实验七 主成分分析 PCA

一、实验目的

1. 掌握主成分分析的基本原理
2. 编程实现简单的主成分分析
3. 掌握主成分分析里的关键语句
4. 能够用主成分分析解决实际问题

二、实验步骤

1. 生成或调用数据集
2. 生成或调用 PCA 函数
3. 绘图
4. 得出结果并分析不同参数对结果的影响

三、实验内容

1. 运行、调试源代码
2. 分析 PCA 的基本步骤
3. 分析运行结果中输出参数的含义
4. 编程解决实际问题

四、实验原理

具体分析见课本。以下为实现步骤：

Step 1: 求平均值以及做 normalization

Step 2: 求协方差矩阵(Covariance Matrix)

Step 3: 求协方差矩阵的特征根和特征向量

Step 4: 选择主要成分

Step 5: 转化得到降维的数据

任务 1：编程实现 PCA

（1）结合附件程序,写出 PCA 实现原理；

（2）给每段程序加注释；

（3）分析最后的结果图中红、蓝色实线，和红、蓝、绿离散点所代表的意义；

（4）以图的内容为每幅图命名；

（5）分析程序中蓝色语句的作用，如果去掉对结果有什么影响，说明原因。

任务 2：编程实现和调用 PCA 库比较

打开原文链接，完成 1-6 部分。

https://blog.csdn.net/m0_52118763/article/details/121514464

（1）调试运行程序，结合 print 和 show 指令显示文中结果；

（2）说明为什么编程实现和用 sklearn 实现 PCA 结果不同；要想使显示结果一样，应如何修改程序；

（3）保存结果，用图像内容命名；

（4）分析调用 sklearn 实现 PCA 的过程，体会其便捷性；

（5）思考:以 sklearn 实现 PCA 为例，如果想提取 1 维或 3 维，需要做哪些修改。

附件 1：

```
import numpy as np
import matplotlib.pyplot as plt
x=np.array([2.5,0.5,2.2,1.9,3.1,2.3,2,1,1.5,1.1])
y=np.array([2.4,0.7,2.9,2.2,3,2.7,1.6,1.1,1.6,0.9])
```

```
mean_x=np.mean(x)
mean_y=np.mean(y)
scaled_x=x-mean_x
scaled_y=y-mean_y
data=np.matrix([[scaled_x[i],scaled_y[i]] for i in range(len(scaled_x))])

plt.subplot(121)
plt.plot(x,y,'d')
plt.subplot(122)
plt.plot(scaled_x,scaled_y,'o')
plt.show()

cov=np.cov(scaled_x,scaled_y)
print('cov=',cov)
eig_val, eig_vec = np.linalg.eig(cov)
print('eig_val=',eig_val)
print('eig_vec=',eig_vec)

xmin ,xmax = scaled_x.min(), scaled_x.max()
ymin, ymax = scaled_y.min(), scaled_y.max()
dx = (xmax - xmin) * 0.2
dy = (ymax - ymin) * 0.2
plt.xlim(xmin - dx, xmax + dx)
plt.ylim(ymin - dy, ymax + dy)

new_data=np.transpose(np.dot(eig_vec,np.transpose(data)))

plt.plot([eig_vec[:,0][0],0],[eig_vec[:,0][1],0],color='red')
plt.plot([eig_vec[:,1][0],0],[eig_vec[:,1][1],0],color='red')
plt.show()

eig_pairs = [(np.abs(eig_val[i]), eig_vec[:,i]) for i in range(len(eig_val))]
eig_pairs.sort(reverse=True)
feature=eig_pairs[0][1]
new_data_reduced=np.transpose(np.dot(feature,np.transpose(data)))
print('new_data_reduced = ')
print(new_data_reduced)
plt.plot(scaled_x,scaled_y,'o',color='red')
plt.plot([eig_vec[:,0][0],0],[eig_vec[:,0][1],0],color='red')
plt.plot([eig_vec[:,1][0],0],[eig_vec[:,1][1],0],color='blue')
plt.plot(new_data[:,0],new_data[:,1],'^',color='blue')
plt.plot(new_data_reduced[:,0],[1.2]*10,'*',color='green')
plt.show()
```