**作业3.5**

1. **实验要求**

编程实现线性判别分析，并给出西瓜数据集3.0a上的结果。

1. **实验原理**

给定训练集样例，设法将样例投影到一条直线上，使得同类样例的投影点尽可能接近，异类样例的投影点尽可能远离，这样在对新样例进行分类时，将投影点到同样的直线上，就能根据投影点的位置来确定样本的类别。

关于LDA的原理及参数求解，可参考书上p61、62。所谓线性判别。类似PCA，LDA可将较高维数据投影到较低维空间上，分析其降维后的数据特征的类别区分情况。

优化目标为最大化下式（Sw-类内散度，Sb-类间散度，w-直线方向向量）：

这里写图片描述

我们的目的是最大化上面的式子，根据书中推导，最优解（直线参数）如下式：

这里写图片描述

即可求出w。

1. **实验过程**

本题有两种思路，分别进行了尝试。

第一种思路是使用tensorflow,利用其中的优化工具对3.35式求解最优解，得到w，过程如下：

1. 前面准备数据设置标记雷同上题。
2. 分别求出两种矩阵的均值和协方差矩阵。
3. 计算Sw-类内散度，Sb-类间散度。
4. 算出J，利用tensorflow将其设置为优化目标，通过训练得出w。

代码如下所示：

from numpy import \*

import matplotlib.pyplot as plt

import tensorflow as tf

data = mat([[0.697,0.460,1],

[0.774,0.376,1],

[0.634,0.264,1],

[0.608,0.318,1],

[0.556,0.215,1],

[0.403,0.237,1],

[0.481,0.149,1],

[0.437,0.211,1],

[0.666,0.091,0],

[0.243,0.267,0],

[0.245,0.057,0],

[0.343,0.099,0],

[0.639,0.161,0],

[0.657,0.198,0],

[0.360,0.370,0],

[0.593,0.042,0],

[0.719,0.103,0]])

X0 = array(data[:8,0:2])

X1 = array(data[8:,0:2])

u0=mean(X0,axis=0)

u1=mean(X1,axis=0)

cov1=cov(X0-u0,rowvar=0)

cov2=cov(X1-u1,rowvar=0)

s\_w=(cov1+cov2).astype(float32)

s\_b=((u0-u1).reshape(-1,1)\*(u0-u1)).astype(float32)

#print(s\_w)

w=tf.Variable(tf.zeros([2,1]))

wT=tf.transpose(w)

s1=tf.matmul(wT,s\_b)

s2=tf.matmul(s1,w)

s3=tf.matmul(wT,s\_w)

s4=tf.matmul(s3,w)

loss = tf.reduce\_mean(tf.clip\_by\_value(-s2/s4, 1e-8,tf.reduce\_max(-s2/s4)))

train = tf.train.GradientDescentOptimizer(1).minimize(loss)

init = tf.initialize\_all\_variables()

sess = tf.Session()

sess.run(init)

for step in range(10000):

sess.run(train)

print(step, sess.run(w).flatten())

第二种思路是直接根据已有数据利用3.39式求解w,过程如下：

1、前面准备数据设置标记雷同上题。

2、分别求出两种矩阵的均值和协方差矩阵。

3、计算Sw-类内散度。

4、直接计算得出w。

相关代码如下：

from numpy import \*

import matplotlib.pyplot as plt

data = mat([[0.697,0.460,1],

[0.774,0.376,1],

[0.634,0.264,1],

[0.608,0.318,1],

[0.556,0.215,1],

[0.403,0.237,1],

[0.481,0.149,1],

[0.437,0.211,1],

[0.666,0.091,0],

[0.243,0.267,0],

[0.245,0.057,0],

[0.343,0.099,0],

[0.639,0.161,0],

[0.657,0.198,0],

[0.360,0.370,0],

[0.593,0.042,0],

[0.719,0.103,0]])

X0 = array(data[:8,0:2])

X1 = array(data[8:,0:2])

u0=mean(X0,axis=0)

u1=mean(X1,axis=0)

cov1=cov(X0-u0,rowvar=0)

cov2=cov(X1-u1,rowvar=0)

w=mat(cov1+cov2).I\*(u0-u1).reshape(-1,1)

print(w)

x=arange(0,1.0,0.1)

y=-w[0]\*x/w[1]

#print(array(y[0]).flatten())

plt.plot(x,array(y[0]).flatten())

ax = plt.subplot(111)

ax.scatter(X0[:,0],X0[:,1],c='r',label='+')

ax.scatter(X1[:,0],X1[:,1],c='b',label='-')

1. **实验结果**

方法一未能得到相应的正确结果，返回值为nan，即无法找到满足条件的w。

经过分析，可能是w初值自行为0，使得求解陷入到局部最优无法跳出导致。

方法二得出w= [[ 1.01674285], [ 5.52422978]],图如下：

