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： 专业选修

实验题目： PCA 模型实验

学号：120L030501
姓名： 张明远

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）

二、实验要求及实验环境

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）。

环境：

- Win10，x86-64
- Pycharm 2019.1
- python 3.7.1
- 使用 python 库：numpy（计算）, matplotlib.pyplot(用于可视化)

三、设计思想（本程序中的用到的主要算法及数据结构）

3.1 生成数据

本实验要求对多维数据进行 PCA 降维。利用多维高斯分布函数来生成数据，为便于画图展示，主要使用三维数据。在这里生成满足朴素贝叶斯假设的数据，这里选取中心点为（0，0）。

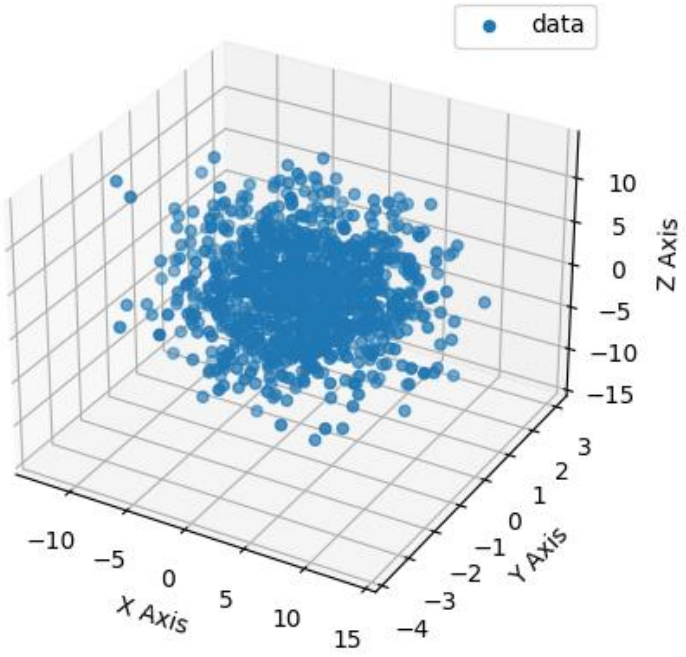
若满足朴素贝叶斯，则认为 X,Y,Z 的各维度数据条件独立，则协方差矩阵为

```
Cov = [[cov11,cov12,cov13]
       [cov12,cov22,cov23]
       [cov13,cov23,cov33]]
```

其中 cov12 = cov13 = cov23 = 0

cov11 就是 X 的方差， cov22 就是 Y 的方差，cov33 就是 Z 的方差

根据协方差矩阵添加高斯噪声。令 XYZ 方差分别为 20，10，20 生成样本点 1000 个，生成数据如下，数据保存在 data.csv：



可以看到数据点在 X 轴和 Z 轴分布较为分散，而 Y 轴则较为紧凑，区分度不大，满足可以降维数据的特征。

代码核心部分如下：

```
# 单类点生成器
def generatePoints(center, rand_normal_x, rand_normal_y, rand_normal_z, nums_p):
    # 中心
    x_p = center[0]
    y_p = center[1]
    z_p = center[2]
    # 协方差矩阵
    cov = [[rand_normal_x, naive_rate_xy, naive_rate_xz], [naive_rate_xy, rand_normal_y, naive_rate_yz],
           [naive_rate_xz, naive_rate_yz, rand_normal_z]]
    # x 各维生成
    ps = np.random.multivariate_normal((x_p, y_p, z_p), cov, nums_p)
    return ps
```

3.2 PCA

3.2.1 数据降维

降维就是一种对高维度特征数据预处理方法。降维是将高维度的数据保留下最重要的一些特征，去除噪声和不重要的特征，从而实现提升数据处理速度的目的。

在实际的生产和应用中，降维在一定的信息损失范围内，可以为我们节省大量的时间和成本。降维也成为应用非常广泛的数据预处理方法。最典型的降维算法之一就是 PCA 主成分分析。

3.2.2 PCA 原理

PCA(Principal Component Analysis)，即主成分分析方法，是一种使用最广泛的数据降维算法。

PCA 的主要思想是将 n 维特征映射到 k 维上，这 k 维是全新的正交特征也被称为主成分，是在原有 n 维特征的基础上重新构造出来的 k 维特征。PCA 的工作就是从原始的空间中顺序地找一组相互正交的坐标轴,新的坐标轴的选择与数据本身是密切相关的。其中，第一个新坐标轴选择是原始数据中方差最大的方向，第二个新坐标轴选取是与第一个坐标轴正交的平面中使得方差最大的,第三个轴是与第 1,2 个轴正交的平面中方差最大的。依次类推，可以得到 n 个这样的坐标轴。

通过这种方式获得的新的坐标轴，我们发现，大部分方差都包含在前面 k 个坐标轴中，后面的坐标轴所含的方差几乎为 0。于是，我们可以忽略余下的坐标轴，只保留前面 k 个含有绝大部分方差的坐标轴。事实上，这相当于只保留包含绝大部分方差的维度特征，而忽略包含方差几乎为 0 的特征维度，实现对数据特征的降维处理。

通过计算数据矩阵的协方差矩阵，然后得到协方差矩阵的特征值特征向量，选择特征值最大(即方差最大)的 k 个特征所对应的特征向量组成的矩阵。这样就可以将数据矩阵转换到新的空间当中，实现数据特征的降维。

由于得到协方差矩阵的特征值特征向量有两种方法：特征值分解协方差矩阵、奇异值分解协方差矩阵，所以 PCA 算法有两种实现方法：基于特征值分解协方差矩阵实现 PCA 算法、基于 SVD 分解协方差矩阵实现 PCA 算法。

这里主要使用基于 SVD 分解协方差矩阵的方法实现 PCA 算法，具体流程可表示为：

python 实现：

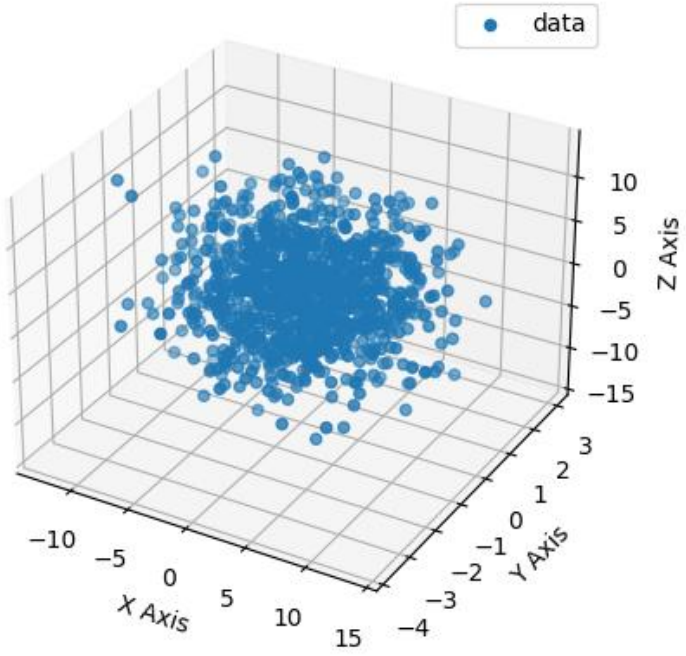
```
# PCA 实现，返回降维后的数据，特征向量基和降维后重构的数据
def PCA(data, dimension_to_cut):
    # 零均值化
    all_zero_mean(data)
```

```
    # 计算协方差矩阵
    Cov = get_cov_matrix(data)
    # svd 计算特征值和特征向量
    u, s, vT = np.linalg.svd(Cov)
    # 取前 k 个向量
    vT = vT[:][:dimension_to_cut]
    # 压缩数据
    data = data @ vT.T
    # 返回降维后的数据，特征向量基和降维后重构的数据
    return data, vT, data @ vT
```

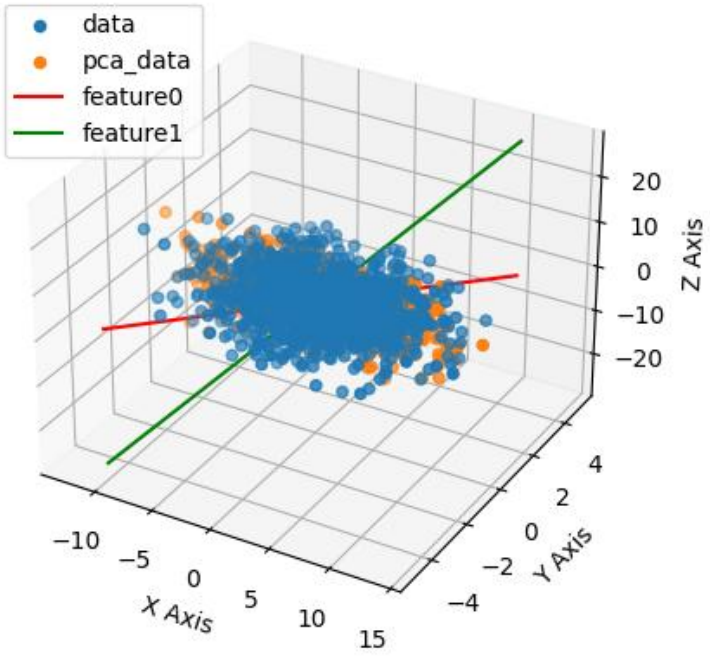
四、实验结果与分析

4.1 生成数据 PCA 降维

原始数据为



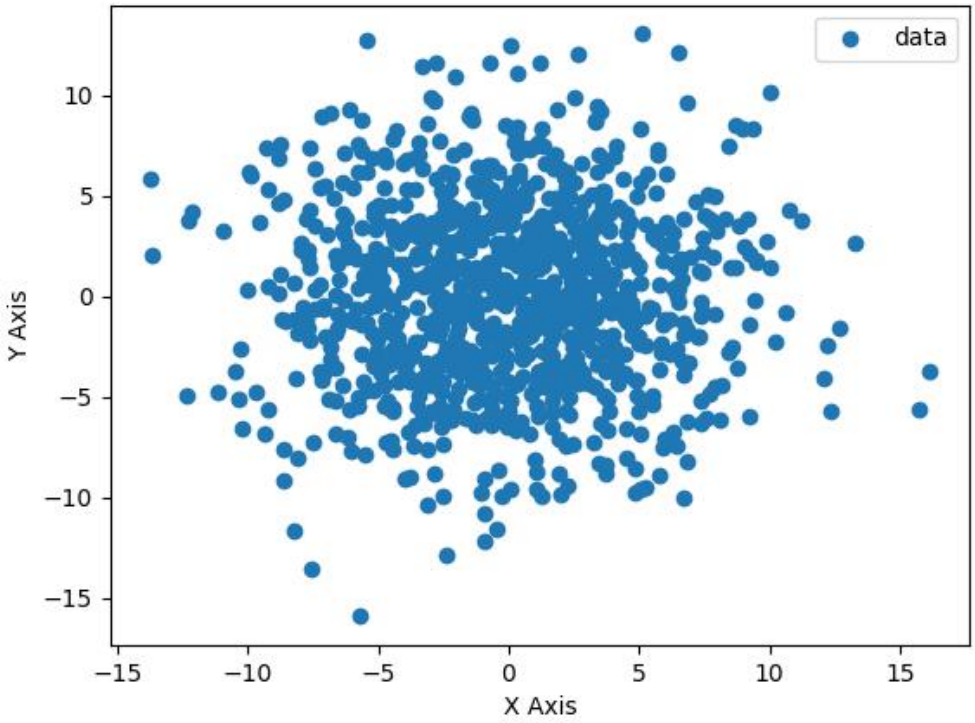
PCA 将其降至二维，找到两个特征向量，所在直线分别为 feature0 和 feature1，pca_data 为 data 在 feature0 和 feature1 张开的二维空间的投影。



两个特征值向量分别为：

```
特征向量为
[[-0.35965223 -0.00273235 -0.93308242]
 [ 0.93303727  0.00921067 -0.3596618 ]]
```

投影到二维空间，可以发现数据依然区分地很好



可见，我们重构的数据就是处于这 2 个特征向量所张开的线性空间中，这也验证了我们的 P C A 算法的原理。

4.2 人脸数据 PCA 应用

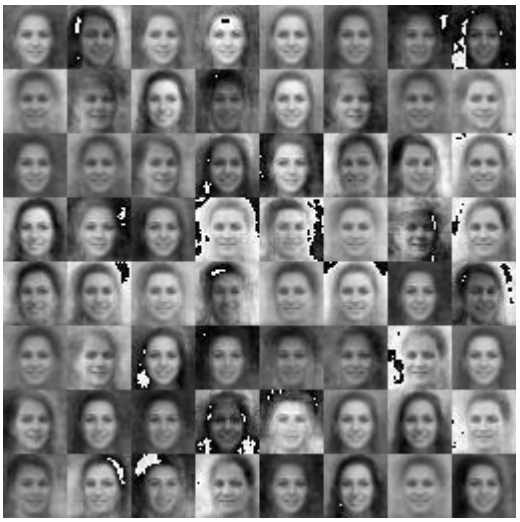
人脸图像数据是一种带有噪声的结构化数据，通过 PCA 主成分分析，可以提取到图像中的特征，进而对图像进行降维，去除噪声。

下面是我们选取的图片数据集：

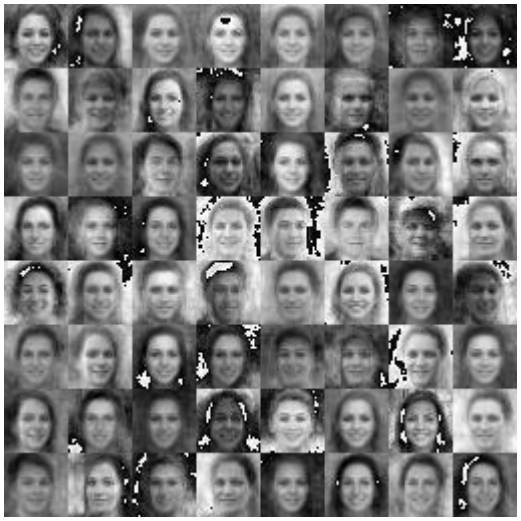
这是一张 1024*1024 清晰度的图片，由 8 行 8 列人脸图片构成，为了加快计算速度，将其压缩为 254*254 的灰度图像如下

接下来对这张图片进行 PCA 主成分提取，
令特征数 n 分别为 1，5，10，20，50，100，200，得到下列图像：

n = 1



n = 5



n = 10

n = 20

n = 50

n = 100

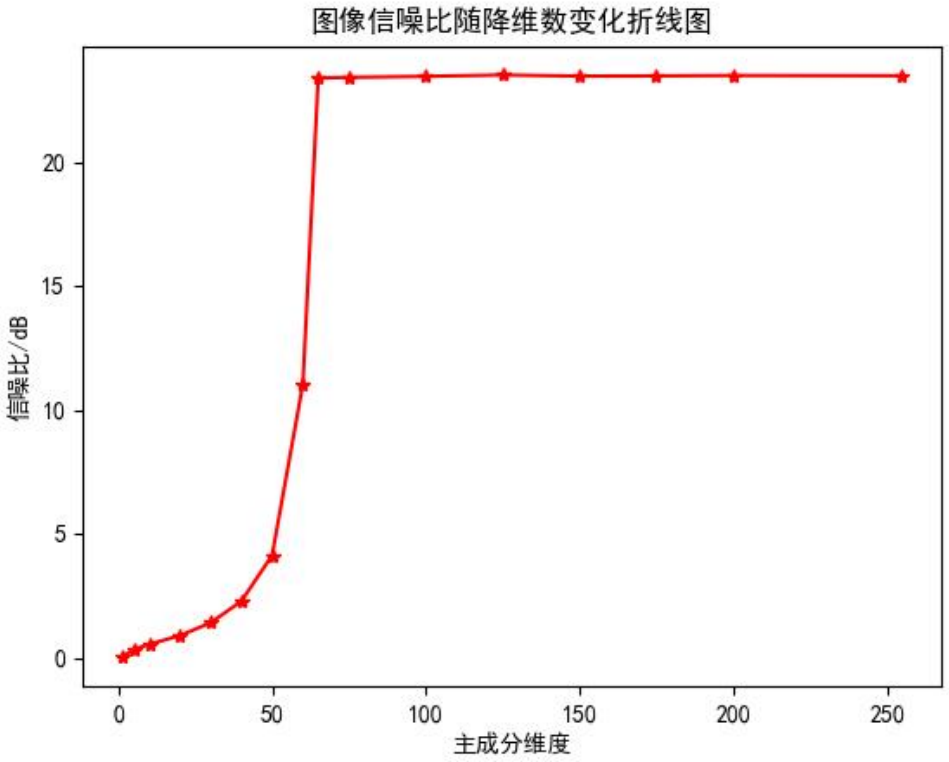
n = 200

可以看到，从 n = 1 到 n = 200,随着特征值的不断增加，图像变得越来越清晰，差异化越来越明显，但都保持着人脸的特征。降到 1 维时，各个压缩图片之间很接近，降到 5 维时，可见能勉强表示出原图像的特征， 但还是有部分图像之间比较接近，提高到 100 维时，才能够地完整地表示出原图像地特征。通过上述图片之间地比较，我们可以发现随着：保留的维度越高，压缩后的图像与原图越接近，也就是说保留了更多的信息。

前面只是从定性角度去分析，下面将从定量角度去分析图像所保留信息的多少和压缩维度之间的关系。

信噪比是指一个电子设备或者电子系统中信号与噪声的比例。这里面的信号指的是来自设备外部需要通过这台设备进行处理的电子信号,噪声是指经过该设备后产生的原信号中并不存在的无规则的额外信号（或信息），并且该种信号并不随原信号的变化而变化。

下面我将展示“信噪比”随着“所降低到的维度”变化，计算程序为 SNR_calculate.py



可见，随着维度的增大，信噪比同样在增大，说明所保留的信息在增多。

另外，在维度大于 65 之后，信噪比变化很小，说明我们的图像数据的前 65 个主成分提供了绝大多数的信息，其余的“次要成分”所能提供的信息十分有限，这也证明了 PCA 的重要性。

五、结论

PCA 算法通过寻找协方差矩阵的特征值最大的 K 个特征向量，以作为新的一组基，将原数据映射到这一组新的基上来完成数据的降维，也就是说 PCA 算法可以找到前 K 个主成分。

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“次要成分”，一定程度上起到了降噪的效果。

PCA 降低了是在训练集的基础上提取主成分，舍弃“次要成分”； 但是对于测试集而言，被舍弃的也许正好是重要的信息，也就是说 PCA 可能会加剧过拟合

PCA 可以应用到图像的降维压缩等领域，以提高效率，避免“维度灾难”

六、参考文献

1）Pattern Recognition and Machine Learning.
2）Gradient descent wiki
3）Conjugate gradient method wiki
4）Shewchuk J R. An introduction to the conjugate gradient method without the agonizing pain[J]. 1994.

七、附录：源代码（带注释）

data_generator.py

```
import numpy as np
import matplotlib.pyplot as plt
from config import *


# 单类点生成器
def generatePoints(center, rand_normal_x, rand_normal_y, rand_normal_z, nums_p):
    # 中心
    x_p = center[0]
    y_p = center[1]
    z_p = center[2]
    # 协方差矩阵
    cov = [[rand_normal_x, naive_rate_xy, naive_rate_xz], [naive_rate_xy, rand_normal_y, naive_rate_yz],
           [naive_rate_xz, naive_rate_yz, rand_normal_z]]
    # x 各维生成
    ps = np.random.multivariate_normal((x_p, y_p, z_p), cov, nums_p)
    return ps


ax3d = plt.subplot(projection="3d")
# 生成点并画出点图
matrix = generatePoints([0, 0, 0], rand_normal_x, rand_normal_y, rand_normal_z, nums)
x = [matrix[i][0] for i in range(matrix.shape[0])]
y = [matrix[i][1] for i in range(matrix.shape[0])]
z = [matrix[i][2] for i in range(matrix.shape[0])]
ax3d.scatter(x, y, z)
ax3d.set_xlabel("X Axis")
ax3d.set_ylabel("Y Axis")
ax3d.set_zlabel("Z Axis")
plt.legend(['data'])
plt.show()
np.savetxt(fname=data_file, X=matrix)
```

PCA.py

```
import numpy as np
import matplotlib.pyplot as plt
from config import *


# 获取数据
def get_data():
    matrix = np.loadtxt(data_file)
    return matrix


# 零均值化
def all_zero_mean(data):
    # 计算均值
    mean = np.array([np.mean(data[:, index]) for index in
range(data.shape[1])])
    data = data - mean
    return mean, data


# 计算协方差矩阵
def get_cov_matrix(data):
    return data.T @ data / data.shape[0]


# PCA 实现，返回降维后的数据，特征向量基和降维后重构的数据
def PCA(data, dimension_to_cut):
    # 零均值化
    all_zero_mean(data)
    # 计算协方差矩阵
    Cov = get_cov_matrix(data)
    # svd 计算特征值和特征向量
    u, s, vT = np.linalg.svd(Cov)
    # 取前 k 个向量
    vT = vT[:][:dimension_to_cut]
    # 压缩数据
    data = data @ vT.T
    # 返回降维后的数据，特征向量基和降维后重构的数据
    return data, vT, data @ vT


def show2D(data):
    # 平面图
```

```
        x = [data[i][0] for i in range(data.shape[0])]
        y = [data[i][1] for i in range(data.shape[0])]
        plt.scatter(x, y)
        plt.legend(['data'])
        ax = plt.subplot()
        ax.set_xlabel("X Axis")
        ax.set_ylabel("Y Axis")
        plt.show()


    def show3D(data, fvs, data_o):
        # 散点图
        ax3d = plt.subplot(projection="3d")
        x = [data[i][0] for i in range(data.shape[0])]
        y = [data[i][1] for i in range(data.shape[0])]
        z = [data[i][2] for i in range(data.shape[0])]
        ax3d.scatter(x, y, z)
        x = [data_o[i][0] for i in range(data_o.shape[0])]
        y = [data_o[i][1] for i in range(data_o.shape[0])]
        z = [data_o[i][2] for i in range(data_o.shape[0])]
        ax3d.scatter(x, y, z)
        ax3d.set_xlabel("X Axis")
        ax3d.set_ylabel("Y Axis")
        ax3d.set_zlabel("Z Axis")
        # 直线
        x = np.linspace(-10, 10, 20)
        y = np.linspace(-5, 5, 20)
        colors = ['r', 'g', 'b']
        for i in range(fvs.shape[0]):
            z = -(fvs[i][0] / fvs[i][2]) * x + -(fvs[i][1] / fvs[i][2])
* y
            ax3d.plot(x, y, z, colors[i])
        plt.legend(['data', 'pca_data', 'feature0', 'feature1'])
        plt.show()


    def main():
        data_set = get_data()
        print('原数据为')
        print(data_set)
        pca_data_set, fvs, data_o = PCA(data_set, dimension)
        print('经过降维后')
        print(pca_data_set)
        print('特征向量为')
```

```
        print(fvs)
        show3D(data_set, fvs, data_o)
        show2D(pca_data_set)


    if __name__ == '__main__':
        main()
```

face_pca.py

```
    import cv2
    import numpy as np
    import matplotlib.pyplot as plt
    from config import *
    from PCA import all_zero_mean, get_cov_matrix


    # PCA 实现，返回降维后重构的数据，特征向量基
    def PCA(data, dimension_to_cut):
        # 零均值化
        mean, data = all_zero_mean(data)
        # 计算协方差矩阵
        Cov = get_cov_matrix(data)
        # svd 计算特征值和特征向量
        u, s, vT = np.linalg.svd(Cov)
        # 取前 k 个向量
        vT = vT[:][:dimension_to_cut]
        # 压缩数据
        data = data @ vT.T
        # 还原数据
        data = data @ vT
        data += mean
        # 返回降维后重构的数据，特征向量基
        return data, vT


    # 从图像读取人脸数据集
    def get_faces():
        img = cv2.imread(faces_file, cv2.IMREAD_GRAYSCALE)
        face_list = []
        for i in range(col):
            for j in range(col):
                picture = img[i * single_pic_size:i * single_pic_size +
single_pic_size,   j   *   single_pic_size:j   *   single_pic_size   +
single_pic_size]
                picture                                                 =
```

```
picture.reshape(single_pic_size*single_pic_size, )
                face_list.append(picture)
        return np.array(face_list)


    # 将人脸数据集恢复为图像
    def go_back(pca_faces):
        result = []
        for face in pca_faces:
            face = face.reshape(single_pic_size,single_pic_size)
            result.append(face)
        picture       =       np.array([0.0       for       i       in
range(single_pic_size*col*single_pic_size*col)])
        picture                                                    =
picture.reshape(single_pic_size*col,single_pic_size*col)
        for i in range(col):
            for j in range(col):
                picture[i  *  single_pic_size:i  *  single_pic_size  +
single_pic_size,   j   *   single_pic_size:j   *   single_pic_size   +
single_pic_size]+=result[col*i+j]
        return picture


    # 主程序
    def main():
        faces = get_faces()
        pca_faces = PCA(faces, face_dimension)[0]
        solved_faces = go_back(pca_faces)
        solved_faces = solved_faces.astype('uint8')
        cv2.imshow('img.jpg', solved_faces)
        cv2.waitKey(0)
        cv2.imwrite('faces/img{}.jpg'.format(face_dimension),
solved_faces)


    if __name__ == '__main__':
        main()
```

SNR_calculate.py

```
    import cv2
    import matplotlib.pyplot as plt
    from numpy import log10

    img_ori = r'faces/newpic.jpg'
    img = r'faces/img{}.jpg'
```

```
    # 设置字形
    plt.rc('font', family='SimHei')
    plt.rc('axes', unicode_minus=False)


    def get_SNR(img_ori, img):
        img_ori = cv2.imread(img_ori, cv2.IMREAD_GRAYSCALE)
        img = cv2.imread(img, cv2.IMREAD_GRAYSCALE)
        c = img - img_ori
        c = c.reshape(c.shape[0] * c.shape[1], )
        var_c = sum(c**2)
        img = img.reshape(img.shape[0] * img.shape[1], )
        var_img = sum(img**2)
        return 10 * log10(var_img / var_c)


    xs = [1, 5, 10, 20, 30, 40, 50, 60, 65, 75, 100, 125, 150, 175, 200,
255]
    ys = [get_SNR(img_ori, img.format(x)) for x in xs]
    plt.plot(xs, ys, 'r*-')
    ax = plt.subplot()
    ax.set_xlabel("主成分维度")
    ax.set_ylabel("信噪比/dB")
    plt.title('图像信噪比随降维数变化折线图')
    plt.show()
```

config.py

```
    # 单类数据量
    nums = 1000
    # 数据标准差
    rand_normal_x = 20
    rand_normal_y = 1
    rand_normal_z = 20
    # 协方差参数
    naive_rate_xy = 0
    naive_rate_xz = 0
    naive_rate_yz = 0
    # 降维数
    dimension = 2
    face_dimension = 65
    single_pic_size = 32
    col = 8
    # 文件路径
    data_file = 'data.csv'
```

```
faces_file = 'faces/newpic.jpg'
```