\mathbf{1}_N\mathbf{1}_N^T D_2\frac{1}{N}\mathbf{1}_N\mathbf{1}_N^T)/2 = S$$

**Learning method:**

- In [15], $Z$ comes from the first $p$ $N$-dimensional eigenvectors of $\tau(D)$, which is exactly the same as done in Table 4.

Metric MDS is an unsupervised, layered, continuous, and explicit feature transform. Compared to PCA, the input data form is more flexible, both scalar product and pairwise distance are available. There are several extensions on metric MDS, either aim at performing MDS on new coming samples [30], inducing nonlinear properties, or try to make MDS into non-metric formulation [31][32]. Besides, following this distance preservation idea, well-known techniques such as Sammon's nonlinear mapping [33] and curvilinear component analysis (CCA) [34] searches another criteria and optimization methods which do preserve distance rather than scalar product.

## 7. Isomap and Its Extensions

The two unsupervised and one supervised DR techniques mentioned in the previous three sections all preserve or exploit linear properties (scalar product, covariance, and scatter matrix). These methods have an implicit assumption: the original $d$-dimensional space is a Euclidean space, meaning that the Euclidean operator such as scalar product and $L^2$ distance are meaningful in the data distribution. While as introduced in Section 3.2, if there is a lower dimensional latent space that controls the original feature space, the observed data set or data distribution then contains some structures and cannot span the whole feature space. If so, preserving Euclidean properties misleads the DR process. To solve this problem, nonlinear dimensionality reduction techniques, are proposed and developed, especially in the last ten years. In this and the next five sections, we discuss six different kinds of manifold learning based on the modeling, the criterion to be optimized, and the optimization methods.

In 2000, two outstanding manifold learning methods, Isomap [15] and LLE [16], are published. The criteria of them are different, while they both provide ability to discover the latent space which is nonlinearly embedded in the original feature space.

Isomap exploits the idea of pairwise geodesic **distance** and tries to preserve these distances after DR. On the other hand, LLE exploits the relationships between each point and its **neighbors**, and aims at maintaining these relationships after DR. Because both these methods assume that the high dimensional space is locally smooth or locally Euclidean, which is equivalent to the definition of manifold, so nonlinear dimensionality reduction based on this assumption is also called manifold learning. Many manifold learning methods published after them took the same assumption and preserved one or both the properties with some further improvement [35][36]. As milestones in manifold learning, we first introduce these two techniques, Isomap first, and LLE later.

### 7.1 Isomap: Isometric Feature Mapping

Isomap is published by J. B. Tenenbaum et al. [15]. To better describe the underlying structure, Isomap tries to find a new distance metric in the original feature space, and the geodesic distance seems to be a good idea, which measures distance between two points along the topology structure embedded in the feature space. In their method, the geodesic distance between neighboring points is approximated by the Euclidean distance in $R^d$, and for faraway points, the geodesic distance is approximated by adding up a sequence of "short hops" between neighbor points. This processing can be thought of having a graph with edges defined only between neighbor points, the geodesic distance between faraway points is approximated by the shortest path on the graph. As a consequence, the geodesic distance is also called the graph distance in manifold learning.

After calculating the geodesic distance for all pair of feature vectors in the data set $X = \{\boldsymbol{x}^{(n)} \in R^d\}_{n=1}^N$, Isomap aims at preserving these distance in the reduced feature set $Z = \{\boldsymbol{z}^{(n)} \in R^p\}_{n=1}^N$, and that why its name comes from (isometric means distance preserving). In order to compute the distance in *Z*, the distance metric in the reduced feature space should also be defined. In general, these kinds of distance preserving methods assume the reduced space or more specifically the space for re-embedding as a Euclidean space, so the distance in the reduced space can be simply computed. As mentioned in Section **錯誤! 找不到參照來源。**, the MDS is used for Euclidean reduced space, so it is a suitable choice for distance preserving DR. In Table 7, we summarize the algorithm of Isomap.

Table 7 Algorithm and criterion of Isomap

**Presetting:**

- Given a training set $X = \{\boldsymbol{x}^{(n)} \in R^d\}_{n=1}^N$
- Define a $N \times N$ matrix $D$ where $[d_{ij} = \infty]_{1 \le i,j \le N}$
- The set containing all the neighbors of $\boldsymbol{x}^{(i)}$ is denoted as $Nei(i)$ (undefined yet)
- There are generally two definitions of geodesic neighbors:

  (1) $\varepsilon$-neighbors: $\boldsymbol{x}^{(j)} \in Nei(i)$ if $\left\|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\right\| \le \varepsilon$

  (2) $K$-nearest neighbors: $\boldsymbol{x}^{(j)} \in Nei(i)$ if $\boldsymbol{x}^{(j)} \in KNN(i)$, or $\boldsymbol{x}^{(i)} \in KNN(j)$, where $KNN(i)$ denotes the set containing $K$-nearest neighbors of $\boldsymbol{x}^{(i)}$ in the sense of Euclidean distance.

**Goal:**

- Desire to transform the original $d$-dimensional vector into $p$-dimensional ($p \le d$)
- Get a transformed set: $Z = \{\boldsymbol{z}^{(n)} \in R^p\}_{n=1}^N$, denote $Z$ as a $p \times N$ matrix.
- There are three steps of Isomap

**Step 1: Geodesic distance in the neighborhood**

- Algorithm

```
for i = 1 : N
    for j = 1 : N
        if (x^(j) ∈ Nei(i) and i ≠ j)
            d_ij = ||x^(i) − x^(j)||
        end
    end
end
```

**Step 2: Geodesic distance between faraway points**

- In [15], Floyd's algorithm is chosen to find the shortest path between pairs of points.

- Algorithm

$D(0) \leftarrow D$

for $t = 1:\infty$

  $D(t) \leftarrow D(t-1)$

  for $i = 1:N$

    for $j = 1:N$

      for $k = 1:N$

        $d_{ij} \leftarrow \min\{d_{ij}, d_{ik} + d_{kj}\}$

      end

    end

  end

  if $(D(t) = D(t-1))$

    return $D \leftarrow D(t)$

    break

  end

end

**Step 3: MDS (**as mentioned in Table 6**)**

- Algorithm

(1) First compute $\tau(D) = -HD_2H/2$

, where $H$ is the centering matrix with $h_{ij} = \delta(i,j) - 1/N$ and $D_2 = [d_{ij}^2]_{1 \le i,j \le N}$

(2) Perform EVD on $\tau(D)$, $\tau(D) = U\Lambda U^T = (U\Lambda^{1/2})(\Lambda^{1/2}U^T) = (\Lambda^{1/2}U^T)^T(\Lambda^{1/2}U^T)$

, where eigenvalues in $\Lambda$ are sorted in descending order

(3) $Z = I_{p \times N}\Lambda^{1/2}U^T$

As shown in Table 7, Isomap is an explicit, layered, and unsupervised feature transform. In the original paper Isomap, the Isomap is used mainly for visualization purpose, and examples such as face images, handwritten digits, and the well-know Swiss roll data set show its great power on discovering the latent space. Because it is explicit, there is no mapping function trained during DR learning, and DR for new coming feature vector is originally unavailable. While based on $X$ and $Z$ at hand, we could use supervised learning techniques such as multi-layer perceptron to learn the multiple mapping relationship between these two sets [35][37].

### 7.2 Extensions of Isomap

There are many extensions of Isomap. In [38], V. D. Silva provides two new Isomap formulations, conformal Isomap (C-Isomap) and landmark Isomap (L-Isomap). C-Isomap exploits the idea of conformal embedding. Conformal embedding preserves the angels but not lengths in local geometry, which means the local geodesic distance is free to multiply a scaling factor. In order to define the scaling factors, the authors assumed that the reduced feature space $Z$ is uniformly sampled, and then reformulated the computation of local geodesic distance to meet their assumption. From their experimental results, C-Isomap performs better than Isomap in learning the structure of certain curved manifolds. On the other hand, L-Isomap is proposed to take advantage of LLE and Laplacian eigenmap, where the matrix to perform EVD is sparse. Towards this goal, the authors reformulated the computation of MDS into a new procedure called landmark MDS (LMDS) [39]. Within LMDS, a set of $M$ ($M \ll$ ) landmark points are first mapped into the reduced feature space by classical metric MDS based on their pairwise geodesic distances. Then, the remaining points are embedded with a simple computation according to their geodesic distances to the landmark points. The LMDS reduces the computational complexity of the third step in Isomap from $O(N^3)$ to $O(M^2N)$. Also in this paper, the authors provided several key notes of manifold learning, such as the ways of embedding and the properties to preserve during DR.

In [35], M. Vlachos et al. proposed several modifications on Isomap (as well as LLE). When the original feature set $X$ is naturally clustered into several groups, Isomap and LLE don't perform well in their experiments. To solve this problem, they proposed to change the neighbor finding step of both Isomap and LLE, where the $K$-nearest neighbors are replaced by $K$/2-nearest and $K$/2-farthest neighbors. The experimental results showed this modification did preserve the clustering nature of $X$. And to make Isomap supervised, which means the label information is considered and feature samples with the same label are desired to be close in $Z$, a new Isomap procedure, WeightedIso, was proposed. After finding out the $K$-nearest neighbors of each sample, WeightedIso reduces the local Euclidean distance between points with the same label by a scaling factor, and then compute the global geodesic and MDS with the same procedure of Isomap. For new coming samples, the RBF neural network is used to learn the mapping from $X$ to $Z$. The experimental results showed that the combination of WeightedIso and the $K$NN classifier achieves comparable multi-class classification results with some existing multi-class classifiers while requires less computational time.

Based on the idea of WeightedIso, X. Geng [37] later proposed a new distance scaling function for supervised Isomap, called S-Isomap. In their assumption, samples with the same label are intrinsically close to each other in local geometry in both the original and reduced feature space. While with some possible noise, this intrinsic property is distorted in the observed data *X*. To take noise into consideration, instead of using a linear scaling factor to reduce the local geodesic distance between samples with the same label, they proposed a pair of nonlinear scaling functions, one for neighbors with the same label and the other one for neighbors with different labels. Based on these nonlinear scaling functions, in the neighborhood of a sample, geodesic distance of neighbors with the same label will always be smaller than neighbors with different labels. In their experiments, S-Isomap performed better than Isomap, WeightedIso, and LLE in supervised visualization tasks. Besides, S-Isomap achieved comparable classification results with WeightedIso, SVM, decision tree, etc.

Besides considering label information by scaling factors or functions, M. H. Yang [40][41] proposed another idea to extend Isomap for classification problem. After the first two steps in the original Isomap to compute the geodesic-distance matrix *D*, he exploited each row of *D* as the new feature vector of each sample. LDA is then performed on these new feature vectors to extract projection bases for supervised dimensionality reduction. For a new test example, the pairwise geodesic distance to *N* training samples are computed, then the trained LDA matrix are used to compute the reduced feature vector. In fact, there is no MDS computing in Yang's method, and it is probably the first extension of Isomap for supervised learning. The experimental results showed that Yang's method outperforms Isomap in classification problem.

# 8. Locally Linear Embedding and Its Extensions

## 8.1 Locally Linear Embedding (LLE)

Compared to Isomap, the locally linear embedding (LLE) [16] exploits another idea to preserve the structure in the original feature space: the **neighborhood preservation**. Based on the same assumption that in a small scale of neighborhood the local distance metric is Euclidean in $R^d$ , LLE preserves the local relationship, also called the local geometry, of each sample with its neighbors while ignores the global geometry in large scale. The authors claimed that if the manifold is well-sampled with sufficient data, each point and its neighbors are expected to lie or closed to a locally linear patch of the manifold, and the local geometry of these patches can be

characterized by **linear coefficients** that reconstruct each sample from its neighbors. The linear coefficients can be computed in a constraint least square fashion, and it is robust to translation, scaling, and rotation of each point and its neighbors.

To re-embed the high dimensional structure into a low dimensional space, the reduced space is assumed to be locally smooth, meaning that the local linear geometry characterized by reconstruction coefficients in the original feature space is still meaningful in the reduced space. Based on this assumption, LLE re-embeds these local patches into the low dimensional global coordinate with neighborhood preservation, and the mean square error of the linear reconstruction coefficients measured in the original feature space to the reduced feature space is a valid evaluation to judge the quality of DR. The algorithm of LLE is summarized in Table 8, and the theoretical justifications are presented in Table 9.

LLE is layered, explicit, and unsupervised. It provided a different model assumption based on manifold properties to perform nonlinear dimensionality reduction. Instead of preserving all pairs of geodesic distance between training samples, the formulation of LLE only preserves the local geometry through linear reconstruction coefficients. The experimental results in 0 showed that LLE is able to nonlinearly discover the latent variables. In addition, LLE illustrates a general principle of manifold learning that even if the global geometry is not explicitly preserved, with overlapped local neighborhoods, the global geometry can still be reconstructed (or say generated) in the reduced feature space. The matrix for recording neighbor information of LLE is sparse, where most of the entries are zero. This property does facilitate the computation of EVD in LLE against the one in Isomap.

Form the discussions in 0, manifold learning is intrinsically an ill-posed problem and requires some additional restrictions to make it solvable. Two possibilities of the restrictions are presented, one is isometric embedding and the other one is conformal embedding. The isometric embedding such as Isomap preserves infinitesimal (local) lengths and angles, while the conformal embedding such as LLE only preserves angles. From another perspective, LLE is casted as a local approach of nonlinear dimensionality reduction which attempts to preserve the local geometry of data. Local approaches only seek to map nearby points on the manifold to nearby points in the reduce feature space. On the other hand, Isomap is denoted as a global approach, which attempts to preserve geometry at all scales based on geodesic distance. Global approaches not only map nearby points on the manifold to nearby points in the reduced feature space, but also map faraway points to faraway points.

Table 8 The algorithm and criterion of LLE

**Presetting:**

- Given a training set $X = \{\boldsymbol{x}^{(n)} \in R^d\}_{n=1}^N$, denote $X$ as a $d \times N$ matrix.
- Define a $N \times N$ matrix $W$ to record the linear reconstruction coefficients, where the $i$th row $[w_{ij} = 0]_{1 \le j \le N}$ is the initial coefficients of sample $i$ from its neighbors.
- The set containing all the neighbors of $\boldsymbol{x}^{(i)}$ is denoted as $Nei(i)$ (undefined yet)
- There are generally two definitions of neighbor:

  (1) $\varepsilon$ -neighbor: $\boldsymbol{x}^{(j)} \in Nei(i)$ if $\left\|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\right\| \le \varepsilon$

  (2) $K$-nearest neighbors: $\boldsymbol{x}^{(j)} \in Nei(i)$ if $\boldsymbol{x}^{(j)} \in KNN(i)$ , or $\boldsymbol{x}^{(i)} \in KNN(j)$, where $KNN(i)$ denotes the set containing $K$-nearest neighbors of $\boldsymbol{x}^{(i)}$ in the sense of Euclidean distance.
- Define $N_i = \left|Nei(i)\right| = \#$ neighbors of $\boldsymbol{x}_i$.

**Goal:**

- Desire to transform the original $d$-dimensional vector into $p$-dimensional ( $p \le d$ )
- Get a transformed set: $Z = \{\boldsymbol{z}^{(n)} \in R^p\}_{n=1}^N$, denote $Z$ as a $p \times N$ matrix.
- If we choose to use $K$-nearest neighbor, then generally $K > p$ to locate $z$ uniquely.
- There are three steps of LLE

**Step 1: Find the neighbor of each sample**

- Algorithm

for $i = 1 : N$

define a $0 \times 1$ vector $\boldsymbol{h}^{(i)}$ to record the neighbor indices of $\boldsymbol{x}^{(i)}$

define a $d \times 0$ matrix $\Gamma^{(i)}$ to record the neighbor vectors of $\boldsymbol{x}^{(i)}$

define a $N_i \times 1$ vector $\boldsymbol{\eta}^{(i)}$ to record the coefficients of $\boldsymbol{x}^{(i)}$ coming from $\Gamma^{(i)}$

for $j = 1 : N$

if ($\boldsymbol{x}^{(j)} \in Nei(i)$ and $i \neq j$)

$\boldsymbol{f}^{(i)} \leftarrow [\boldsymbol{f}^{(i)T}\ j]^T$

$\Gamma^{(i)} \leftarrow [\Gamma^{(i)}\ \boldsymbol{x}^{(j)}]$

end

end

end

**Step 2: Compute the linear reconstruction coefficients**

- After defining the neighbors of each sample, now we want to compute the $N \times 1$ linear coefficients $\boldsymbol{w}^{(i)}$ for each sample $i$ with least reconstruction error:

$$\boldsymbol{w}^{(i)} = \arg\min_{\boldsymbol{w}} \left\| \boldsymbol{x}^{(i)} - X\boldsymbol{w}^{(i)} \right\|^2 .$$

- Two constraints are set on $\boldsymbol{w}^{(i)}$ :

(1) $\boldsymbol{w}_j^{(i)} = 0,\ \text{if } j \neq Nei(i)$ (neighborhood constraint)

(2) $\sum_{j=1}^{N} \boldsymbol{w}_j^{(i)} = 1$ (for translation invariant)

- Algorithm

for $i = 1 : N$

$C^{(i)} \leftarrow (\boldsymbol{x}^{(i)}\mathbf{1}^T - \Gamma^{(i)})^T (\boldsymbol{x}^{(i)}\mathbf{1}^T - \Gamma^{(i)})$

$\boldsymbol{\eta}^{(i)} \leftarrow \dfrac{C^{(i)-1}\mathbf{1}}{\mathbf{1}^T C^{(i)-1}\mathbf{1}}$

for $j = 1 : N_i$

$w_{h_j^{(i)}}^{(i)} \leftarrow \eta_j^{(i)}$

end

end

$W = \left[ \boldsymbol{w}^{(1)}, \boldsymbol{w}^{(1)}, ....., \boldsymbol{w}^{(N)} \right]^T$

**Step 3: Re-embedding in the reduced feature space**

- After the previous step, now we get a matrix $W$ which contains the information of local linear reconstruction coefficients. Now we also expect these coefficients are preserved in the reduced feature space $Z$. To achieve this, the following function is desired to be minimized:

$$\Phi(Z)=\sum_{i=1}^{N}\left\|z^{(i)}-\sum_{j=1}^{N}w_{ij}z^{(j)}\right\|^2$$

- Two constraints are made on $Z$ to avoid degradation or trivial solution:

  (1) $\sum_{n=1}^{N}\boldsymbol{z}^{(n)}=\mathbf{0}$

  (2) $\frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{z}^{(n)}-\bar{\boldsymbol{z}})(\boldsymbol{z}^{(n)}-\bar{\boldsymbol{z}})^T=\frac{1}{N}\sum_{n=1}^{N}\boldsymbol{z}^{(n)}\boldsymbol{z}^{(n)T}=ZZ^T=I$

- Algorithm:

  (1) Define a $N\times N$ matrix $M=(I_{N\times N}-W)^T(I_{N\times N}-W)=(I_{N\times N}-W^T-W+W^TW)$

  (2) Perform EVD on $M$, $MV=V\Lambda$, $M=V\Lambda V^T$, where $\Lambda$ is in descending order.

  (3) $Z=\left[\boldsymbol{v}^{(N-p)},\boldsymbol{v}^{(N-p+1)},.....,\boldsymbol{v}^{(N-1)}\right]^T=V\left[O_{p\times(N-1-p)}\mid I_{p\times p}\mid O_{p\times 1}\right]^T$

  , where $O_{p\times(N-1-p)}$ means a $p\times(N-1-p)$ zero matrix

Table 9 The theoretical justifications of LLE

**The computation of linear reconstruction coefficients**

- Follow the notations and constraints in Table 8
- The original function to be minimized in this step is:

$$E(W)=\sum_{i=1}^{N}\left\|\boldsymbol{x}^{(i)}-\sum_{j=1}^{N}w_{ij}\boldsymbol{x}^{(j)}\right\|_{L^2}^2.$$

This function can be separated into $N$ sub-functions:

$$e(\boldsymbol{w}^{(i)})=\left\|\boldsymbol{x}^{(i)}-X^{(i)}\boldsymbol{w}^{(i)}\right\|^2.$$

- With the constraints on $\boldsymbol{w}^{(i)}$, $e(\boldsymbol{w}^{(i)})$ can be rewritten with a Lagrange multiplier:

$$\begin{aligned}e(\boldsymbol{\eta}^{(i)},\lambda)&=\left\|\boldsymbol{x}^{(i)}-\Gamma^{(i)}\boldsymbol{\eta}^{(i)}\right\|^2-\lambda(\mathbf{1}^T\boldsymbol{\eta}^{(i)}-1)=\left\|\sum_{m=1}^{N_i}\eta_m^{(i)}\boldsymbol{x}^{(i)}-\Gamma^{(i)}\boldsymbol{\eta}^{(i)}\right\|^2-\lambda(\mathbf{1}^T\boldsymbol{\eta}^{(i)}-1)\\&=\left\|\sum_{m=1}^{N_i}\eta_m(\boldsymbol{x}^{(i)}-\boldsymbol{x}^{(h_m^{(i)})})\right\|^2-\lambda(\mathbf{1}^T\boldsymbol{\eta}^{(i)}-1)=\left\|(\boldsymbol{x}^{(i)}\mathbf{1}^T-\Gamma^{(i)})\boldsymbol{\eta}^{(i)}\right\|^2-\lambda(\mathbf{1}^T\boldsymbol{\eta}^{(i)}-1)\\&=\boldsymbol{\eta}^{(i)T}(\boldsymbol{x}^{(i)}\mathbf{1}^T-\Gamma^{(i)})^T(\boldsymbol{x}^{(i)}\mathbf{1}^T-\Gamma^{(i)})\boldsymbol{\eta}^{(i)}-\lambda(\mathbf{1}^T\boldsymbol{\eta}^{(i)}-1)\\&=\boldsymbol{\eta}^{(i)T}C^{(i)}\boldsymbol{\eta}^{(i)}-\lambda(\mathbf{1}^T\boldsymbol{\eta}^{(i)}-1)\end{aligned}$$

, where $C^{(i)}$ denotes a $N_i \times N_i$ matrix (assume that it is full ranks, which means $N_i \le d$).

- The solution $\left[\boldsymbol{\eta}^{(i)*}, \lambda^*\right] = \arg\min_{\boldsymbol{\eta}^{(i)},\lambda} e(\boldsymbol{\eta}^{(i)}, \lambda)$ can be achieved by partial derivation:

$$\frac{\partial e}{\partial \boldsymbol{\eta}^{(i)}} = 2C^{(i)}\boldsymbol{\eta}^{(i)} - \lambda \mathbf{1} = 0$$
$$\frac{\partial e}{\partial \lambda} = \mathbf{1}^T \boldsymbol{\eta}^{(i)} - 1 = 0$$

, where $2C^{(i)}\boldsymbol{\eta}^{(i)} = \lambda \mathbf{1} \rightarrow \boldsymbol{\eta}^{(i)} = \frac{\lambda}{2} C^{(i)-1}\mathbf{1}$. Then the solution $\boldsymbol{\eta}^{(i)} = \frac{C^{(i)-1}\mathbf{1}}{\mathbf{1}^T C^{(i)-1}\mathbf{1}}$

**The computation of the reduced feature set**

- The function to be minimized in this step is:

$$\begin{aligned}\Phi(Z) &= \sum_{i=1}^{N} \left\| \boldsymbol{z}^{(i)} - \sum_{j=1}^{N} w_{ij}\boldsymbol{z}^{(j)} \right\|^2 = \left\| Z - ZW^T \right\|_{L^2}^2 \\ &= tr\{(Z - ZW^T)^T (Z - ZW^T)\} = tr\{(I_{N\times N} - W)Z^T Z (I_{N\times N} - W^T)\} \\ &= tr\{Z(I_{N\times N} - W^T)(I_{N\times N} - W)Z^T\} = tr\{Z(I_{N\times N} - W^T - W + W^T W)Z^T\} \\ &= tr\{ZMZ^T\}\end{aligned}$$

, where $m_{ij} = [\delta_{ij} - w_{ij} - w_{ji} + \sum_{k=1}^{N} w_{ki}w_{kj}]_{1\le i,j\le N}^{N}$.

- With the constraints on *Z*, now the problem we are solving becomes:

$$Z^* = \min_{ZZ^T = I} tr\{ZMZ^T\}$$

The Rayleitz-Ritz theorem is applied to solve this problem, where the last *p* eigenvectors of *M* are the solution. Because the last eigenvector $\frac{1}{\sqrt{N}}\mathbf{1}_N$ with eigenvalue 0 is indeed a trivial solution, we discard it and add the $(p+1)$th smallest one in.

- As you can see, the formula above looks very similar to the one in PCA, while now we perform minimization rather than maximization.
- The proof that $\mathbf{1}_N$ is an eigenvector of *M* with zero eigenvalue, which equivalently means the summation of each row of *M* is 0:

$$\sum_{j=1}^{N} m_{ij} = \sum_{j=1}^{N} [\delta_{ij} - w_{ij} - w_{ji} + \sum_{k=1}^{N} w_{ki} w_{kj}]$$
$$= 1 - \sum_{j=1}^{N} w_{ij} - \sum_{j=1}^{N} w_{ji} + \sum_{j=1}^{N} \sum_{k=1}^{N} w_{ki} w_{kj}$$
$$= -\sum_{j=1}^{N} w_{ji} + \sum_{k=1}^{N} w_{ki} \sum_{j=1}^{N} w_{kj} = -\sum_{j=1}^{N} w_{ji} + \sum_{k=1}^{N} w_{ki} = 0.$$

Despite the differences of LLE and Isomap, they do share some important characteristics in common. First, the scale of neighborhood ($\varepsilon$-neighbor) or the number of neighbors (*K*NN) need to be defined before learning. Second, there exist closed form and global optimal solutions for both Isomap and LLE, which is critical against some other nonlinear dimensionality reduction methods such as autoencoder neural networks [42] and self-organizing maps [21], etc. Third, both of them are explicit methods, which mean there is no direct mapping function for unseen and future samples.

### 8.2 Extensions of LLE

After the first version of LLE, S. T. Roweis et al. later published one introduction paper [43] and one comprehensive paper [44] to further discuss LLE, such as how many neighbors to choose for building plausible dimensionality reduction results, etc. In order to make LLE available for new coming samples, X. He et al. [36] proposed a linearization framework, called Neighborhood preserving embedding (NPE), to learn projection bases based on the LLE criterion. Then, new samples can be embedded into the reduced feature space simply by matrix operation. In Table 10, we present the algorithm of NPE, and in Table 11, the theoretical justifications are discussed.

Table 10 The algorithm of NPE

**Presetting:**

- Follow the presetting and goal of LLE in Table 8
- **Pre-center the data set**, where each $\boldsymbol{x}^{(n)}$ is replaced by $\boldsymbol{x}^{(n)} - \bar{\boldsymbol{x}}$.

**Goal:**

- Find a $d \times p$ transformation matrix $B$, then $\boldsymbol{z} = B^T(\boldsymbol{x} - \bar{\boldsymbol{x}})$ for new coming samples

, where $\boldsymbol{z} \in R^p$ .

- There are three steps in NPE, and only the third step is different.

**Step 3: Compute the projection matrix *B***

- Follow the same constraints of LLE for re-embedding in
- Table
- Algorithm

$M = (I - W)^T (I - W)$

$XMX^T \boldsymbol{v}^{(i)} = \lambda^{(i)} XX^T \boldsymbol{v}^{(i)} \rightarrow XMX^T V = XX^T V \Lambda,$ where $\Lambda$ is in descending order

$B = \left[ \boldsymbol{v}^{(N-p+1)}, \boldsymbol{v}^{(N-p+2)}, ....., \boldsymbol{v}^{(N)} \right] = V \left[ O_{p \times (N-p)} \mid I_{p \times p} \right]^T$

Table 11 The theoretical justifications of NPE

**The modeling of projection matrix**

- MPE nearly follows all the procedure of LLE, while assumes that the reduced vector $\boldsymbol{z}$ can be represented as $\boldsymbol{z} = B^T (\boldsymbol{x} - \bar{\boldsymbol{x}})$. During training, $X$ is pre-centered.
- Put the model above into the objective function and constraints of LLE:

$$\Phi(Z) = \sum_{i=1}^{N} \left\| \boldsymbol{z}^{(i)} - \sum_{j=1}^{N} w_{ij} \boldsymbol{z}^{(j)} \right\|^2 = tr(ZMZ^T) \rightarrow tr(B^T XMX^T B),$$

$$ZZ^T = B^T XX^T B = I.$$

Then, the optimization problem can be rewritten as:

$$B^* = \arg \min_{B^T XX^T B = I} B^T XMX^T B$$

, which can be solved through generalized eigenvector problem because both $XMX^T$ and $XX^T$ are symmetric positive semi-definite matrices.

**Singular problem of $XX^T$ :**

- When $XX^T$ is singular, $XMX^T \boldsymbol{v}^{(i)} = \lambda^{(i)} XX^T \boldsymbol{v}^{(i)}$ cannot be solve. To figure out this problem, we can first perform PCA on $X$ to reach a lower dimensional $X_{l \times N}$ where $XX^T$ is nonsingular. Now $B_X W_{PCA}$ becomes the new projection matrix.

NPE exploits the criterion of LLE and is an implicit method where dimensionality reduction of new coming samples can be easily computed. It can be easily noticed that the projection model assumption of NPE restricts the power of LLE, and the authors of NPE suggests the idea of kernel NPE to release this problem. In addition, they claimed that NPE can become supervised when the label information is considered in $W$, but did not describe the implementation in detailed. Compared to PCA, NPE exploits the nonlinear local geometry properties instead of the linear covariance information. Although both of them are based on linear model, the properties to preserve are different and the achieved projection bases are different. Besides, the authors claimed that NPE is less sensitive to outliers than PCA. The experimental results of NPE showed it outperforms PCA and LDA for face recognition task on the ORL face database. Y. Fu et al. [45] also mentioned how to make LLE available for new coming samples and discuss more on considering label information for supervised dimensionality reduction. There proposed method is called locally embedded analysis (LEA).

# 9. Laplacian Eigenmap and Its Extensions

## 9.1 Laplacian Eigenmap

Away from Isomap and LLE, in 2002, M. Belkin et al. [19][46] proposed another way of thinking in manifold learning, called the Laplacian eigenmap. In their claim, an optimal embedding should keep neighbor points in the $d$-dimensional space still close in the $p$-dimensional space. This idea can be formulated as minimizing the summation of the Laplacian-Beltrami operator over the entire manifold. The Laplacian-Beltrami operator computes the divergence of mapping function from a point $\boldsymbol{x} \in R^d$ on the manifold to $\boldsymbol{z} \in R^p$. Because now we only have finite samples from the manifold, the Laplacian-Beltrami operator on the manifold is approximated by the Laplacian of the graph obtained from the data samples. In Table 12, we list the algorithm of Laplacian eigenmap, and in Table 13, we provide the theoretical justifications of the algorithm.

Laplacian eigenmap is based on the same idea of LLE, preserving only the local geometry, while uses another method to model the objective function. The similarity matrix $W$ is sparse which facilitates the computation of EVD. The experimental results in [19][46] showed that Laplacian eigenmap could achieve nonlinear dimensionality reduction in both visualization and data representation tasks. The

algorithm of Laplacian eigenmap is easier than Isomap and LLE, while there is an additional parameter $t$ to define in the heat kernel. Laplacian eigenmap is an unsupervised, explicit, and layered feature transform, so it is also suffered from the new sample problem.

Table 12 The algorithm of Laplacian eigenmap

**Presetting:**

- Given a training set $X = \{\boldsymbol{x}^{(n)} \in R^d\}_{n=1}^N$, denote $X$ as a $d \times N$ matrix.
- Define a $N \times N$ matrix $W = [w_{ij} = 0]_{1 \le j \le N}$ to record the (spatial) similarities between points, where $w_{ij}$ is the similarity between $\boldsymbol{x}^{(i)}$ and $\boldsymbol{x}^{(j)}$.

**Goal:**

- Desire to transform the original $d$-dimensional vector into $p$-dimensional ($p \le d$)
- Get a transformed set: $Z = \{\boldsymbol{z}^{(n)} \in R^p\}_{n=1}^N$, denote $Z$ as a $p \times N$ matrix.
- The set containing all the neighbors of $\boldsymbol{x}^{(i)}$ is denoted as $Nei(i)$. The neighbors can be defined through $K$-nearest neighbor or $\varepsilon$-neighbor.

**Step 1: Find the neighbor of each sample**

- Follow the step 1 in
- Table.

**Step 2: Set the similarity between neighbors**

- Algorithm:
  for each pair $\{i, j\}$

  $$\text{if } (\boldsymbol{x}^{(j)} \in Nei(i)) \rightarrow w_{ij} = \exp\left(-\frac{\left\|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\right\|^2}{t}\right)$$

  end
- The parameter $t$ is adjustable. A simple choice is $t = \infty$ ($w_{ij} = 1$ iff $\boldsymbol{x}^{(j)} \in Nei(i)$), where all pair of neighbors get a unit similarity.

**Step 3: Set the similarity between neighbors**

- Algorithm:

  (1) Define an $N \times N$ matrix $D$, where $d_{ii} = \sum_{j=1}^{N} w_{ij}$, and the other entries $= 0$

  (2) Define the Laplacian matrix $L = (D - W)$, which is symmetric positive semi-definite.

  (3) Solve the generalized eigenvector problem:

  $$L\boldsymbol{v} = \lambda D\boldsymbol{v} \rightarrow D^{-1}LV = \Lambda V, \text{ where } \Lambda \text{ is in descending order}$$

The solution of $Z$ is $Z = \left[\boldsymbol{v}^{(N-p)}, \boldsymbol{v}^{(N-p+1)}, ....., \boldsymbol{v}^{(N-1)}\right]^T = V\left[O_{p\times(N-1-p)} \mid I_{p\times p} \mid O_{p\times 1}\right]^T$

Table 13 The justification of Laplacian eigenmap

**The criterion of Laplacian eigenmap:**

- The criterion is to keep neighbors in $R^d$ still close in $R^p$, which can be formulated as the following function:

$$E(Z) = \sum_{i=1}^{N}\sum_{j=1}^{N} \left\| \boldsymbol{z}^{(i)} - \boldsymbol{z}^{(j)} \right\|^2 w_{ij}$$

, where $w_{ij}$ measures (spatial) similarity between $\boldsymbol{x}^{(i)}$ and $\boldsymbol{x}^{(j)}$. When two points are close in the original feature space, the similarity between them is large. Minimizing the above function aims at keeping points with high similarity close in the reduced feature set Z, while don't care points with small similarity (if two points are not neighbors to each other, $w_{ij} = 0$)

- $E(Z)$ can be rewritten as:

$$\begin{aligned} E(Z) &= \sum_{i=1}^{N}\sum_{j=1}^{N} \left\| \boldsymbol{z}^{(i)} - \boldsymbol{z}^{(j)} \right\|^2 w_{ij} = \sum_{i=1}^{N}\sum_{j=1}^{N} (\boldsymbol{z}^{(i)} - \boldsymbol{z}^{(j)})^T (\boldsymbol{z}^{(i)} - \boldsymbol{z}^{(j)}) w_{ij} \\ &= \sum_{i=1}^{N}\sum_{j=1}^{N} \boldsymbol{z}^{(i)T}\boldsymbol{z}^{(i)} w_{ij} - 2\sum_{i=1}^{N}\sum_{j=1}^{N} \boldsymbol{z}^{(i)T}\boldsymbol{z}^{(j)} w_{ij} + \sum_{i=1}^{N}\sum_{j=1}^{N} \boldsymbol{z}^{(j)T}\boldsymbol{z}^{(j)} w_{ij} \\ &= 2\sum_{i=1}^{N} \boldsymbol{z}^{(i)T}\boldsymbol{z}^{(i)} \left(\sum_{j=1}^{N} w_{ij}\right) - 2\sum_{i=1}^{N}\sum_{j=1}^{N} \boldsymbol{z}^{(i)T}\boldsymbol{z}^{(j)} w_{ij} \\ &= 2\sum_{i=1}^{N} tr\{d_{ii}\boldsymbol{z}^{(i)T}\boldsymbol{z}^{(i)}\} - 2\cdot tr\{Z^T W Z\} \\ &= 2\cdot tr\{ZDZ^T\} - 2\cdot tr\{ZWZ^T\} = 2tr\{ZLZ^T\} \end{aligned}$$

- With this reformulation, now the minimization problem is denoted as:

$$Z = \arg\min_{ZDZ^T = I} E(Z) \equiv \arg\min_{ZDZ^T = I} trace(ZLZ^T)$$

, where the constraint $ZDZ^T = I$ is used to prevent degradation. The solution to this problem can be computed from $L\boldsymbol{v} = \lambda D\boldsymbol{v}$, while we discard the last eigenvector with eigenvalue 0 because it is a trivial solution $\mathbf{1}_N$.

### 9.2 Extensions of Laplacian Eigenmap

Just like the work of NPE, X. He et al. [20] also proposed the linearization of Laplacian eigenmap, called locality preserving projection (LPP). LPP is probably the first work to use linearization for making manifold learning available for new coming samples. To further release the restrictions of linear projection, kernelization is performed on LPP, which is called kernel LPP. In Table 14, we present the algorithm of LPP, and in Table 15, the theoretical justifications are discussed.

Table 14 The algorithm of LPP

**Presetting:**

- Follow the presetting and goal of Laplacian eigenmap in Table 12.
- **Pre-center the data set**, where each $\boldsymbol{x}^{(n)}$ is replaced by $\boldsymbol{x}^{(n)} - \bar{\boldsymbol{x}}$.

**Goal:**

- Find a $d \times p$ transformation matrix $B$**,** then $\boldsymbol{z} = B^T(\boldsymbol{x} - \bar{\boldsymbol{x}})$ for new coming samples , where $\boldsymbol{z} \in R^p$.
- There are three steps in LPP, and only the third step is different.

**Step 3: Compute the projection matrix *B***

- Algorithm

  $L = D - W$, where $d_{ii} = \sum_j w_{ij}$

  $XLX^T\boldsymbol{v}^{(i)} = \lambda^{(i)} XDX^T\boldsymbol{v}^{(i)} \rightarrow XLX^TV = XDX^TV\Lambda$, where $\Lambda$ is in descending order

  $B = \left[\boldsymbol{v}^{(N-p+1)}, \boldsymbol{v}^{(N-p+2)}, ....., \boldsymbol{v}^{(N)}\right] = V\left[O_{p\times(N-p)} \mid I_{p\times p}\right]^T$

Table 15 The theoretical justifications of LPP

**The modeling of projection matrix**

- Put the model of LPP into the objective function and constraints of Laplacian eigenmap:

$$E(Z)=\sum_{i=1}^{N}\sum_{j=1}^{N}\left\|\boldsymbol{z}^{(i)}-\boldsymbol{z}^{(j)}\right\|^2 w_{ij}\equiv tr\{ZLZ^T\}\rightarrow tr\{B^TXLX^TB\}$$

$$ZDZ^T=B^TXDX^TB=I.$$

Then, the optimization problem can be rewritten as:

$$B^*=\arg\min_{B^TXDX^TB=I} B^TXLX^TB$$

, which can be solved through generalized eigenvector problem because both $XLX^T$ and $XDX^T$ are symmetric positive semi-definite matrices.

**Singular problem of $XX^T$ :**

- When $XDX^T$ is singular, $XMX^T\boldsymbol{v}^{(i)}=\lambda^{(i)}XX^T\boldsymbol{v}^{(i)}$ cannot be solve. The same solution using PCA in Table 11 can be directly exploited.

Table 16 The algorithm of OLPP (only the third step is shown)

**Step 3: Compute the projection matrix *B***

- Algorithm

$B^{(p)}=\left[\boldsymbol{b}^{(1)},\boldsymbol{b}^{(2)},.....,\boldsymbol{b}^{(p)}\right]$

$\boldsymbol{b}^{(1)}$ is the smallest eigenvector of $Q=(XDX^T)^{-1}XLX^T$

for $k=2:p$

compute $G^{(k-1)}=[B^{(k-1)}]^T(XDX^T)^{-1}B^{(k-1)}$

$\boldsymbol{b}^{(k)}$ is the smallest eigenvector of $(I-(XDX^T)^{-1}B^{(k-1)}[G^{(k-1)}]^{-1}[B^{(k-1)}]^T)Q$

end

- The theoretical justification can be found in [10], which used the Lagrange multipliers to solve the orthogonal bases constraint.

LPP has been experimented in [20] to outperform PCA on the visualization task and both LDA and PCA on face recognition using the Yale database. Laplacianfaces proposed in 0 directly exploits LPP for face recognition, and more comparisons

among PCA, LDA, and LPP can be found in that paper.

The generalized eigenproblem doesn't guarantee that the projection matrix is orthogonal. This phenomenon makes LPP difficult to reconstruct data, and may distort the metric structure when using LPP for face recognition. In [10], D. Cai et al. proposed the orthogonal version of LPP, called the orthogonal locality reserving projection (OLPP). The experimental results showed that OLPP has more locality preserving power than LPP and outperforms LPP in the face recognition tasks using Yale, ORL, and PIE database. OLPP is also adopted in [8][47] for facial age estimation tasks. Table presented the algorithm of OLPP, which is different from LPP only in the third step.

Rather than using neural networks or linearization to extend Isomap, LLE, and Laplacian eigenmap for new samples, Y. Bengio proposed [30] an out-of-sample extension framework which can be applied on the above three manifold learning methods as well as MDS and spectral clustering [48]. The framework is based on seeing these algorithms as learning eigenfunctions of a data-dependent kernel. In practice, only a set of examples of the underlying manifold can be observed, which means these algorithms can only learn eigenvectors rather than eigenfunctions. While as the number of samples increases, the learned eigenvalues and eigenvectors will converge. Based on the empirical properties extracted from the observed samples, the reduced feature vectors of new coming samples can be seen as weighted summations of reduced feature vectors obtained from the training samples. The weights of combination are computed from the distances or similarities between a new sample and all the training samples.

# 10.Graph Embedding and Its Extensions

## 10.1 The Graph Embedding Framework

The previous six sections discuss different algorithms of dimensionality reduction. In fact, there are some common characteristics among them, such as building a distance, covariance, or similarity matrix among training samples and solving the optimization problem by generalized eigenvector decomposition. In 2007, S. C. Yan et al. [26] proposed a unified framework for dimensionality reduction, known as graph embedding. Graph embedding views each sample of the training set as a graph vertex, and the relationship (weight on the edge) between each pair of vertices can be measured by a graph similarity matrix that characterizes certain statistical or geometric properties of the dataset. The purpose of graph embedding is

to represent each vertex of the graph as a low-dimensional vector that preserves the measured similarity between the vertex pairs. And the solution comes from the eigenvectors corresponding to the leading eigenvalues of the graph Laplacian matrix with certain constraints. In order to deal with new coming samples, linearization and kernelization are presented to transfer manifold learning algorithms into a linear or nonlinear projection matrix. In addition, to preserve the higher-order statistics of certain dataset, tensorization is also presented to make manifold learning available not only for vectors but also for tensor projects.

Graph embedding claims that the properties of a labeled or unlabeled dataset can be characterized by two graphs, an intrinsic graph and a penalty graph. The intrinsic graph describes the desired statistical or geometrical properties of the dataset to be preserved, while the penalty graph characterizes the properties that should be avoided. When there is no explicit property to be avoided, the penalty graph is replaced by certain constraints for scale normalization to prevent degradation and trivial solutions. In Table 17, the concepts and formulations of graph embedding are summarized.

Table 17 The concepts and formulations of graph embedding

**Presetting:**

- Training set $X = \{\boldsymbol{x}^{(n)} \in R^d\}_{n=1}^N$, $Y = \{y^{(n)} \in L\}_{n=1}^N$, $L = \{l_1, l_2, ....., l_c\}$
- **Pre-center the data set**, where each $\boldsymbol{x}^{(n)}$ is replaced by $\boldsymbol{x}^{(n)} - \bar{\boldsymbol{x}}$.

**Goal:**

- Desire to transform the original *d*-dimensional vector into *p*-dimensional ( $p \le d$ )
- Get a transformed set: $Z = \{\boldsymbol{z}^{(n)} \in R^p\}_{n=1}^N$, denote $Z$ as a $p \times N$ matrix.

**The optimization criterion of graph embedding:**

- Objective function: $E(Z) = \sum_{i=1}^{N}\sum_{j=1}^{N} \left\|\boldsymbol{z}^{(i)} - \boldsymbol{z}^{(j)}\right\|^2 w_{ij} = tr(ZLZ^T)$
- Constraints:

$$tr(ZPZ^T) = \begin{cases} \sum_{i=1}^{N} \left\|\boldsymbol{z}^{(i)}\right\|^2 P_{ii} = \gamma \text{ (a constant), for normalization constraints} \\ \sum_{i=1}^{N}\sum_{j=1}^{N} \left\|\boldsymbol{z}^{(i)} - \boldsymbol{z}^{(j)}\right\|^2 w_{ij}^p = \gamma, \text{ for penalty graph proerties} \end{cases}$$

- Optimization criterion: $Z^* = \arg\min_{tr(ZPZ^T)=\gamma} tr(ZLZ^T) \equiv \arg\min \frac{tr(ZLZ^T)}{tr(ZPZ^T)}$
- $L$ is the Laplacian matrix of the intrinsic graph $G = \{X, W\}$, where $W$ is an $N \times N$ edge weight matrix recording the pairwise similarity properties to be preserved in $X$. The similarities can be measured by various methods though domain knowledge, Euclidean distance, local geometry, or label information. The Laplacian matrix of $G$ is defined as $L = D - W$, where $D$ is a diagonal matrix and $D_{ii} = \sum_{j=1}^{N} d_{ij}$.
- $P$ is the constraint matrix to avoid trivial solutions of the objective function. $P$ typically is an $N \times N$ diagonal matrix for scale normalization, and may also be the Laplacian matrix $L^p$ of a penalty graph $G^p = \{X, W^p\}$. The penalty graph $G^p$ shares the same vertex set as $G$, while the edge weight matrix $W^p$ records similarity properties to be avoid during dimensionality reduction.

**Linearization:**

- Modeling: $\boldsymbol{z} = B^T \boldsymbol{x} \rightarrow Z = B^T X$
- Optimization criterion:

$$(1) B^* = \arg\min_{tr(B^T XPX^T B)=\gamma} tr(B^T XLX^T B) \equiv \arg\min_B \frac{tr(B^T XLX^T B)}{tr(B^T XPX^T B)}$$

$$(2) B^* = \arg\min_{tr(B^T B)=\gamma} tr(B^T XLX^T B) \equiv \arg\min_B \frac{tr(B^T XLX^T B)}{tr(B^T B)}$$

Sometimes the (1) scale normalization on $Z$ will be transferred onto (2) $B$.

**Kernelization:**

- Extends the original $d$-dimensional $\boldsymbol{x}$ into $d_\Phi$-dimensional $\Phi(\boldsymbol{x})$ by feature transform introduce in Section 錯誤! 找不到參照來源。. Then the kernel trick is exploited to represent the inner product of these transformed feature vectors.
- Modeling:

$$\Phi : \boldsymbol{x} \in \boldsymbol{R}^d \mapsto \quad \boldsymbol{x} \quad \boldsymbol{R}^{d_\Phi}$$

$$K = [k_{ij}]_{1 \le i,j \le N}, \text{ where } k_{ij} = k(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) = \Phi(\boldsymbol{x}^{(i)})^T \Phi(\boldsymbol{x}^{(j)})$$

$$B = \left[ \boldsymbol{b}^{(1)}, \boldsymbol{b}^{(2)}, ....., \boldsymbol{b}^{(p)} \right] = \left[ \sum_{i=1}^{N} a_{i1} \Phi(\boldsymbol{x}^{(i)}), ....., \sum_{i=1}^{N} a_{ip} \Phi(\boldsymbol{x}^{(N)}) \right] = \Phi(X) A_{N \times p}$$

$$\boldsymbol{z} = B^T \Phi(\boldsymbol{x}) = A^T \Phi(X)^T \Phi(\boldsymbol{x}) = \left[ \sum_{i=1}^{N} a_{i1} k(\boldsymbol{x}^{(i)}, \boldsymbol{x}), ....., \sum_{i=1}^{N} a_{ip} k(\boldsymbol{x}^{(i)}, \boldsymbol{x}) \right]^T$$

- Optimization criterion:

$$(1)\,A^* = \arg\min_{tr(A^T KPK^T A)=\gamma} tr(A^T KLK^T A) \equiv \arg\min_A \frac{tr(A^T KLK^T A)}{tr(A^T KPK^T A)}$$

$$(2)\,A^* = \arg\min_{tr(A^T KA)=\gamma} tr(A^T KLK^T A) \equiv \arg\min_A \frac{tr(A^T KLK^T A)}{tr(A^T KA)}$$

**Optimization:**

- Through Rayleigh quotient and generalized eigenvector decomposition:

$$L\boldsymbol{v}^{(i)} = \lambda^{(i)} P\boldsymbol{v}^{(i)} \rightarrow LV = PV\Lambda.$$

, where $L = L, XLX^T, KLK,$ and $P = I, P, K, XPX^T, KPK^T$. The eigenvalue matrix $\Lambda$ is in descending order.

- The $p \times N$ matrix $Z$, $d \times p$ matrix $B$, and the $N \times p$ matrix $A$ are composed of the last $p$ eigenvectors in $V$ with smallest eigenvalues. As $L = L$, the last eigenvector is a trivial solution with equal entries, so we simply omit it.
- If $P$ is singular, the generalized eigenvector problem is generally unsolvable. To deal with this problem, PCA is often used to pre-reduce the dimensionality of $X$, and then $P$ will probably become nonsingular.

Tensorization requires additional notations, operators, and iterative optimization methods. Besides, in our implementation, the features are fully represented in vector form rather than the tensor object form. So, in this section, we ignore the tensorization part. Graph embedding doesn't explicitly claim how to define the entries in $W$ and $W^p$. The functionality of these weights is similar to the one in Laplacian eigenmap, where points with large weights in $W$ should be mapped into nearby points in $Z$. In contrary, points with large weights in $W^p$ should be pulled away in $Z$. In [26], Yan et al. showed how the criterion of PCA, LDA, Isomap, LLE, Laplacian eigenmap, and LPP can be transformed into the graph embedding framework. And based on this framework, linearization and kernelization are easily to perform. In addition, compared to other proposed framework [30][49][50], graph embedding is claimed more flexible in manifold learning design and suitable for supervised learning. Manifold learning algorithms which can be explained by graph embedding are layered.

### 10.2 Marginal Fisher Analysis

Graph embedding not only provides a common framework for dimensionality reduction, but can also be used as a platform for developing new DR algorithms. To

show this functionality, a new supervised dimensionality reduction algorithm, called Marginal Fisher analysis (MFA), is proposed. The idea of MFA is based on LDA mentioned in Section 5 and seeks to preserve neighbors with the same label while pull away neighbors with different labels. The intrinsic graph in MFA characterizes the intra-class compactness and connects each sample with its neighbor samples of the same class, while the penalty graph connects the marginal samples (samples with different class label) and characterizes the interclass separability. In Table 18, we show the algorithm of MFA.

Table 18 The algorithm of MFA

**Presetting and goal:**

- Follow the presetting and goal of graph embedding in Table 17.
- MFA is a linearization form, and can be extended to kernelization form.
- There are four steps in MFA algorithm

**Criterion of MFA:**

- **Intra-class:**

$$S_W = \sum_{i=1}^{N} \sum_{j \in Nei^+(i)} \left\| B^T \boldsymbol{x}^{(i)} - B^T \boldsymbol{x}^{(j)} \right\|^2 w_{ij} \equiv tr\{B^T X (D - W) X^T B\},$$

$$\text{where } w_{ij} = \begin{cases} 1, \text{ if } \boldsymbol{x}^{(j)} \in Nei^+(i) \text{ iff } \boldsymbol{x}^{(j)} \in KNN^+(i) \text{ or } \boldsymbol{x}^{(i)} \in KNN^+(j) \\ 0, \text{ else} \end{cases}$$

- **Inter-class:**

$$S_B = \sum_{i=1}^{N} \sum_{j \in Nei^-(i)} \left\| B^T \boldsymbol{x}^{(i)} - B^T \boldsymbol{x}^{(j)} \right\|^2 w_{ij}^p \equiv tr\{B^T X (D^p - W^p) X^T B\},$$

$$\text{where } w_{ij}^p = \begin{cases} 1, \text{ if } \boldsymbol{x}^{(j)} \in Nei^-(i) \text{ iff } \boldsymbol{x}^{(j)} \in KNN^-(i) \text{ or } \boldsymbol{x}^{(i)} \in KNN^-(j) \\ 0, \text{ else} \end{cases}$$

- $KNN^+(i)$ means $k_1$-nearest same-label samples around $\boldsymbol{x}^{(i)}$
  $KNN^-(i)$ means $k_2$-nearest different-label samples around $\boldsymbol{x}^{(i)}$
- **Marginal Fisher criterion:** $B^* \equiv \arg\min_B \frac{tr(B^T X (D - W) X^T B)}{tr(B^T X (D^p - W^p) X^T B)}$

**Step 1: PCA for singular $(D^p - W^p)$ to reach $X$ and $W_{PCA}$**

**Step 2: Find the $k_1$ same-label and $k_2$ different-label neighbors of each sample.**

**Step 3: Build $W$ and $W^p$.**

**Step 4: Perform EVD and get the projection basis *B*:**

$$\{X(D-W)X^T\}V = \{X(D^p - W^p)X^T\}V\Lambda \rightarrow B = V\left[O_{p\times(N-p)} \mid I_{p\times p}\right]^T$$

**Discussions:**

- For new coming samples, $\boldsymbol{z} = B^T W_{PCA}(\boldsymbol{x} - \bar{\boldsymbol{x}})$.
- $k_2$ will affects the dimensionality of the reduced feature space $p$.

Compared to LDA, MFA doesn't utilize the between-class and within-class scatter for measuring inter-class separability and intra-class compactness. This means MFA releases the assumption that data of each class is approximated Gaussian distributed. The authors claimed that without a prior assumption on data distributions, the inter-class margin used in MFA can better characterizes the separability of different classes than the inter-class scatter in LDA. From the face recognition results performed on several face database, MFA outperforms PCA and LDA, and the further extended kernel MFA (KMFA) and tensor MFA (TMFA) improves the recognition accuracy.

### 10.3 Extensions and Related Work of Graph Embedding

Based on or related to the platform of graph embedding, several dimensionality reduction techniques in the linearization form were proposed. H. T. Chen et al. [51] proposed a supervised dimensionality reduction algorithm called local discriminant embedding (LDE). LDE exploits both the neighbor and label information of the data and learns the embedding for the submanifold of each class. The derivation and of LDE is very similar to MFA, while the neighbor weighting step and the objective function is different. Besides, PCA is optional in LDE for singular problem. To our best knowledge, LDE is the first manifold learning algorithm to use the intrinsic and penalty graphs together for supervised learning. In Table 19, the algorithm of LDE is presented.

D. Cai et al. [52] proposed the locality sensitive discriminate analysis (LSDA) for supervised dimensionality reduction. The idea of LSDA is very similar to MFA and LDE, while it adjusts the importance of the two graphs and modifies the optimization criterion. MFA and LDE build the intrinsic graph with the $k_1$-nearest

same-label neighbors and the penalty graph with the $k_2$-nearest different-label neighbors around each sample. This definition implies that the importance ratio between intra- and inter- class information is equally at each sample, and the total importance ratio between the intrinsic and the penalty graph is controlled by the choices of $k_1$ and $k_2$. LSDA releases these implicit restrictions, and performs better than LDE and MFA on face recognition with the Yale database. In Table 20, the algorithm of LSDA is summarized.

Table 1 The algorithm of LDE

**Presetting and goal:**

- Follow the presetting, goal, and the notations of MFA in Table.
- There are three steps in LDE, and the last two are different from MFA.

**Criterion of LDE:** $B^* \equiv \arg \max_{tr(B^T X(D-W)X^T B)=1} tr(B^T X(D^p - W^p)X^T B)$

**Step 1: Find the $k_1$ same-label and $k_2$ different-label neighbors of each sample.**

**Step 2: Build $W$ and $W^p$:**

- $$w_{ij} = \begin{cases} \exp(-\dfrac{\left\| \boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)} \right\|^2}{t}), & \text{if } \boldsymbol{x}^{(j)} \in Nei^+(i) \text{ iff } \boldsymbol{x}^{(j)} \in KNN^+(i) \text{ or } \boldsymbol{x}^{(i)} \in KNN^+(j) \\ 0, & \text{else} \end{cases}$$
- $$w_{ij}^p = \begin{cases} \exp(-\dfrac{\left\| \boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)} \right\|^2}{t}), & \text{if } \boldsymbol{x}^{(j)} \in Nei^-(i) \text{ iff } \boldsymbol{x}^{(j)} \in KNN^-(i) \text{ or } \boldsymbol{x}^{(i)} \in KNN^-(j) \\ 0, & \text{else} \end{cases}$$

**Step 3: Perform EVD and get *B*:** (largest $p$ eigenvectors are preferred)

- $\{X(D^p - W^p)X^T\}V = \{X(D-W)X^T\}V\Lambda \rightarrow B = V\left[I_{p\times p} \mid O_{p\times(N-p)}\right]^T$

Table 20 The algorithm of LSDA

**Presetting and goal:**

- Follow the presetting, goal, and the notations of MFA in Table 18.
- There are three steps in LSDA:

**Criterion of LSDA:**

- **Objective function:**

$$\min_B \sum_{i=1}^{N}\sum_{j}^{N}\left\|B^T\boldsymbol{x}^{(i)}-B^T\boldsymbol{x}^{(j)}\right\|^2 w_{ij} \equiv \min_B tr\{B^T X(D-W)X^T B\}$$

$$\max_B \sum_{i=1}^{N}\sum_{j}^{N}\left\|B^T\boldsymbol{x}^{(i)}-B^T\boldsymbol{x}^{(j)}\right\|^2 w_{ij}^p \equiv \max_B tr\{B^T X(D^p-W^p)X^T B\}$$

- **Constraint:** $tr(B^T XDX^T B)=1$
- **Optimization criterion:** $B^* = \arg\max_{tr\{B^T XDX^T B\}=1} tr\{B^T X(\alpha L^p+(1-\alpha)W)X^T B\}$ ,where $\alpha$ is a suitable constant that $0\le\alpha\le 1$.

**Step 1: Find the neighbors of each sample.** ( $\varepsilon$ -neighbors or *K*-nearest neighbors)

**Step 2: Build $W$ and $W^p$:**

- $$w_{ij}=\begin{cases}1, \text{ if } \boldsymbol{x}^{(j)}\in Nei(i) \text{ and } y^{(i)}=y^{(j)}\\ 0, \text{ else}\end{cases}$$
- $$w_{ij}^p=\begin{cases}1, \text{ if } \boldsymbol{x}^{(j)}\in Nei(i) \text{ and } y^{(i)}\neq y^{(j)}\\ 0, \text{ else}\end{cases}$$

**Step 3: Perform EVD and get *B*:** (largest *p* eigenvectors are preferred)

- $\{X(\alpha L^p+(1-\alpha)W)X^T\}V=\{XDX^T\}V\Lambda \rightarrow B=V\left[I_{p\times p}\,|\,O_{p\times(N-p)}\right]^T$

There are still other formulations of the intrinsic and penalty graphs, and other optimization criteria as well as methods for supervised dimensionality reduction in the linearization form. Q. You et al. [53] proposed the neighborhood discriminant projection (NDP), which builds the intrinsic graph by the linear reconstruction coefficients as in LLE and NPE. The penalty graph of NDP directly follows the one of LDSA, while the optimization criterion is similar to the one in LDE. To solve the singular problem, NDP exploits the null space method used in [54], rather than the PCA method used in NPE, LPP, and MFA, etc.

Y. Fu et al. proposed the conformal embedding analysis (CEA) [11] and further used it in facial age estimation task [9]. Incorporating both conformal mapping and discriminating analysis, CEA first projects the high-dimensional data onto the unit hypersphere by vector normalization and preserves intrinsic neighbor relations with the intrinsic graph modeling. The distance metric used in CEA for finding neighbors and setting the weight of graph edges is defined as $dis(\boldsymbol{x}^{(i)},\boldsymbol{x}^{(j)})=1-<\boldsymbol{x}^{(i)},\boldsymbol{x}^{(j)}>$.

Through the manifold learning procedure similar to MFA and LDE, the subspace learned by CEA is claimed to be gray-level variation tolerable because of the cosine-angle metric and the normalization processing.

## 11. Other Nonlinear Dimensionality Reduction Techniques

In this section, we briefly introduce other popular techniques of nonlinear dimensionality reduction. Some of them are based the concept of manifold, while some of them are based on either tensor objects, independence among features, or perspectives of neural networks, etc. We will list the references of each method for extended studies.

### 11.1 Manifold Alignment

In addition to the manifold learning algorithms introduced in the previous four sections, there are still other criteria and idea on how to learn the manifold structure from given data. The manifold alignment idea, probably first named in [55], seeks to locally build a set of low dimensional coordinates, and then jointly aligns them into a global low dimensional coordinate. For example, image that a group of students distribute in the playground. Each student is asked to describe his or her relative locations to other nearby students. Based on this information, the teacher labels each student with a unique location in a global coordinate. In fact, LLE implicitly utilizes this idea and exploits linear reconstruction coefficients to record the local geometry. The local tangent space alignment algorithm (LTSA), proposed by Z. Y. Zhang et al. [56], is also based on the idea of manifold alignment.

Originated from the neighbor preserving idea of LLE, LTSA further considers the reconstruction of the original feature space from the reduced feature space. The basic assumption of LTSA is that the reduced feature set $Z$ controls the observed feature set $X$, and the mapping function $f : \boldsymbol{z} \in R^p \mapsto \boldsymbol{x}$ is locally very smooth and with a certain degree of noise. Then, the neighborhood relationship defined in $Z$ will be preserved in $X$. Using first-order Taylor's expansion on $f(\cdot)$ at each $\boldsymbol{z}^{(i)}$, we can build a local tangent space around $\boldsymbol{z}^{(i)}$ and analyzes how a small variation in $\boldsymbol{z}$ linearly affects the variation in $\boldsymbol{x}$. The tangent space characterizes the local geometry around each $\boldsymbol{z}^{(i)}$ and its neighbors, while in practice, neither $f(\cdot)$ and $Z$ is given.

With the smooth assumption on $f(\cdot)$, LTSA first approximately build the set of

local tangent spaces from the information in $X$. This can be down by generating a set of $p$-dimensional feature spaces as well as $d \times p$ projection matrices to represent the local tangent geometry of $X$ through SVD or PCA. These coordinate systems are locally meaningful, but misaligned from the global view. The second step of LTSA is then to align these local coordinates into a global coordinates system $Z$. Given each local coordinates system the freedom to de individually transformed by an affine matrix, LTSA jointly learn these transformation matrices and minimizes the error between local coordinates and the global coordinate. LTSA is explicit, unsupervised, and standalone, and the linearization of LTSA was proposed later by T. H. Zhang et al. [57].

## 11.2 Maximum Variance Unfolding

Also based on the concept of manifold, K. Q. Weinberger et al. [58] proposed a new manifold learning algorithm which can be solved by semi-definite programming and EVD. Follow the fundamental notion of isometry, where the manifold can be locally rotated and translated into another homeomorphic manifold, this new method suggests to preserve local Euclidean distances as well as local angles. This property can be further modified as preserving distances in the fully connected clique generated by each sample and its neighbors. Set these properties as the constraints of an optimization problem, Weinberger et al. proposed to maximize the sum of all pairwise distances between the reduced feature vectors in $Z$. This objective function can be thought of to unfold a manifold as much as possible, and implicitly indicates that the reduced space is globally Euclidean, just like the assumption in MDS and Isomap.

The equality constraints induced by distance preserving are quadratic, which makes the optimization problem not convex. To solve this problem, Weinberger et al. first transferred the argument $Z$ into $K = Z^T Z$, which makes the optimization problem fully with linear equality constraints and an additional convex cone constraint. The modified optimization problem with argument $K$ is exactly an instance of semi-definite programming (SDP) and can be solved by standard convex optimization solver. After we get the optimized $K$, a similar procedure like MDS is performed to extract the desired representation $Z$.

There are two modifications proposed on the constraints, including changing the equality constraints into inequality constraints, and adding slack variables to balance the violation of equality constraints and the variance gain. In addition, the authors further compare this maximum variance unfolding algorithm with LLE, Isomap,

Laplacian eigenmap and list their advantages and disadvantages. Maximum variance unfolding is explicit, unsupervised, and layered.

### 11.3 Kernel PCA and Kernel LDA

The kernel trick mentioned in Section 10 can also be applied on the traditional PCA and LDA, which yields the so-called Kernel PCA (KPCA) [59] and Kernel LDA (KLDA or KFLD) [60]. In [61], M. H. Yang exploits both these two methods in face recognition, and showed that KFLD outperforms LDA as well as ICA in both Yale and AT&T database.

### 11.4 Independent Component Analysis

Base on the same model assumption as PCA where $\boldsymbol{z} = W^T \boldsymbol{x}$, independent component analysis (ICA) aims at making the entries of $\boldsymbol{z}$ as independent as possible with each other. From another perspective, PCA decorrelates the entries of $\boldsymbol{z}$ with the Gaussian distribution assumption on data, while ICA requires the distribution of data to be non-Gaussian and maximizes the independence of entries in $\boldsymbol{z}$. There are several criteria on how to represent the independencies between features and the probability distribution of features, and readers interested in ICA could referred to [4][12][62] for more details.

### 11.5 Other Methods

There are still several popular nonlinear dimensionality reduction techniques. For example, the Self-Organizing Map (SOM) proposed by T. Kohonen [21], the Hessian eigenmap proposed by D. L. Donoho et al. [63], and the autoencoder neural networks [42]. To get more detailed understanding and comparisons on nonlinear dimensionality reduction, the textbook [17] written by J. A. Lee et al. is a good reference. And to get an overview on several dimensionality reduction algorithms, the survey paper written by I. K. Fodor [64] and the one written by L. Maaten et al. [65] are recommended.

## 12. Feature Selection

Instead of performing feature transform in the original *d*-dimensional space to extract the *p*-dimensional representation *Z*, feature selection directly select a subset of

$p$ features from the original $d$ features. Feature transform is usually combined with supervised learning problems to reduce the model complexity while keep the important features for classification and regression. There are generally three categories of feature selection methods, embedded feature selection, wrapper, and filtering.

- **Embedded feature selection:** This category means that during supervised learning, the classification or regression model can simultaneously learns the importance of each feature. For example, the Lasso regression [66], random forest (RF) [67], and even SVM have these functionalities.

- **Wrapper:** This category can be seen as a brute-force searching method, which tries different subset of features to train the classifiers or regressors, and the subset with the best generalization performance is selected. Wrapper can be used on nearly any kind of supervised learning algorithm, and in order to speed-up the searching process, several mechanisms on how to select the sub-optimal subset are proposed [23].

- **Filtering:** This category utilized the statistical analysis between each feature and the label information to denote the feature importance. For example, the correlation between each feature and the corresponding class label, or the mutual information between them.

In [68], I. Guyon et al. provided a comprehensive survey on feature and variable selection, which is a good starting point of feature selection.

## 13. Conclusion

In this tutorial, the theoretical and practical aspects of dimensionality reduction are presented. At first, we introduce the purposes and categorization of dimensionality, and a brief mathematical overview of "curse of dimensionality" is given. From Section 4 to Section 5, two unsupervised linear dimensionality reduction techniques, PCA and MDS, and one supervised technique, LDA, are introduced with detailed mathematical justifications. And from Section 7 to Section 9, three important manifold learning techniques, ISOMAP, LLE, and Laplacian eigenmap are presented with their mathematical proves as well as algorithms. Several extensions based these three methods are also overviewed in these three sections. In Section 10, a general

framework of manifold learning and dimensionality reduction called graph embedding is introduced, and we also discuss a few supervised manifold learning techniques based on graph embedding. At the end of this chapter, other kinds of dimensionality reduction methods are surveyed, and a short introduction of feature selection is presented. The techniques introduced and discussed in this chapter are the main part of the proposed human aging estimation method. And even recently in statistics and computer science, dimensionality reduction is still a hot research topic. We hope that this chapter to give readers a theoretical and practical guidelines to use dimensionality reduction techniques.

## Reference


[1] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neuroscience*, vol. 3, no.1, pp. 72-86, 1991.

[2] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, 711-720, 1997.

[3] X. He, S. Yan, Y. Hu, P. Niyogi, and H. Zhang, "Face recognition using Laplacianfaces," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 27, no. 3, pp. 328-340, 2005.

[4] A. Hyvärinen, E. Oja, "Independent component analysis: algorithms and applications," *Neural Networks*, vol. 13, pp. 411–30, 2000.

[5] T. F. Cootes, G. J. Edwards, and C. J. Taylor, "Active appearance models," *Computer Vision - ECCV'98*, vol. 2, 1998. Springer, Lecture Notes in Computer Science 1407, pp. 484-498, 1998.

[6] G. J. Edwards, T. F. Cootes, and C. J. Taylor, "Face recognition using active appearance models," *Proc. European Conf. Computer Vision*, vol. 2, pp. 581-695, 1998.

[7] T. F. Cootes, G. J. Edwards, and C. J. Taylor, "Active appearance models," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol.23, no. 6, pp. 681-685, 2001.

[8] Y. Fu, Y. Xu, and T.S. Huang, "Estimating Human Ages by Manifold Analysis of Face Pictures and Regression on Aging Features," *Proc. IEEE Conf. Multimedia and Expo*, pp. 1383-1386, 2007.

[9] Y. Fu and T.S. Huang, "Human Age Estimation with Regression on Discriminative Aging Manifold," *IEEE Trans. Multimedia*, vol. 10, no. 4, pp. 578-584, June 2008.

[10] D. Cai, X. He, J. Han, and H.-J. Zhang, "Orthogonal Laplacianfaces for Face

Recognition," *IEEE Trans. Image Processing*, vol. 15, no. 11, pp. 3608-3614, Nov. 2006.

[11] Y. Fu, M. Liu, and T.S. Huang, "Conformal Embedding Analysis with Local Graph Modeling on the Unit Hypersphere," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*–Workshop Component Analysis, 2007.

[12] S. Makeig, A. J. Bell, T. P. Jung, and T. Sejnowski, "Independent component analysis of electroencephalographic data," *Advances in Neural Information Processing Systems*, vol. 8, 1996.

[13] T. Hofmann, "Probabilistic latent semantic indexing," *Proceedings of the Twenty-Second Annual International SIGIR Conference*, 1999.

[14] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *In Computational Learning Theory: Eurocolt '95*, pp. 23-37, 1995.

[15] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, 290, 2319–2323, 2000.

[16] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, 290, 2323–2326, 2000.

[17] J.A. Lee and M. Verleysen. *Nonlinear dimensionality reduction*, Springer, New York, NY, USA, 2007.

[18] R. Bellman, *Adaptative Control Processes: A Guided Tour*, Princeton University Press, Princeton, NJ, 1961.

[19] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Computation,* 15, 1373–1396, 2003.

[20] X. He and P. Niyogi, "Locality Preserving Projections," *Advance in Neural Information Processing Systems*, vol. 16, 2003.

[21] T. Kohonen, *Self-organizing maps*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997.

[22] C. M. Bishop, *Neural networks for pattern recognition*, Oxford University Press, 1995.

[23] S. Theodoridis and K. Koutroumbas, *Pattern recognition*, 4th ed., Academic Press, 2009.

[24] E. Alpaydin, *Introduction to machine learning*, 2nd ed., The MIT Press, 2010.

[25] M. E. Tipping and C. M. Bishop, "Probabilistic principal component analysis," *Journal of the Royal Statistical Society*, B, vol. 6, no. 3, pp. 611–622, 1999.

[26] S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, and S. Lin, "Graph embedding

and extension: A general framework for dimensionality reduction," *IEEE Transactions on PAMI*, vol. 29, no. 1, 2007.

[27] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, 179-188, 1936.

[28] R. O. Duda, P. E. Hart, D. G. Stoke, *Pattern classification*, 2nd ed., John Wiley & Sons, 2001.

[29] R. A. Horn and C. A. Johnson, *Matrix Analysis*, Cambridge University Press. pp. 176–180, 1985.

[30] Y. Bengio, J. Paiement, P. Vincent, O. Delalleau, N. Roux, and M. Ouimet, "Out-of-Sample Extensions for LLE, ISPMAP, MDS, Eigenmaps, and Spectral Clustering," *Advances in Neural Information Processing Systems*, vol. 16, 2004

[31] J. B. Kruskal, "Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis," *Psychometrika*, vol. 29, pp. 1-28, 1964.

[32] R. N. Shepard, "The analysis of proximities: Multidimensional scaling with an unknown distance function (parts 1 and 2)," *Psychometrika*, vol.27, pp. 125-140, pp. 219-249, 1962.

[33] J. W. Sammon, "A nonlinear mapping algorithm for data structure analysis," *IEEE Transactions on Computers*, vol. 18, no. 5, pp. 401-409, 1969.

[34] P. Demartines and J. H'erault, "CCA: Curvilinear component analysis," *In 15th Workshop GRETSI*, Juan-les-Pins (France), September 1995.

[35] M. Vlachos, C. Domeniconi, D. Gunopulos, G. Kollios, and N. Koudas, "Non-linear dimensionality reduction techniques for classification and visualization," *In Proceedings of 8th SIGKDD*, pp. 645-651, 2002.

[36] X. He, D. Cai, S. Yan, and H.-J. Zhang, "Neighborhood preserving embedding," *In Proceedings of the 10th IEEE International Conference on Computer Vision*, pp. 1208-1213, 2005.

[37] X. Geng, D. C. Zhan, and Z. H. Zhou, "Supervised nonlinear dimensionality reduction for visualization and classification," *IEEE Transactions on Systems, Man, and Cybernetics*–Part B: Cybernetics, vol. 35, pp. 1098-1107, 2005.

[38] V. de Silva and J.B. Tenenbaum, "Global versus local methods in nonlinear dimensionality reduction," *In Advances in Neural Information Processing Systems*, vol. 15, pp. 721-728, 2003.

[39] V. de Silva and J.B. Tenenbaum, "Sparse multidimensional scaling using landmark points," Tech. rep., Stanford University, June 2004.

[40] M. H. Yang, "Face recognition using extended isomap," *in Proc. IEEE International Conference on Image Processing*, vol. 2, pp. 117–120, 2002.

[41] M. H. Yang, "Extended Isomap for pattern classification," AAAI-02, pp. 224–229,

2002.

[42] D. DeMers and G. Cottrell, "Non-linear dimensionality reduction," *In Advances in Neural Information Processing Systems*, vol. 5, pp. 580–587, 1993.

[43] L. K. Saul, and S. T. Roweis, "An introduction to Locally Linear Embedding," 2001. URL: http://www.cs.toronto.edu/~roweis/lle/publications.html.

[44] L. K. Saul, and S. T. Roweis, "Think globally, fit locally: unsupervised learning of low dimensional manifolds," *Journal of Machine Learning Research*, vol. 4, pp. 119-155, 2003.

[45] Y. Fu and T. Huang, "Locally Linear Embedded Eigenspace Analysis," IFP-TR, Univ. of Illinois at Urbana-Champaign, Jan. 2005.

[46] M. Belkin and P. Niyogi, "Laplacian Eigenmaps and spectral techniques for embedding and clustering," *In Advances in Neural Information Processing Systems*, vol. 14, pp. 585–591, 2002.

[47] G. Guo, Y. Fu, C. Dyer, and T.S. Huang, "Image-Based Human Age Estimation by Manifold Learning and Locally Adjusted Robust Regression," *IEEE Trans. Image Processing*, vol. 17, no. 7, pp. 1178-1188, 2008.

[48] A. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: analysis and an algorithm," *Advances in Neural Information Processing Systems*, vol. 14, 2002.

[49] J. Ham, D. Lee, S. Mika, and B. Scholkopf, "A Kernel View of the Dimensionality Reduction of Manifolds," *Proc. Int'l Conf. Machine Learning*, pp. 47-54, 2004.

[50] M. Brand, "Continuous Nonlinear Dimensionality Reduction by Kernel Eigenmaps," Technical Report 2003-21, Mitsubishi Electric Research Labs, 2003.

[51] H. T. Chen, H. W. Chang, and T. L. Liu, "Local discriminant embedding and its variants," *In Proc. 2005 Internal Conference on Computer Vision and Pattern Recognition*, 2005.

[52] D. Cai, X. He, K. Zhou, J. Han, and H. Bao, "Locality sensitive discriminant analysis," *Proc. 20th Int'l Joint Conf. Artificial Intelligence* (IJCAI '07), 2007.

[53] Q. You, N. Zheng, S. Du, and Y. Wu, "Neighborhood discriminant projection for face recognition," *Pattern Recognition Letters*, vol. 28, Issue 10, pp. 1156-1163, 2007.

[54] H. Cevikalp, M. Neamtu, M. Wilkes, and A. Barkaba, "Discriminative common vectors for face recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 1, pp. 4-13, 2005.

[55] S. Roweis, L. Saul, and G. Hinton, "Global coordination of linear models," *In NIPS*, vol. 13, 2002.

[56] Z. Zhang and H. Zha, "Principal manifolds and nonlinear dimensionality

reduction via local tangent space alignment," *SIAM Journal of Scientific Computing*, vol. 26, no. 1, pp. 313-338, 2004.

[57] T. Zhang, J. Yang, D. Zhao, and X. Ge, "Linear local tangent space alignment and application to face recognition," *Neurocomputing*, vol. 70, pp. 1547-1533, 2007.

[58] K. Q. Weinberger and L. K. Saul, "Unsupervised learning of image manifolds by semidefinite programming," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 988-995, 2004.

[59] B. Scholkopf, A.J. Smola, and K.-R. Muller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, no. 5, pp. 1299-1319, 1998.

[60] G. Baudat and F. Anouar, "Generalized discriminant analysis using a kernel approach," *Neural Computation*, vol. 12, no. 10, pp. 2385-2404, 2000.

[61] M. H. Yang, "Kernel eigenfaces vs. kernel fisherfaces: Face recognition using kernel methods," *in Proc. IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 215-220, 2002.

[62] P. Comon, "Independent component analysis, a new concept?," *Signal Process.*, Special Issue on Higher Order Statistics, vol. 36 , pp. 287–314, 1994.

[63] D. L. Donoho, and C. E. Grimes, "Hessian eigenmaps: locally linear embedding techniques for high-dimensional data," *Proceedings of the National Academy of Arts and Sciences*, vol. 100, pp. 5591–5596, 2003.

[64]I.K. Fodor, "A survey of dimension reduction techniques," Technical report UCRL-ID-148494, LLNL, 2002.

[65] L.J.P. van der Maaten, E.O. Postma, and H.J. van den Herik, "Dimensionality reduction: A comparative review," Online Preprint, 2008.

[66] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, 2nd ed., Springer, 2005.

[67] L. Breiman, "Random forests," Technical report, Department of Statistics, University of California, 2001.

[68] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning*, res. 3, pp. 1157-1182, 2003.