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： 专业限选

实验题目： PCA 模型实验

学号：7203610121
姓名： 刘天瑞

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

Windows 11；Visual Studio 2022；Python3.9.7 ：需要调用 numpy 库和 matplotlib 库

三、设计思想（本程序中的用到的主要算法及数据结构）

3.1. PCA 算法原理

PCA(主成分分析算法) 是一种常见的数据分析方式，主要功能是常用于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，即降维，来求在数据压缩的情况下保留最主要的特征信息分量。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入手，前者的优化条件为划分后方差最大，而后者的优化条件为原始数据点到划分平面的距离最小，但是这二者可以得到等价推导。在本实验中采取最大可分性入手进行算法实现。

要解决降维问题，要找到一组基，使用原始数据集上的点在这一组基下的投影结果即可。在线性代数，基(basis)（也称为基底）是描述和刻画向量空间的基本工具。向量空间的基是它的一个特殊的子集，基的元素称为基向量。PCA 的目的就是找到一组基，在这组基构建的空间中表示原始的数据。

可以写出基变换的简单矩阵表示形式：

$$(p_1, p_2, .., p_R)^T(a_1, a_2, .. a_M) = \begin{pmatrix} p_1a_1\ p_2a_2 \ldots p_1a_M \\ \ldots \\ p_Ra_1\ p_Ra_2 \ldots p_Ra_M \end{pmatrix}$$

其中 p_i 表示一个行向量，表示第 i 个基，a_j 表示一个列向量，也就是发生映射之前的原始数据的值。而上述的矩阵相乘事实上就是一种线性空间的变换，将原始数据映射到了由 p_i 表示的向量空间中。

具体而言，最大可分性的优化角度就是将高维数据向低维空间中的某一个平面投影，希望映射投影后得到的数据点组成数据集的方差最大，即映射之后的点之间的距离越远越好，从而尽可能多的保留原数据的特征。从熵的角度来理解，尽管对数据集进行了降维，但是仍旧希望降维之后的数据能携带较多的信息，而如果映射之后的数据的距离太近则会导致熵相对较小，携带的信息也相对较少。

PCA 算法的推导过程如下所示：

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

$$\bar{x} = \frac{1}{N}\sum_{n=1}^{N} \mu_1^T x_n$$

投影方差为：

$$\frac{1}{N}\sum_{n=1}^{N}(\mu_1^T x_n - \mu_1^T \bar{x})^2 = \mu_1^T \left(\frac{1}{N}\sum_{n=1}^{N}(x_n - \bar{x})(x_n - \bar{x})^T\right) \mu_1$$

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

使用拉格朗日乘子法来最大化目标函数，有：

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

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。又因为

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

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。

2. PCA 算法具体步骤

我总结的 PCA 算法的具体步骤如下所示：

1. 首先给定样本集 X={x_1，x_2，…，x_m}和要降维到的维数 d；
2. 将原始数据按列组成 n 行 m 列矩阵 X；

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

4. 求出协方差矩阵 $C = \frac{1}{m} X^T X$；

5. 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；

6. 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 $W=\{w_1, w_2, \ldots, w_d\}$；

7. 输出投影矩阵 W 与样本均值 μ；

PCA 算法具体运行代码的主要部分如下图所示：

```
38  def PCA(data, new_dim):
39      "实现PCA算法"
40      x_mean = np.sum(data, axis = 0) / num#求取原数据均值
41      decentral_data = data - x_mean#将散点数据中心化处理
42      cov = decentral_data.T @ decentral_data#计算中心化数据的协方差矩阵
43      eigenvalues, eigenvectors = np.linalg.eig(cov)#求取特征值和特征向量，再进行分解
44      eigenvectors = np.real(eigenvectors)#取实部值（去虚部处理）
45      dim_order = np.argsort(eigenvalues)#按照从小到大排序来获得特征值的索引
46      PCA_vector = eigenvectors[:, dim_order[:-(new_dim + 1):-1]]#选取最大的特征值对应的特征向量
47      x_pca = decentral_data @ PCA_vector @ PCA_vector.T + x_mean#计算经过PCA算法分析之后的x值
48
49      return PCA_vector, x_mean, x_pca#返回特征向量矩阵，降维前数据均值以及中心化数据
```

图 1 实现 PCA 算法的具体运行代码

思路：首先按照维度求出原数据的均值，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 ，并且求 cov 的特征值和特征向量，然后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，最后返回的是特征向量矩阵 PCA_vector，降维前数据均值 x_mean 以及中心化数据 x_pca 。

3. 人工生成数据

为了便于可视化，不妨人工生成二维与三维数据进行 PCA 算法处理前后的对比实验。在生成相关数据时，利用多维高斯分布（GMM）生成样本点，其中具体的关键参数与函数如下图所示：

```python
def generate_data(number, dimension):
    "生成数据"
    data = np.zeros((dimension, number))

    if dimension == 2:#二维数据情况下
        mean = [2, -2]
        cov = [[1, 0], [0, 0.01]]
    elif dimension == 3:#三维数据情况下
        mean = [2, 2, 3]
        cov = [[1, 0, 0], [0, 1, 0], [0, 0, 0.01]]

    data = np.random.multivariate_normal(mean, cov, number)#利用多维高斯分布生成样本点
    #print(data.shape)#(100,2)

    return data
```

图 2 生成数据函数的具体运行代码

不难发现，在二维数据情况下，生成的数据第二方差远小第一维差；而在三维数据情况下，生成的数据第三维方差远小于第一、二维方差，这便于在测试后得到更直观的实验结果。

4. 导入人脸数据

本次实验的下一个环节是导入人脸数据，人脸数据压缩我采用了 9 张均来自我自己拍的图像，先将九张图像裁剪为正方形大小尺寸，从而保证程序运行速度不至于过慢，在实际使用时将图像大小压缩至 size = (175, 175)，并将图像处理为灰度图，然后将图片展平。该部分读入人脸数据的函数主要运行代码如下图所示：

```python
def read_face_data():
    "读入人脸数据并且进行展示"
    img_list = os.listdir(filepath)#获取图像文件夹名列表
    data = []
    i = 1#初始化每行的第i个图像

    for img in img_list:
        path = os.path.join(filepath, img)#输入图像
        plt.subplot(3, 3, i)#直接指定划分方式和位置进行绘图，共有3*3张图像
        with open(path) as _:
            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)#RGB三通道图像转换为灰度图
            plt.imshow(img_gray)#进行图像预览
            h, w = img_gray.shape#设置灰度图的高度和宽度数据
            img_col = img_gray.reshape(h * w)#对(h,w)的图像数据进行拉平
            data.append(img_col)
        i += 1#顺次迭代
    plt.show()#图像展示

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

图 3 读入人脸数据函数的具体运行代码

原来的 9 张图像经过初步处理后得到的灰度图结果如下图所示：

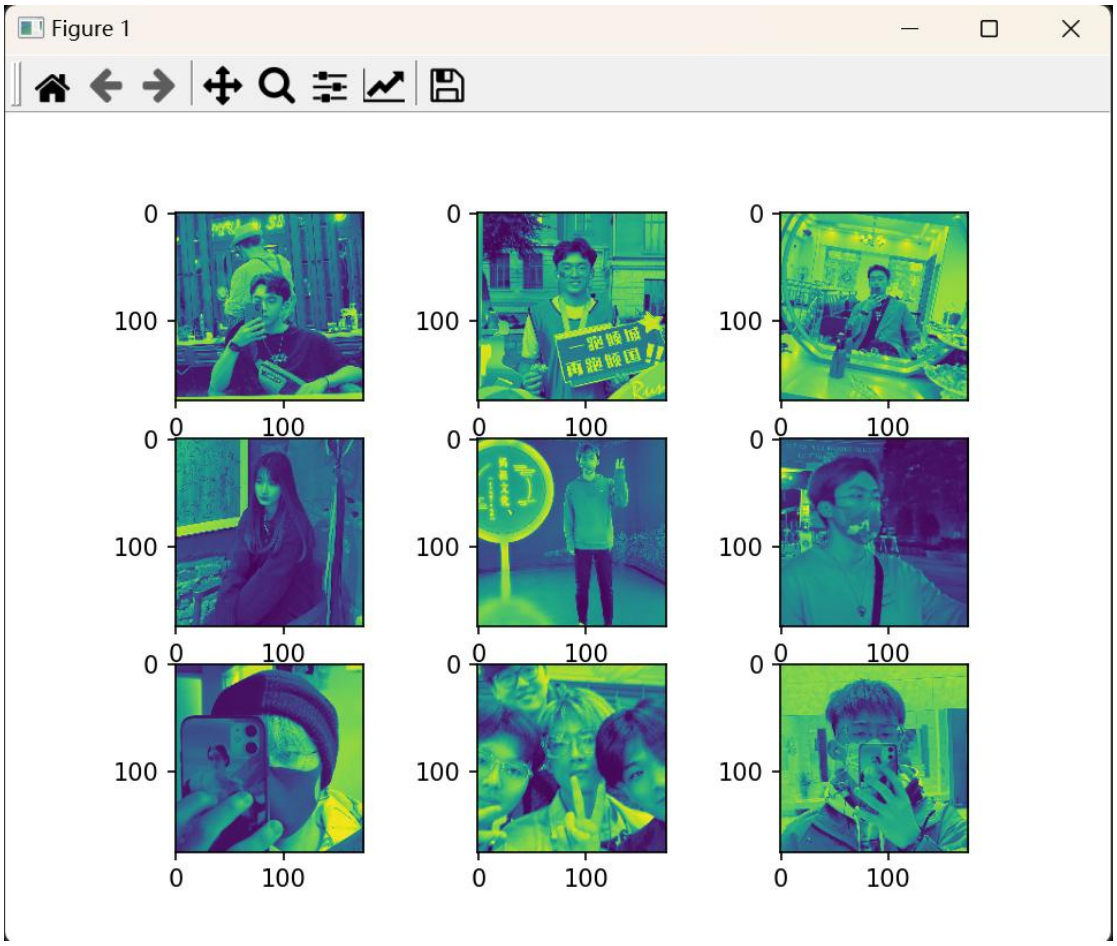


图 4 原始灰度图像处理结果

5. 计算信噪比

图像的信噪比与图像的清晰度一样，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，信噪比可以用来衡量两个信号之间的相似程度，信噪比越大表示两个信号越相似，所以也经常用作图像压缩等领域中信号重建质量的测量方法。其定义公式为：

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

其中，MN 是图像尺寸大小：M*N，f 是真实图像，$\hat{f}$ 是退化图像。

计算信噪比的具体实现函数的运行代码如下图所示：

```
102 def PSNR(img_1, img_2):
103     "计算图像信噪比"
104     diff = (img_1 - img_2) ** 2#进行二阶幂运算
105     mse = np.sqrt(np.mean(diff))#求取均值的开方
106
107     return 20 * np.log10(255.0 / mse)#PSNR计算公式
```

图 5 计算信噪比函数的具体运行代码

所以在本实验人脸图像数据压缩环节中，信噪比就将作为衡量压缩后失真程度的具体指标。

四、实验结果与分析

1. 自行生成数据

首先不妨设置生成的样本点数量为 300 个；

1) 二维样本数据集

设置原样本点维度 dim = 2，降维后的新维度 new_dim = 1，运行程序后得到如下图所示的实验结果：

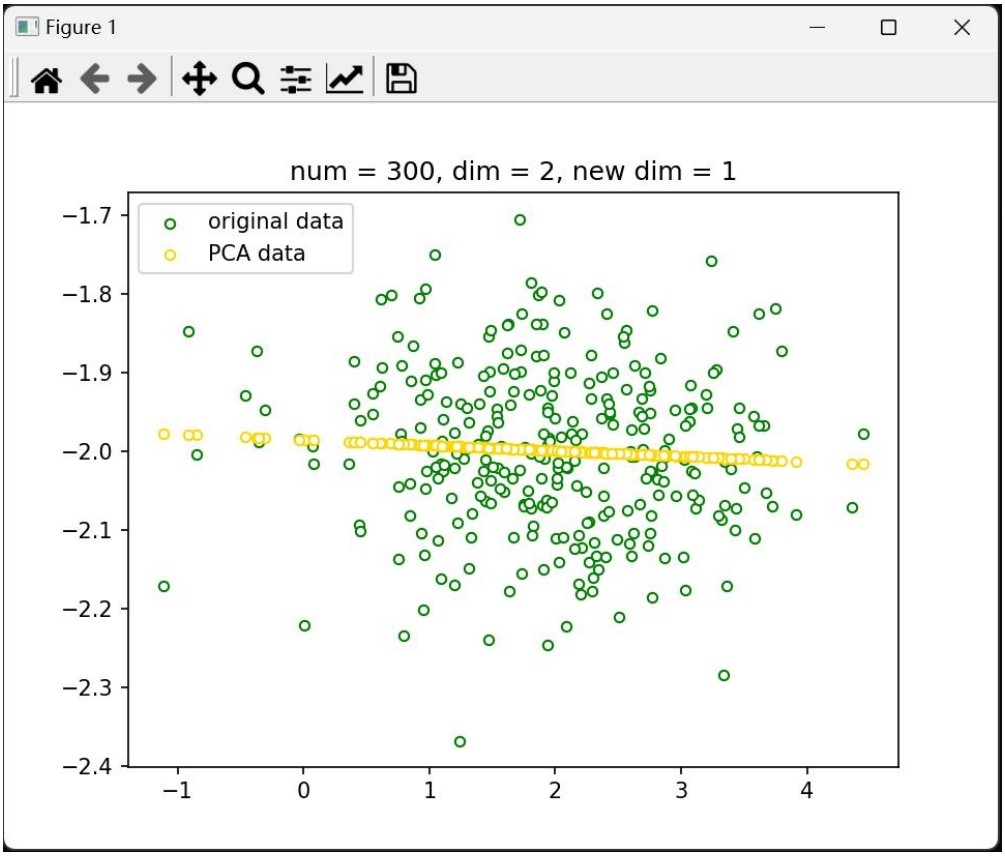
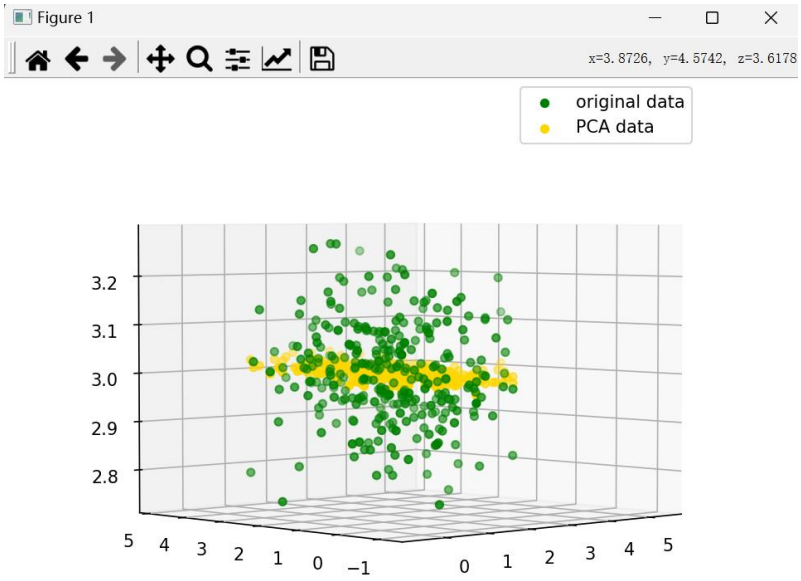


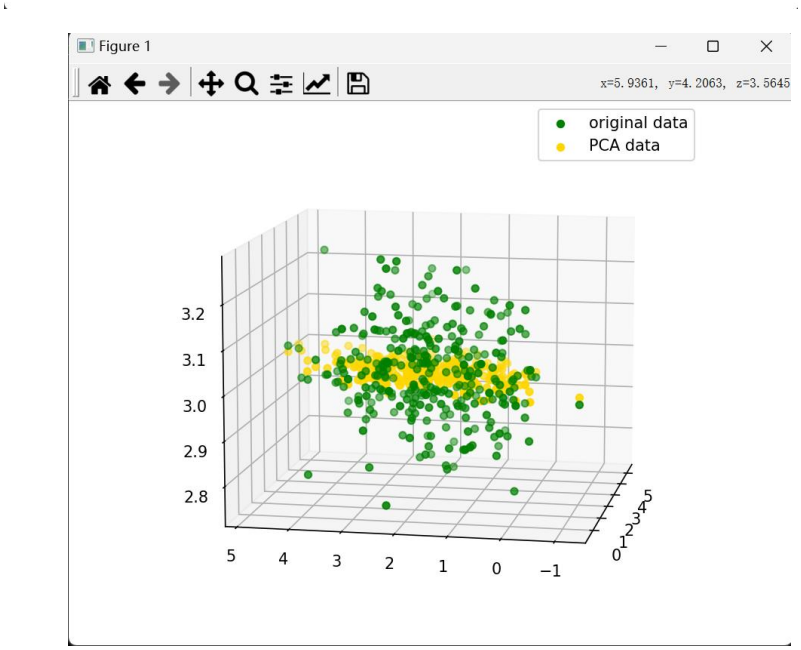
图 6 二维样本数据集二维降至一维散点图

可以看到在经过 PCA 算法处理后数据分布在一条一维的直线上。除此之外，明显发现散点数据在横轴方向上方差更大，纵轴方向上方差更小，在进行 PCA 算法处理后得到的直线与横轴接近。这与生成数据时第一维方差远大于第二维方差的理论结果相吻合。

2) 三维样本数据集

设置原样本点维度 dim = 3，降维后的新维度为 new_dim = 2，运行程序后得到如下图所示的实验结果（为了方便观察更换角度）：





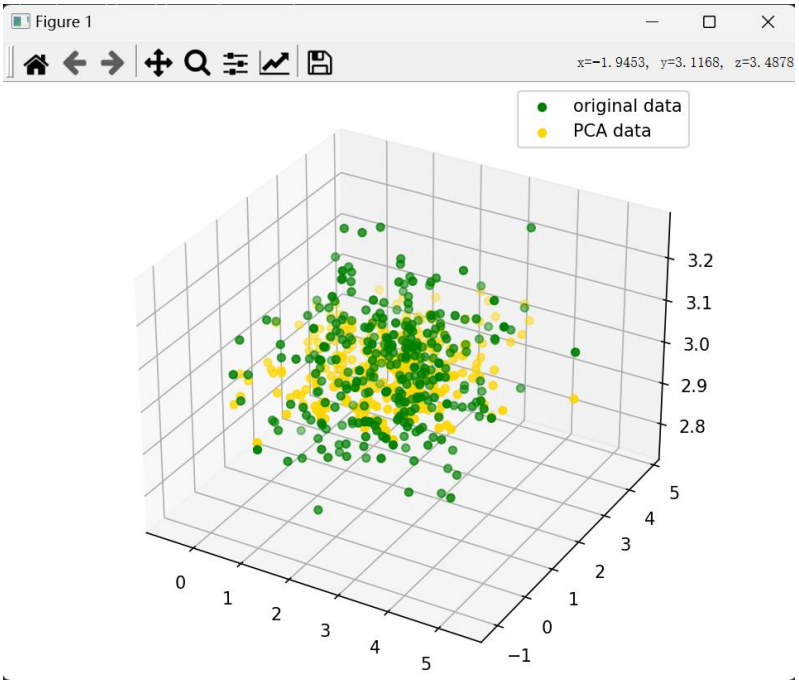
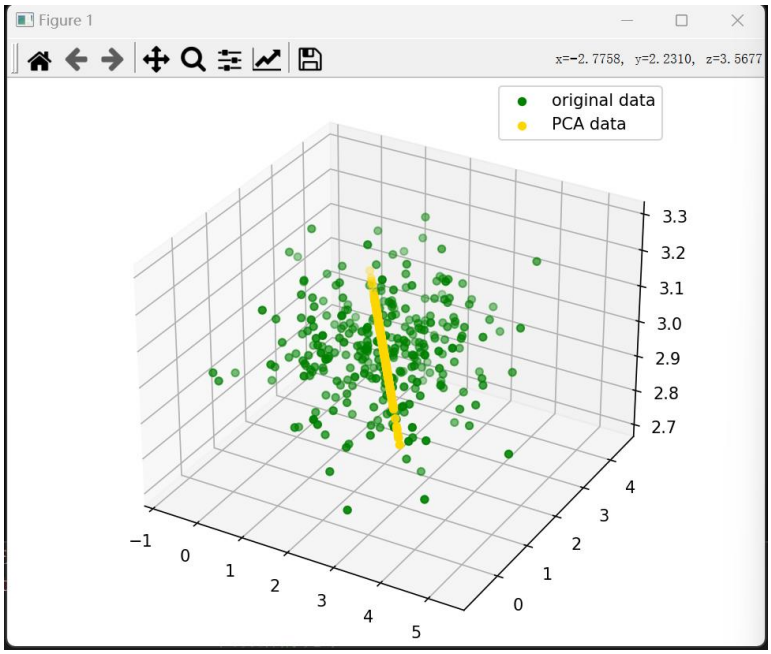


图 7 三维样本数据集三维降至二维散点图

同理可得在经过 PCA 算法处理后数据投影分布在一个二维平面上。这与生成数据时第三维方差远小于第一维和第二维方差的理论结果相吻合。

不妨再设定降维后的新维度为 new_dim = 1，运行程序后得到如下图所示的实验结果（为了方便观察更换角度）：



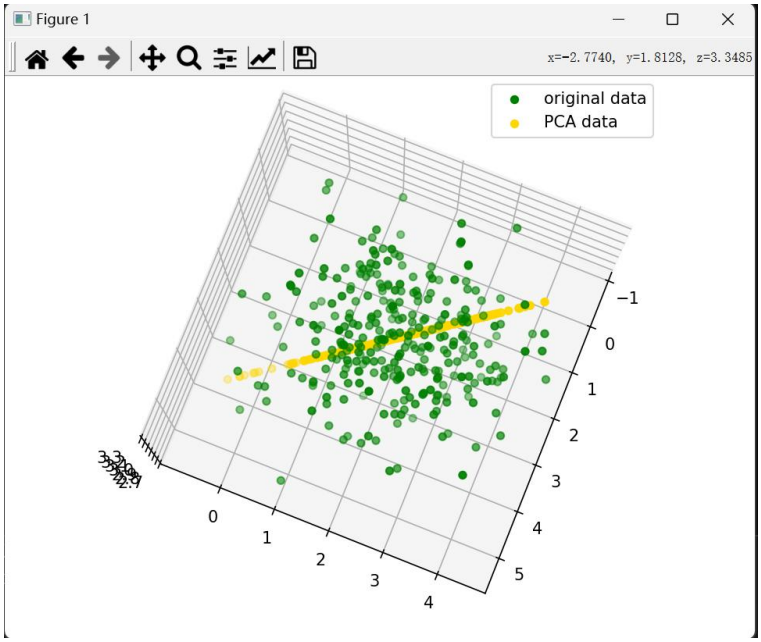


图 8 三维样本数据集三维降至一维散点图

可以看出降维后的数据大致分布在一条一维的直线上，这与实验前的理论预期相符。总体来说设计好的 PCA 算法程序取得了较好的实验效果。

2. 人脸图像数据压缩

采用我自己拍摄的 9 张包含人脸的图像进行数据压缩，设置压缩后的维度分别为 1，2，4，6，8，10，100 维，运行程序后观察得到如下图所示的实验结果：

1) 图像数据压缩后维度降至为一维：

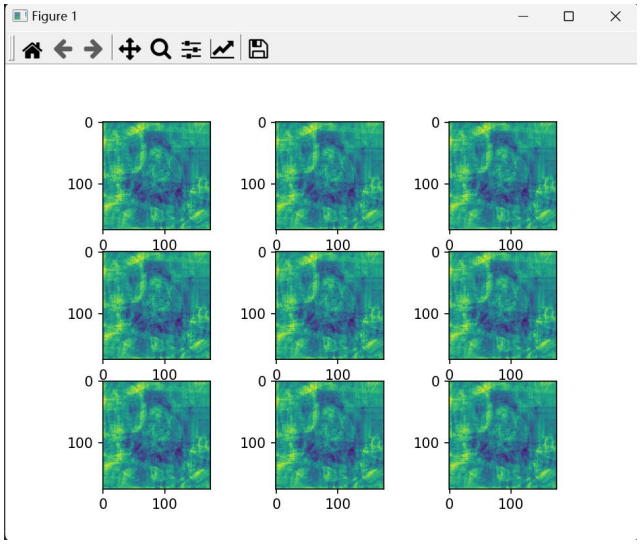


图 9 图像数据压缩后维度降至为一维的实验结果

图 10 降至一维后图像的信噪比

2) 图像数据压缩后维度降至为二维：

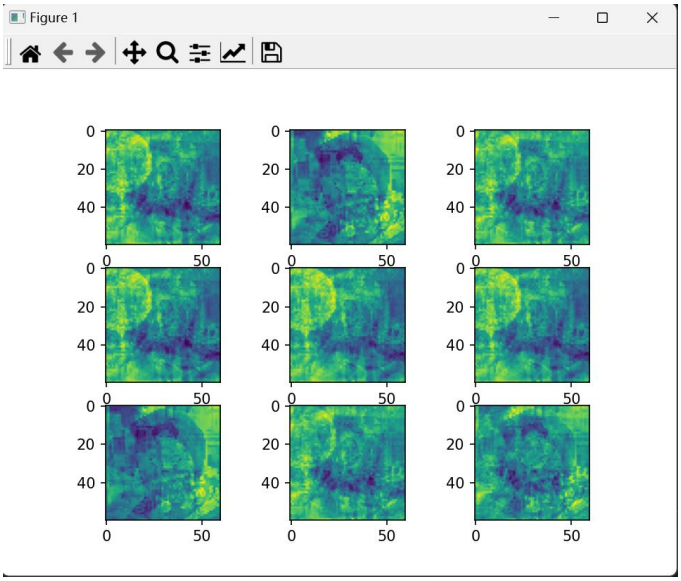


图 11 图像数据压缩后维度降至为二维的实验结果

图 12 降至二维后图像的信噪比

3) 图像数据压缩后维度降至为四维：

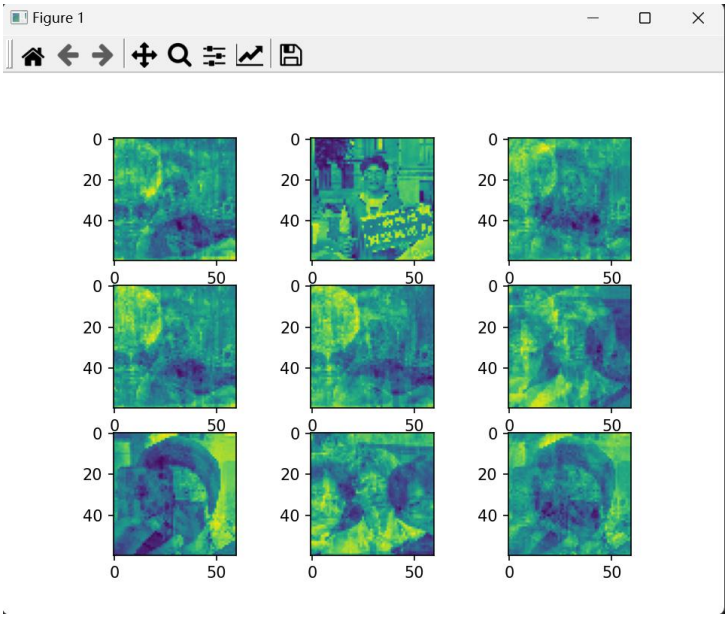


图 13 图像数据压缩后维度降至为四维的实验结果

```
C:\Users\刘天瑞\AppData\Local\Programs\Python\Python39\python.exe
压缩后维度为 4，信噪比如下：
图1，信噪比：15.056
图2，信噪比：28.341
图3，信噪比：15.697
图4，信噪比：16.180
图5，信噪比：16.493
图6，信噪比：15.473
图7，信噪比：19.218
图8，信噪比：20.172
图9，信噪比：14.895
Press any key to continue . . .
```

图 14 降至四维后图像的信噪比

4) 图像数据压缩后维度降至为六维：

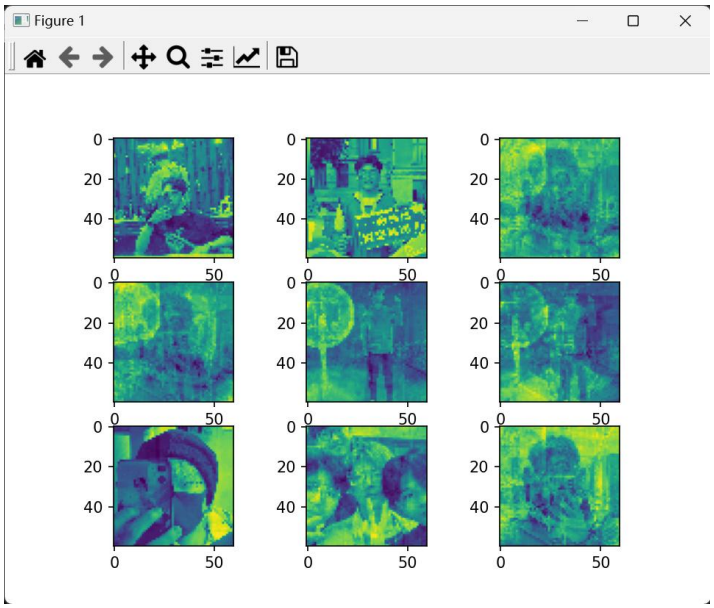


图 11 图像数据压缩后维度降至为六维的实验结果

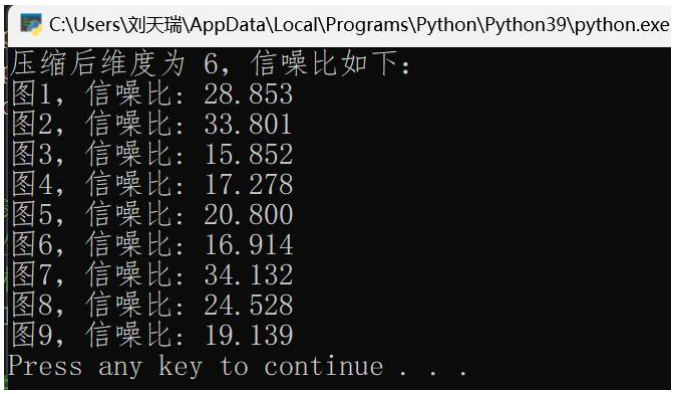


图 12 降至六维后图像的信噪比

5) 图像数据压缩后维度降至为八维：

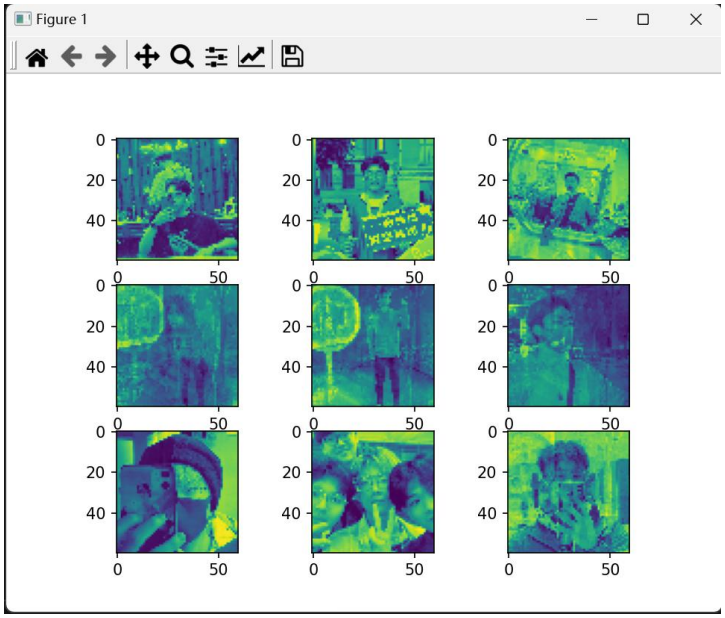


图 11 图像数据压缩后维度降至为八维的实验结果

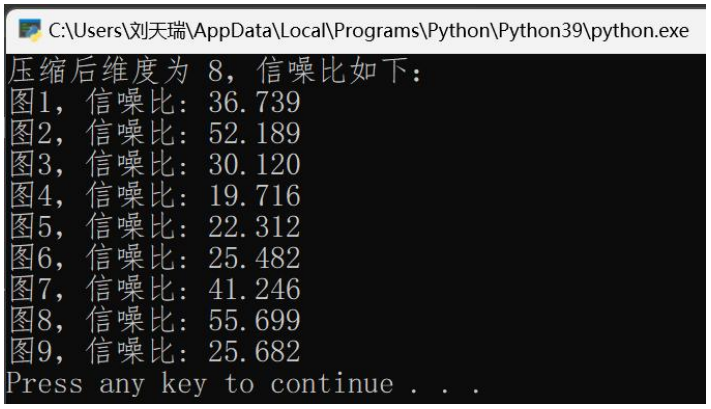


图 12 降至八维后图像的信噪比

6) 图像数据压缩后维度降至为十维：

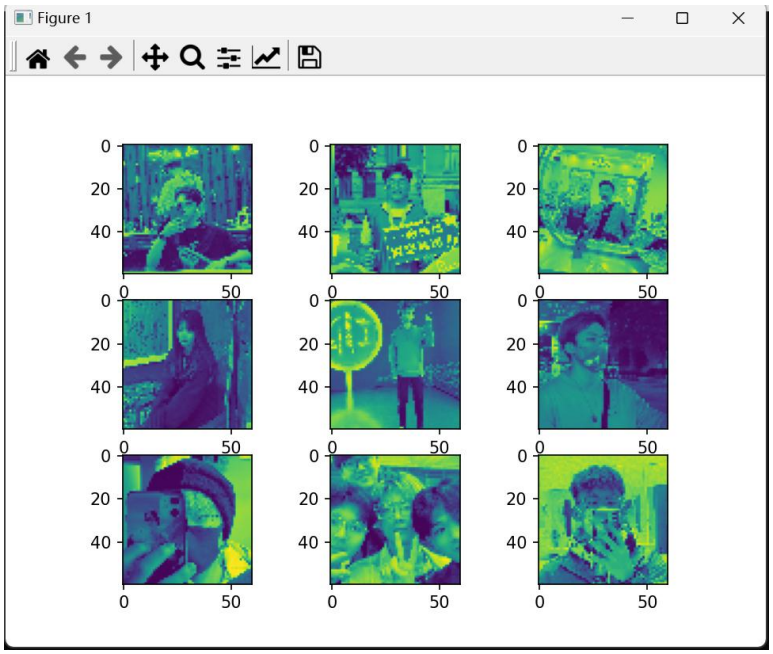


图 11 图像数据压缩后维度降至为十维的实验结果

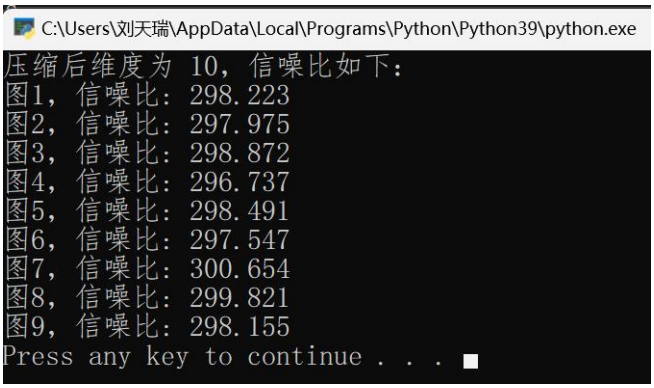


图 12 降至十维后图像的信噪比

7) 图像数据压缩后维度降至为一百维：

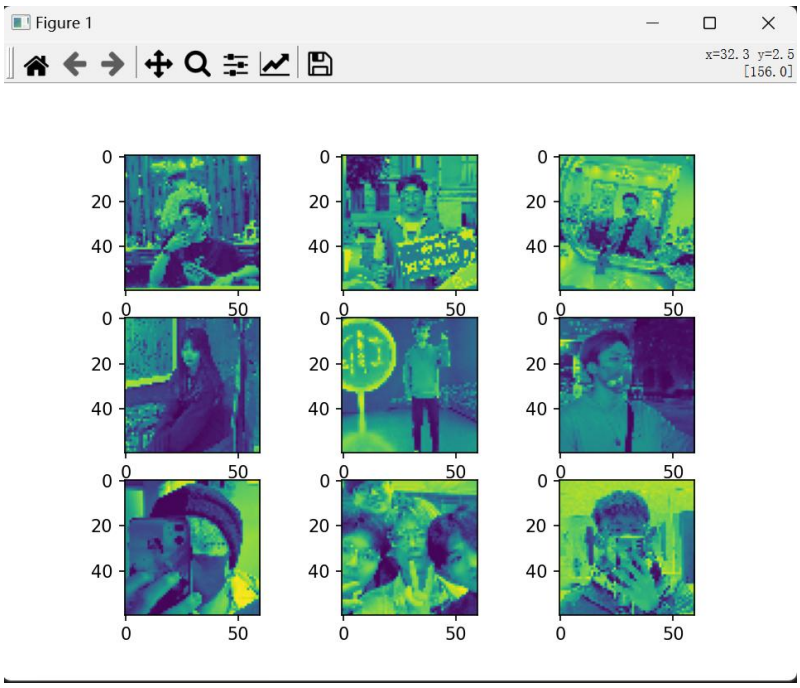


图 11 图像数据压缩后维度降至为一百维的实验结果

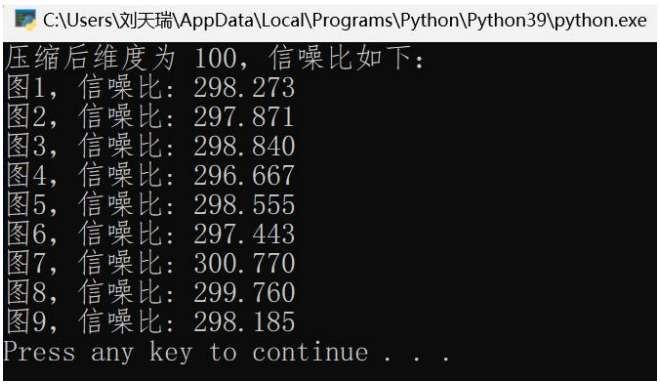


图 12 降至一百维后图像的信噪比

从实验结果中可以看出：随着维度从一维开始逐渐升高，重构图像与原始图像清晰度也越来越接近，信噪比也越来越高。在维度较低时，由于只能选取很少的主成分，相似图像会混叠在一起（如维度为一维时，所有图像几乎相同的模糊），只存在基本轮廓，细节与原图差距较大。而随着维度的升高，主要特征较为明显的图像逐渐得到了较好地还原，当维度为 10 时，几乎所有图像都已经得到了很好还原，其特征也被很好地保留，重构图像与原图像差异也较小。

总体来说 2500 维左右的图像在被 PCA 算法处理压缩至 10 维时仍然具有较明显的辨识度，即可以发现降维至 10 维之后重构图像和原图像在肉眼看来基本一致，和 PSNR 信噪比结果显示的基本一致。这说明通过 PCA 算法处理可以极大地压缩图片数据尺寸大小并且同时保留图片基本的明显特征。

五、结论

1. 通过 PCA 算法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，丢失的信息也就越多；

2. PCA 算法通过寻找协方差矩阵的最大的 K 个特征值对应的向量作为一组基来表示源数据达到降维的效果，通过实验发现这种手段是相当可靠的；

3. 在人脸图像数据压缩的实验中，采用 50*50，即 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地压缩图片大小，同时使图片不至于过于失真；

4. 同时需要注意的是 PCA 算法提高了样本的采样密度，并且由于较小特征值对应的特征向量往往容易受到噪声的影响，PCA 算法舍弃了这部分所谓的“次要成分”，在一定程度上起到了降噪的效果。

六、参考文献

[1] 周志华，《机器学习》，清华大学出版社，2016

[2] 机器学习——PCA 降维（我至今为止遇见的最好的博文）_zouxiaolv 的博客-CSDN 博客_pca 降维

https://blog.csdn.net/zouxiaolv/article/details/100590725?ops_request_misc=&request_id=&biz_id=102&utm_term=%E6%9C%BA%E5%99%A8%E5%AD%A6%E4%B9%A0PCA%E9%99%8D%E7%BB%B4&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default-9-10059072

[3] (一)OpenCV-Python 学习—基础知识 - silence_cho - 博客园 (cnblogs.com)

https://www.cnblogs.com/silence-cho/p/10926248.html

[4] 图像信噪比 SNR 求解_lien0906 的博客-CSDN 博客_图像信噪比计算公式

https://blog.csdn.net/lien0906/article/details/30059747?ops_request_misc=&request_id=&biz_id=102&utm_term=%E5%9B%BE%E5%83%8F%E4%BF%A1%E5%99%AA%E6%AF%94SNR&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default-1-30059747.142^v62^opensearch_v2,201^v3^add_ask,213^v1^t3_esquery_v1&spm=1018.2226.3001.4187

七、附录：源代码（带注释）

```
# -*- coding: gbk -*-

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


#实验关键参数配置
number = 300#生成的原样本点数量
dimension = 3#原样本点的维度
new_dimension = 1#PCA算法处理后样本点的维度
method = 2#1:用自己生成的数据进行PCA算法处理；2:利用人脸图像数据进行
PCA算法处理
face_dimension = 100#人脸图像要降至的维度
filepath = "C:\\face"#人脸数据图像的具体路径
size = (60, 60)#设置裁剪为统一图像尺寸大小


def generate_data(number, dimension):
    "生成数据"
    data = np.zeros((dimension, number))

    if dimension == 2:#二维数据情况下
        mean = [2, -2]
        cov = [[1, 0], [0, 0.01]]
    elif dimension == 3:#三维数据情况下
        mean = [2, 2, 3]
        cov = [[1, 0, 0], [0, 1, 0], [0, 0, 0.01]]

    data = np.random.multivariate_normal(mean, cov, number)#利用多维高斯分布
生成样本点
    #print(data.shape)#(100,2)

    return data


def PCA(data, new_dimension):
    "实现PCA算法"
    x_mean = np.sum(data, axis = 0) / number#求取原数据均值
    decentral_data = data - x_mean#将散点数据中心化处理
    cov = decentral_data.T @ decentral_data#计算中心化数据的协方差矩阵
    eigenvalues, eigenvectors = np.linalg.eig(cov)#求取特征值和特征向量，再进行
```

```
分解
    eigenvectors = np.real(eigenvectors)#取实部值（去虚部处理）
    dimension_order = np.argsort(eigenvalues)#按照从小到大排序来获得特征值的索引
    PCA_vector = eigenvectors[:, dimension_order[:-(new_dimension + 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#计算经过PCA算法分析之后的x值

    return PCA_vector, x_mean, x_pca#返回特征向量矩阵，降维前数据均值以及中心化数据


def PCA_show():
    "可视化PCA算法结果"
    data = generate_data(number, dimension)#生成数据
    _, _, x_pca = PCA(data, new_dimension)#执行PCA算法

    #绘制散点图
    if dimension == 2:#原样本点维数为2
        plt.scatter(data.T[0], data.T[1], c='w',
                    edgecolors='#008000', s=20, marker='o', label='original data')#原始图像散点为深绿色
        plt.scatter(x_pca.T[0], x_pca.T[1], c='w',
                    edgecolors='#FFD700', s=20, marker='o', label='PCA data')#PCA后图像散点为金色

    elif dimension == 3:#原样本点维数为3
        fig = plt.figure()
        ax = Axes3D(fig)
        ax.scatter(data.T[0], data.T[1], data.T[2],
                   c='#008000', s=20, label='original data')#原始图像散点为深绿色
        ax.scatter(x_pca.T[0], x_pca.T[1], x_pca.T[2],
                   c='#FFD700', s=20, label='PCA data')#PCA后图像散点为金色

    plt.title("number = %d, dimension = %d, new dimension = %d" % (number, dimension, new_dimension))
    plt.legend()#给出图例标签说明
    plt.show()#展示散点图

    return
```

```
def read_face_data():
    "读入人脸数据并且进行展示"
    img_list = os.listdir(filepath)#获取图像文件夹名列表
    data = []
    i = 1#初始化每行的第i个图像

    for img in img_list:
        path = os.path.join(filepath, img)#输入图像
        plt.subplot(3, 3, i)#直接指定划分方式和位置进行绘图，共有3*3张图像
        with open(path) as _:
            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)#RGB
三通道图像转换为灰度图
            plt.imshow(img_gray)#进行图像预览
            h, w = img_gray.shape#设置灰度图的高度和宽度数据
            img_col = img_gray.reshape(h * w)#对(h,w)的图像数据进行拉平
            data.append(img_col)
        i += 1#顺次迭代
    plt.show()#图像展示

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


def PSNR(img_1, img_2):
    "计算图像信噪比"
    diff = (img_1 - img_2) ** 2#进行二阶幂运算
    mse = np.sqrt(np.mean(diff))#求取均值的开方

    return 20 * np.log10(255.0 / mse)#PSNR计算公式


def face_show():
    "人脸数据PCA"
    data = read_face_data()
    n, _ = data.shape
    _, _, x_pca = PCA(data, face_dimension)
    x_pca = np.real(x_pca)#取实部值（去虚部处理）

    # 绘制PCA算法处理后的图像
    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("压缩后的维度为：%d，信噪比如下列各行所示：" % face_dimension)

        for i in range(n):
            psnr = PSNR(data[i], x_pca[i])
            print("图%d，信噪比为: %.3f" % (i + 1, psnr))

        return


#选择实验方案
if method == 1:
    PCA_show()
elif method == 2:
    face_show()
```