一、问题描述

1.Fisher辨别分析

要求：在UCI数据集上的Iris和sonar数据上验证算法的有效性；Iris数据3类，4维，150个数据；Sonar数据2类，60维，208个样本；

训练和测试样本有三种方式进行划分：（三选一）

1）将数据随机分训练和测试，多次平均求结果

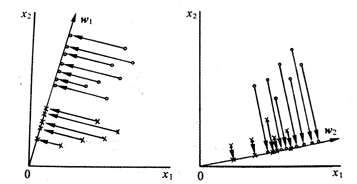
2）k折交叉验证

3）留1法

仿真结果+报告。二、解决方法

1、Fisher线性判别分析（LDA）

两类的线性判别问题可以看作是把所有的样本都投影到一个方向上，然后在这个然后在这个一维空间中确定一个分类的阈值。过这个阈值点且与投影方向垂直的超平面就是两类的分类面。Fisher线性判别的思想就是，选择投影方向，使投影后两类相隔尽可能远，而同时每一类内部的样本又尽可能聚集，如图1所示。



这里只讨论两分类的问题，训练样本集是X ={x1,…,xN},每一个样本是一个d维向量，其中w1类的样本是X1 ={x11,…,xN11}，w2类的样本是X2 ={x12,…,xN12}。我们要寻找一个投影方向W（W也是一个d维向量），投影以后的样本变成

在原样本空间中，类均值向量为

定义各类的类内离散度矩阵(within-class scatter matrix)为

总类内离散度矩阵(pooled within-class scatter matrix)为

类间离散度矩阵(between-class scatter matrix)为

Fisher判别准则变为Rayleigh商

最佳投影方向

决策规则

2、算法描述

1）留一法

留一法是K折法的一种极端情况。

在K折法中，将全部训练集S分成k个不相交的子集，假设S中的训练样例个数为N，那么每一个子集有N/k个训练样例，相应的子集称作{s1,s2,…,sk}。每次从分好的子集中里面，拿出一个作为测试集，其它k-1个作为训练集，根据训练训练出模型或者假设函数。然后把这个模型放到测试集上，得到分类率，计算k次求得的分类率的平均值，作为该模型或者假设函数的真实分类率。

当取K的值为样本个数N时，即将每一个样本作为测试样本，其它N-1个样本作为训练样本。这样得到N个分类器，N个测试结果。用这N个结果的平均值来衡量模型的性能，这就是留一法。在UCI数据集中，由于数据个数较少，采用留一法可以使样本利用率最高。

2）Iris数据集

在Iris数据集中，有150个样本，共三类Setosa、Versicolour、Virginica，一共有4维特征，分别为sepal length、sepal width、petal length、petal width.

在Fisher线性判别方法中，由于只能够判别两类，所以将Iris数据集分三种情况：1-2类，1-3类，2-3类，分别判断每一种情况的Fisher判别准确率。用留一法对训练和测试样本进行划分之后，计算出每一类之间的最优投影方向W1、W2、W3之后，再进行准确率测试。

3）Sonar 数据集

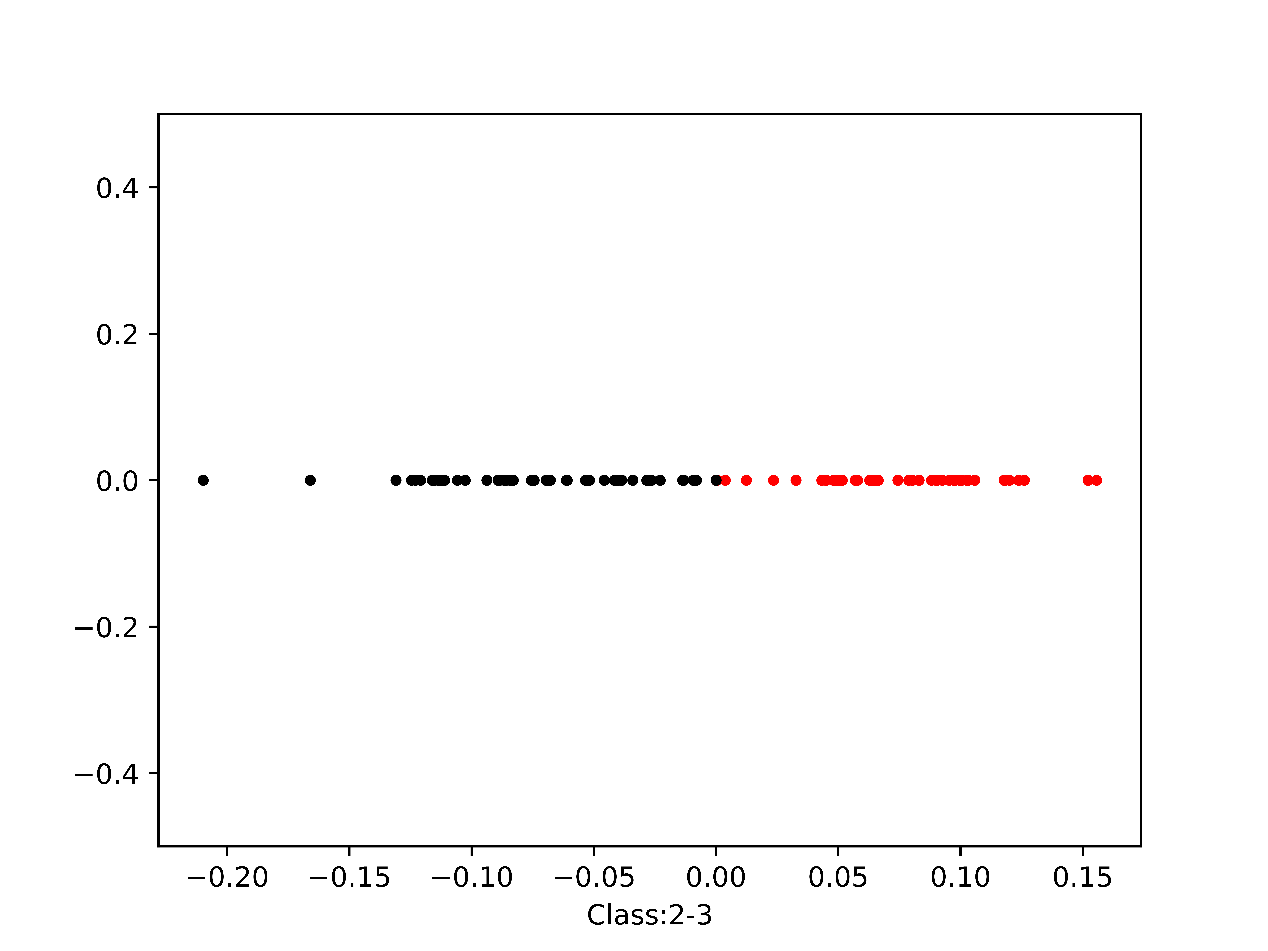
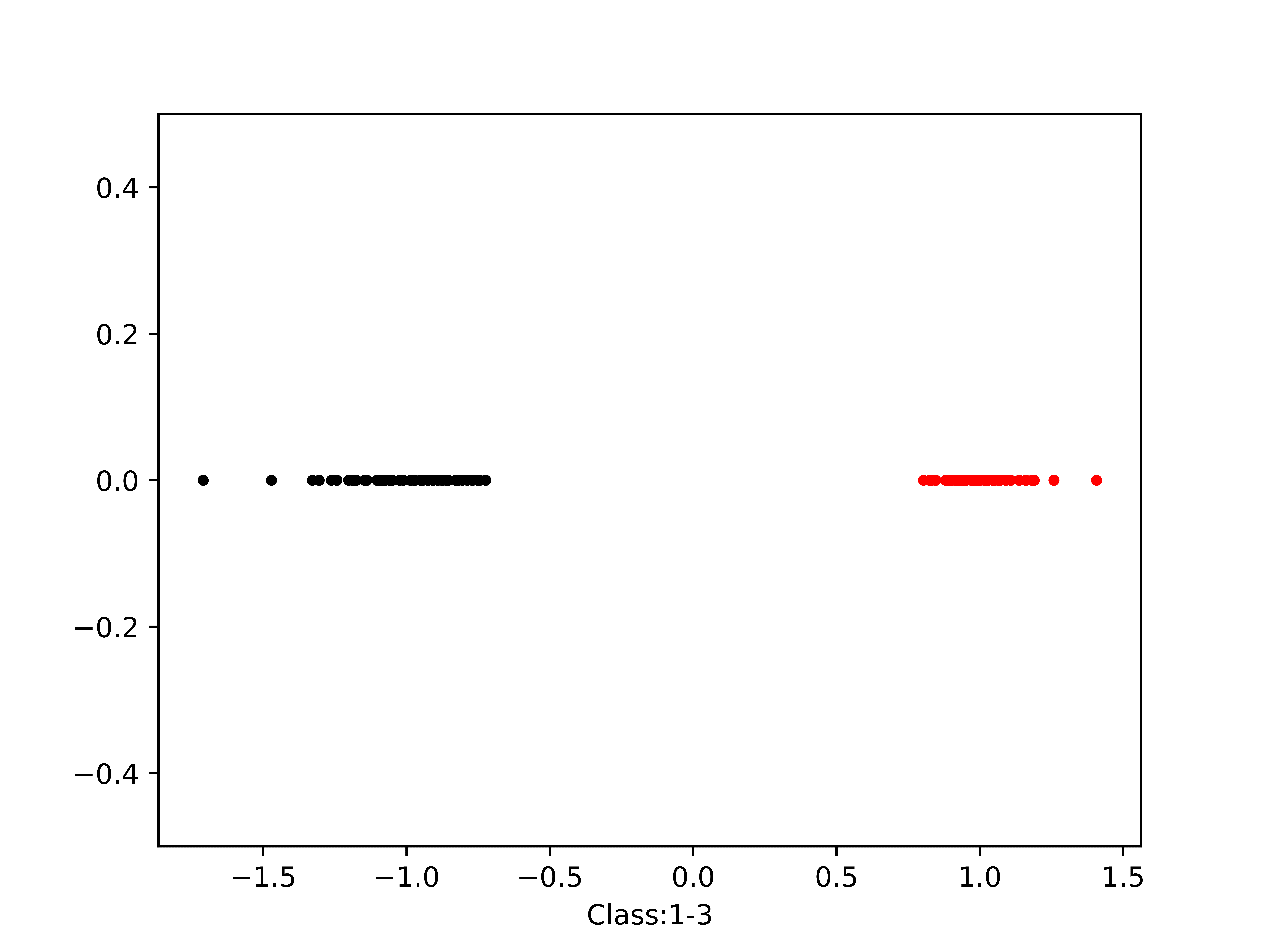
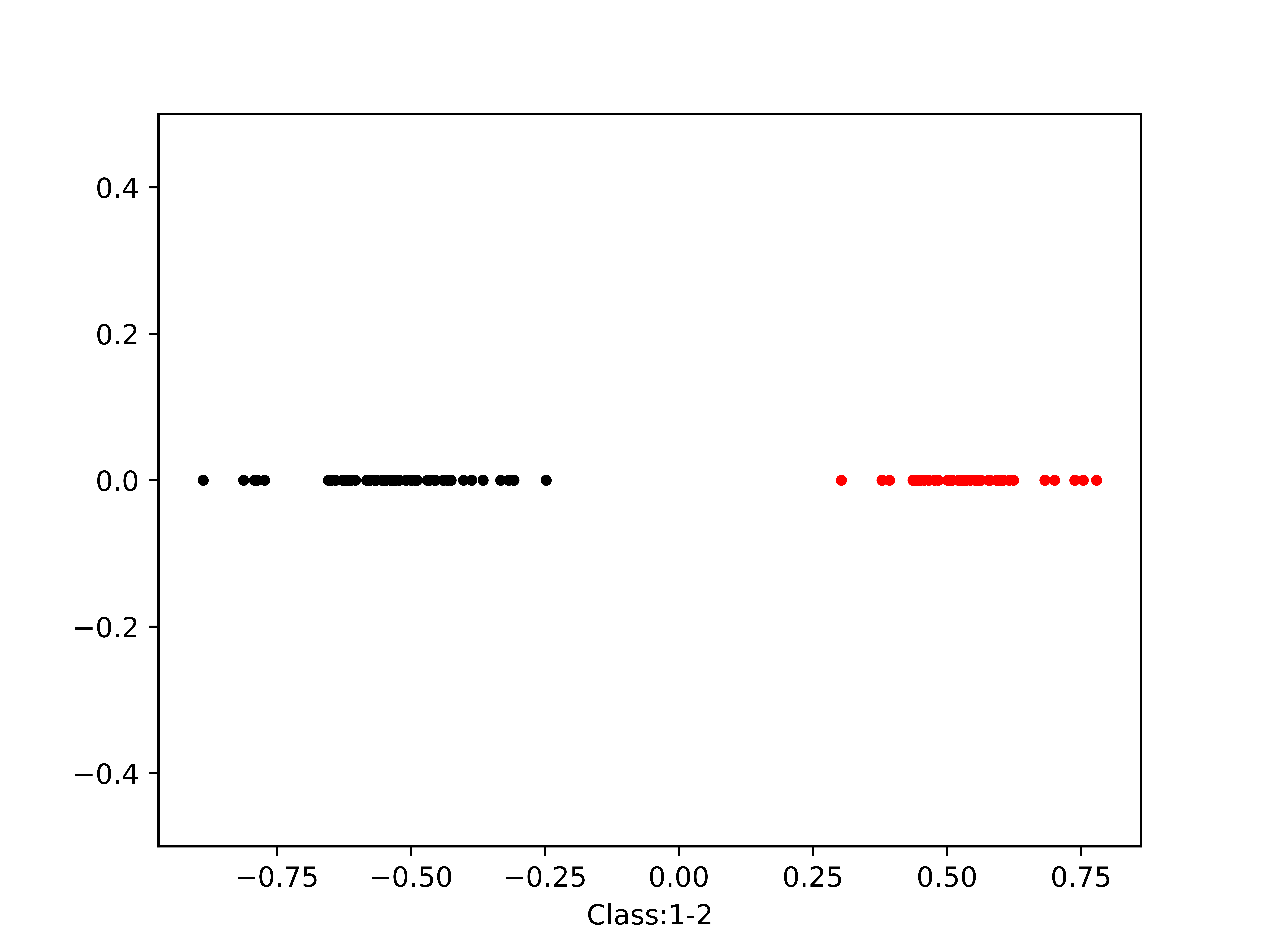
在Sonar数据集中，有208个样本，共两类：字母“R”（岩石）和“M”（矿井），并且有一共有60维的特征。

针对多维的情况，我们可以得出数据集在不同维度下的Fisher线性判别准确率。在这个时候出现了一个问题：如何在60维中选取维数？假设我们要计算在42维的准确率，选哪些维数才最接近真实值？对此，我采用多次取样求平均的方法，对样本特征进行打乱，选前 42维的特征进行计算，重复十次打乱的过程，求准确率的平均值，这样可以规避由于某些特征比较易于区分而导致的准确率高于平均水平的情况。

三、仿真结果

1、Iris数据集

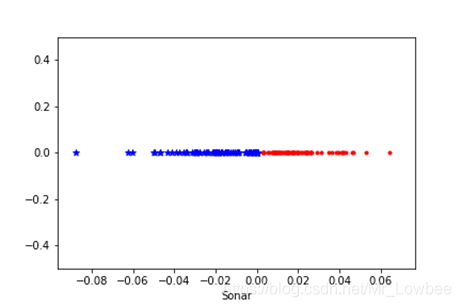
在最佳投影方向W0投影的结果，其中黑色的点为W1类，红色的点为W2类，分类阈值为（0，0）点



从投影点可以看出，第一类和第二类、第一类和第三类相隔特别远，利用线性判别能够完全将它们分开，而第二类和第三类相隔比较近，但利用线性判别也能基本将其分开，得出的准确率也能证明这一点。

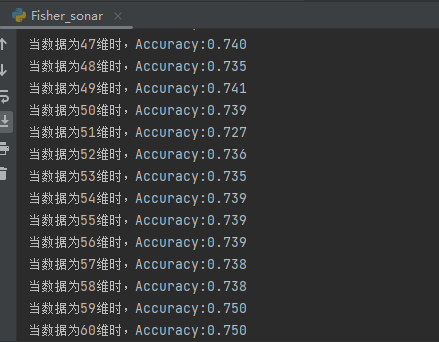
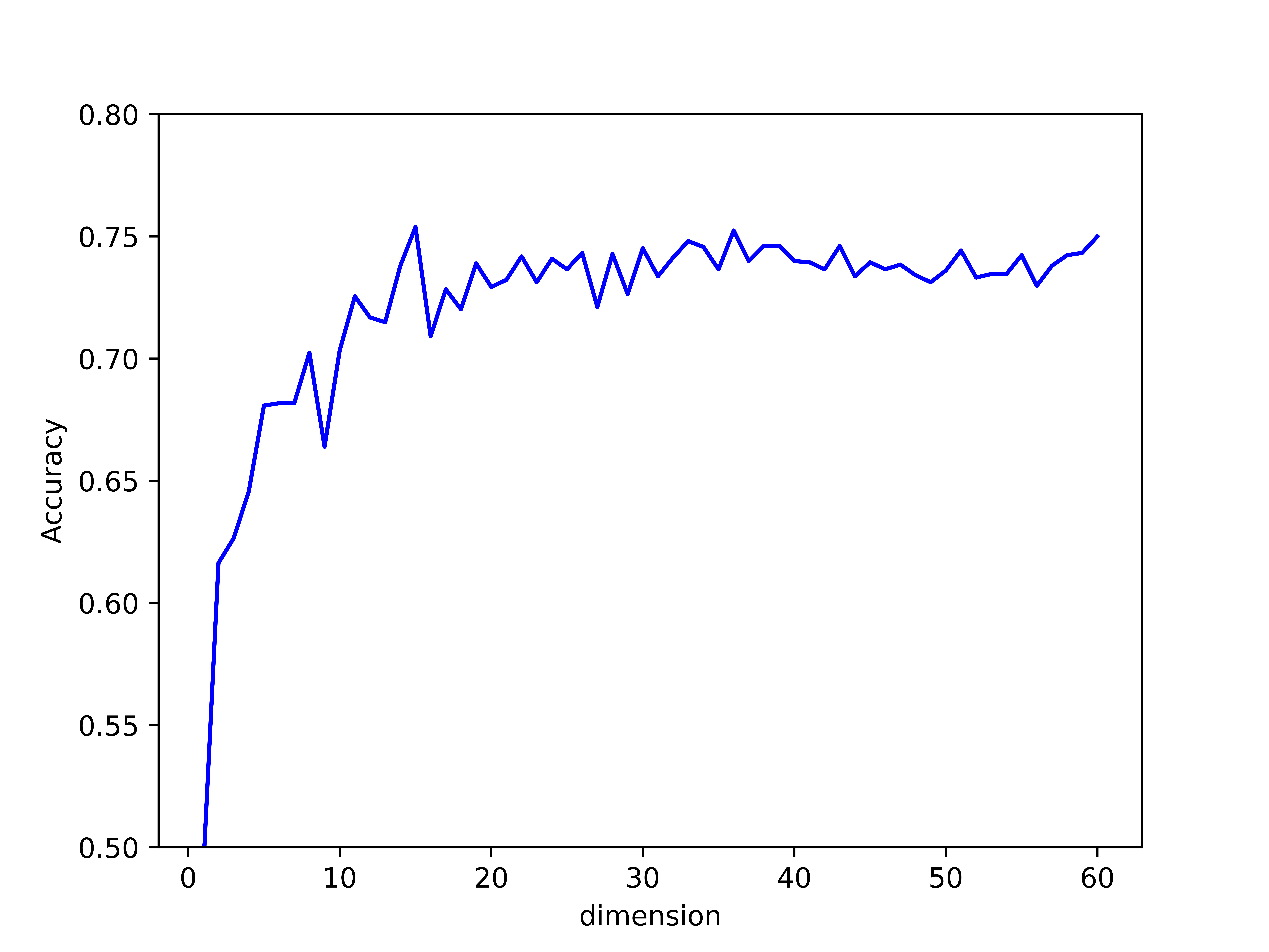
2、Sonar数据集

首先对60维的时候进行投影，得到两类的投影结果



从图中可以看出，两类的分界面处交集较多，准确率约为0.75

将60维扩大到1 ~ 60维，得出每一维的准确率，针对不同维数，画出曲线图。

从图中可以看出，在只有一维的时候，利用Fisher线性判别得出的准确率约为0.5，随着维数的增长，准确率慢慢提高，最终趋于一个稳定值0.75左右。

四、代码

1.Iris数据

# -\*- coding: utf-8 -\*-

import pandas as pd

import numpy as np

def Fisher(X1,X2,n,c):

# 计算三类样本的类均值向量

m1=(np.mean(X1,axis = 0))

m2=(np.mean(X2,axis = 0))

m1 = m1.reshape(n,1) # 将行向量转换为列向量以便于计算

m2 = m2.reshape(n,1)

#计算类内离散度矩阵

S1 = np.zeros((n,n)) # m1 = within\_class\_scatter\_matrix1

S2 = np.zeros((n,n)) # m2 = within\_class\_scatter\_matrix2

if c == 0: # 第一种情况

for i in range(0,49):

S1 += (X1[i].reshape(n,1)-m1).dot((X1[i].reshape(n,1)-m1).T)

for i in range(0,50):

S2 += (X2[i].reshape(n,1)-m2).dot((X2[i].reshape(n,1)-m2).T)

if c == 1:

for i in range(0,50):

S1 += (X1[i].reshape(n,1)-m1).dot((X1[i].reshape(n,1)-m1).T)

for i in range(0,49):

S2 += (X2[i].reshape(n,1)-m2).dot((X2[i].reshape(n,1)-m2).T)

#计算总类内离散度矩阵S\_w

S\_w = S1 + S2

#计算最优投影方向 W

W = np.linalg.inv(S\_w).dot(m1 - m2)

#在投影后的一维空间求两类的均值

m\_1 = (W.T).dot(m1)

m\_2 = (W.T).dot(m2)

#计算分类阈值 W0(为一个列向量)

W0 = -0.5\*(m\_1 + m\_2)

return W,W0

def Classify(X,W,W0):

y = (W.T).dot(X) + W0

return y

#导入sonar.all-data数据集

iris = pd.read\_csv('iris.data',header=None,sep=',')

iris1 = iris.iloc[0:150,0:4]

iris2 = np.mat(iris1)

Accuracy = 0

accuracy\_ = np.zeros(10)

P1 = iris2[0:50,0:4]

P2 = iris2[50:100,0:4]

P3 = iris2[100:150,0:4]

G121 = np.ones(50)

G122 = np.ones(50)

G131 = np.zeros(50)

G132 = np.zeros(50)

G231 = np.zeros(50)

G232 = np.zeros(50)

# 留一法验证准确性

# 第一类和第二类的线性判别

count = 0

for i in range(100):

if i <= 49:

test = P1[i]

test = test.reshape(4,1)

train = np.delete(P1,i,axis=0) # 训练样本是一个列数为t的矩阵

W,W0 = Fisher(train,P2,4,0)

if (Classify(test,W,W0)) >= 0:

count += 1

G121[i] = Classify(test,W,W0)

else:

test = P2[i-50]

test = test.reshape(4,1)

train = np.delete(P2,i-50,axis=0)

W,W0 = Fisher(P1,train,4,1)

if (Classify(test,W,W0)) < 0:

count += 1

G122[i-50] = Classify(test,W,W0)

Accuracy12 = count/100

print("第一类和二类的分类准确率为:%.3f"%(Accuracy12))

# 第一类和第三类的线性判别

count = 0

for i in range(100):

if i <= 49:

test = P1[i]

test = test.reshape(4,1)

train = np.delete(P1,i,axis=0) # 训练样本是一个列数为t的矩阵

W,W0 = Fisher(train,P3,4,0)

if (Classify(test,W,W0)) >= 0:

count += 1

G131[i] = Classify(test,W,W0)

else:

test = P3[i-50]

test = test.reshape(4,1)

train = np.delete(P3,i-50,axis=0)

W,W0 = Fisher(P1,train,4,1)

if (Classify(test,W,W0)) < 0:

count += 1

G132[i-50] = Classify(test,W,W0)

Accuracy13 = count/100

print("第一类和三类的分类准确率为:%.3f"%(Accuracy13))

# 第二类和第三类的线性判别

count = 0

for i in range(100):

if i <= 49:

test = P2[i]

test = test.reshape(4,1)

train = np.delete(P2,i,axis=0) # 训练样本是一个列数为t的矩阵

W,W0 = Fisher(train,P3,4,0)

if (Classify(test,W,W0)) >= 0:

count += 1

G231[i] = Classify(test,W,W0)

else:

test = P3[i-50]

test = test.reshape(4,1)

train = np.delete(P3,i-50,axis=0)

W,W0 = Fisher(P2,train,4,1)

if (Classify(test,W,W0)) < 0:

count += 1

G232[i-50] = Classify(test,W,W0)

Accuracy23 = count/100

print("第二类和三类的分类准确率为:%.3f"%(Accuracy23))

# 画相关的图

import matplotlib.pyplot as plt

y1 = np.zeros(50)

y2 = np.zeros(50)

plt.figure(1)

plt.ylim((-0.5,0.5)) # y坐标的范围

#画散点图

plt.scatter(G121, y1,c='red', alpha=1, marker='.')

plt.scatter(G122, y2,c='k', alpha=1, marker='.')

plt.xlabel('Class:1-2')

plt.savefig('iris 1-2.png',dpi=2000)

plt.figure(2)

plt.ylim((-0.5,0.5)) # y坐标的范围

#画散点图

plt.scatter(G131, y1,c='red', alpha=1, marker='.')

plt.scatter(G132, y2,c='k', alpha=1, marker='.')

plt.xlabel('Class:1-3')

plt.savefig('iris 1-3.png',dpi=2000)

plt.figure(3)

plt.ylim((-0.5,0.5)) # y坐标的范围

#画散点图

plt.scatter(G231, y1,c='red', alpha=1, marker='.')

plt.scatter(G232, y2,c='k', alpha=1, marker='.')

plt.xlabel('Class:2-3')

plt.savefig('iris 2-3.png',dpi=2000)

2.Sonar数据

# -\*- coding: utf-8 -\*-

import pandas as pd

import numpy as np

def Fisher(X1,X2,n,c):

# 计算三类样本的类均值向量

m1=(np.mean(X1,axis = 0))

m2=(np.mean(X2,axis = 0))

m1 = m1.reshape(n,1) # 将行向量转换为列向量以便于计算

m2 = m2.reshape(n,1)

#计算类内离散度矩阵

S1 = np.zeros((n,n)) # m1 = within\_class\_scatter\_matrix1

S2 = np.zeros((n,n)) # m2 = within\_class\_scatter\_matrix2

if c == 0: # 第一种情况

for i in range(0,96):

S1 += (X1[i].reshape(n,1)-m1).dot((X1[i].reshape(n,1)-m1).T)

for i in range(0,111):

S2 += (X2[i].reshape(n,1)-m2).dot((X2[i].reshape(n,1)-m2).T)

if c == 1:

for i in range(0,97):

S1 += (X1[i].reshape(n,1)-m1).dot((X1[i].reshape(n,1)-m1).T)

for i in range(0,110):

S2 += (X2[i].reshape(n,1)-m2).dot((X2[i].reshape(n,1)-m2).T)

#计算总类内离散度矩阵S\_w

S\_w = S1 + S2

#计算最优投影方向 W

W = np.linalg.inv(S\_w).dot(m1 - m2)

#在投影后的一维空间求两类的均值

m\_1 = (W.T).dot(m1)

m\_2 = (W.T).dot(m2)

#计算分类阈值 W0(为一个列向量)

W0 = -0.5\*(m\_1 + m\_2)

return W,W0

def Classify(X,W,W0):

y = (W.T).dot(X) + W0

return y

#导入sonar.all-data数据集

sonar = pd.read\_csv('sonar.all-data',header=None,sep=',')

sonar1 = sonar.iloc[0:208,0:60]

sonar2 = np.mat(sonar1)

Accuracy = np.zeros(60)

accuracy\_ = np.zeros(10)

for n in range(1,61): # n是当前的维数

for t in range(10): # 每一维都求十次平均值

sonar\_random = (np.random.permutation(sonar2.T)).T # 对原sonar数据进行每列打乱

P1 = sonar\_random[0:97,0:n]

P2 = sonar\_random[97:208,0:n]

count = 0

#留一法验证准确性

for i in range(208):

if i <= 96:

test = P1[i]

test = test.reshape(n,1)

train = np.delete(P1,i,axis=0) # 训练样本是一个列数为t的矩阵

W,W0 = Fisher(train,P2,n,0)

if (Classify(test,W,W0)) >= 0:

count += 1

else:

test = P2[i-97]

test = test.reshape(n,1)

train = np.delete(P2,i-97,axis=0)

W,W0 = Fisher(P1,train,n,1)

if (Classify(test,W,W0)) < 0:

count += 1

accuracy\_[t] = count/208

for k in range(10):

Accuracy[n-1] += accuracy\_[k]

Accuracy[n-1] = Accuracy[n-1]/10

print("当数据为%d维时，Accuracy:%.3f"%(n,Accuracy[n-1]))

# 画相关的图

import matplotlib.pyplot as plt

x = np.arange(1,61,1)

plt.xlabel('dimension')

plt.ylabel('Accuracy')

plt.ylim((0.5,0.8)) # y坐标的范围

#画图

plt.plot(x,Accuracy,'b')