# 使用Fisher线性分类器实现手写数字识别

## 实验目的

通过实现Fisher线性分类器算法，实现手写数字的识别。

## 实验方法

### 搭建平台

使用Windows系统自带的画图工具分别构建训练样本集和测试样本集，使用MATLAB GUI编程环境对它们进行处理、识别。

### 特征描述

将手写部分从图像中分割，并转化为灰度图像，而后将其分割为6\*6=36个小图片，用每个小图片中黑色像素占总像素的比重作为该样本的特征，也就是一个36维的向量。

子函数get\_feature(image)代码如下：

function[feature] = get\_feature(image)

%从图像中分割出手写部分并转为灰度

image2 = pretreat(image);

A = mat2cell(image2,[10,10,10,10,10,10],[7,7,7,7,7,7]);

B = reshape(A,1,36);

for i=1:length(B)

temp = cell2mat(B(i));

%feature(i)表示黑像素点占一个小块的比例

feature(i) = sum(temp(:) ~= 255) / numel(temp);

end

%feature

end

### 关键算法

#### 两类情况下的判别函数

多元正态分布的概率密度函数为：



其中，是d维列向量；

是d维均值向量；

是d\*d维协方差矩阵。

对于某一待测试样本，其在多元正态概率型（）下的判别函数为

。

决策面方程为

，

即

。

下面讨论的情况。

这是一种比较简单的情况，表示各类的协方差矩阵都相等，从几何来看，相当于各类样本集中于以该类均值点为中心的同样大小和形状的超椭球内。

若去掉与i无关的项，并假设各类的先验概率都相等，则判别函数可以写成：

，

其中



。

#### 决策面方程

在两类情况下，决策面方程应满足



即：



其中





以上式子等价于



其中



。

#### 线性判别函数的基本概念

由前面的推导，我们已经知道，两类情况下的决策面方程为，即。

假设和都在决策面上，则有



或



这表明，w是决策面的法向量。一般来说，决策面把特征空间分成两个半空间，当x在类的决策域时，；当x在的决策域时，。

判别函数可以看作是特征空间中某点x到决策面的距离的一种代数度量。因为若把x写成



式中，

是x在决策面上的射影向量；

r是x到决策面的垂直距离；

是w方向上的单位向量。

那么，，或写作。

总之，判别函数正比于x点到决策面的超平面的代数距离（带正负号）。当x在决策面正侧时，；在负侧时，。

#### Fisher线性判别

两类的线性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界面。

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。

讨论两类分类的问题。训练样本集是，其中，类的样本为，类的样本为。我们要寻找一个投影方向w，投影以后样本变为



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为



总类内离散度矩阵为



类间离散度矩阵为



在投影后的一维空间，两类的均值分别为



类内离散度矩阵不再是一个矩阵，而是一个值



总类内离散度为



而类间离散度就成为两类均值差的平方



前面已经说过，我们要选择投影方向，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，这一目标可以表示为如下准则



将前述定义带入，Fisher判别准则变为



最优投影方向w应满足



求解可得



由于我们只关心w的方向，故可取



在不考虑先验概率的不同时，可取



至此，Fisher线性分类器已经设计完毕。

#### 算法实现

将Fisher线性分类器用MATLAB编程实现如下。

function [num] = g(x,num1,num2)

%参数x:待识别样本，num1,num2:线性分类的两种可能结果，函数返回其中一个可能性较大的数

%线性分类器g(x) = w'x+w0

%其中，w=inv(S1+S2)\*(m1-m2)

%x为待识别样本

%w0取-0.5(m1+m2)'\*inv(S1+S2)\*(m1-m2)

%最后算出g(x)为一个标量

%S为各类的类内离散度矩阵

load S

%mean\_feature为类均值向量

load mean\_feature

S1 = S((36\*num1+1):(36\*num1+1+35),:);

S2 = S((36\*num2+1):(36\*num2+1+35),:);

m1 = (mean\_feature(num1+1,:))';

m2 = (mean\_feature(num2+1,:))';

%为保证其非奇异，在对角线加一常数，对结果几乎不影响

%Sw为总类内离散度矩阵

Sw = S1+S2+diag(10\*ones(1,length(S(1,:))));

%带公式求出投影方向w

w = inv(Sw)\*(m1-m2);%d行1列

w0 = -0.5\*(m1+m2)'\*inv(Sw)\*(m1-m2);%标量

temp = w'\*x'+w0;

if(temp>0)

num = num1;

else

num = num2;

end

end

由于该分类器只考虑了两类情况，故在使用时应多次重复调用若干次，直至每个可能的结果都参与了比较。

这一过程可用MATLAB描述如下。

%get\_number(image),传入image,返回识别结果recog\_num

function[recog\_num] = get\_number(image)

new\_feature = get\_feature(image);

%多次循环，选出可能性最大的数字

for i = 1:9

if (i==1)

%第一次判断

%子函数g(x,num1,num2),返回两个数字中可能性较大的值

temp = g(new\_feature,0,1);

else

temp = g(new\_feature,temp,i);

end

end

recog\_num = temp;

end

## 实验结果

### 训练样本集构建

使用Windows系统自带的画图软件创建训练样本集，使用10个文件夹分别保存10个类的训练集，如下图。



图 三-1 训练样本集

每一个样本都有一个固定的模板图片，用画图工具在该图片的指定位置手写数字，模板图片如下图。

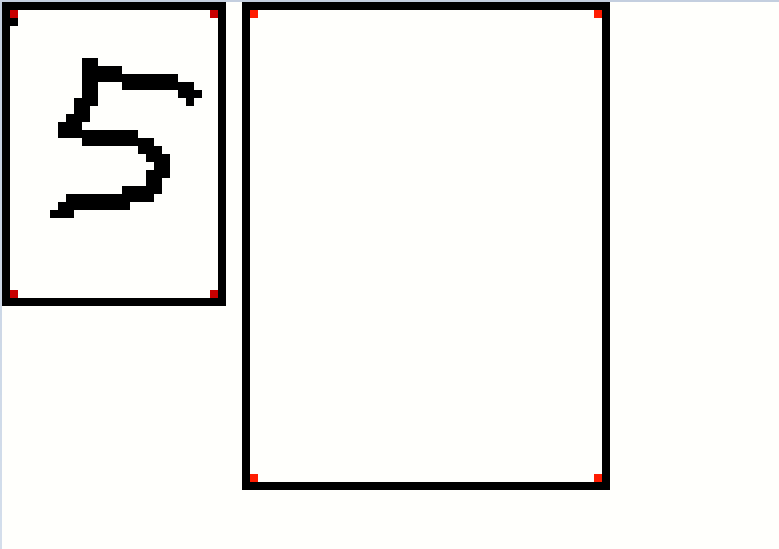


图 三-2 样本构建的模板图片

某一手写完成的样本如下图。

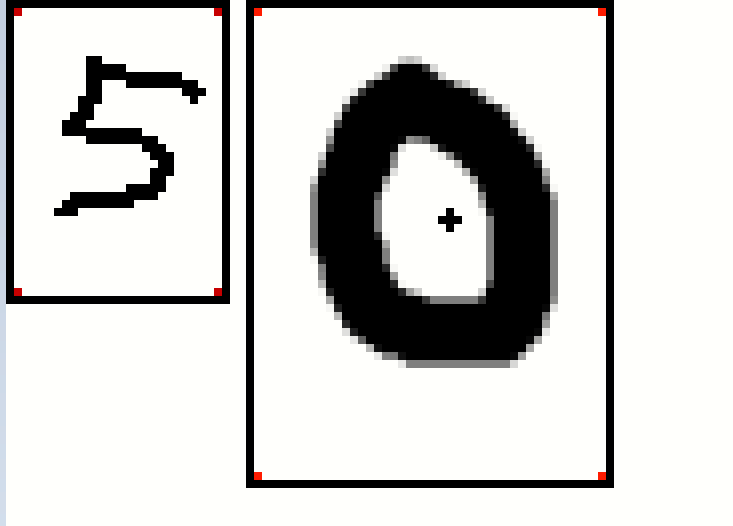


图 三-3 某一手写完成的样本

### 测试集构建

测试集样本的制作与训练集基本相同，不同的是，将所有图片都保存至同一文件夹“测试样本集”中，以供程序读取识别。

本次实验测试样本集如下图。

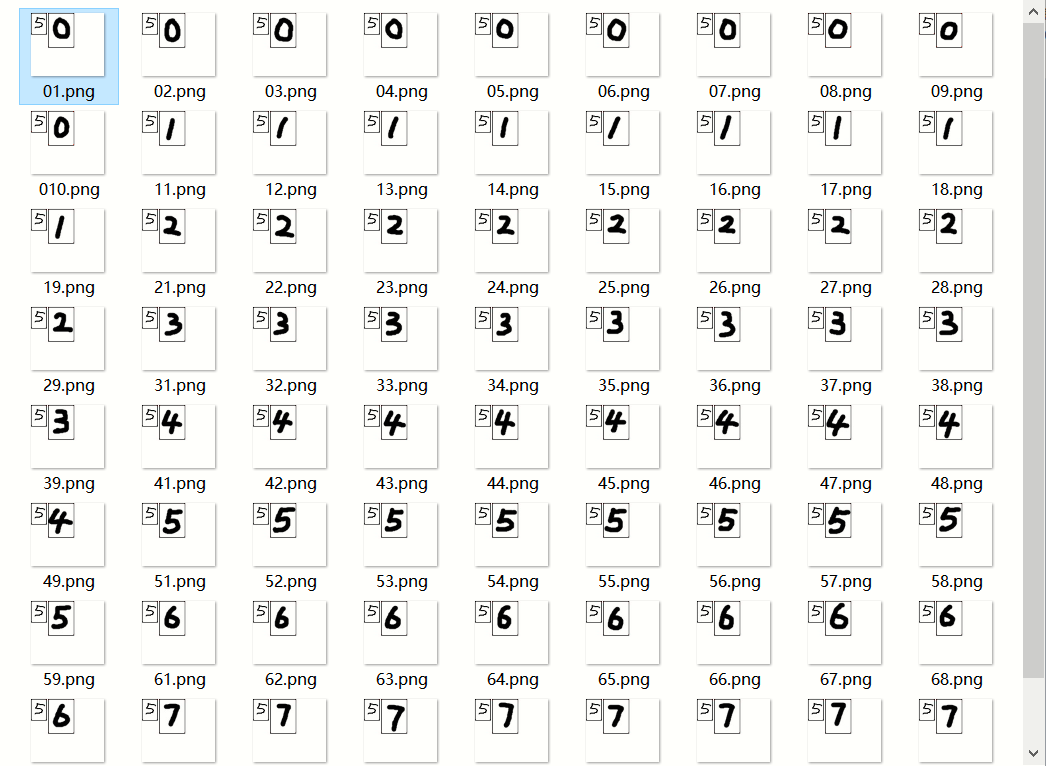


图 三-4 测试样本集

该样本集中有手写数字0-9各10个，它们的命名以各自的数字开头，用于实验中计算正确率。

### 实验结果

实验的运行环境是MATLAB GUI，本次实验各控件布局如下图。

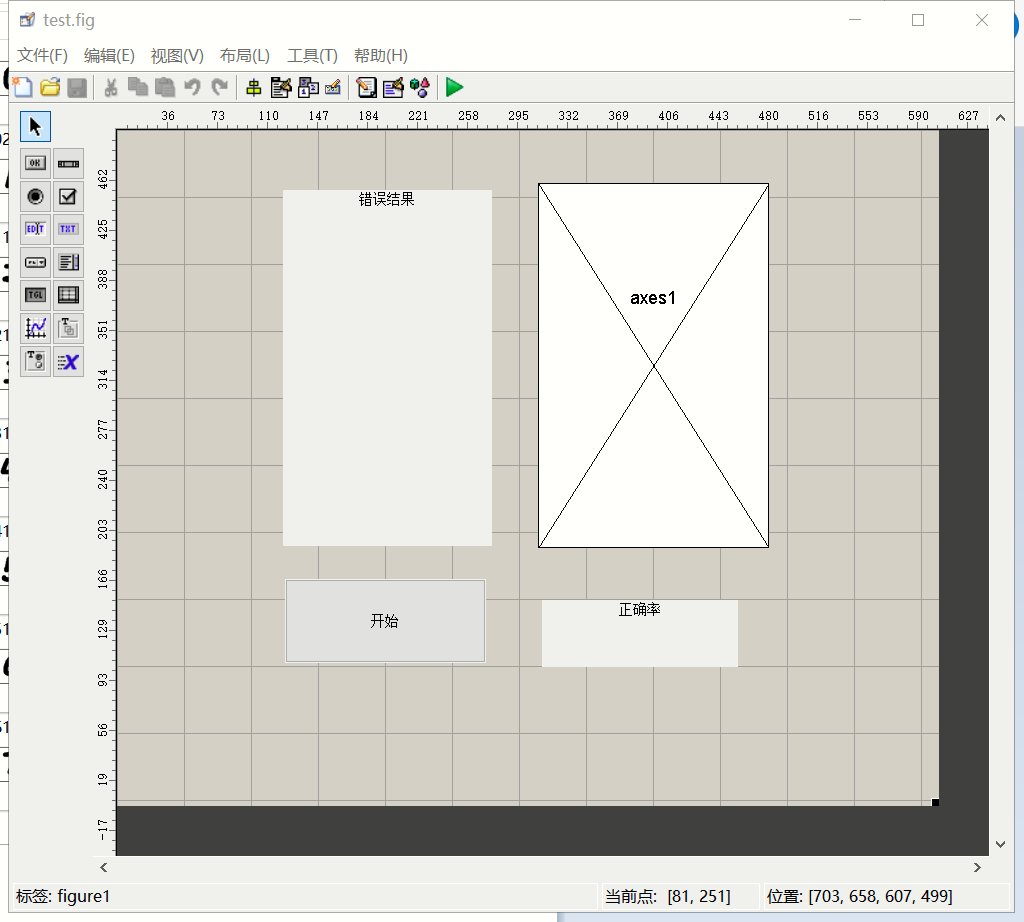


图 三-5 GUI控件的布局

图中已表明了各控件的作用，其中axes1用于运行时显示正在识别的样本。

程序运行时的效果如下图：

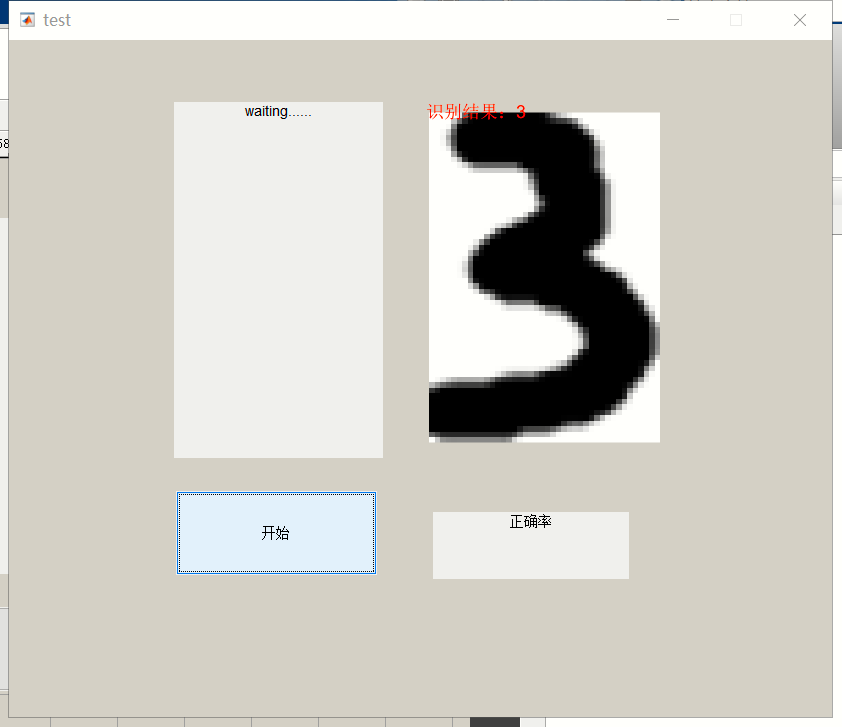


图 三-6 程序正在运行

程序运行结果如下图：

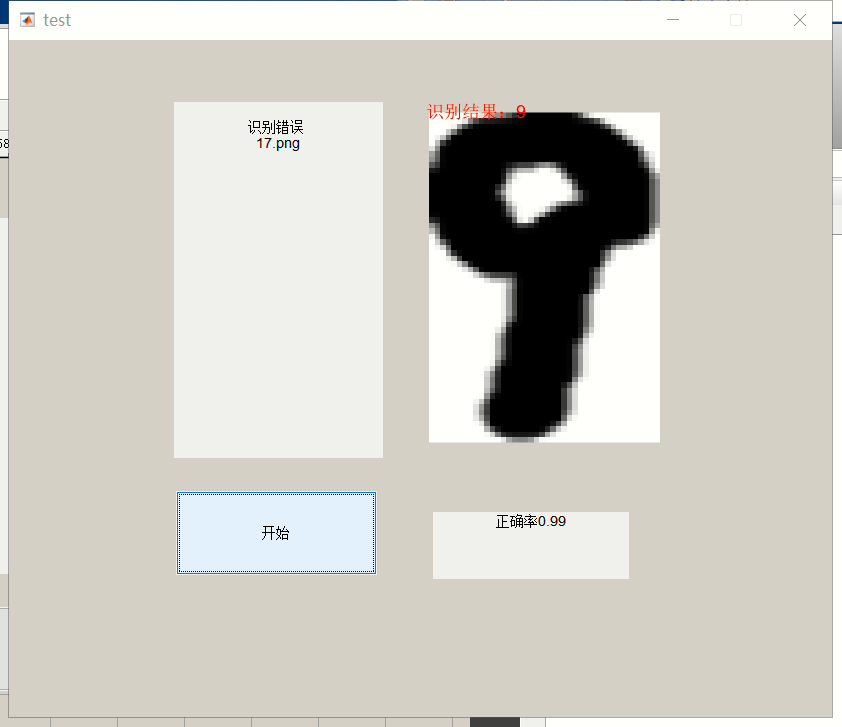


图 三-7 程序运行结束

### 性能量化评估

运行结果显示，程序对100个样本识别的正确率为99%，达到了较好的效果。

## 结果讨论

运行结果显示，对数字1的样本“17.png”识别错误，在MATLAB命令行中对其进行测试如下：

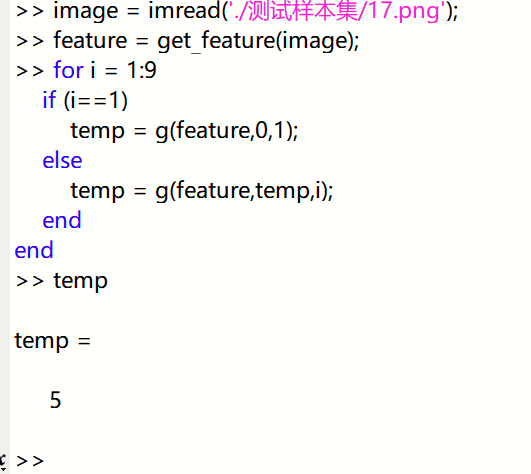


图 四-1 对识别错误的样本进行测试

测试结果说明，程序认为该样本为数字5。

单独执行分类器函数，并打印出运行结果如下图。

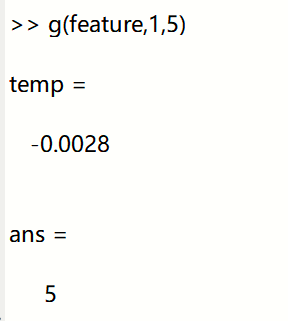


图 四-2 单独执行分类器

在对数字1和其他数字的比较中，判别结果temp的值在0.01~0.06之间，所以在数字5的比较结果相对来说-0.0028非常小，这样通过增加样本的数量就可能使该判别正确。

同时，在运行过程中发现本次实验中，程序的运行速度比先前明显要快，主要是因为使用了MATLAB中的save和load指令，避免了重复计算。

## 结论

本次实验使用MATLAB编程环境，完成了Fisher分类器算法的编写，实现了手写数字的识别，通过对测试样本集（100个样本）的识别，结果显示正确率达99%，表示实验过程中各个样本集的建立以及所用算法较为可靠。