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： 选修

实验题目： PCA 模型实验

学号：1190200208
姓名：李旻翀

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

1. 实验要求

（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. 实验环境

VS code 2021 + Python + numpy + matplotlib

三、设计思想（本程序中的用到的主要算法及数据结构）

1. PCA 算法原理

PCA(主成分分析) 是一种常见的数据分析方式，主要功能是从高维数据中提取一部分特征（主成分），然后根据这些特征向低维变换，以求在数据压缩的情况下保留最主要的信息。PCA 算法常用于压缩数据大小以及高维数据可视化。

PCA 的数学推导可以从最大可分型和最近重构性两方面进行，前者的优化条件为划分后方差最大，后者的优化条件为点到划分平面距离最小，在本实验中采取最大可分型进行算法实现。

具体而言，最大可分型就是将高维数据向低维空间中的某一个平面投影，希望投影得到的数据点的方差最大，从而尽可能多的保留原数据的特征。

PCA 算法的推导如下：

设有一组数据$X=\{x_1,x_2,...,x_n\}$，其中的x_i都是D维空间的向量。规定μ_1是一个投影基，投影距离为$z=X^T\mu_1$。由此，我们可以得到投影均值为：

$$\overline{x}=\frac{1}{N}\Sigma_{n=1}^{N}\mu_1^T x_n$$

投影方差为：

$$\frac{1}{N}\Sigma_{n=1}^{N}(\mu_1^T x_n-\mu_1^T\overline{x})^2$$
$$=\mu_1^T(\frac{1}{N}\Sigma_{n=1}^{N}(x_n-\overline{x})(x_n-\overline{x})^T)\mu_1$$

设$S=\frac{1}{N}(x_n-\overline{x})(x_n-\overline{x})^T$，则方差可以表示为${\mu_1}^T S\mu_1$。

用拉格朗日乘子法最大化目标函数，有：

$$L(\mu_1)=\mu_1^T S\mu+\lambda(1-\mu_1^T\mu_1)$$

可以解得：

$$S\mu_1=\lambda\mu_1$$

而μ_1与λ是一组对应的S的特征向量和特征值。又

$$\mu_1^T S\mu_1=\lambda$$

故求得最大化方差，即求最大的特征值。要将D维数据降维到d维，只需计算前d个最大的特征值，将其对应的特征向量组合成特征向量矩阵，然后用右乘数据矩阵的转置即可实现降维压缩[1]。

2. PCA 算法步骤

总结一下 PCA 的算法步骤[2]：

给定样本集$X=\{x_1,x_2,...,x_m\}$和要降维到的维数d。

1) 将原始数据按列组成n行m列矩阵X；

2) 将X的每一行进行去中心化，即所有样本x_j减去$\mu=\frac{1}{m}\sum_{j=1}^{m}x_j$；

3) 求出协方差矩阵$C=\frac{1}{m}X^T X$；

4) 对协方差矩阵进行特征值分解，求出其特征值及对应的特征向量；

5) 取最大的d个特征值对应的单位特征向量$w_1,w_2,...,w_d$，构造投影矩阵

$W=\{w_1,w_2,...,w_d\}$

6) 输出投影矩阵W与样本均值μ

PCA 算法的核心代码如下：

```python
def PCA(data, new_dim):
    "实现PCA算法"
    x_mean = np.sum(data, axis=0) / num  # 求均值
    decentral_data = data - x_mean  # 中心化
    cov = decentral_data.T @ decentral_data  # 计算协方差
    eigenvalues, eigenvectors = np.linalg.eig(cov)  # 特征值分解
    eigenvectors = np.real(eigenvectors)
    dim_order = np.argsort(eigenvalues)  # 按从小到大获得特征值的索引
    PCA_vector = eigenvectors[:, dim_order[:-(new_dim + 1):-1]] # 选取最大的特征值对应的特征向量
    x_pca = decentral_data @ PCA_vector @ PCA_vector.T + x_mean  # 计算PCA之后的x值
    return PCA_vector, x_mean, x_pca
```

首先，我们按维度求得原始数据的均值，根据求得的均值对原数据进行中心化处理，然后生成中心化数据的协方差矩阵 cov ，并求 cov 的特征值和特征向量，求得后，我们对特征向量进行去除虚部的处理，并对特征值排序，取前 new_dim 个最大的特征值，并选取它们对应的特征向量组成特征向量矩阵，最后返回特征向量矩阵 PCA_vector ，降维前数据均值 x_mean，中心化数据 x_pca 。

3. 人工生成数据

为方便可视化，我们人工生成二维与三维数据进行 PCA 前后的对比实验。在生成相关数据时，利用多维高斯分布生成样本点，具体的参数为：

```python
def gen_data(num, dim):
    "生成数据"
    data = np.zeros((dim, num))

    if dim == 2:
        mean = [2, -2]
        cov = [[1, 0], [0, 0.01]]
    elif dim == 3:
        mean = [2, 2, 3]
        cov = [[1, 0, 0], [0, 1, 0], [0, 0, 0.01]]

    data = np.random.multivariate_normal(mean, cov, num)
    # print(data.shape) # (100,2)
    return data
```

在二维数据的情况下，我们生成的数据第一维方差远大于第二维方差；在三维数据的情况下，我们生成的数据第三维方差远小于第一、二维方差。这便于我们在测试后得到比较直观的结果。

4. 导入人脸数据

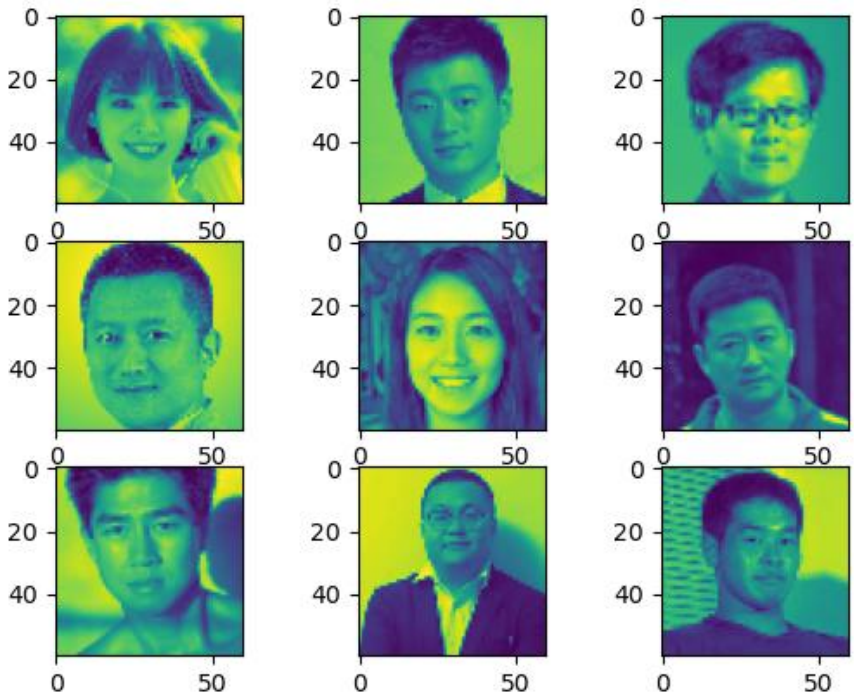
本次实验的第二个环节：人脸数据压缩采用了 9 张图片，均来自于谷歌图

片搜索，人工将九张图片裁剪为正方形大小，为保证运行速度不至于过慢，在实际使用时将图片大小压缩至 size = (60, 60)，并将图片处理为灰度图[3]，在此之后，将图片展平。此部分的核心函数如下：

```python
def read_data():
    "读入人脸数据并展示"
    img_list = os.listdir(filepath)  # 获取文件名列表
    data = []
    i = 1
    for img in img_list:
        path = os.path.join(filepath, img)
        plt.subplot(3, 3, i)
        with open(path) as f:
            img_data = cv2.imread(path)  # 读取图像
            img_data = cv2.resize(img_data, size)  # 压缩图像至size大小
            img_gray = cv2.cvtColor(img_data, cv2.COLOR_BGR2GRAY)  # 三通道转换为灰度图
            plt.imshow(img_gray)  # 预览
            h, w = img_gray.shape
            img_col = img_gray.reshape(h * w)  # 对(h,w)的图像数据拉平
            data.append(img_col)
        i += 1
    plt.show()

    return np.array(data)  # (9, 1600)
```

原图片经过初步处理后的结果如下：



5. 计算信噪比

图像的信噪比和图像的清晰度一样，都是衡量图像质量高低的重要指标。

具体而言，图像的信噪比是指视频信号的大小与噪波信号大小的比值，其公式为：

$$PSNR = 10\log_{10}\frac{MN}{\left\|f-\hat{f}\right\|^2}$$

其中，MN 是图形的大小，M*N，f 是真实图像，f~是退化图像[4]。
具体的实现函数如下：

```
def SNR(img1, img2):
    "计算信噪比"
    diff = (img1 - img2) ** 2
    mse = np.sqrt(np.mean(diff))
    return 20 * np.log10(255.0 / mse)
```

在本实验的人脸图像压缩环节，信噪比将作为衡量压缩后失真程度的指标。

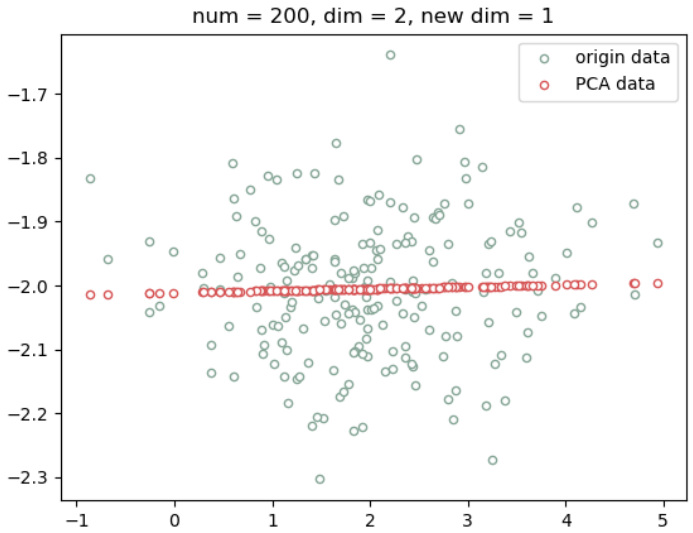
四、实验结果与分析

1. 自行生成数据

设置生成的样本点数为 200

1） 二维数据

设置原样本点维度 dim = 2，降维后的维度 new_dim = 1，运行程序得到如下结果：

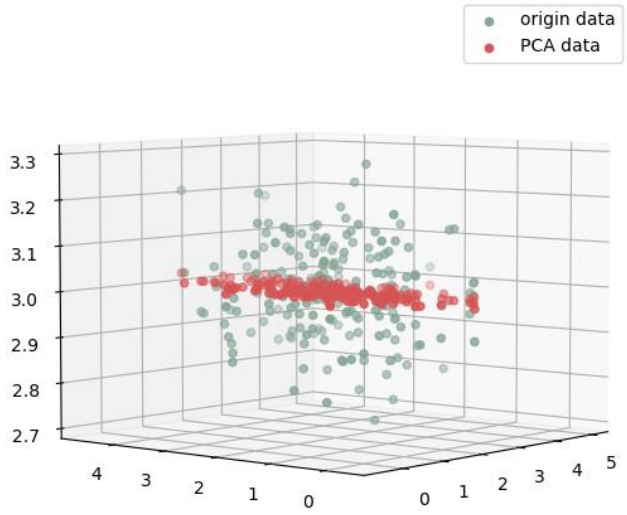


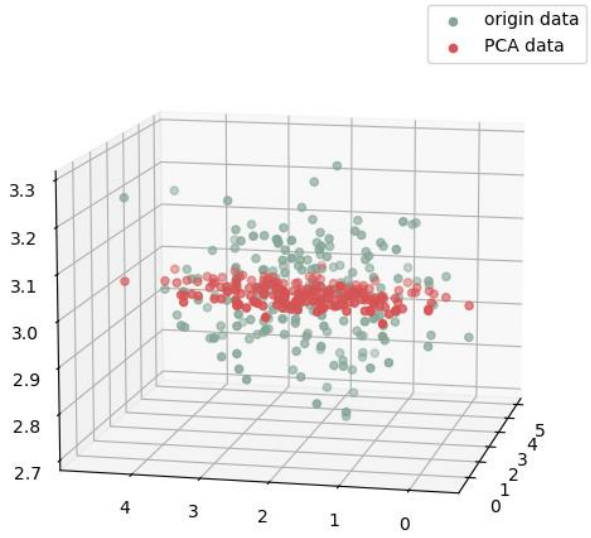
可以看到在 PCA 之后的数据分布在 1 维直线上。除此之外，PCA 后数据明显在横轴方向方差更大，纵轴方向方差更小，在进行 PCA 后得到的直线与横轴接近。这与我们生成数据时，第一维方差远大于第二维方差的结果

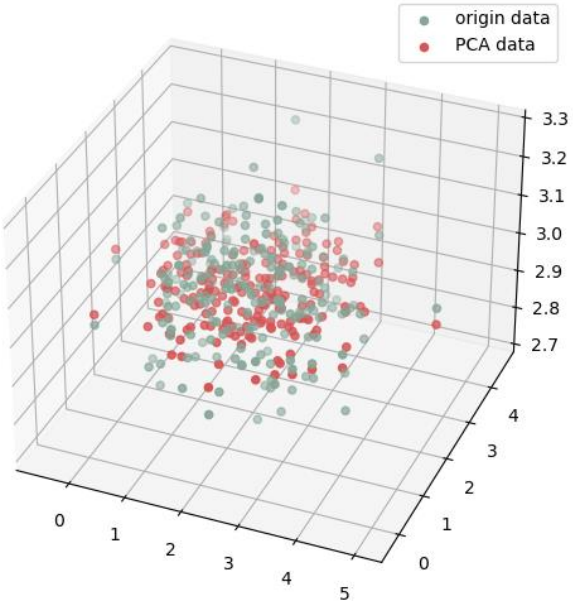
相吻合。

2) 三维数据

设置原样本点维度 dim = 3，降维后的维度 new_dim = 2，运行程序得到如下结果：

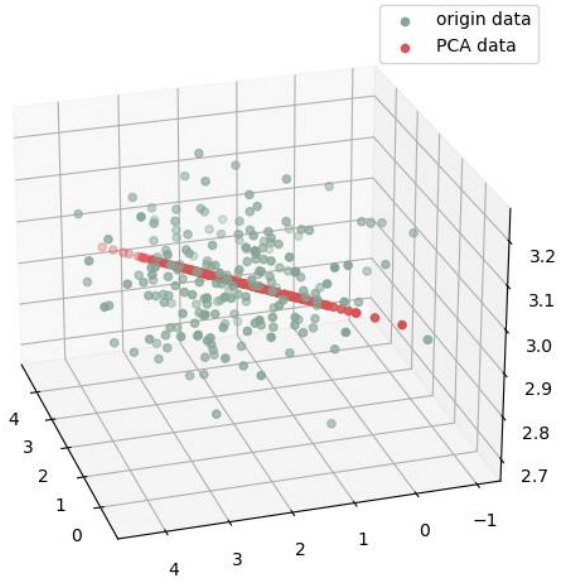


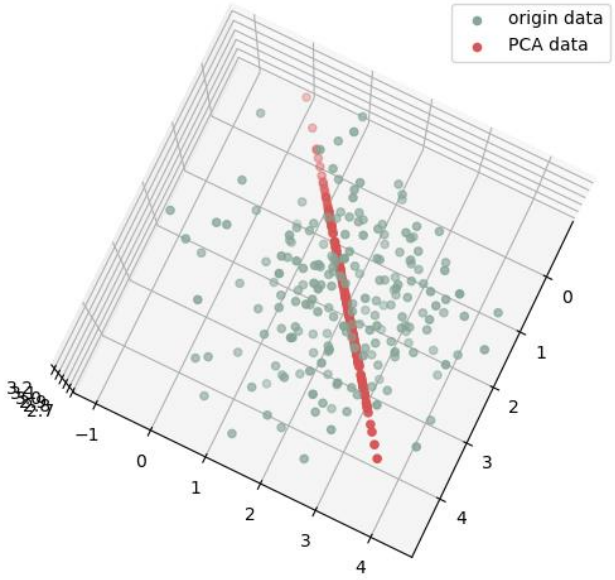




同样，可以看出在 PCA 之后的数据分布在二维平面上。这与我们生成数据时，第三维方差远小于第一、二维方差的结果相吻合。

我们再设定降维后的维度 new_dim = 1，观察实验结果：

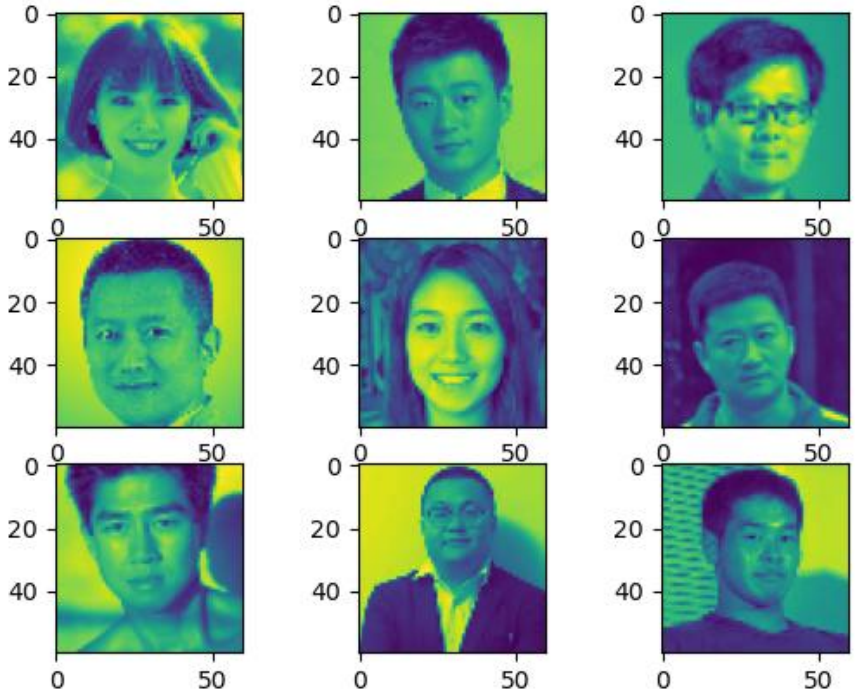




可以看出降维后的数据大致分布于一条直线上，这与我们的预期相符。总的来说，我们的 PCA 取得了较好的效果。

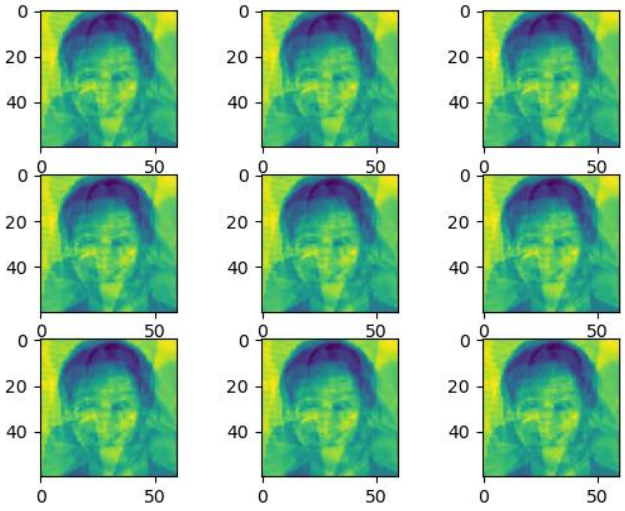
2. 人脸图片数据压缩

采用网络上搜集到的 9 张人脸图片进行数据压缩，原图如下：



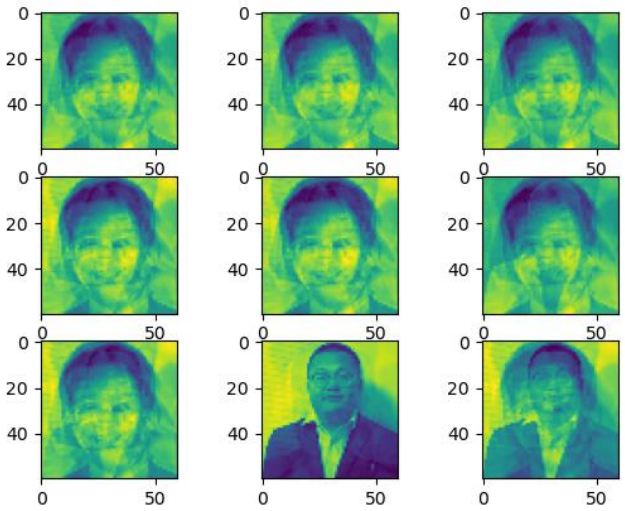
设置压缩后维度分别为 1, 3, 5, 7, 8, 9, 10，观察实验结果：

1） 压缩后维度为 1



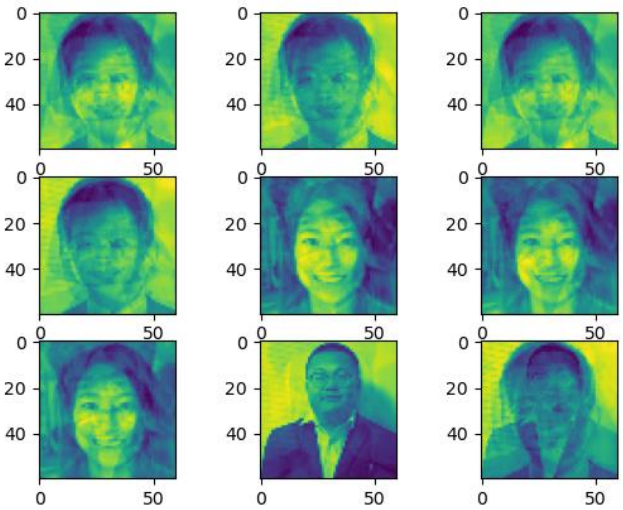
```
压缩后维度为 1，信噪比如下：
图 1，信噪比：12.459
图 2，信噪比：14.288
图 3，信噪比：15.990
图 4，信噪比：16.767
图 5，信噪比：11.858
图 6，信噪比：16.486
图 7，信噪比：13.198
图 8，信噪比：9.676
图 9，信噪比：15.477
```

2） 压缩后维度为 2



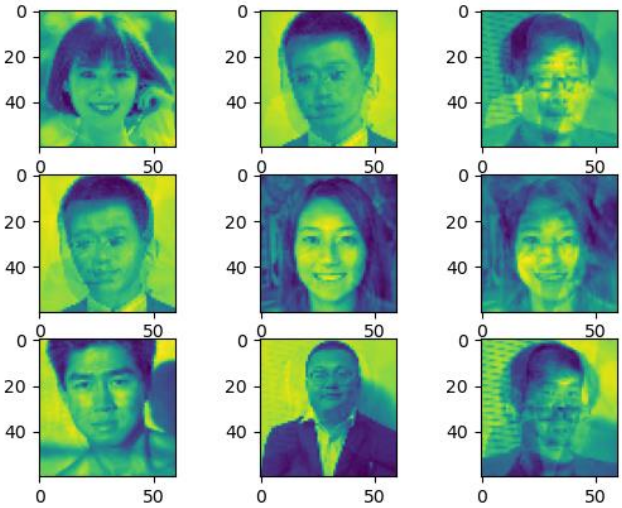
```
压缩后维度为 2，信噪比如下：
图 1，信噪比：13.111
图 2，信噪比：14.967
图 3，信噪比：17.503
图 4，信噪比：17.006
图 5，信噪比：11.913
图 6，信噪比：17.084
图 7，信噪比：13.210
图 8，信噪比：25.856
图 9，信噪比：15.944
```

3） 压缩后维度为 4



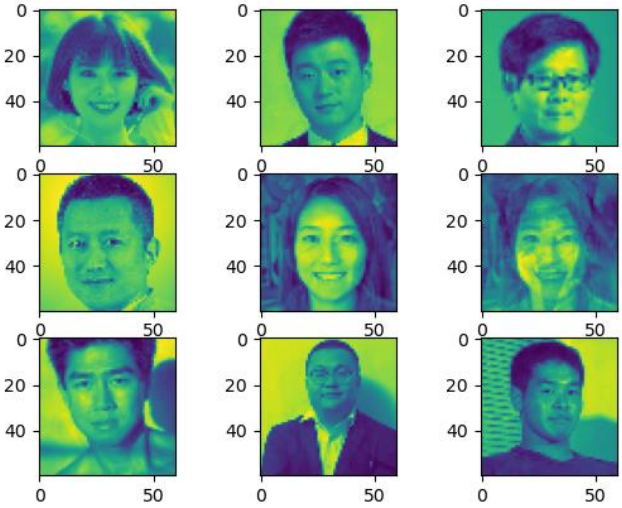
```
压缩后维度为 4，信噪比如下：
图 1，信噪比：24.557
图 2，信噪比：19.827
图 3，信噪比：17.544
图 4，信噪比：18.898
图 5，信噪比：18.731
图 6，信噪比：19.892
图 7，信噪比：15.513
图 8，信噪比：28.573
图 9，信噪比：17.495
```

4） 压缩后维度为 6



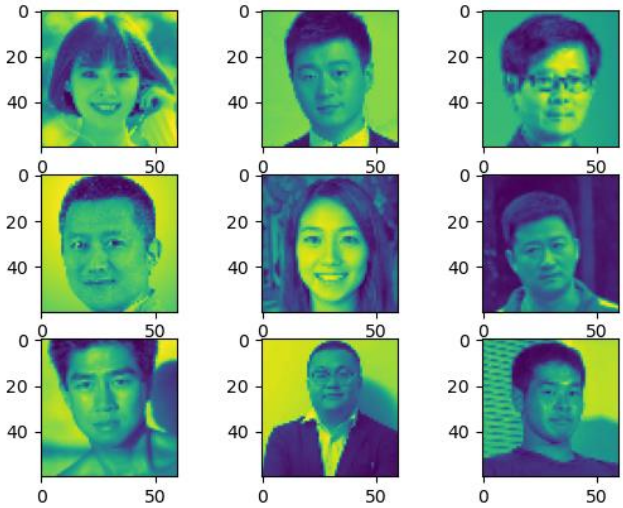
```
压缩后维度为 6，信噪比如下：
图 1，信噪比：32.827
图 2，信噪比：21.618
图 3，信噪比：20.560
图 4，信噪比：19.981
图 5，信噪比：27.784
图 6，信噪比：19.952
图 7，信噪比：32.561
图 8，信噪比：29.218
图 9，信噪比：21.315
```

5） 压缩后维度为 8



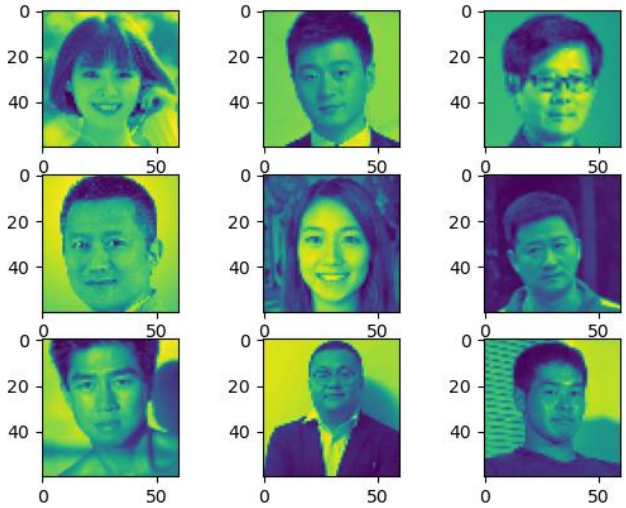
```
压缩后维度为 8，信噪比如下：
图 1，信噪比：36.689
图 2，信噪比：37.448
图 3，信噪比：49.020
图 4，信噪比：44.565
图 5，信噪比：29.341
图 6，信噪比：20.103
图 7，信噪比：37.068
图 8，信噪比：38.778
图 9，信噪比：43.159
```

6） 压缩后维度为 10



```
压缩后维度为 10，信噪比如下：
图 1，信噪比：292.418
图 2，信噪比：294.355
图 3，信噪比：293.292
图 4，信噪比：292.299
图 5，信噪比：289.915
图 6，信噪比：291.311
图 7，信噪比：291.823
图 8，信噪比：291.084
图 9，信噪比：290.850
```

7） 压缩后维度为 100



```
压缩后维度为 100，信噪比如下：
图 1，信噪比：292.436
图 2，信噪比：294.268
图 3，信噪比：293.236
图 4，信噪比：292.275
图 5，信噪比：289.908
图 6，信噪比：291.232
图 7，信噪比：291.793
图 8，信噪比：291.035
图 9，信噪比：290.874
```

可以从实验结果看出，随着维度从 1 开始升高，重构图像与原图像也越来越接近。信噪比也越来越高。在维度较低时，由于只能选取很少几个主成分，相似的图片会混在一起（如维度为 1 时，所有图片几乎都是一个样），而随着维度升高，主要特征较为明显的图片逐渐得到了较好的还原，当维度为 10 时，几乎所有的图片都得到了很好的还原，其特征被很好地保留，重构图像与原图像差异较小。

总的来说，2500 维的图片在被 PCA 压缩到 10 维时仍具有比较明显的辨识度，这说明通过 PCA 进行图片压缩可以极大地压缩图片大小，同时保留图片的明显特征。

五、结论

1. 通过 PCA 进行数据降维后会保留主要的特征，即主成分，不重要的特征将被忽略。数据降维后，会丢失一部分信息，降维越多，丢失的信息越多。
2. 在人脸图片数据压缩的实验中，2500 维的图片最低在压缩到 10 维时仍能保持比较好的辨识度，这说明 PCA 应用于图像数据压缩领域可以极大的压缩图片大小，同时使图片不至于过于失真。

六、参考文献

[1] 阿泽.【机器学习】降维——PCA（非常详细）[EB/OL].2020[2021-10-26].https://zhuanlan.zhihu.com/p/77151308.

[2] Microstrong0305.主成分分析（PCA）原理详解[EB/OL].2018[2021-10-26].https://blog.csdn.net/program_developer/article/details/80632779.
[3] GISer_Lin.opencv 学习 11——灰度图像读取[EB/OL].2018[2021-10-26].https://blog.csdn.net/nominior/article/details/82832314.
[4] lien0906.图像信噪比 SNR 求解[EB/OL].2014[2021-10-27].https://blog.csdn.net/lien0906/article/details/30059747.

七、附录：源代码（带注释）

```
import os
import cv2
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D


# 实验配置
num = 200  # 生成的原样本点数
dim = 3  # 原样本点维度
new_dim = 1  # PCA 后样本点维度

method = 2  # 1:自己生成数据进行 PCA 2:利用人脸数据进行 PCA

face_dim = 10  # 人脸图像要降至的维度
filepath = ".\\Face"  # 人脸数据图片的路径
size = (60, 60)  # 设置统一图片大小


def gen_data(num, dim):
    "生成数据"
    data = np.zeros((dim, num))

    if dim == 2:
        mean = [2, -2]
        cov = [[1, 0], [0, 0.01]]
    elif dim == 3:
        mean = [2, 2, 3]
        cov = [[1, 0, 0], [0, 1, 0], [0, 0, 0.01]]

    data = np.random.multivariate_normal(mean, cov, num)
    # print(data.shape) # (100,2)
    return data
```

```
def PCA(data, new_dim):
    "实现 PCA 算法"
    x_mean = np.sum(data, axis=0) / num  # 求均值
    decentral_data = data - x_mean  # 中心化
    cov = decentral_data.T @ decentral_data  # 计算协方差
    eigenvalues, eigenvectors = np.linalg.eig(cov)  # 特征值分解
    eigenvectors = np.real(eigenvectors)
    dim_order = np.argsort(eigenvalues)  # 按从小到大获得特征值的索引
    PCA_vector = eigenvectors[:, dim_order[:-(new_dim + 1):-1]] # 选取最
大的特征值对应的特征向量
    x_pca = decentral_data @ PCA_vector @ PCA_vector.T + x_mean  # 计算
PCA 之后的 x 值
    return PCA_vector, x_mean, x_pca


def PCA_show():
    "可视化 PCA 结果"
    data = gen_data(num, dim)  # 生成数据
    PCA_vector, x_mean, x_pca = PCA(data, new_dim)  # 执行 PCA 算法

    # 绘制散点图
    if dim == 2:  # 维数为 2
        plt.scatter(data.T[0], data.T[1], c='w',
                    edgecolors='#86A697', s=20, marker='o',
label='origin data')
        plt.scatter(x_pca.T[0], x_pca.T[1], c='w',
                    edgecolors='#D75455', s=20, marker='o', label='PCA
data')

    elif dim == 3:  # 维数为 3
        fig = plt.figure()
        ax = Axes3D(fig)
        ax.scatter(data.T[0], data.T[1], data.T[2],
                   c='#86A697', s=20, label='origin data')
        ax.scatter(x_pca.T[0], x_pca.T[1], x_pca.T[2],
                   c='#D75455', s=20, label='PCA data')

    plt.title("num = %d, dim = %d, new dim = %d" % (num, dim, new_dim))
    plt.legend()
    plt.show()
    return


def read_data():
```

```
    "读入人脸数据并展示"
    img_list = os.listdir(filepath)  # 获取文件名列表
    data = []
    i = 1
    for img in img_list:
        path = os.path.join(filepath, img)
        plt.subplot(3, 3, i)
        with open(path) as f:
            img_data = cv2.imread(path)  # 读取图像
            img_data = cv2.resize(img_data, size)  # 压缩图像至size大小
            img_gray = cv2.cvtColor(img_data, cv2.COLOR_BGR2GRAY)  # 三
通道转换为灰度图
            plt.imshow(img_gray)  # 预览
            h, w = img_gray.shape
            img_col = img_gray.reshape(h * w)  # 对(h,w)的图像数据拉平
            data.append(img_col)
        i += 1
    plt.show()

    return np.array(data)  # (9, 1600)


def SNR(img1, img2):
    "计算信噪比"
    diff = (img1 - img2) ** 2
    mse = np.sqrt(np.mean(diff))
    return 20 * np.log10(255.0 / mse)


def face_show():
    "人脸数据 PCA"
    data = read_data()
    n, pixel = data.shape
    PCA_vector, x_mean, x_pca = PCA(data, face_dim)
    x_pca = np.real(x_pca)  # 仅保留实部
    # 绘制PCA后的图像
    plt.figure()
    for i in range(n):
        plt.subplot(3, 3, i+1)
        plt.imshow(x_pca[i].reshape(size))
    plt.show()
    # 计算信噪比
    print("压缩后维度为 %d，信噪比如下：" % face_dim)
    for i in range(n):
```

```
        snr = SNR(data[i], x_pca[i])
        print("图 %d, 信噪比: %.3f" % (i+1, snr))
    return


if method == 1:
    PCA_show()
elif method == 2:
    face_show()
```