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

学号：1190201018
姓名： 李昆泽

Lab4-Report

1. 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）。

2. 实验要求及实验环境

实验要求

- 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。
- 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：Win 10

Python 3.8

3. 设计思想

1. 算法原理

主成分分析（principal component analysis, PCA）是一种常用的无监督学习方法，一般用来将数据降维，将高维的数据转换为由少数几个线性无关变量表示的数据，这些线性无关的变量即称为主成分。

一般有两种关于PCA的推导，分别是从最大方差形式和最小误差形式两种方面进行的。由于这两种形式实质上等价，本次报告从最小误差形式的角度推导算法原理。

考虑一个数据集 $\{x_1, \ldots, x_N\}, x_n \in R^D$ ，我们的目标是把这些数据投影到一个 $M(M < D)$ 维的空间中。

引入 D 维单位正交基集合 $\{u_1, \ldots, u_D\}$ ，且满足

$$u_i^T u_j = \delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

这时，每个数据点均可以被精确地表示为基向量的线性组合，即

$$x_n = \sum_{i=1}^{D} \alpha_{ni} u_i, \quad \alpha_{ni} = x_n^T u_i$$

我们的目标时是使用 $M(M < D)$ 维的空间来近似表示原数据点，不失一般性，我们用前 M 个基向量来表示

$$\tilde{x}_n = \sum_{i=1}^{M} z_{ni} u_i + \sum_{i=M+1}^{D} b_i u_i$$

其中 z_{ni} 依赖于数据点， b_i 是常数。

我们的目标是最小化误差

$$J = \frac{1}{N} \sum_{n=1}^{N} ||x_n - \tilde{x}_n||^2$$

对上式展开得

$$\begin{aligned}
J &= \frac{1}{N} \sum_{n=1}^{N} ||x_n - \tilde{x}_n||^2 \\
&= \frac{1}{N} \sum_{n=1}^{N} (x_n - \tilde{x}_n)^T (x_n - \tilde{x}_n) \\
&= \frac{1}{N} \sum_{n=1}^{N} (x_n - \sum_{i=1}^{M} z_{ni} u_i - \sum_{i=M+1}^{D} b_i u_i)^T (x_n - \sum_{i=1}^{M} z_{ni} u_i - \sum_{i=M+1}^{D} b_i u_i) \\
&= \frac{1}{N} \sum_{n=1}^{N} (x_n^T - \sum_{i=1}^{M} z_{ni} u_i^T - \sum_{i=M+1}^{D} b_i u_i^T)(x_n - \sum_{i=1}^{M} z_{ni} u_i - \sum_{i=M+1}^{D} b_i u_i) \\
&= \frac{1}{N} \sum_{n=1}^{N} (x_n^T x_n - 2 \sum_{i=1}^{M} z_{ni} x_n^T u_i - 2 \sum_{i=M+1}^{D} b_i x_n^T u_i + \sum_{i=1}^{M} z_{ni}^2 + \sum_{i=M+1}^{D} b_i^2)
\end{aligned}$$

J 对 z_{ni} 和 b_i 分别求偏导得

$$\frac{\partial J}{\partial z_{ni}} = \frac{1}{N} (-2 x_n^T u_i + 2 z_{ni}) = 0$$
$$\frac{\partial J}{\partial b_i} = \frac{1}{N} \sum_{n=1}^{N} (-2 x_n^T u_i + 2 b_i) = 0$$

可得

$$z_{ni} = x_n^T u_i$$
$$b_i = \bar{x}^T u_i$$

又由于 $\alpha_{ni} = z_{ni}, i = 1, \ldots, M$ ，因此

$$
\begin{aligned}
x_n - \tilde{x}_n &= \sum_{i=1}^{D} \alpha_{ni} u_i - (\sum_{i=1}^{M} z_{ni} u_i + \sum_{i=M+1}^{D} b_i u_i) \\
&= \sum_{i=M+1}^{D} (\alpha_{ni} - b_i) u_i
\end{aligned}
$$

从而（注意到 $u_i^T u_j = 1 \iff i = j$ ）

$$
\begin{aligned}
||x_n - \tilde{x}_n||^2 &= (x_n - \tilde{x}_n)^T (x_n - \tilde{x}_n) \\
&= \sum_{i=M+1}^{D} (\alpha_{ni} - b_i) u_i^T \cdot \sum_{j=M+1}^{D} (\alpha_{nj} - b_j) u_j \\
&= \sum_{i=M+1}^{D} (\alpha_{ni} - b_i)^2 \\
&= \sum_{i=M+1}^{D} (x_n^T u_i - \bar{x}^T u_i)^2
\end{aligned}
$$

代入得

$$
\begin{aligned}
J &= \frac{1}{N} \sum_{n=1}^{N} ||x_n - \tilde{x}_n||^2 \\
&= \frac{1}{N} \sum_{n=1}^{N} \sum_{i=M+1}^{D} (x_n^T u_i - \bar{x}^T u_i)^2 = \sum_{i=M+1}^{D} u_i^T S u_i
\end{aligned}
$$

而 $S u_i = \lambda_i u_i$ ，故

$$
J = \sum_{i=M+1}^{D} \lambda_i
$$

最小化 J 即选择 $D - M$ 个最小特征值对应的特征向量。

2. 算法实现

给定样本集 $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ 和低维空间的维数 m ：

1. 对所有的样本进行中心化操作：
    - 计算样本均值 $\mu = \frac{1}{m} \sum_{j=1}^{m} \mathbf{x}_j$
    - 所有样本减去均值 $\mathbf{x}_j = \mathbf{x}_j - \mu, \ j \in 1, 2, \ldots, m$
2. 计算样本的协方差矩阵 $\Sigma = \mathbf{X}^T \mathbf{X}$
3. 求出协方差矩阵 Σ 的特征值和特征向量

4. 若要降到 d 维，则取最大的 d 个特征值 $\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_d$ 对应的单位特征向量 $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_d$ ，得到投影矩阵 $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_d)$
5. 降维后数据是 d 维，若要可视化，需要将数据转换到原来 m 维的坐标系下，转换公式如下：

$$\tilde{X} = XWW^T$$

4. 实验结果及分析

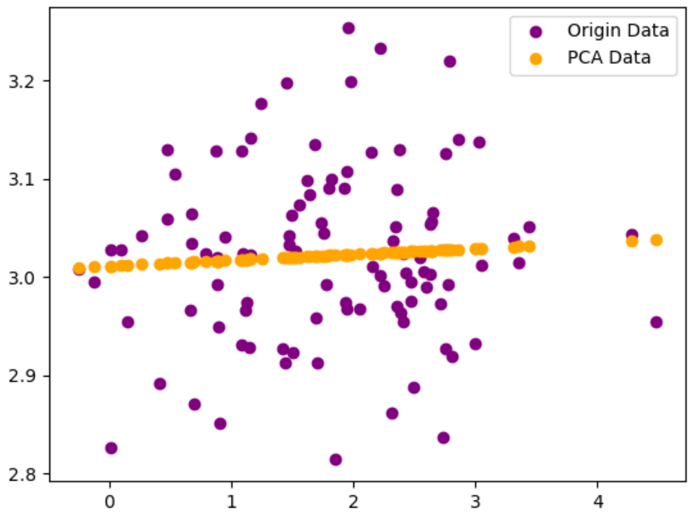
1. 手工生成数据测试

为了方便进行数据可视化，分别手工生成服从二维和三维高斯分布的数据以供实验。为了使得降维PCA的“最大可分性”效果明显(降到的维度方差大)，生成数据的时候让其中一维的方差明显比其他维度的小(其中一维的方差为0.01，其他的都为1)。手工生成数据代码如下：

Python

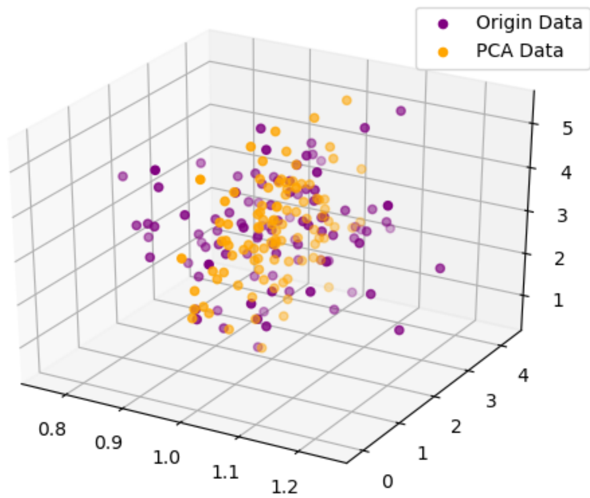
```python
def generate_data(dimension, number):
    if dimension is 2:
        mean = [2, 3]
        cov = [[1, 0], [0, 0.01]]
    elif dimension is 3:
        mean = [1, 2, 3]
        cov = [[0.01, 0, 0], [0, 1, 0], [0, 0, 1]]
    else:
        assert False
    sample_data = []
    for index in range(number):
        sample_data.append(np.random.multivariate_normal(mean, cov).tolist())
    return np.array(sample_data)
```

对于二维数据，其协方差矩阵为 $\begin{bmatrix} 0.01 & 0 \\ 0 & 1 \end{bmatrix}$ ，将其降成一维，降维结果如下(注意这里dim0和dim1的单位刻度不同，dim1的方差实际上是很小的)：

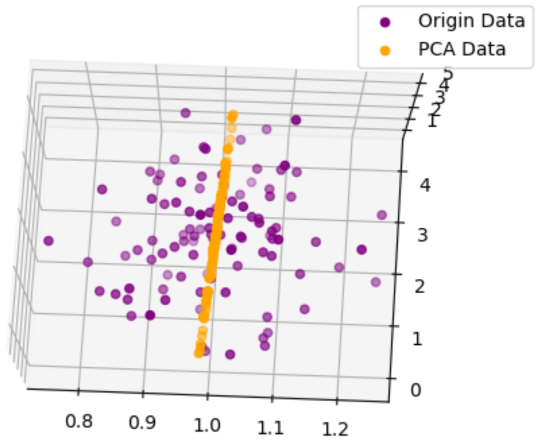


通过协方差矩阵可以得知，第一维的方差远小于第二维的方差(0.01<<1)，即第二维包含了更多的信息。从实验结果也可以明显看到，PCA将数据降到了几乎和dim0相同的方向，这也印证了PCA算法的"最大可分性"，即将数据降维到方差大的方向。

对于三维数据，方差为 $\begin{bmatrix} 0.01 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ ，将其降成两维，得到结果如下：



这样看并不是很明显，将其拖动换一个角度，即可明显的看出投影后的点是一个平面，且由于第0维的方差是远小于其余两个维度的，所以第0维相较于其他两维信息更少，故这个平面是dim1和dim2的方向(即数据的方差大的两个方向)：



总结：

综上，实现的PCA降维算法的效果还是不错的，通过将某一维方差设置的极小，也验证了PCA算法的最大可分性(降到方差最大的维度)。

2. 人脸数据

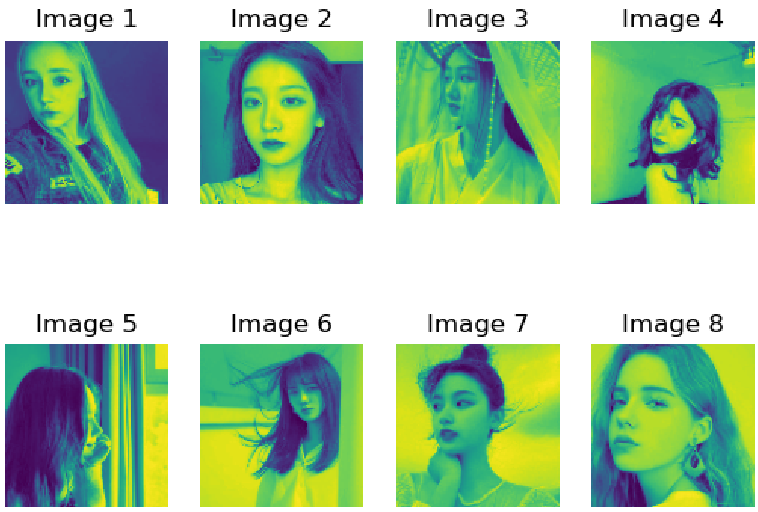
要用信噪比来衡量降维后图像和原图像的差别，信噪比公式如下：

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

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

从网上选取了8张人脸数据进行PCA降维实验。原图像素较高，在实验中出现了样本维度过高，方差矩阵过大，求解特征值非常缓慢的问题。因此，调用cv2包，将原图像压缩至 size=(100,100)大小，让运行速度更快。

8张人脸图如下，原图是三通道RGB的图像，维度过高，求解过慢，故将其转换为单通道的灰度图。但这里又没有采用黑白双色的表示，那样的图片太恐怖且不美观，采用这样的单通道稍微好一点(只是绿色偏色严重)：

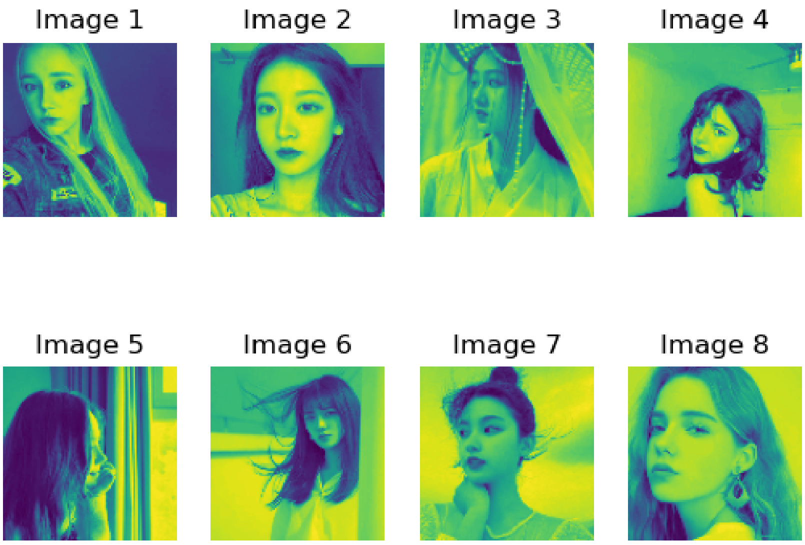


考虑对整组图片进行降维，将它们按列拼接在一起，作为一个整体进行PCA分析。下面进行降维，分别降至50、20、10、5、3、1维。先展示实验效果：

降至50维：

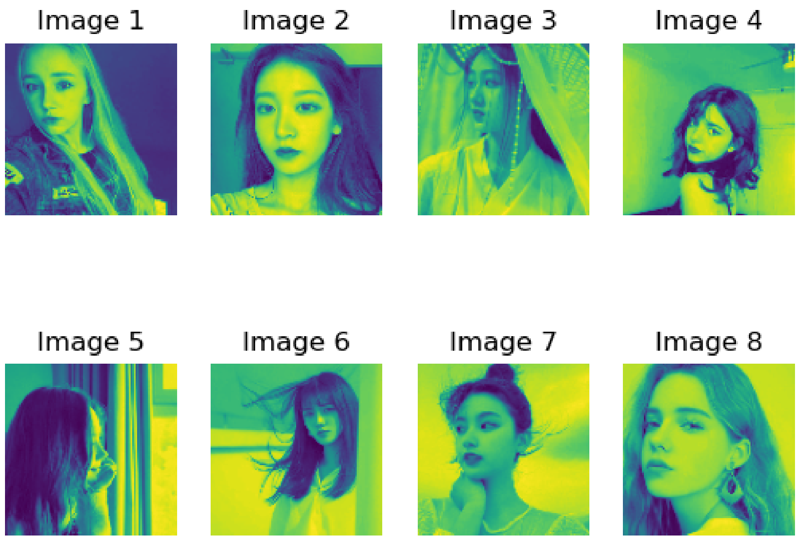
```
Image 1 PSNR: 312.1643225765052
Image 2 PSNR: 311.677812313108
Image 3 PSNR: 311.42599145969774
Image 4 PSNR: 312.3752221767646
Image 5 PSNR: 312.6516439017562
Image 6 PSNR: 313.8285834088318
Image 7 PSNR: 308.66203269340195
Image 8 PSNR: 310.8640386310304
```

降至20维：



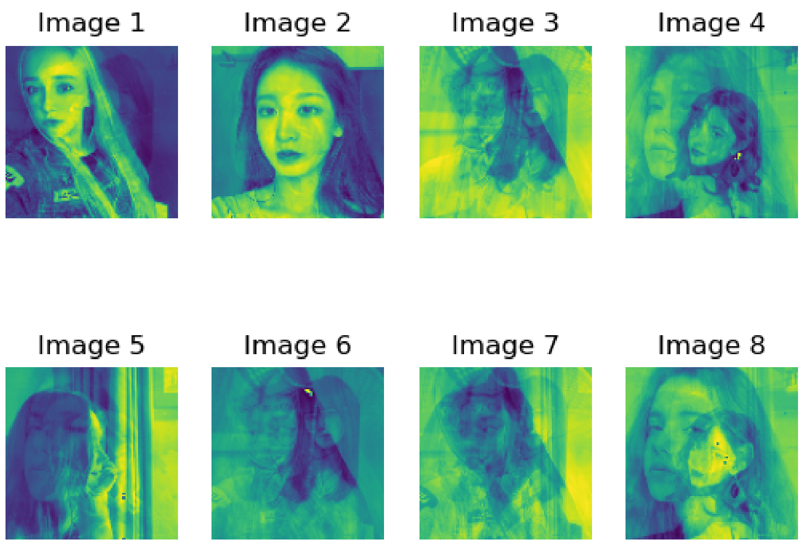

```
Image 1 PSNR: 312.1643225765052
Image 2 PSNR: 311.677812313108
Image 3 PSNR: 311.42599145969774
Image 4 PSNR: 312.3752221767646
Image 5 PSNR: 312.6516439017562
Image 6 PSNR: 313.8285834088318
Image 7 PSNR: 308.66203269340195
Image 8 PSNR: 310.8640386310304
```

降至10维：



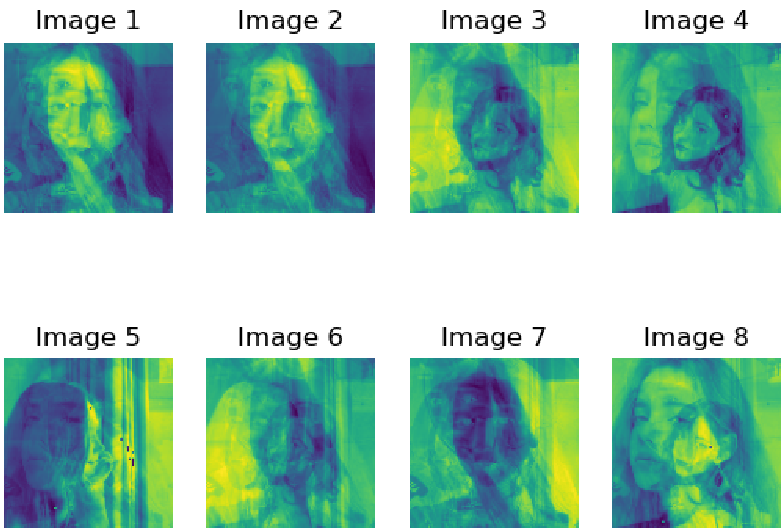

```
Image 1 PSNR:  312.1643225765052
Image 2 PSNR:  311.677812313108
Image 3 PSNR:  311.42599145969774
Image 4 PSNR:  312.3752221767646
Image 5 PSNR:  312.6516439017562
Image 6 PSNR:  313.8285834088318
Image 7 PSNR:  308.66203269340195
Image 8 PSNR:  310.8640386310304
```

降至5维：



```
Image 1 PSNR: 25.982468336499963
Image 2 PSNR: 29.262656544365527
Image 3 PSNR: 20.217560646958773
Image 4 PSNR: 19.479827308517482
Image 5 PSNR: 24.312961662958912
Image 6 PSNR: 20.601957378886986
Image 7 PSNR: 18.311290515067892
Image 8 PSNR: 19.47789376999191
```

降至3维：




```
Image 1 PSNR: 15.893324056880246
Image 2 PSNR: 15.791775405287376
Image 3 PSNR: 17.41029268614373
Image 4 PSNR: 18.933764013901584
Image 5 PSNR: 21.630632529850665
Image 6 PSNR: 16.58693105641641
Image 7 PSNR: 17.260696783719183
Image 8 PSNR: 19.234380552794136
```

从图像直观地可以看出，随着维度的降低，保留的特征少了，重构图像越来越不相似，美女不再美了。从信噪比也可以看出这一点，随着维度的降低，信噪比也越来越低，保留的信息量越来越少。

5. 结论

1. PCA降低了训练数据的维度的同时保留了主要信息，但在训练集上的主要信息未必是重要信息，被舍弃掉的信息未必无用，只是在训练数据上没有表现，因此PCA也有可能加重了过拟合。

2. PCA算法中舍弃了 $n-d$ 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可以用于图片的降维，从而极大地缓解存储压力。例如原本有1000个512*512的图片，通过PCA降维，可以将图片降成50维，图像信息还保留的较好，但存储空间节约了近5000倍！

6. 参考文献

[1] 周志华 著. 机器学习, 北京: 清华大学出版社, 2016.1

[2] 李航 著. 统计学习方法, 北京: 清华大学出版社, 2019.5

[3] 知乎：如何理解矩阵特征值 https://www.zhihu.com/question/21874816/answer/181864044

[4] CSDN：图像的峰值信噪比 https://blog.csdn.net/xrinosvip/article/details/88569111

7. 附录：源代码

Python

```python
import numpy as np
import cv2
from matplotlib import pyplot as plt

# 超参数
DIMENSION = 3
SAMPLE_SIZE = 100

FACE_NUM = 8
WIDTH = 100
HEIGHT = 100
CHANNEL = 1
REDUCED_DIMENSION = 10


# 生成数据
def generate_data(dimension, number):
    if dimension is 2:
        mean = [2, 3]
        cov = [[1, 0], [0, 0.01]]
    elif dimension is 3:
        mean = [1, 2, 3]
        cov = [[0.01, 0, 0], [0, 1, 0], [0, 0, 1]]
    else:
        assert False
    sample_data = []
```

```python
    for index in range(number):
        sample_data.append(np.random.multivariate_normal(mean, cov).tolist())
    return np.array(sample_data)


# 去中心化
def decentralise(data):
    return data - np.mean(data, axis=0)


# pca恢复
def pca_resume(data, feature_vectors):
    return decentralise(data).dot(feature_vectors).dot(feature_vectors.T) + np.mean(data, axis=0)


def pca(data, reduced_dimension):
    """
    主成分分析
    """
    data = np.float32(np.mat(data))
    # 数据去中心化
    decentralise_x = decentralise(data)
    # 计算协方差矩阵
    cov = np.cov(decentralise_x, rowvar=0)
    # 特征值分解
    eigenvalues, feature_vectors = np.linalg.eig(cov)
    # 选取最大的特征值对应的特征向量
    _min_index = np.argsort(eigenvalues)
    feature_vectors = feature_vectors[:, _min_index[-1:-(reduced_dimension + 1):-1, ]]

    return pca_resume(data, feature_vectors)


def psnr(source, target):
    MSE = np.mean(np.square(source-target))
    PSNR = 20*np.log10(255.0/np.sqrt(MSE))
    return PSNR


def draw_data(dimension, origin_data, pca_data):
    if dimension is 2:
        plt.scatter(origin_data[:, 0], origin_data[:, 1], color="purple", label="Origin Data")
        plt.scatter(pca_data[:, 0].tolist(), pca_data[:, 1].tolist(), color='orange', label='PCA Data')
    elif dimension is 3:
```

```python
    elif dimension is 3:
        fig = plt.figure()
        ax = fig.gca(projection='3d')
        ax.scatter(origin_data[:, 0], origin_data[:, 1], origin_data[:, 2],
                   color="purple", label='Origin Data')
        ax.scatter(pca_data[:, 0], pca_data[:, 1], pca_data[:, 2], color='orange', label='PCA Data')
    else:
        assert False
    plt.legend()
    plt.show()


def load_faces(path, number):
    image = cv2.imread(path + str(number) + '.jpg')
    image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    image = cv2.resize(image, (WIDTH, HEIGHT))
    return image


def show_faces(data):
    for i in range(len(data)):
        plt.subplot(2, 4, i+1), plt.imshow(data[i]), plt.title(f'Image {i+1}')
        plt.axis('off')
    plt.show()


# 生成数据测试
X = generate_data(DIMENSION, SAMPLE_SIZE)
X_PCA = pca(X, DIMENSION-1)
draw_data(DIMENSION, X, X_PCA)

# 人脸数据测试
path = './faces/'
_PCA_faces = []
faces = np.array(load_faces(path, 1)).reshape((-1, 1))
for i in range(1, FACE_NUM):
    face = np.array(load_faces(path, i+1)).reshape((-1, 1))
    faces = np.column_stack((faces, face))

PCA_faces = pca(faces, REDUCED_DIMENSION)
PCA_faces = np.real(PCA_faces)
for i in range(FACE_NUM):
    print(f"Image {i + 1} PSNR: {psnr(PCA_faces[:, i].reshape((HEIGHT, WIDTH)), faces[:, i].reshape((HEIGHT, WIDTH)))}")
    _PCA_faces.append(PCA_faces[:, i].reshape((HEIGHT, WIDTH)))
show_faces(np.array(_PCA_faces, dtype='uint8'))
```