第一次模式识别大作业

王峥 1600510011

一、问题描述

数据：Sonar Iris

方法：Fisher 线性判别分析法

测试方法：10 倍交叉验证法 或者 随机抽样法（10 次平均）

任务：编程实现采用 Fisher 线性判别分析法对 Sonar 数据 和 Iris 数据前两类进行降维 并利用阈值法对其进行分类 选用一种方法实现测试。

二、算法简介

所谓分类器，一般是将输入空间 X，根据需要划分的类别，将输入空间划分为一些互不相交的区域，这些区域的边界一般叫做决策面（decision boundaries）。预测函数的形式不同，会使得决策面或者光滑，或者粗糙。其中有一种比较特别的就是判别面是参数的线性函数的，称为线性决策面，形成的分类器就是线性分类器。

在讲分类器的时候，肯定会遇到判别式函数这个概念。分类器会为每个类别分配一个判别函数，根据判别函数来判断一个新的样本是否是这个类别的。比如，假设有 K 个类别，那么分类器肯定会得到 K 个判别函数 $\delta k(x);k\in[1,2,\cdots,K]$。如果有一个新的样本 x，那么一般是找到最大的 $\delta k(x)$，就可以认为，新的样本属于第 k 类。

得到了线性判别分析的公式，这里从另外一个角度来看线性判别分析，也就是常说的 Fisher 判别式。其实 Fisher 判别式就是线性判别分析（LDA），只是在讨论 Fisher 判别式的时候，更侧重于 LDA 的数据降维的能力。

在应用统计学方法解决模式识别、机器学习中的问题的时候，有一个问题总是会出现：维数问题。很多在低维空间里可以解析或者计算的算法，在高维空间里面往往行不通，因此，数据降维就成了处理实际问题中的关键。

单纯的从数学角度考虑，将 d 维空间的数据压缩称为 1 维的数据是非常的容易的。然而，即便样本集合在 d 维空间里形成若干紧凑的相互分得开的集合，当把它们投影到某一个直线上的时候，就可能使得这些样本混合在一起无法分别开来。但，在实践中发现，总可以找到某个方向，使得在这个方向的直线上，样本的投

影能分开的最好。

Fisher 线性判别式的基本思想就是要最大化类间距离，同时最小化类内距离。这里就定义一个比值：J(w)=(m~1－m~2)2s~21+s~22，J(w)=(m~1－m~2)2s~12+s~22。

将这个比值最大化，就可以达到最大化类间距离的同时，最小化类内距离了。对于原始的数据而言，其类内方差可以写为：s2k=∑n∈Ck(yn－mk)2=∑x∈Ck(x－mk)(x－mk)T，sk2=∑n∈Ck(yn－mk)2=∑x∈Ck(x－mk)(x－mk)T。

投影后的类内方差为：$s\tilde{}2k = \sum n \in Ck(yn - m\tilde{}k)2 = \sum x \in Ck(wTx - wTmk)(wTx - wTmk)T = \sum x \in Ck(wT(x - mk)(x - mk)Tw) = wTs2kw$，$s\sim k2 = \sum n \in Ck(yn - m\sim k)2 = \sum x \in Ck(wTx - wTmk)(wTx - wTmk)T = \sum x \in Ck(wT(x - mk)(x - mk)Tw) = wTsk2w$。

这里定义原始数据的类内总方差为：Sw=s21+s22，　Sw=s12+s22。

那么投影后的类内总方差为：s~21+s~22=wT(s21+s22)w=wTSw，s~12+s~22=wT(s12+s22)w=wTSw，投影后，类间距离也可以写成相似的形式：(m~1－m~2)2=(wTm1－wTm2)2=wT(m1－m2)(m1－m2)Tw，(m~1－m~2)2=(wTm1－wTm2)2=wT(m1－m2)(m1－m2)Tw。

这里定义：Sb=(m1－m2)(m1－m2)T，Sb=(m1－m2)(m1－m2)T，那么，投影后的类间距离可以写成为：(m~1－m~2)2=(wTm1－wTm2)2=wT(m1－m2)(m1－m2)Tw=wTSbw，(m~1－m~2)2=(wTm1－wTm2)2=wT(m1－m2)(m1－m2)Tw=wTSbw。

这样，就是可以把 Fisher 判别式的判别准则重写为下面这个形式：J(w)=(m~1－m~2)2s~21+s~22=wTSbwTSw，J(w)=(m~1－m~2)2s~12+s~22=wTSbwTSw。其中：Sb=(m1－m2)(m1－m2)T，Sb=(m1－m2)(m1－m2)T，Sw=∑x∈C1(x－m1)(x－m1)T+∑x∈C2(x－m2)(x－m2)T，Sw=∑x∈C1(x－m1)(x－m1)T+∑x∈C2(x－m2)(x－m2)T。

上面这个式子对 w 求导之后，可以得到：w=Sw－w(m1－m2)，w=Sw－w(m1－m2)。

这样，我们就可以得到将投影后的类间距离最大化，同时投影后的类内距离最小化的之间 w。

三、数据集介绍

Iris 数据集是常用的分类实验数据集，由 Fisher, 1936 收集整理。Iris 也称鸢尾花卉数据集，是一类多重变量分析的数据集。数据集包含 150 个数据集，分为 3 类，每类 50 个数据，每个数据包含 4 个属性。可通过花萼长度，花萼宽度，花瓣长度，花瓣宽度 4 个属性预测鸢尾花卉属于（Setosa，Versicolour，Virginica）三个种类中的哪一类。

Sonar 数据集包含 208 个数据集，有 60 维，分为 2 类，第一类为 97 个数据，第二类为 111 个数据，每个数据包含 60 个属性，用于区分岩石与矿井，是在数据挖掘、数据分类中非常常用的测试集、训练集。

三、算法设计

1.获得数据

从 UCI 官网中找到相应的数据集（iris、sonar 数据集），然后从对应的目录下复制数据，再分别创建 iris.txt、sonar.txt 文件。

2.处理数据集

因为 matlab 中没有直接可以读取的函数，所以要对数据集做相应的处理，如：把 txt 文件中数据的分隔符‘,’用‘ ’（空格）代替，把类标用“1”或者“0”代替这样，可以很好的方便函数去读取数据。

3.数据的读入

对于 iris 数据集，我用的 matlab 中的 txtread 函数去读取数据，因为 iris 数据的维数很低格式化输入不繁琐，也支持有分隔符的函数。把读出的每一列作为相应的属性，命名为 attrib 列向量。

对于 sonar 数据集，我用 matlab 中的 importdata 函数去读取数据，因为 sonar 数据的维数很高不适合用 textread 格式化读取数据，并且用修改过后的数据集可以很方便的读取数据。可以读成一个 208*61 矩阵，其中的最后一列作为列标（0(R)或 1(M)），在对数据操作时只对其他进行操作。

4.产生随机数

因为要采用随机抽样法（十次平均）为了不生成重复的随机数在这里我用

matlab 中的 randperm 函数，randperm（m）其中 m 作为产生行向量的范围，即从 1~m 的正整数。把新生成的数组前 60%作为训练样本的下标数（从原始数据拿出来的凭证），后 40%作为测试样本的下标数，但再 sonar 数据中正确率较低，采用 9：1 的比列取训练和测试样本。

5.对训练样本分类

采用随机抽样法会打乱原先已排好的顺序，所以再用 fisher 线性判别前先对打乱的数据进行排序，重新构建成 class1（第一类）、class2（第二类）矩阵，矩阵中存的依然是数据的序号是一个行向量。

6.重建数据矩阵并计算均值向量、类内离散度矩阵、类间离散度矩阵

还原原始数据用 class1_row、class2_row 和 text_row 表示，class1、class2 矩阵用做 attrib 的索引序号，这样可以得到第一类的原始矩阵、第二类的原始矩阵和测试的原始矩阵，再对原始数据矩阵求第一类、第二类的均值向量，再求他们的类内离散度矩阵，求类间离散度矩阵。

7.求最佳投影方向并重新分类

通过$\mathrm{w} = S_w^{-1}(m_1 - m_2)$公式去求最佳投影方向，通过$w_0 = (m_1 + m_2)/2$求出阈值 w0，然后通过最佳投影方向的转置与 text_row 相乘得到一个行向量，其中每一个的值都与阈值比较，大于的为第一类，小于的为第二类。

8.计算正确率

用程序计算出的类标与原始类标进行比较，当不一样时就增加 h（原先为 0），即 h 加 1，正确率%=100*（总的测试集样本的个数-错误的个数）/样本个数。

四、算法结果展示

1.Iris 数据集结果

```
>> iris123
正确率率为100.000%
正确率率为100.000%
正确率率为100.000%
正确率率为100.000%
正确率率为100.000%
正确率率为100.000%
正确率率为100.000%
正确率率为100.000%
正确率率为100.000%
正确率率为100.000%
平均错误0个，平均正确率为100.000%
时间已过 0.063986 秒。
```

2.Sonar 数据集结果

（1）训练：测试=6：4

```
>> sonar
正确率为54.217%
正确率为39.759%
正确率为44.578%
正确率为44.578%
正确率为42.169%
正确率为51.807%
正确率为40.964%
正确率为53.012%
正确率为51.807%
正确率为49.398%
平均错误43个，平均正确率为47.229%
时间已过 0.396233 秒。
```

（2）训练：测试=9：1

```
>> sonar_1
正确率为86.747%
正确率为85.542%
正确率为90.361%
正确率为89.157%
正确率为90.361%
正确率为86.747%
正确率为87.952%
正确率为83.133%
正确率为89.157%
正确率为84.337%
平均错误10个，平均正确率为87.349%
时间已过 0.379650 秒。
```

五、源代码

1.iris 数据集

```
tic
clear all
[attrib1, attrib2, attrib3, attrib4, class] = textread('iris.txt', '%f%f%f%f%s', 'delimiter', ' ');  %将数据导入到数组中 其中 class 表示标号
k=[];
for j=1:10
    a=randperm(100);        %产生一个行向量       内容为 1：100 的乱序
    a_test=a(61:100);       %产生一个 test 变量     记录测试的下标
    a_train=a(1:60);        %产生一个 train 变量    记录训练的下标   训练与测试比为 6：4
    class1=[];class2=[];n=1;m=1;    %class1 class2 用于存放第一类、第二类元素的数组 m 计数
    %将 train 分类
    for i=1:60
        if(strcmp(class{a_train(i)},'Iris-setosa'))
            class1(m)=a_train(i);m=m+1;
```

```
        elseif(strcmp(class{a_train(i)},'Iris-versicolor'))
            class2(n)=a_train(i);n=n+1;
        end

    end

    %重新组建第一类 第二类 相应的矩阵 按列进行排序
    class1_row=[attrib1(class1)    attrib2(class1)    attrib3(class1)
attrib4(class1)]';
    class2_row=[attrib1(class2)    attrib2(class2)    attrib3(class2)
attrib4(class2)]';
    test_row=[attrib1(a_test)     attrib2(a_test)     attrib3(a_test)
attrib4(a_test)]';

    %求均值列向量 class1_ave  class2_ave
    class1_ave=sum(class1_row,2)/length(class1);
    class2_ave=sum(class2_row,2)/length(class2);

    %求类内离散度矩阵 s_1 s_2  总类内离散度矩阵 s_w
    s_1=(class1_row-class1_ave)*(class1_row-class1_ave)';
    s_2=(class2_row-class2_ave)*(class2_row-class2_ave)';
    s_w=s_1+s_2;

    %求类间离散度矩阵 s_b
    s_b=(class1_ave-class2_ave)*(class1_ave-class2_ave)';

    %求最佳投影方向 w
    w=inv(s_w)*(class1_ave-class2_ave);
```

```
    %不考虑 先验概率情况   计算 w0
    m_1=w'*class1_ave;
    m_2=w'*class2_ave;
    w_0=(m_1+m_2)/2;

    %计算经过投影之后的值
    t_1=w'*class1_row;
    t_2=w'*class2_row;
    t_3=w'*test_row;
    t=[t_1 t_2 t_3];

    %计算正确率
    h=0;                %错误的个数
    index=find(t_3>w_0);        %index 为列向量
    k_1=a(index+60)>50;
    h=sum(k_1);
    index=find(t_3<=w_0);        %index 为列向量
    k_1=a(index+60)<50;
    k(j)=h+sum(k_1);
    fprintf('正确率率为%.3f%%\n',100-100*k(j)/40)
end
fprintf('平均错误%d 个, 平均正确率为%.3f%%\n',floor(sum(k)/10),100-10*sum(k)/40)
toc
```

2.sonar

```
clear all
tic
```

```
delimiterIn=' ';                                %间隔为 空格（ ）
A = importdata('sonar1.txt', delimiterIn);   %将数据导入到数组中 其中 class 表示标号
B=A(:,1:60);k=[];                               %将除列标的数据给 B 矩阵
for j=1:10
    a=randperm(208);        %产生一个行向量        内容为 1：208 的乱序
    a_test=a(126:208);       %产生一个 test 变量      记录测试的下标
    a_train=a(1:125);       %产生一个 train 变量     记录训练的下标   训练与测试比为 6：4
    class1=[];class2=[];n=1;m=1;    %class1 class2 用于存放第一类、第二类元素的数组 m 计数
    %将 train 分类
    for i=1:125
        if(A(a_train(i),61)==1)
            class1(m)=a_train(i);m=m+1;
        elseif(A(a_train(i),61)==0)
            class2(n)=a_train(i);n=n+1;
        end

    end

    %重新组建第一类 第二类 相应的矩阵 按列进行排序
    class1_row=B(class1,:)';
    class2_row=B(class2,:)';
    test_row=B(a_test,:)';

    %求均值列向量 class1_ave  class2_ave
    class1_ave=sum(class1_row,2)/length(class1);
```

```
class2_ave=sum(class2_row,2)/length(class2);

%求类内离散度矩阵 s_1 s_2  总类内离散度矩阵 s_w
s_1=(class1_row-class1_ave)*(class1_row-class1_ave)';
s_2=(class2_row-class2_ave)*(class2_row-class2_ave)';
s_w=s_1+s_2;

%求类间离散度矩阵 s_b
s_b=(class1_ave-class2_ave)*(class1_ave-class2_ave)';

%求最佳投影方向 w
w=inv(s_w)*(class1_ave-class2_ave);

%不考虑 先验概率情况  计算 w0
m_1=w'*class1_ave;
m_2=w'*class2_ave;
w_0=(m_1+m_2)/2;

%计算经过投影之后的值
t_1=w'*class1_row;
t_2=w'*class2_row;
t_3=w'*test_row;
t=[t_1 t_2 t_3];

%计算正确率
h=0;                %错误的个数
index=find(t_3>w_0);        %index 为列向量
k_1=a(index+60)>97;
h=sum(k_1);
```

```
    index=find(t_3<=w_0);          %index 为列向量
    k_1=a(index+60)<98;
    k(j)=h+sum(k_1);
    fprintf('正确率为%.3f%%\n',100-100*k(j)/83)
end
fprintf('平均错误%d 个，平均正确率为%.3f%%\n',floor(sum(k)/10),100-
10*sum(k)/83)
toc
```