哈尔滨工业大学计算机科学与技术学院

机器学习实验报告

姓名	徐亚楠
学号	1190201224
实验题目	PCA
报告时间	2021.11.20

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

2.1 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）。

2.2 实验环境：Pycharm 2021 windows10 python3.8

三、设计思想（本程序中的用到的主要算法及数据结构）

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

3.1 中心化

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 $X = \{x_1, x_2, \ldots, x_n\}$，其中$x_i = \{x_{i1}, x_{i2}, \ldots, x_{id}\}$，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 $S = \frac{1}{n}\sum_{i=1}^{n} x_i\, x_i^T = \frac{1}{n}X^T X$

设使用的投影坐标系的一组**标准正交基**为

$Uk \times d = u_1, u_2, \ldots, u_k, k < d, u_i = ui1, ui2, \ldots, uid$ ，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 $\hat{X} = YU = XU^T U$。

3.2 最大投影方差

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

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

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^Tu_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 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。

$$Y = XU^T$$

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 $Y = XU^T$ 。

3.3 最小投影距离

现在考虑整个样本集，希望所有的样本到这个超平面的距离足够近，也就是得到 Y 后，与 X 的距离最小。即求：

$$\begin{aligned}\arg \min_U \sum_{i=1}^{n} ||\hat{x}_i - x_i||_2^2 \quad &= \arg \min_U \sum_{i=1}^{n} ||x_i U^T U - x_i||_2^2 \\ &= \arg \min_U \sum_{i=1}^{n} ((x_i U^T U)(x_i U^T U)^T - 2(x_i U^T U)x_i^T + x_i x_i^T) \\ &= \arg \min_U \sum_{i=1}^{n} (\, x_i U^T U U^T U x_i^T - 2 x_i U^T U x_i^T + x_i x_i^T)\end{aligned}$$

$$= \arg \min_U \sum_{i=1}^{n} (-x_i U^T U x_i^T + x_i x_i^T)$$

$$= \arg \min_U - \sum_{i=1}^{n} x_i U^T U x_i^T + \sum_{i=1}^{n} x_i x_i^T$$

$$\Leftrightarrow \arg \min_U - \sum_{i=1}^{n} x_i U^T U x_i^T$$

$$\Leftrightarrow \arg \max_U \sum_{i=1}^{n} x_i U^T U x_i^T$$

$$= \arg \max_U \mathrm{tr}(U(\sum_{i=1}^{n} x_i^T x_i)U^T)$$

$$= \arg \max_U \mathrm{tr}(UX^T X U^T) \quad s.t. UU^T = 1$$

可以看到，最小投影距离与我们在最大投影方差中得到的结果是一致的，只是两种不同的思考角度。

四、实验结果与分析

4.1 生成数据测试

我们通过可视化二维和三维数据的降维来观察 PCA 的效果。

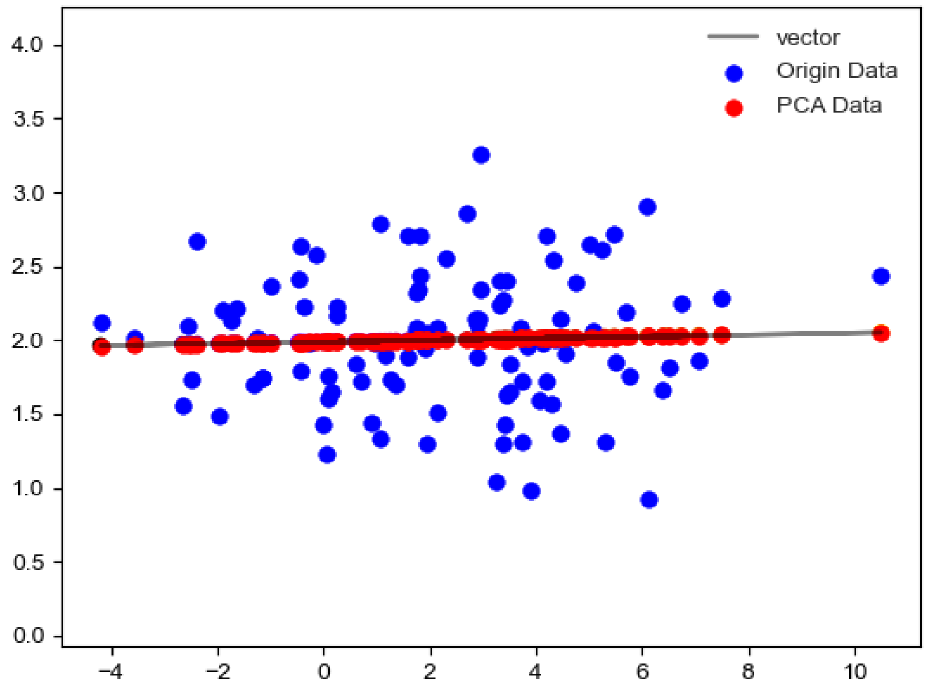
4.1.1 二维降到一维

生成高斯分布数据的参数：

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

可以看到第 2 维的方差远小于第 1 维的方差，所以第 2 维包含了更多的信息，

进行PCA，得到的结果如下：



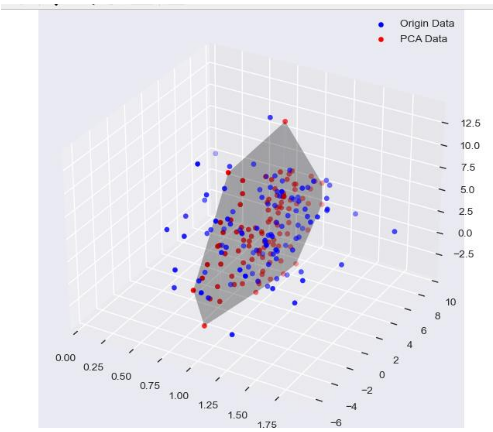
可以看到在 PCA 之后的数据分布在直线(1 维)上，另外其在横轴上的方差更大，纵轴上的方差更小，所以在进行PCA 之后得到的直线与横轴接近。

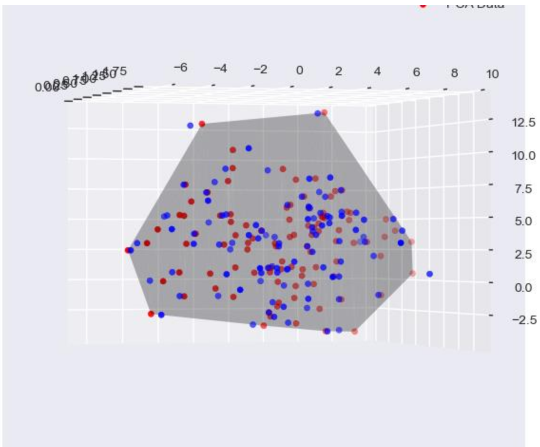
4.1.2 三维降到二维

生成高斯分布数据的参数：

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

第 1 维的方差是远小于其余两个维度的，所以在第 1 维相较于其他两维信息更少，如下是 PCA 得到的结果：





4.2 人脸数据测试

信噪比计算公式

100 维

```
图 1 的信噪比:  35.69060963432164
图 2 的信噪比:  53.83782522145776
图 3 的信噪比:  49.35074606553644
图 4 的信噪比:  55.25253202516963
图 5 的信噪比:  53.201233100212534
图 6 的信噪比:  50.74696591453579
图 7 的信噪比:  53.72391340798232
图 8 的信噪比:  53.10265678269357
图 9 的信噪比:  50.92012244157718
图 10 的信噪比:  48.35438069431231
```

50 维

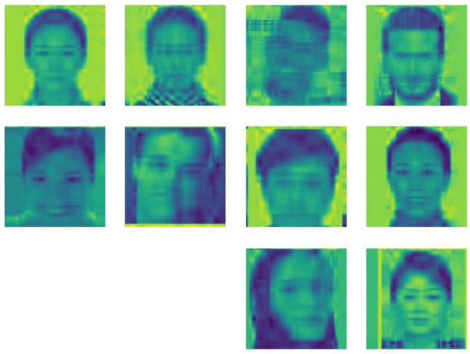
```
图 1 的信噪比:  27.946678008616296
图 2 的信噪比:  40.85001885932469
图 3 的信噪比:  36.11452191546023
图 4 的信噪比:  41.699714590843854
图 5 的信噪比:  39.901096248431465
图 6 的信噪比:  39.11563914841115
图 7 的信噪比:  40.72605339338024
图 8 的信噪比:  41.01949717220044
图 9 的信噪比:  38.51551868250142
图 10 的信噪比:  35.673095288737784
```

10 维

```
图 1 的信噪比:  21.499237852474806
图 2 的信噪比:  23.548622886481652
图 3 的信噪比:  23.616403351568955
图 4 的信噪比:  28.534902585302987
图 5 的信噪比:  26.71775055510786
图 6 的信噪比:  25.99311149764793
图 7 的信噪比:  26.697227999595917
图 8 的信噪比:  28.029300368233756
图 9 的信噪比:  25.54310843169032
图 10 的信噪比:  22.143690566372797
```

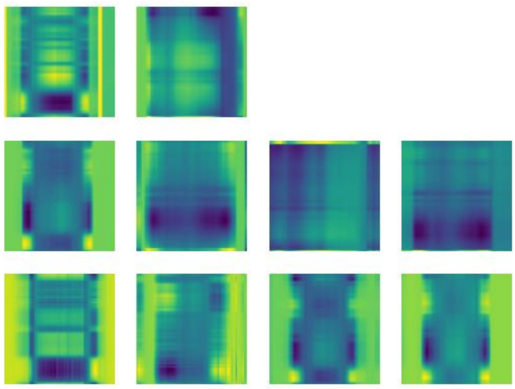
5 维

```
图 1 的信噪比:  19.454001862854845
图 2 的信噪比:  20.397135725188306
图 3 的信噪比:  20.197475347867716
图 4 的信噪比:  24.868676388823907
图 5 的信噪比:  23.483291547807568
图 6 的信噪比:  22.202950517415143
图 7 的信噪比:  22.880999792040623
图 8 的信噪比:  24.734718652097058
图 9 的信噪比:  21.99117415922329
图 10 的信噪比:  18.875996993408485
```



1 维

```
图 1 的信噪比:  14.4335396476551
图 2 的信噪比:  16.070754708883545
图 3 的信噪比:  15.13072931014722
图 4 的信噪比:  17.600787364612273
图 5 的信噪比:  16.921365516509184
图 6 的信噪比:  14.659476354745875
图 7 的信噪比:  17.19475524083047
图 8 的信噪比:  19.852714938872232
图 9 的信噪比:  14.820272275255192
图 10 的信噪比:  14.235771968366146
```



十张图片原为 250 * 250 像素大小。通过结果可以看出，随着维度的降低，信噪

比也越来越低。 k=50 和 100 时，都能较好的保留图片特征，人眼几乎无法分辨损失，降到 k=10 时，图片有了较为明显的损失，k=5 时，能勉强辨别出图片，但在降到 k=1 时，图片已完全模糊，失真。

这十张图片中，有相似也有差别较大的，可以看到，从降 10 维降到 5 维时，图一图二已经没有什么辨识度。而第 8 张图，直到降至 3 维，仍保留着主要特征，说明它和其他图的差别较大，从信噪比也可以看出。图 8 始终保持着极高的信噪比。在实验的过程中，还会遇到求解特征向量时，特征向量出现虚部的问题，但是发现只有在降维程度较高的时候（10 维以上），特征向量矩阵的后面才会出现虚部。在 10 维以内的特征向量矩阵求解结果，全部都是实特征向量。

五、结论

1. PCA 降低了训练数据的维度的同时保留了主要信息，但在训练集上保留下来的特征未必是模型真正所需特征，被舍弃掉的特征未必无用，只是在训练数据上没有表现，因此 PCA 有可能加重了过拟合。
2. PCA 算法中舍弃了 d-k 个最小的特征值对应的特征向量，一定会导致低维空间与高维空间不同，但是通过这种方式有效提高了样本的采样密度；并且由于较小特征值对应的往往与噪声相关，通过 PCA 在一定程度上起到了降噪的效果。
3. 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。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

[1] 周志华 著. 机器学习, 北京: 清华大学出版社, 2016 年 1 月

七、附录：源代码（带注释）

```python
import DataMaker

if __name__ == '__main__':
    size = 100
    mu = [1, 2, 3]
    sigma = [[0.1, 0, 0], [0, 10, 0], [0, 0, 10]]

```

```
    train_x=DataMaker.generate_data(mu,sigma,size)
    DataMaker.show_3D(train_x)

    # mu = [2, 2]
    # sigma = [[10, 0], [0, 0.2]]
    # train_x = DataMaker.generate_data(mu, sigma, size)
    # DataMaker.show_2D(train_x)
    # , 10, 5, 3, 1
    # k_list = [100,50, 10, 5, 3, 1]
    # DataMaker.face_process('photo',k_list)
```

```
import math

import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import os
from PIL import Image
# size = (50, 50) # 由于较大的数据在求解特征值和特征向量时很慢，故统一压缩图像为size 大小
import cv2

def pca(x,k):
    n = x.shape[0]
    mu = np.sum(x, axis=0) / n
    x_center = x - mu
    cov = (x_center.T @ x_center) / n

    values, vectors = np.linalg.eig(cov)
    index = np.argsort(values)  # 从小到大排序后的下标序列
    vectors = vectors[:, index[:-(k + 1):-1]].T  # 把序列逆向排列然后取前 k 个，转为行向量
    vectors=np.real(vectors)#为了防止当维度过高时产生复数
    return x_center, mu, vectors


#生成高斯分布数据

def generate_data(mu,sigma,size):
    """
        生成高斯分布数据
        @:param mu
```

```
        @:param sigma
    """
    x = np.random.multivariate_normal(mu, sigma, size)
    return x
#二维降到一维
def show_2D(x):
    x_center, mu, vectors = pca(x, 1)

    x_pca = x_center @ vectors.T @ vectors + mu
    plt.scatter(x_pca[:, 0].tolist(), x_pca[:, 1].tolist(), c=x_pca[:, 0].tolist(), cmap=plt.cm.gnuplot)

    plt.style.use('seaborn')
    plt.scatter(x[:, 0], x[:, 1], c="b", label="Origin Data")
    plt.scatter(x_pca[:, 0], x_pca[:, 1], c='r', label='PCA Data')
    plt.plot(x_pca[:, 0], x_pca[:, 1], c='k', label='vector', alpha=0.5)
    plt.legend(loc='best')
    plt.ylim(np.min(x[:, 1]) - 1, np.max(x[:, 1]) + 1)
    plt.show()
    # plt.style.use('default')
#将三维降到二维
def show_3D(x):
    x_center, mu, vectors = pca(x, 2)
    x_pca = x_center @ vectors.T @ vectors + mu
    # show
    plt.style.use('seaborn')
    fig = plt.figure()
    ax = Axes3D(fig)
    ax.scatter(x[:, 0], x[:, 1], x[:, 2], c="b", label='Origin Data')
    ax.scatter(x_pca[:, 0], x_pca[:, 1], x_pca[:, 2], c='r', label='PCA Data')
    ax.plot_trisurf(x_pca[:, 0], x_pca[:, 1], x_pca[:, 2], color='k', alpha=0.3)
    ax.legend(loc='best')
    plt.show()
    plt.style.use('default')
#读取指定目录下的照片
def face_process(path,k_list):
    file_list = os.listdir(path)
    x_list = []
    for file in file_list:
        file_path = os.path.join(path, file)
        img = plt.imread(file_path)
        plt.imshow(img)
```

```
        plt.axis('off')
        # plt.show()
        pic = Image.open(file_path).convert('L')  # 读入图片，并将三通道转换为灰度图

        x_list.append(np.asarray(pic))

    # n_samples, n_features = x_list.shape
    # print(data)
    for k in k_list:

        x_pca_list = []
        x_psnr_list = []
        for x in x_list:
            x_centerlized, mu, vectors = pca(x, k)  # PCA 降维
            x_pca = x_centerlized @ vectors.T @ vectors + mu  # 重建数据
            x_pca_list.append(x_pca)
            x_psnr_list.append(psnr(x, x_pca))
        print(len(x_pca_list))
        show_pic(k, x_pca_list, x_list, x_psnr_list)
#计算信噪比
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)
def show_pic(k, x_pca_list, x_list, x_psnr_list):
    #plt.figure(figsize=(8,5), frameon=False)
    size = math.ceil((len(x_list) + 1) / 2)
    # print(size)
    plt.subplot(2, size, 1)
    plt.title('Real Image')
    #print(x)
    # plt.imshow(x)
    # plt.axis('off')  # 去掉坐标轴
    #print(x_pca_list[1])
    # print(len(x_list))
    # print(len(x_pca_list))
    for i in range(len(x_list)):
        plt.subplot(3,4,i+1)
        # print(x_pca_list[i])
        # plt.title('k = ' + str(x_pca_list[i]) + ', PSNR = ' + '{:.2f}'.format(x_psnr_list[i]))
```

```python
        # print(i)
        plt.imshow(x_pca_list[i])
        print('图', i+1, '的信噪比: ', x_psnr_list[i])
        plt.axis('off')
    plt.show()
```