Machine Learning Methods
MDS and Kernel PCA

By Zhu Xuelin
Email: `xuelin@u.nus.edu`

September 28, 2024

When it comes to dimension reduction, PCA is the most famous tool, which projects the raw data to a linear subspace to get dimension reduction. However, linear reduction may not capture the characters of the raw data such as the example in Figure 1.

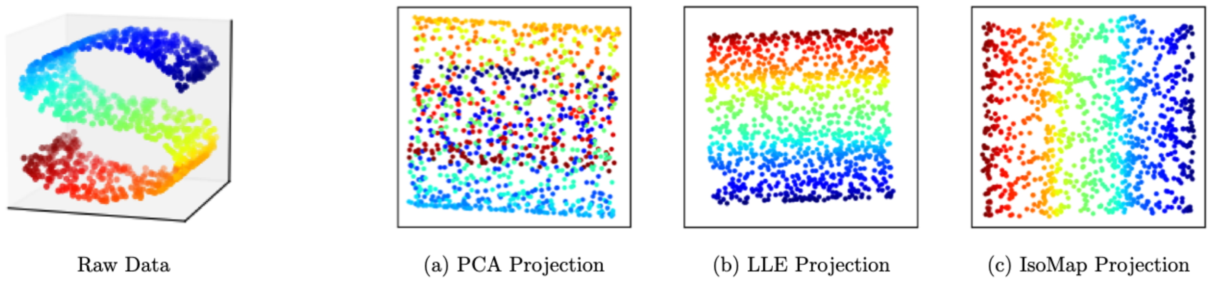

Figure 1: Manifold raw data space

As we see, the raw data lie in a curved surface in $\mathbb{R}^3$ with the shape of the letter 'S'. Under this circumstance, no matter which linear subspace we choose, PCA would not behave good enough as we wee in Figure 1(a). This is why we need the non-linear dimension reduction, as 1(b) and 1(c).

1 Multidimensional Scaling

We start with a linear method called *multidimensional scaling* (MDS), which is the source of non-linear method. Though it is a linear method, unlike the PCA utilizing the covariance structure, MDS aims to find a lower dimensional space, which minimizes the loss of similarity of raw data.

Problem Setup (MDS)**.** Suppose the raw observed data are $\mathbf{y}_1, \ldots, \mathbf{y}_N \in \mathbb{R}^q$, with a distance (dissimilarity) measure on that space:

$$d_1(\cdot\ ,\ \cdot) : \mathbb{R}^q \times \mathbb{R}^q \mapsto \mathbb{R}.$$

The goal is to find a lower dimensional representation of the data $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^p$, also with a distance measure $d_2(\cdot\ ,\ \cdot)$ on that space such that minimizes the dissimilarity differences, i.e.

$$\sum_{1 \le i,j \le N} \Big\{ d_1\left(\mathbf{y}_i, \mathbf{y}_j\right) - d_2\left(\mathbf{x}_i, \mathbf{x}_j\right) \Big\}^2.$$

Note that the distances d_1, d_2 should be previously set, so the optimization is with respect to $\mathbf{x}$'s.

The distance could be defined in many ways, such as

$$\text{Euclidean distance:} \quad d_{ij} = ||\mathbf{y}_i - \mathbf{y}_j|| \quad \text{for each pair } \mathbf{y}_i, \mathbf{y}_j \in \mathbb{R}^q.$$

No matter which distances we use, our goal is the same: dimension reduction with the similarity structure of observations preserved. And with some complex choices of d, we may recover non-linear structure, as we shall see in the kernel PCA topics. Let's start with the classical MDS.

1.1 The Classical MDS

In fact, to minimize the dissimilarity difference is equivalent to minimize the similarity difference. So, the classical MDS works with a similarity measure for $\mathbf{y}$'s and $\mathbf{x}$'s, letting

$$d_1(\mathbf{y}_i, \mathbf{y}_j) = \mathbf{y}_i^\top \mathbf{y}_j \quad \text{and} \quad d_2(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j.$$

The formal definition is as follow.

Definition 1 (Classical MDS)**.** Suppose the raw observed data are $\mathbf{y}_1, \ldots, \mathbf{y}_N \in \mathbb{R}^q$. The goal is to find a lower dimensional representation of the data $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^p$ such that

$$\min_{\mathbf{x}_1, \ldots, \mathbf{x}_N} \sum_{1 \le i,j \le N} \left(\mathbf{y}_i^\top \mathbf{y}_j - \mathbf{x}_i^\top \mathbf{x}_j \right)^2.$$

As usual, we assume the data is centered for simplicity. If we write the design matrices as

$$\mathbb{Y}_{N \times q} = \begin{pmatrix} \mathbf{y}_1^\top \\ \mathbf{y}_2^\top \\ \vdots \\ \mathbf{y}_N^\top \end{pmatrix} \quad \text{and} \quad \mathbb{X}_{N \times p} = \begin{pmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_N^\top \end{pmatrix},$$

the minimization problems could be written as

$$\min_{\mathbb{X}} \left|\left| \mathbb{Y}\mathbb{Y}^\top - \mathbb{X}\mathbb{X}^\top \right|\right|_F,$$

where the norm is Frobenius's norm, which is defined as

$$||A||_F := \sqrt{\operatorname{Tr}(A^\top A)} = \sqrt{\sum_{i,j} a_{ij}^2} \quad \text{(sum of square of all terms)}.$$

If we SVD design matrices as (suppose $q < N$)

$$\mathbb{Y} = U_{N \times q} D_{q \times q} V_{q \times q}^\top \quad \text{and} \quad \mathbb{X} = U_{N \times p}^* D_{p \times p}^* (V^*)_{p \times p}^\top,$$

we can see

$$\mathbb{Y}\mathbb{Y}^\top = U_{N \times q} D_{q \times q}^2 U_{q \times N}^\top \quad \text{and} \quad \mathbb{X}\mathbb{X}^\top = U_{N \times p}^* (D^*)_{p \times p}^2 (U^*)_{p \times N}^\top.$$

In essence, we are finding a low rank approximation $\mathbb{X}\mathbb{X}^\top$ for $\mathbb{Y}\mathbb{Y}^\top$. And we will call

$$\text{Gram matrix:} \quad \mathbb{Y}\mathbb{Y}^\top.$$

Fortunately, the classical MDS's optimization is completed by the following theorem.

Theorem 2 (Eckart–Young Theorem)**.** *The best p-rank approximation of $\mathbb{Y}\mathbb{Y}^\top$ is*

$$\mathbb{Y}\mathbb{Y}^\top \approx \sum_{j=1}^{p} \lambda_j \mathbf{u}_j \mathbf{u}_j^\top, \quad \textit{where } \lambda_j \textit{ and } \mathbf{u}_j \textit{ are the first } p \textit{ eigens of } \mathbb{Y}\mathbb{Y}^\top.$$

1.2 Comparison between MDS and PCA

The Eckart–Young Theorem states a very similar result to PCA:

- In the classical MDS, we do spectral decomposition of Gram matrix $\left(\mathbb{Y}\mathbb{Y}^\top\right)_{N\times N}$
- In the classical PCA, we do spectral decomposition of the sample covariance matrix $\left(\mathbb{Y}^\top\mathbb{Y}\right)_{q\times q}$

In fact, we shall see they are mathematically the same in the view of SVD of $\mathbb{Y}$.

1.2.1 Same Procedures by SVD

Now suppose

$$\mathbb{Y}_{N\times q} = U_{N\times r} D_{r\times r} V_{r\times q}^\top,$$

by simple algebra, leading to

$$\mathbb{Y}^\top\mathbb{Y} = VD^2V^\top \quad \text{and} \quad \mathbb{Y}\mathbb{Y}^\top = UD^2U^\top,$$

meaning that

- the columns of V are the eigenvectors of $(n-1)S = \mathbb{Y}^\top\mathbb{Y}$,
- the columns of U are the eigenvectors of Gram matrix $\mathbb{Y}\mathbb{Y}^\top$,
- and the two kinds of decomposition share the same eigenvalues D^2 (normally ordered $d_1 > \cdots > d_r$).

1.2.2 Same Projection Scores

Additionally, the classical MDS has the same projection scores $\mathbb{X}$ as the classical PCA. W.L.O.G., we assume we take $p = r$, and the proof for other case is the same. Recall that the score vector $\mathbf{x}_i$ and score matrix $\mathbb{X}$ are (PCA loading matrix now is denoted by V)

$$\mathbf{x}_i = V^\top \mathbf{y}_i \quad \Rightarrow \quad \mathbb{X} = \mathbb{Y}V = UDV^\top V = UD.$$

It remains to show that MDS has the same score matrix. Recall that in the last section, when we minimizing

$$\min_{\mathbb{X}} \left|\left|\mathbb{Y}\mathbb{Y}^\top - \mathbb{X}\mathbb{X}^\top\right|\right|_F,$$

Eckart and Young give the best approximation

$$\mathbb{Y}\mathbb{Y}^\top \approx \sum_{j=1}^{p} \lambda_j \mathbf{u}_j \mathbf{u}_j^\top, \quad \text{where } \lambda_j \text{ and } \mathbf{u}_j \text{ are the first } p \text{ eigens of } \mathbb{Y}\mathbb{Y}^\top,$$

i.e., the best approximates $\mathbb{X}\mathbb{X}^\top$ is (when $p = r$)

$$\mathbb{X}\mathbb{X}^\top = \sum_{j=1}^{r} \lambda_j \mathbf{u}_j \mathbf{u}_j^\top = U \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_r \end{pmatrix} U^\top = UD^2U^\top = (UD)(UD)^\top.$$

Therefore, the score matrix $\mathbb{X} = UD$, is the same as PCA!

2 Kernel PCA

Though the kernel PCA is named after PCA, the idea is more like the MDS, i.e. calculating the similarity measure of observations $\mathbb{Y}\mathbb{Y}^\top$, NOT the covariance structure $\mathbb{Y}^\top\mathbb{Y}$. And by choosing different kernels, we are 'projecting' raw data, which is not linear separable, to higher dimensions, where they become separable. Here is an example of $\mathbb{R}^2$ non-separable but $\mathbb{R}^3$ separable data.

Example 3. Originally, data sit in the $\mathbb{R}^2$ plane, circle-like. In this case, no matter which linear subspace we choose, the two classes could not be separated. However, if we choose the *higher dimension projection* function

$$\forall \mathbf{y} \in \mathbb{R}^2: \quad \phi(\mathbf{y}) = \begin{pmatrix} \mathbf{x}_1 = \mathbf{y}_1 \\ \mathbf{x}_2 = \mathbf{y}_2 \\ \mathbf{x}_3 = \mathbf{y}_1^2 + \mathbf{y}_2^2 \end{pmatrix},$$

then it becomes possible to separate them now by a linear subspace in $\mathbb{R}^3$.

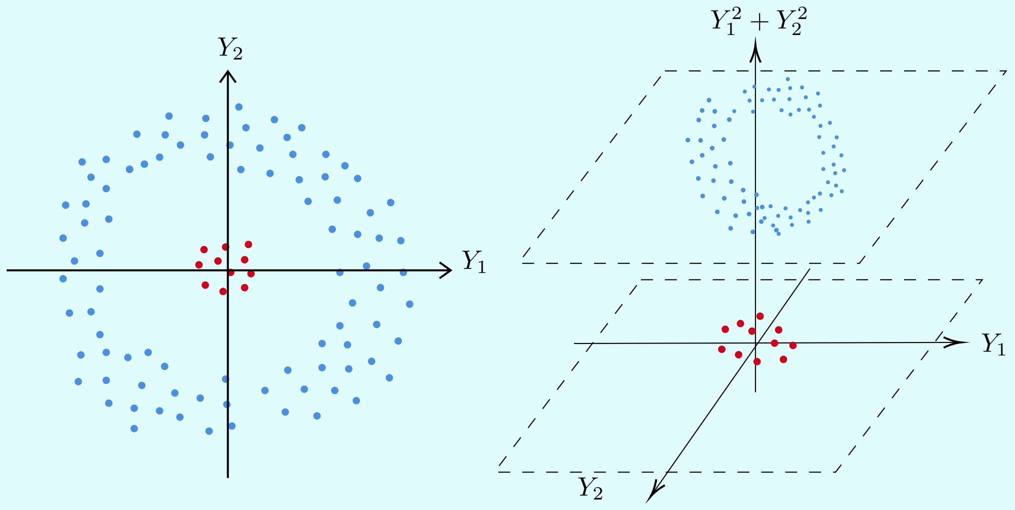

Figure 2: Higher dimension projection

So, we can conclude the steps for kernel PCA now. First, find a function ϕ which maps the data to a higher dimension space. And then do the classical MDS on the new data.

2.1 Kernel Functions

By the idea of the last paragraph, how to choose the ϕ function becomes a problem. By the magic of mathematics, we don't need to choose the function $\phi : \mathbb{R}^q \mapsto \mathbb{R}^p$, but a kernel function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$.

Definition 4 (Kernel function)**.** A function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is called kernel function if it is

- symmetric, i.e. $k(\mathbf{y}_1, \mathbf{y}_2) = k(\mathbf{y}_2, \mathbf{y}_1)$;
- positive semi-definite, i.e. $\forall n \in \mathbb{N}$, $\forall \mathbf{y}_1, \dots, \mathbf{y}_n \in \mathbb{R}^q$ and $\forall \alpha_1, \dots, \alpha_n \in \mathbb{R}$,

$$\sum_{i,j} \alpha_i \alpha_j k(\mathbf{y}_i, \mathbf{y}_j) \geq 0.$$

The reason to introduce the kernel functions is the following theorem.

Theorem 5 (Kernel implies embedding)**.** *A function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is a kernel if and only if there exists a Hilbert space $\mathscr{H}$ (with an inner product) and a feature map $\phi : \mathcal{X} \mapsto \mathscr{H}$ such that*

$$k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle .$$

Remark. To simply understand it, we can use the following idea. In practice, we don't need to pick the feature map $\phi(\cdot) : \mathbb{R}^q \mapsto \mathbb{R}^p$, but pick the kernel function $k(\cdot, \cdot) : \mathbb{R}^q \times \mathbb{R}^q \mapsto \mathbb{R}$, because once the kernel k is chosen, the feature map ϕ is automatically chosen as well.

Proof. The if part is easy by checking that $k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$ is indeed a symmetric kernel function. Now we prove the only if part, assuming that function k is a kernel. Our goal is to construct a Hilbert space $\mathscr{H}$ with an inner product $\langle \cdot, \cdot \rangle$, and a mapping $\phi : \mathcal{X} \mapsto \mathbb{R}$ such that

$$k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle .$$

First we consider the feature map $\phi : \mathcal{X} \mapsto \mathbb{R}^{\mathcal{X}}$, where $\mathcal{X}$ is the sample space and $\mathbb{R}^{\mathcal{X}}$ is the space of all real-valued functions from $\mathcal{X} \mapsto \mathbb{R}$, defined as

$$\mathbf{x} \mapsto \phi(\mathbf{x}) := k_{\mathbf{x}} := k(\mathbf{x}, \cdot).$$

That is, the point $\mathbf{x} \in \mathcal{X}$ is mapped to the function $k_{\mathbf{x}} : \mathcal{X} \mapsto \mathbb{R}$, $k_{\mathbf{x}}(\mathbf{y}) = k(\mathbf{x}, \mathbf{y})$. Now we collect all images of this kind of functions

$$\mathcal{G} := \left\{ \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \cdot) : \alpha_i \in \mathbb{R},\ n \in \mathbb{N},\ \mathbf{x}_i \in \mathcal{X} \right\},$$

i.e. the linear span of all feature maps $k_{\mathbf{x}}$. So, $\mathcal{G}$ contains all finite linear combinations of feature functions, and we can verify it is a vector space. As for the inner product on $\mathcal{G}$, we define

$$\begin{cases} \langle k_{\mathbf{x}}, k_{\mathbf{y}} \rangle = \langle k(\mathbf{x}, \cdot), k(\mathbf{y}, \cdot) \rangle := k(\mathbf{x}, \mathbf{y}) & \text{if } \mathbf{x}, \mathbf{y} \in \mathcal{X}, \\ \langle f, g \rangle := \sum_{ij} \alpha_i \beta_j k(\mathbf{x}_i, \mathbf{y}_j) & \text{if } f = \sum_i \alpha_i k(\mathbf{x}_i, \cdot) \text{ and } g = \sum_j \alpha_j k(\mathbf{y}_i, \cdot). \end{cases}$$

Defining this way, we can check (left as exercises) that this operation is well defined (the same function $f \in \mathcal{G}$ could have different linear combination presentation, verify they lead to the same result), and satisfies all properties that needed to be an inner product (this is where positive definiteness comes to use).

Finally, to make $\mathcal{G}$ a Hilbert space, we take its topological completion $\bar{\mathcal{G}}$, adding all limits of Cauchy sequences into this space. This result space $\mathcal{H} := \bar{\mathcal{G}}$ is the wanted space, and called the *reproducing kernel Hilbert space* (RKHS). And by construction, we have for all $\mathbf{x}, \mathbf{y} \in \mathcal{X}$,

$$k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle .$$

□

Why it is called reproducing kernel Hilbert space? Because of the following property.

Proposition 6. *Suppose $\mathscr{H}$ is a reproducing kernel Hilbert space and $f \in \mathscr{H}$ has the presentation of $f = \sum_i \alpha_i k(\mathbf{x}_i, \cdot)$, then for all $\mathbf{x} \in \mathcal{X}$,*

$$\langle f, k(\mathbf{x}, \cdot) \rangle = f(\mathbf{x}).$$

Proof. The result follows from

$$\langle f, k(\mathbf{x}, \cdot) \rangle = \left\langle \sum_i \alpha_i k(\mathbf{x}_i, \cdot), k(\mathbf{x}, \cdot) \right\rangle = \sum_i \alpha_i \langle k(\mathbf{x}_i, \cdot), k(\mathbf{x}, \cdot) \rangle = \sum_i \alpha_i k(\mathbf{x}_i, \mathbf{x}) = f(\mathbf{x}).$$

□

By this idea, what we need to do in practice, is choosing a kernel k, and calculating the Gram matrix $K = \left\{ k(\mathbf{y}_i, \mathbf{y}_j) \right\}_{ij}$, and implementing MDS. Doing this way, we even gain a benefit as in the next examples.

Example 7. Let's see few examples of kennel functions, some of whose feature map could be derived, but the others are not.

(a) Linear PCA chooses the kernel to be $k(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top \mathbf{y}$. So for a fixed point $\mathbf{x}$ in the sample $\mathcal{X}$, its feature map is defined by

$$\phi(\mathbf{x}) := k(\mathbf{x}, \cdot) := k_{\mathbf{x}}(\cdot) \quad \text{with inner product} \;\; k(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top \mathbf{y} = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle .$$

Therefore, we can tell the feature map $\phi(\mathbf{x}) = k_{\mathbf{x}}(\cdot)$ is an identity function that

$$\forall \mathbf{y} \in \mathcal{X} : \quad k_{\mathbf{x}}(\mathbf{y}) = \mathbf{x}.$$

(b) Polynomial kernel with order d chooses

$$k(\mathbf{y}_i, \mathbf{y}_j) = \left(1 + \mathbf{y}_i^\top \mathbf{y}_j\right)^d .$$

For example, when $d = 2$, $q = 2$, it is

$$\begin{aligned} k(\mathbf{y}_i, \mathbf{y}_j) &= \left(1 + \mathbf{y}_i^\top \mathbf{y}_j\right)^2 = (1 + y_{11} y_{21} + y_{21} y_{22})^2 \\ &= 1 + 2 y_{11} y_{21} + 2 y_{12} y_{22} + y_{11}^2 y_{21}^2 + y_{21}^2 y_{22}^2 + 2 y_{11} y_{12} y_{21} y_{22}. \end{aligned}$$

After simplification, we can also find the feature map $\mathbb{R}^2 \mapsto \mathbb{R}^6$:

$$\phi(\mathbf{y}) = k_{\mathbf{y}}(\cdot) = \begin{pmatrix} 1 & \sqrt{2}y_1 & \sqrt{2}y_2 & y_1^2 & y_2^2 & \sqrt{y_1 y_2} \end{pmatrix}^\top \quad \text{and} \quad k(\mathbf{y}_i, \mathbf{y}_j) = \langle \phi(\mathbf{y}_i), \phi(\mathbf{y}_j) \rangle \, .$$

(c) Gaussian kernel with dispersion parameter σ^2 is

$$k(\mathbf{y}_i, \mathbf{y}_j) = \exp\left\{ -\frac{||\mathbf{y}_i - \mathbf{y}_j||^2}{\sigma^2} \right\}.$$

However, we can not find an explicit expression for the feature map ϕ under this kernel, but its existence is guaranteed by the last theorem.

These examples show that specifying kernel function is sometimes more powerful than specifying the feature map since some kernel, such as the Gaussian, conveys a good meaning of distance but has no explicit feature map expression.

2.2 Procedures of Kernel PCA

2.2.1 Kernel Matrix Method

Here we give the general procedures of Kernel PCA implementation.

Algorithm 8. The procedure to do Kernel PCA for $\mathbb{Y}$ is:

(a) Pick a kernel function $k(\mathbf{y}_i, \mathbf{y}_j) : \mathbb{R}^q \times \mathbb{R}^q \mapsto \mathbb{R}$.

(b) Calculate the Gram matrix of the kernel data

$$K = \begin{pmatrix} k(\mathbf{y}_1, \mathbf{y}_2) & k(\mathbf{y}_1, \mathbf{y}_3) & \cdots & k(\mathbf{y}_1, \mathbf{y}_N) \\ k(\mathbf{y}_2, \mathbf{y}_2) & k(\mathbf{y}_2, \mathbf{y}_3) & \cdots & k(\mathbf{y}_2, \mathbf{y}_N) \\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{y}_N, \mathbf{y}_2) & k(\mathbf{y}_N, \mathbf{y}_3) & \cdots & k(\mathbf{y}_N, \mathbf{y}_N) \end{pmatrix}.$$

(c) Center the kernel matrix

$$\tilde{K} = \left(I - \frac{1}{N} \mathbf{1}\mathbf{1}^\top \right) K \left(I - \frac{1}{N} \mathbf{1}\mathbf{1}^\top \right).$$

(d) Implement the classical MDS to the centered kernel matrix $\tilde{K}$, in essence, spectral decompose

$$\tilde{K}\mathbf{u}_i = \lambda \mathbf{u}_i \quad \iff \quad \tilde{K} = Q\Lambda Q^\top.$$

(e) For any new data $\mathbf{y}^*$, we can project it to each new dimension j

$$\mathbf{x}_j^* = \sum_{i=1}^{N} k(\mathbf{y}_i, \mathbf{y}^*) \mathbf{u}_{ij}.$$

2.2.2 Dissimilarity Matrix Method

Since the inner product in $\mathscr{H}$ uses the view of similarity measure, as discussed in the classical MDS, the practical case is sometimes different. Specifically, we are given the dissimilarity matrix, or distance matrix. If the distance is defined as the Euclidean distance, i.e.

$$d_{ij}^2 = ||\mathbf{y}_i - \mathbf{y}_j||^2,$$

we can transform the distance matrix $D^2 = (d_{ij})^2$ into kernel gram matrix K by

$$K = -\frac{1}{2}\left(I - \frac{1}{N}\mathbf{1}\mathbf{1}^\top\right) D^2 \left(I - \frac{1}{N}\mathbf{1}\mathbf{1}^\top\right).$$

And more generally, if the distance is other type rather than the Euclidean one, we can also apply this formula (but the kernel will be unknown).

2.3 Mathematical View of Kernel PCA

Theorem 6 says if we choose a kernel k to do kernel PCA, we are actually doing the classical PCA on the new feature data $\phi(\mathbf{y}_i)$'s. But recall that $\phi(\mathbf{y})$ is a function! What is the PCA of functions? In Example 6, we see that the polynomial kernel with order 2 has the feature map:

$$\phi(\mathbf{y}) := k_{\mathbf{y}}(\cdot) = \begin{pmatrix} 1 & \sqrt{2}y_1 & \sqrt{2}y_2 & y_1^2 & y_2^2 & \sqrt{y_1 y_2} \end{pmatrix}^\top.$$

This is a constant function when $\mathbf{y}$ is specified, so it is understandable. What if this function $k_{\mathbf{y}}(\cdot)$ is NOT constant?

To avoid too much involved in functional analysis, we assume conditions

- feature data are centered, i.e. $\sum_{i=1}^N \phi(\mathbf{y}_i) = 0$;
- the RKHS $\mathscr{H}$ is M-dim (in the true case, it should be ∞) and $M > q$.

2.3.1 Solution of Feature Data PCA

Now suppose the kernel k is pre-specified. Though we don't know what the feature functions $\phi(\mathbf{y}_i)$'s are, we know it exists and we denote the sample covariance matrix of features as

$$C_{M\times M} = \frac{1}{N}\sum_{i=1}^N \phi(\mathbf{y}_i)\phi(\mathbf{y}_i)^\top$$

with spectral decomposition is

$$j = 1, \ldots, M: \quad C\mathbf{v}_j = \lambda_j \mathbf{v}_j.$$

The key idea is that though C is unknown, the kernel Gram matrix K is known. If we somehow relate λ's and $\mathbf{v}$'s to the known matrix K, then we can solve the PCA of C without working directly in $\mathscr{H}$. Recall

$$k(\mathbf{y}_i, \mathbf{y}_j) = \langle \phi(\mathbf{y}_i), \phi(\mathbf{y}_j) \rangle,$$

and this $K = (k)_{ij}$ matrix is known. And for the matrix C, by its spectral decomposition, we know

$$\left[\frac{1}{N}\sum_{i=1}^{N}\phi(\mathbf{y}_i)\phi(\mathbf{y}_i)^\top\right]\mathbf{v}_j = C\mathbf{v}_j = \lambda_j\mathbf{v}_j,$$

where the LHS could be simplified as

$$\left[\frac{1}{N}\sum_{i=1}^{N}\phi(\mathbf{y}_i)\phi(\mathbf{y}_i)^\top\right]\mathbf{v}_j = \left[\frac{1}{N}\sum_{i=1}^{N}\phi(\mathbf{y}_i)\left\langle\phi(\mathbf{y}_i),\mathbf{v}_j\right\rangle\right] = \lambda_j\mathbf{v}_j.$$

Note that $\langle\phi(\mathbf{y}_i),\mathbf{v}_j\rangle$ is a scalar, so it means $\mathbf{v}_j$ is a linear combination of $\phi(\mathbf{y}_i)$'s, write

$$\mathbf{v}_j = \sum_{i=1}^{N} a_{ji}\phi(\mathbf{y}_i) = \Phi^\top\mathbf{a}_j, \quad \text{where} \quad \Phi = \begin{pmatrix}\phi(\mathbf{y}_1)^\top\\ \vdots \\ \phi(\mathbf{y}_N)^\top\end{pmatrix} \text{ and } \mathbf{a}_j = \begin{pmatrix}a_{j1}\\ \vdots \\ a_{jN}\end{pmatrix}. \tag{1}$$

Using this notation, the matrix C could be rewritten as

$$C = \frac{1}{N}\Phi^\top\Phi \quad\Rightarrow\quad C\mathbf{v}_j = \frac{1}{N}\Phi^\top\Phi\mathbf{v}_j = \lambda_j\mathbf{v}_j \quad\Rightarrow\quad \frac{1}{N}\Phi\Phi^\top\Phi\mathbf{v}_j = \lambda_j\Phi\mathbf{v}_j. \tag{2}$$

Note that $\Phi\Phi^\top$ is the kernel Gram matrix:

$$\left(\Phi\Phi^\top\right)_{ij} = \langle\phi(\mathbf{y}_i),\phi(\mathbf{y}_j)\rangle = (K)_{ij},$$

and by construction, $\mathbf{v}_j = \Phi^\top\mathbf{a}_j$. So, plugging these two expressions in to (2), it follows that

$$\frac{1}{N}K\Phi\Phi^\top\mathbf{a}_j = \lambda_j\Phi\Phi^\top\mathbf{a}_j \quad\Rightarrow\quad \frac{1}{N}K^2\mathbf{a}_j = \lambda_j K\mathbf{a}_j \quad\Rightarrow\quad \frac{1}{N}K\mathbf{a}_j = \lambda_j\mathbf{a}_j \text{ (assume } K \text{ is invertible)}$$

i.e. the constructed $\{\mathbf{a}_j : j = 1,\ldots,N\}$ and $\{\lambda_j : j = 1,\ldots,N\}$ are the eigens of K/N, which is solvable! Now, once we solve $\mathbf{v}_j$'s, we are done (goal is to decompose C, i.e. finding $\mathbf{v}_j$ and λ_j). However, this is not feasible. What we could do is only standardize $\mathbf{v}_j$. Recall the assumption that

$$j = 1,\ldots,M: \quad C\mathbf{v}_j = \lambda_j\mathbf{v}_j.$$

We only assume directions for $\mathbf{v}_j$, but not require it is unit vector. So, $\mathbf{v}_j$'s are not orthogonal basis now. But by construction

$$\mathbf{v}_j = \sum_{i=1}^{N} a_{ji}\phi(\mathbf{y}_i) = \Phi^\top\mathbf{a}_j.$$

Therefore it follows that

$$\begin{aligned} ||\mathbf{v}_j||^2 &= \mathbf{a}_j^\top\Phi\Phi^\top\mathbf{a}_j = \mathbf{a}_j^\top K\mathbf{a}_j \\ \text{(by construction (1) of } \mathbf{a}) \quad &= N\lambda_j\mathbf{a}_j^\top\mathbf{a}_j \\ (\mathbf{a} \text{ is orthogonal basis}) \quad &= N\lambda_j. \end{aligned}$$

So, the orthogonal basis for C should be

$$U_{M\times N} = \begin{bmatrix} \frac{\mathbf{v}_1}{\sqrt{N\lambda_1}} & \frac{\mathbf{v}_2}{\sqrt{N\lambda_2}} & \cdots & \frac{\mathbf{v}_N}{\sqrt{N\lambda_N}} \end{bmatrix} = \begin{bmatrix} \frac{\Phi^\top \mathbf{a}_1}{\sqrt{N\lambda_1}} & \frac{\Phi^\top \mathbf{a}_2}{\sqrt{N\lambda_2}} & \cdots & \frac{\Phi^\top \mathbf{a}_N}{\sqrt{N\lambda_N}} \end{bmatrix}, \tag{3}$$

where $\mathbf{a}_j$ is computable by K, but $\mathbf{v}_j$ remains unknown since Φ is not.

2.3.2 Computation of Feature Projection

One use of the decomposition of C is to compute the feature projection for any new or old observation. Suppose we now have the training data $\mathbf{y}_1, \dots, \mathbf{y}_N$ and a new observation $\mathbf{y} \in \mathbb{R}^q$. And we want to compute the feature projection of the new observation $\mathbf{y}$.

Since now we project $\mathbf{y}$ into the feature space $\phi(\mathbf{y})$, and the basis of that space is (2), the kernel PC scores (feature projection) of $\mathbf{y}$ is

$$\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_N \end{pmatrix} = U^\top \phi(\mathbf{y}).$$

If we consider the first kernel PC component, it is

$$\begin{aligned} x_1 &= \frac{1}{\sqrt{N\lambda_1}} \mathbf{v}_1^\top \phi(\mathbf{y}) \\ \text{(by construction (1) of } \mathbf{a}\text{)} \quad &= \frac{1}{\sqrt{N\lambda_1}} \mathbf{a}_1^\top \Phi \phi(\mathbf{y}) \\ &= \frac{1}{\sqrt{N\lambda_1}} \mathbf{a}_1^\top \begin{pmatrix} \phi(\mathbf{y}_1)^\top \\ \vdots \\ \phi(\mathbf{y}_N)^\top \end{pmatrix} \phi(\mathbf{y}), \quad \text{where } \mathbf{y}_1, \dots, \mathbf{y}_N \text{ are training data} \\ &= \frac{1}{\sqrt{N\lambda_1}} \mathbf{a}_1^\top \begin{pmatrix} k(\mathbf{y}_1, \mathbf{y}) \\ \vdots \\ k(\mathbf{y}_N, \mathbf{y}) \end{pmatrix}. \end{aligned}$$

Therefore, all kernel PC scores of the new observation $\mathbf{y}$ are

$$\mathbf{x}^\top = \begin{bmatrix} k(\mathbf{y}_1, \mathbf{y}) & \cdots & k(\mathbf{y}_N, \mathbf{y}) \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{N\lambda_1}} \mathbf{a}_1 & \cdots & \frac{1}{\sqrt{N\lambda_N}} \mathbf{a}_N \end{bmatrix}.$$

As for the kernel PC scores for the training data, we use the notation $\mathbb{Y}$ to denote the design matrix of training data, and put the first p kernel PC scores of $\mathbf{y}$ into a matrix:

$$\mathbb{X}_{N\times p} = \begin{bmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_N^\top \end{bmatrix} = \begin{bmatrix} k(\mathbf{y}_1, \mathbf{y}_1) & \cdots & k(\mathbf{y}_1, \mathbf{y}_N) \\ \vdots & \ddots & \vdots \\ k(\mathbf{y}_N, \mathbf{y}_1) & \cdots & k(\mathbf{y}_N, \mathbf{y}_1) \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{N\lambda_1}} \mathbf{a}_1 & \cdots & \frac{1}{\sqrt{N\lambda_N}} \mathbf{a}_N \end{bmatrix}.$$