# 哈尔滨工业大学计算机科学与技术学院

# 实验报告

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：1190301610

姓名： 王家琪

## 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：ios，pycharm，python3.7

## 三、设计思想（本程序中的用到的主要算法及数据结构）

## 3.1、实验原理

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。

## 3.2、去中心化

在开始PCA之前需要对数据进行预处理，也就是数据中心化，简而言之就是让数据的均值为0，几何意义是将坐标原点放到中心。这样方便之后计算方差。

设数据为$X=\{x_1,x_2,.....,x_n\}$,其中$x_i=\{x_{i1},x_{i2},.....,x_{id}\}$,则$X$为$n\times d$的矩阵，则此数据集的中心向量为：$\mu=\frac{1}{n}\sum_{i=0}^{n}x_i$,对每个样本进行操作$x_i=x_i-\mu$,得到了中心化之后的数据，此时有$\sum_{i=0}^{n}x_i=0$。中心化之后的数据可以为之后的运算带来便利，比如计算协方差矩阵的时候，直接可以用$S=\frac{1}{n}X^TX$来计算。

## 3.3、协方差矩阵和散度矩阵

下面我们来补充一些代数的相关知识。样本的均值为$\mu=\frac{1}{n}\sum_{i=0}^{n}x_i$,样本的方差$S^2=\frac{1}{n-1}\sum_{i=0}^{n}(x_i-\mu)^2$,样本X和样本Y的协方差为$Cov(X,Y)=E[(X-E(X))(Y-E(Y))]=\frac{1}{n-1}\sum_{i=0}^{n}(x_i-\mu_x)(y_i-\mu_y)$

一般的，对于三维数据$(x,y,z)$,他们的协方差矩阵为：

$$Cov(X,Y,Z)=\begin{bmatrix}Cov(x,x) & Cov(x,y) & Cov(x,z)\\ Cov(y,x) & Cov(y,y) & Cov(y,z)\\ Cov(z,x) & Cov(z,y) & Cov(z,z)\end{bmatrix}$$

## 3.4、基于特征值分解协方差矩阵实现PCA算法

我们接下来要讨论PCA实现的具体步骤，在1、实验原理里面我们知道，PCA降维的本质是找到k个含有比较大方差的相互正交的坐标轴，将方差比较小的坐标轴上的数据舍弃掉，将现在的点投影到新的还有k个坐标轴的坐标系里面。

所以现在的关键在于怎么确定这k个坐标向量。实际上，这k个坐标向量就是原数据协方差矩阵的k个特征向量，并且这k个特征向量对应的是前k大的特征值。我们知道矩阵的特征向量是正交的，可以认为是一组正交的基底，可以理解为特征向量对应的特征值表示的是点在该特征向量上的分布情况，特征值越大，点在这个特征向量上的信息也就越多，也就是我们之前说的方差越大。

具体算法如下：

输入：数据集$X=\{x_1,x_2,.....,x_n\}$，降到的维数k

1、取平均值，即去中心化，计算$\mu = \frac{1}{n}\sum_{i=0}^{n} x_i$，令$x_i = x_i - \mu$。

2、计算协方差矩阵$S = \frac{1}{n}X^T X$，这里除或者不除n或者n-1对特征向量没有影响。

3、计算协方差矩阵$S$的特征值以及特征向量

4、对特征值从大到小排序，选择其中最大的k个，然后将他们对应的特征向量分别作为行向量，组成特征向量矩阵$P$。

5、将数据降到k维，即计算$Y = PX$。

# 四、实验结果与分析

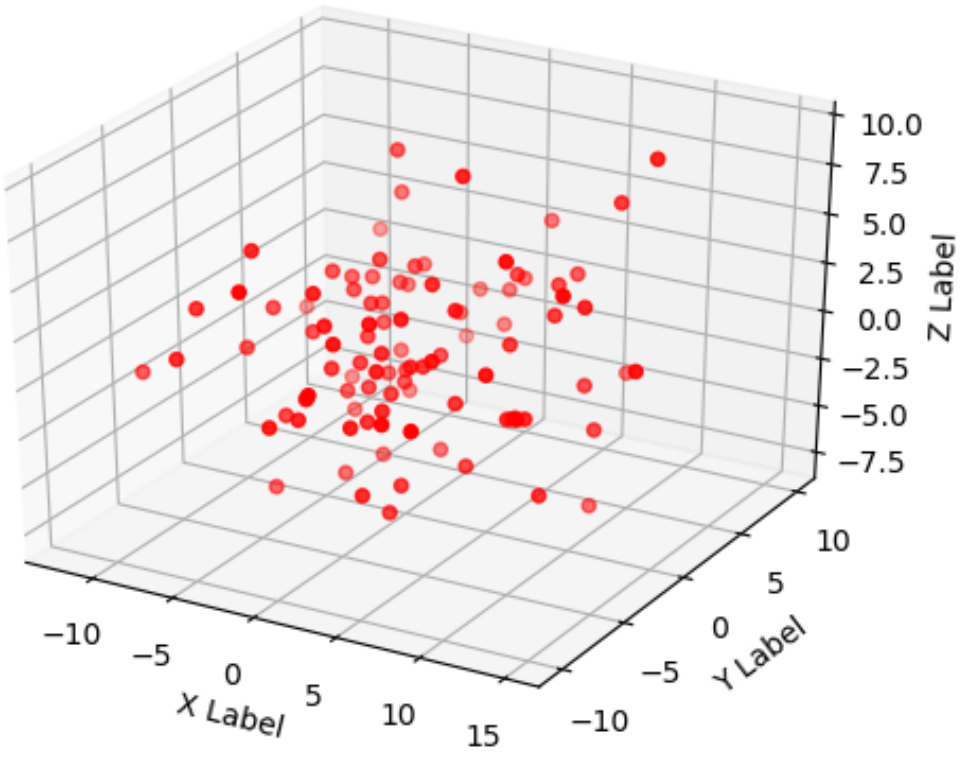
## 4.1、手工数据验证

为了方便进行数据可视化，这里只进行了三维数据降到二维的PCA实验前后对比。

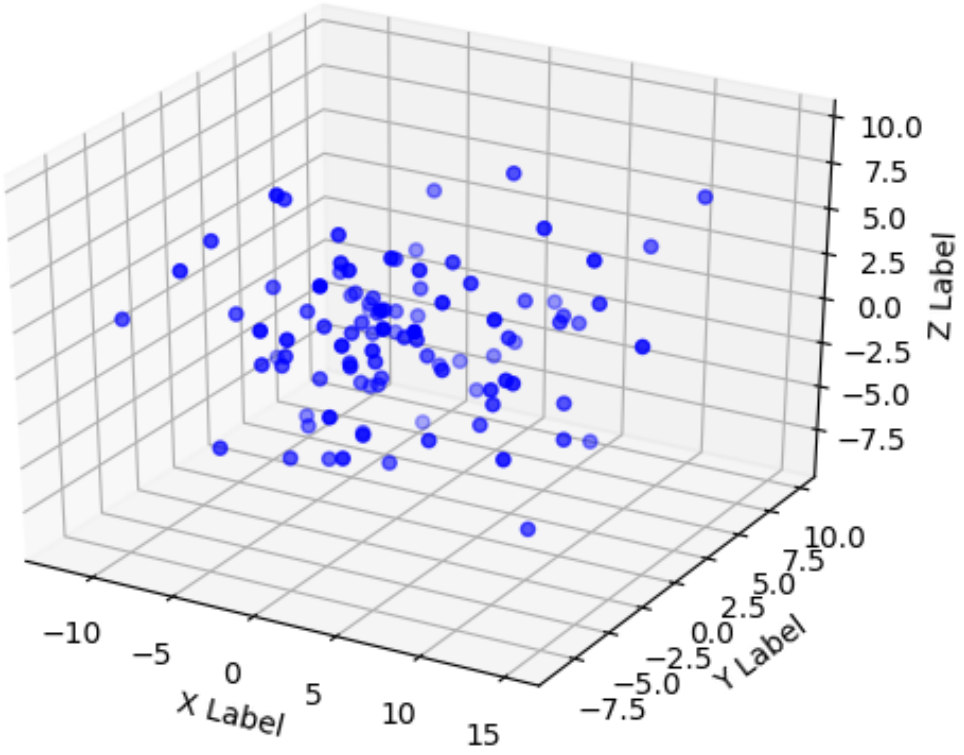
生成的高斯分布数据的参数为：

$$mean = [0, 0, 0], Cov = \begin{bmatrix} 40 & 0 & 0 \\ 0 & 30 & 0 \\ 0 & 0 & 10 \end{bmatrix}$$

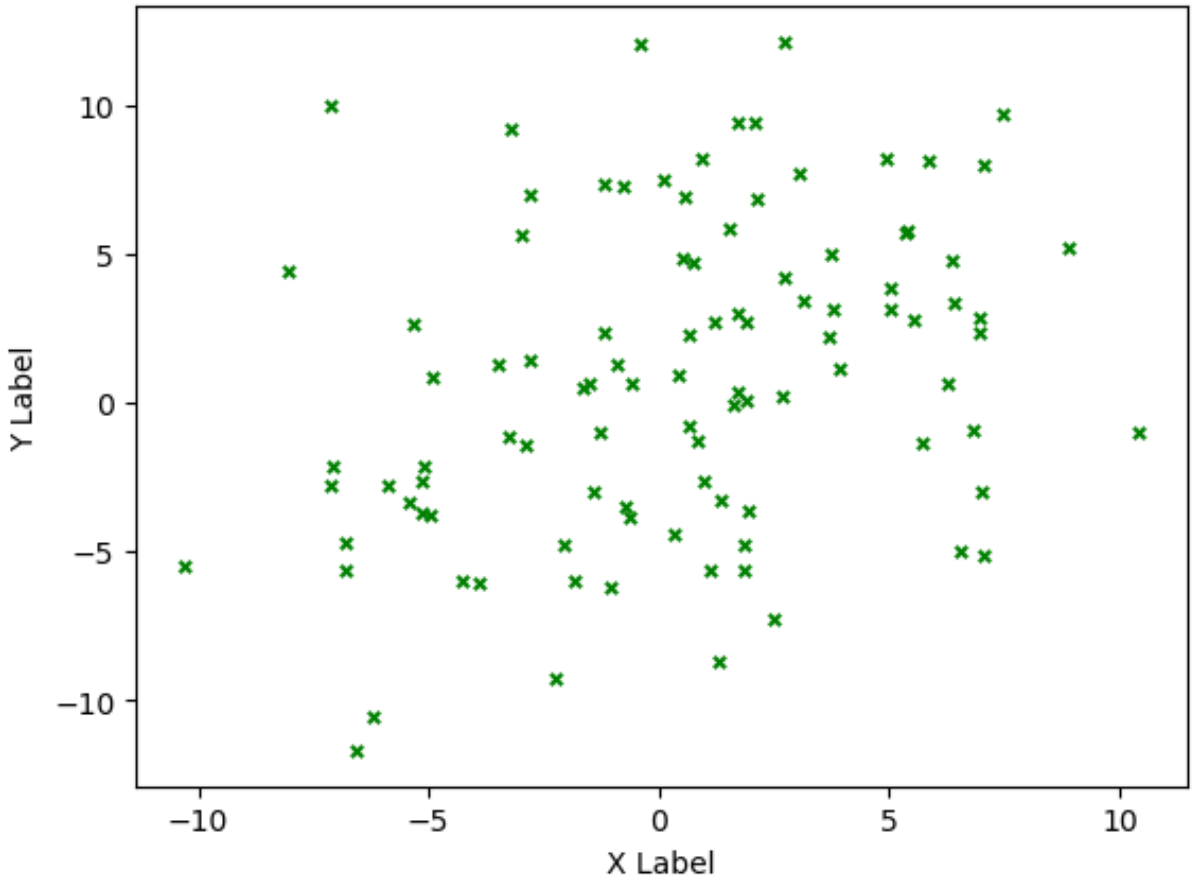
为了方便验证我们将两个维度方差设定的比另一个维度要大很多。数据可视化为：



将数据旋转了$30^o$,得到：




PCA降到二维的数据为：




## 4.2、人脸数据验证

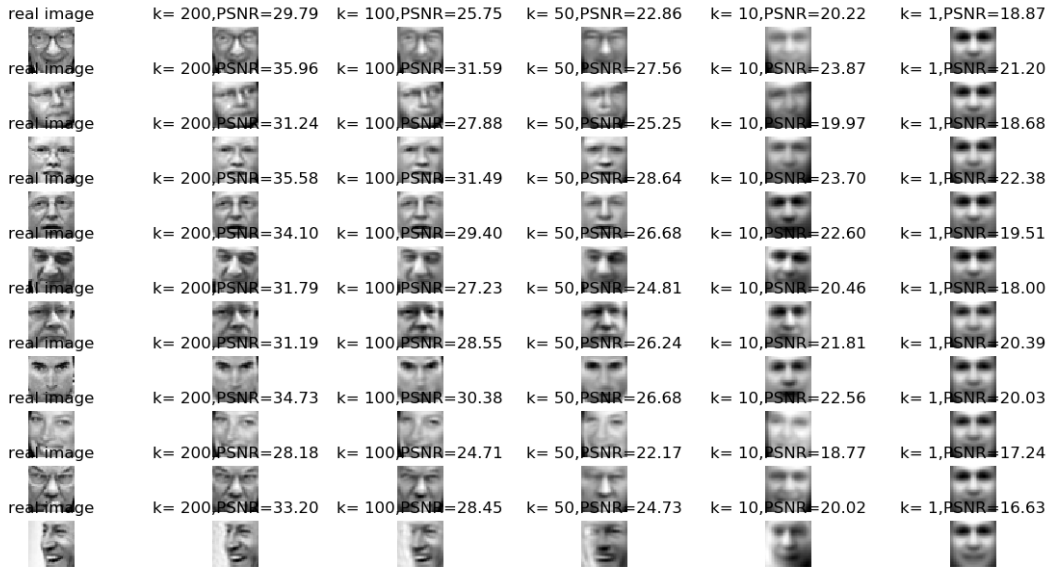
我们应用5000组人脸数据，数据的特征维度为1024（转换为32×32的图像格式）来进行验证。相当于数据的大小为$5000 \times 1024.$

信噪比的计算公式为：

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

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

我们可视化展示10组人脸数据，并将它们分别降到200，100，50，1维，计算相应的信噪比。效果图如下：




通过结果可以看出，在降到100维的时候，图像的特征可以很好的保留，人眼几乎分辨不出来有损失。在降到50维及其以下的时候，图片就有些模糊了，当降到10维及以下的时候，人脸的特征已经很模糊了，分辨不出不同的人之间的区别了。

但是从信噪比的角度看，人脸的特征降维之后的信噪比一直是降低的，可以看出降维之后信噪比更大了，也就意味着图片的信息更少，看起来也更模糊了。

# 五、结论

1、PCA降低了数据的维度，保留了有效的信息，一定程度上可以节约资源，但是舍弃的数据不能太多，降维太多会导致数据损失太多，而无法复原，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$d-k$个最小的特征值对应的特征向量，一定会导致低维和高纬空间不同，但是通过这种方式提高了采样的密度，并且由于小特征值对应的往往和噪声有关，通过PCA可以起到降噪的作用。

3、PCA用于图片的降维可以极大地缓解存储压力，尤其是在如今像素越来越高的情况下。使用PCA降维我们只需要存储三个比较小的矩阵，能够较大地节省存储空间。

# 六、参考文献

矩阵旋转https://www.cnblogs.com/meteoric_cry/p/7987548.html

主成分分析https://zhuanlan.zhihu.com/p/37777074

# 七、附录：源代码（带注释）

1、手工生成数据代码

```
\begin{lstlisting}
import numpy as np
import matplotlib as mpl
```

```
mpl.use('TkAgg')
import matplotlib.pyplot as plt

#手工生成三维数据
def generateData(path,dim=3):
    f = open(path, 'w')
    #各维度均值为0
    mean = [0,0,0]
    cov = np.mat([[40, 0, 0], [0, 30, 0],[0 ,0 ,10]])
    x = np.random.multivariate_normal(mean, cov, 100)
    for i in range(len(x)):
        line=[]
        line.append("{:.3f}".format(x[i][0]))
        line.append("{:.3f}".format(x[i][1]))
        line.append("{:.3f}".format(x[i][2]))
        line=','.join(str(i) for i in line)
        line=line+'\n'
        f.write(line)

#绘制三维散点图，输入的是文件地址
def visualData(path):
    #绘制图框
    xs=[]
    ys=[]
    zs=[]
    with open(path,'r')as file:
        for line in file:
            l=line.split(',')
            xs.append(float(l[0]))
            ys.append(float(l[1]))
            zs.append(float(l[2]))
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    ax.scatter(xs, ys, zs, s=20, c='r',marker='o')
    ax.set_xlabel('X Label')
    ax.set_ylabel('Y Label')
    ax.set_zlabel('Z Label')

#3维的数据可视化，输入的是矩阵
def visualMatRotate(data):
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    ax.scatter(data[0], data[1], data[2], s=20, c='b', marker='o')
    ax.set_xlabel('X Label')
    ax.set_ylabel('Y Label')
    ax.set_zlabel('Z Label')
    plt.show()

#2维的数据可视化
```

```
def visualLowdata(data):
    plt.scatter(np.array(data[0]),np.array(data[1]),s=15,c='g',marker='x')
    plt.xlabel('X Label')
    plt.ylabel('Y Label')
    plt.show()

#利用特征值法降维
def PCA(datapath,dim=3,lowdim=2):
    data=[]
    k=lowdim #降维的维度
    with open(datapath, 'r')as file:
        for line in file:
            l = line.split(',')
            x = []
            x.append(float(l[0]))
            x.append(float(l[1]))
            x.append(float(l[2]))
            data.append(x)
    data=np.mat(data).T
    alpha=np.pi/6 #旋转角度
    #沿x轴旋转
    rotate=[[1,0,0],[0,np.cos(alpha),np.sin(alpha)],[0,-np.sin(alpha),np.cos(alpha)]]
    datarotate=np.dot(rotate,data)
    #visualMatRotate(datarotate)
    #计算协方差矩阵
    covRoData=np.cov(rotate)
    #计算特征值，特征向量
    eigenvalue, featurevector = np.linalg.eig(covRoData)
    #上面得到的sorted_indices就是特征值排序后的结果，
    # 巧妙的是这里是用数组下标来表示的，
    # 也就是说其中存放的是特征值由小到大的顺序排序时对应的下标[0, 4, 1, 3, 2]，而不是直接存放特征值。
    eigenvalueIndex=np.argsort(eigenvalue)
    #提取前k大的特征向量
    topk_evecs = featurevector[:, eigenvalueIndex[:-k - 1:-1]]
    topk_evecs=topk_evecs.T
    #降维
    dataLowDim=np.dot(topk_evecs,datarotate)
    return dataLowDim


datapath='exp4_3_dim.txt'
#generateData(datapath)
#visualData(datapath)
dataLowDim=PCA(datapath)
visualLowdata(dataLowDim)
\end{lstlisting}
```

2、人脸PCA数据代码

```
\begin{lstlisting}
import numpy as np
import matplotlib as mpl
mpl.use('TkAgg')
import matplotlib.pyplot as plt
import scipy.io as io
mat_path="ex7faces.mat"
def pca(x, k):
    """
    PCA
    """
    n = x.shape[0]
    mu = np.sum(x, axis=0) / n
    x_centralized = x - mu #去中心化
    #计算协方差矩阵
    cov = (x_centralized.T @ x_centralized) / n
    #计算特征值，特征向量
    values, vectors = np.linalg.eig(cov)
    index = np.argsort(values) # 从小到大排序后的下标序列
    # 提取前k大的特征向量
    vectors = vectors[:, index[:-(k+1):-1]].T # 把序列逆向排列然后取前k个，转为行向量
    return x_centralized, mu, vectors

def readface(mat_path):
    x=io.loadmat(mat_path)
    #dict_keys(['__header__', '__version__', '__globals__', 'X'])
    #5000,1024
    a= x['X']
    a=np.mat(a)
    return a

def face_pca(mat_path,k_list):
    '''

    :param mat_path: 文件路径
    :param k_list: 降到不同的维度
    :return:
    '''
    facedata=readface(mat_path)
    Psnrlist=[]
    Facedata_lowdim=[]
    for k in k_list:
        x_centralized, mu, vectors=pca(facedata,k)
        facedata_lowdim=x_centralized @vectors.T @vectors +mu #数据重建
        Facedata_lowdim.append(facedata_lowdim)
        #计算信噪比
        psnrlist=[psnr(facedata[i],facedata_lowdim[i]) for i in
range(np.shape(facedata)[0])]
        Psnrlist.append(psnrlist)
```

```
    show_facesum(facedata,Facedata_lowdim,k_list,Psnrlist)#i 表示前10张

def psnr(source, target):
    """
    计算峰值信噪比
    """
    source=np.array(source)
    target=np.array(target)
    rmse = np.sqrt(np.mean((source - target) ** 2))
    return 20 * np.log10(255.0 / rmse)

def show_facesum(x, x_pca_list, k_list, x_psnr_list):
    people=10
    plt.figure(figsize=(12,8),frameon=False)
    for i in range(people):
        plt.subplot(10,len(k_list)+1,(len(k_list)+1)*i+1)
        plt.title('real image',)
        plt.imshow(x[i].reshape(32,32).T,cmap='gray')
        plt.axis('off')
    for j in range(people):
        for i in range(len(k_list)):
            plt.subplot(people,len(k_list)+1,(len(k_list)+1)*j+i+2)
            plt.title('k= '+str(k_list[i])+',PSNR='+'{:.2f}'.format(x_psnr_list[i][j]))
            plt.imshow(x_pca_list[i][j].reshape(32,32).T,cmap='gray')
            plt.axis('off')

def show_faces(x, x_pca_list, k_list, x_psnr_list,i):
    """
    在一行展示降维后的结果
    """
    plt.figure(figsize=(24, 16),  frameon=False)
    size = np.ceil((len(k_list) + 1) / 2)
    plt.subplot(i, size, 1)
    plt.title('Real Image',fontsize='1')
    plt.imshow(x.reshape(32,32).T,cmap='gray')
    plt.axis('off')  #去掉坐标轴
    for i in range(len(k_list)):
        plt.subplot(2, size , i+2)
        plt.title('k = ' + str(k_list[i]) + ', PSNR = ' +
'{:.2f}'.format(x_psnr_list[i]),fontsize='1')
        plt.imshow(x_pca_list[i].reshape(32,32).T ,cmap='gray')
        plt.axis('off')

k_list=[200,100,50,10,1]
face_pca(mat_path,k_list)
plt.show()
\end{lstlisting}
```