哈尔滨工业大学计算机科学与技术学院

实验报告

课程名称：机器学习

课程类型：必修

实验题目：PCA模型实验

学号：1190200501

姓名：林燕燕

一、实验目的

实现一个PCA模型，能够对给定数据进行降维（即找到其中的主成分）

二、实验要求及实验环境

实验要求

测试：

1. 首先人工生成一些数据（如三维数据），让它们主要分布在低维空间中，如首先让某个维度的方差远小于其它唯独，然后对这些数据旋转。生成这些数据后，用你的PCA方法进行主成分提取。
2. 找一个人脸数据（小点样本量），用你实现PCA方法对该数据降维，找出一些主成分，然后用这些主成分对每一副人脸图像进行重建，比较一些它们与原图像有多大差别（用信噪比衡量）。

实验环境

OS： Windows 11 Python： 3.7.9

三、设计思想

PCA(主成分分析，Principal Component Analysis)是最常用的一种降维方法。PCA 的主要思想是将 D 维特征通过一组投影向量映射到 K 维上，这 K 维是全新的正交特征，称之为主成分，采用主成分作为数据的代表，有效地降低了数据维度，且保留了最多的信息。关于 PCA 的推导有两种方式：最大投影方差和最小投影距离。

- 最大投影方差：样本点在这个超平面上的投影尽可能分开
- 最小投影距离：样本点到这个超平面的距离都足够近

在开始 PCA 之前需要对数据进行预处理，即对数据中心化。设数据集 $X=\{x_1,x_2,\dots,x_n\}$，其中 $x_i=\{x_{i1},x_{i2},\dots,x_{id}\}$，即 X 是一个 $n\times d$ 的矩阵。则此数据集的中心向量（均值向量）为：

$$\mu=\frac{1}{n}\sum_{i=1}^{n}x_i$$

对数据集每个样本均进行操作：$x_i=x_i-\mu$，就得到了中心化后的数据，此时有 $\sum_{i=1}^{n}x_i=0$。

中心化可以给后面的计算带来极大的便利，因为中心化之后的常规线性变换就是绕原点的旋转变化，也就是坐标变换。此时，协方差为 $S=\frac{1}{n}\sum_{i=1}^{n}x_ix_i^T=\frac{1}{n}X^TX$

设使用的投影坐标系的一组**标准正交基**为
$U_{k\times d}=\{u_1,u_2,\dots,u_k\},\ k<d,u_i=\{u_{i1},u_{i2},\dots,u_{id}\}$，故有 $UU^T=1$，使用这组基变换中心化矩阵 X，得降维压缩后的矩阵 $Y_{n\times k}=XU^T$，重建得到 $\hat{X}=YU=XU^TU$。

对于任意一个样本 x_i，在新的坐标系中的投影为 $y_i=x_iU^T$，在新坐标系中的投影方差为 $y_i^Ty_i=Ux_i^Tx_iU^T$。要使所有的样本的投影方差和最大，也就是求 $\arg\max_U\sum_{i=1}^{n}Ux_i^Tx_iU^T$，即

$$\arg\max_U\ tr(UX^TXU^T)\qquad s.t.\ UU^T=1$$

求解：在 u_1 方向投影后的方差

$$\frac{1}{n}\sum_{i=1}^{n}\{u_1^T x_i - u_1^T \mu\}^2 = \frac{1}{n}(Xu_1^T)^T(Xu_1^T) = \frac{1}{n}u_1 X^T X u_1^T = u_1 S u_1^T$$

因为 u_1 是投影方向，且已经假设它是单位向量，即 $u_1^T u_1 = 1$，用拉格朗日乘子法最大化目标函数：

$$L(u_1) = u_1^T S u_1 + \lambda_1(1 - u_1^T u_1)$$

对 u_1 求导，令导数等于 0，解得 $Su_1 = \lambda_1 u_1$，显然，u_1 和 λ_1 是一组对应的 S 的特征向量和特征值，所以有 $u_1^T S u_1 = \lambda_1$，结合在 u_1 方向投影后的方差式，可得求得最大化方差，等价于求最大的特征值。

要将 d 维的数据降维到 k 维，只需计算前 k 个最大的特征值，将其对应的特征向量（$d \times 1$ 的）转为行向量（$1 \times d$ 的）组合成特征向量矩阵 $U_{k\times d}$，则降维压缩后的矩阵为 $Y = XU^T$ 。

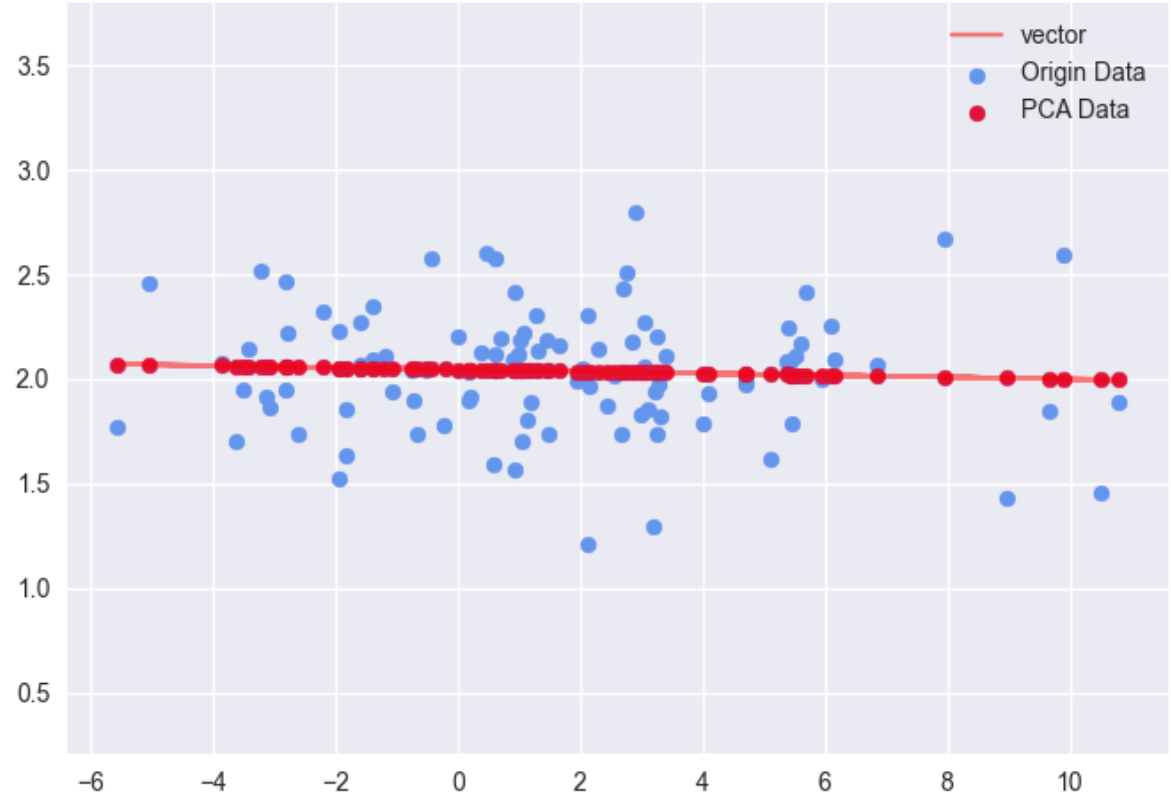
四、实验结果分析

1、二维降到一维

生成高斯分布数据的参数：

$$\mu = [2, 2],\ \sigma = \begin{bmatrix} 10 & 0 \\ 0 & 0.1 \end{bmatrix}$$

降维结果如下：

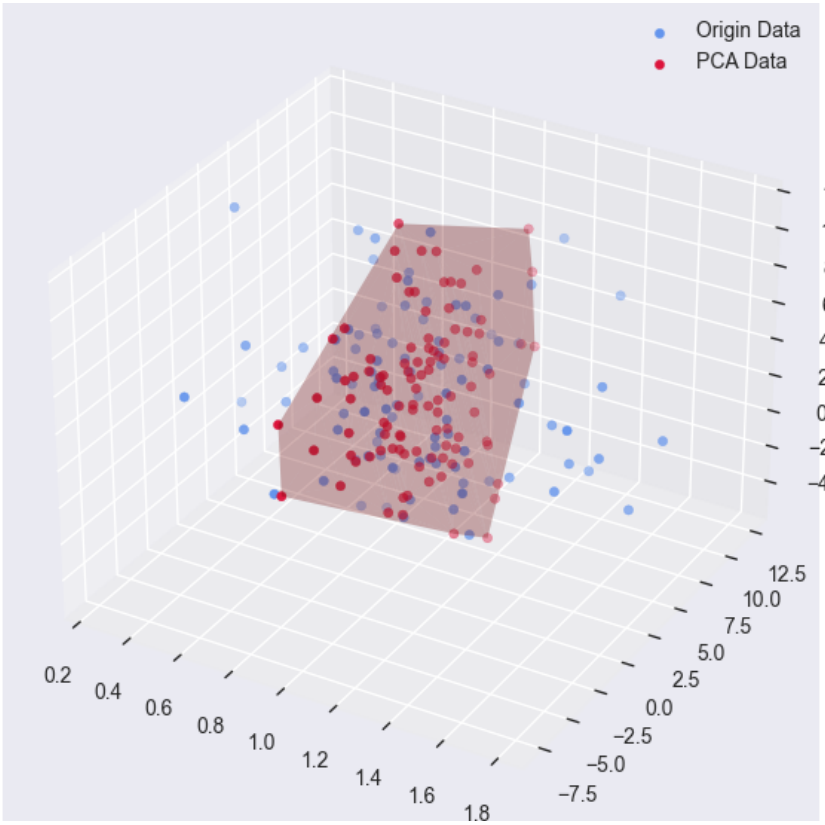


2、三维降到二维

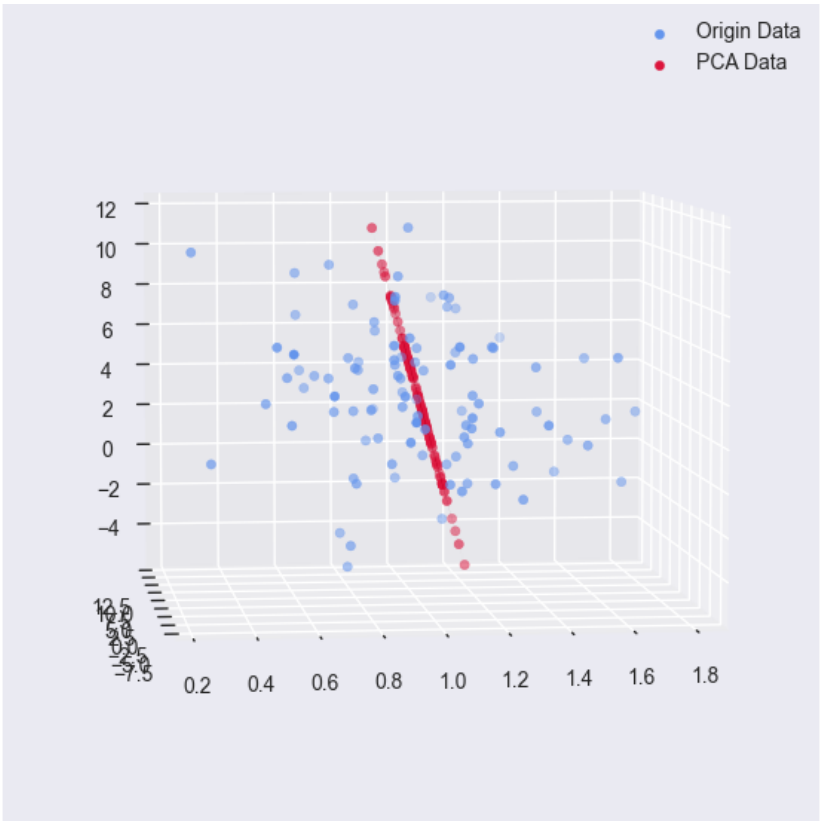
生成高斯分布数据的参数：

$$\mu = [1, 2, 3],\ \sigma = \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 10 \end{bmatrix}$$

降维结果如下：



可以看到，各点降维到一个平面：



3、人脸数据测试

$$MSE = \frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}||I(i,j) - K(i,j)||^2$$

信噪比计算公式：

$$PSNR = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) = 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right)$$

使用三张图片，降维后维度分别设置为$(50, 30, 20, 10, 5, 4, 2, 1)$，测试如下：

Original Picture

k = 50, PSNR = 29.95

k = 30, PSNR = 26.27

k = 20, PSNR = 24.16

k = 10, PSNR = 21.36

k = 5, PSNR = 19.16

k = 4, PSNR = 18.60

k = 2, PSNR = 16.88

k = 1, PSNR = 15.04

Original Picture

k = 50, PSNR = 26.55

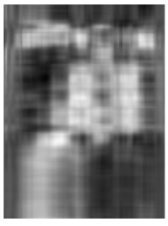
k = 30, PSNR = 25.19

k = 20, PSNR = 24.14

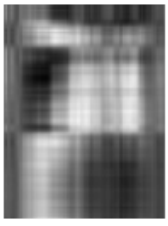
k = 10, PSNR = 22.37

k = 5, PSNR = 20.90

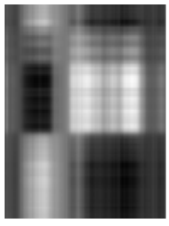
k = 4, PSNR = 20.38

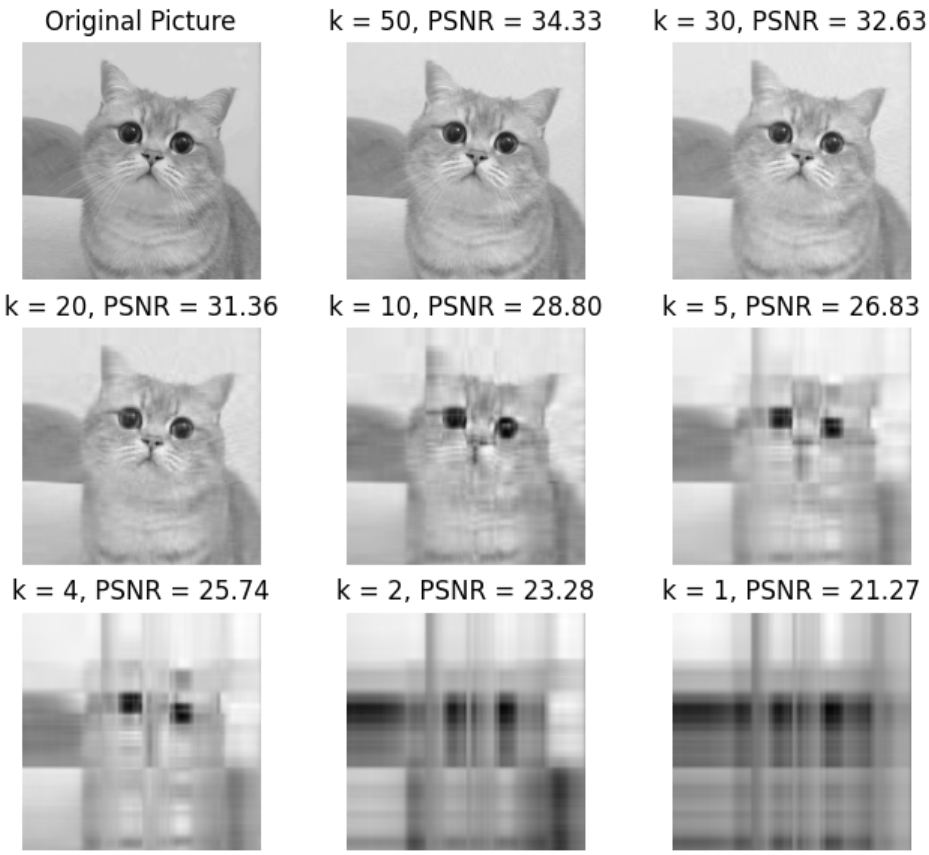


k = 2, PSNR = 19.09



k = 1, PSNR = 17.92





在维度等于20时，还能较好地保留图片特征，人眼几乎无法分辨损失；维度为5时，已经能够看出图片明显地损失；当目标维度降到2甚至1时，已经无法分辨出原来的图像。同时，信噪比随着目标维度的下降而下降。

五、结论

PCA 用于图片的降维可以极大地缓解存储压力，尤其是在如今像素越来越高的情况下，但在维度的选择上需要合适，太低会使图片无法辨认。使用 PCA 降维我们只需要存储三个比较小的矩阵，能够较大地节省存储空间。

六、参考文献

《机器学习》

主成分分析（PCA）原理详解 - 知乎

七、附录:源代码(带注释)

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import os
from PIL import Image

def pca(x, k):
    """
        pca降维
```

```
    """
    mu = np.mean(x, axis=0)
    cov = np.cov(x, rowvar=False)
    values, vectors = np.linalg.eig(cov) # 特征值 特征向量
    index = np.argsort(values)[: -(k + 1): -1]  # 取最大的 k 个的下标值
    vectors = vectors[:, index]  # 取对应下标的特征向量
    x_pca = (x - mu).dot(vectors).dot(vectors.T) + mu  # 重建数据
    return x_pca

def generate_data(mu, sigma, n=100):
    """
        生成高斯分布数据
    """
    x = np.random.multivariate_normal(mean=mu, cov=sigma, size=n)
    return x


def faces_pca(path, k_list):
    """
        path: 文件夹名称
        k_list: 不同维度列表
    """
    x_list = faces(path)
    for x in x_list:
        x_pca_list = []
        psnr_list = []
        for k in k_list:
            x_pca = pca(x, k)
            x_pca_list.append(x_pca)
            psnr_list.append(psnr(x, x_pca))
        show_faces(x, x_pca_list, k_list, psnr_list)


def faces(path):
    """
        读取指定目录下的所有文件
    """
    file_list = os.listdir(path)
    x_list = []
    for file in file_list:
        file_path = os.path.join(path, file)
        pic = Image.open(file_path).convert("L")  # 读入图片 转换为灰度图
        x_list.append(np.asarray(pic))
    return x_list



def show_faces(x, x_pca_list, k_list, psnr_list):
    """
        展示降维后的结果
    """
    plt.figure(figsize=(12, 8), frameon=False)
    # 原图
    plt.subplot(3, 3, 1)
    plt.title("Original Picture")
    plt.imshow(x, cmap='gray')
    plt.axis("off")  # 去掉坐标轴
    # 降维后的图
```

```
    for i in range(len(k_list)):
        plt.subplot(3, 3, i + 2)
        plt.title(
            "k = " + str(k_list[i]) + ", PSNR = " + "{:.2f}".format(psnr_list[i])
        )
        plt.imshow(x_pca_list[i], cmap='gray')
        plt.axis("off")
    plt.show()


def psnr(source, target):
    """
        计算峰值信噪比
    """
    rmse = np.sqrt(np.mean((source - target) ** 2))
    return 20 * np.log10(255.0 / rmse)



if __name__ == '__main__':
    """
        三维到二维
    """
    mean = [1, 2, 3]
    cov = [[0.1, 0, 0], [0, 10, 0], [0, 0, 10]]
    x = generate_data(mean, cov)  # 默认100个数据
    # print(x)
    x_pca = pca(x, 2)
    # print(x_pca)
    plt.style.use("seaborn")
    fig = plt.figure()
    ax = Axes3D(fig,auto_add_to_figure=False)
    fig.add_axes(ax)
    ax.scatter(x[:, 0], x[:, 1], x[:, 2], c="cornflowerblue", label="Origin Data")
    ax.scatter(x_pca[:, 0], x_pca[:, 1], x_pca[:, 2], c="crimson", label="PCA Data")
    ax.plot_trisurf(x_pca[:, 0], x_pca[:, 1], x_pca[:, 2], color="r", alpha=0.3)
    ax.legend(loc="best")
    plt.show()
    plt.style.use("default")

    """
        人脸数据
    """
    k_list = [50, 30, 20, 10, 5, 4, 2, 1]
    faces_pca('faces', k_list)
```