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：1190201303
姓名： 王艺丹

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

2.1 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）用高斯分布产生 k 个高斯分布的数据（不同均值和方差）（其中参数自己设定）。

2.2 实验环境

Windows 10；Anaconda 4.8.4；python 3.8.8；jupyter notebook 6.0.1

三、 概念设计思想（主要算法及数据结构）

3.1 PCA 基本原理：

PCA（principal components analysis）即主成分分析技术旨在利用降维的思想，把多指标转化为少数几个综合指标：

从信息熵的角度来看，当降维后的数据方差值最大，此时保留的信息越多，因此我们的问题可以描述为：将 N 个 M 维向量将为 K 维，其目标是选择 K 个单位正交基，使得原始数据变换到这组基上后，各变量两两间协方差为 0，变量方差尽可能大。即在正交的约束下，取最大的 K 个方差

(一)假设 x 为 m 维随机变量，其均值为 μ，协方差矩阵为 Σ，考虑由 m 维随机变量 x 到 m 维随机变量 y 的线性变换：

$$y_i = \alpha_i^T x = \sum_{k=1}^{m} \alpha_{ki} x_k, \quad i = 1, 2, \cdots, m$$

其中$\alpha_i^T = (\alpha_{1i}, \alpha_{2i}, \cdots, \alpha_{mi})$

如果该线性变换满足以下条件，则称之为总体主成分：

1) $\alpha_i^T \alpha_i = 1, i = 1,2,\cdots,m$ ；
2) $\mathrm{cov}(y_i, y_j) = 0, (i \neq j)$
3) 变量 y_1 是 x 的所有线性变换中方差最大的；y_2 是与 y_1 不相关的 x 的所有线性变换中方差最大的；一般地，y_i是与$y_1, y_2, \cdots, y_{i-1}, (i = 1,2,\cdots,m)$ 都不相关的 x 的所有线性变换中方差最大的；这时分别称 $y_1, y_2, \cdots, y_m$ 为 x 的第一主成分、第二主成分、…、第 m 主成分

(二)假设 x 是 m 维随机变量，其协方差矩阵是 Σ ， Σ 的特征值分别是 $\lambda 1 \geq \lambda 2 \geq \cdots \geqslant \lambda m \geq 0$ ，特征值对应的单位特征向量分别是 $\alpha 1, \alpha 2, \cdots, \alpha m$ ，则 x 的第 2 主成分可以写作:

$$y_i = \alpha_i^T x = \sum_{k=1}^{m} \alpha_{ki} x_k, \quad i = 1, 2, \cdots, m$$

并且， x 的第 i 主成分的方差是协方差矩阵 Σ 的第 i 个特征值，即:

$$\mathrm{var}(y_i) = \alpha_i^T \Sigma \alpha_i = \lambda_i$$

(三)主成分有以下性质:

主成分y的协方差矩阵是对角矩阵:

$$\mathrm{cov}(y) = \Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \cdots, \lambda_m)$$

主成分y的方差之和等于随机变量x的方差之和

$$\sum_{i=1}^{m} \lambda_i = \sum_{i=1}^{m} \sigma_{ii}$$

主成分 yk 与变量 $x2$ 的相关系数 $\rho(yk,xi)$ 称为因子负荷量（factor loading)，它表示第 k 个主成分 yk 与变量 x 的相关关系，即 yk 对 x 的贡献程度：

$$\rho(y_k, x_i) = \frac{\sqrt{\lambda_k}\alpha_{ik}}{\sqrt{\sigma_{ii}}}, \quad k, i = 1, 2, \cdots, m$$

(四)样本主成分分析就是基于样本协方差矩阵的主成分分析

给定样本矩阵：

$$X = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix}$$

其中xj=($x1j$, $x2j$, ···, xmj)T是x的第j个独立观测样本，j=1, 2，···, n。

X的样本协方差矩阵为：

$$S = [s_{ij}]_{m\times m}, \quad s_{ij} = \frac{1}{n-1} \sum_{k=1}^{n} (x_{ik} - \overline{x}_i)(x_{jk} - \overline{x}_j)$$

$$i = 1, 2, \cdots, m, \quad j = 1, 2, \cdots, m$$

给定样本数据矩阵 X ，考虑向量 x 到 y 的线性变换:

$$y = A^T x$$

$$A = \begin{bmatrix} a_1 & a_2 & \cdots & a_m \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mm} \end{bmatrix}$$

如果该线性变换满足以下条件，则称之为样本主成分。样本第一主成分 $y1=aT1x$ 是在 $aT1a1=1$ 条件下，使得 $aT1xj(j=1,2,\cdots,n)$ 的样本方差 $aT1Sa1$ 最大的 x 的线性变换；样本第二主成分 $y2=aT2x$ 是在 $aT2a2=1$ 和 $aT2xj$ 与 $aT1xj(j=1,2,\cdots,n)$ 的样本协方差 $aT1Sa2=0$ 条件下，使得 $aT2xj(j=1,2,\cdots,n)$ 的样本方差 $aT2Sa2$ 最大的 x 的线性变换；

一般地，样本第 i 主成分 $yi=aTix$ 是在 $aTiai=1$ 和 $aTixj$ 与 $aTkxj(k<i,j=1,2,\cdots,n)$ 的样本协方差 $aTkSai=0$ 条件下，使得 $aTixj(j=1,2,\cdots,n)$ 的样本方差 $aTkSai$ 最大的 x 的线性变换

3.2 PCA 实现

主成分分析方法主要有两种，可以通过相关矩阵的特征值分解或样本矩阵的奇异值分解进行：

（1）相关矩阵的特征值分解算法。针对$m \times n$样本矩阵X，求样本相关矩阵

$$R = \frac{1}{n-1} X X^T$$

再求样本相关矩阵的k个特征值和对应的单位特征向量，构造正交矩阵

$$V = (v_1, v_2, \cdots, v_k)$$

V的每一列对应一个主成分，得到$k \times n$样本主成分矩阵

$$Y = V^T X$$

（2）矩阵X的奇异值分解算法。针对$m \times n$样本矩阵X

$$X' = \frac{1}{\sqrt{n-1}} X^T$$

对矩阵X'进行截断奇异值分解，保留k个奇异值、奇异向量，得到

$$X' = U S V^T$$

V的每一列对应一个主成分，得到$k \times n$样本主成分矩阵Y

$$Y = V^T X$$

本次实验选择通过奇异值分解实现，具体代码如下：

```python
def pca(x, k):
    """
    PCA降维并映射还原至高维度
    """
    X = (x - x.mean()) / x.std() # 数据归一化
    X = np.matrix(X)
    cov = (X.T * X) / X.shape[0] # n*n
    U, S, V = np.linalg.svd(cov) # 奇异值分解
    U_reduced = U[:, :k] # 左奇异向量，由大至小取前k个特征值，并转为行向量 n*k
    X_reduced = np.dot(X, U_reduced) # 主成分矩阵 m*k
    X_recovered = np.dot(X_reduced, U_reduced.T) # 映射回高维空间 m*n
    x_pca = X_recovered * x.std() + x.mean() # 还原数据
    return x_pca
```

3.3 峰值信噪比（PSNR）

PSNR 是图像压缩质量的评估指标，PSNR 越高，压缩后失真越小，压缩算法效果越好。计算方法如下：

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}||I(i,j)-K(i,j)||^2$$

$$PSNR = 10\log_{10}(\frac{MAX_I^2}{MSE})$$

其中，其中 I 和 K 代表参考图像与失真图像，均为 M*N 的图片,MAX=255 以给定的各类别中心，分别生成对应个数的样本点：

3.4 结果可视化

思想较为简单，不作赘述，看代码及注释即可：

```
def show_pca_2D(x):
      x_pca = pca(x, 1)
      plt.scatter(x[:, 0].tolist(), x[:, 1].tolist(), c='c',alpha=0.5, label="original data")
      plt.scatter(x_pca[:, 0].tolist(), x_pca[:, 1].tolist(), c='b', alpha=0.2, label='PCA data')
      plt.plot(x_pca[:, 0], x_pca[:, 1], c='b',alpha=0.1, label='vector')
      plt.legend(loc='upper left')
      plt.show()

def show_pca_3D(x):
      x_pca = pca(x, 2)
      fig = plt.figure()
      ax = Axes3D(fig)
      ax.scatter(x[:, 0].tolist(), x[:, 1].tolist(), x[:, 2].tolist(), c='c',alpha=0.5, label="original data")
      ax.scatter(x_pca[:, 0].tolist(), x_pca[:, 1].tolist(), x_pca[:, 2].tolist(), c='b', alpha=0.2, label='PCA data')
      ax.legend(loc='upper left')
      plt.show()
```

3.5 人脸数据处理

主要处理流程为：

1. 从网络获取人脸图片
2. 预处理：对图片的尺寸重构，转为灰度图象
3. 使用 PCA 进行降维
4. 重构图片
5. 计算峰值信噪比 PSNR

```python
def read_image_data():
    file_dir_path = 'data/'
    file_list = os.listdir(file_dir_path)
    image_data = []
    plt.figure(figsize=(50, 50))
    i = 1
    for file in file_list:
        file_path = os.path.join(file_dir_path, file)
        print("open figure " + file_path)
        plt.subplot(3, 3, i)
        with open(file_path) as f:
            img = cv.imread(file_path, cv.IMREAD_GRAYSCALE)
            print(img.shape)
            img = cv.resize(img, (400, 400), interpolation=cv.INTER_NEAREST)
            print(img.shape)
            data = np.asarray(img)
            image_data.append(data)
            plt.imshow(img, cmap=plt.cm.gray)
        i += 1
    plt.show()
    return np.asarray(image_data)

def cal_nr(img_1, img_2):
    noise = np.mean(np.square(img_1 / 255. - img_2 / 255.))
    if noise < 1e-10:
        return 100
    return 20 * np.log10(1 / np.sqrt(noise))

def pca_images(k):
    data = read_image_data()
    image_number, image_feature = data[0].shape
    print(data.shape)
    pca_data = []
    for i in range(len(data)):
        # print(pca_data)
        pca_data.append(PCA.pca(data[i], k))
    plt.figure(figsize=(50, 50))
    for i in range(len(data)):
        plt.subplot(3, 3, i + 1)
        plt.imshow(pca_data[i], cmap=plt.cm.gray)
    plt.show()

    print("the signal to noise ratio of the image after PCA:")
    for i in range(len(data)):
        ratio = cal_nr(data[i], pca_data[i])
        print('The noise ratio of image ' + str(i) + ' is ' + str(ratio))
```

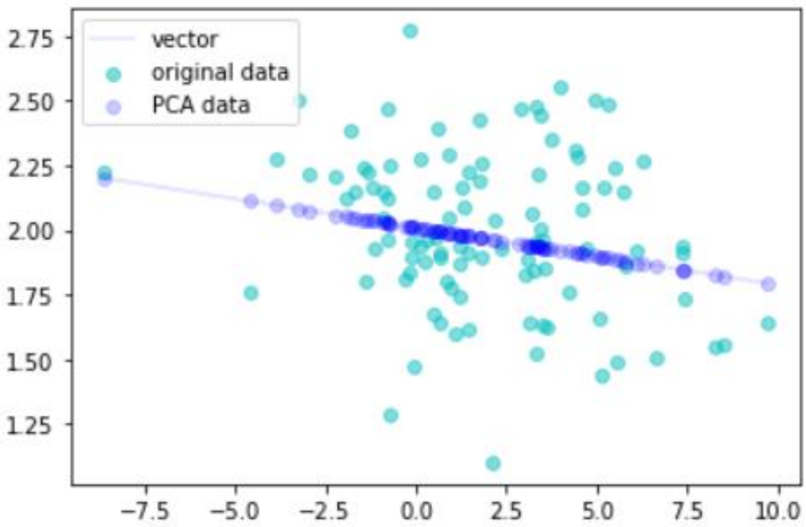
四、 实验结果与分析

4.1 自生成数据集：

```python
def generate_data(mu, sigma, n):
    """
    生成高斯分布数据
    """
    x = np.random.multivariate_normal(mean=mu, cov=sigma, size=n)
    return x
```

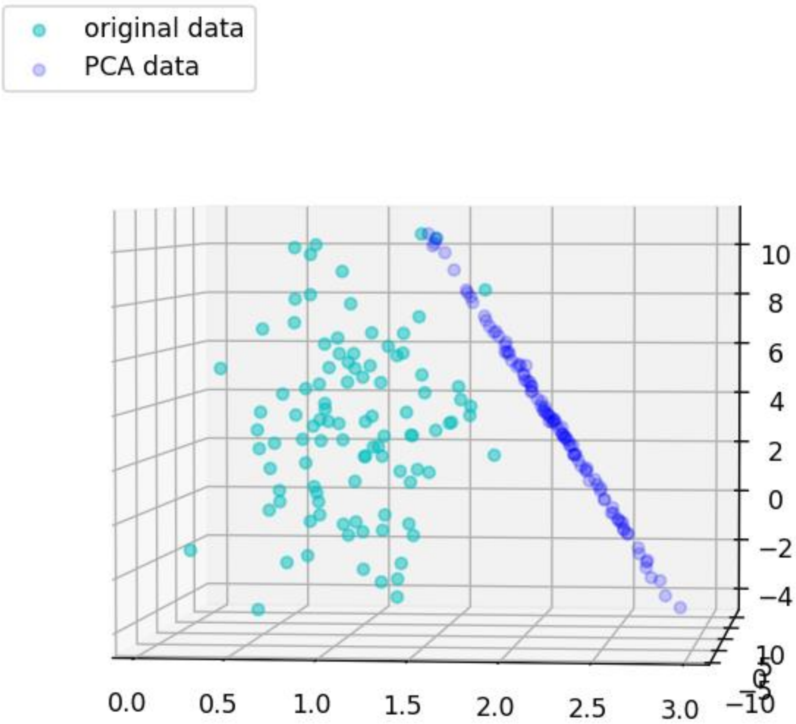
二维：

```python
mean = [2, 2]
cov = [[10, 0], [0, 0.1]]
x_2d = generate_data(mean, cov, n=100)
PCA.show_pca_2D(x_2d)
```



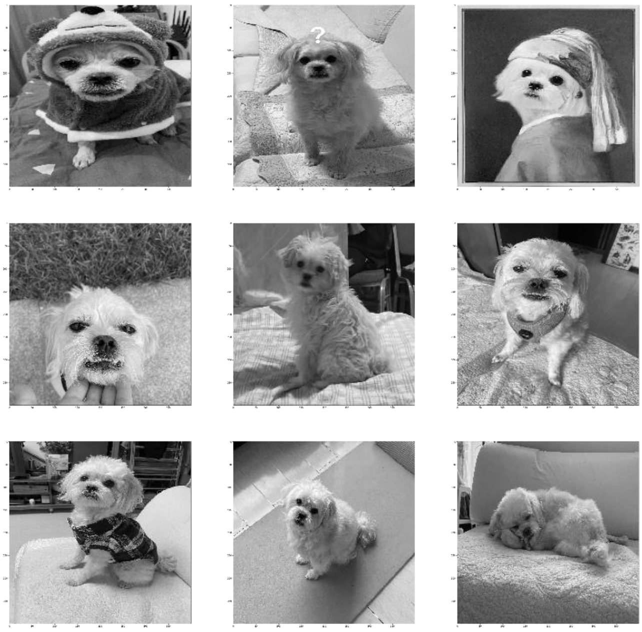
三维：

```
mean_3d = [1, 2, 3]
cov_3d = [[0.1, 0, 0], [0, 10, 0], [0, 0, 10]]
x_3d = generate_data(mean_3d, cov_3d, n=100)
%matplotlib notebook
PCA.show_pca_3D(x_3d)
```

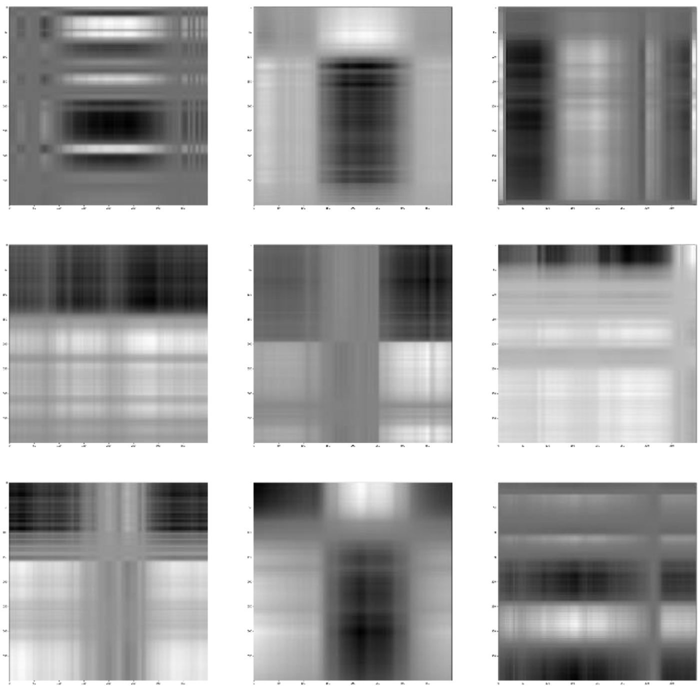


4.2 照片数据

选择 9 张萌宠图片预处理结果如下：

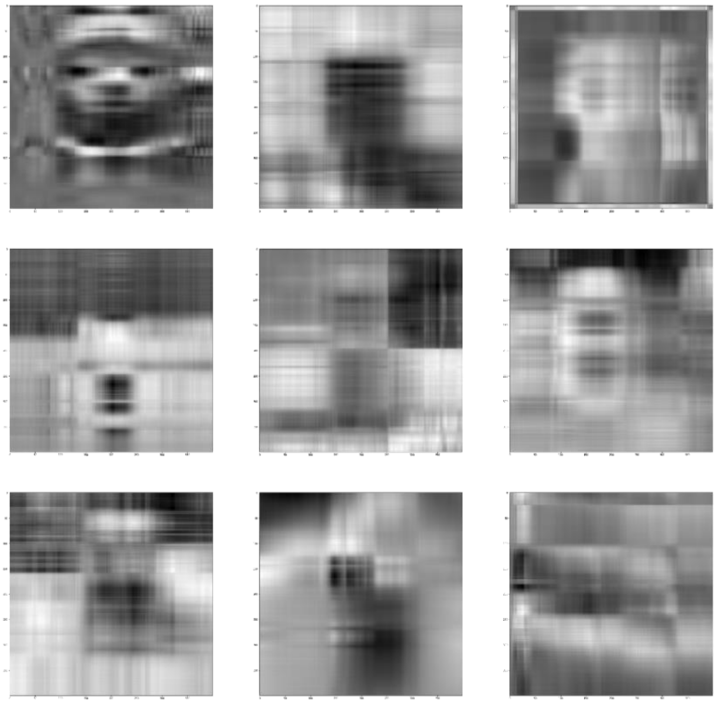


k = 1:



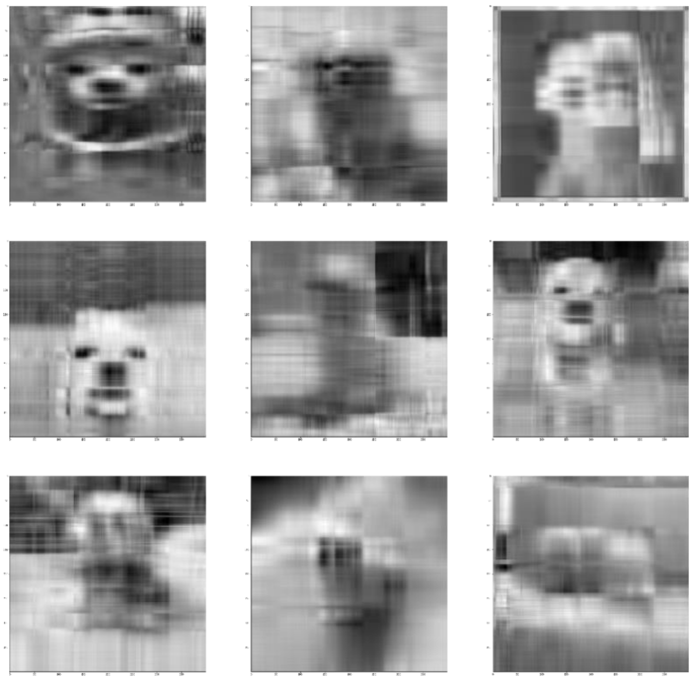
```
the signal to noise ratio of the image after PCA:
The noise ratio of image 0 is 15.349356419109066
The noise ratio of image 1 is 15.7928172182522
The noise ratio of image 2 is 14.69316030520325
The noise ratio of image 3 is 17.12106789850705
The noise ratio of image 4 is 16.613649998251493
The noise ratio of image 5 is 14.95692089082574
The noise ratio of image 6 is 13.18594412781659
The noise ratio of image 7 is 18.114393788332613
The noise ratio of image 8 is 17.971830760495642
The average noise ratio of image is 15.977682378532627
```

k = 3:



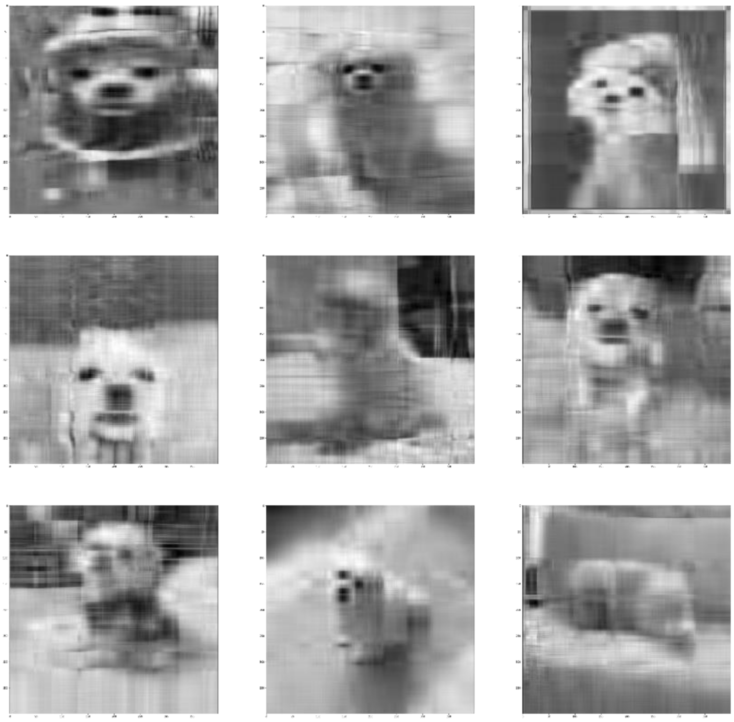
```
the signal to noise ratio of the image after PCA:
The noise ratio of image 0 is 17.51642211500737
The noise ratio of image 1 is 19.87311010212547
The noise ratio of image 2 is 18.10670849775694
The noise ratio of image 3 is 20.126344533541168
The noise ratio of image 4 is 19.3329165467524
The noise ratio of image 5 is 17.814308746507585
The noise ratio of image 6 is 16.81041519771184
The noise ratio of image 7 is 21.77083441966692
The noise ratio of image 8 is 20.83265688228976
The average noise ratio of image is 19.131524115706608
```

k = 5:



```
the signal to noise ratio of the image after PCA:
The noise ratio of image 0 is 18.925485237421107
The noise ratio of image 1 is 21.56324283780936
The noise ratio of image 2 is 20.731283977007603
The noise ratio of image 3 is 21.69827572159228
The noise ratio of image 4 is 21.740858168900164
The noise ratio of image 5 is 19.01475649742052
The noise ratio of image 6 is 18.91475477252484 8
The noise ratio of image 7 is 23.42229540210678 4
The noise ratio of image 8 is 22.13401357816679
The average noise ratio of image is 20.90499624366105
```

k = 7:

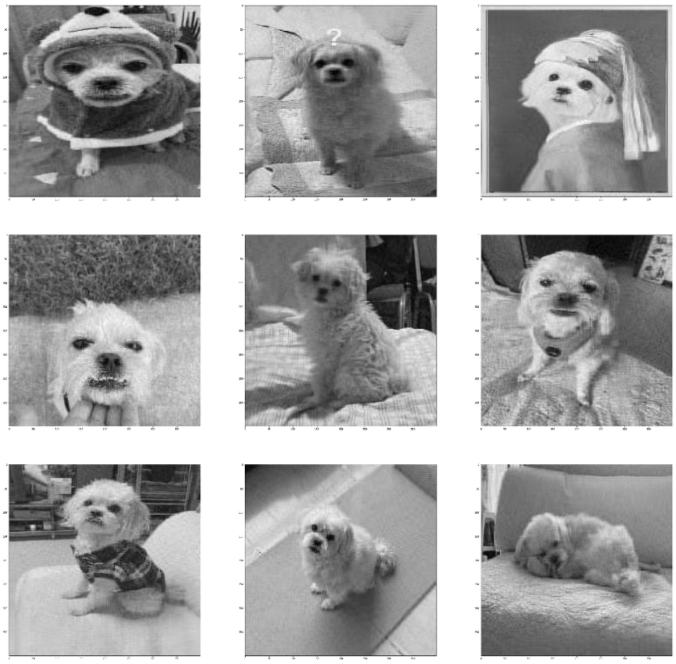


```
the signal to noise ratio of the image after PCA:
The noise ratio of image 0 is 20.0888670110514
The noise ratio of image 1 is 22.414916110528264
The noise ratio of image 2 is 22.230078166239853
The noise ratio of image 3 is 22.406439442599698
The noise ratio of image 4 is 23.189389819100615
The noise ratio of image 5 is 20.007997940123957
The noise ratio of image 6 is 20.230252896477
The noise ratio of image 7 is 24.57162902585533
The noise ratio of image 8 is 22.96841763654555
The average noise ratio of image is 22.01199867205796
```

k = 10:

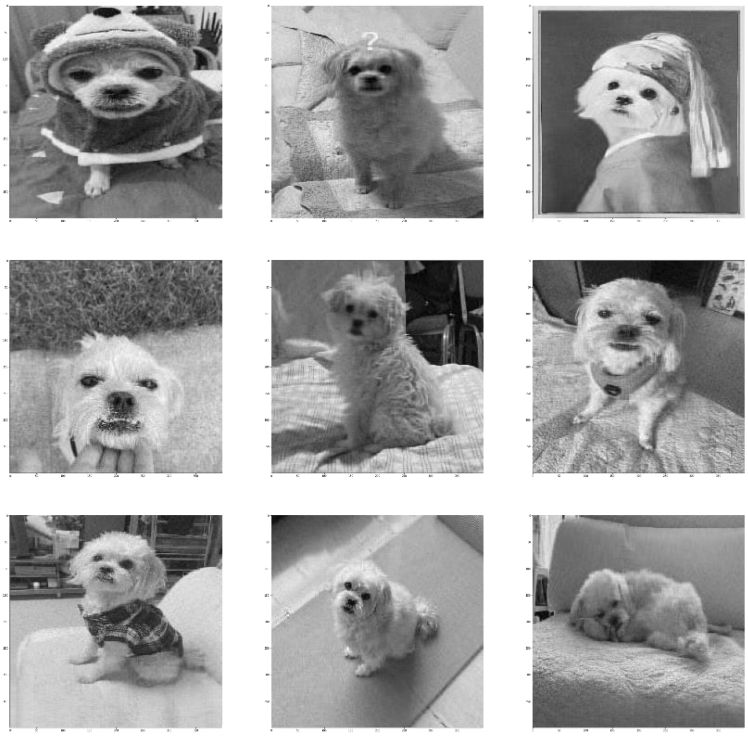
```
the signal to noise ratio of the image after PCA:
The noise ratio of image 0 is 21.411190631615106
The noise ratio of image 1 is 23.266830679698508
The noise ratio of image 2 is 24.20452599720466
The noise ratio of image 3 is 23.136677053508112
The noise ratio of image 4 is 24.6935185416123
The noise ratio of image 5 is 21.124316518614012
The noise ratio of image 6 is 21.557413689841418
The noise ratio of image 7 is 25.580856000872508
The noise ratio of image 8 is 23.764053804366633
The average noise ratio of image is 23.193264768592584
```

k = 40:



```
the signal to noise ratio of the image after PCA:
The noise ratio of image 0 is 27.65916949125351
The noise ratio of image 1 is 26.67759922306186
The noise ratio of image 2 is 30.86215083434835
The noise ratio of image 3 is 26.918695191599245
The noise ratio of image 4 is 31.21495151753875
The noise ratio of image 5 is 24.835813241061572
The noise ratio of image 6 is 25.755463320651227
The noise ratio of image 7 is 28.634695055322844
The noise ratio of image 8 is 26.856144765266198
The average noise ratio of image is 27.71274251556706
```

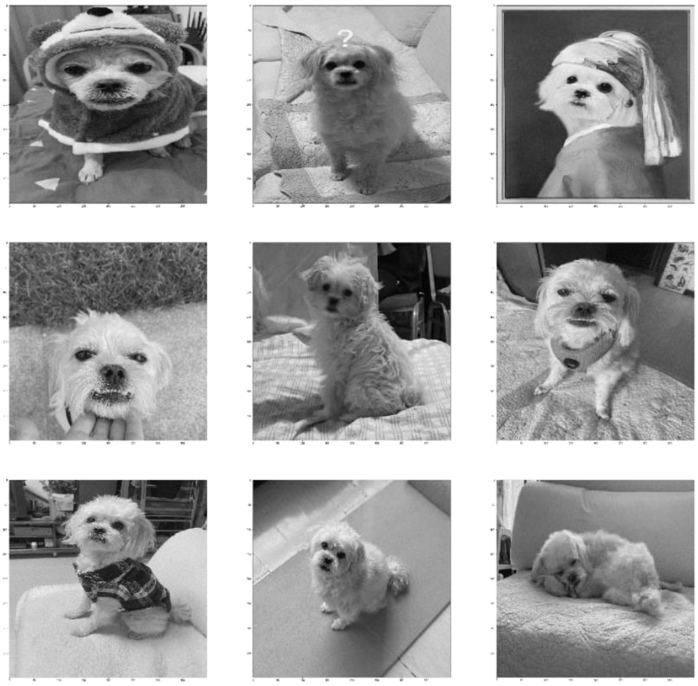
k = 50:



```
the signal to noise ratio of the image after PCA:
The noise ratio of image 0 is 28.708011844309045
The noise ratio of image 1 is 27.430826984872766
The noise ratio of image 2 is 31.936850020997184
The noise ratio of image 3 is 27.88570557100714
The noise ratio of image 4 is 32.6032275661955
The noise ratio of image 5 is 25.62641526060707
The noise ratio of image 6 is 26.511256935989994
The noise ratio of image 7 is 29.242455731142663
The noise ratio of image 8 is 27.590963203139847
The average noise ratio of image is 28.615079235362355
```

k = 100:

可以看到，k=50 与 k = 100 时，psnr 相差不大，图片清晰度相差也不大，已经基本可以清晰的展示图片了。



```
the signal to noise ratio of the image after PCA:
The noise ratio of image 0 is 32.79717938188914
The noise ratio of image 1 is 30.80578267651859
The noise ratio of image 2 is 35.78517252411273
The noise ratio of image 3 is 32.441552895866174
The noise ratio of image 4 is 38.209110835679866
The noise ratio of image 5 is 29.141316102408382
The noise ratio of image 6 is 29.680510923735284
The noise ratio of image 7 is 32.021129043243455
The noise ratio of image 8 is 31.15784350231423
The average noise ratio of image is 32.44884420952976
```

平均计算峰值信噪比与对应的 k 值关系如下：

k	PSNR
1	15.977682378532627
2	17.6620185743855
3	19.131524115706608
4	20.156715432128262
5	20.90499624366105
6	21.51988367025307
7	22.01199867205796
8	22.45515524563625
10	23.193264768592584
30	26.682729440973954
50	28.615079235362355
100	32.44884420952976

可以看到，k=50 与 k = 100 时，psnr 相差不大，图片清晰度相差也不大，已经基本可以清晰的展示图片了。

4.3 矩阵奇异值的解决

从线性代数的角度来看，此时

$$r(X) \leq N = 10$$

则样本协方差矩阵S

$$S = \frac{1}{N} X X^T$$

$$r(S) \leq r(X) \leq 10$$

而S是一个$D * D$的矩阵，所以出现了奇异的情况，也就是说此时样本协方差矩阵最多只有N个特征值，相对应的最多只有N个特征向量，因此降维到$D' = 10$的时候已经可以包括所有的信息。

五、 结论

✧ 从结果上来看，将图片压缩的维度越多，直观上图片失真越严重，并且峰值信噪比越小

✧ 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。

✧ PCA 算法中舍弃了 n-k 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

六、 参考文献

[1] 李航, 《统计学习方法》(第三版)

[2] 周志华, 《机器学习》

[3] Pattern Recognition and Machine Learning.

七、 附录：源代码（带注释）

PCA.py

```
#!/usr/bin/env python

# coding: utf-8



# In[1]:
```

```python
import numpy as np

import matplotlib.pyplot as plt

from mpl_toolkits.mplot3d import Axes3D




# In[2]:




def pca(x, k):

    """

    PCA 降维并映射还原至高维度

    """

    X = (x - x.mean()) / x.std() # 数据归一化

    X = np.matrix(X)

    cov = (X.T * X) / X.shape[0] # n*n

    U, S, V = np.linalg.svd(cov) # 奇异值分解

    U_reduced = U[:,:k] # 左奇异向量，由大至小取前 k 个特征值，并转为行向量 n*k

    X_reduced = np.dot(X, U_reduced) # 主成分矩阵 m*k

    X_recovered = np.dot(X_reduced, U_reduced.T) # 映射回高维空间 m*n

    x_pca = X_recovered * x.std() + x.mean() # 还原数据

    return x_pca




# In[3]:




def show_pca_2D(x):

    x_pca = pca(x, 1)
```

```
    plt.scatter(x[:, 0].tolist(), x[:, 1].tolist(), c='c',alpha=0.5, label="original data")

    plt.scatter(x_pca[:, 0].tolist(), x_pca[:, 1].tolist(), c='b', alpha=0.2, label='PCA data')

    plt.plot(x_pca[:, 0], x_pca[:, 1], c='b',alpha=0.1, label='vector')

    plt.legend(loc='upper left')

    plt.show()


def show_pca_3D(x):

    x_pca = pca(x, 2)

    fig = plt.figure()

    ax = Axes3D(fig)

    ax.scatter(x[:, 0].tolist(), x[:, 1].tolist(), x[:, 2].tolist(), c='c',alpha=0.5, label="original data")

    ax.scatter(x_pca[:, 0].tolist(), x_pca[:, 1].tolist(), x_pca[:, 2].tolist(), c='b', alpha=0.2, label='PCA data')

    ax.legend(loc='upper left')

    plt.show()
```

Faces.py

```
#!/usr/bin/env python

# coding: utf-8


# In[1]:



import numpy as np

import matplotlib.pyplot as plt

import cv2 as cv

import os
```

```
import PCA



# In[3]:

def read_image_data():

    file_dir_path = 'data/'

    file_list = os.listdir(file_dir_path)

    image_data = []

#     plt.figure(figsize=(50, 50))

    i = 1

    for file in file_list:

        file_path = os.path.join(file_dir_path, file)

#         print("open figure " + file_path)

#         plt.subplot(3, 3, i)

        with open(file_path) as f:

            img = cv.imread(file_path, cv.IMREAD_GRAYSCALE)

#             print(img.shape)

            img = cv.resize(img, (400, 400), interpolation=cv.INTER_NEAR
EST)

#             print(img.shape)

            data = np.asarray(img)

            image_data.append(data)

#             plt.imshow(img, cmap=plt.cm.gray)

        i += 1

#     plt.show()

    return np.asarray(image_data)


def cal_nr(img_1, img_2):

    noise = np.mean(np.square(img_1 / 255. - img_2 / 255.))

    if noise < 1e-10:
```

```
        return 100
    return 20 * np.log10(1 / np.sqrt(noise))


def pca_images(k):
    data = read_image_data()
    image_number, image_feature = data[0].shape
#     print(data.shape)
    pca_data = []
    for i in range(len(data)):
        pca_data.append(PCA.pca(data[i], k))
    plt.figure(figsize=(50, 50))
    for i in range(len(data)):
        plt.subplot(3, 3, i + 1)
        plt.imshow(pca_data[i], cmap=plt.cm.gray)
    plt.show()
    ratio_sum = 0
    print("the signal to noise ratio of the image after PCA:")
    for i in range(len(data)):
        ratio = cal_nr(data[i], pca_data[i])
        ratio_sum += ratio
        print('The noise ratio of image ' + str(i) + ' is ' + str(rati
o))
    print('The average noise ratio of image is ' + str(ratio_sum/9))
```

test.ipynb

```
import numpy as np
import matplotlib.pyplot as plt
import Faces
import PCA
def generate_data(mu, sigma, n):
```

```
    """
    生成高斯分布数据
    """
    x = np.random.multivariate_normal(mean=mu, cov=sigma, size=n)
    return x
mean = [2, 2]
cov = [[10, 0], [0, 0.1]]
x_2d = generate_data(mean, cov, n=100)
PCA.show_pca_2D(x_2d)

mean_3d = [1, 2, 3]
cov_3d = [[0.1, 0, 0], [0, 10, 0], [0, 0, 10]]
x_3d = generate_data(mean_3d, cov_3d, n=100)
%matplotlib notebook
PCA.show_pca_3D(x_3d)
Faces.pca_images(1)
Faces.pca_images(2)
Faces.pca_images(3)
Faces.pca_images(4)
Faces.pca_images(5)
Faces.pca_images(6)
Faces.pca_images(7)
Faces.pca_images(8)
Faces.pca_images(10)
Faces.pca_images(30)
Faces.pca_images(50)
Faces.pca_images(100)
```