# Lab\_1 基于PCA/LDA降维方法的特征可视化

## 一、实习目的与要求

1、结合实际应用理解数据降维；

2、理论结合实践，采用PCA或者LDA对所提供的3840\*415维数据进行降维，并进行可视化；

3、通过进一步查阅文献，了解相关研究方向的最新研究进展。

## 二、实习题目

基于PCA/LDA降维方法的特征可视化

#### 【实验数据】

BU3D-feature.mat， 3840\*415维， 每一行表示一个特征向量，每一个特征向量的最后一列（第415列）为特征向量的类别，共3840个特征向量。

#### 【过程描述】

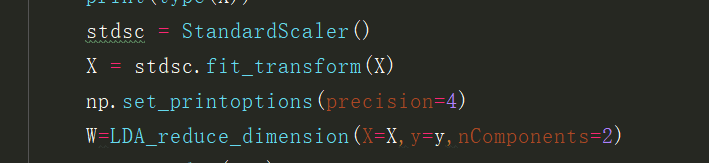
主要是利用PCA和LDA降维方法对数据进行降维，然后利用python的matplotlib库进行数据可视化。详见文档中的其他文件

#### 【源代码】

见附录

1. **数据预处理**

**在LDA中对Sw求逆无法求出，所以需要对数据进行下面的处理**

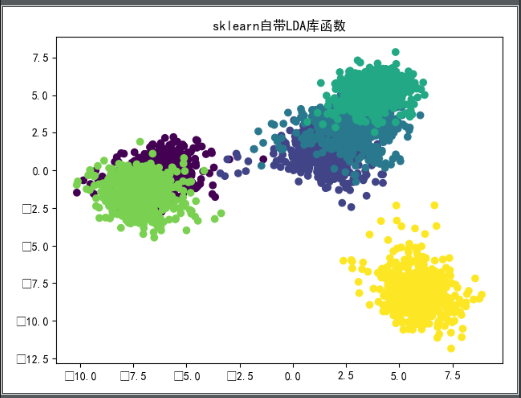
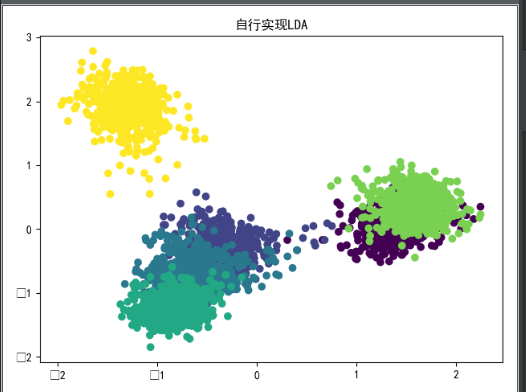


**（2）降维方法（要求至少选了一种降维方法：PCA、LDA、SVD等）**

**本题使用了两种降维方法LDA和PCA，详见文档中其他两个文件**

1. **模型训练**

**（3）可视化评价**



## 三、本次实习小结

（1）对python的matplotlib库的画图函数有所遗忘，这一部分之前学过的但是很久没做了忘记的差不多了，要好好复习一下了。

（2）对机器学习相关算法了解不够深入，以前只听过概念，没有实际操作过，最近主要想打算把清华大学数据挖掘课程看完，然后在找一些题和项目练手。

（3）要加强对python其他库的学习

# 附件

（1）PCA降维

#encoding:utf-8

import scipy.io

import numpy as np

from PCA import pca

import matplotlib

import matplotlib.pyplot as plt

def zeroMean(dataMat):

meanVal=np.mean(dataMat,axis=0)#按列求均值，即求各个特征的均值

newData=dataMat-meanVal

return newData,meanVal

def pca(dataMat,n):

newData,meanVal=zeroMean(dataMat=dataMat)

covMat=np.cov(newData,rowvar=0)#rowvar=0,很重要，说明传入的数据一行代表一个样本

eigVals,eigVects=np.linalg.eig(np.mat(covMat))

#print("eigVals=\n",eigVals)

eigValIndice=np.argsort(eigVals)#对特征值的下标从小到大排序

# sum = 0

# count = 0

#

# for indice in eigValIndice[::-1]:

# sum += eigVals[indice]

# count += 1

# if sum / np.sum(eigVals) >= 0.85:

# print("降低为"+str(count)+"维")

# n = count

# break

#print("eigValIndice=\n",eigValIndice)

n\_eigValIndice=eigValIndice[-1:-(n+1):-1]#[-1，-2),其实也就是第一个

#print("n\_eigValIndice=\n",n\_eigValIndice)

n\_eigVect=eigVects[:,n\_eigValIndice]#二维的

#print("n\_eigVect=\n",n\_eigVect)

lowDataMat=newData\*n\_eigVect

reconMat=(lowDataMat\*n\_eigVect.T)+meanVal

return lowDataMat,reconMat

if '\_\_main\_\_'==\_\_name\_\_:

data=scipy.io.loadmat("BU3D\_feature.mat")

dataMat=data.get("data")

y=np.array(dataMat)[:,-1]

dataMat = np.delete(dataMat, -1, axis=1)

dataMat,reconMat=pca(dataMat=dataMat,n=2)

#plotData(dataMat=dataMat,reconMat=reconMat)

plt.scatter(dataMat[:, 0].tolist(), dataMat[:, 1].tolist(),marker='o',c=y)

plt.title('PCA')

plt.show()

1. LDA降维方法

#encoding:utf-8

import numpy as np

import scipy.io

import matplotlib

import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis

def Iris\_label(s):

it={b'Iris-setosa':0, b'Iris-versicolor':1, b'Iris-virginica':2 }

return it[s]

def LDA\_reduce\_dimension(X,y,nComponents):

'''输入：X为数据集（m\*n）,y为label(m\*1),nComponents为目标维数'''

'''输出：W矩阵n\*nComponents'''

labels=list(set(y))#将集合转化成列表

#将样本分类

xClasses={}

for label in labels:

xClasses[label]=np.array( [X[i] for i in range(len(X)) if y[i]==label])

# 整体均值

meanAll = np.mean(X, axis=0) # 按列求均值，结果为1\*n(行向量)

meanClasses = {}

# 求各类均值

for label in labels:

meanClasses[label] = np.mean(xClasses[label], axis=0) # 1\*n

# 全局散度矩阵

St = np.zeros((len(meanAll), len(meanAll)))

St = np.dot((X - meanAll).T, X - meanAll)

# 求类内散度矩阵

# Sw=sum(np.dot((Xi-ui).T, Xi-ui)) i=1...m

Sw = np.zeros((len(meanAll), len(meanAll))) # n\*n

for i in labels:

Sw += np.dot((xClasses[i] - meanClasses[i]).T,(xClasses[i] - meanClasses[i]))

print(Sw)

# 求类间散度矩阵

Sb = np.zeros((len(meanAll), len(meanAll))) # n\*n

Sb = St - Sw

# 求类间散度矩阵

# Sb=sum(len(Xj) \* np.dot((uj-u).T,uj-u)) j=1...k

# Sb=np.zeros((len(meanAll), len(meanAll) )) # n\*n

# for i in labels:

# Sb+= len(xClasses[i]) \* np.dot( (meanClasses[i]-meanAll).T.reshape(len(meanAll),1),

# (meanClasses[i]-meanAll).reshape(1,len(meanAll))

# )

# 计算Sw-1\*Sb的特征值和特征矩阵

eigenValues, eigenVectors = np.linalg.eig(

np.dot(np.linalg.pinv(Sw), Sb)

)

# 提取前nComponents个特征向量

sortedIndices = np.argsort(eigenValues) # 特征值排序

W = eigenVectors[:, sortedIndices[:-nComponents - 1:-1]] # 提取前nComponents个特征向量

return W

if '\_\_main\_\_'==\_\_name\_\_:

data = scipy.io.loadmat("BU3D\_feature.mat")

dataMat = data.get("data")

y=np.array(dataMat)[:,-1]

print(y.shape)

dataMat = np.delete(dataMat, -1, axis=1)

X=dataMat

stdsc=StandardScaler()

X=stdsc.fit\_transform(X)

W=LDA\_reduce\_dimension(X=X,y=y,nComponents=2)

newX=np.dot(X,W)#m\*n

matplotlib.rcParams['font.sans-serif'] = ['SimHei']

plt.figure(1)

print(y.shape)

plt.scatter(newX[:,0],newX[:,1],marker='o', c=y)

plt.title('自行实现LDA')

X=dataMat

#4.与sklearn自带库函数对比

lda\_Sklearn=LinearDiscriminantAnalysis(n\_components=2)

lda\_Sklearn.fit(X,y)

newX1=lda\_Sklearn.transform(X)

plt.figure(2)

plt.scatter(newX1[:, 0], newX1[:, 1],marker='o', c=y)

plt.title('sklearn自带LDA库函数')

plt.show()

=======================================================

我的联系方式：

电话：13349830890

邮箱：liuyy@cug.edu.cn