

18

Kernel Principal Component Analysis

核主成分分析

用核技巧，将非线性数据投影到高维度空间，再投影



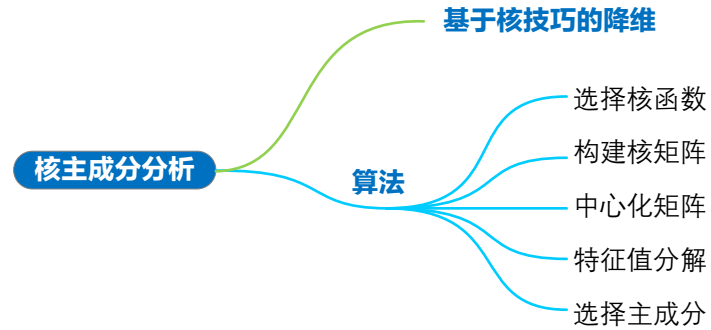
能够对一个想法抱有兴趣，而不全盘接受它，是受过教育的标志。

It is the mark of an educated mind to be able to entertain a thought without accepting it.

—— 亚里士多德 (Aristotle) | 古希腊哲学家 | 384 ~ 322 BC



- ▶ `numpy.argsort()` 返回数组中元素排序索引
- ▶ `numpy.linalg.eigh()` 计算实对称矩阵的特征值和特征向量的函数
- ▶ `sklearn.datasets.make_circles()` 生成一个具有圆形决策边界的二维二分类数据集
- ▶ `sklearn.decomposition.KernelPCA()` 核主成分分析工具
- ▶ `sklearn.metrics.pairwise.euclidean_distances()` 计算欧氏距离矩阵
- ▶ `sklearn.preprocessing.KernelCenterer()` 中心化核矩阵的函数
- ▶ `sklearn.preprocessing.StandardScaler()` 将数据进行标准化处理



本 PDF 文件为作者草稿，发布目的为方便大家在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

18.1 核 PCA

主成分分析不是万能的！

PCA 有个致命前提，PCA 假设数据服从多元高斯分布。如图 1 所示，对于这种非线性数据，PCA 在降维提取最大方差上几乎起不到任何作用。本章要介绍的**核主成分分析** (Kernel Principal Component Analysis, Kernel PCA, KPCA) 却可以帮助我们解决这类问题。

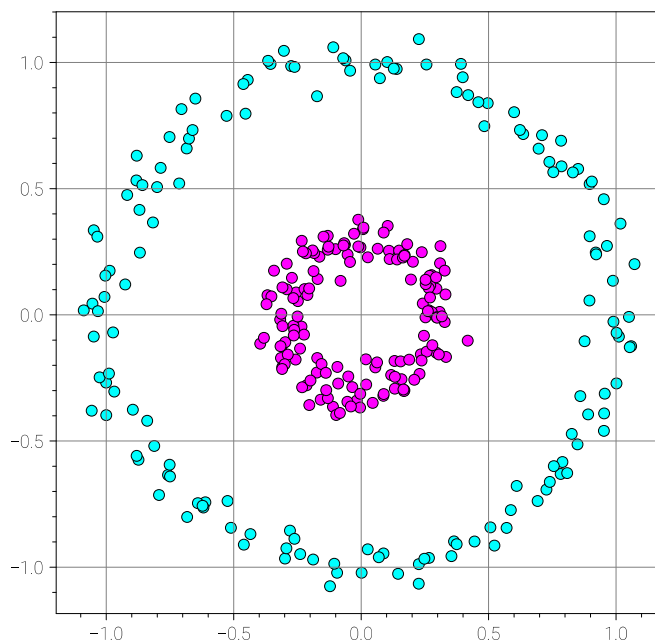


图 1. 主成分分析不能处理的非线性数据

核主成分分析 KPCA 是主成分分析的一种扩展，KPCA 允许处理非线性数据集。PCA 中，数据被投影到一个新的特征空间，以便在新的坐标系中最大化数据的方差。

然而，对于非线性数据，PCA 可能不够灵活。KPCA 使用核技巧来解决这个问题，KPCA 通过应用核函数来映射原始特征空间到一个更高维度的空间，使得数据在这个新空间中可以更好地被线性分离。通过前文学习，大家知道常用的核函数包括多项式核、高斯核、Sigmoid 核。

核 PCA 的步骤如下。

- ▶ 选择核函数：根据数据的性质选择适当的核函数，这取决于数据的非线性结构。
- ▶ 构建核矩阵：计算每对样本之间的核函数值，形成核矩阵。这个矩阵反映了样本在新特征空间中的相似性。
- ▶ 中心化核矩阵：对核矩阵进行中心化处理，确保数据的行列均值同时为零。
- ▶ 计算特征值和特征向量：对中心化后的核矩阵进行特征值分解，得到特征值和对应的特征向量。
- ▶ 选择主成分：选择前 p 个最大特征值对应的特征向量，构成新的特征矩阵。选取的特征向量就是因子得分，相当于投影结果。

本章下文用高斯核为例介绍 KPCA。

18.2 从 PCA 说起

读到这里，相信大家已经对 PCA 了如指掌；即便如此，为了方便展开讲解 KPCA，这一节还是简单回顾 PCA 原理。

第一个格拉姆矩阵分解

如图 2 所示，对矩阵 X (形状 $n \times D$) 的格拉姆矩阵 $X^T X$ (形状 $D \times D$) 进行特征值分解，我们便得到各个主成分对应的载荷 V 。上述运算对应协方差矩阵特征值分解如下。

$$X^T X = V \Lambda V^T \quad (1)$$

特别地，本章默认矩阵 X 已经标准化。标准化数据矩阵 X 的格拉姆矩阵相当于原始数据的相关性系数矩阵。此外，我们也不需要考虑 $(n-1)$ 对相关性系数矩阵的影响。

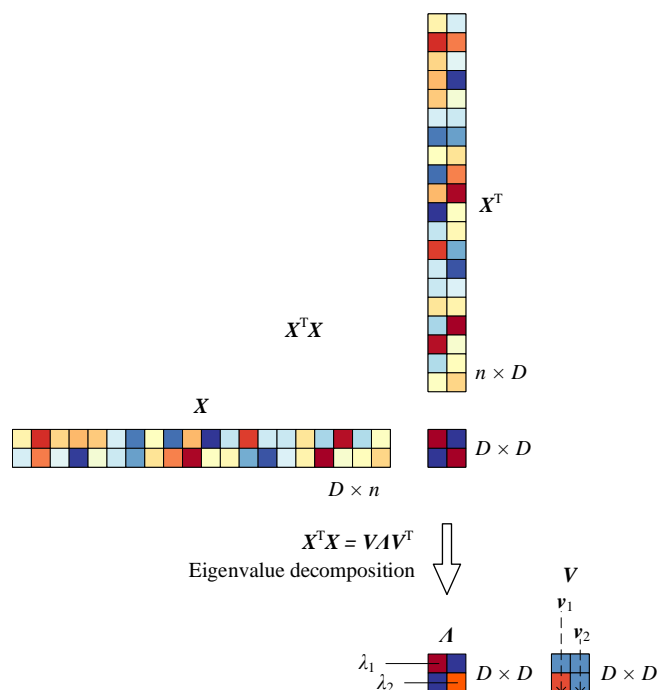


图 2. 第一个格拉姆矩阵的特征值分解

而因子得分 Z 可以通过投影获得。

$$Z = X V \quad (2)$$

根据奇异值分解 ($X = U S V^T$)，因子得分还可以写成。

$$Z = U S \quad (3)$$

上式展开来写。

$$\begin{bmatrix} z_1 & z_2 & \cdots & z_D \end{bmatrix} = \begin{bmatrix} u_1 & u_2 & \cdots & u_D \end{bmatrix} \underbrace{\begin{bmatrix} s_1 & & & \\ & s_2 & & \\ & & \ddots & \\ & & & s_D \end{bmatrix}}_S = \begin{bmatrix} s_1 u_1 & s_2 u_2 & \cdots & s_D u_D \end{bmatrix} \quad (4)$$

U 的每个列向量 \mathbf{u}_j 均为单位向量，即 $\|\mathbf{u}_j\|=1$ 。由于矩阵 S 为对角方阵（对角线元素为奇异值 s_j ）， S 中奇异值 s_j 仅仅对 U 的列向量提供缩放作用。投影结果列向量 \mathbf{z}_j 的模为 s_j ，即 $\|s_j \mathbf{u}_j\| = s_j = \sqrt{\lambda_j}$ 。

第二个格拉姆矩阵分解

而《矩阵力量》反复提过，矩阵 X 还有第二个格拉姆矩阵 XX^T （形状 $D \times D$ ），如图 3 所示。而 XX^T 也相当于线性核，其中每个元素为 $\mathbf{x}^{(i)} (\mathbf{x}^{(j)})^T$ 。

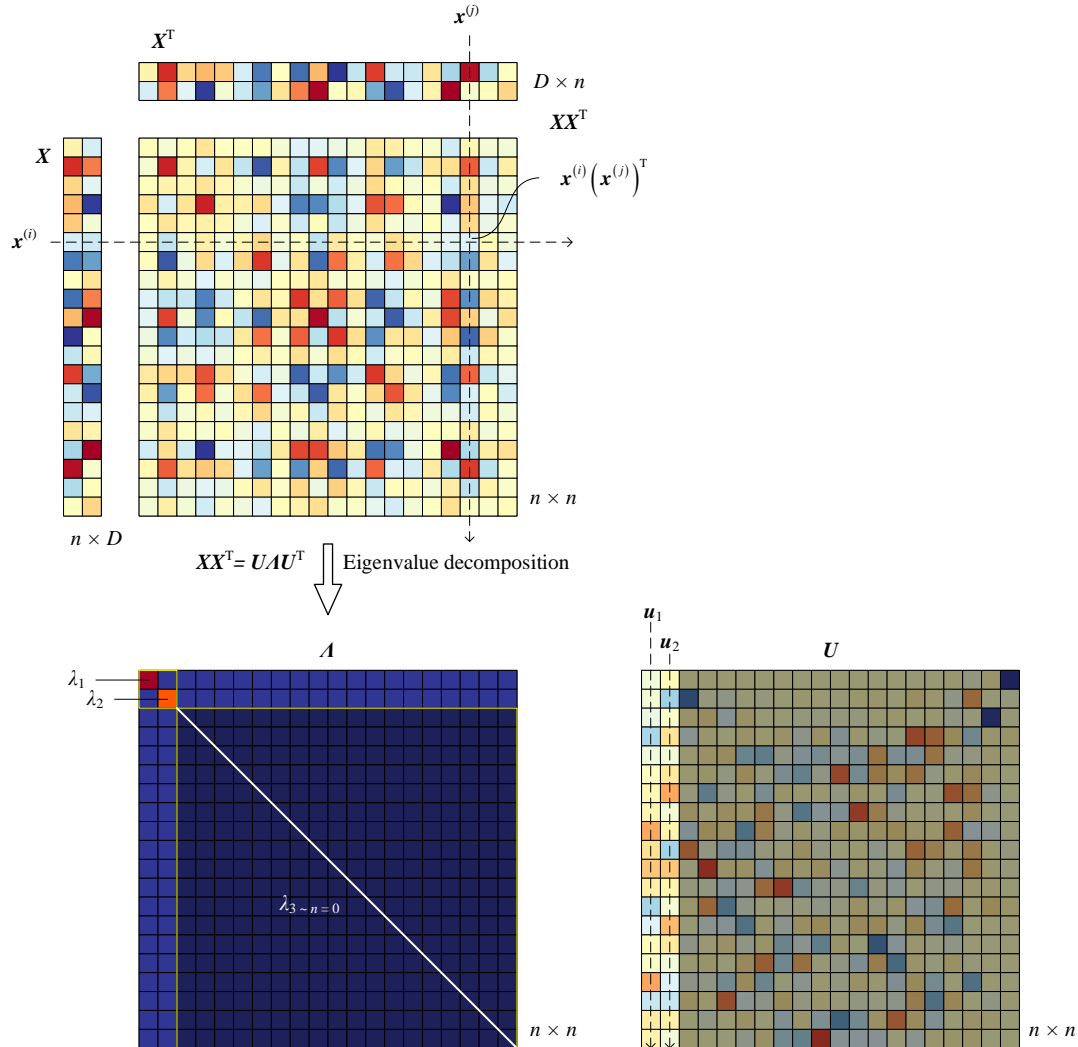


图 3. 第二个格拉姆矩阵的特征值分解

如图 3 所示，对格拉姆矩阵 XX^T 进行特征值分解，我们可以直接获得 \mathbf{u}_j 。

$$XX^T = UAU^T \quad (5)$$

注意，(1) 和 (5) 的特征值方阵形状不同，但是除 0 以外，两者拥有相同特征值。此外，请大家格外注意图 3 中 \mathbf{u}_j ，下一节讲解 KPCA 时我们会用到相同的思路。

如图 4 所示，联系上述两个格拉姆矩阵特征值分解正是奇异值分解。图 4 所示为经济型 SVD 分解，请大家绘制对应完全型 SVD 分解的矩阵运算图解。

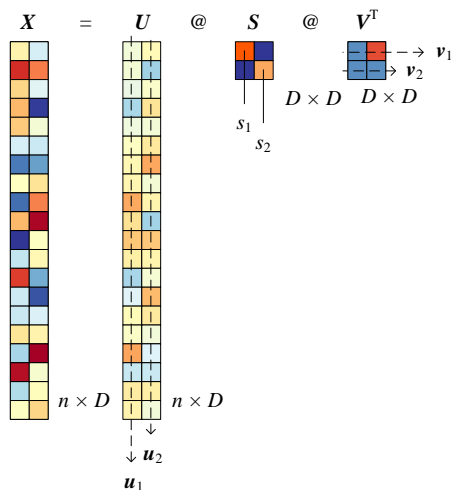


图 4. 奇异值分解联系两个格拉姆矩阵特征值分解

18.3 核主成分分析

本节以高斯核为例介绍如何利用核技巧完成 KPCA。

欧氏距离成对距离矩阵

大家已经知道要想计算高斯核矩阵，我们首先要计算欧氏成对距离矩阵。

如图 5 所示，对于给定的散点，我们先计算其成对欧氏距离矩阵。任意两点， $\mathbf{x}^{(i)}$ 和 $\mathbf{x}^{(j)}$ ，的欧氏距离，即 L^2 范数 $\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|_2$ 。

如图 6 所示为如何计算欧式距离平方，即 $\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|_2^2$ 。

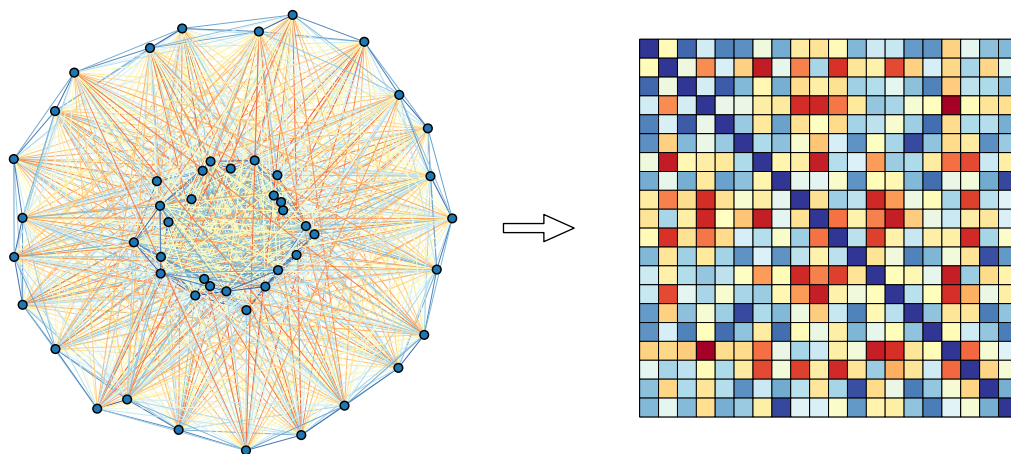


图 5. 成对欧氏距离

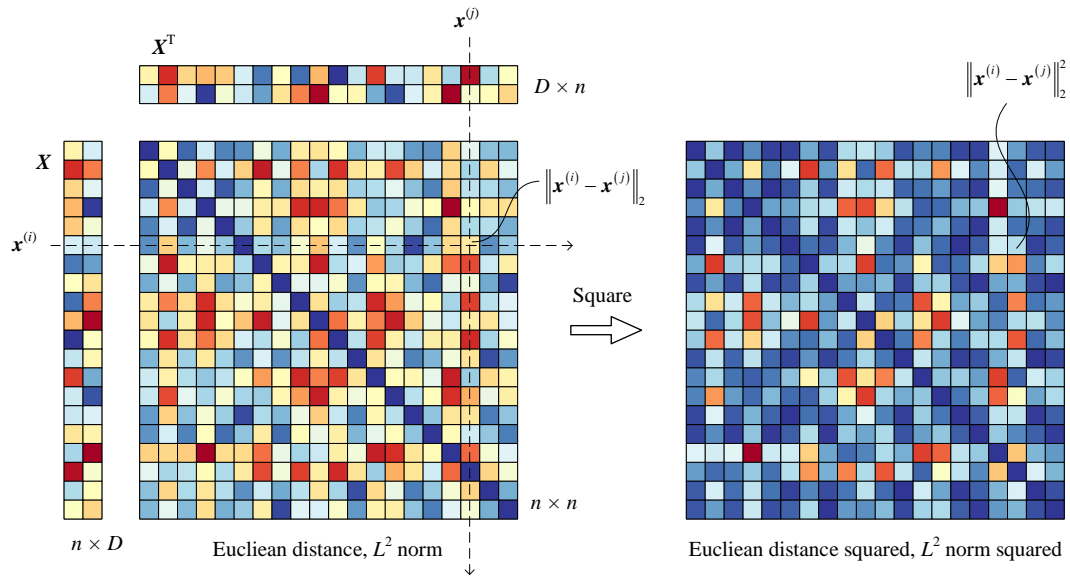


图 6. 欧氏距离平方

高斯核

如图 7 所示根据欧氏距离平方 $\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|_2^2$ ，我们可以计算高斯核 $\kappa(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp(-\gamma \|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|_2^2)$ 。通过高斯函数，我们把距离度量转化为“亲近度”。

高斯核中的 γ 是需要调整的模型参数。注意，不同算法中高斯核函数的形式可能稍有差别。

通过前文学习，我们知道核技巧是一种在机器学习中常用的技术，主要用于处理非线性问题。它的基本思想是通过一个称为核函数的函数，将输入的特征映射到高维空间。

核技巧的主要优势在于它避免了直接在高维空间中进行计算，而是通过核函数在低维空间中的计算得到高维空间中的内积。这样做的好处是可以节省计算成本，并更有效地处理复杂的非线性关系。

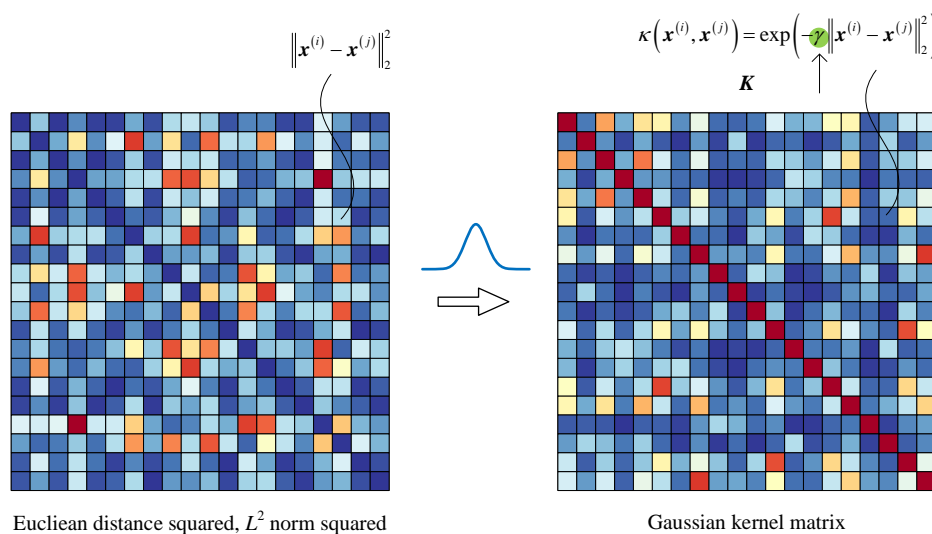


图 7. 高斯核矩阵

核中心化

下一步，高斯核矩阵 K 还需要经过行列中心化，获得 K_c 。 K_c 为**中心矩阵** (centering matrix)。中心矩阵的每一行、每一列的均值都是 0。



鸢尾花书读者对于中心化这个概念应该不陌生，《矩阵力量》第 22 章第 4 节。

图 8 中的矩阵 M 就是中心化矩阵，具体如下。

$$M = I - \frac{1}{n} \mathbf{1}\mathbf{1}^T \quad (6)$$

▲ 注意， K 为对称矩阵， M 也是对称矩阵。

上式中， $\frac{1}{n} \mathbf{1}\mathbf{1}^T$ 是一个 $n \times n$ 矩阵，每个元素值都是 $\frac{1}{n}$ 。

首先对高斯核矩阵 K 列中心化，即去均值。

$$K_{\text{col_demean}} = MK \quad (7)$$

然后再对上述矩阵行中心化，结果就是 K_c 。

$$K_c = \left(M \left(K_{\text{col_demean}} \right)^T \right)^T = \left(M (MK)^T \right)^T = MKM^T \quad (8)$$

K_c 也是对称矩阵。

将 (6) 代入 (8) 展开可以得到。

$$\begin{aligned} K_c &= \left(I - \frac{1}{n} \mathbf{1}\mathbf{1}^T \right) K \left(I - \frac{1}{n} \mathbf{1}\mathbf{1}^T \right)^T \\ &= \left(I - \frac{1}{n} \mathbf{1}\mathbf{1}^T \right) K \left(I - \frac{1}{n} \mathbf{1}\mathbf{1}^T \right) \\ &= K - \frac{1}{n} \mathbf{1}\mathbf{1}^T K - K \frac{1}{n} \mathbf{1}\mathbf{1}^T + \frac{1}{n} \mathbf{1}\mathbf{1}^T K \frac{1}{n} \mathbf{1}\mathbf{1}^T \end{aligned} \quad (9)$$

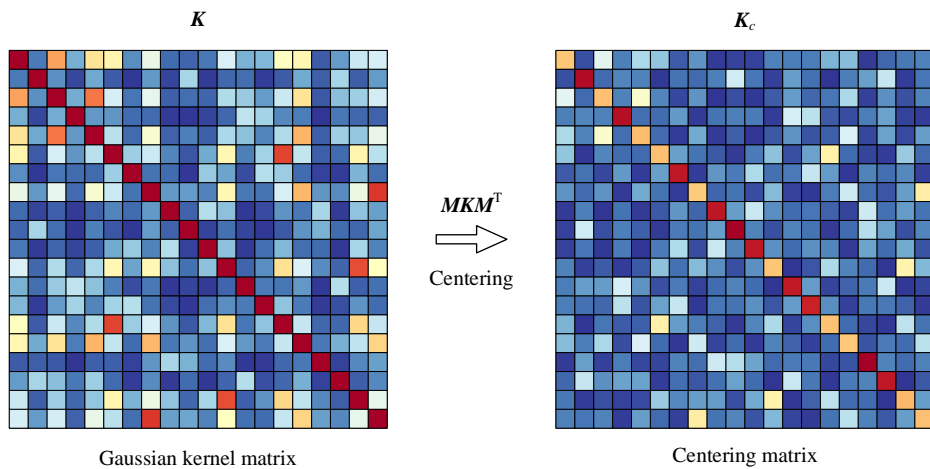


图 8. 高斯核中心化

高斯核的特征值分解

如图 9，下一步就是对 K_c 特征值分解。将特征值从大到小排列后，取出排名靠前的特征向量，这就是经过“非线性投影”得到的因子得分。

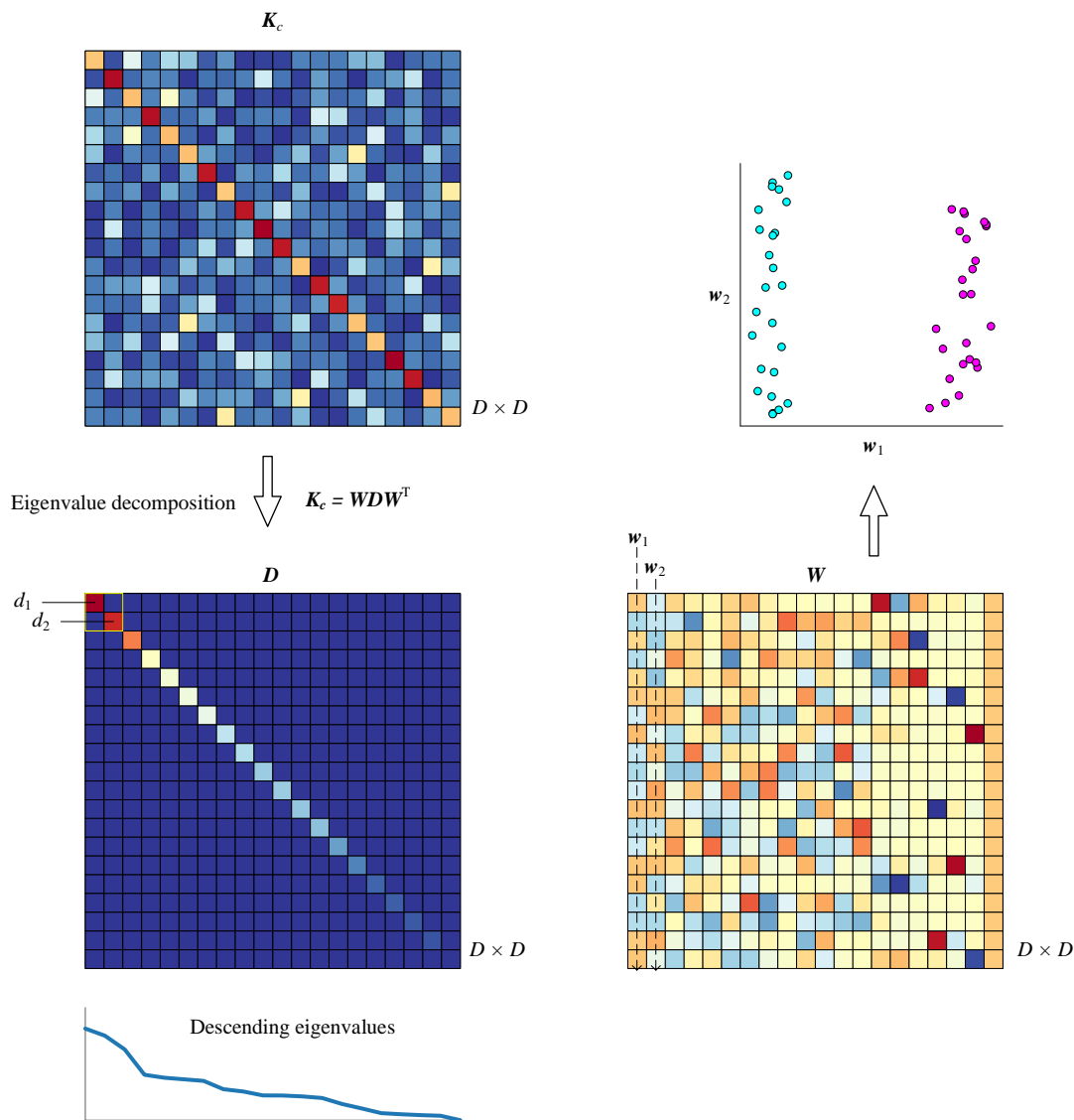


图 9. 高斯核的特征值分解

Bk7_Ch18_01.ipynb 一步步实现上述核 PCA 运算，下面让我们聊聊其中关键语句。

- a 利用 `sklearn.datasets.make_circles()` 生成环形数据集。
- b 利用 `sklearn.preprocessing.StandardScaler()` 中 `fit_transform()` 方法对数据进行标准化。
- c 用 `sklearn.metrics.pairwise.euclidean_distances()` 计算成对欧氏距离平方。
- d 将上述成对欧氏距离平方矩阵转化为高斯核矩阵。

e 利用 `sklearn.preprocessing.KernelCenterer()` 中 `fit_transform()` 对高斯核矩阵中心化。当然，这一句被注释掉，请大家自行和下文代码比较运算结果。

f 计算中心化矩阵 M 。

g 对高斯核矩阵中心化。

h 利用 `numpy.linalg.eigh()`：对中心化后的核矩阵进行特征值分解，得到特征值 `eig_vals` 和特征向量 `eig_vecs`。注意，`numpy.linalg.eigh()` 专门用于对称/Hermitian 矩阵的特征值和特征向量的计算。对于 Hermitian 矩阵，特征值是实数，而且特征向量是正交的。

`numpy.linalg.eig()` 则适用于一般的矩阵，不要求输入矩阵是对称的，结果特征值可以是复数。

i 利用 `numpy.argsort()` 获取特征值从大到小排序的索引。

j 取出核主成分分析前两个主成分。

```
from sklearn.datasets import make_circles

# 生成数据
a X_original, y = make_circles(n_samples=200,
                               factor=0.3,
                               noise=0.05,
                               random_state=0)

# 标准化
b from sklearn.preprocessing import StandardScaler
X = StandardScaler().fit_transform(X_original)

# 计算欧氏距离（平方）矩阵
c from sklearn.metrics.pairwise import euclidean_distances
dist = euclidean_distances(X, X, squared=True)

# 计算核函数矩阵，高斯核
gamma = 1 # 模型参数需要优化
d K = np.exp(-gamma * dist)

# 中心化
e from sklearn.preprocessing import KernelCenterer
# Kc = KernelCenterer().fit_transform(K)
# 比较结果

n = len(K)
f M = (np.identity(n) - 1/n*np.ones((n,n)))
g Kc = M @ K @ M.T

# 特征值分解
h eig_vals, eig_vecs = np.linalg.eigh(Kc)

# 按特征值大小排序
i idx = np.argsort(eig_vals)[::-1]
eig_vals = eig_vals[idx]
eig_vecs = eig_vecs[:,idx]

# 取出前两个主成分
num_PCs = 2
j Xpca = eig_vecs[:, :2]
```

代码 1. 逐步完成核 PCA |  Bk7_Ch18_01.ipynb

本 PDF 文件为作者草稿，发布目的为方便大家在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微视频均发布在 B 站——生姜 DrGinger: <https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

升维过程

图 10 展示的就是非线性投影产生的网格变化。虽然本例中，原始特征只有两个，但是经过非线性变换我们可以得到各种奇形怪状的非线性投影网格。

这个过程就是通过核函数达到的“升维”的效果。

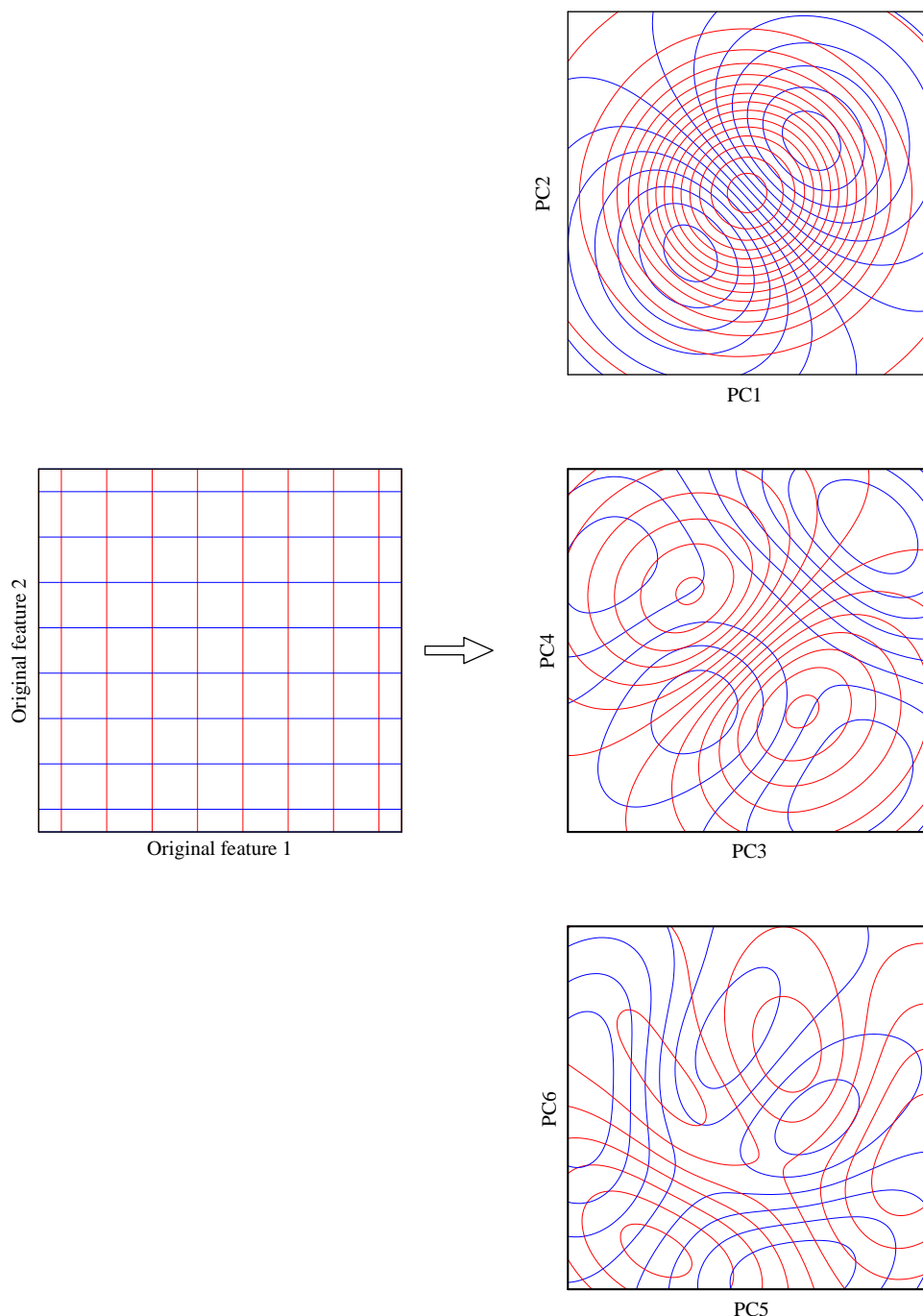


图 10. 非线性映射网格变化

本 PDF 文件为作者草稿，发布目的为方便大家在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger: <https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

举例

在 Scikit-learn 中有 `sklearn.decomposition.KernelPCA()` 函数专门完成核 PCA。图 11 所示为利用这个函数中的高斯核函数完成的环形数据的核 PCA 分析，下面聊聊 Bk7_Ch18_01.ipynb 这部分代码。

- a** 用 `sklearn.decomposition.KernelPCA()` 完成核 PCA。
`n_components=2` 指定要保留的主成分数量为 2。`kernel='rbf'` 选择使用径向基函数核（RBF kernel），并设置核函数的参数 `gamma` 为 1。请大家翻阅技术文档，尝试使用其他核函数，并比较结果。
- b** 对输入数据 `X` 进行核主成分分析，将结果存储在 `SK_PC_X` 中。这一步将数据映射到新的主成分空间。
- c** 使用散点图可视化映射后的主成分空间。`X` 轴使用第一个主成分，`Y` 轴使用第二个主成分。点的颜色由标签 `y` 决定，使用 `'cool'` 颜色映射，边缘颜色为黑色，透明度为 `0.5`。

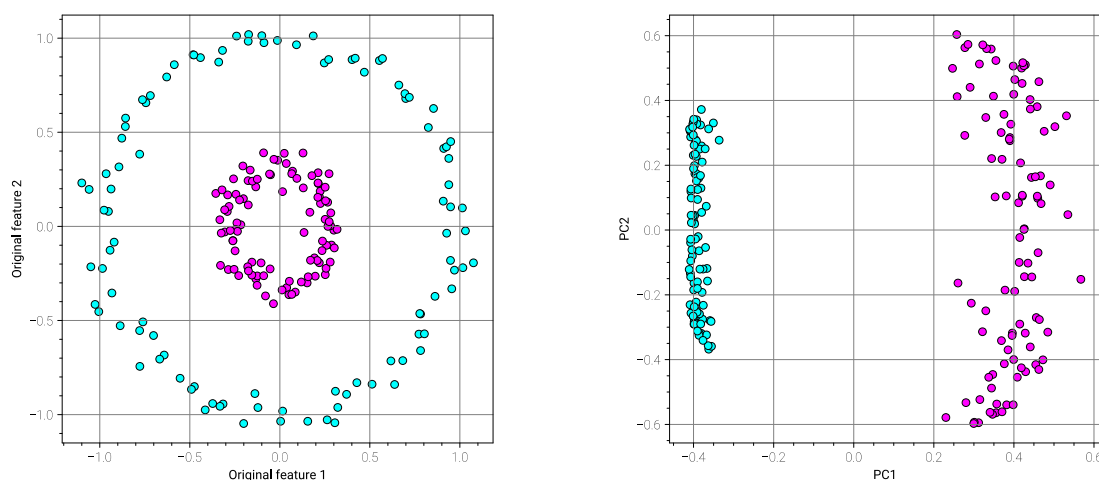


图 11. 核 PCA 示例 | [Bk7_Ch18_01.ipynb](#)

```
from sklearn.decomposition import KernelPCA

# 调用核PCA工具
a SK_PCA = KernelPCA(n_components=2, kernel='rbf', gamma=1)

# 对输入数据X进行核主成分分析
b SK_PC_X = SK_PCA.fit_transform(X)

# 可视化
fig, ax = plt.subplots(figsize = (6,6))

c ax.scatter(SK_PC_X[:, 0], SK_PC_X[:, 1],
             c=y, cmap = 'cool',
             edgecolors = ['k'], alpha = 0.5)
ax.set_xlabel("PC1")
ax.set_ylabel("PC2")
```

代码 2. 用 `sklearn.decomposition.KernelPCA()` 完成核主成分分析 | [Bk7_Ch18_01.ipynb](#)

本 PDF 文件为作者草稿，发布目的为方便大家在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger: <https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

图 12 和图 13 所示为利用 Streamlit 搭建的两个 App，展示高斯核 KPCA 中参数 Gamma 对结果影响。

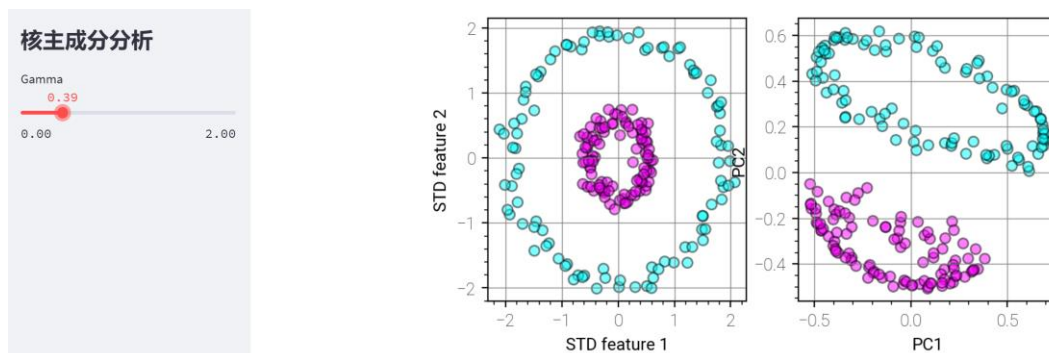


图 12. 展示 Gamma 对核主成分分析结果影响的 App，环形数据，Streamlit 搭建 | [Streamlit_Bk7_Ch18_02.py](#)

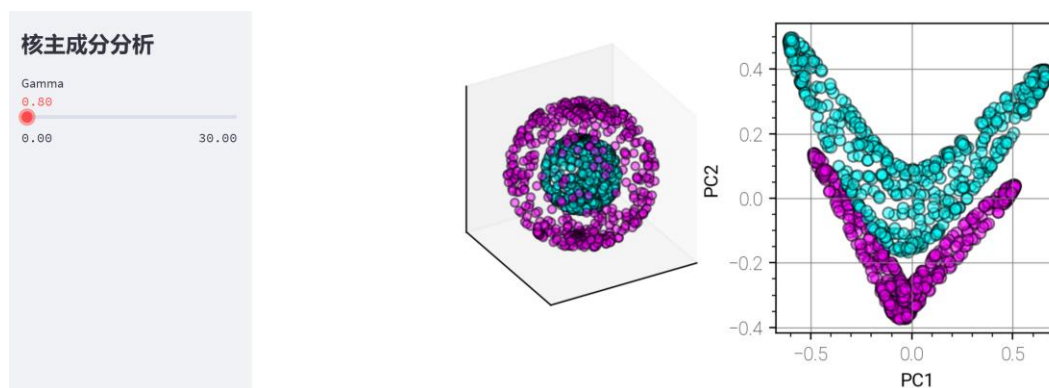


图 13. 展示 Gamma 对核主成分分析结果影响的 App，球形数据，Streamlit 搭建 | [Streamlit_Bk7_Ch18_03.py](#)

核主成分分析通常用于非线性降维，它通过将数据映射到高维特征空间，然后在该空间中执行主成分分析。虽然它本身不是分类或聚类算法，但在降维后，可以使用其他算法进行分类或聚类分析。

有关核主成分分析背后的数学原理，请大家参考如下文章。

<https://arxiv.org/pdf/1207.3538.pdf>