Chapter 4

Nonlinear Extensions to Principal Component Analysis

"One geometry cannot be more true than another; it can only be more convenient."

– Henri Poincaré

In the previous chapters, we studied the problem of fitting a low-dimensional linear or affine subspace to a collection points. In practical applications, however, a linear or affine subspace may not be able to capture nonlinear structures in the data. For example, consider the set of all images of a face obtained by rotating it about its main axis of symmetry. While all such images live in a high-dimensional space whose dimension is the number of pixels, there is only one degree of freedom in the data, namely the angle of rotation. In fact, the space of all such images is a one-dimensional circle embedded in a high-dimensional space, whose structure is not well captured by a one-dimensional line.

In this chapter, we consider the problem of fitting a low-dimensional manifold to a collection of points. Specifically, let $X = \{\boldsymbol{x}_j \in \mathbb{R}^D\}_{j=1}^N$ be a set of N points drawn from a d-dimensional manifold $\mathcal{M}$ embedded in $\mathbb{R}^D$, where $d < D$ (see e.g., Figure 4.1). The goal is to find a set of N points $Y = \{\boldsymbol{y}_j \in \mathbb{R}^d\}_{j=1}^N$ whose geometry resembles that X. To address this problem, we present an extension of PCA, called Nonlinear PCA, which is based on embedding the data via a nonlinear mapping into a high-dimensional space and then fitting a linear or affine space to the embedded data. We also present other extensions of PCA, which used the data to approximate the local geometry of the manifold and build a low-dimensional embedding of the data directly from these local approximations. This type of extensions are useful for applications where we are not so interested

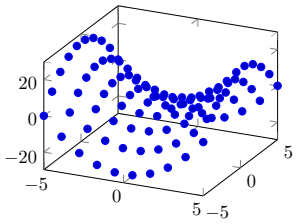


Figure 4.1. A set of points drawn from the two-dimensional surface in $\mathbb{R}^3$, $x_3 = x_1^2 - x_2^2$. The goal is to find a two-dimensional embedding of this manifold.

in a parametric model of the manifold, but rather in the low-dimensional points $Y = \{\boldsymbol{y}_j\}$ themselves.

Although the data could lie in complex manifolds with multiple connected components, in this chapter we will assume the data all lie on a single continuous nonlinear manifold. Datasets that cannot be modeled by a single subspace or manifold and are instead clustered into multiple components will be studied in Part II. Or alternatively, even for a nonlinear continuous manifold such as the one shown in Figure 4.1, one can also choose to approximate it with a piece-wise linear model that may consist of many (local) linear subspaces. Again, we will pursue such a multiple-subspace solution in Part II.

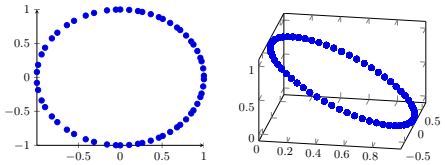
4.1 Nonlinear and Kernel PCA

In this section, we present a nonlinear extension of PCA called Nonlinear PCA (NLPCA). The key idea behind NLPCA is that, instead of applying PCA to the given data, we apply it to a nonlinear embedding of the data into a high-dimensional space $\mathcal{H}$. The rationale is that the structure of the data becomes (approximately) linear after applying a nonlinear embedding to it. In practice, however, the dimension of $\mathcal{H}$ may be too high to be able to compute the *nonlinear principal components* from the eigenvectors of the embedded covariance matrix. To address this issue, we also present a method called Kernel PCA (KPCA). This method computes the nonlinear principal components from the eigenvectors of the so-called *kernel matrix*, which can be computed directly from the given data.

4.1.1 Nonlinear Principal Component Analysis (NLPCA)

As discussed before, the main idea behind NLPCA is that we may be able to find an embedding of the data into a high-dimensional space such that the structure of the embedded data becomes (approximately) linear. To see why this may be possible, consider a set of points $(x_1, x_2) \in \mathbb{R}^2$ lying in a conic of the form

$$c_1 x_1^2 + c_2 x_1 x_2 + c_3 x_2^2 + c_4 = 0. \tag{4.1}$$



(a) $x_1^2 + x_2^2 - 1 = 0$ (b) $z_1 + z_3 - 1 = 0$

Figure 4.2. A circle in $\mathbb{R}^2$ is embedded into a plane in $\mathbb{R}^3$ by the mapping in (4.2) .

Notice that if we define the map $\phi : \mathbb{R}^2 \to \mathbb{R}^3$ as

$$(z_1, z_2, z_3) = (x_1^2, \sqrt{2}x_1x_2, x_2^2), \tag{4.2}$$

then the conic in $\mathbb{R}^2$ transforms into the following affine subspace in $\mathbb{R}^3$

$$c_1z_1 + \frac{c_2}{\sqrt{2}}z_2 + c_3z_3 + c_4 = 0. \tag{4.3}$$

Therefore, instead of learning a nonlinear manifold in $\mathbb{R}^2$, we can simply learn an affine manifold in $\mathbb{R}^5$. This example is illustrated in Figure 4.2.

More generally, we seek a nonlinear transformation (usually an embedding):

$$\phi(\cdot) : \mathbb{R}^D \to \mathbb{R}^M, \tag{4.4}$$

$$\boldsymbol{x} \mapsto \phi(\boldsymbol{x}), \tag{4.5}$$

such that the structure of the embedded data $\{\phi(\boldsymbol{x}_j)\}_{j=1}^N$ becomes approximately linear. In machine learning, $\phi(\boldsymbol{x}) \in \mathbb{R}^M$ is called the *feature* of the data point $\boldsymbol{x} \in \mathbb{R}^D$, and the space $\mathbb{R}^M$ is called the *feature space*.

Let $\bar{\boldsymbol{\phi}} = \frac{1}{N}\sum_{j=1}^N \phi(\boldsymbol{x}_j)$ be the sample mean in the feature space and define the mean subtracted (centered) embedded data matrix as

$$\Phi \doteq [\phi(\boldsymbol{x}_1) - \bar{\boldsymbol{\phi}}, \phi(\boldsymbol{x}_2) - \bar{\boldsymbol{\phi}}, \ldots, \phi(\boldsymbol{x}_N) - \bar{\boldsymbol{\phi}}] \in \mathbb{R}^{M\times N}. \tag{4.6}$$

It follows from the results in Chapter 2 that the principal components in the feature space can be obtained from the eigenvectors of the sample covariance matrix[1]

$$\Sigma_{\phi(\boldsymbol{x})} \doteq \frac{1}{N}\sum_{j=1}^N (\phi(\boldsymbol{x}_j) - \bar{\boldsymbol{\phi}})(\phi(\boldsymbol{x}_j) - \bar{\boldsymbol{\phi}})^\top = \frac{1}{N}\Phi\Phi^\top \in \mathbb{R}^{M\times M}. \tag{4.7}$$

Specifically, let $\boldsymbol{u}_i \in \mathbb{R}^M$, $i = 1, \ldots, M$, be the M eigenvectors of $\Sigma_{\phi(\boldsymbol{x})}$, i.e.,

$$\Sigma_{\phi(\boldsymbol{x})}\boldsymbol{u}_i = \lambda_i\boldsymbol{u}_i, \quad i = 1, 2, \ldots, M. \tag{4.8}$$

Then the d *nonlinear principal components* of every data point $\boldsymbol{x}$ are given by

$$y_i \doteq \boldsymbol{u}_i^\top(\phi(\boldsymbol{x}) - \bar{\boldsymbol{\phi}}) \in \mathbb{R}, \quad i = 1, 2, \ldots, d. \tag{4.9}$$

[1] In principle, we should use the notation $\hat{\Sigma}_{\phi(\boldsymbol{x})}$ to indicate that it is the estimate of the actual covariance matrix. But for simplicity, we will drop the hat in the sequel and simply use $\Sigma_{\phi(\boldsymbol{x})}$. The same goes for the eigenvectors and the principal components.

Unfortunately, the map $\phi(\cdot)$ is generally not known beforehand and searching for the map that makes the embedded data approximately linear is a difficult task. In such cases, the use of nonlinear PCA becomes limited. However, in some practical applications, good candidates for the map $\phi(\cdot)$ can be found from the nature of the problem. In such cases, the map, together with PCA, can be very effective in extracting the overall geometric structure of the data.

Example 4.1 (Veronese Map for an Arrangement of Subspaces). As we will see later in this book, if the data points belong to a union of multiple subspaces, then a natural choice of the transformation $\phi(\cdot)$ is the Veronese map:

$$\begin{aligned} \nu_n(\cdot): \quad \boldsymbol{x} &\mapsto \nu_n(\boldsymbol{x}), \\ (x_1, \ldots, x_D) &\mapsto (x_1^n, x_1^{n-1}x_2, \ldots, x_D^n), \end{aligned}$$

where the monomials are ordered in the degree-lexicographic order. Under such a mapping, the multiple low-dimensional subspaces are mapped into a single subspace in the feature space, which can then be identified via PCA. ■

4.1.2 NLPCA in a High-dimensional Feature Space

A potential difficulty associated with NLPCA is that the dimension M of the feature space can be very high. Thus, computing the principal components in the feature space may become computationally prohibitive. For instance, if we use a Veronese map of degree n, the dimension of the feature space is $M = \binom{n+D-1}{n}$, which grows exponentially fast. When M exceeds N, the eigenvalue decomposition of $\Phi\Phi^\top \in \mathbb{R}^{M\times M}$ becomes more costly than that of $\Phi^\top\Phi \in \mathbb{R}^{N\times N}$, although the two matrices have the same eigenvalues.

This motivates us to examine whether the computation of PCA in the feature space can be reduced to a computation with the lower-dimensional matrix $\Phi^\top\Phi$. The answer is actually yes. The key is to notice that, despite the dimension of the feature space, every eigenvector $\boldsymbol{u} \in \mathbb{R}^M$ of $\Phi\Phi^\top$ associated with a non-zero eigenvalue is always in the span of the matrix Φ,[2] that is:

$$\Phi\Phi^\top\boldsymbol{u} = \lambda\boldsymbol{u} \quad \Longleftrightarrow \quad \boldsymbol{u} = \Phi(\lambda^{-1}\Phi^\top\boldsymbol{u}) \in \text{range}(\Phi). \tag{4.10}$$

Thus, if we let $\boldsymbol{w} \doteq \lambda^{-1}\Phi^\top\boldsymbol{u} \in \mathbb{R}^N$, we have $\|\boldsymbol{w}\|^2 = \lambda^{-2}\boldsymbol{u}^\top\Phi\Phi^\top\boldsymbol{u} = \lambda^{-1}$. Moreover, since $\Phi^\top\Phi\boldsymbol{w} = \lambda^{-1}\Phi^\top\Phi\Phi^\top\boldsymbol{u} = \Phi^\top\boldsymbol{u} = \lambda\boldsymbol{w}$, the vector $\boldsymbol{w}$ is an eigenvector of $\Phi^\top\Phi$ with the same eigenvalue λ. Once such a $\boldsymbol{w}$ is computed from $\Phi^\top\Phi$, we can recover the corresponding $\boldsymbol{u}$ in the feature space as

$$\boldsymbol{u} = \Phi\boldsymbol{w}, \tag{4.11}$$

and compute the d nonlinear principal component of $\boldsymbol{x}$ under the map $\phi(\cdot)$ as:

$$y_i \doteq \boldsymbol{u}_i^\top(\phi(\boldsymbol{x}) - \bar{\boldsymbol{\phi}}) = \boldsymbol{w}_i^\top\Phi^\top(\phi(\boldsymbol{x}) - \bar{\boldsymbol{\phi}}) \in \mathbb{R}, \quad i = 1, \ldots, d, \tag{4.12}$$

where $\boldsymbol{w}_i \in \mathbb{R}^N$ is the ith leading eigenvector of $\Phi^\top\Phi \in \mathbb{R}^{N\times N}$.

[2] The remaining $M - N$ eigenvectors of $\Phi\Phi^\top$ are associated with the eigenvalue zero.

4.1.3 Kernel PCA (KPCA)

A very interesting property of the above NLPCA method is that the computation of the nonlinear principal components involves only inner products of the features. More specifically, in order to compute the nonlinear principal components, y_i, we simply need to compute the entries of the matrix $\Phi^\top\Phi$ and the entries of the vectors $\Phi^\top\phi(\boldsymbol{x})$ and $\Phi^\top\bar{\boldsymbol{\phi}} = \frac{1}{N}\sum\Phi^\top\phi(\boldsymbol{x}_i)$. In what follows, we show that all of these quantities can be obtained from inner products of the form $\phi(\boldsymbol{x})^\top\phi(\boldsymbol{y})$.

Before proceeding further, we will need some definitions.

Definition 4.2. *The space of all square integrable functions is defined as*

$$L^2(\mathbb{R}^D) = \{f : \mathbb{R}^D \to \mathbb{R} \text{ such that } \int f(\boldsymbol{x})^2 d\boldsymbol{x} < \infty\}. \tag{4.13}$$

Definition 4.3. *A function $k : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ is symmetric if for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^D$ we have $k(\boldsymbol{x}, \boldsymbol{y}) = k(\boldsymbol{x}, \boldsymbol{y})$. A symmetric function is positive semi-definite if for all $f \in L^2(\mathbb{R}^D)$ we have*

$$\iint_{\mathbb{R}^D\times\mathbb{R}^D} f(\boldsymbol{x})k(\boldsymbol{x}, \boldsymbol{y})f(\boldsymbol{y})\, d\boldsymbol{x}d\boldsymbol{y} \geq 0. \tag{4.14}$$

Let us define the "kernel function" $k : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ of two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^D$ to be the inner product of their features

$$k(\boldsymbol{x}, \boldsymbol{y}) \doteq \phi(\boldsymbol{x})^\top\phi(\boldsymbol{y}) \in \mathbb{R}. \tag{4.15}$$

One can show that (see Exercise **??**) that k is a symmetric positive semi-definite function in $\boldsymbol{x}$ and $\boldsymbol{y}$. Let us also define the centered kernel as

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) \doteq (\phi(\boldsymbol{x}) - \bar{\boldsymbol{\phi}})^\top(\phi(\boldsymbol{y}) - \bar{\boldsymbol{\phi}}) \in \mathbb{R}, \tag{4.16}$$

where $\bar{\boldsymbol{\phi}} = \frac{1}{N}\sum_i \phi(\boldsymbol{x}_i)$. We may compute κ from k as

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = k(\boldsymbol{x}, \boldsymbol{y}) - \frac{1}{N}\sum_j k(\boldsymbol{x}, \boldsymbol{x}_j) - \frac{1}{N}\sum_i k(\boldsymbol{x}_i, \boldsymbol{y}) + \frac{1}{N^2}\sum_i\sum_j k(\boldsymbol{x}, \boldsymbol{y}). \tag{4.17}$$

We can use these functions to compute the nonlinear principal components as follows. Define a *kernel matrix* $K = [k_{ij}] \in \mathbb{R}^{N\times N}$ as $k_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. The mean subtracted (centered) kernel matrix $\mathcal{K} = \Phi^\top\Phi$ can be computed from K as:

$$\mathcal{K} = K - \frac{1}{N}K\mathbf{1}\mathbf{1}^\top - \frac{1}{N}\mathbf{1}\mathbf{1}^\top K + \frac{\mathbf{1}^\top K\mathbf{1}}{N^2}\mathbf{1}\mathbf{1}^\top \tag{4.18}$$

$$= (I - \frac{1}{N}\mathbf{1}\mathbf{1}^\top)K(I - \frac{1}{N}\mathbf{1}\mathbf{1}^\top) = JKJ, \tag{4.19}$$

where $J = I - \frac{1}{N}\mathbf{1}\mathbf{1}^\top$ is called the *centering matrix*. Let us also define the vector $\kappa_{\boldsymbol{x}} = \Phi^\top(\phi(\boldsymbol{x}) - \bar{\boldsymbol{\phi}}) = [\kappa(\boldsymbol{x}_1, \boldsymbol{x}), \kappa(\boldsymbol{x}_2, \boldsymbol{x}), \ldots, \kappa(\boldsymbol{x}_N, \boldsymbol{x})]^\top \in \mathbb{R}^N$. This vector can be computed from $k_{\boldsymbol{x}} = [k(\boldsymbol{x}_1, \boldsymbol{x}), k(\boldsymbol{x}_2, \boldsymbol{x}), \ldots, k(\boldsymbol{x}_N, \boldsymbol{x})]^\top \in \mathbb{R}^N$ as:

$$\kappa_{\boldsymbol{x}} = k_{\boldsymbol{x}} - \frac{1}{N}K\mathbf{1} - \frac{1}{N}\mathbf{1}\mathbf{1}^\top k_{\boldsymbol{x}} + \frac{\mathbf{1}^\top K\mathbf{1}}{N^2}\mathbf{1}. \tag{4.20}$$

With this notation, we may compute the nonlinear principal components of $\boldsymbol{x}$ as

$$y_i = \boldsymbol{w}_i^\top \Phi^\top (\phi(\boldsymbol{x}) - \bar{\boldsymbol{\phi}}) = \boldsymbol{w}_i^\top \kappa_{\boldsymbol{x}}, \quad i = 1, \dots, d, \tag{4.21}$$

where $\boldsymbol{w}_i$ is the eigenvectors of $\mathcal{K}$ associated with its i-th largest eigenvalue, λ_i, and normalized so that $\|\boldsymbol{w}_i\| = \lambda_i^{-2}$. That is, $\begin{bmatrix}\boldsymbol{w}_1, \dots, \boldsymbol{w}_N\end{bmatrix} = V_d \Lambda_d^{-1/2}$, where V_d and Λ_d are obtained from the top d eigenvectors and eigenvalues in the EVD of $\mathcal{K} = V \Lambda V^\top$. Since $\mathcal{K} = \begin{bmatrix}\kappa_{\boldsymbol{x}_1}, \dots, \kappa_{\boldsymbol{x}_N}\end{bmatrix}$, it follows that we can compute the low-dimensional coordinates of the entire dataset as

$$Y = \Lambda_d^{-1/2} V_d^\top \mathcal{K} = \Lambda_d^{-1/2} V_d^\top V_d \Lambda_d V_d^\top = \Lambda_d^{1/2} V_d^\top. \tag{4.22}$$

In other words, the low dimensional coordinates can be obtained from the top d eigenvectors and eigenvalues of the centered kernel matrix.

Example 4.4 (PCA as a particular case of KPCA) For the linear kernel $k(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^\top \boldsymbol{y}$, we have $\phi(\boldsymbol{x}) = \boldsymbol{x}$, hence KPCA reduces to PCA. ■

Example 4.5 For the polynomial embedding of degree 2 in (4.2), we have

$$k(\boldsymbol{x}, \boldsymbol{y}) = [x_1^2, \sqrt{2}x_1x_2, x_2^2][y_1^2, \sqrt{2}y_1y_2, y_2^2]^\top = (x_1y_1 + x_2y_2)^2 = (\boldsymbol{x}^\top \boldsymbol{y})^2, \tag{4.23}$$

which can be computed directly in $\mathbb{R}^2$ without having to compute the embedding into $\mathbb{R}^3$. ■

In summary, we have shown that the nonlinear principal components can be computed directly from the kernel function $k(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x})^\top \phi(\boldsymbol{y})$ without having to compute $\phi(\boldsymbol{x})$. Nonetheless, given any (positive-definite) kernel function, according to a fundamental result in functional analysis, one can in principle decompose the kernel and recover the associated map $\phi(\cdot)$ if one wishes to.

Theorem 4.6 (Mercer's Theorem)**.** *Suppose $k : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ is a symmetric real valued function such that for some $C > 0$ and almost every $(\boldsymbol{x}, \boldsymbol{y})$[3] we have $|k(\boldsymbol{x}, \boldsymbol{y})| \leq C$. Suppose that the linear operator $\mathcal{L} : L^2(\mathbb{R}^D) \to L^2(\mathbb{R}^D)$,*

$$\mathcal{L}(f)(\boldsymbol{x}) \doteq \int_{\mathbb{R}^D} k(\boldsymbol{x}, \boldsymbol{y}) f(\boldsymbol{y}) d\boldsymbol{y}, \tag{4.24}$$

is positive semi-definite. Let ψ_i be the normalized orthogonal eigenfunctions of $\mathcal{L}$ associated with the eigenvalues $\lambda_i > 0$, sorted in non-increasing order, and let M be the number of nonzero eigenvalues. Then

- *The sequence of eigenvalues is absolutely convergent, i.e., $\sum_{i=1}^M |\lambda_i| < \infty$.*
- *The kernel k can be expanded as $k(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^M \lambda_i \psi_i(\boldsymbol{x}) \psi_i(\boldsymbol{y})$ for almost all $(\boldsymbol{x}, \boldsymbol{y})$. If $M = \infty$, the series is absolutely and uniformly convergent for almost all $(\boldsymbol{x}, \boldsymbol{y})$.*

[3] "Almost every" means except for a set of measure zero.

The interested readers may refer to [Mercer, 1909] for a proof of the theorem. It follows from the theorem that, given a positive semi-definite kernel k, we can always associate with it an embedding function ϕ as

$$\phi_i(\boldsymbol{x}) = \sqrt{\lambda_i}\psi_i(\boldsymbol{x}) \quad i = 1, \ldots, M. \tag{4.25}$$

Notice that the dimension of the embedding, M, could be rather large, sometimes even infinity. Nevertheless, an important reason for computing with the kernel function is that we do not need to compute the embedding function or the features. Instead, we simply evaluate the dot products $k(\boldsymbol{x}, \boldsymbol{y})$ in the original space $\mathbb{R}^D$.

Example 4.7 (Examples of Kernels). There are several popular choices for the nonlinear kernel function, such as the polynomial kernel and the Gaussian kernel, respectively,

$$k_P(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}^\top \boldsymbol{y})^n \quad \text{and} \quad k_G(\boldsymbol{x}, \boldsymbol{y}) = \exp\big(-\frac{\|\boldsymbol{x} - \boldsymbol{y}\|^2}{2}\big). \tag{4.26}$$

Evaluation of such functions only involves the inner product or the difference between two vectors in the original space $\mathbb{R}^D$. This is much more efficient than evaluating the inner product in the associated feature space, whose dimension grows exponentially for the first kernel and is infinite for the second kernel. ■

We summarize our discussion in this section with Algorithm 4.1.

Algorithm 4.1 (Nonlinear Kernel PCA)

Input: A set of zero-mean data points $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N] \in \mathbb{R}^{D \times N}$, a map $\phi(\boldsymbol{x})$ such that $\phi(\mathbf{0}) = \mathbf{0}$ or a kernel function $k(\boldsymbol{x}, \boldsymbol{y})$ such that $k(\mathbf{0}, \mathbf{0}) = 0$.

1: Compute the inner product matrix

$$\Phi^\top \Phi \;=\; \big(\phi(\boldsymbol{x}_i)^\top \phi(\boldsymbol{x}_j)\big) \;\text{ or }\; \big(k(\boldsymbol{x}_i, \boldsymbol{x}_j)\big) \;\in \mathbb{R}^{N \times N}; \tag{4.27}$$

2: Compute the eigenvectors $\boldsymbol{w}_i \in \mathbb{R}^N$ of $\Phi^\top \Phi$:

$$\Phi^\top \Phi \boldsymbol{w}_i = \lambda_i \boldsymbol{w}_i, \tag{4.28}$$

and normalize so that $\|\boldsymbol{w}_i\|^2 = \lambda_i^{-1}$;

3: For any data point $\boldsymbol{x}$, its ith nonlinear principal component is given by

$$y_i \;=\; \boldsymbol{w}_i^\top \Phi^\top \phi(\boldsymbol{x}) \;\text{ or }\; \boldsymbol{w}_i^\top [k(\boldsymbol{x}_1, \boldsymbol{x}), \ldots, k(\boldsymbol{x}_N, \boldsymbol{x})]^\top, \tag{4.29}$$

for $i = 1, 2, \ldots, d$.

Output: A set of points $\{\boldsymbol{y}_j\}_{j=1}^N$ lying in $\mathbb{R}^d$.

4.2 Nonparametric Manifold Learning

In the previous section we described NLPCA, a nonlinear extension of PCA based on embedding the data $X = \{\boldsymbol{x}_j \in \mathbb{R}^D\}_{j=1}^N$ into a high-dimensional space $\mathcal{H}$ and applying PCA in the embedded space to obtain the low-dimensional representation $Y = \{\boldsymbol{y}_j \in \mathbb{R}^d\}_{j=1}^N$ with $d < D$. In this section we present a family of

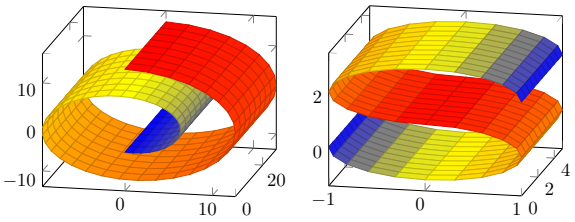


Figure 4.3. Two examples of manifolds typically used in manifold learning. Left: swiss roll. Right: s-curve.

manifold learning methods, which search directly for the low-dimensional representation Y without first embedding the data into a high-dimensional space. Such methods are based on approximating the geometry of the manifold (pairwise distances, local neighborhoods, local linear relationships, etc.) and using these approximations to find a global low-dimensional embedding. Different methods differ on how certain geometric properties of of X are intended to be preserved or approximated. In what follows, we discuss three representative popular manifold learning methods, namely Multidimensional Scaling (MDS), Laplacian Eigenmaps (LE), and Locally Linear Embedding (LLE). For a more comprehensive review of other manifold learning methods, we refer the reader to [Burges, 2005, Lee and Verleysen, 2007, Burges, 2010].

4.2.1 Multidimensional Scaling (MDS)

One of the oldest manifold learning methods is Multidimensional Scaling (MDS) [Torgerson, 1958, Kruskal, 1964, Gower, 1966, Cox and Cox, 1994]. This approach aims to capture the geometry of the manifold by finding a representation Y whose pairwise distances approximate the pairwise distances in X as well as possible. Specifically, let δ_{jk} be a distance between points $\boldsymbol{x}_j$ and $\boldsymbol{x}_k$, such as the Euclidean distance $\delta_{jk} = \|\boldsymbol{x}_j - \boldsymbol{x}_k\|$ or any other distance. More generally, δ_{jk} can be any dissimilarity measure between pairs of points. Given a matrix of dissimilarities $\Delta = [\delta_{jk}] \in \mathbb{R}^{N\times N}$, the goal of MDS is to find a low-dimensional representation $Y = \big[\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N\big] \in \mathbb{R}^{d\times N}$ that minimizes the following objective

$$\min_{Y} \sum_{j=1}^{N}\sum_{k=1}^{N}(\|\boldsymbol{y}_j - \boldsymbol{y}_k\| - \delta_{jk})^2. \tag{4.30}$$

Notice that, unlike PCA, MDS operates directly on the dissimilarities, hence it does not require us to have the matrix of data points $X = \big[\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\big] \in \mathbb{R}^{D\times N}$. Moreover, notice that MDS can capture nonlinear structures in the data by using a dissimilarity other than the Euclidean distance, e.g., a geodesic distance. However,

in general, the minimization over Y cannot be carried out in closed form, and gradient descent methods (see Appendix A) are typically used.[4]

However, notice that if instead of trying to approximate dissimilarities, we try to approximate similarities obtained from a dot product, then the solution to MDS can be obtained in closed form from the SVD of the similarity matrix. More specifically, let $A = [a_{jk}] \in \mathbb{R}^{N \times N}$ be the matrix of all pairwise similarities. For example, A can be defined as $a_{jk} = (\boldsymbol{x}_j - \boldsymbol{\mu})^\top (\boldsymbol{x}_k - \boldsymbol{\mu})$, where $\boldsymbol{\mu} = \frac{1}{N} X \mathbf{1}$ is the mean of the data.[5] More generally, A can be obtained after embedding the data into a high-dimensional space, as in NLPCA. In fact, we can think of A as a centered kernel matrix, which can be obtained as in (4.18). The only important property is that A is symmetric positive semidefinite. Given A, our goal is to find a low-dimensional representation Y such that the dot products between pairs of points best approximate the given similarities, i.e., we wish to minimize

$$\sum_{j=1}^{N} \sum_{k=1}^{N} (\boldsymbol{y}_j^\top \boldsymbol{y}_k - a_{jk})^2 = \|Y^\top Y - A\|_F^2. \tag{4.31}$$

Letting $Z = Y^\top Y$ and noticing that $\operatorname{rank}(Z) = \operatorname{rank}(Y) = d$, we arrive at the following optimization problem:

$$\min_{Z} \ \|Z - A\|_F^2 \quad \text{such that} \quad \operatorname{rank}(Z) = d, \quad Z = Z^\top, \quad Z \succeq 0. \tag{4.32}$$

Notice that, except for the symmetric positive semidefiniteness constraint on Z, this problem is identical to the low-rank matrix approximation problem in (2.35). However, since A is symmetric positive semidefinite, this additional constraint is not necessary. To see this, notice that if we use Theorem 2.6 to find the optimal solution, we obtain the optimal Z from the SVD of $A = U \Sigma U^\top$ as $Z = U_d \Sigma_d U_d^\top$, where U_d consists of the top d columns of U and Σ_d consists of the top $d \times d$ sub-block of Σ. Notice that this solution automatically satisfies the symmetric positive semidefiniteness constraint. Given $Z = U_d \Sigma_d U_d^\top$, we can obtain Y as $Y = R \Sigma_d^{1/2} U_d^\top$ for any orthogonal matrix $R \in O(d)$.

In summary, when the similarity A is a centered kernel matrix, MDS gives the same low-dimensional representation as KPCA, up to an arbitrary orthogonal transformation R. For further connections between MDS, PCA and KPCA, we refer the reader to [Williams, 2002].

4.2.2 *Laplacian Eigenmaps (LE)*

Another popular manifold learning algorithm is Laplacian Eigenmaps (LE) [Belkin and Niyogi, 2002]. This approach aims to capture the geometry of the manifold by finding a low-dimensional representation such that nearby points in the manifold are mapped to nearby points in the low-dimensional embedding.

[4] See [Davison, 1983] for alternative optimization methods for minimizing the objective in (4.30).
[5] Notice that $A = JX^\top XJ$, where $J = I - \frac{1}{N} \mathbf{1}\mathbf{1}^\top$ is the centering matrix.

More specifically, if $X = \begin{bmatrix}\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\end{bmatrix}$ is the given set of points in a manifold $\mathcal{M}$, LE finds a low-dimensional embedding $Y = \begin{bmatrix}\boldsymbol{y}_1, \dots, \boldsymbol{y}_N\end{bmatrix} \in \mathbb{R}^{d\times N}$ such that if $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are close to each other, so are $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$. This is done by minimizing the following objective

$$\phi(Y) = \sum_{ij} w_{ij}\|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2 \tag{4.33}$$

subject to appropriate constraints on Y that prevent the trivial solution $Y = 0$.

The weights $w_{ij} \geq 0$ are designed so that a small penalty is paid when $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are far so that $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ are allowed to be far, and a large penalty is paid when $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are close, but $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ are far. For this purpose, a local neighborhood of each point $\boldsymbol{x}_j$ is defined using the K nearest neighbors (K-NN) rule with some distance dist on $\mathcal{M}$, and the weights are chosen as

$$w_{ij} = \begin{cases} e^{-\frac{\mathrm{dist}(\boldsymbol{x}_i,\boldsymbol{x}_j)^2}{2\sigma^2}} & \text{if } \boldsymbol{x}_i \text{ is a } K\text{-NN of } \boldsymbol{x}_j \text{ or viceversa} \\ 0 & \text{else} \end{cases} \tag{4.34}$$

where $\sigma > 0$ is a parameter.

Letting $\mathcal{D} \in \mathbb{R}^{N\times N}$ be a diagonal matrix with diagonal entries $d_{ii} = \sum_j w_{ij}$ and $W \in \mathbb{R}^{N\times N}$ be the matrix of weights, we may rewrite the objective function as

$$\begin{aligned}\phi(Y) &= \sum_{ij} w_{ij}\big(\|\boldsymbol{y}_i\|^2 + \|\boldsymbol{y}_j\|^2 - 2\boldsymbol{y}_i^\top \boldsymbol{y}_j\big) && (4.35)\\ &= 2\sum_i d_{ii}\boldsymbol{y}_i^\top \boldsymbol{y}_i - 2\sum_{ij} w_{ij}\boldsymbol{y}_i^\top \boldsymbol{y}_j && (4.36)\\ &= 2\,\mathrm{trace}(Y\mathcal{D}Y^\top) - 2\,\mathrm{trace}(YWY^\top) = 2\,\mathrm{trace}(YLY^\top), && (4.37)\end{aligned}$$

where $L = \mathcal{D} - W$ is a symmetric positive semi-definite matrix such that $L\mathbf{1} = \mathbf{0}$.

To prevent the trivial solution $Y = 0$. LE requires the low-dimensional representation Y to satisfy the following additional constraints

$$Y\mathcal{D}\mathbf{1} = \mathbf{0} \quad \text{and} \quad Y\mathcal{D}Y^\top = I. \tag{4.38}$$

The first constraint requires the scaled low-dimensional representation[6] $Y\mathcal{D}$ to be centered at the origin, similar to what it is done in PCA (see Chapter 2). The second constraint ensures that $\mathrm{rank}(Y) = d$ and removes an arbitrary scaling factor in the embedding. Therefore, LE finds the low-dimensional representation by solving the following minimization problem

$$\min_Y \ \mathrm{trace}(YLY^\top) \quad \text{s.t.} \quad Y\mathcal{D}\mathbf{1} = \mathbf{0} \quad \text{and} \quad Y\mathcal{D}Y^\top = I. \tag{4.39}$$

The solution to this optimization problem is given by the next result.

[6] By scaled low-dimensional representation we mean replacing $\boldsymbol{y}_j$ by $d_{jj}\boldsymbol{y}_j$.

Proposition 4.8. *The solution to the optimization problem* (4.39) *is given by the matrix Y whose rows are the d generalized eigenvectors of the matrix $(L, \mathcal{D})$ associated with its second to $(d+1)$-th smallest generalized eigenvalues.*

Proof. Notice that the Lagrangian for this problem can be written as

$$\mathcal{L}(Y, \boldsymbol{\lambda}, \Lambda) = \operatorname{trace}(YLY^\top) + \boldsymbol{\lambda}^\top Y\mathcal{D}\mathbf{1} + \operatorname{trace}(\Lambda(I - Y\mathcal{D}Y^\top)), \tag{4.40}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^d$ and $\Lambda = \Lambda^\top \in \mathbb{R}^{d\times d}$ are, respectively, a vector and matrix of Lagrange multipliers. Computing the derivative of $\mathcal{L}$ w.r.t. Y and setting it to zero yields $2YL + \boldsymbol{\lambda}\mathbf{1}^\top\mathcal{D} - 2\Lambda Y\mathcal{D} = \mathbf{0}$. Multiplying on the right by $\mathbf{1}$ and using the constraints $L\mathbf{1} = \mathbf{0}$ and $Y\mathcal{D}\mathbf{1} = \mathbf{0}$, we obtain $\boldsymbol{\lambda} = 0$. As a consequence,

$$YL = \Lambda Y\mathcal{D} \implies LY^\top = \mathcal{D}Y^\top\Lambda. \tag{4.41}$$

Following the same argument as in the proof of Theorem 2.3, one can show that Λ is diagonal. Therefore, the rows of Y are generalized eigenvectors of $(L, \mathcal{D})$ with generalized eigenvalues in the diagonal entries of Λ. Moreover, $YLY^\top = \Lambda Y\mathcal{D}Y^\top = \Lambda$ and so $\operatorname{trace}(YLY^\top) = \operatorname{trace}(\Lambda)$. Therefore, we must choose the smallest generalized eigenvalues of $(L, \mathcal{D})$. Since $\mathbf{1}$ is an eigenvector of L with zero eigenvalue, and the eigenvectors of L must be orthogonal to $\mathcal{D}\mathbf{1}$ (because $Y\mathcal{D}\mathbf{1} = \mathbf{0}$), the rows of the optimal Y are the d generalized eigenvectors of $(L, \mathcal{D})$ associated with its second to $(d+1)$-th smallest eigenvalues, as claimed. □

The LE algorithm is summarized in Algorithm 4.2.

Algorithm 4.2 (Laplacian Eigenmaps)

Input: A set of points $\{\boldsymbol{x}_j\}_{j=1}^N$ lying in a manifold $\mathcal{M}$ and integers K and d.

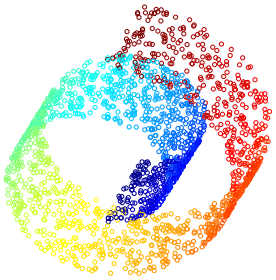
1: Find the K-nearest neighbors (K-NN) of each data point $\boldsymbol{x}_j, j = 1, \dots, N$, according to some distance dist in $\mathcal{M}$.

2: Define a matrix of weights $W \in \mathbb{R}^{N\times N}$ whose entries w_{ij} measure the similarity between two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ and are computed as

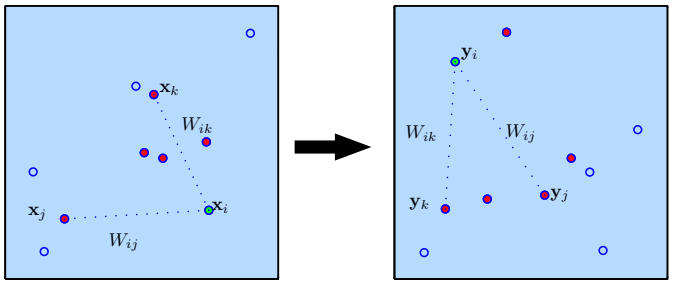
$$w_{ij} = \begin{cases} e^{-\frac{\operatorname{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j)^2}{2\sigma^2}} & \text{if } \boldsymbol{x}_i \text{ is a } K\text{-NN of } \boldsymbol{x}_j \text{ or viceversa} \\ 0 & \text{else} \end{cases} \tag{4.42}$$

3: Let $\mathcal{D}$ be a diagonal matrix with entries $d_{ii} = \sum_j w_{ij}$ and let $L = \mathcal{D} - W$. Find a matrix $Y = [\boldsymbol{y}_1, \dots, \boldsymbol{y}_N] \in \mathbb{R}^{d\times N}$ whose rows are the d generalized eigenvectors of the pair $(L, \mathcal{D})$ associated with its second to $(d+1)$th smallest generalized eigenvalues. That is, solve for Y from $YL = \Lambda Y\mathcal{D}$, where Λ is a diagonal matrix with the generalized eigenvalues in its diagonal.

Output: A set of points $\{\boldsymbol{y}_j\}_{j=1}^N$ lying in $\mathbb{R}^d$.



(a) A set of points drawn from a 2-dimensional manifold embedded in $\mathbb{R}^3$



(b) Learning matrix of coefficients (c) Low-dimensional embedding

Figure 4.4. Dimensionality reduction from $\mathbb{R}^D$ to $\mathbb{R}^d$ using LLE

4.2.3 Locally Linear Embedding (LLE)

Another popular manifold learning approach is Locally Linear Embedding (LLE) [Roweis and Saul, 2000, Roweis and Saul, 2003]. This approach aims to capture the geometry of a manifold $\mathcal{M}$ by exploiting the fact that the local neighborhood of a point $\boldsymbol{x} \in \mathcal{M}$ can be well approximated by the affine subspace spanned by $\boldsymbol{x}$ and its K-NNs. These *locally linear* approximations are then used to find a low-dimensional embedding that gives a small reconstruction error with respect to such approximations.

The first step of LLE is to approximate each data point $\boldsymbol{x}_j$ as an affine combination of its K-NNs. Intuitively, this step is analogous to approximating the tangent space of the manifold at the point $\boldsymbol{x}_j$ by the affine subspace spanned by $\boldsymbol{x}_j$ and its K-NNs. For a manifold of dimension d, the tangent space at each point is a d-dimensional affine subspace. Therefore, we need at least d-NNs to reconstruct this subspace, i.e., we need to choose $K \geq d$. On the other hand, K cannot be chosen to be too large. Otherwise, each data point would be written as an affine combination of too many points and the affine coefficients would not be unique. Given K, the K-NNs $\{\boldsymbol{x}_{j_k}\}_{k=1}^K$ of each data point $\boldsymbol{x}_i$ are typically found using the Euclidean distance. However, other distances can be used as well.

To approximate each data point $\boldsymbol{x}_j$ as an affine combination of its K-NNs, we search for a matrix $C = [c_{ij}] \in \mathbb{R}^{N \times N}$ that minimizes the reconstruction error

$$E(C) = \frac{1}{2} \sum_{j=1}^{N} \left\| \boldsymbol{x}_j - \sum_{i=1}^{N} c_{ij} \boldsymbol{x}_i \right\|^2, \tag{4.43}$$

subject to (i) $c_{ij} = 0$ if $\boldsymbol{x}_i$ is not a K-NN of $\boldsymbol{x}_j$ and (ii) $\sum_{i=1}^{N} c_{ij} = 1$. The first constraint expresses point $\boldsymbol{x}_j$ as an affine combination of only its K-NNs, while the second constraint ensures that the combination of the K-NNs is affine.

Let $j_1, \ldots, j_K$ denote the indexes of the K-NN of $\boldsymbol{x}_j$. Since $c_{ij} = 0$ when $\boldsymbol{x}_i$ is not a K-NN of $\boldsymbol{x}_j$, we only need to keep track of K affine coefficients for each point $\boldsymbol{x}_j$. Let $\boldsymbol{c}_j = [c_{j_1,j}, \ldots, c_{j_K,j}]^\top \in \mathbb{R}^K$ be the vector of such coefficients and let $G_j = [g^j_{il}] \in \mathbb{R}^{K \times K}$ be the local Gram matrix at $\boldsymbol{x}_j$, which is defined as $g^j_{il} = (\boldsymbol{x}_i - \boldsymbol{x}_j)^\top (\boldsymbol{x}_l - \boldsymbol{x}_j)$ for i, l such that $\boldsymbol{x}_i$ and $\boldsymbol{x}_l$ are K-NNs of $\boldsymbol{x}_j$. With this notation, the j-th term of (4.43) can be written as

$$\begin{aligned} \left\| \boldsymbol{x}_j - \sum_{i=1}^{N} c_{ij} \boldsymbol{x}_i \right\|^2 &= \left\| \sum_{i=1}^{N} c_{ij} (\boldsymbol{x}_i - \boldsymbol{x}_j) \right\|^2 = \sum_{il} c_{ij} c_{lj} (\boldsymbol{x}_i - \boldsymbol{x}_j)^\top (\boldsymbol{x}_l - \boldsymbol{x}_j) \\ &= \sum_{il} c_{ij} c_{lj} g^j_{il} = \boldsymbol{c}_j^\top G_j \boldsymbol{c}_j. \end{aligned} \tag{4.44}$$

Therefore, the optimization problem in (4.43) is equivalent to

$$\min_{\{\boldsymbol{c}_j\}} \ \frac{1}{2} \sum_{j=1}^{N} \boldsymbol{c}_j^\top G_j \boldsymbol{c}_j \quad \text{s.t.} \quad \mathbf{1}^\top \boldsymbol{c}_j = 1. \tag{4.45}$$

The Lagrangian for this problem is $\mathcal{L} = \frac{1}{2} \sum_{j=1}^{N} \boldsymbol{c}_j^\top G_j \boldsymbol{c}_j + \lambda_j (1 - \mathbf{1}^\top \boldsymbol{c}_j)$. Thus, the first order conditions for optimality are given by $G_j \boldsymbol{c}_j = \lambda_j \mathbf{1}$ and $\mathbf{1}^\top \boldsymbol{c}_j = 1$. Therefore, $\boldsymbol{c}_j = \lambda_j G_j^{-1} \mathbf{1}$ and $\lambda_j^{-1} = \mathbf{1}^\top G_j^{-1} \mathbf{1}$, so that

$$\boldsymbol{c}_j = \frac{G_j^{-1} \mathbf{1}}{\mathbf{1}^\top G_j^{-1} \mathbf{1}} \in \mathbb{R}^K, \tag{4.46}$$

provided that G_j is of full rank K.

Notice that the affine coefficients c_{ij} are invariant to rotations, translations and scalings of all the data points. The invariance to rotations and translations follows from the invariance properties of the Gram matrix G_j. Specifically, notice that if each $\boldsymbol{x}_j$ is transformed to $R\boldsymbol{x}_j + \boldsymbol{t}$, where $R \in SO(3)$ and $\boldsymbol{t} \in \mathbb{R}^3$, then $\boldsymbol{x}_j - \boldsymbol{x}_i$ is transformed to $R(\boldsymbol{x}_j - \boldsymbol{x}_i)$, and so G_j is not affected. The invariance to scalings follows from the fact that the Gram matrix appears both in the numerator and denominator of (4.46). Therefore, the affine coefficients characterize the intrinsic geometric properties of each neighborhood of the data in $\mathbb{R}^D$.

The second step of LLE is to find a representation $Y = [\boldsymbol{y}_1, \dots, \boldsymbol{y}_N] \in \mathbb{R}^{d \times N}$ that minimizes

$$\phi(Y) = \sum_{j=1}^{N} \left\| \boldsymbol{y}_j - \sum_{i=1}^{N} c_{ij} \boldsymbol{y}_i \right\|^2. \tag{4.47}$$

Notice that the objective in (4.47) is the same as the reconstruction error in (4.43), but obtained with respect to the low-dimensional representation Y rather than with respect to the original data X. Notice also that the global minimum is obtained when $Y = 0$, thus we need to impose additional constraints on the low-dimensional representation in order to avoid trivial solutions. LLE requires the low-dimensional representation Y to satisfy the following constraints

$$\sum_{j=1}^{N} \boldsymbol{y}_j = \mathbf{0} \quad \text{and} \quad \frac{1}{N} \sum_{j=1}^{N} \boldsymbol{y}_j \boldsymbol{y}_j^\top = I. \tag{4.48}$$

The first constraint requires the low-dimensional representation to be centered at the origin, as in the case of PCA (see Chapter 2). The second constraint enforces the low-dimensional representation to have unit covariance and is an arbitrary constraint to ensure that $\operatorname{rank}(Y) = d$.

To find the optimal Y, notice that the optimization problem can be written as

$$\min_{Y} \ \|Y - YC\|_F^2 \quad \text{s.t.} \quad Y\mathbf{1} = \mathbf{0} \quad \text{and} \quad \frac{1}{N} YY^\top = I. \tag{4.49}$$

Proposition 4.9. *The solution to the optimization problem* (4.49) *is given by the matrix Y whose rows are the d eigenvectors of the matrix $L = (I - C)(I - C)^\top$ associated with its second to $(d+1)$-th smallest eigenvalues.*

Proof. Notice that $\|Y - YC\|_F^2 = \operatorname{trace}(Y(I-C)(I-C)^\top Y^\top)$. Therefore, the optimization problem (4.49) is of the same form as that in (4.39), with L replaced by $(I-C)(I-C)^\top$ and $\mathcal{D}$ replaced by $\frac{I}{N}$. Therefore, the results follows by direct application of Proposition 4.8. □

In summary, LLE is a manifold learning algorithm that uses the data X to construct a matrix of affine coefficients C, which captures the local geometry of the manifold. The low-dimensional representation is then obtained from the eigenvectors of the matrix $L = (I-C)(I-C)^\top$ associated to its 2-nd to $(d+1)$-th smallest eigenvalues. The LLE algorithm is summarized in Algorithm 4.3.

4.3 Spectral Embedding and K-Means Clustering

The reader probably has noticed in the above section that low-dimensional embeddings given by LE and LLE, at the high-level, are rather similar in several aspects:

Algorithm 4.3 (Locally Linear Embedding)

Input: A set of points $\{\boldsymbol{x}_j\}_{j=1}^N$ lying in a manifold $\mathcal{M}$ and integers K and d.

1: Find the K-nearest neighbors (K-NN) of each data point $\boldsymbol{x}_j, j = 1, \ldots, N$, according to some distance dist in $\mathcal{M}$.

2: Approximate each point $\boldsymbol{x}_j \approx \sum c_{ij}\boldsymbol{x}_i$ as an affine combination of its K-NN with coefficients the c_{ij} obtained as per (4.46)

3: Let the rows of the matrix $Y = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N] \in \mathbb{R}^{d\times N}$ be the d eigenvectors of the matrix $L = (I - C)(I - C)^\top$ associated with its second to $(d + 1)$-th smallest eigenvalues.

Output: A set of points $\{\boldsymbol{y}_j\}_{j=1}^N$ lying in $\mathbb{R}^d$.

1. They both map the original data points $\boldsymbol{x}_i \in \mathbb{R}^D$ to a new set of data points $\boldsymbol{y}_i$ in $\mathbb{R}^d$. The mapping does not aim to provide any parametric representation of $\boldsymbol{x}_i$ in its original space, and instead only aim to preserve certain geometric properties of the original data such as local position or neighboring relationships.

2. They both start with constructing a pairwise weight w_{ij} between any pair of points that reflects the desired geometric properties to be preserved. A higher weight indicates the two points are "similar" in such properties.

3. If we view the points and their pairwise weights as a weighted graph $G = (V, E)$, where V has N vertices (corresponding to the N sample points) and E are the weighted edges, then the final embeddings of LE and LLE are given by the second to $(d+1)$th eigenvectors of the Laplacian matrix L of the graph or its normalized version LD^{-1}.

Indeed, these methods can be viewed as special cases of a very general data modeling method that uses spectrum of graph Laplacians to provide new representations for data, known as *spectral embedding*. Notice that there is one more common feature about the LE and LLE methods: the smallest eigenvalue of their Laplacian is always zero and there is only a unique zero eigenvalue if the pairwise weights give a connected graph. Although in the context of LE or LLE, the eigenvector associated with the zero eigenvalue gives a trivial embedding of the data – mapping all points to the same coordinate, this seemingly useless fact has significant implications in using spectral mapping to extract important topological properties such as whether the graph is connected or not or equivalently, whether the data have multiple clusters.

Since data clustering will be a central theme for the rest of the book[7], we here give a brief survey for spectral embedding and its key properties, which will become useful in our future development.

[7] as we will see that the spectral mapping method will play a crucial role in one of the approaches to subspace clustering

Given a set of data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathbb{R}^D$, we associate it with an undirected graph $G = (V, E)$ of N vertices $v_1, \ldots, v_N$, each vertex v_i corresponding to one data point $\boldsymbol{x}_i$. A weight $w_{ij} \geq 0$ is given on the edge e_{ij} for each pair of vertices (v_i, v_j) and $w_{ij} = 0$ mean the two vertices are not connected. The weight w_{ij} is used to describe "similarity" between the two points $\boldsymbol{x}_i, \boldsymbol{x}_j$ in terms of their properties in the original space $\mathbb{R}^D$. For instance, w_{ij} can be chosen as in the Laplacian Embedding, or we can set $w_{ij} = 1$ if and only if $\boldsymbol{x}_i$ is within a small neighborhood of $\boldsymbol{x}_j$. We denote the set of all $N \times N$ weights as a matrix $W = (w_{ij})$. As the graph is undirected, that implies the matrix W is symmetric: $W = W^\top$.

We further define an $N \times N$ diagonal matrix D whose diagonal entries d_{ii} are the degrees of the vertices: $d_{ii} = \sum_j w_{ij}$. The *Laplacian* of the graph G is defined to be the matrix $L \doteq D - W$.

Proposition 4.10 (Basic Properties of Laplacian)**.** *Given an undirected graph G of N vertices with non-negative weights, its Laplacian matrix $L \in \mathbb{R}^{N \times N}$ has the following properties:*

- *For any vector $u \in \mathbb{R}^N$, we have $u^\top L u = \sum_{i,j} w_{ij}(u_i - u_j)^2 \geq 0$. Hence L is positive semi-definite.*
- *The smallest eigenvalue is zero since $L\mathbf{1} = 0$ by definition of the degree matrix.*

We leave the rest of the proof of this statement as an exercise to the reader.

Although in LE and LLE, we are more interested in the non-zero eigenvalues and eigenvectors, it turns out that the zero eigenvector(s) encode very important topological properties of the graph: the number of zero eigenvalues correspond to the number of connected components of the graph; and the eigenvectors encode information which component each vertex belongs to. More precisely, we have the following properties.

Proposition 4.11 (Number of Connected Subgraphs)**.** *Given an undirected graph G of N vertices with non-negative weights, the number of zero eigenvalues n of its Laplacian matrix L is exactly the number of connected components of the graph $G = G_1 \cup G_2 \cup \cdots \cup G_n$ with $G_i \cap G_j = 0$. The null space null(L) is exactly spanned by the indicator functions of these disconnected subgraphs:*

$$\textit{null}(L) = \textit{span}\{\mathbf{1}_{G_1}, \mathbf{1}_{G_2}, \ldots, \mathbf{1}_{G_n}\}. \tag{4.50}$$

Proof. Suppose $u \in \mathbb{R}^N$ is an eigenvector associated with a zero eigenvalue, then we have

$$u^\top L u = \sum_{i,j} w_{ij}(u_i - u_j)^2 = 0. \tag{4.51}$$

Since $w_{ij} \geq 0$, for the above equality to hold, we must have $u_i = u_j$ whenever $w_{ij} > 0$. That is, if two vertices v_i and v_j belong to the same connected component, the corresponding values of the eigenvector must be equal. From this fact,

we can conclude that the dimension of the null space must be less or equal to the number of connected components. To prove the equality must hold, it is easy to verify that if the graph has n disconnected components, each of the indicator functions $\mathbf{1}_{G_i}$ is indeed an (independent) eigenvector associated with the zero eigenvalue. □

The above property of the Laplacian matrix implies that the null space of the Laplacian matrix encodes precise information about the membership of the vertices in the n connected components.

Proposition 4.12 (Null Space of Laplacian)**.** *Any n linearly independent vectors $u_1, \ldots u_n \in \mathbb{R}^N$ in the null space of L (hence also that of LD^{-1}) has the form:*

$$[u_1, \ldots, u_n]^\top = A[\mathbf{1}_{G_1}, \mathbf{1}_{G_2}, \ldots, \mathbf{1}_{G_n}] \in \mathbb{R}^{n \times N} \tag{4.52}$$

for some non-singular matrix $A \in \mathbb{R}^{n \times n}$.

We leave the proof of this fact as an exercise for the reader. If we view the columns of the matrix $Y \doteq [u_1, \ldots, u_n]^\top \in \mathbb{R}^{n \times N}$ as a new embedding of the points in $\mathbb{R}^n$, then Y has a very simple but important property: $\boldsymbol{y}_i = \boldsymbol{y}_j$ if and only if the two vertices v_i and v_j belong to the same connected component. That is, all the points $\boldsymbol{y}_i$ belong to one of the n centers in $\mathbb{R}^n$. Hence, if we are only interested in the topology of the graph, we only care about eigenvectors that are associated with the zero eigenvalue, not other eigenvectors (like LE or LLE does).

In practice the data may contain noise and the similarity measure $W = (w_{ij})$ introduced may be imprecise. As result, the eigen-subspace that is associated with the n smallest eigenvalues of L might be noisy, so are points $\{\boldsymbol{y}_i\}$ computed from it. Nevertheless, these points should still cluster around n centers in $\mathbb{R}^n$. For simplicity, we may assume that each cluster of points are distributed around their center (mean) as an isotropic Gaussian in $\mathbb{R}^n$. Hence, all the points $\boldsymbol{y}_i$ can be modeled as a mixture of n Gaussians. To identify these n centers and group the points to their respective centers, a very popular method is the K-means algorithm which essentially estimates the n Gaussians via the minimax principle in Appendix B.3.2.

If we combine the spectral embedding technique with the K-means method, it leads to a popular data clustering algorithm, known as *spectral clustering*. For completeness and clarity, we summarize the overall process in Algorithm 4.5.

Notice that in the ideal noise free case, the null spaces of L and LD^{-1} are exactly the same. Nevertheless, when there are noise in the data or the similarity measure W is imprecise, using LD^{-1} introduces additional "normalization" effect and typically leads to more numerically stable estimate of the null space than directly using L, especially when the degrees of the graph vertices are not so balanced. In the simple two-cluster case, the null spectrum of LD^{-1} gives exactly the same solution to the well-known "normalized cut" formulation [**?**], which was initially proposed to approximately solve the NP-hard minicut problem of a graph. Early empirical success of this formulation in problems such as image segmentation has made spectral methods very popular in the community of computer

Algorithm 4.4 (K-means for Mixture of Isotropic Gaussians)

Input: A set of points $\{\boldsymbol{y}_j\}_{j=1}^N$ clustered around n centers in $\mathbb{R}^n$.

1: **Initialization:** Given n initial guess of the n centers $\boldsymbol{c}_1^{(0)}, \ldots, \boldsymbol{c}_n^{(0)}$.
2: **while** (the clusters and their centers do not converge) **do**
3: For each point $\boldsymbol{y}_j$, assign it to the cluster $C_i^{(k)}$ with

$$i = \arg\min_m \|\boldsymbol{y}_j - \boldsymbol{c}_m^{(k)}\|_2^2.$$

4: Update the centers $\boldsymbol{c}_i^{(k+1)}$ to be the mean of all points $\boldsymbol{y}_j$ that belong to the cluster $C_i^{(k)}$:

$$\boldsymbol{c}_i^{(k+1)} \leftarrow \frac{1}{|C_i^{(k)}|} \sum_{j \in C_i^{(k)}} \boldsymbol{y}_j.$$

5: **end while**

Output: The converged n clusters and their centers: $\{C_i, \boldsymbol{c}_i\}$.

Algorithm 4.5 (Spectral Clustering)

Input: Construct a similarity graph G for set of points $\{\boldsymbol{x}_j\}_{j=1}^N \in \mathbb{R}^D$ with a weight matrix $W \in \mathbb{R}^{N \times N}$. Specify the number of clusters n.

1: **Initialization:** Compute the degree matrix $D = \text{diag}(W\mathbf{1})$ and the Laplacian matrix $L = D - W$.
2: Compute the first n left eigenvectors $(u_1, \ldots, u_n)$ of L or LD^{-1} associated with the n smallest eigenvalues.
3: Let $Y \doteq [u_1, \ldots, u_n]^\top \in \mathbb{R}^{n \times N}$ be the matrix that collect the n left eigenvectors as its rows.
4: Let $\boldsymbol{y}_j, j = 1, \ldots, N$ be the N column vectors of Y.
5: Cluster the N points $\{\boldsymbol{y}_j\}$ using the K-means Algorithm 4.4 into n clusters $C_1, \ldots, C_n$.

Output: The converged n clusters: $\{C_i\}$.

vision and machine learning. In the more general n-cluster case, when the similarity measure is noisy and the resulting graph does not have clearly disconnected subgraphs, the resulting clustering using small eigenvalues of its Laplacian gives be a partition of the graph that approximately minimizes the multi-way cutting cost.

In essence, the role of Laplacian is, through its null space, map the original data points $\{\boldsymbol{x}_j\} \subset \mathbb{R}^D$ to the embedded points $\{\boldsymbol{y}_j\} \subset \mathbb{R}^n$. The original data may have complex structures and deny a simple clustering solution; whereas the (cluster) structures of the embedded data become much simpler, leading to arguably the simplest class of clustering problems. Of course, there is no free lunch. The difficulty in clustering the original data needs to be alleviated through the design

of a good similarity measure $W = (w_{ij})$. The performance of the spectral clustering method highly depends on the design of the similarity measure $W = (w_{ij})$. In general, there is no theory that characterizes precisely how certain choice of similarity measure would influence the resulting clusters. Nevertheless, in Chapter **??** we will see that at least for the situation when the clusters of original data are subspaces, one could design similarity measures in a principled manner with good theoretical guarantee.

4.4 Bibliographic Notes

Nonlinear dimensionality reduction (NLDR) refers to the problem of finding a low-dimensional representation for a set of points lying in a nonlinear manifold embedded in a high-dimensional space. This question of how to detect and represent low-dimensional structure in high-dimensional data is fundamental to many disciplines and several attempts have been made in different areas to address this question. For example, the number of pixels in an image can be rather large, yet most computer vision models use only a few parameters to describe the geometry, photometry and dynamics of the scene. Since most datasets often have fewer degrees of freedom than the dimension of the ambient space, NLDR is fundamental to many problems in computer vision, machine learning and pattern recognition.

When the data lives in a low-dimensional linear subspace of a high-dimensional space, simple linear methods such as Principal Component Analysis (PCA) [Hotelling, 1933] and metric Multi-Dimensional Scaling (MDS) [Cox and Cox, 1994] can be used to learn the subspace and its dimension. However, when the data lies in a low-dimensional submanifold, its structure may be highly nonlinear, hence linear dimensionality reduction methods are likely to fail. This has motivated extensive efforts toward developing NLDR algorithms for computing low-dimensional embeddings. One of the first generalizations of PCA to nonlinear manifolds is the work of [Hastie, 1984] and [Hastie and Stuetzle, 1989] on principal curves and surfaces. The principal curve of a dataset, which generalizes the notion of a principal component, is a curve that passes through the *middle* of the data points and minimizes the sum of squared distances from the data points to the curve. A more general approach however is to find a nonlinear embedding map, or equivalently a kernel function, such that the embedded data lie on a linear subspace. Such methods are referred to as nonlinear kernel PCA [Schölkopf et al., 1998, Schölkopf and Smola, 2002]. A huge family of such algorithms computes a low-dimensional representation from the eigenvectors of a matrix constructed from the local geometry of the manifold. Such algorithms include ISOMAP [Tenenbaum et al., 2000], locally linear embedding (LLE) [Roweis and Saul, 2000, Roweis and Saul, 2003], and its variants such as Laplacian Eigenmaps (LE) [Belkin and Niyogi, 2002], Hessian LLE [Donoho and Grimes, 2003], Local Tangent Space Alignment (LTSA) [Zhang and Zha, 2005], maximum variance unfolding [Weinberger and Saul, 2004], and conformal eigen-

maps [Sha and Saul, 2005]. For a survey of many of these algorithms, we refer the reader to [Burges, 2005, Lee and Verleysen, 2007, Burges, 2010].

Survey about spectral method. Such as Ng, Weiss and Jordan [Ng et al., 2001], or Luxberg [von Luxburg, 2007] for a more thorough review etc.

4.5 Exercises

Exercise 4.1 Show that the following functions are positive-semidefinite kernels

1. $k(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x})^\top \phi(\boldsymbol{y})$ for some embedding function $\phi : \mathbb{R}^D \to \mathbb{R}^M$.
2. $k_P(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}^\top \boldsymbol{y})^n$ for fixed $n \in \mathbb{N}$.
3. $k_G(\boldsymbol{x}, \boldsymbol{y}) = \exp\big(-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{2}\big)$.

Exercise 4.2 Consider the polynomial kernel in $[-1, 1]^2 \times [-1, 1]^2$ defined as $k(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}^\top \boldsymbol{y})^2 = (x_1 y_1 + x_2 y_2)^2$. Define the operator

$$\mathcal{L}(f)(\boldsymbol{x}) = \int k(\boldsymbol{x}, \boldsymbol{y}) f(\boldsymbol{y}) d\boldsymbol{y}. \tag{4.53}$$

Show that the eigenfunctions of $\mathcal{L}$ corresponding to nonzero eigenvalues are of the form $\psi(\boldsymbol{x}) = c_1 x_1^2 + c_2 x_1 x_2 + c_3 x_2^2$. Show that there are three such eigenfuctions, where (c_1, c_2, c_3) and λ are obtained from

$$\begin{bmatrix} 4/5 & 0 & 4/9 \\ 0 & 8/9 & 0 \\ 4/9 & 0 & 4/5 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \lambda \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}. \tag{4.54}$$

Exercise 4.3 (Karhunen-Loève Transform). The Karhunen-Loève transform (KLT) can be thought of as a generalization of PCA from a (finite-dimensional) random vector $\boldsymbol{x} \in \mathbb{R}^D$ to an (infinite-dimensional) random process $x(t), t \in \mathbb{R}$. When $x(t)$ is a (zero-mean) second-order stationary random process, its auto correlation function is defined to be $K(t, \tau) \doteq \mathbb{E}[x(t)x)(\tau)]$ for all $t, \tau \in \mathbb{R}$.

1. Show that $K(t, \tau)$ has a family of orthonormal eigen-functions $\{\phi_i(t)\}_{i=1}^\infty$ that are defined as

$$\int K(t, \tau)\phi_i(\tau)\, d\tau = \lambda_i \phi_i(t), \quad i = 1, 2, \ldots. \tag{4.55}$$

   (Hint: First show that $K(t, \tau)$ is a positive definite function and then use Mercer's Theorem.)

2. Show that with respect to the eigen-functions, we original random process can be decomposed as

$$x(t) = \sum_{i=1}^{n} x_i \phi_i(t), \tag{4.56}$$

   where $\{x_i\}_{i=1}^\infty$ are a set of uncorrelated random variables.

Exercise 4.4 (Full Rank of Gaussian RBF Gram Matrices) Suppose that you are given N distinct points $\{\boldsymbol{x}_i\}_{i=1}^N$. If $\sigma \neq 0$, then the matrix $K \in \mathbb{R}^{N\times N}$ given by

$$K_{ij} = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right) \tag{4.57}$$

has full rank.

Exercise 4.5 Let $\{\boldsymbol{x}_i \in \mathbb{R}^D\}_{j=1}^N$ be a set of points you believe live in a manifold of dimension d. Imagine you have applied PCA, KPCA with kernel k and LLE with K-NN to the data. Assume now you are given a new point $\boldsymbol{x} \in \mathbb{R}^D$ and you wish to find its corresponding point $\boldsymbol{y} \in \mathbb{R}^d$ according to each one of the three methods. How would you compute $\boldsymbol{y} \in \mathbb{R}^d$ without applying PCA, KPCA or LLE from scratch to the $N+1$ points? Under what conditions the solution you propose is equivalent to applying PCA, KPCA or LLE to the $N+1$ points?

Exercise 4.6 Implement the KPCA algorithm for an arbitrary kernel function `kernel.m`. The format of your function should be as follows.

Function	**`[y]=kpca(x,d,kernel,params)`**
Parameters	
`x`	$D \times N$ matrix whose columns are the data points
`d`	dimension of the projected dataset
`kernel`	name of the MATLAB function that computes the kernel `k = kernel(x1,x2,params)`
`params`	parameters needed by the kernel function, such as the degree in the polynomial kernel or the standard deviation in the Gaussian kernel
Returned values	
`y`	$d \times N$ matrix containing the projected coordinates
Description	
Computes the kernel principal components of a set of points.	

Also implement the functions `k = poly_kernel(x1,x2,n)` for the polynomial kernel $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = (\boldsymbol{x}_1^\top \boldsymbol{x}_2)^n$ and `k = gauss_kernel(x1,x2,sigma)` for the Gaussian kernel $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \exp(-\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2/\sigma^2)$, where $k \in \mathbb{R}^{N\times N}$ and $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^{D\times N}$. Try your code on the example given in class. The code generating the data can be found at `http://www.kernel-machines.org/code/kpca_toy.m`

4.A Laplacian Eigenmaps: Continuous Formulation

Laplacian Eigenmaps (LE) [Belkin and Niyogi, 2002] is a popular dimensionality reduction method. It aims to capture the geometry of a manifold by finding a low-dimensional representation such that nearby points in the manifold are mapped to nearby points in the low-dimensional embedding. In the chapter, we have seen how such a goal can be achieved for a collection of sample points drawn from the manifold. Nevertheless, the original derivation of LE draws inspiration from a similar goal for embedding a continuous manifold into a (low-dimensional) Euclidean space. To complement the discrete LE method described in the chapter, in this appendix, we describe LE in the continuous setting,.

In the continuous setting, the goal of LE is to find d functions from a manifold $\mathcal{M}$ to the real line $\mathbb{R}$ that preserve *locality*, i.e., functions that map nearby points in the manifold to nearby points in the real line. When $\mathcal{M} = \mathbb{R}$, one such function is the identity map, which perfectly preserves locality. For higher-dimensional manifolds, a function f that maps nearby points to nearby points should have a small derivative, hence we could find it by minimizing $\int_{\mathbb{R}} (\frac{df}{dx})^2 dx$.

More generally, let $f : \mathcal{M} \to \mathbb{R}$ be a map from a manifold $\mathcal{M}$ to the real line and assume that it is twice differentiable. We can find functions that map nearby points in the manifold to nearby points in the real line by minimizing

$$\int_{\mathcal{M}} \|\nabla f(\boldsymbol{x})\|^2 \, d\boldsymbol{x} \quad \text{s.t.} \quad \|f\|^2 = \int_{\mathcal{M}} f(\boldsymbol{x})^2 d\boldsymbol{x} = 1, \tag{4.58}$$

where $\nabla f \in T_{\boldsymbol{x}}\mathcal{M}$ is the gradient of f and the constraint $\|f\| = 1$ is added to prevent the trivial solution $f \equiv 0$

We can solve the above optimization problem using the method of Lagrange multipliers. The Lagrangian is given by

$$\mathcal{L}(f, \lambda) = \int \big(\|\nabla f(\boldsymbol{x})\|^2 + \lambda (f^2(\boldsymbol{x}) - 1) \big) d\boldsymbol{x}. \tag{4.59}$$

Using calculus of variations, we can compute the gradient of $\mathcal{L}$ w.r.t. f as

$$\nabla_f \mathcal{L} = -2\Delta f + 2\lambda f. \tag{4.60}$$

where Δ is the Laplace-Beltrami operator on $\mathcal{M}$, which can be expressed in tangent coordinates z_i as $\Delta f = \sum_i \frac{\partial^2 f}{\partial z_i^2}$. Setting the gradient to zero, we obtain $\Delta f = \lambda f$, hence f is an eigenfunction of Δ with associated eigenvalue λ. Noticing that the cost function may be rewritten as

$$\int_{\mathcal{M}} \|\nabla f(\boldsymbol{x})\|^2 \, d\boldsymbol{x} = \int_{\mathcal{M}} \Delta f(\boldsymbol{x}) f(\boldsymbol{x}) \, d\boldsymbol{x} = \lambda \int_{\mathcal{M}} f^2(\boldsymbol{x}) \, d\boldsymbol{x} = 1, \tag{4.61}$$

we conclude that the functions f that minimize the cost function are eigenfunctions of Δ associated with the smallest eigenvalues. One such eigenfunction is the constant function, which is associated with the zero eigenvalue. This function maps all points in the manifold $\mathcal{M}$ to a single point on the real line $\mathbb{R}$. The optimal d-dimensional embedding is hence given by the d eigenfunctions corresponding to the second to $(d+1)$-th smallest eigenvalues.