

kmeans算法实验报告

**机器学习实验报告**

课程名称 机器学习

专业班级 计算机少61

姓 名 熊兴宇

学 号 2140506094

目录

[一、 实验要求](#_Toc3520_WPSOffice_Level1) [3](#_Toc3520_WPSOffice_Level1)

[1. 实现PCA方法](#_Toc22177_WPSOffice_Level2) [3](#_Toc22177_WPSOffice_Level2)

[2. 在人脸数据集上尝试PCA降维效果](#_Toc24320_WPSOffice_Level2) [3](#_Toc24320_WPSOffice_Level2)

[二、 实验原理](#_Toc22177_WPSOffice_Level1) [3](#_Toc22177_WPSOffice_Level1)

[1. 实验步骤](#_Toc31521_WPSOffice_Level2) [3](#_Toc31521_WPSOffice_Level2)

[2. 实验内容](#_Toc27185_WPSOffice_Level2) [3](#_Toc27185_WPSOffice_Level2)

[三、 实验过程](#_Toc24320_WPSOffice_Level1) [3](#_Toc24320_WPSOffice_Level1)

[1. 加载面部数据库](#_Toc26906_WPSOffice_Level2) [3](#_Toc26906_WPSOffice_Level2)

[2. 生成PCA实例，展示返回结果](#_Toc1224_WPSOffice_Level2) [4](#_Toc1224_WPSOffice_Level2)

[3. 读取数据](#_Toc12912_WPSOffice_Level2) [4](#_Toc12912_WPSOffice_Level2)

[4. 中心化数据 4](#_Toc1224_WPSOffice_Level2)

[5. 对数据PCA降维](#_Toc22177_WPSOffice_Level3) [4](#_Toc22177_WPSOffice_Level3)

[6. 返回降维数据结果](#_Toc24320_WPSOffice_Level3) [4](#_Toc24320_WPSOffice_Level3)

[四、 实验分析](#_Toc31521_WPSOffice_Level1) [7](#_Toc31521_WPSOffice_Level1)

[五、 实验总结](#_Toc27185_WPSOffice_Level1) [11](#_Toc27185_WPSOffice_Level1)

1. 实验要求
2. 实现PCA方法
3. 在人脸数据集上尝试PCA降维效果
4. 实验原理
5. 实验步骤

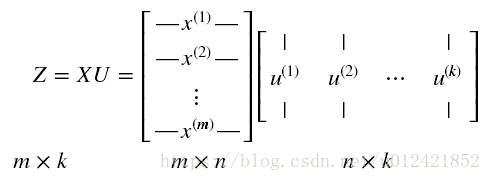
Step1：计算训练数据集X的协方差矩阵C

Step2：计算C的特征值和特征向量

Step3：特征值按照数值降序排列，特征向量按照自己特征值的顺序也依次排列

Step4：如果要得到训练数据集X的k阶降维矩阵，则取前K个特征向量u1,u2,...,uk，组成降维转换矩阵U={u1, u2, ..., uk}

Step5：然后经过X的降维转换Z = XU，X 是 m × n 的矩阵， U 是 n × k 的矩阵，Z 是 m × k 的矩阵，就得到X降维后的降维矩阵Z：



作者：CV\_ML\_DP

来源：CSDN

原文：https://blog.csdn.net/u012421852/article/details/80458340

1. 实验内容

我主要实现对于单张图片的降维，即图片压缩，可以保存降维后的矩阵Z和特征向量矩阵U，在需要时将这两个矩阵相乘ZUt映射回原空间。

1. 实验过程
2. 加载面部数据库

tests.py

1. **def** loadFace():
2. pgm\_files=[]
3. path=work\_path+r'/yaleB01'
4. files=os.listdir(path)
5. **try**:
6. **for** f **in** files:
7. fl=os.path.join(path,f)
8. **if** os.path.isfile(fl) **and** os.path.splitext(fl)[1]=='.pgm':
9. pgm\_files.append(fl)
10. **except** Exception:
11. **print**('file search fail')
12. pgm\_ims=[np.array(Image.open(file)) **for** file **in** pgm\_files]
13. **return** pgm\_ims

读取存在工作路径下的某个数据库所有后缀名为pgm的文件，以数据矩阵的形式保存在pgm\_ims中

1. 生成PCA实例，展示返回结果

tests.py

1. **def** test\_pca(self):
2. pic\_num=END\_INDEX-START\_INDEX
3. images=loadFace()
5. **for** i **in** range(len(images)):
6. images[i]=images[i][:,:]
8. pca=PCA(images[START\_INDEX:END\_INDEX])
9. pca\_ims=pca.ret()
10. **for** i **in** range(START\_INDEX,END\_INDEX):
11. before\_pca=Image.fromarray(images[i])
12. after\_pca=Image.fromarray(pca\_ims[i-START\_INDEX])
13. fig=plt.figure('pca')
14. ax=fig.add\_subplot(pic\_num,2,i\*2+1-START\_INDEX\*2)
15. ax.imshow(before\_pca,cmap='gray',vmin=0,vmax=255)
16. ax=fig.add\_subplot(pic\_num,2,i\*2+2-START\_INDEX\*2)
17. ax.imshow(after\_pca,cmap='gray',vmin=0,vmax=255)
18. plt.show()

给定任意张图片（超过十张绘图会报错），读取图片并展示降维效果，维数在PCA.py文件中给定

1. 读取数据

PCA.py

1. REMAIN\_DIM=6
2. REMAIN\_RATE=0.999999
3. ENABLE\_RATE=False
5. **class** PCA():
6. **def** \_\_init\_\_(self,args):
7. self.raw\_data=np.array(args)
8. self.len=len(self.raw\_data)
9. self.data=self.centerize(self.raw\_data)

以np的形式保存，再中心化

1. 中心化数据

PCA.py

1. **def** centerize(self,args):
2. center\_data=[0 **for** i **in** range(self.len)]
3. **for** i **in** range(self.len):
4. center\_data[i]=np.array([arr-(arr.sum()/len(arr)) **for** arr **in** args[i]])
5. **return** center\_data

对矩阵的每个列向量取平均值后每一列减去该列平均值

1. 对数据PCA降维

PCA.py

1. **def** pca(self,X):
2. m=X.shape[0]
3. cov=np.matmul(X,X.transpose())/(m-1)
4. #对数位作出一些限制，否则会出现小数相减最后不归0的结果
5. **for** i **in** range(cov.shape[0]):
6. **for** j **in** range(cov.shape[1]):
7. cov[i][j]=round(cov[i][j],8)
8. lamda\_v,lamda\_a=np.linalg.eig(cov)
9. lamda={lamda\_v[i]:lamda\_a[:,i] **for** i **in** range(len(lamda\_a))}
10. lamda=sorted(lamda.items(),key=**lambda** x:x[0],reverse=True)
11. pca\_mat2=np.array([tuple[1] **for** tuple **in** lamda])
13. sum=np.sum(lamda\_v)\*REMAIN\_RATE
14. remain\_sum=0
15. pca\_mat=[]
16. **if** ENABLE\_RATE:
17. **for** i **in** range(len(lamda)):
18. **if**(remain\_sum<sum):
19. remain\_sum+=lamda[i][0]
20. pca\_mat.append(lamda[i][1])
21. **else**:
22. **for** i **in** range(len(lamda)):
23. **if** i<REMAIN\_DIM:
24. pca\_mat.append(lamda[i][1])
25. pca\_mat=np.array(pca\_mat)
26. **return** pca\_mat

使用np.linalg.eig保存特征值和特征向量，注意特征向量要按列存，然后按特征值大小排序根据ENABLE\_RATE布尔值情况，要么给定REMAIN\_DIM维度保留一定维数，要么给定特征值之和的保留比REMAIN\_RATE。将特征向量矩阵保存在pca\_mat中

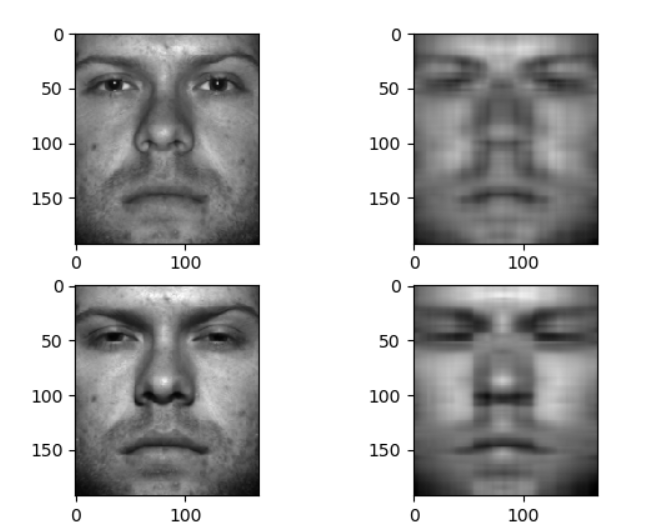
1. 返回降维数据结果

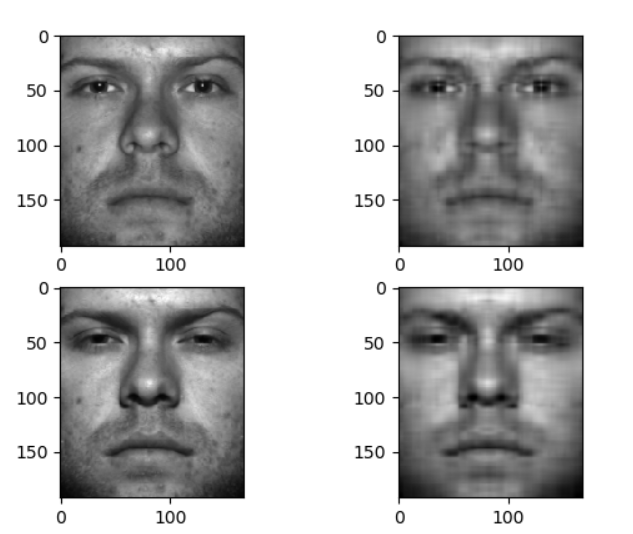
PCA.py

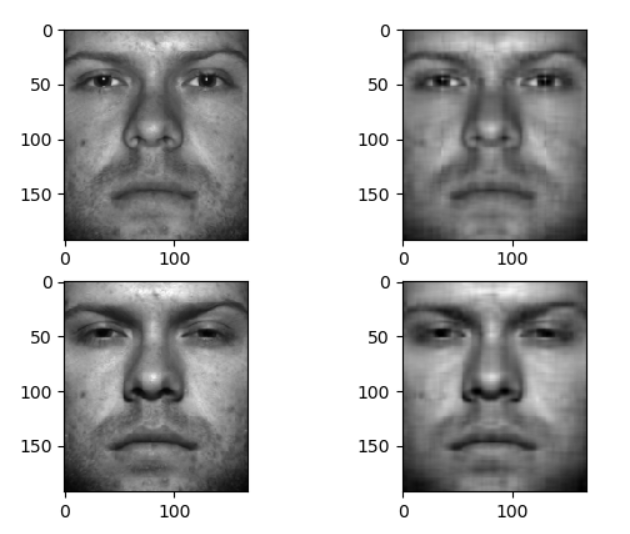
1. **def** ret(self):
2. self.return\_data=[0 **for** i **in** range(self.len)]
3. **for** i **in** range(len(self.data)):
4. tran\_mat=self.pca(self.data[i])
5. tran\_mat=np.matmul(tran\_mat.transpose(),tran\_mat)
6. rev\_pca\_data=np.matmul(tran\_mat,self.data[i])
7. self.return\_data[i]=rev\_pca\_data+self.raw\_data[i]-self.data[i]
8. **return** self.return\_data

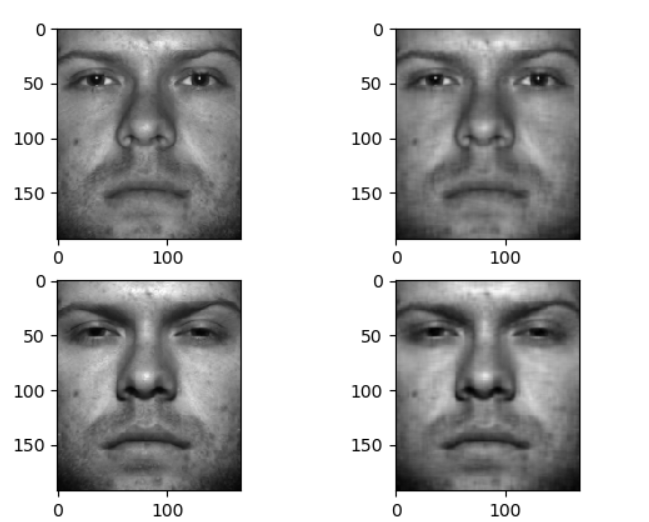
返回映射回原空间的降维结果，并逆中心化

1. 实验分析

3维

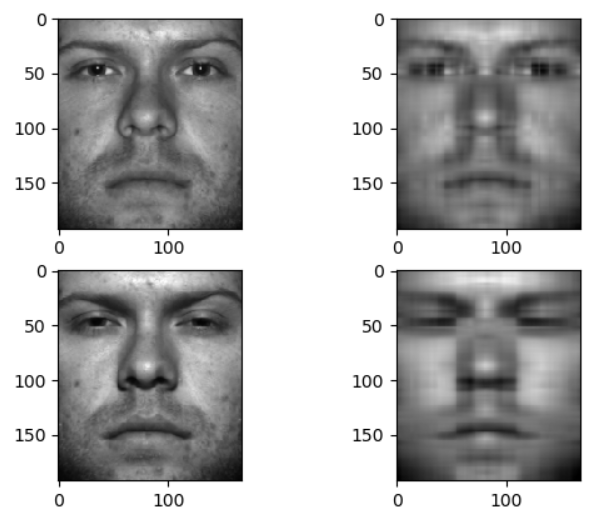
6维

10维

20维

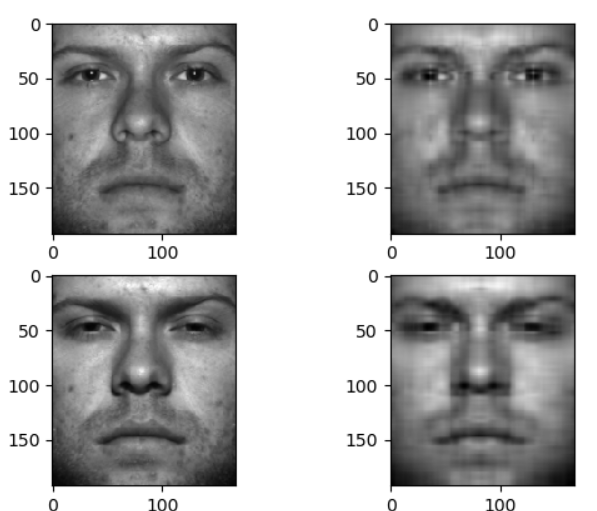
可以看出，20维基本上能保证192\*168的图片不太失真。

特征值之和保留0.8

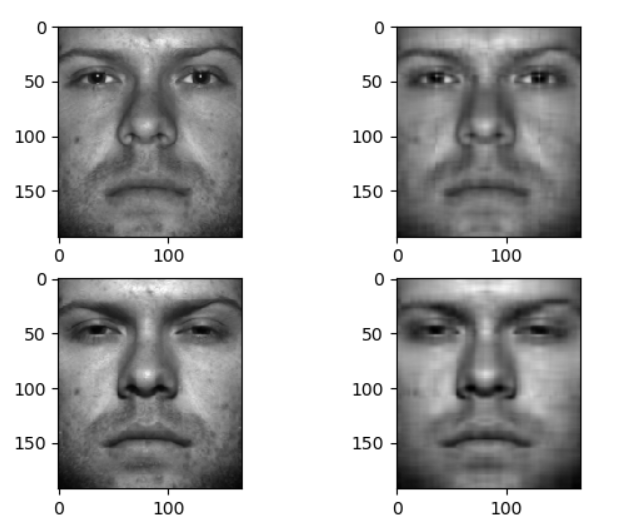


勉强还能看出是个人脸吧

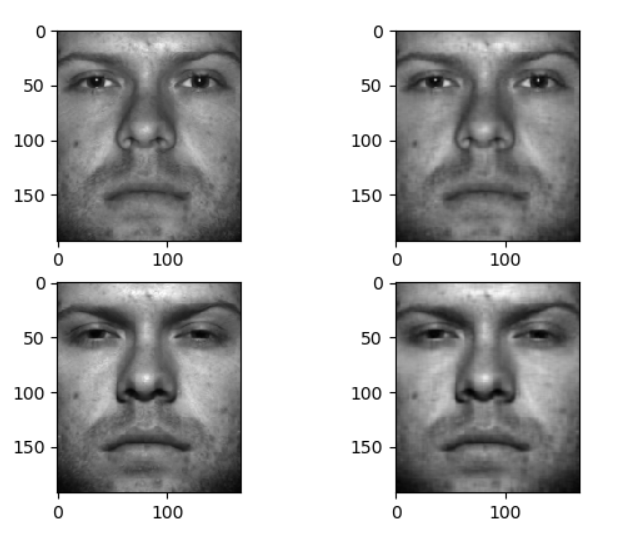
0.9



0.95



0.99



1. 实验总结

本实验有一个在计算协方差矩阵特征向量和特征值时有一个特别容易出错的地方，当然我也使用numpy.linalg.eig，返回的特征向量是一个np矩阵，其中的每一个array都是一个行向量而非我们通常默认的列向量，我在这里找了很久也没找出来这个bug最后和同学讨论交流后发现了。

还有一点就是数值分析课还是很重要的。协方差矩阵是实对称的，它的特征值应该也是实数。但我在运算过程却算出了复数，打印协方差矩阵一看，也没有发现不对称的数值。我估计是由于小数位太多导致两个本应相等的数相减后不精确了，没有得到0，而是下溢了。于是我对协方差矩阵取保留8位小数避免下溢。

运用PCA可以对图片集和单张图片进行降维。我这里只做了单图的降维，相当于压缩。多图的PCA将单个图片的数据矩阵扁平化后作为一个列向量，多张图的列向量合并为矩阵作为PCA的输入，单图的将数据矩阵直接作为输入。