
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 | 赵仁杰 |
| 学号 | 1180300113 |
| 班号 | 1803001 |
| 电子邮件 | 1579974122@qq.com |
| 手机号码 | 15122925619 |

# 目录



# 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、环境

## 2.1 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）。

## 2.2 实验环境

x86-64,Win 10

Pycharm 2019.1

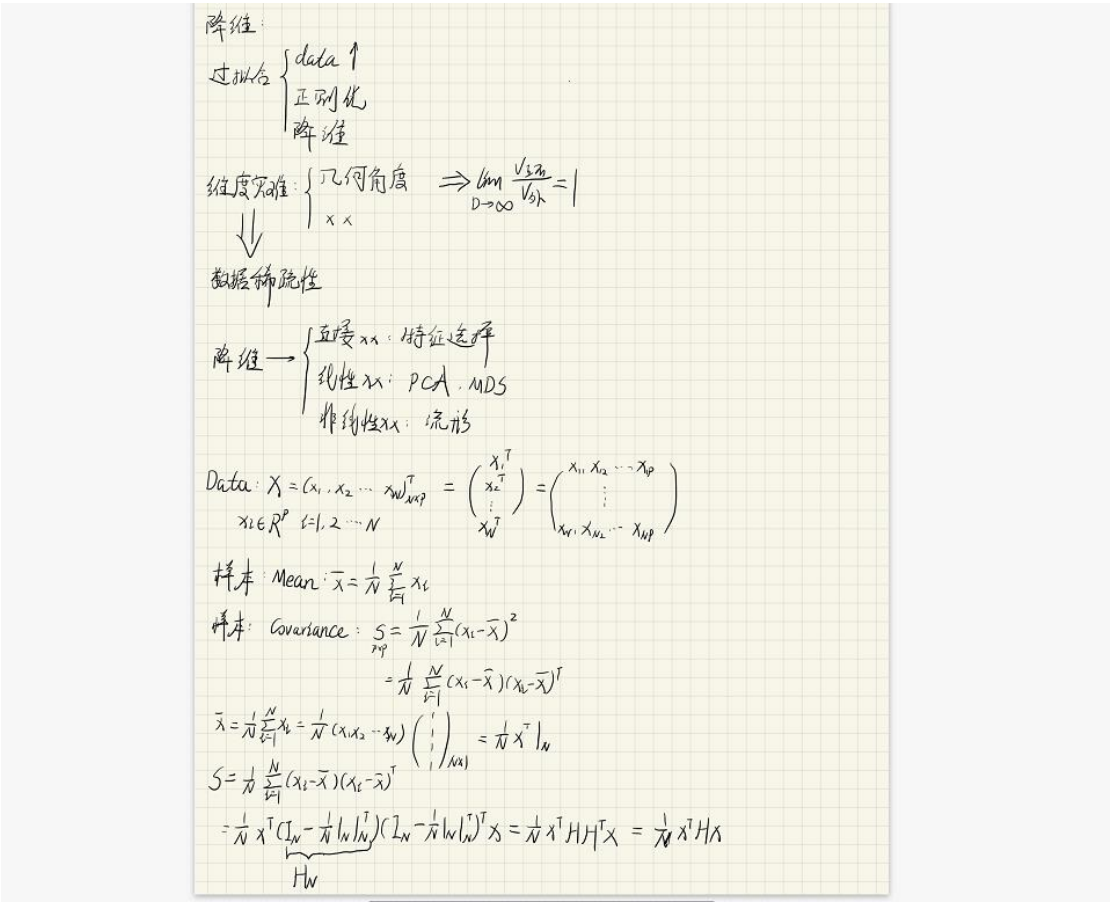
python 3.7

# 3 设计思想

## 3.1 降维介绍

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，去除噪声和一些不重要的特征。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。




上图简略的展示了解决过拟合的几种方法，以及样本均值、样本方差的一些简单的推导。

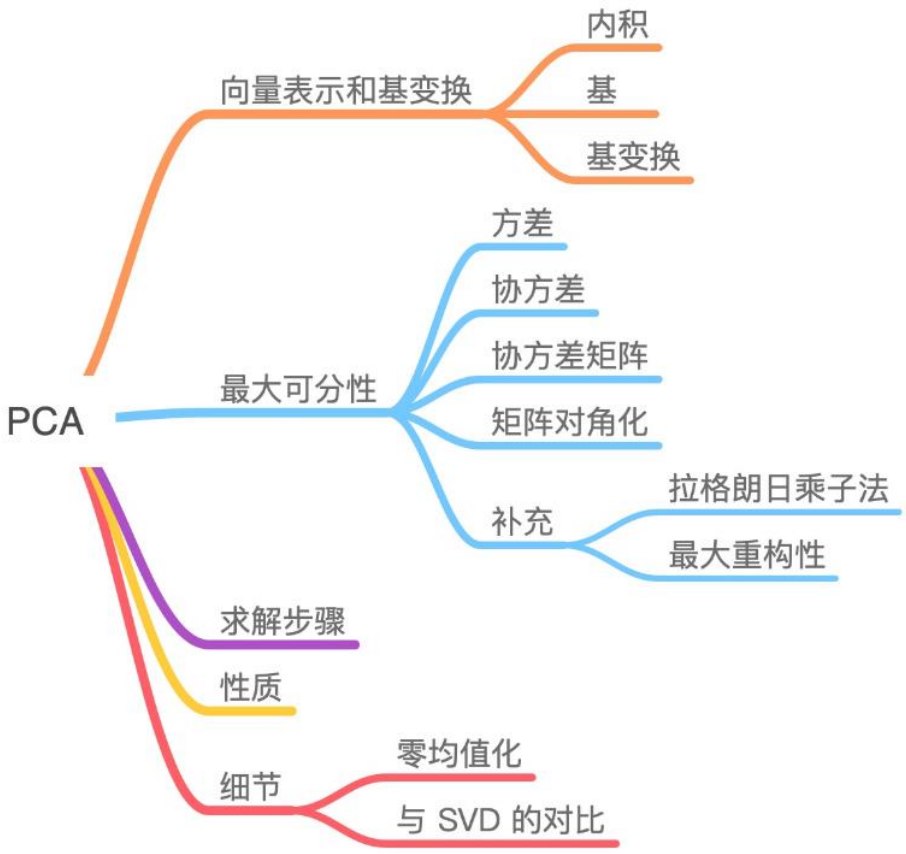
## 3.1.1 降维的意义、分类

上面展示了因维度的升高产生的维度灾难，具体定义为维数灾难：在给定精度下，准确地对某些变量的函数进行估计，所需样本量会随着样本维数的增加而呈指数形式增长。

降维的意义：克服维数灾难，获取本质特征，节省存储空间，去除无用噪声，实现数据可视化

数据降维分为特征选择和特征提取两种方法，次实验是特征提取方法，即经已有特征的某种变换获取约简特征。

## 3.2 线性降维方法：PCA 介绍



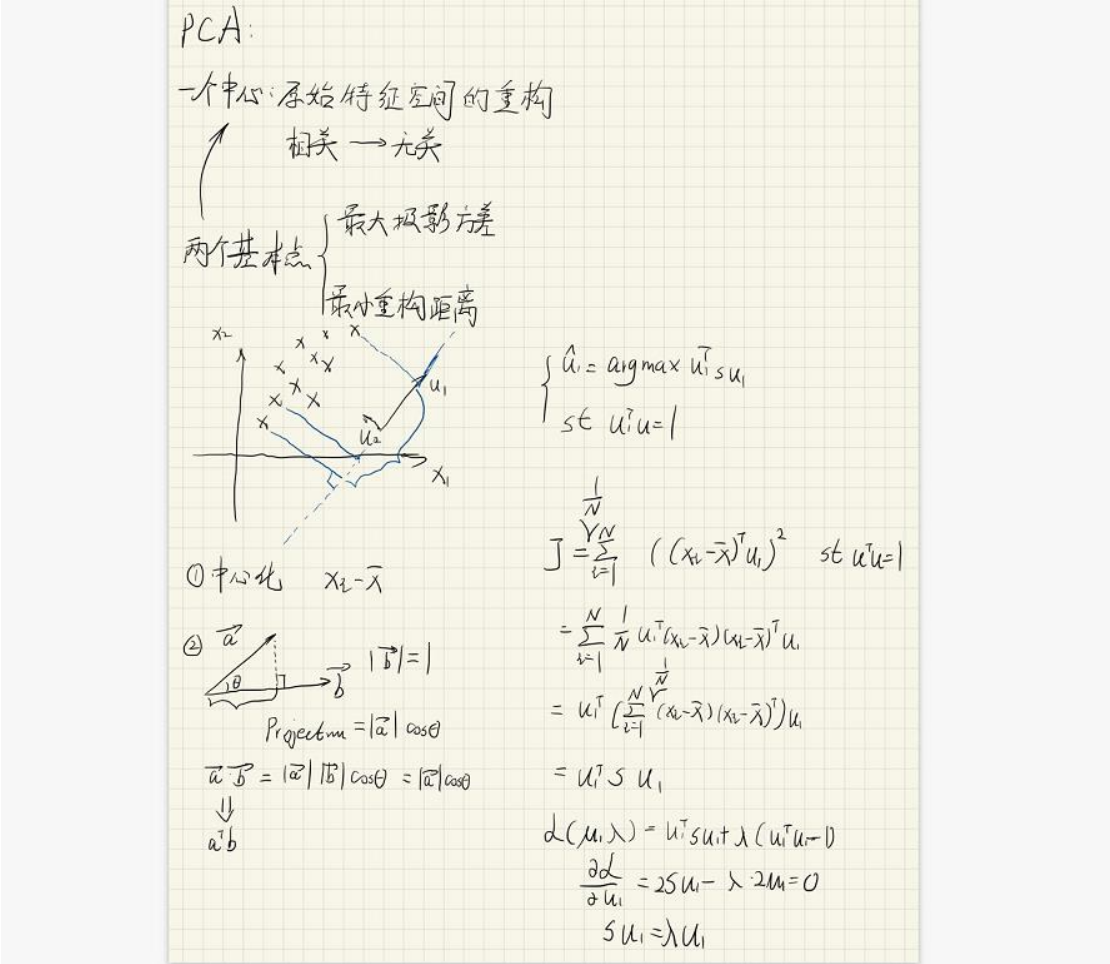

PCA 原理结构如上图所示，下面仅从几个方面去阐述这个原理，接受优缺点以及意义。

PCA（主成分分析）：通过坐标变换将 D 维数据降维到 K 维数据，其中 K 维的基向量为正交向量，为了保证降维后数据信息量，我们尽量采取能够保留原数据特征的线性变换。其中有两种对于信息量的解释（两者具有相同的原理，只不

过阐述的角度有所不同)：

最大投影方差：样本点在这个超平面上的投影尽可能分开

最小投影距离：样本点到这个超平面的距离都足够近




上图是有关最大投影方差的简单推导。

其中包括 PAC 里面需要的线代知识，拉格朗日乘子法以及基底变化等。

## 3.2.1 求解步骤

这里的求解步骤与上面推导过程略有不同：

设有 m 条 n 维数据。

1:将原始数据按列组成 n 行 m 列矩阵 X;

2: 将 X 的每一行进行零均值化，即减去这一行的均值;

3: 求出协方差矩阵 C=(1/m)X.TX ;

4:求出协方差矩阵的特征值及对应的特征向量;

5:将特征向量按对应特征值大小从上到下按行排列成矩阵，取前 k 行组成矩阵 P;

6:Y=PX 即为降维到 k 维后的数据。

### 3.2.2 PCA 性质

降噪：当数据受到噪声影响时，最小特征值对应的特征向量往往与噪声有关，将它们舍弃能在一定程度上起到降噪的效果;

过拟合：PCA 保留了主要信息，但这个主要信息只是针对训练集的，而且这个主要信息未必是重要信息。有可能舍弃了一些看似无用的信息，但是这些看似无用的信息恰好是重要信息，只是在训练集上没有很大的表现，所以 PCA 也可能加剧了过拟合;

特征独立：PCA 不仅将数据压缩到低维，它也使得降维之后的数据各特征相互独立;

### 3.2.3 最小重构距离

这里只是进行简述，具体的不做细微证明。

最大投影方差是通过一条直线使得样本点投影到该直线上的方差最大。除此之外，我们还可以将其转换为线型回归问题，其目标是求解一个线性函数使得对应直线能够更好地拟合样本点集合。这就使得我们的优化目标从方差最大转化为平方误差最小，因为映射距离越短，丢失的信息也会越小。区别于最大可分性，这是从最近重构性的角度进行论证。

# 4：算法实现

## 4.1 中心化:

在求解数据之前首先对数据进行中心化，首先求得数据集的平均值

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

此后对每个数据进行操作是的 X = X-μ，通过去中心使得μ值为 0

```
mean = 1.0 / rows * np.sum(data, axis=0)
# 把每一个数据的均值都变为 0
X = data - mean
```

2 求协方差

由于已经中心化，所以协方差矩阵等于 X*X.T

Cov = np.dot(data,data.T)

3 求特征值特征向量

使用 numpy 自带函数操作即可，这里要用 argsort 对其排序

```
# 特征值分解
eigenvalues, feature_vectors = np.linalg.eig(cov)
# 排序
min_d = np.argsort(eigenvalues)
```

4 取 k 返回转换矩阵以及数据处理即可

```
feature_vectors = np.delete(feature_vectors, min_d[:columns -
dimension], axis=1)
return feature_vectors, mean
```

# 5 实验结果分析：

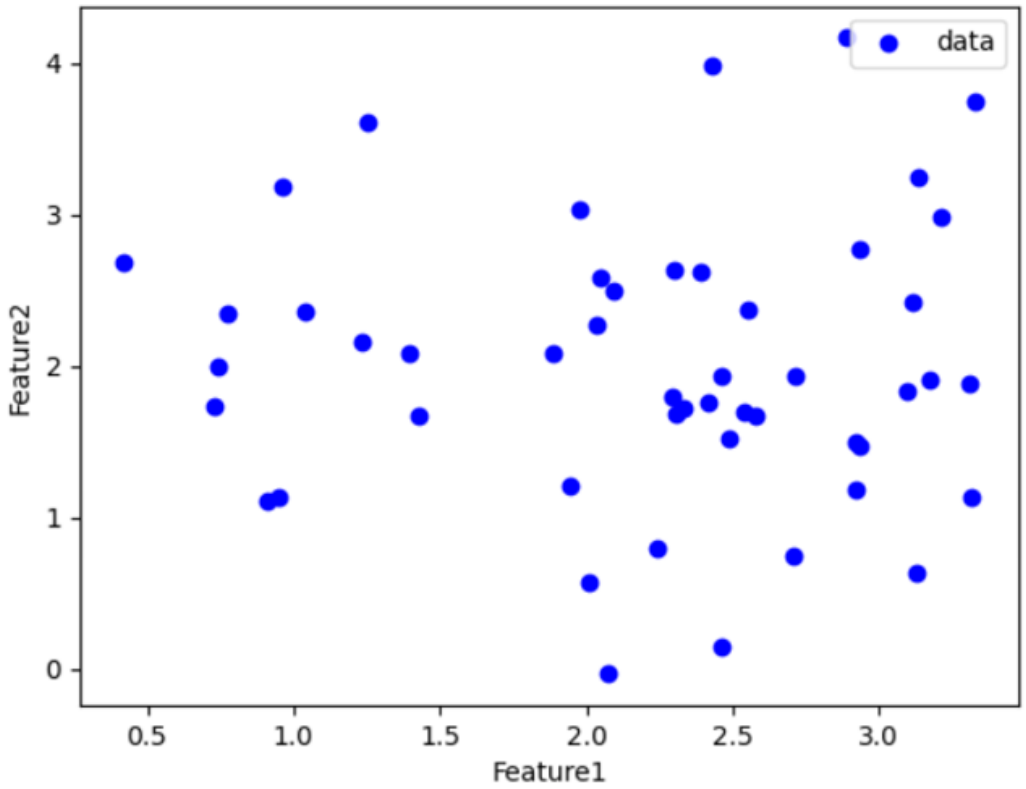
由于进行数据可视化，这里针对二维数据和三位数据进行降维处理，便于理解。

## 5.1 生成数据:

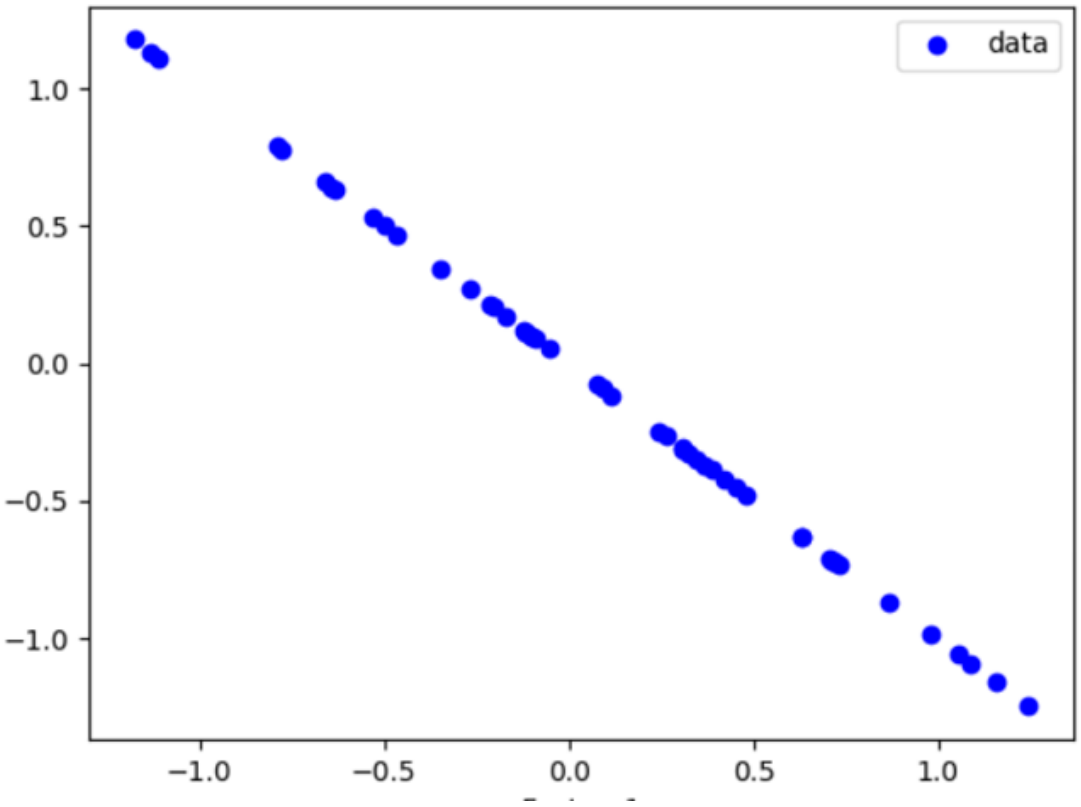
### 5.1.1 生成数据-2 维：

降维前:

由下面的数据进行 PCA 降维，依照最大方差距离的选择，选取包含更多信息的维度，得到下面的图像。


data
Feature2
Feature1
4
3
2
1
0
0.5
1.0
1.5
2.0
2.5
3.0

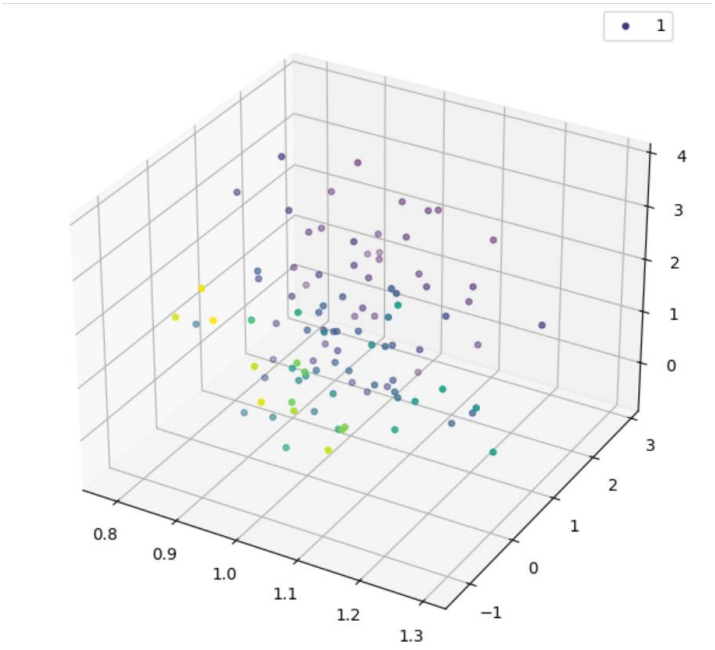
降维后：


data
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0

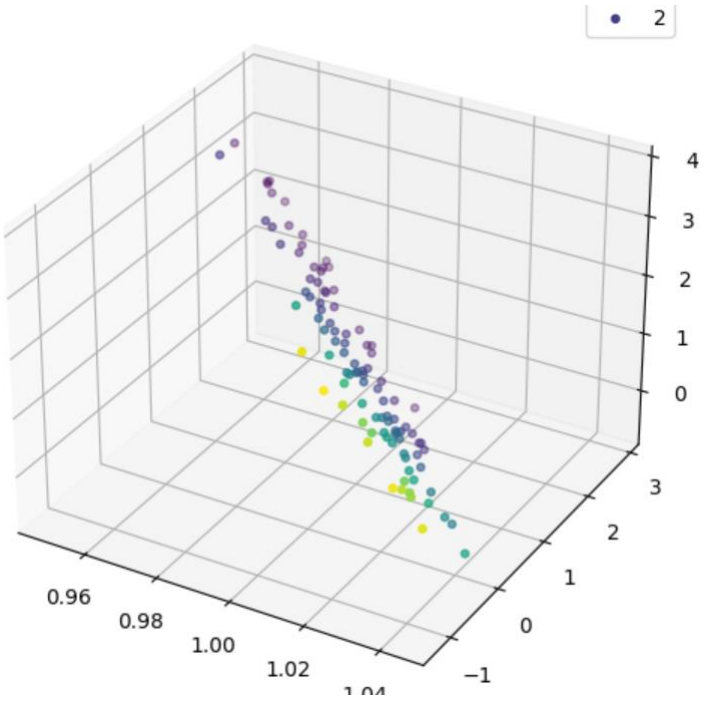
可以看到，分布在直线上的一维数据，其方差是最大的。

## 5.1.2 生成数据-三维

降维前：




降维后：




可以看到第一位的方差是远小于其他两个维度的，从而得出结论，在第一维

相对于其他两个维度信息更少，所以得出上述结果。

## 5.2 人脸数据测试：

处理思路：首先利用 cv2 对图片的灰度进行分析，得到一个关于灰度的数组，由于像素较高，对其进行压缩，得到高维数组，由于数组形式是 m*n 的所以将其转换成 mn*1 的形式满足 mn 维的向量，对其进行 PCA 处理最后还原成 m*n 的数据最后使用图片显示函数将其显示出即可。

信噪比计算公式：

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

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

PS：之前我将电脑的内置软件删了一些，做完才发现打开 bmp 格式的图片很困难，进行对比更困难。

下面是降维前后的对比：

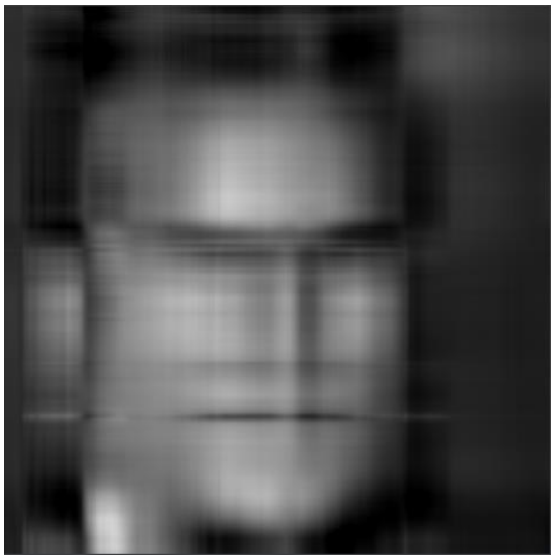
原图：

k=50

K =10

k=5

K=3



数据分析：对于该人脸照片，如果我们把它降到 50 的情况下，它的图片特征没有什么改变，只是转化为黑白照片；当 k=10 的化，出现了肉眼可见的损失；当 k=3 的话，损失的更为严重，只能看出是个人脸以及轮廓。

有关信噪比的对比下，不同的人脸数据会得出不一样的数值，但是可以总结为：当背景影响越大的情况下，它的信噪比会更大一点。

# 6 结论

1PCA 保留了训练数据的维度的同时保留了主要信息，从图片处理可以见到 PCA 对于信息提取能力还是很强的。

2：由上面的结论进行衍生，主要信息不一定是最为重要的信息，被舍弃的也可能是比较重要的信息。

3：PCA 对于图片的降维会缓解存储压力并且实现清晰度的保证。

4:因为 PCA 舍弃了排序后小的特征值对应的特征向量，转变为低维空间，但是这样做到了样本采样密度的提升，并且在此同时还实现了降噪。

```
import struct as st

'''做图片的读取'''
class BMP(object):
    def __init__(self, filename):
        with open(filename, 'rb') as f:
            #我们需要的只有宽度和高度两个值，其他的 46 个字节直接一起读出来就可以了
            self.d1 = st.unpack("<18s", f.read(18))[0]
            # 说明图像的宽度，单位为像素
            self.biWidth = st.unpack("<i", f.read(4))[0]
            # 说明图像的高度，单位为像素，为正时表示数据排列是从左下按行到右
            self.biHeight = st.unpack("<i", f.read(4))[0]
            self.d2=  st.unpack("<28s",f.read(28))[0]

            self.data = []
            #用一个二维数组来存储，不过由于一个像素的三个字节要用一个数组，所以实际上应该是一个三维的
            for i in range(self.biHeight):
                count = 0
                self.data.append([])
                for j in range(self.biWidth):
                    pixel = []
                    pixel.append(st.unpack("<B", f.read(1))[0])
                    pixel.append(st.unpack("<B", f.read(1))[0])
                    pixel.append(st.unpack("<B", f.read(1))[0])
                    self.data[i].append(pixel)
                    count = count + 3
```

```
            #四字节对齐，把末尾补的零删去
            while count % 4 != 0:
                print(i)
                f.read(1)
                count = count + 1

        f.close()

    def NewBmp(self, filename):
        # 用已有数据创建一个 bmp 图像并输出，并输出到指定路径
        file = open(filename, 'wb+')
        #创建图像时，前 54 个字节直接照原来的填
        file.write(st.pack("<18s", self.d1))
        file.write(st.pack("<i", self.biWidth))
        file.write(st.pack("<i", self.biHeight))
        file.write(st.pack("<28s", self.d2))

        count = 0
        for i in range(len(self.data)):
            for p in self.data[i]:
                file.write(st.pack("<B", p[0]))
                file.write(st.pack("<B", p[1]))
                file.write(st.pack("<B", p[2]))
                count = count + 3
                #四字节对齐
                if count == self.biWidth * 3:
                    while (count % 4 != 0):
                        file.write(st.pack("<B", 0))
                    count = 0
        file.close()
```

```
'''生成数据，进行 PCA，以及修改参数对不同的照片进行 PCA 降维'''
import numpy as np
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import BMP


def draw_3D(data,name):
    fig = plt.figure()
    ax = Axes3D(fig)
    y3 = np.arctan2(data[:, 0], data[:, 1])
    ax.scatter(data[:, 0], data[:, 1], data[:, 2],
```

```
            c=y3, marker='.', s=50, label=name)
    plt.legend()
    plt.show()

def pca(data,dimension):
    rows, columns = data.shape
    assert dimension <= columns
    mean = 1.0 / rows * np.sum(data, axis=0)
    # 把每一个数据的均值都变为 0
    X = data - mean
    # 计算协方差
    cov = 1.0 / columns *X.T.dot(X)
    # 特征值分解
    eigenvalues, feature_vectors = np.linalg.eig(cov)
    # 排序
    min_d = np.argsort(eigenvalues)
    feature_vectors = np.delete(feature_vectors, min_d[:columns -
dimension], axis=1)
    return feature_vectors, mean

a=int(input())
if(a==0):
    mean = [1, 1, 1]
    cov = [[0.01, 0, 0], [0, 1, 0], [0, 0, 1]]
    data1 = np.random.multivariate_normal(mean, cov, 100)
    w1, mu_data1 = pca(data1, 2)
    pca_data1 = (data1 - mu_data1).dot(w1)
    pca_data_inverse1 = pca_data1.dot(w1.T) + mu_data1
    draw_3D(data1,"1")
    draw_3D(pca_data_inverse1,"2")

else:
    # 读取 BMP 文佳并进行降维
    bmp = BMP.BMP("1.bmp")
    data = []
    for i in range(len(bmp.data)):
        data.append([])
        for pixel in bmp.data[i]:
            data[i].append(pixel[0])

    data = np.array(data)

    w, mu_data = pca(data, 3)
    pca_data = (data - mu_data).dot(w)
```

```
    pca_data_inverse = pca_data.dot(w.T) + mu_data
    for i in range(len(bmp.data)):
        for j in range(len(bmp.data[i])):
            if (pca_data_inverse[i][j] > 255):
                pca_data_inverse[i][j] = 255
            if (pca_data_inverse[i][j] < 0):
                pca_data_inverse[i][j] = 0
    for i in range(len(bmp.data)):
        for j in range(len(bmp.data[i])):
            bmp.data[i][j][0] = round(pca_data_inverse[i][j])
            bmp.data[i][j][1] = round(pca_data_inverse[i][j])
            bmp.data[i][j][2] = round(pca_data_inverse[i][j])

    bmp.NewBmp("2.bmp")
```