**模式识别 Fisher线性判别**

**一、Fisher线性判别算法介绍**

**1.1 介绍**

Fisher两类的判别问题可以看作是把所有样本都投影到一个方向上，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。问题是如何根据实际情况找到这条最好的、最易于分类的投影线，这就是Fisher线性判别算法要解决问题。

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。以下部分仅讨论两类问题。

**1.2 Fisher准则函数中的基本参量**

**（1）样本**

① 训练样本集是(每个样本是一个d维向量)：

，

② 其中类的样本是：

③ 其中类的样本是：

目标:寻找一个投影w（w也是一个d维列向量）,使得投影后的样本变成：

**（2）在原来的样本空间**

1. **类均值向量**为：
2. **各类的类内离散度矩阵**(within-class scatter matrix)为：
3. **总类内离散度**矩阵(pooled within-class scatter matrix)为：
4. 类间离散度矩阵(between-class scatter matrix)定义为：

**（3）在投影以后的一维空间**

① 两类均值分别为：

② 类内离散度矩阵为：

③ 总类内离散度矩阵为：

1. 类间离散度矩阵为：

**1.3 衡量标准与分类**

两类判别，就是希望寻找的投影方向使投影以后两类尽可能分开，而各类内部又尽可能聚集，这一目标可以表示成如下的准则：

这就是Fisher准则函数(Fisher's Criterion)，可变换为：

可求解得Fisher判别准则下的**最佳投影方向**：

若不考虑先验概率，**阈值**可按以下规则选取：

两类线性判别的一般**决策规则**为：

若考虑先验概率，决策规则可以写成：

**二、实验数据集介绍**

**2.1 Iris数据集介绍**

|  |  |  |  |
| --- | --- | --- | --- |
| Iris 数据集 | | | |
| 类别 | 3 | | |
| 维度 | 4 | | |
| 数据长度 | 150 | 50 | Iris-setosa |
| 50 | Iris-versicolor |
| 50 | Iris-setosa |

**2.2 Sonar数据集介绍**

|  |  |  |  |
| --- | --- | --- | --- |
| Sonar 数据集 | | | |
| 类别 | 2 | | |
| 维度 | 60 | | |
| 数据长度 | 208 | 97 | R |
| 111 | M |

**三、实验设置**

对两类数据集，sonar是两分类问题，直接按决策公式判别；iris是三类问题，需要求出样本之间两两组合的判别函数，按照第二种情况进行三分类（每个模式类和其他模式类之间分别用判别点分开）。

两类数据集处理方法具有相似性，因此实验设置基本一致，具体如下：

**1、读取数据信息。**首先从数据集文件（iris.data、sonar.all-data）中读取数据，将读取到的数据按照一定比例（默认2/5测试）**随机**存放到训练集和测试集中，再将训练集中数据按标签分类（iris分三类、sonar分两类）。

**2、求各类样本的基本参量。**根据1.2（2）中的①②③三个公式，分别求解类样本的均值向量、类内离散度矩阵、两两样本间总类内离散度矩阵（iris中需要求解，sonar中求解）。

**3、求解权向量和阈值。**根据1.3中相应公式，求解样本间的权向量和阈值（iris三类样本，需要求解两两样本间的参数，一共求解三次；sonar只需要求解一次），求解完毕之后即可按照第二类情况规则进行分类。

**4、绘图验证训练效果。**利用判别函数分别将各类样本降至一维，按权向量方向投影至坐标轴，并分颜色绘制各点位置（iris分别按照三个投影方向绘制三次），观察不同色点的分布情况，观察训练效果。

**5、 用测试集测试并计算准确率，绘图显示分类效果。**

**6、重复1-5步20次，计算准确率，绘图显示。**

**四、实验结果展示与分析**

**4.1 Iris数据集分类结果分析**

1、首先将三类样本数据两两组合，利用权向量和阈值将4维数据降至1维，在坐标轴上进行绘制，结果展示如下：

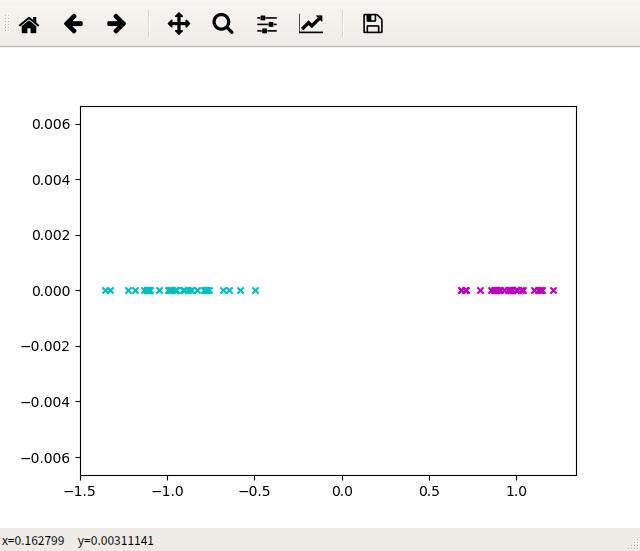


图4.1：Setosa与Versicolor数据训练分类结果

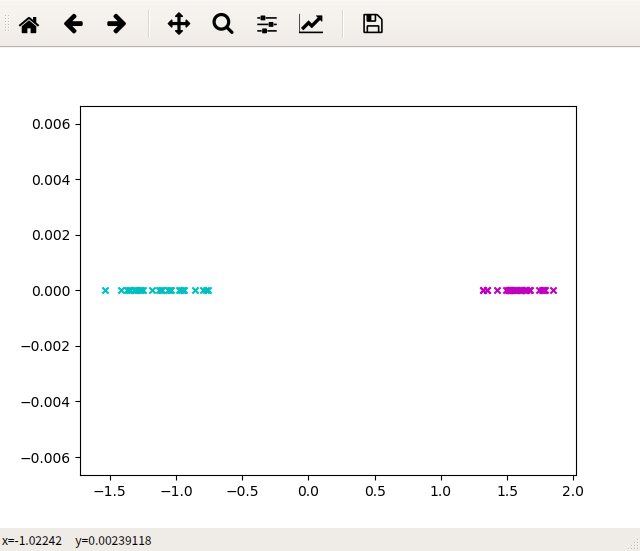


图4.2：Setosa与Virginica数据训练分类结果

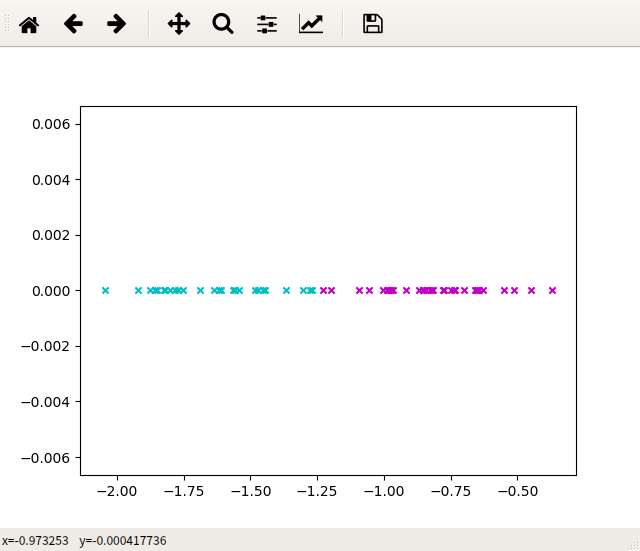


图4.3：Versicolor与Virginica数据训练分类结果

从上述图示可以看出：经训练后，两两样本投影至一维分别聚集到轴的两侧，可以寻找到非常明确的分类点分开两类样本。

2、将训练样本分别按G12、G13、G23（三类样本两两之间的判别函数）进行计算，按照1.3中判别规则进行判别，并将降维后的数据分别投影在Y=3、Y=2、Y=1三条轴上，通过青（3）、紫（2）、蓝（1）三类数据区别三类数据，分类点均为X=0，结果如下图：

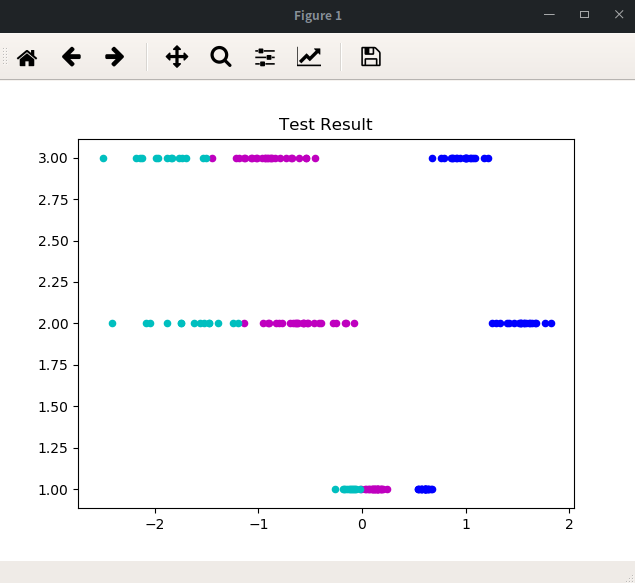


图4.4：分别按三类判别函数将测试数据投影的结果

从上图可以看出按照相对应的判别函数g，两类测试样本均能在分类点X=0两侧很好的区分开来，甚至也能够区分第三类数据。

3、根据第二类分类情况（G12>0、G13>0，则属于第一类；G12<0、G23>0，则属于第二类，G23<0、G13<0，则属于第三类）的分类规则，对测试样本进行分类，对比已知标签，分类正确和错误次数，计算正确率，重新测试二十次，计算平均准确率。结果如下

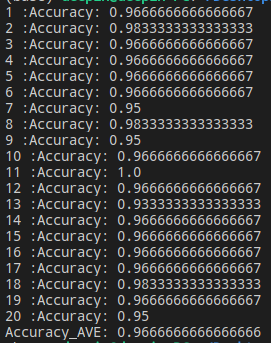
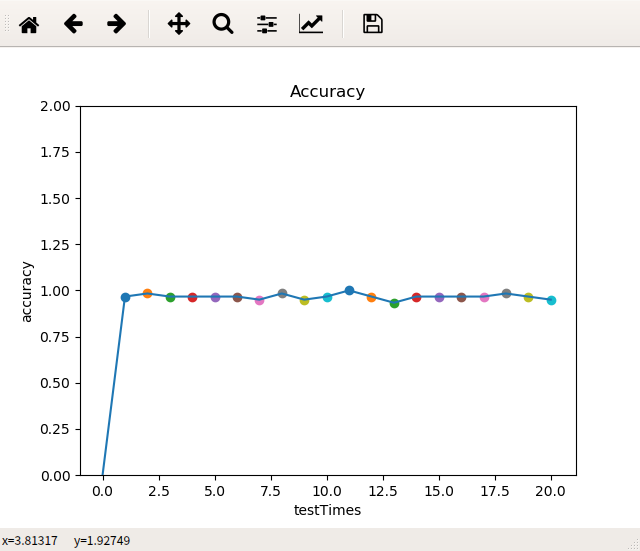


图4.5：测试样本测试准确率

上述结果可以看出测试准确率稳定到96%左右，平均准确率96.7%。

**4.2 sonar数据集分类结果分析**

由于两类数据集处理具有相似性，这里只简要描述

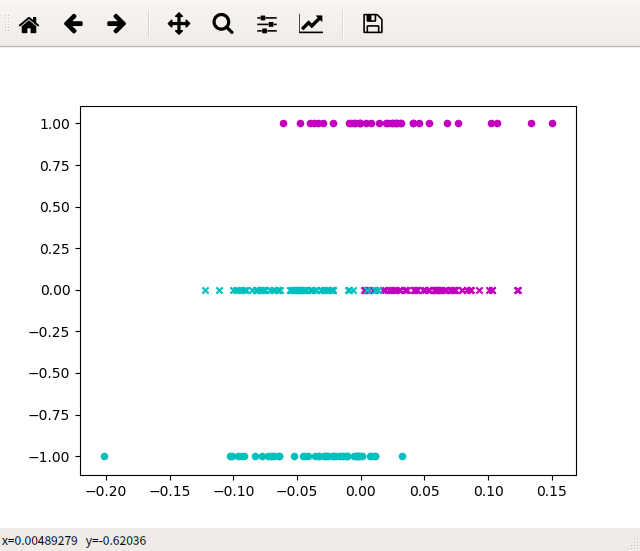
1、求解出判别函数后，把训练集、测试集的投影点分别绘制在一维轴上，两类样本依据颜色区分（训练集：X=0，测试集1类：X=1，测试集二类：X=-1），结果战术如下：

图4.6：训练、测试样本投影到一维的分布

上述结果可以看出，训练集在分类点X=0处有部分重合，测试集的两类均有少量错误判别情况（理论上紫色圆点均在X>0，青色圆点均在X<0）。

2、根据上一步计算结果，对测试样本进行分类，对比已知标签，分类正确和错误次数，计算正确率，重新测试二十次，计算平均准确率。结果如下图所示：

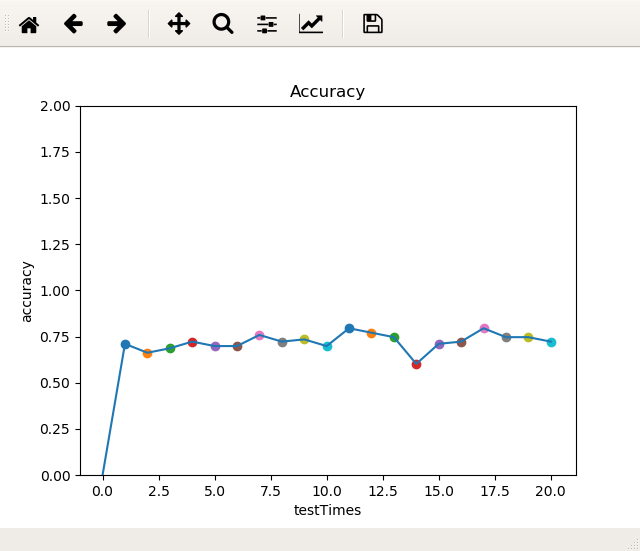
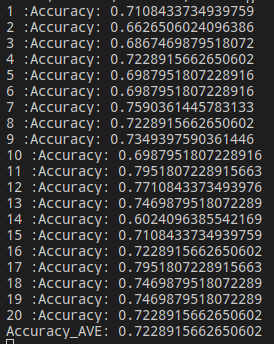


图4.6：测试样本测试准确率

上述结果可以看出测试准确率稳定到70%左右，平均准确率72.3%。

**五、Python代码**

**5.1 iris数据集**

import os

import sys

import numpy as np

from numpy import \*

import random

import matplotlib.pyplot as plt

def readData(DATA\_PATH,DATA\_SIZE,DATA\_DIMENSION,test\_rate):

    test\_Labels = random.sample(range(0,DATA\_SIZE-1),int(DATA\_SIZE\*test\_rate))

    test\_Data = [[0 for i in range(5)] for j in range(len(test\_Labels))]

    train\_Data = [[0 for i in range(5)] for j in range(DATA\_SIZE-len(test\_Labels))]

    #随机提取训练、测试数据

    f = open(DATA\_PATH)

    line = f.readline()

    i = 0

    train\_Label = 0

    test\_Label = 0

    while line:

        temp = line.split(",",4)

        if i in test\_Labels:

            test\_Data[test\_Label] = temp

            test\_Label = test\_Label+1

        else:

            train\_Data[train\_Label] = temp

            train\_Label = train\_Label+1

        i = i+1

        line = f.readline()

    #将要训练的数据分类保存

    \_1\_Num = \_2\_Num = \_3\_Num = 0

    for i in range(len(train\_Data)):

        if train\_Data[i][4] == 'Iris-setosa\n':

            \_1\_Num = \_1\_Num+1

        if train\_Data[i][4] == 'Iris-versicolor\n':

            \_2\_Num = \_2\_Num+1

        if train\_Data[i][4] == 'Iris-virginica\n':

            \_3\_Num = \_3\_Num+1

    \_1\_Data = mat(zeros((\_1\_Num,DATA\_DIMENSION)))

    \_2\_Data = mat(zeros((\_2\_Num,DATA\_DIMENSION)))

    \_3\_Data = mat(zeros((\_3\_Num,DATA\_DIMENSION)))

    \_1\_Num = \_2\_Num = \_3\_Num = 0

    for i in range(len(train\_Data)):

        if train\_Data[i][4] == 'Iris-setosa\n':

            \_1\_Data[\_1\_Num] = train\_Data[i][0:4]

            \_1\_Num = \_1\_Num+1

        if train\_Data[i][4] == 'Iris-versicolor\n':

            \_2\_Data[\_2\_Num] = train\_Data[i][0:4]

            \_2\_Num = \_2\_Num+1

        if train\_Data[i][4] == 'Iris-virginica\n':

            \_3\_Data[\_3\_Num] = train\_Data[i][0:4]

            \_3\_Num = \_3\_Num+1

    f.close()

    return train\_Data,test\_Data,\_1\_Data.T,\_2\_Data.T,\_3\_Data.T

#计算样本均值

def compute\_mean(samples):

    mean\_mat=mean(samples, axis=1)

    return mean\_mat

#计算样本类内离散度

def compute\_withinclass\_scatter(samples, mean):

    #获取样本维数，样本个数

    dimens,nums=samples.shape

    #将所有样本向量减去均值向量

    samples\_mean=samples-mean

    #初始化类内离散度矩阵

    s\_in=0

    for i in range(nums):

        x=samples\_mean[:,i]

        s\_in+=dot(x,x.T)

return s\_in

def showTrainResule(group1,group2,w,w0):

    dimens,nums=group1.shape

    for i in range(nums):

        position = dot(w.T,group1[:,i])+w0

        plt.scatter(float(position),0,20,'m','x')

    dimens,nums=group2.shape

    for i in range(nums):

        position = dot(w.T,group2[:,i])+w0

        plt.scatter(float(position),0,20,'c','x')

    plt.show()

def LDA\_Fisher(dataPath,dataSize,dataDimension,dataTypeNum,testRate = 2/5):

    Accuracy = 0.0

    correctNum = wrongNum = 0

    train\_Data,test\_Data,group1,group2,group3 = readData(dataPath,dataSize,dataDimension,testRate)

    #求均值向量

    mean1 = compute\_mean(group1)

    mean2 = compute\_mean(group2)

    mean3 = compute\_mean(group3)

    #求类内离散度

    s\_in1 = compute\_withinclass\_scatter(group1, mean1)

    s\_in2 = compute\_withinclass\_scatter(group2, mean2)

    s\_in3 = compute\_withinclass\_scatter(group3, mean3)

    #求总类内离散度矩阵

    s\_w\_12 = s\_in1+s\_in2

    s\_w\_13 = s\_in1+s\_in3

    s\_w\_23 = s\_in2+s\_in3

    #求解权向量

    w\_12 = dot(s\_w\_12.I,mean1-mean2)

    w\_13 = dot(s\_w\_13.I,mean1-mean3)

    w\_23 = dot(s\_w\_23.I,mean2-mean3)

    #求解阈值

    w0\_12 = -0.5\*(dot(w\_12.T,mean1)+dot(w\_12.T,mean2))

    w0\_13 = -0.5\*(dot(w\_13.T,mean1)+dot(w\_13.T,mean3))

    w0\_23 = -0.5\*(dot(w\_23.T,mean2)+dot(w\_23.T,mean3))

    #显示训练结果

    showTrainResule(group1,group2,w\_12,w0\_12)

    showTrainResule(group1,group3,w\_12,w0\_13)

    showTrainResule(group2,group3,w\_12,w0\_23)

    #测试结果

    for i in range(len(test\_Data)):

        test1 = mat(zeros((1,4)))

        test1[0] = test\_Data[i][0:4]

        g\_12 = dot(w\_12.T,test1.T)+w0\_12

        g\_13 = dot(w\_13.T,test1.T)+w0\_13

        g\_23 = dot(w\_23.T,test1.T)+w0\_23

        if test\_Data[i][4] == 'Iris-setosa\n':

            color = 'b'

            if g\_12>0 and g\_13>0:

                correctNum += 1

            else:

                wrongNum += 1

        if test\_Data[i][4] == 'Iris-versicolor\n':

            color = 'm'

            if g\_12<0 and g\_23>0:

                correctNum += 1

            else:

                wrongNum += 1

        if test\_Data[i][4] == 'Iris-virginica\n':

            color = 'c'

            if g\_13<0 and g\_23<0:

                correctNum += 1

            else:

                wrongNum += 1

        # plt.scatter(float(g\_12),3,20,color)

        # plt.scatter(float(g\_13),2,20,color)

        # plt.scatter(float(g\_23),1,20,color)

    Accuracy = correctNum/(correctNum+wrongNum)

    # plt.title("Test Result")

    # plt.show()

    return Accuracy

if \_\_name\_\_ == "\_\_main\_\_":

    Accuracy\_AVE = 0

    Accuracy = []

    Accuracy.append(0)

    testTimes = 20

    for i in range(testTimes):

        temp = LDA\_Fisher("iris.data",150,4,3,2/5)

        Accuracy.append(temp)

        print(i+1,":Accuracy:",temp)

        plt.scatter(i+1,temp)

        Accuracy\_AVE += temp

    Accuracy\_AVE /= testTimes

    print("Accuracy\_AVE:",Accuracy\_AVE)

    plt.title("Accuracy")

    plt.xlabel("testTimes")

    plt.ylabel("accuracy")

    plt.ylim(0,2)

    plt.plot(Accuracy)

    plt.show()

**5.2 sonar数据集**

import os

import sys

import numpy as np

from numpy import \*

import random

import matplotlib.pyplot as plt

def readData(DATA\_PATH,DATA\_SIZE,DATA\_DIMENSION,test\_rate):

    test\_Labels = random.sample(range(0,DATA\_SIZE-1),int(DATA\_SIZE\*test\_rate))

    test\_Data = [[0 for i in range(5)] for j in range(len(test\_Labels))]

    train\_Data = [[0 for i in range(5)] for j in range(DATA\_SIZE-len(test\_Labels))]

    #随机提取训练、测试数据

    f = open(DATA\_PATH)

    line = f.readline()

    i = 0

    train\_Label = 0

    test\_Label = 0

    while line:

        temp = line.split(",",DATA\_DIMENSION)

        if i in test\_Labels:

            test\_Data[test\_Label] = temp

            test\_Label = test\_Label+1

        else:

            train\_Data[train\_Label] = temp

            train\_Label = train\_Label+1

        i = i+1

        line = f.readline()

    #将要训练的数据分类保存

    \_1\_Num = \_2\_Num = \_3\_Num = 0

    for i in range(len(train\_Data)):

        if train\_Data[i][DATA\_DIMENSION] == 'R\n':

            \_1\_Num = \_1\_Num+1

        if train\_Data[i][DATA\_DIMENSION] == 'M\n':

            \_2\_Num = \_2\_Num+1

        if train\_Data[i][DATA\_DIMENSION] == 'Iris-virginica\n':

            \_3\_Num = \_3\_Num+1

    \_1\_Data = mat(zeros((\_1\_Num,DATA\_DIMENSION)))

    \_2\_Data = mat(zeros((\_2\_Num,DATA\_DIMENSION)))

    \_3\_Data = mat(zeros((\_3\_Num,DATA\_DIMENSION)))

    \_1\_Num = \_2\_Num = \_3\_Num = 0

    for i in range(len(train\_Data)):

        if train\_Data[i][DATA\_DIMENSION] == 'R\n':

            \_1\_Data[\_1\_Num] = train\_Data[i][0:DATA\_DIMENSION]

            \_1\_Num = \_1\_Num+1

        if train\_Data[i][DATA\_DIMENSION] == 'M\n':

            \_2\_Data[\_2\_Num] = train\_Data[i][0:DATA\_DIMENSION]

            \_2\_Num = \_2\_Num+1

        if train\_Data[i][DATA\_DIMENSION] == 'Iris-virginica\n':

            \_3\_Data[\_3\_Num] = train\_Data[i][0:DATA\_DIMENSION]

            \_3\_Num = \_3\_Num+1

    f.close()

    return train\_Data,test\_Data,\_1\_Data.T,\_2\_Data.T,\_3\_Data.T

#计算样本均值

def compute\_mean(samples):

    mean\_mat=mean(samples, axis=1)

    return mean\_mat

#计算样本类内离散度

def compute\_withinclass\_scatter(samples, mean):

    #获取样本维数，样本个数

    dimens,nums=samples.shape

    #将所有样本向量减去均值向量

    samples\_mean=samples-mean

    s\_in=0

    for i in range(nums):

        x=samples\_mean[:,i]

        s\_in+=dot(x,x.T)

    return s\_in

def showTrainResule(group1,group2,w,w0):

    dimens,nums=group1.shape

    for i in range(nums):

        position = dot(w.T,group1[:,i])+w0

        plt.scatter(float(position),0,20,'m','x')

    dimens,nums=group2.shape

    for i in range(nums):

        position = dot(w.T,group2[:,i])+w0

        plt.scatter(float(position),0,20,'c','x')

    plt.show()

def LDA\_Fisher(dataPath,dataSize,dataDimension,dataTypeNum,testRate = 2/5):

    Accuracy = 0.0

    correctNum = wrongNum = 0

    train\_Data,test\_Data,group1,group2,group3 = readData(dataPath,dataSize,dataDimension,testRate)

    #求均值向量

    mean1 = compute\_mean(group1)

    mean2 = compute\_mean(group2)

    mean3 = compute\_mean(group3)

    #求类内离散度

    s\_in1 = compute\_withinclass\_scatter(group1, mean1)

    s\_in2 = compute\_withinclass\_scatter(group2, mean2)

    s\_in3 = compute\_withinclass\_scatter(group3, mean3)

    #求总类内离散度矩阵

    s\_w = s\_in1+s\_in2

    #求解权向量

    w = dot(s\_w.I,mean1-mean2)

    #求解阈值

    w0 = -0.5\*(dot(w.T,mean1)+dot(w.T,mean2))

    #绘图，训练样本投影

    # showTrainResule(group1,group2,w,w0)

    #绘图，测试样本投影

    for i in range(len(test\_Data)):

        test1 = mat(zeros((1,dataDimension)))

        test1[0] = test\_Data[i][0:dataDimension]

        g = dot(w.T,test1.T)+w0

        if test\_Data[i][dataDimension] == 'R\n':

            if g>0:

                correctNum += 1

            else:

                wrongNum += 1

            # plt.scatter(float(g),1,20,'m')

        if test\_Data[i][dataDimension] == 'M\n':

            if g<0:

                correctNum += 1

            else:

                wrongNum += 1

            # plt.scatter(float(g),-1,20,'c')

    Accuracy = correctNum/(correctNum+wrongNum)

    # plt.show()

    return Accuracy

if \_\_name\_\_ == "\_\_main\_\_":

    Accuracy\_AVE = 0

    Accuracy = []

    Accuracy.append(0)

    testTimes = 20

    for i in range(testTimes):

        temp = LDA\_Fisher("sonar.all-data",208,60,2,2/5)

        Accuracy.append(temp)

        print(i+1,":Accuracy:",temp)

        plt.scatter(i+1,temp)

        Accuracy\_AVE += temp

    Accuracy\_AVE /= testTimes

    print("Accuracy\_AVE:",Accuracy\_AVE)

    plt.title("Accuracy")

    plt.xlabel("testTimes")

    plt.ylabel("accuracy")

    plt.ylim(0,2)

    plt.plot(Accuracy)

    plt.show()