哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

2020 年春季学期

计算学部《机器学习》课程

Lab 4 实验报告

姓名	刘璟烁
学号	
班号	1803104
电子邮件	
手机号码	

目录

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

操作系统：MacOS 10.15.7

python 版本：Python 3.8.3

三、设计思想及算法实现

1.算法原理

据李航老师的《统计学习方法》一书中的定义，主成分分析（PCA）指一种常用的无监督学习方法，这一方法利用正交变换把线性相关变量表示的观测数据转换为少数几个由线性无关变量表示的数据，线性无边的变量称为主成分。主成分的个数通常小于原始变量的个数，所以主成分分析属于降维方法。主成分分析有两种等价推导，分别基于最近重构性和最大可分性来进行推导。

以最大可分性的推导为例：

对于一组数据 $X = \{x_1,x_2...,x_n\}$，其中 x_i 为 m 维向量，我们希望对其进行投影变换，通过线性变换 **WX** 得到其在新空间中的超平面上的一组投影 $Y= \{y_1,y_2,...,y_n\}$，其中 y_i 为降维后的 k 维向量。

首先对样本进行中心化，即对数据集 $X = \{x_1,x_2...,x_n\}$ 中的每个样本，减去整个数据集的均值向量：

$$x_i = x_i - \frac{1}{n}\sum_{j=1}^{n} x_j$$

进行中心化后常规的线性变换就是绕原点的旋转变化，也就是坐标变换；样本的协方差矩阵可直接由 $\Sigma = XX^T$ 计算得到。

若使所有样本点的投影尽可能分开，则应该使投影后样本点的方差最大化，投影后样本点的协方差矩阵为：

$$\sum_i W^T x_i x_i^T W$$

则优化目标可以写为：

$$\max_{W} tr(W^T X X^T W)$$

$$\text{s.t. } W^T W = I$$

由于已经对样本进行了中心化，有 $\Sigma = XX^T$ 为数据集 X 的协方差矩阵，则可由上式定义拉格朗日函数：

$$W\Sigma W - \lambda(W^T W - I)$$

其中λ为拉格朗日乘子，将拉格朗日函数对 W 中的每个标准正交基向量求导，得：

$$\Sigma w_i - \lambda w_i = 0$$

于是λ为 Σ 的特征值，w_i 为其对应的单位特征向量，且 $w_i^T w = 1$，则λ为优化目标函数的一个值，则只需要对样本的协方差矩阵 Σ 进行特征分解，找出 Σ 最大的 k 个特征值并生成对应的线性变换矩阵 W 即可。

2.算法实现

下面给出周志华老师的《机器学习》一书中关于 PCA 的伪代码，需要注意的是该书中的实现并没有对样本进行归一化操作，但如果样本各属性间的量纲差距较大，应该对各特征属性进行归一化操作，即每个属性除以样本集上该属性对应的样本方差，本次试验中人脸数据的 RGB 值均在 0-255 之间，故可省略归一化这一步，伪代码如下：

输入：样本集 $X = \{x_1, x_2, ..., x_n\}$；低维空间维数 k

过程：

1.对所有的样本进行中心化操作

a.计算样本均值**μ** $= \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i$;

b.所有样本减去均值 $x_i = x_i - \mu$;

2.计算样本的协方差矩阵 $\Sigma = XX^T$;

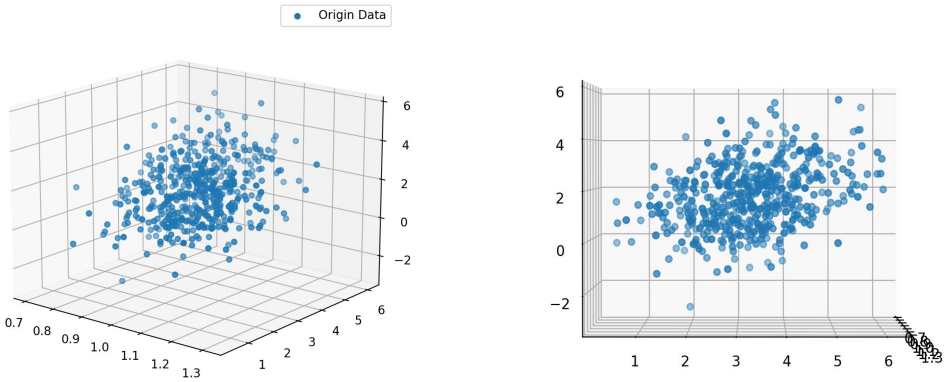
3.对协方差矩阵 Σ 进行特征值分解;

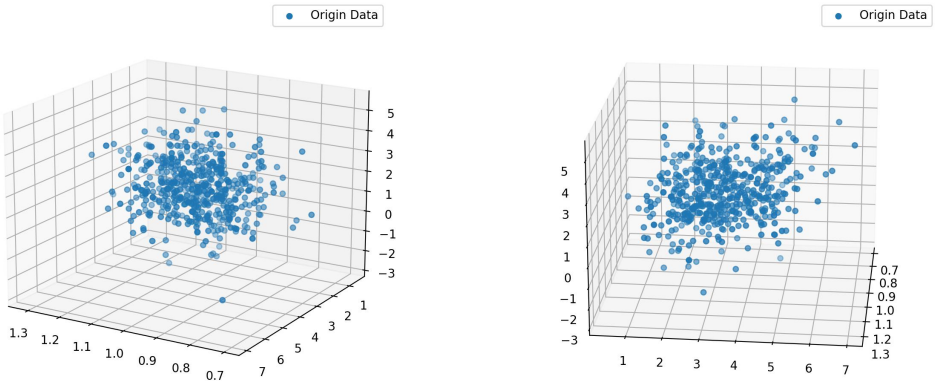
4.取最大的 k 个特征值对应的单位特征向量输出投影矩阵 W 与样本均值**μ**。

四、实验结果分析

1.数据生成

本次实验手动生成了若干(默认为 500)个服从多元正态分布的三维空间中的数据点，取其中一个维度的方差为 0.01，其他分别为 1 和 2，则第一个维度的点均集中在对应坐标平面附近。随后对数据点进行了 30 度角旋转（与对应的线性变换矩阵作矩阵乘法即可），具体代码请见方法 generate_data()，如下为某两次次随机生成的样本点：

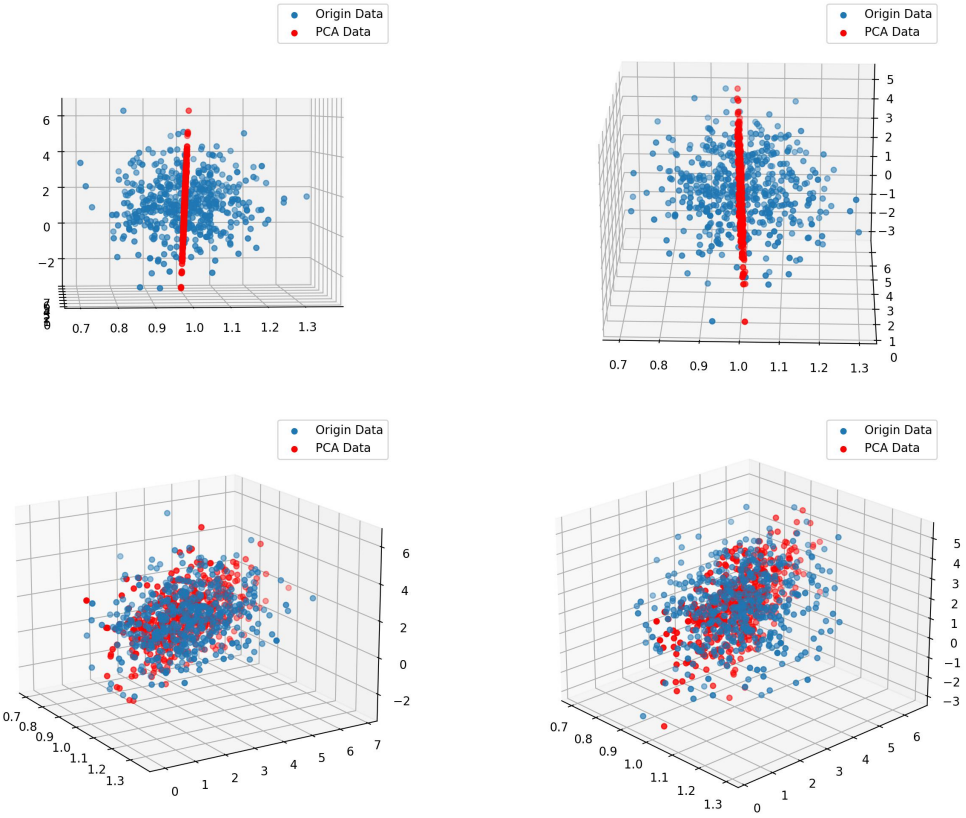




可以看到样本点经过了一定角度旋转,并在三个维度中有不同的长度计量。

2.PCA 模型在手动数据集上的效果

对于上面手动生成的数据集合，我们据算法原理部分所述的方法使用 PCA 模型求取其对应的线性变换矩阵 W，将三维数据样本点进行线性变换并映射到二维空间中去，得到点集 Y；随后对 Y 中的点进行重构，展示其在三维空间中分布情况如下：



可以看到，数据集方差较小的一维被映射、重构到一个平面上，从而实现了降维（主成分抽取）的目的。

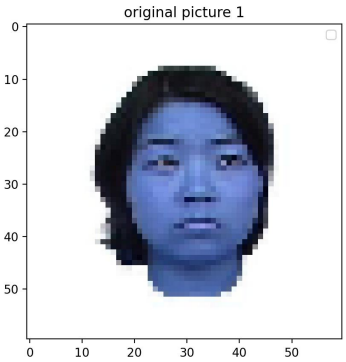
3.PCA 模型在人脸数据集上的效果

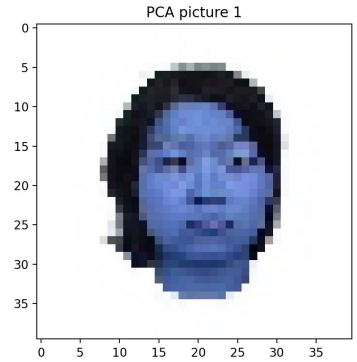
本次实验应用了 CMU 提供的人脸数据集，资源 url 请见参考资料部分。这里我们节选了亚洲人脸照片中的十张已经经过压缩（方便训练）进行 PCA 训练，即找出图片的主成分，并用这些主成分重新对十张人脸图像进行降维、重构，观察图像的变化情况，我们用峰值信噪比 PSNR 作为信噪比的度量方法：

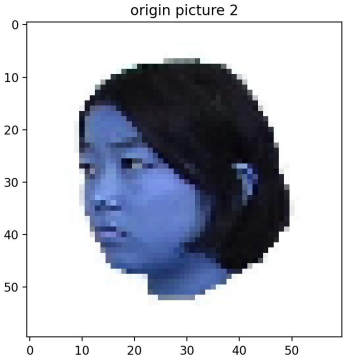
$$PSNR = 20\log_{10}(\frac{255}{\sum_{i=1}^{n}||f_i - g_i||})$$

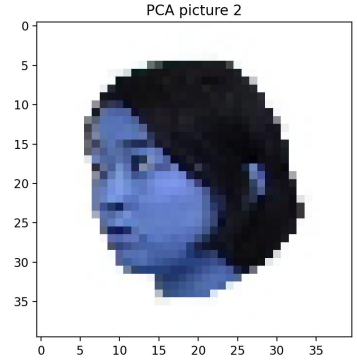
其中 f 和 g 为图像处理前后对应的特征向量，则对与降至的维数 k 分别取 1、5、10、50，第一张图片的变化和十张图片的信噪比如下：

k 取 50，即降至 50 维，左为原始图片，右为处理后的图片：





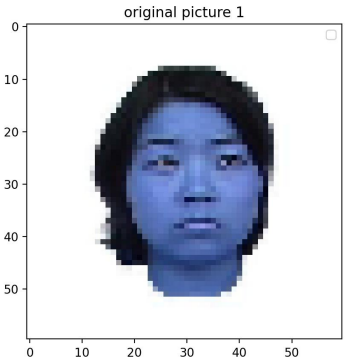


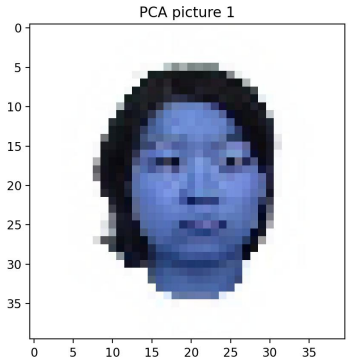


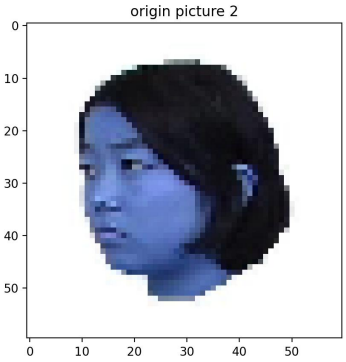
信噪比为：

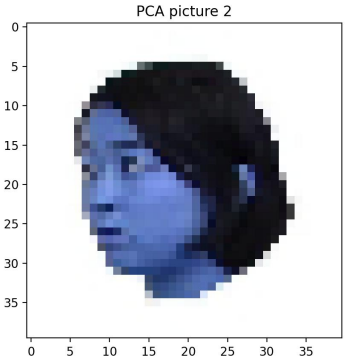
```
第1张图片的信噪比为：53.691657686629675
第2张图片的信噪比为：54.2799567288146
第3张图片的信噪比为：54.01250291867434
第4张图片的信噪比为：53.86433573060699
第5张图片的信噪比为：54.58092054157203
第6张图片的信噪比为：54.67379970284468
第7张图片的信噪比为：54.05123231438348
第8张图片的信噪比为：54.06185509542946 5
第9张图片的信噪比为：54.474364074057874
第10张图片的信噪比为：53.75395845417721
```

k 取 10，即降至 10 维，左为原始图片，右为处理后的图片，可以发现与降维至 50 分别较小：





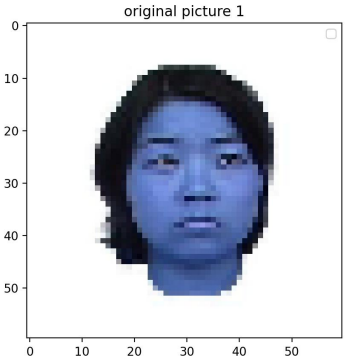


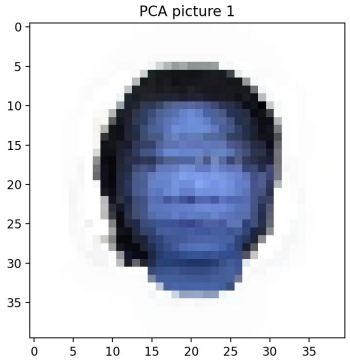


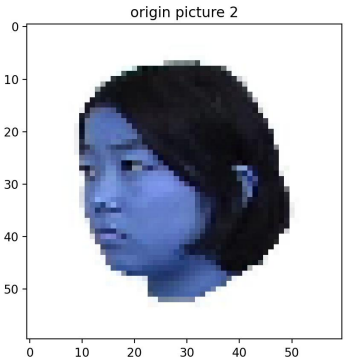
信噪比为：

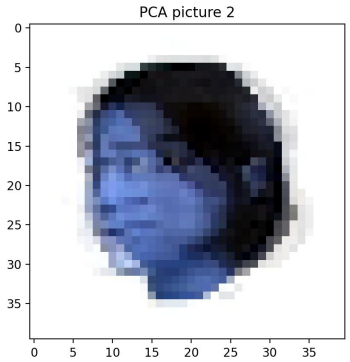
```
第1张图片的信噪比为：53.70469957276412
第2张图片的信噪比为：54.564968476551556
第3张图片的信噪比为：54.01600953559399
第4张图片的信噪比为：53.83395355977078
第5张图片的信噪比为：54.592923160411296
第6张图片的信噪比为：54.592923160411296
第7张图片的信噪比为：54.191397436248O5
第8张图片的信噪比为：54.173172714501476
第9张图片的信噪比为：54.592923160411296
第10张图片的信噪比为：53.830590845844 32
```

k 取 5，即降至 5 维，左为原始图片，右为处理后的图片，可以看出处理后的图像仍有大致轮廓：





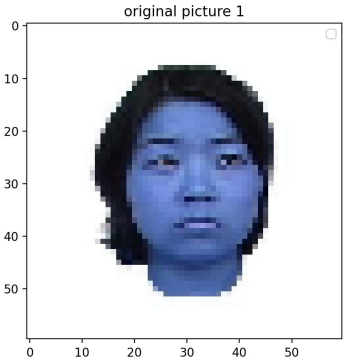


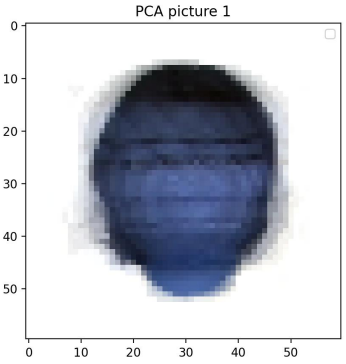


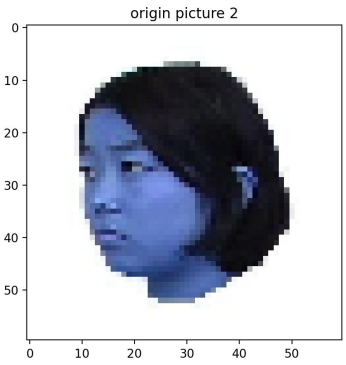
信噪比为：

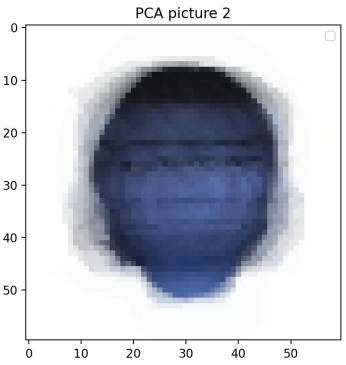
```
第1张图片的信噪比为：22.789539998099176
第2张图片的信噪比为：22.195928423402695
第3张图片的信噪比为：22.328305765642014
第4张图片的信噪比为：25.07604042235153
第5张图片的信噪比为：24.353519072333846
第6张图片的信噪比为：52.446341704381965
第7张图片的信噪比为：21.345028052698733
第8张图片的信噪比为：21.761630630024293
第9张图片的信噪比为：30.385740397424055
第10张图片的信噪比为：21.940625076454793
```

k 取 1，即降至 1 维，左为原始图片，右为处理后的图片,轮廓已经不再分明：









信噪比为：

```
第1张图片的信噪比为：17.34190853594932
第2张图片的信噪比为：15.180598170325945
第3张图片的信噪比为：17.772237096282357
第4张图片的信噪比为：15.892403828088176
第5张图片的信噪比为：14.387388601476767
第6张图片的信噪比为：18.098848809590116
第7张图片的信噪比为：16.574781591797244
第8张图片的信噪比为：16.140064161910267
第9张图片的信噪比为：13.640004083856441
第10张图片的信噪比为：15.379309849796039
```

可以看出随着维数 k 的减少，图片的信噪比降低。

五、结论

根据上述实验结果以及算法原理的分析，我们可以得出以下结论：

1. PCA 算法中选取了样本协方差矩阵最大 k 个特征值对应的特征向量所形成的线性变换矩阵，从而尽可能保留样本原有的信息而又达到降维减少运算量的目的。

2.PCA 降维处理后的样本信息往往相对处理前含有更少的噪声。

3.通过图像的信噪比计算公式，我们可以观察到，随着 PCA 降维至的维数 k 的减少，图片的信噪比将降低，图片逐渐模糊。

六、参考资料

吴恩达：机器学习

Christopher Bishop. Pattern Recognition and Machine Learning.

周志华：机器学习

CMU Face Place.

七、源代码

1.main.py:

该部分为 client 端，可以分别通过调用 hand_gene_samples()、

asian_face_pca()方法观察 PCA 模型分别在手动生辰和人脸数据集上的效果：

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


def PCA(X,reduce_to_dim):
    """
    PCA 降维
    :param X: 输入数据(n,m)
    :param reduce_to_dim: 要降低到的维度
    :return: W,mu 线性变换阵，以及数据规范化前均值
    """
    size = X.shape[0]
    mu = np.array([1 / size * np.sum(X, axis=0)])
    var_x = 1 / (size - 1) * np.sum((X - mu)*(X - mu), axis=0)  # 样本
    X = (X - mu)  # 标准化
    R = 1 / (size - 1) * np.dot(X.T, X)  # (m,m)相关矩阵
    values, feat_vectors = np.linalg.eig(R)
    sorted_indices = np.argsort(values)
    a = sorted_indices[:-reduce_to_dim-1:-1]
```

```
    W = feat_vectors[:,a]
    return W,mu,var_x**(1/2)


def generate_data(data_dimension = 3, number=500):
    """ 生成 3 维数据 """
    mean = [1, 2, 3]
    cov = [[0.01, 0, 0], [0, 1, 0], [0, 0, 2]]
    sample_data = []
    for index in range(number):
        sample_data.append(np.random.multivariate_normal(mean,\
cov).tolist())
    X = np.array(sample_data)
    X[:,1:3] = X[:,1:3].dot(np.array([[3**(1/2)/2,-1./2],[1./2,3**(1/2)/2]]))
# 对数据进行旋转
    return X


def draw_data(dimension, origin_data, pca_data):
    """ 将 PCA 前后的数据进行可视化对比 """
    # 绘制 3 维框图
    fig = plt.figure()
    ax = Axes3D(fig)
    ax.scatter(origin_data[:, 0], origin_data[:, 1], origin_data[:, 2],
```

```
                    label='Origin Data')
    # ax.scatter(pca_data[:, 0], pca_data[:, 1], pca_data[:, 2], color='r',
label='PCA Data')
    # ax.zaxis.set_major_locator(plt.MultipleLocator(1))
    # # ax.xaxis.set_major_locator(plt.MultipleLocator(1))
    # ax.yaxis.set_major_locator(plt.MultipleLocator(1))
    # plt.xlim(-5,5)
    # plt.ylim(-5,5)
    plt.title("PCA Model")
    plt.legend()
    plt.show()


def hand_gene_test():
    """
    对人工生成数据进行 PCA 降维，并展示
    :return:
    """
    data = generate_data(3)
    W, mu, standard_div =PCA(data,2)   # 返回由训练集生成的的变换矩
阵，以及训练集的均值、标准差
    print("生成的线性变换矩阵为：")
    print(W)
```

```
    pca_data = np.dot(data - mu,W)
    refac_data = np.dot(pca_data,W.T) + mu   # 对降维的数据进行重构
    draw_data(3, data, refac_data)
    return


def read_fac_image(file_path):
    face_list = os.listdir(file_path)   # 从该目录中读取文件名列表
    data = []
    for face_file in face_list:   # 处理每一张人脸图片
        this_path = os.path.join(file_path, face_file)
        with open(this_path) as f:
            face_img = cv2.imread(this_path)   # 利用 opencv 读取图片
            size = (40,40)   # 压缩图片 加快处理速度
            face_img = cv2.resize(face_img, size)
            row,col,z = face_img.shape
            face_img = face_img.reshape(row * col * z)
            data.append(face_img)
    data = np.array(data)
    return data



def asian_face_pca(reduce_to):
```

```
    """
    读取人脸数据集
    :param reduce_to: 希望降低到的维度
    :return:
    """

    data = read_fac_image("/Users/VitoLiu/Downloads/asia")  # 从该文件夹读取人脸照片 共含有 10 张人脸照片
    W, mu, standard_div = PCA(data, reduce_to)  # 返回由数据集生成的的变换矩阵，以及训练集的均值、标准差
    W = np.real(W)  # 如果产生复数均转化为实数
    pca_data = np.dot(data - mu, W)  # 降维后的数据
    refac_data = np.dot(pca_data, W.T) + mu  # 对降维的数据进行重构
    refac_data = refac_data.astype(int)

    # 显示每一张降维后的人脸照片
    j = 0
    for i in range(refac_data.shape[0]):
        psnr = calc_psnr(data[i],refac_data[i])
        print("第"+ str(i+1)+"张图片的信噪比为：",end='')
        print(psnr)
        if j < 2:
```

```
                plt.imshow(refac_data[i].reshape(40,40,3))
                plt.title("PCA picture "+ str(j +1))
                plt.show()
                j += 1
    return


def calc_psnr(formal_img, pca_img):
    mse = np.mean((formal_img - pca_img) ** 2)
    psnr = 20 * np.log10(255 / (mse)**(1/2))
    return psnr



# hand_gene_test()   # 观察手动生成数据集的效果
asian_face_pca(10)   # 观察人脸数据集的效果
```