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：必修
实验题目：PCA模型实验

学号：1181000420
姓名：韦昆杰

一、实验目的

目标：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）

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）。

二、实验要求及实验环境

实验环境：

- Windows10
- PyCharm
- VSCode

三、设计思想(本程序中的用到的主要算法及数据结构)

1. 算法原理

主成分分析(Principal Component Analysis,简称PCA)是最常用的一种降维方法。关于PCA的推导，我们可以分别基于最近重构性和最大可分性推导。

> 如果超平面可以对正交属性空间的所有样本进行恰当表达，应该满足下面两个性质
>
> - 最近重构性：样本点到这个超平面的距离都足够近；
> - 最大可分性：样本点在这个超平面上的投影尽可能分开.

1.1最近重构性

假定数据样本进行了中心化，即$\sum_i x_i = 0$；再假定投影变换后得到的新坐标系为$w_1, w_2, \ldots, w_d$，样本点x_i在低维坐标系中的投影为$z_i = (z_{i1}, z_{i2}, \cdots, z_{id'})$，若基于$z_i$来重构$x_i$，则会得到$\hat{x}_i = \sum_{j=1}^{d'} z_{ij} w_j$

考虑整个训练集，原样本点x_i与投影后的样本点$\hat{x}_i$之间的距离为

$$
\begin{aligned}
\sum_{i=1}^{m}\left\|\sum_{j=1}^{d'} z_{ij}\boldsymbol{w}_j-\boldsymbol{x}_i\right\|_2^2 &= \sum_{i=1}^{m}\boldsymbol{z}_i{}^{\mathrm{T}}\boldsymbol{z}_i-2\sum_{i=1}^{m}\boldsymbol{z}_i{}^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}\boldsymbol{x}_i+\ \mathrm{const} \\
&\propto -\operatorname{tr}\left(\mathbf{W}^{\mathrm{T}}\left(\sum_{i=1}^{m} x_i\boldsymbol{x}_i^{\mathrm{T}}\right)\mathbf{W}\right)
\end{aligned} \tag{1}
$$

根据最近重构性，上式应被最小，有

$$
\begin{aligned}
&\min_{\mathbf{W}} -\operatorname{tr}\left(\mathbf{W}^{\mathrm{T}}\mathbf{X}\mathbf{X}^{\mathrm{T}}\mathbf{W}\right) \\
&\text{s.t. } \mathbf{W}^{\mathrm{T}}\mathbf{W}=\mathbf{I}
\end{aligned} \tag{2}
$$

这就是主成分分析的优化目标

1.2最大可分性

样本点$\mathbf{x}_i$降维后在新空间中超平面上的投影为$\mathbf{W}^T\mathbf{x}_i$，若所有样本点的投影能尽可能分开，则应该使样本点在投影后的方差最大化，即使下式最大化：

$$
\begin{aligned}
\arg\max_{\mathbf{W}} &= \arg\max_{\mathbf{W}}\sum_{i=1}^{m}\mathbf{W}^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{W} \\
&= \arg\max_{\mathbf{W}} tr(\mathbf{W}^T\mathbf{X}\mathbf{X}^T\mathbf{W}) \\
\textbf{s.t. } & \mathbf{W}^T W=\mathbf{I}
\end{aligned} \tag{3}
$$

可以看到式(2)与(3)等价，PCA的优化问题就是要求解协方差矩阵$\mathbf{X}^T\mathbf{X}$的特征值。

因此，只需对$\mathbf{X}^T\mathbf{X}$进行特征值分解，将求得的特征值排序：$\lambda_1\geq\lambda_2\geq\cdots\geq\lambda_d$，提取前$d'$大的特征值对应的特征向量即为PCA的解。

2. 算法实现

输入：样本数据集$D=x_1,x_2,\cdots,x_m$；低维空间维数d'

过程：

1. 对所有样本进行中心化
2. 计算样本的协方差矩阵$\mathbf{X}^T\mathbf{X}$
3. 对协方差矩阵$\mathbf{X}^T\mathbf{X}$做特征值分解
4. 取最大的d'个特征值所对应的特征向量$w_1,w_2,...,w_{d'}$

输出：投影矩阵$W_*=(w_1,w_2,...,w_{d'})$

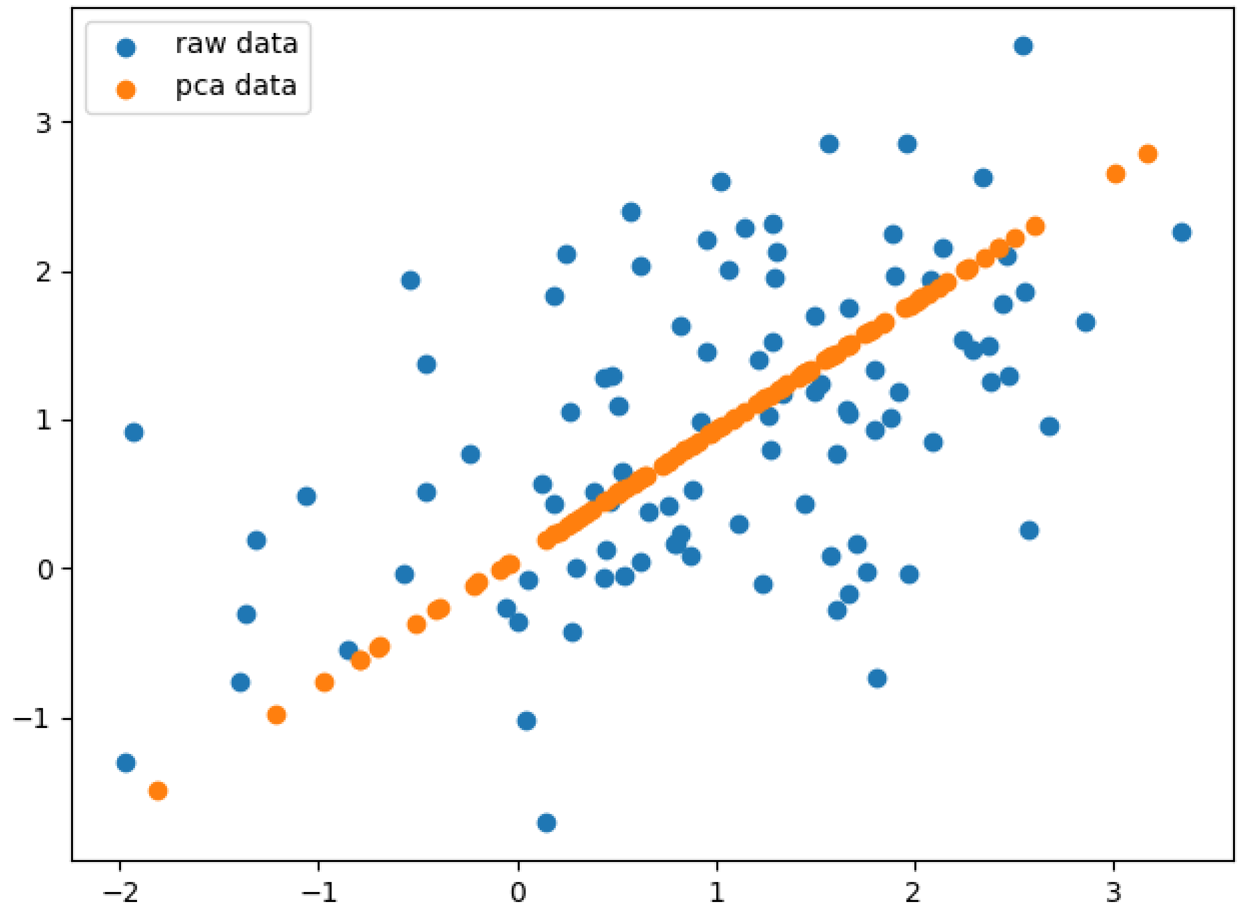
四、实验结果分析

1.生成数据的PCA

使用2维高斯分布产生样本，使用的参数为：

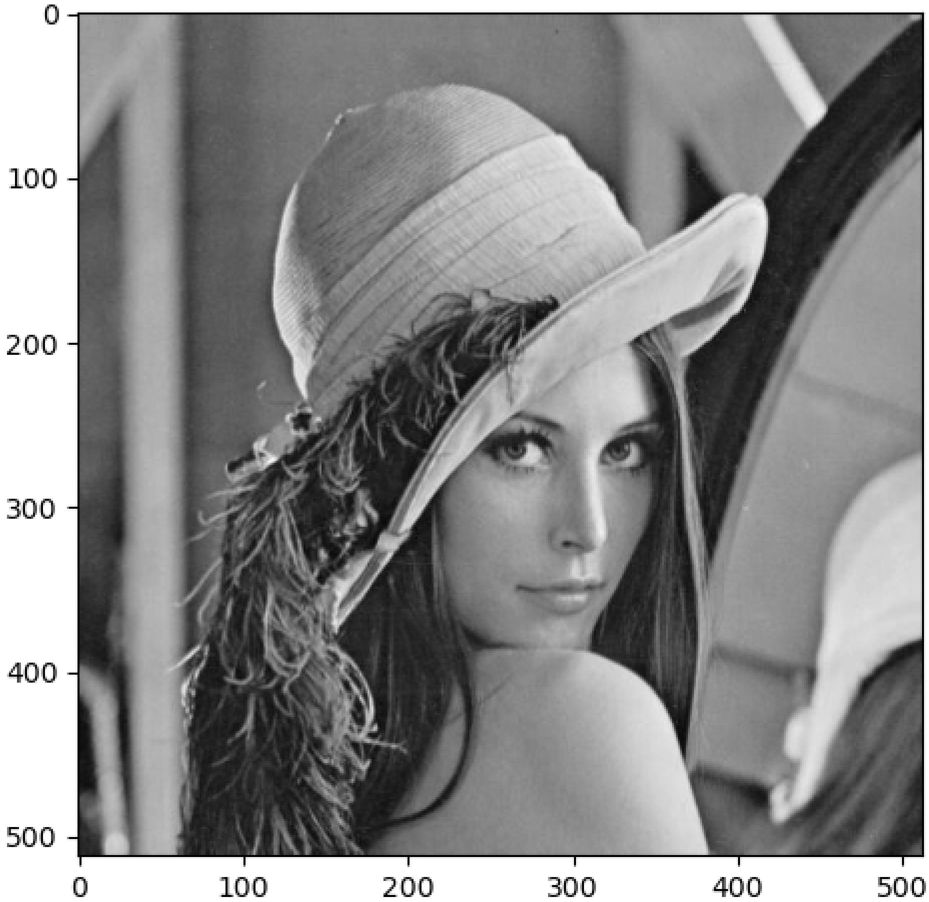
$$mean = [1, 1], cov = [[1, 0.5], [0.5, 1]]$$

对人工生成数据进行PCA，结果如下

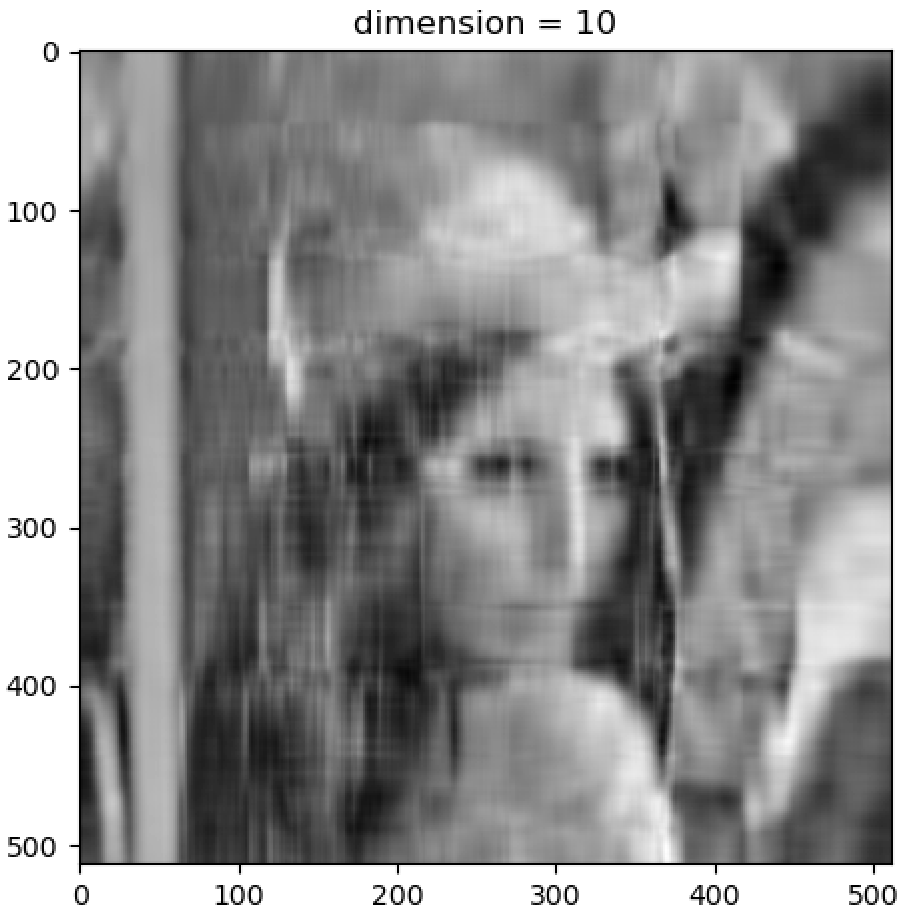


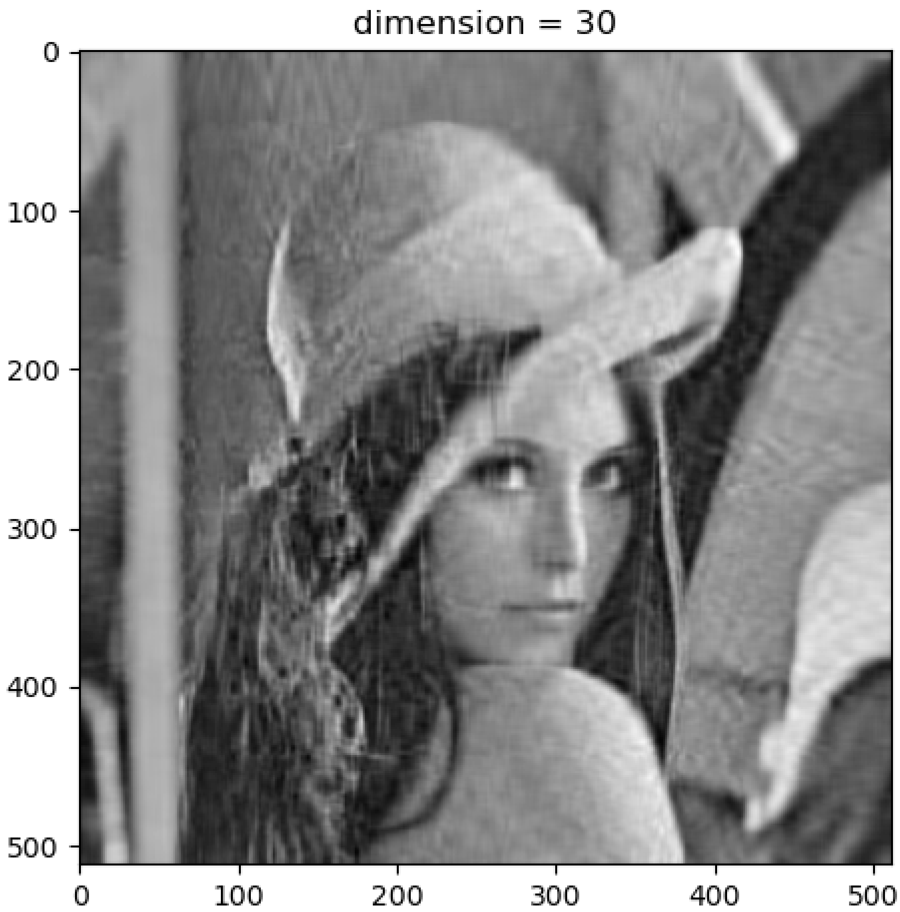
2.人脸数据的PCA

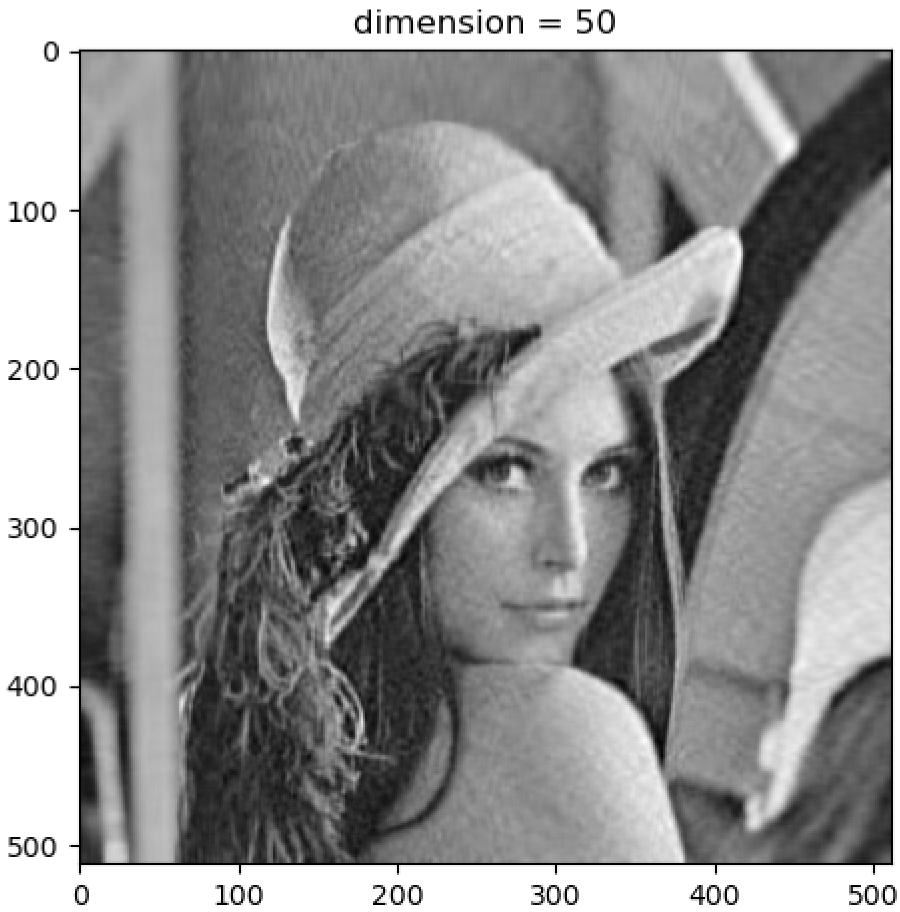
选取一张人脸图片

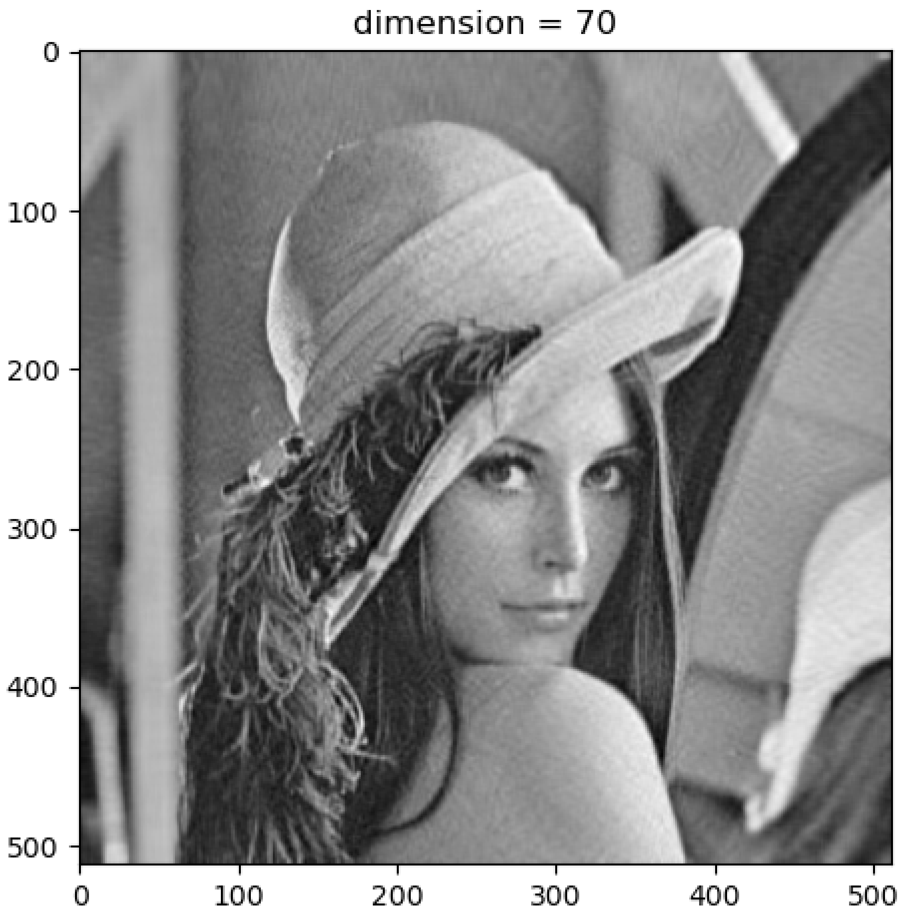


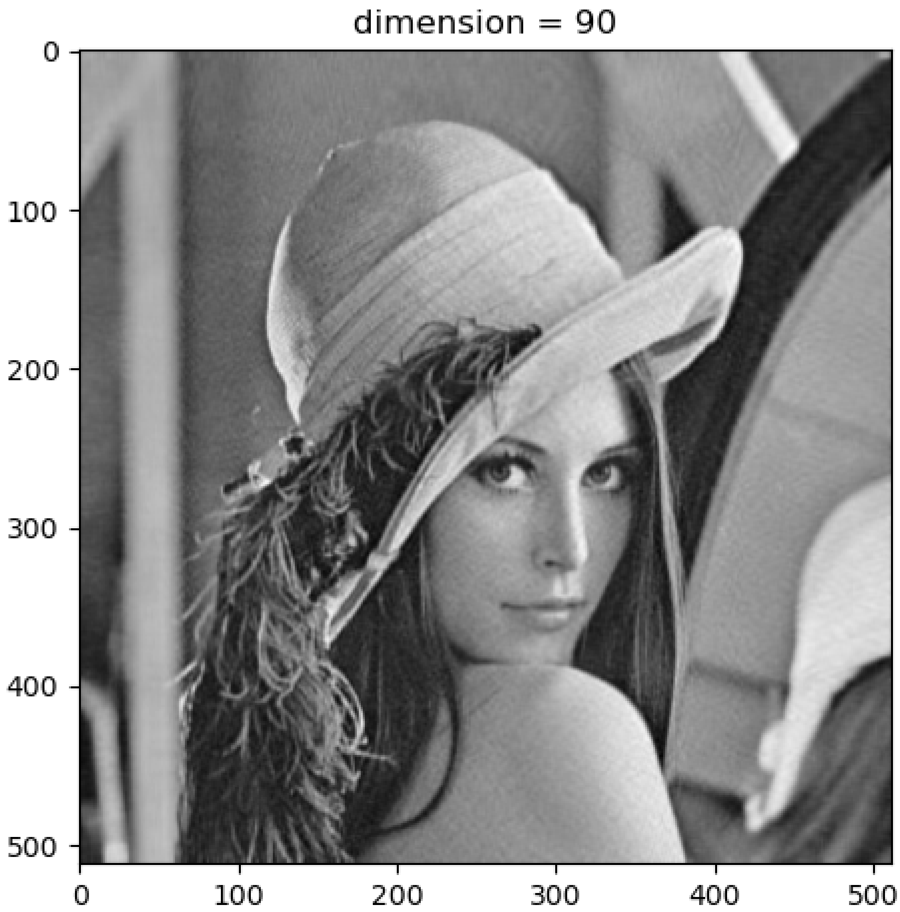
对其做PCA的结果如下（维数分别取10，30，50，70，90）


dimension = 10
0
100
200
300
400
500
0
100
200
300
400
500


dimension = 30
0
100
200
300
400
500
0
100
200
300
400
500


dimension = 50
0
100
200
300
400
500
0
100
200
300
400
500


dimension = 70
0
100
200
300
400
500
0
100
200
300
400
500



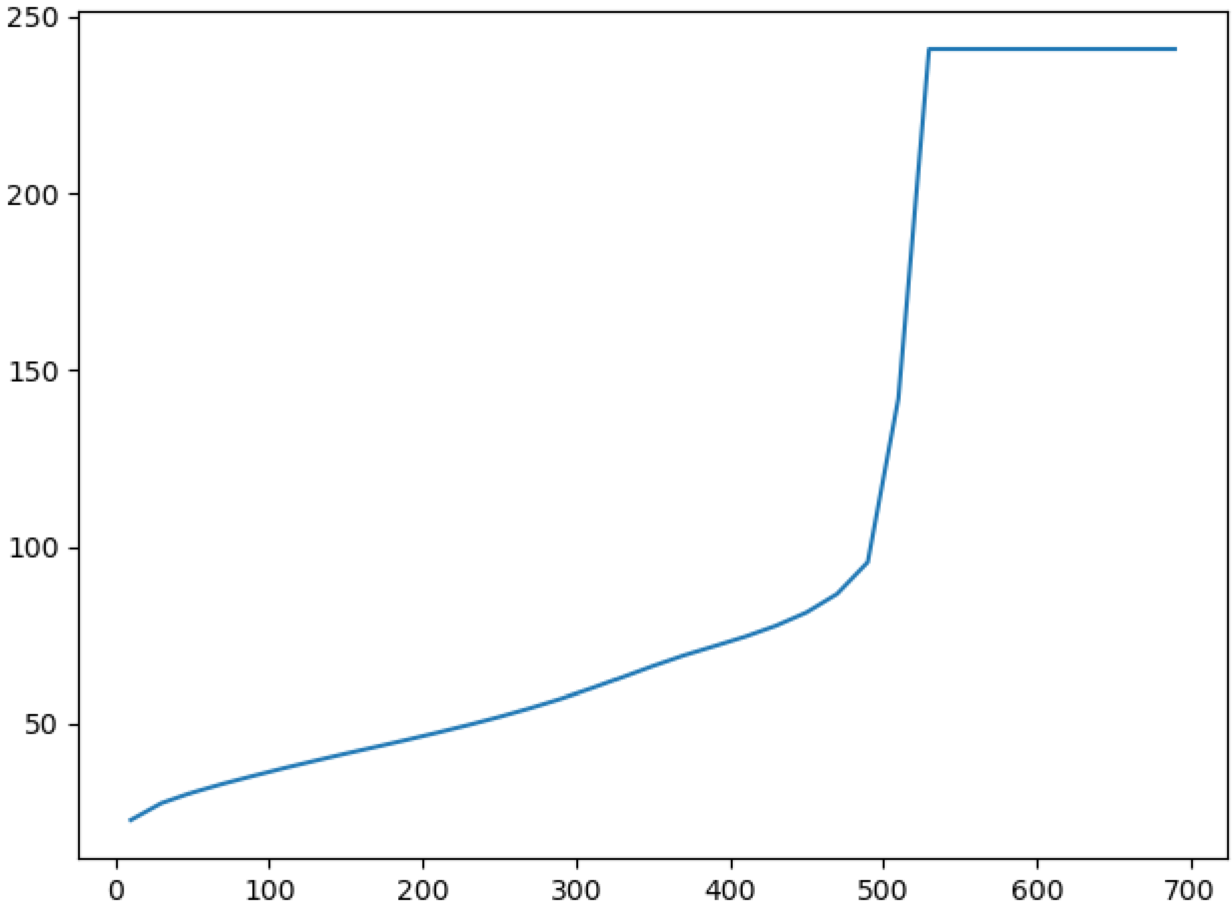
可以看到，随着低维空间维数的提高，源图片数据的信息保留更完全。

我们定义信噪比公式如下：

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

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

PCA取不同维度，信噪比变化的情况如下：



可以看到，随着低维空间维数的提高，信噪比也不断增大，说明源图片数据的信息保留更完全。

五、结论

1. PCA算法仅仅需要以方差衡量信息量，不受数据集以外的因素影响。
2. PCA算法将数据压缩到低维，各主成分之间正交，可消除原始数据成分间的相互影响的因素。
3. 选取主成分数量越多，保留原样本数据的信息效果越好。
4. 主成分各个特征维度的含义具有一定的模糊性，不如原始样本特征的解释性强，因而难以理解结果的含义。
5. 方差小的非主成分也可能含有对样本差异的重要信息，因降维丢弃可能对后续数据处理有影响，容易产生过拟合。

六、参考文献

- Christopher Bishop. Pattern Recognition and Machine Learning.
- 周志华 著. 机器学习, 北京: 清华大学出版社, 2016.1

七、附录:源代码(带注释)

```python
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image


def generate_data(dimension=2):
    if dimension == 2:
        mean = [1, 1]
        cov = [[1, 0.5], [0.5, 1]]
        x, y = np.random.multivariate_normal(mean, cov, 100).T
        data = np.zeros((x.shape[0], 2))
        data[:, 0] = x
        data[:, 1] = y
        return data


# 标准化
def standardize(data):
    mean = np.mean(data, axis=0)
    data = data - mean
    return data


def pca(data, n_components=1):
    data = standardize(data)
    # 计算协方差矩阵
    cov = np.cov(data.T)
    # 特征值和特征向量
    eigenvalues, eigenvectors = np.linalg.eig(cov)
    # 特征值从大到小的对应序列号
    index = np.argsort(eigenvalues)[::-1]
    # 选取最大的n个特征值对应的特征向量作为主成分
    principal_component = eigenvectors[:, index[0:n_components]]
    return principal_component


def plot_data(n_components):
    image = Image.open('lena.jpg')
    data = np.array(image)
    w = pca(data, n_components)
    mean = np.mean(data, axis=0)
    pca_data = (data - mean) @ w @ w.T + mean
    plt.imshow(pca_data, cmap='gray')


def PSNR(original, compressed):
    mse = np.mean((original - compressed) ** 2)
    if mse == 0:
        return 100
    max_pixel = 255.0
    psnr = 20 * np.log10(max_pixel / np.sqrt(mse))
    return psnr
```

```python
# generate data and pca
data = generate_data()
w = pca(data)
mean = np.mean(data, axis=0)
pca_data = (data - mean) @ w @ w.T + mean

# plot raw data and pca data
plt.scatter(data[:, 0], data[:, 1], label='raw data')
plt.scatter(pca_data[:, 0], pca_data[:, 1], label='pca data')
plt.legend()
plt.show()

# compare original image and pca image
image = Image.open('lena.jpg')
data = np.array(image)
plt.imshow(data, cmap='gray')
plt.show()
for i in range(10, 100, 20):
    plot_data(n_components=i)
    plt.title(f'dimension = {i}')
    plt.show()
# compute psnr
x = []
psnr = []
for i in range(10, 700, 20):
    x.append(i)
    w = pca(data, n_components=i)
    mean = np.mean(data, axis=0)
    pca_data = (data - mean) @ w @ w.T + mean
    psnr.append(PSNR(data, pca_data))
plt.plot(x, psnr)
plt.show()
```