

**2020年春季学期  
计算学部《机器学习》课程**

**Lab 4实验报告**

|  |  |
| --- | --- |
| 姓名 | 陈啸 |
| 学号 | 1180300121 |
| 班号 | 1803104 |
| 电子邮件 | [2919730935@qq.com](mailto:2919730935@qq.com) |
| 手机号码 | 13685536606 |

**目录**

[1 实验要求 2](#_Toc32315)

[2 实验环境 2](#_Toc31121)

[3 生成数据 3](#_Toc18509)

[4 主成分分析 3](#_Toc14608)

[4.1 中心化 3](#_Toc12886)

[4.2 Pca 3](#_Toc31881)

[5 人脸数据压缩 4](#_Toc29137)

[5.1 选取一张人脸 4](#_Toc29826)

[5.2 使用matplotlib中的image库 4](#_Toc27199)

[5.3 输出图像 5](#_Toc24391)

[5.4 使用不同的维度进行效果比较 5](#_Toc20065)

[6 信噪比衡量 6](#_Toc25519)

[6.1 信噪比SNR 6](#_Toc21603)

[6.2 代码实现 6](#_Toc10012)

建议写出：问题的描述，解决问题的思路，实验的做法，实验结果的分析，结论，自拟标题

# 实验要求

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）。

# 实验环境

外部库：numpy matploelib

Cuda：10.2

# 生成数据

## 生成一组数据

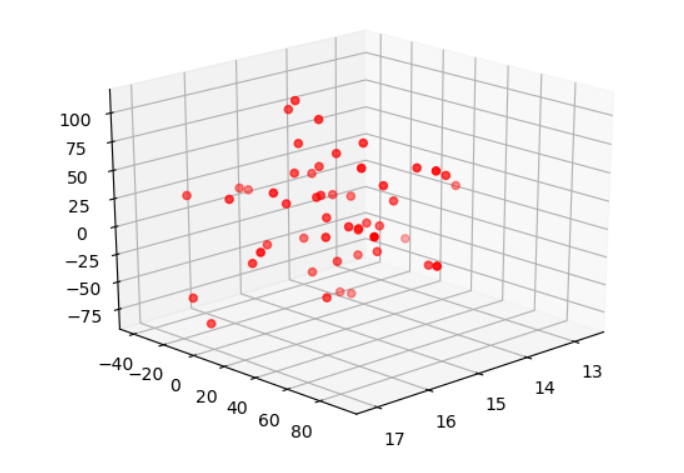
保证其中一个维度的反差远小于别的维度

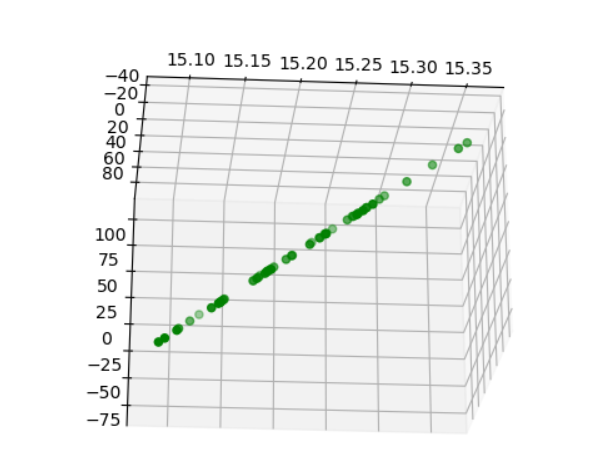
1. def producer():
3. m = 50
4. N = 3
5. scale = np.empty(shape = (N), dtype = **float**)
6. scale[0] = 0.5
7. scale[1] = 20
8. scale[2] = 30
9. temp = np.ndarray(shape = (m, N), dtype = **float**)
11. **for** i in range(N):
12. temp[:, i] = np.random.normal(loc = 10, scale = scale[i], size = m)
14. invert = np.random.random(size = [N])
16. temp = np.dot(temp, np.dot(invert, invert.T))
18. **return** temp, m, N

## 在这组数据上跑pca

1. X, m\_of\_X, N = producer()
2. lowDDataMat , X\_new = pca(X, 2)
3. print(X\_new.shape)
4. fig = plt.figure()
5. ax = fig.gca(projection = '3d')
7. ax.scatter(X[:,0].T, X[:,1].T, X[:,2].T, c = 'r', marker = "o")
8. ax.view\_init(elev = 0, azim = 0)
9. plt.show()
11. fig = plt.figure()
12. ax = fig.gca(projection = '3d')
14. ax.scatter(X\_new[:, 0], X\_new[:, 1], X\_new[:, 2], c = 'g', marker = "o")
15. ax.view\_init(elev = 0, azim = 0)
16. plt.show()

## 实验结果





# 主成分分析

## 中心化

1. # 中心化
2. **def** zeroMean(dataMat):
3. meanVal=np.mean(dataMat,axis=0)     #按列求均值，即求各个特征的均值
4. newData=dataMat-meanVal
5. **return** newData,meanVal

## Pca

1. # Pca
2. **def** pca(dataMat,n):
3. newData,meanVal=zeroMean(dataMat)
4. covMat=np.cov(newData,rowvar=0)    #求协方差矩阵,return ndarray；若rowvar非0，一列代表一个样本，为0，一行代表一个样本
5. eigVals,eigVects=np.linalg.eig(np.mat(covMat))#求特征值和特征向量,特征向量是按列放的，即一列代表一个特征向量
6. eigValIndice=np.argsort(eigVals)            #对特征值从小到大排序
7. n\_eigValIndice=eigValIndice[-1:-(n+1):-1]   #最大的n个特征值的下标
8. n\_eigVect=eigVects[:,n\_eigValIndice]        #最大的n个特征值对应的特征向量
9. lowDDataMat=newData\*n\_eigVect               #低维特征空间的数据
10. reconMat=(lowDDataMat\*n\_eigVect.T)+meanVal  #重构数据
11. **return** lowDDataMat,reconMat

# 人脸数据压缩

## 选取一张人脸

、

（图片来源吴恩达教授百度照片）

## 使用matplotlib中的image库

1. **import** matplotlib.image as mpimg
2. **import** matplotlib.pyplot as plt

这个库中包含了处理图片所需要的各种函数

1. I = mpimg.imread('./wed.jpg')#读取图片
2. image\_shape = I.shape
3. # n , m 表示数据的像素点的长宽
4. m = image\_shape[0]
5. n = image\_shape[1]
6. # k 表示 像素点的颜色 黑白是1 彩色（RGB）一般是3
7. **if** len(image\_shape) == 3:
8. k = image\_shape[2]
9. **else**:
10. k=1
11. **print**(image\_shape)
12. **print**(m, n, k)
13. **print**(np.array(I).shape)
14. **print**(type(I[0,0,0]))
15. x = np.ndarray(shape=(m, n, k) , dtype = float , buffer = I\*1.0)
16. **print**(n , m , k)

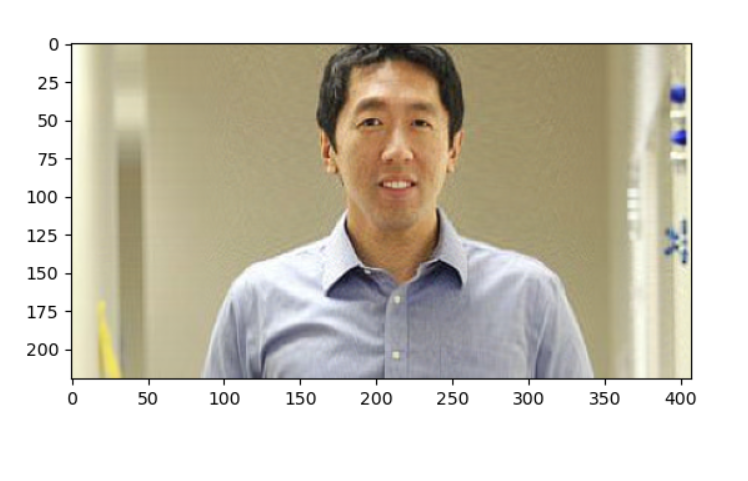
使用imread将一个图片转化为一个像素点的数字矩阵，而后对那个像素矩阵进行pca

## 输出图像

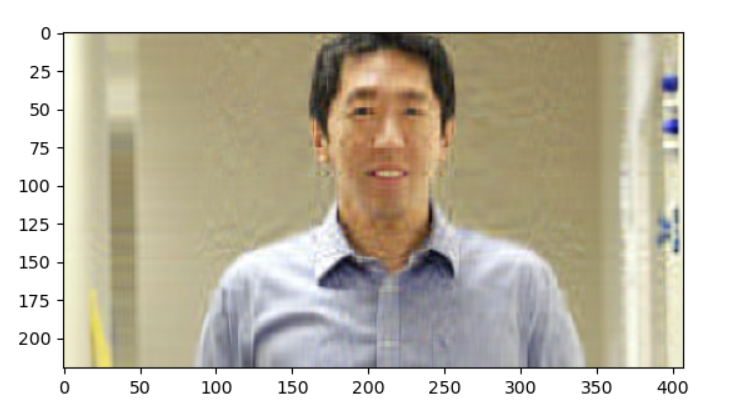
1. # 压缩后的维度
2. d = 50
3. rM = np.ndarray(shape=(m , n , k) , dtype = **float**)
4. **for** i in range(k):
5. lowDDataMat,reconMat = pca (I[ :, :, i] , d)
6. rM[:, :, i] = np.real(reconMat)/255
7. plt.imshow(rM)
8. plt.show()

## 使用不同的维度进行效果比较

D=50



d=20



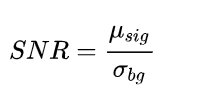
# 信噪比衡量

## 信噪比SNR

信噪比（SIGNAL NOISE RATIO，SNR or S/N），即放大器的输出信号的功率，与同时输出的噪声功率的比值，常常用分贝数表示。设备的信噪比越高表明它产生的杂音越少。一般来说，信噪比越大，说明混在信号里的噪声越小，声音回放的音质量越高，否则相反。

信噪比是科学与工程领域常用的参数，也是常用的评价图像质量的重要指标之一 。图象的信噪比应该等于信号与噪声的功率谱之比，但通常功率谱难以计算，有一种方法可以近似估计图象信噪比，即信号与噪声的方差之比。首先计算图象所有象素的局部方差，将局部方差的最大值认为是信号方差，最小值是噪声方差，求出它们的比值，再转成dB数，最后用经验公式修正。

如果是灰度图像的话，SNR=（洁净图片中的像素点的灰度值之和）/abs（噪声图片的灰度值之和-洁净图片中的灰度值之和）为该图像的信噪比。



## 6.2 代码实现

1. # 计算信噪比
2. sum = 0
3. **for** i in range(m):
4. **for** j in range(n):
5. sum = sum + (np.abs(reconMat[i,j]-x[i,j])\*\*2)
6. sum = sum / n / m
7. print(0)
8. print(20\*np.log10(n/np.sqrt(sum)))

下图为wed。Jpg压缩后的信噪比

