一、问题描述

基于 2 类各 5000 个 57 维的样本数据，采用 fisheri 线性判别产生 100 个预判正确率略超过 50%的弱分类器，基于这些弱分类器进行设计正确率超过 90%的 Adaboost 分类器。

二、方法介绍

AdaBoost 是英文"Adaptive Boosting"（自适应增强）的缩写，它的自适应在于：前一个基本分类器被错误分类的样本的权值会增大，而正确分类的样本的权值会减小，并再次用来训练下一个基本分类器。同时，在每一轮迭代中，加入一个新的弱分类器，直到达到某个预定的足够小的错误率或达到预先指定的最大迭代次数才确定最终的强分类器。

Adaboost 算法可以简述为三个步骤：

（1）首先，是初始化训练数据的权值分布 d。假设有 n 个训练样本数据，则每一个训练样本最开始时，都被赋予相同的权值：1/n。

（2）然后，训练弱分类器 ht。具体训练过程中是：如果某个训练样本点，被弱分类器 ht 准确地分类，那么在构造下一个训练集中，它对应的权值要减小；相反，如果某个训练样本点被错误分类，那么它的权值就应该增大。权值更新过的样本集被用于训练下一个分类器，整个训练过程如此迭代地进行下去。

（3）最后，将各个训练得到的弱分类器组合成一个强分类器。各个弱分类器的训练过程结束后，加大分类误差率小的弱分类器的权重 ai，使其在最终的分类函数中起着较大的决定作用，而降低分类误差率大的弱分类器权重，使其在最终的分类函数中起着较小的决定作用。

换而言之，误差率低的弱分类器在最终分类器中占的权重较大，否则较小。

三、结果与分析

准确率为 84.25%

数据太多，截取部分数据如下：

数据序号	1	2	3	4	5	6	7	8	9	10
给定分类	1	1	1	1	1	1	1	1	1	1
判别分类	1	1	2	1	1	1	1	1	1	1

准确率会随着迭代次数增加、弱分类器数增加而增大。然而由于数据维数和

样本数量过大，准确率很难进一步在可接受的时间内使用计算机进一步提高。

四、程序源代码

```
%数据处理
%************************************************
load('data.mat');
data=data';
data1=data(1:57,1:5000);
data2=data(1:57,5001:10000);
%分别选取类 1 和类 2 的前 4000 个作为训练样本
x1=data1(1:56,1:4000);
x2=data2(1:56,1:4000);
x_train=[x1 x2];
[m_train,n_train]=size(x_train);
y1=data1(57,1:4000);
y2=data2(57,1:4000);
y_train=[y1 y2];
ys_train=ones(1,8000);
for i=1:8000
    if(y_train(i)==2)
       ys_train(i)=-1;
    end
end
%分别选取类 1 和类 2 的后 1000 个作为测试样本
xt1=data1(1:56,4001:5000);
xt2=data2(1:56,4001:5000);
x_test=[xt1 xt2];
[m_test,n_test]=size(x_test);
yt1=data1(57,4001:5000);
yt2=data2(57,4001:5000);
y_test=[yt1 yt2];
ys_test=ones(1,2000);
for i=1:2000
    if(y_train(i)==2)
       ys_train(i)=-1;
    end
end
%************************************************

%Adaboost 模型计算
%************************************************
%初始化
T=2000; %最大迭代次数
```

```
L=400;  %弱分类器个数
%训练出的 fisher 弱分类器
w=zeros(m_train,L);
m=zeros(m_train,L);
b=zeros(1,L);
h=zeros(L,n_train);
ps=zeros(L,1);  %训练出的 fisher 弱分类器的正确率
%挑选出的 fisher 弱分类器
wc=zeros(m_train,T);
mc=zeros(m_train,T);
bc=zeros(1,T);
hc=zeros(T,n_train);
%权重分布初始化
d1=ones(1,n_train)/n_train;
d2=zeros(T,n_train);
d=[d1;d2];
%错误率和权重初始化
es=zeros(L,T);
k=zeros(1,T);   %挑选因子初始化
emin=zeros(1,T);
a=zeros(L,1);
%训练 L 个弱分类器
K=n_train/2;
R1=unidrnd(K,L,K);
R2=R1+K*ones(L,K);
q=K/50;
for i=1:L
    [w(:,i),m(:,i),b(i)]=fisherwch2(x_train,R1(i,:),R2(i,:),q);
    h(i,:)=fisherwchs(w(:,i),m(:,i),b(i),x_train);
end
%迭代
for t=1:T
    %计算此次迭代中 L 个弱分类器分别对应的错误率
    for i=1:L
        countn=0;
        for j=1:n_train
            if (ys_train(j)~=h(i,j))
                es(i,t)=es(i,t)+d(t,j);
                countn=1+countn;
            end
        end
        ps(i)=1-countn/n_train;
    end
    %找到最小错误率对应的那个分类器并加以保存
```

```
    [emin(t),k(t)]=min(es(:,t));
    wc(:,t)=w(:,k(t));
    mc(:,t)=m(:,k(t));
    bc(:,t)=b(k(t));
    hc(t,:)=h(k(t),:);
    %计算权重
    a(t)=0.5*(log(1-emin(t))-log(emin(t)));
    %更新权重分布
    for i=1:n_train
        if (ys_train(i)==hc(t,i))
            d(t+1,i)=0.5*d(t,i)/(1-emin(t));
        else
            d(t+1,i)=0.5*d(t,i)/emin(t);
        end
    end
    %如果错误率等于 0 或者大于等于 0.5 退出迭代
    if(emin(t)==0 || emin(t) >=0.5)
        break;
    end
end
%************************************************

%判别
%************************************************
f=zeros(1,n_test);
for j=1:(t-1)
    f=f+a(j)*fisherwchs(wc(:,j),mc(:,j),bc(j),x_test);
end
ys_forecast=ones(1,n_test);
y_forecast=ones(1,n_test);
for i=1:n_test
    if (f(i)<0)
        ys_forecast(i)=-1;
        y_forecast(i)=2;
    end
end
%************************************************

%正确率统计
%************************************************
count=0;
for i=1:n_test
    if (y_forecast(i)==y_train(i))
        count=count+1;
```

```
    end
end
p=count/n_test;
%************************************************


%采用 fisher 线性判别建立 i 个弱分类器 h
function [w,m1,b] = fisherwch(x_train,R1,R2,q)
%分别选取前 q 个类 1 和类 2 随机测试样本的数据
w1=x_train(:,R1(1:q));
w2=x_train(:,R2(1:q));
% K1=int32(0.5*n_train*(i-1)/L);
% K2=int32(0.5*n_train*i/L);
% w1=x_train(:,(K1+1):K2);
% w2=x_train(:,(K1+4001):(K2+4000));
%计算类 1 和类 2 的均值
m1=mean(w1,2);
m2=mean(w2,2);
%计算类 1 和类 2 的类内离散度矩阵
s1=cov(w1');
s2=cov(w2');
%类内总离散度矩阵
sw=s1+s2;
%计算基于 Fisher 线性判别的投影方向
w=inv(sw)*(m1-m2);
%计算阈值
b=-1/2*w'*(m1+m2);


%弱分类器的判别
function h = fisherwchs(w,m1,b,X)
%和类 1 均值 m1 对应的 fisher 判别式值符号相同的数据被归为类 1，否则归为类 2
[m_size,n_size]=size(X);
h=ones(1,n_size);
t=w'*m1+b;
for i=1:n_size
    if (((w'*X(:,i)+b)*t)<0)
        h(i)=-1;
    end
end
```