**实验报告**

(郑允芯)

Gentle Boost

1．算法关键点理解：

针对同一个训练及训练出不同的弱分类器，然后把这些弱分类器集合起来，为训练样本分配权重，再构成一个强的最终分类器。它根据每次训练集中每个样本分类正确与否以及上次的总体分类的准确率，来确定每个样本的权值。将修改过权值的新数据集送给下层弱分类器进行训练，最后将每次训练得到的分类器融合起来，作为最后的决策分类器。

2．实验方法：

1. 下载boostingDemo和LabelMeToolbox（包括Gentleboost源代码和训练好的集）。
2. 下载matlab，将matlab工作目录设为boostingDemo的目录下。
3. 打开文件initPath.m，修改addpath参数分别为：

'E:\LabelMeToolbox\LabelMeToolbox'

'E:\boostingDemo\tools'

'E:\boostingDemo\gentleBoost'

1. 打开parameters.m文件修改HOMEIMAGES和HOMEANNOTATIONS参数，这两个参数是指定images和annotations的存放地址。再将最后一行的地址改为存放boostingDemo\data的地址。
2. Parameters.m第二行开始是指定被识别的物体。可选择显示器或者汽车作为被训练的物体。
3. 运行initPath.m，检查是否有问题
4. 运行parameters.m，此时右侧的workspace出现相关参数 ，如下图：

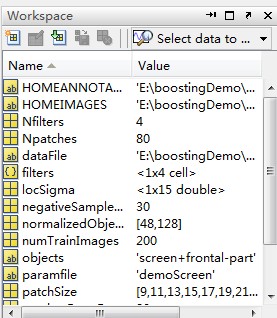


图 1.1

1. 运行demoGentleBoost.m文件，在弹出的窗口中画出自己将要分类的图。左键红色，右键绿色。如下是已画好的图。

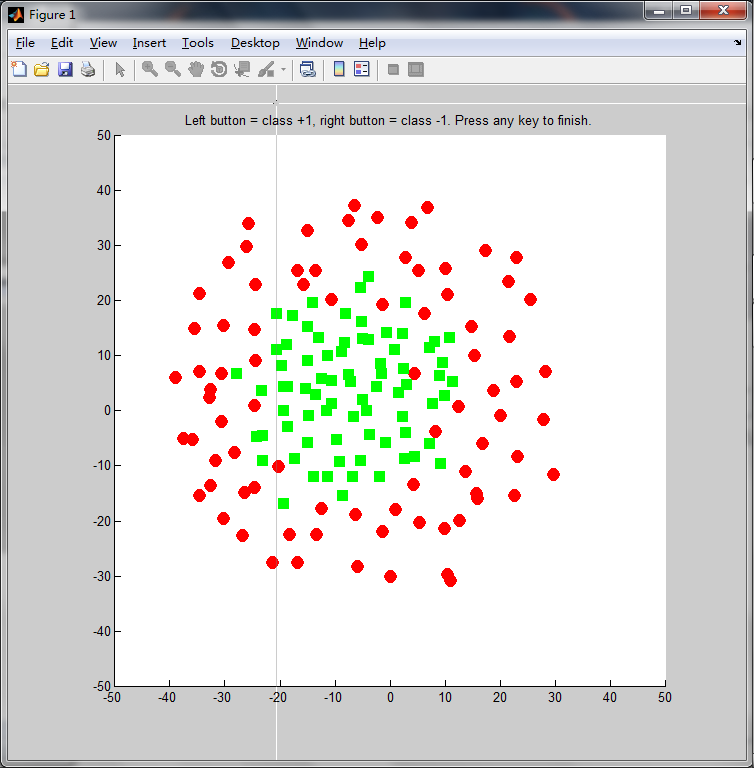


图 1.2

9） 画好后，点击键盘上任意键，例如Enter键，出现下图。左侧为弱分类器，右侧为强分类器。一直点击回车，观察强分类器和弱分类器的变化。代码中将循环限定到100次。循环中的截图如下：

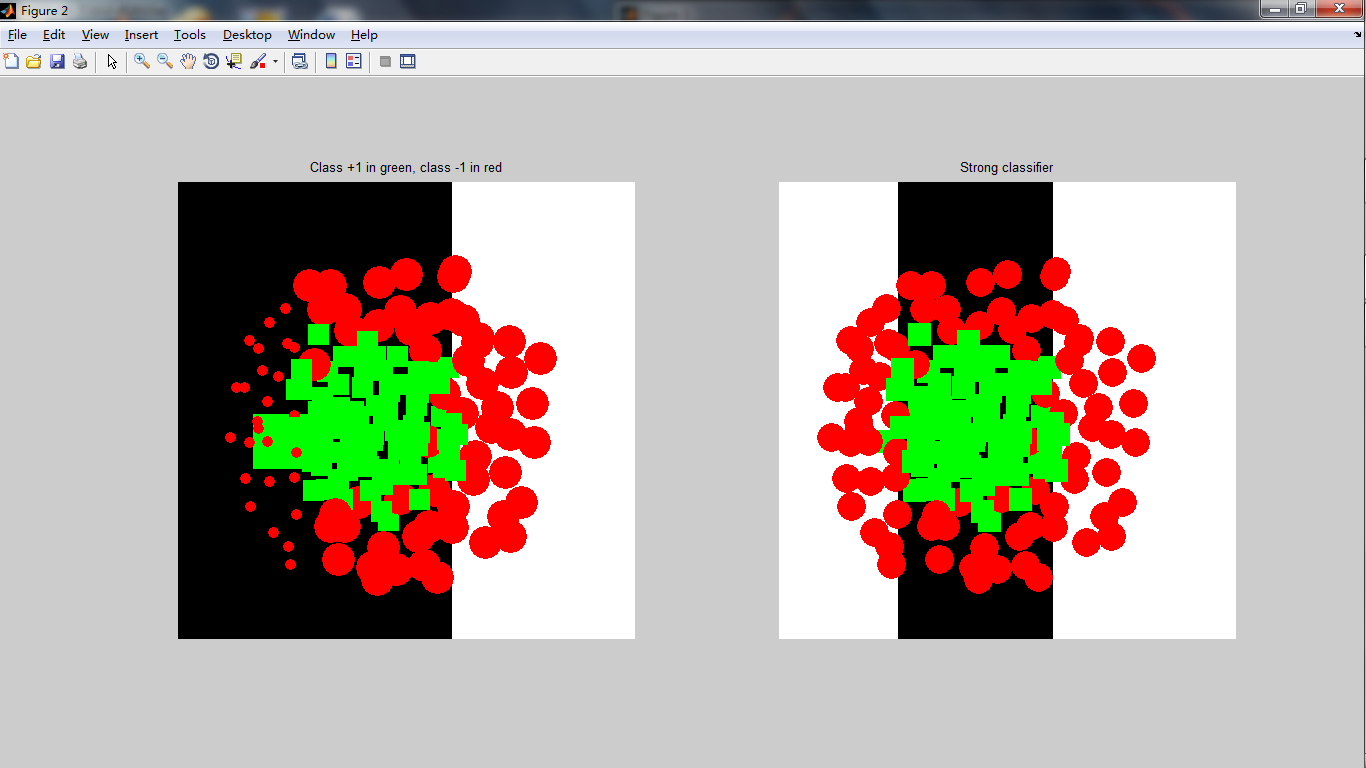
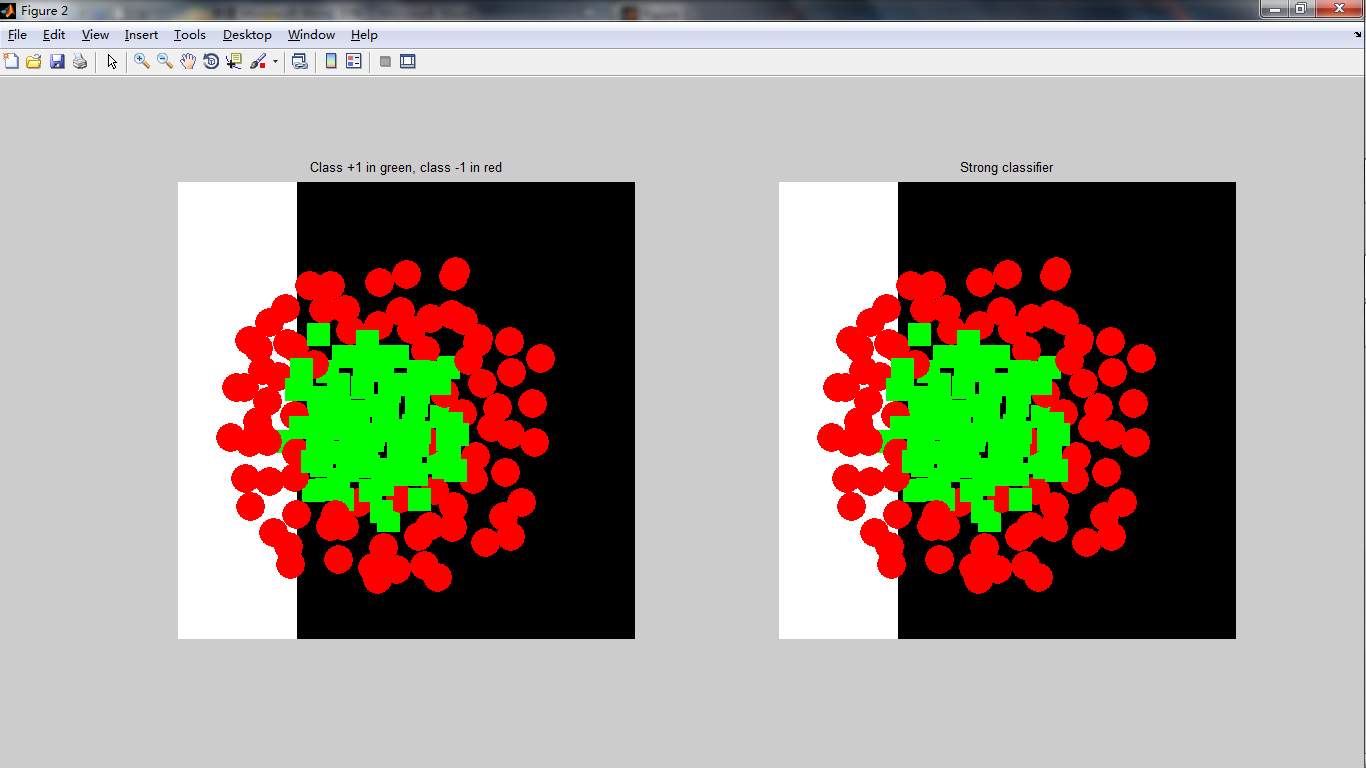


图 1.3

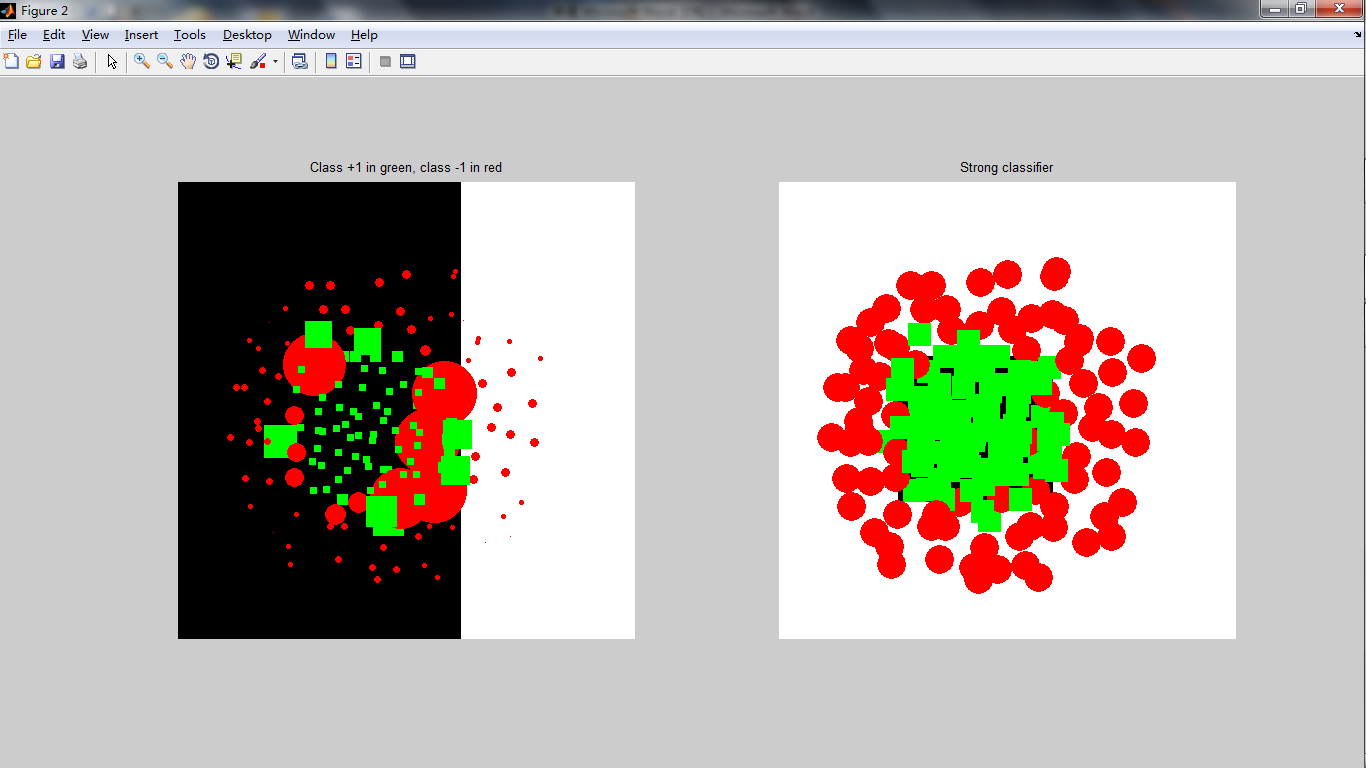
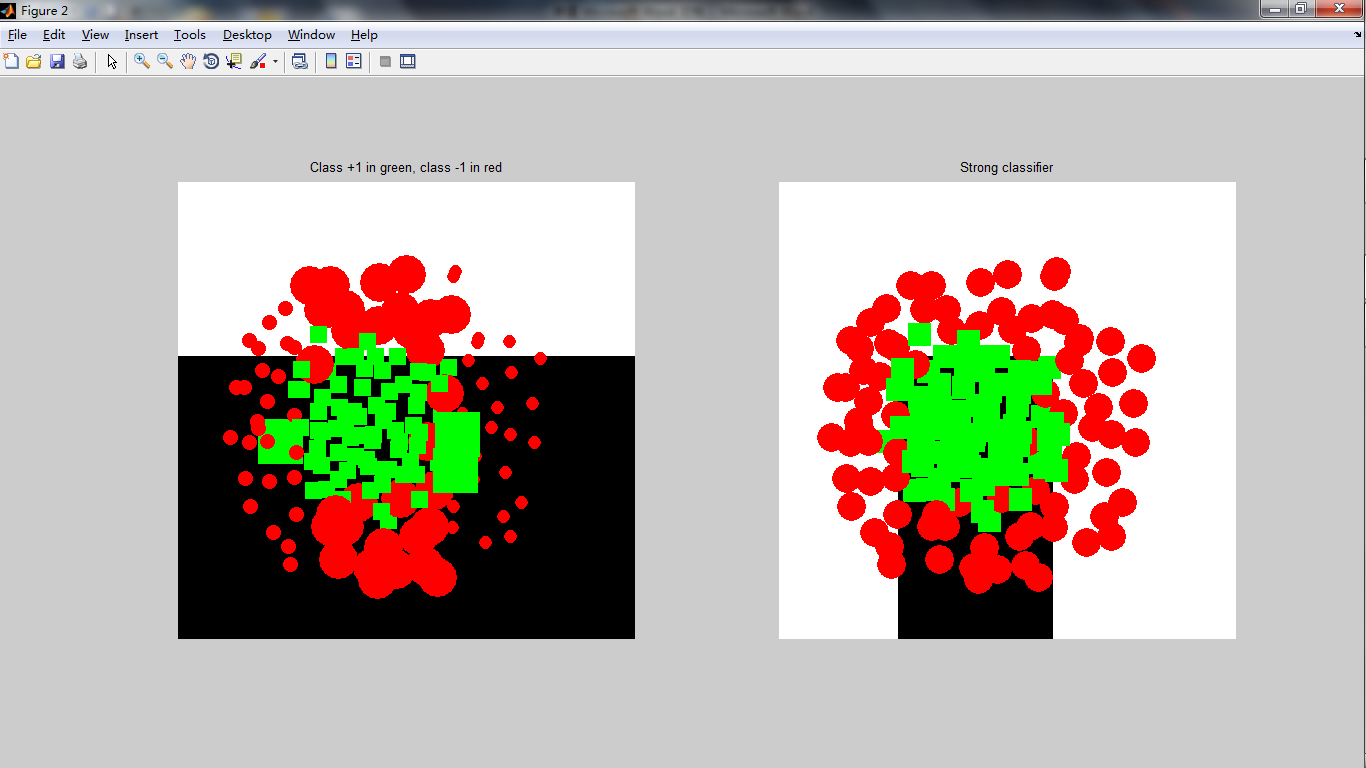


图 1.4

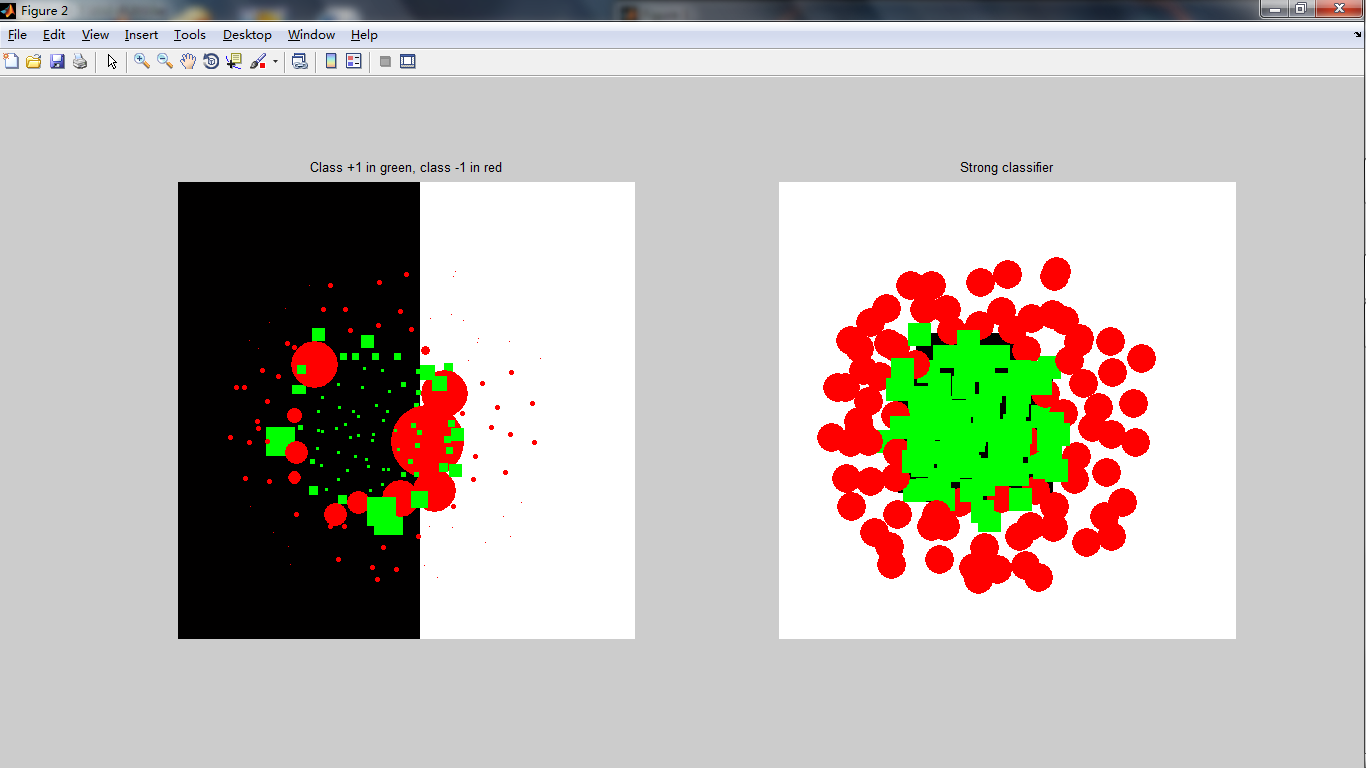
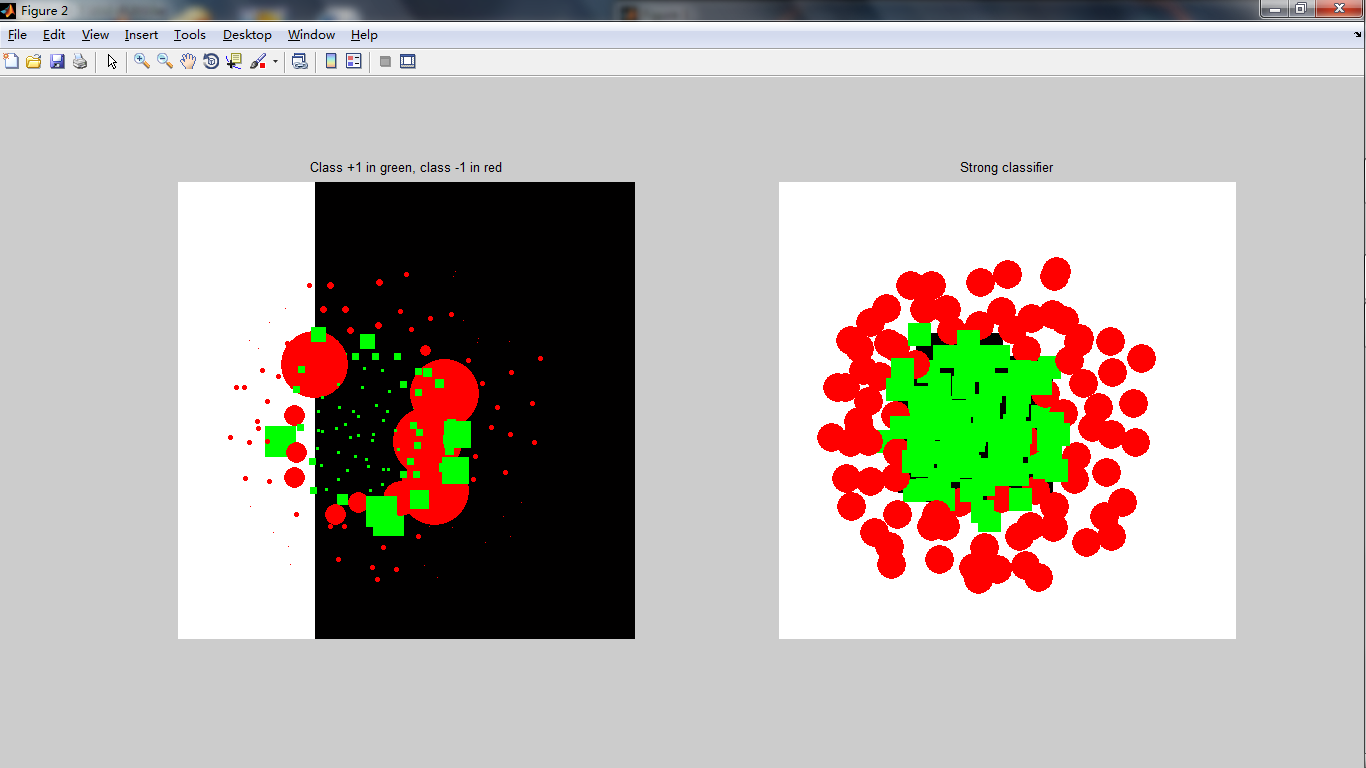


图 1.5

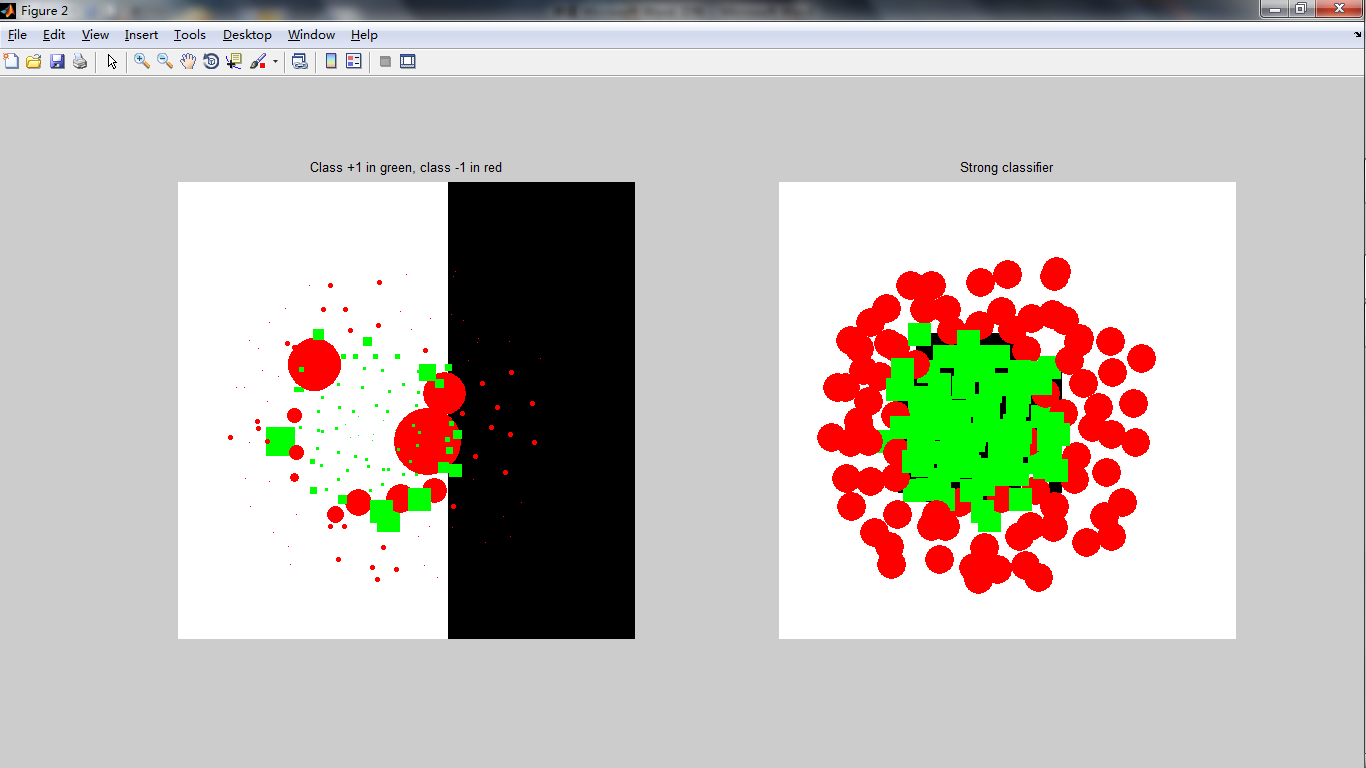
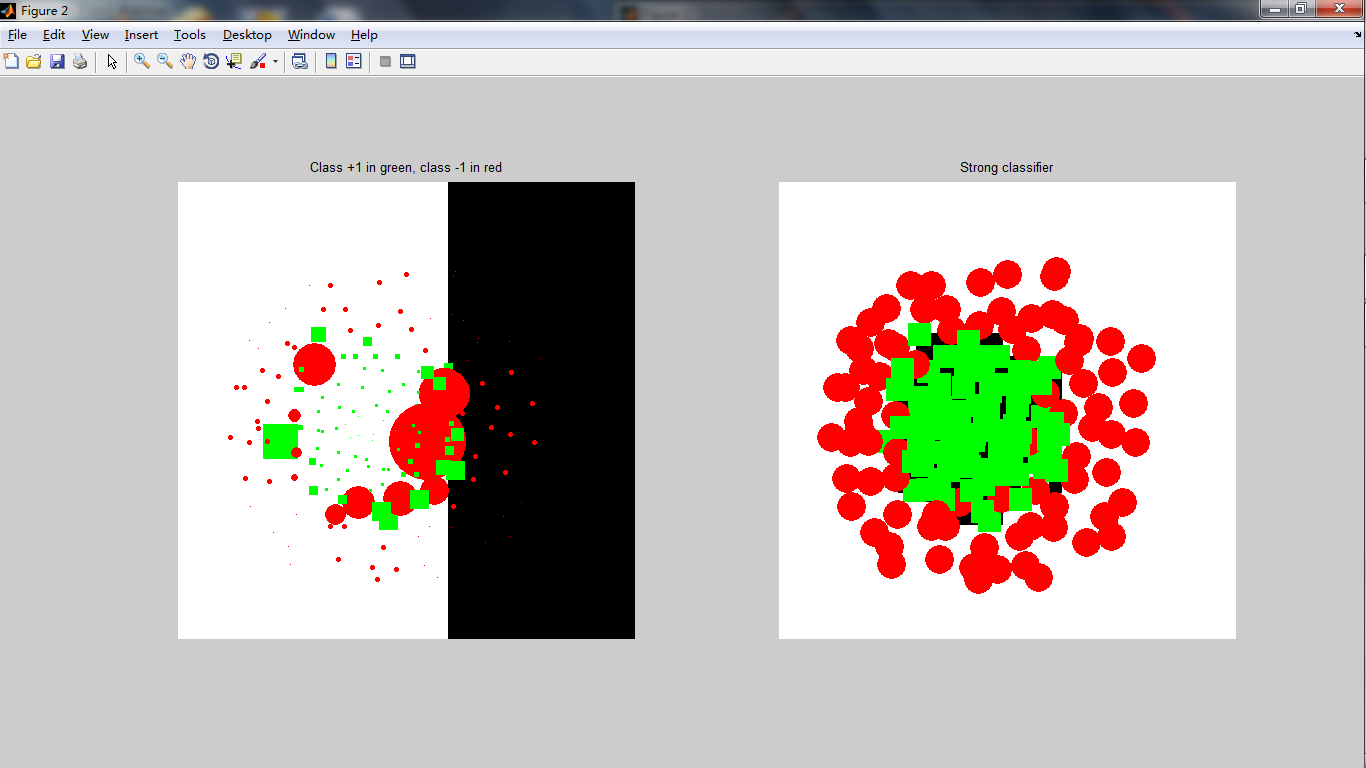


图 1.6

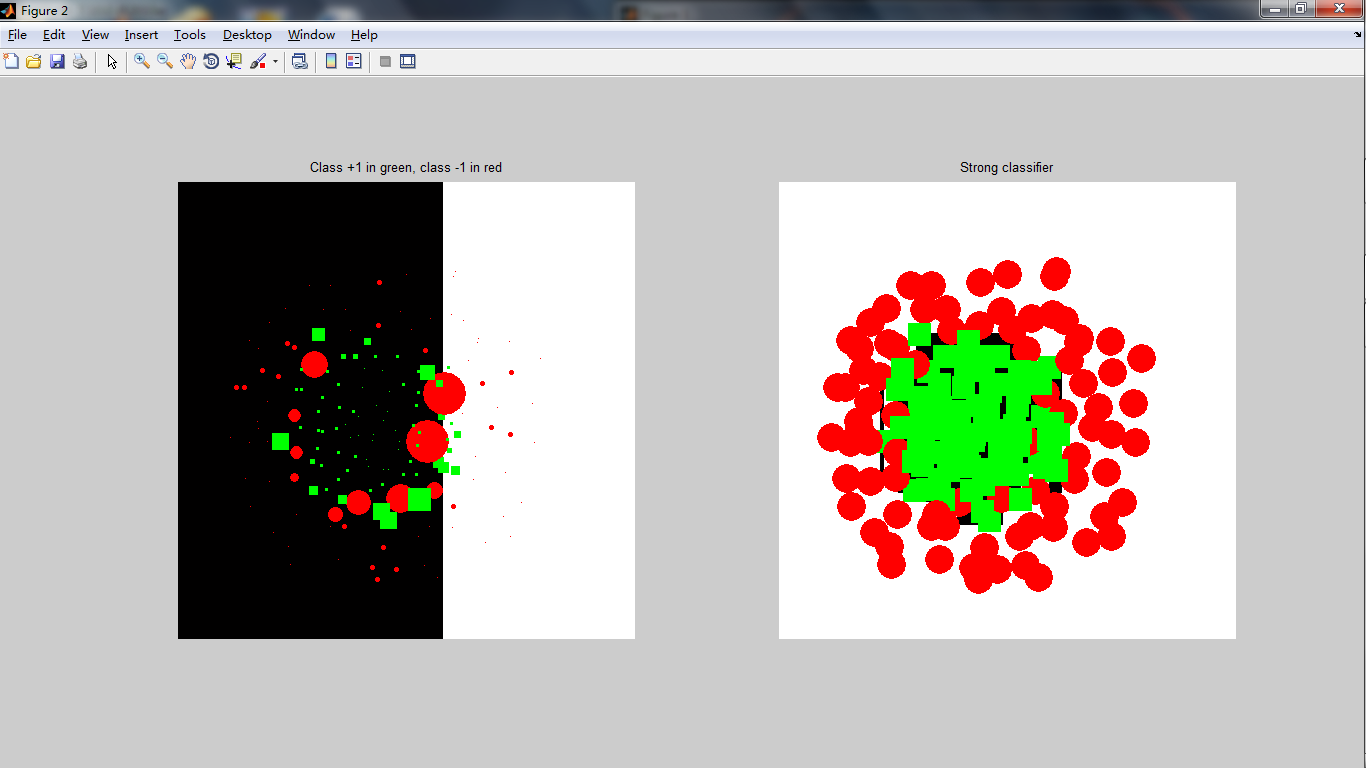
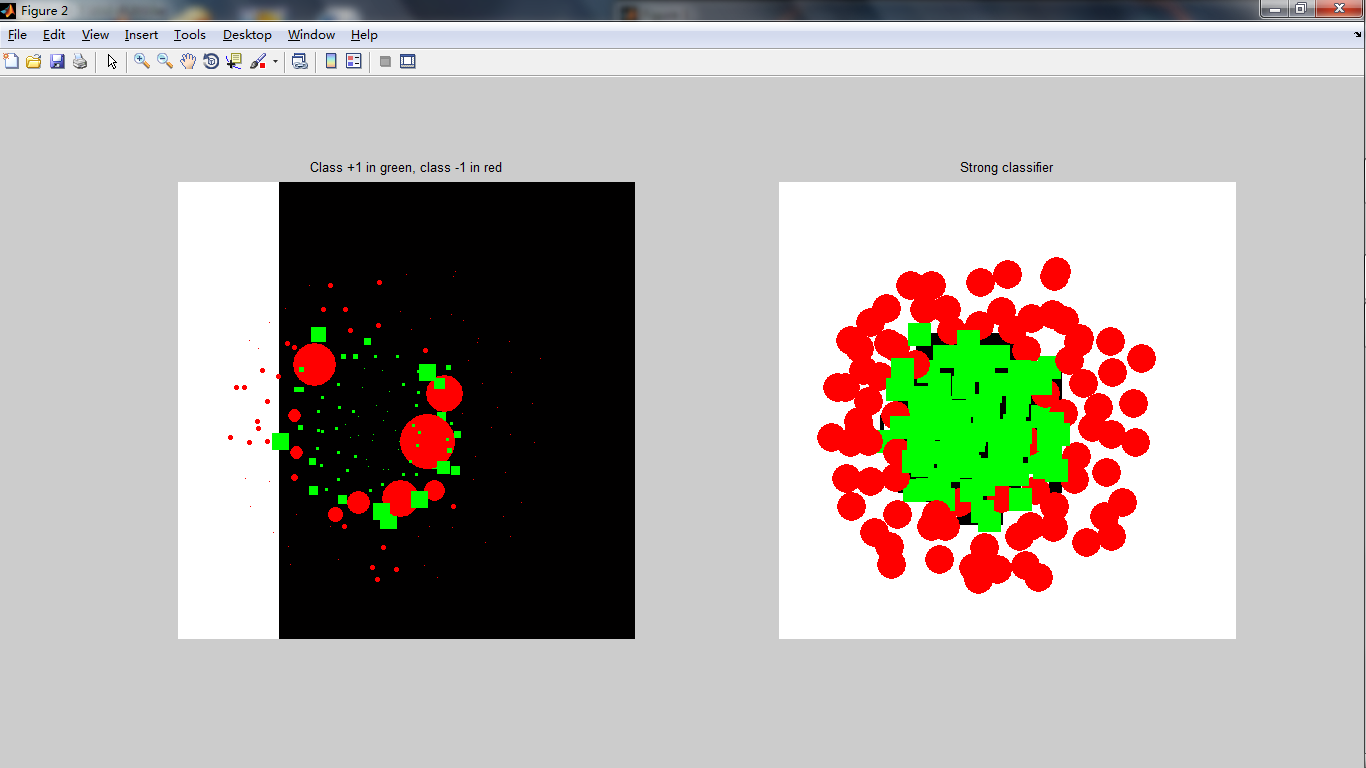


图 1.7

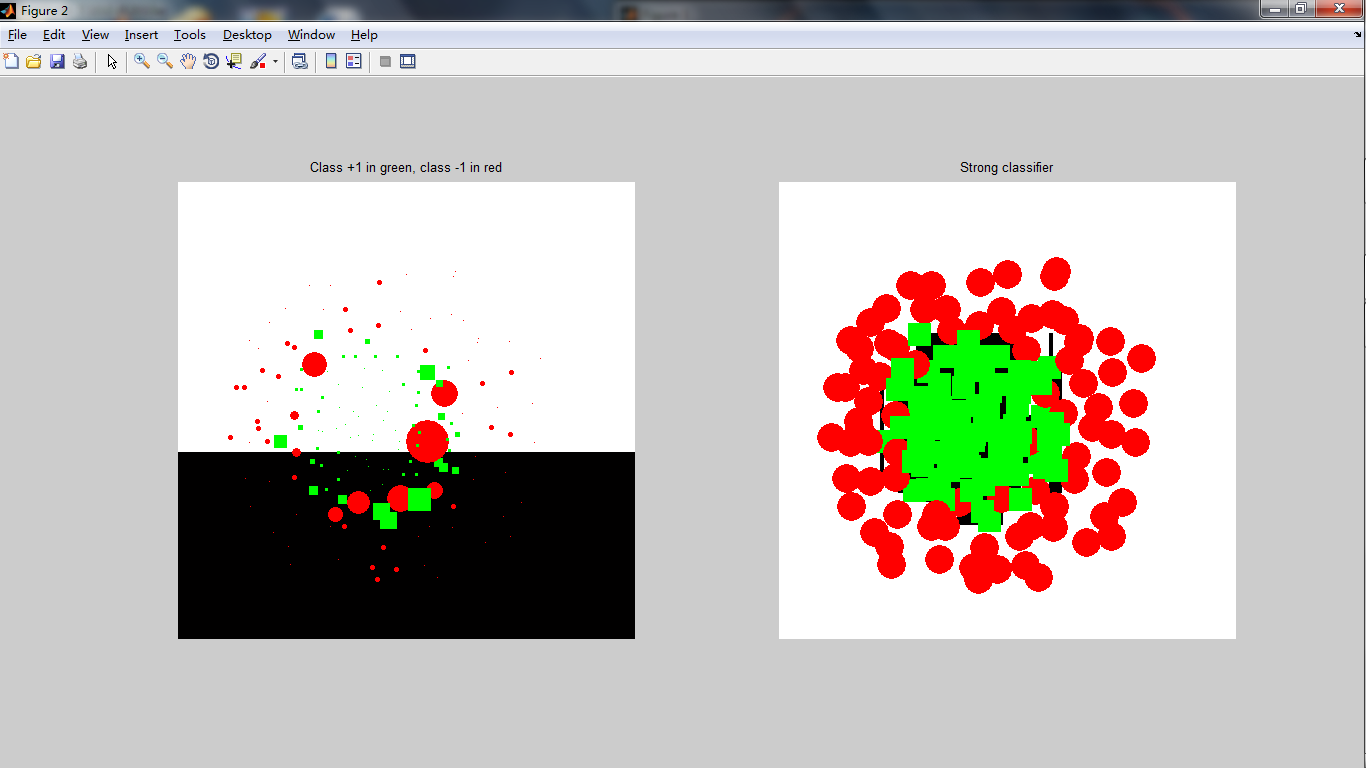


图 1.8

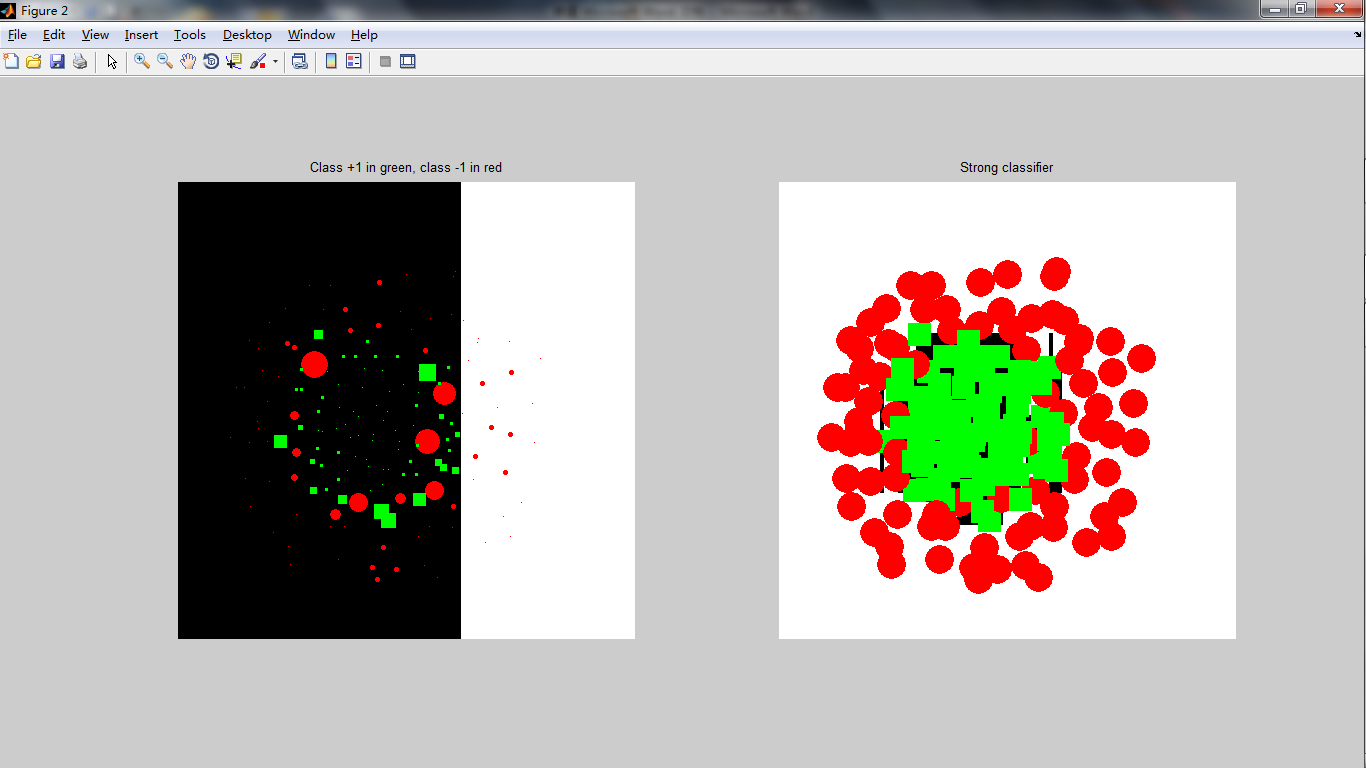
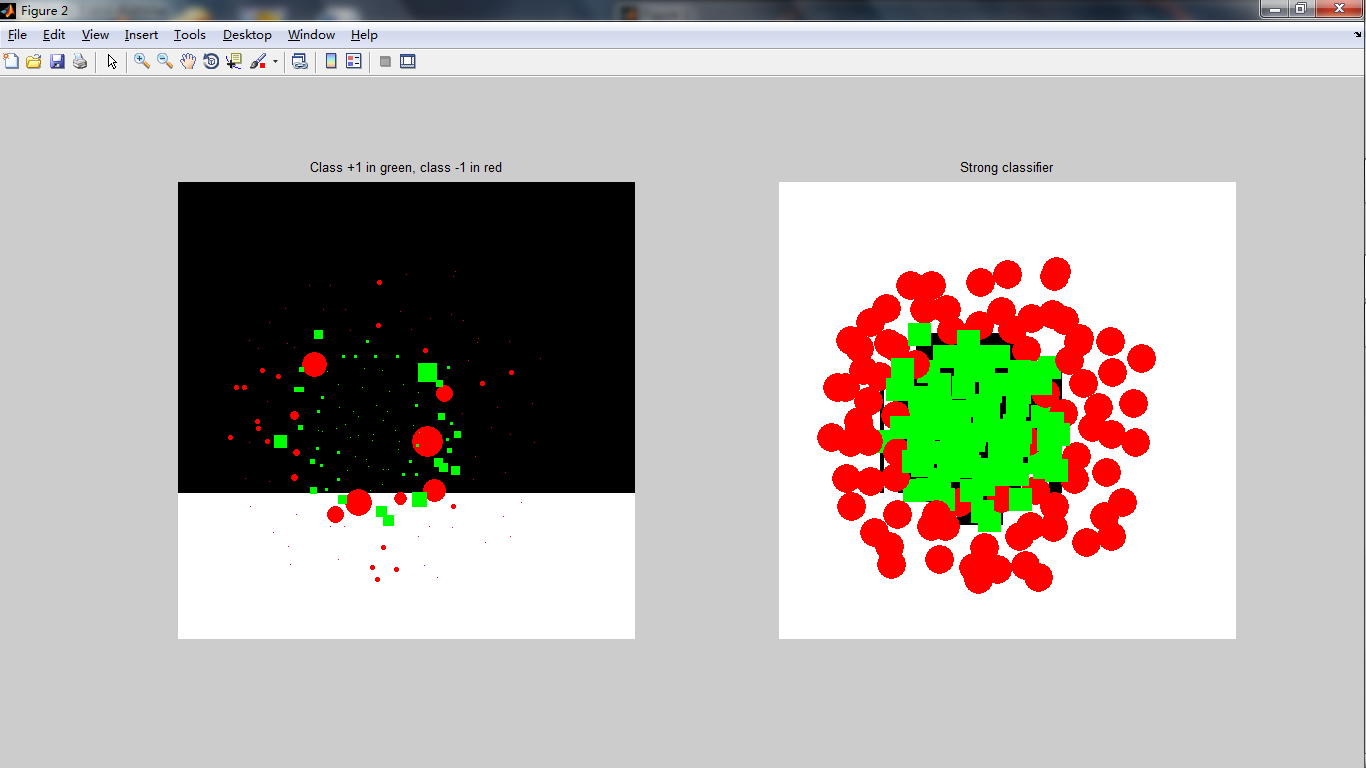


图 1.9

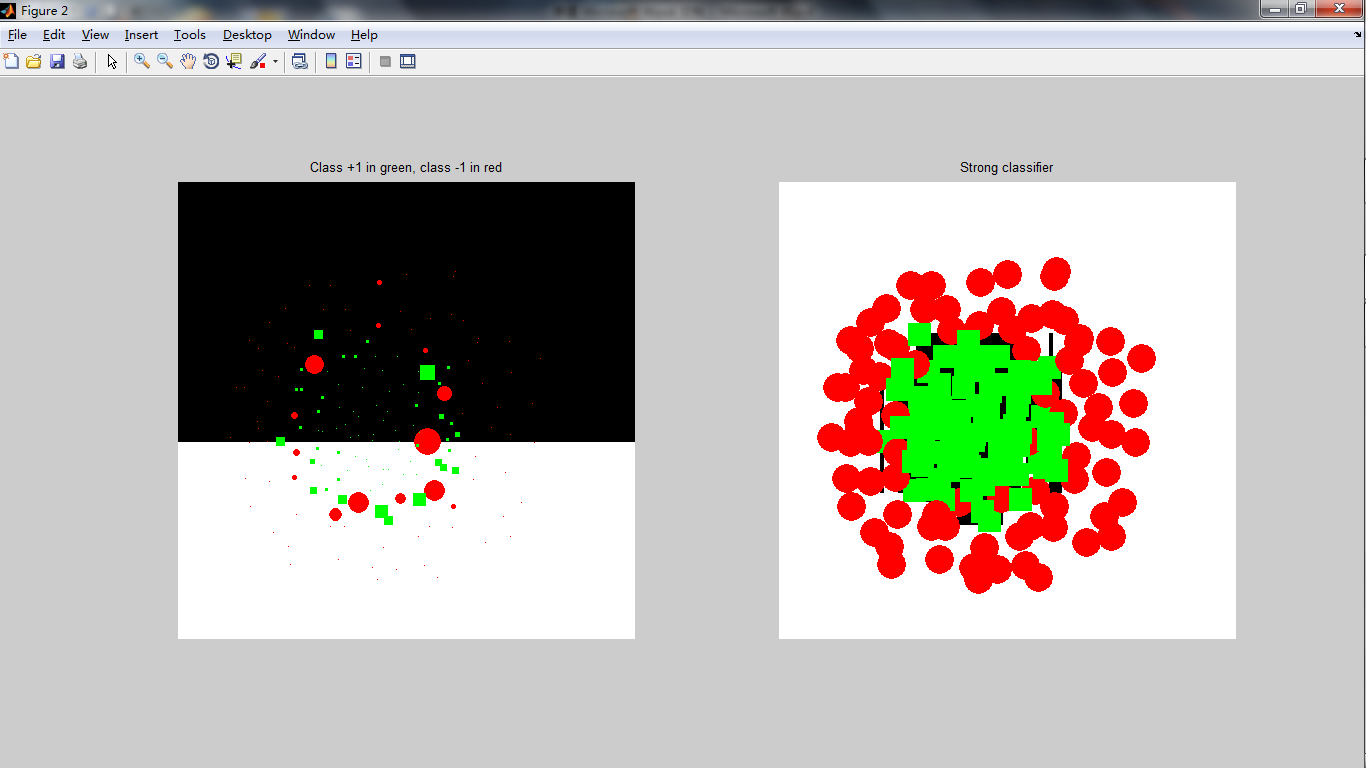


图 1.10

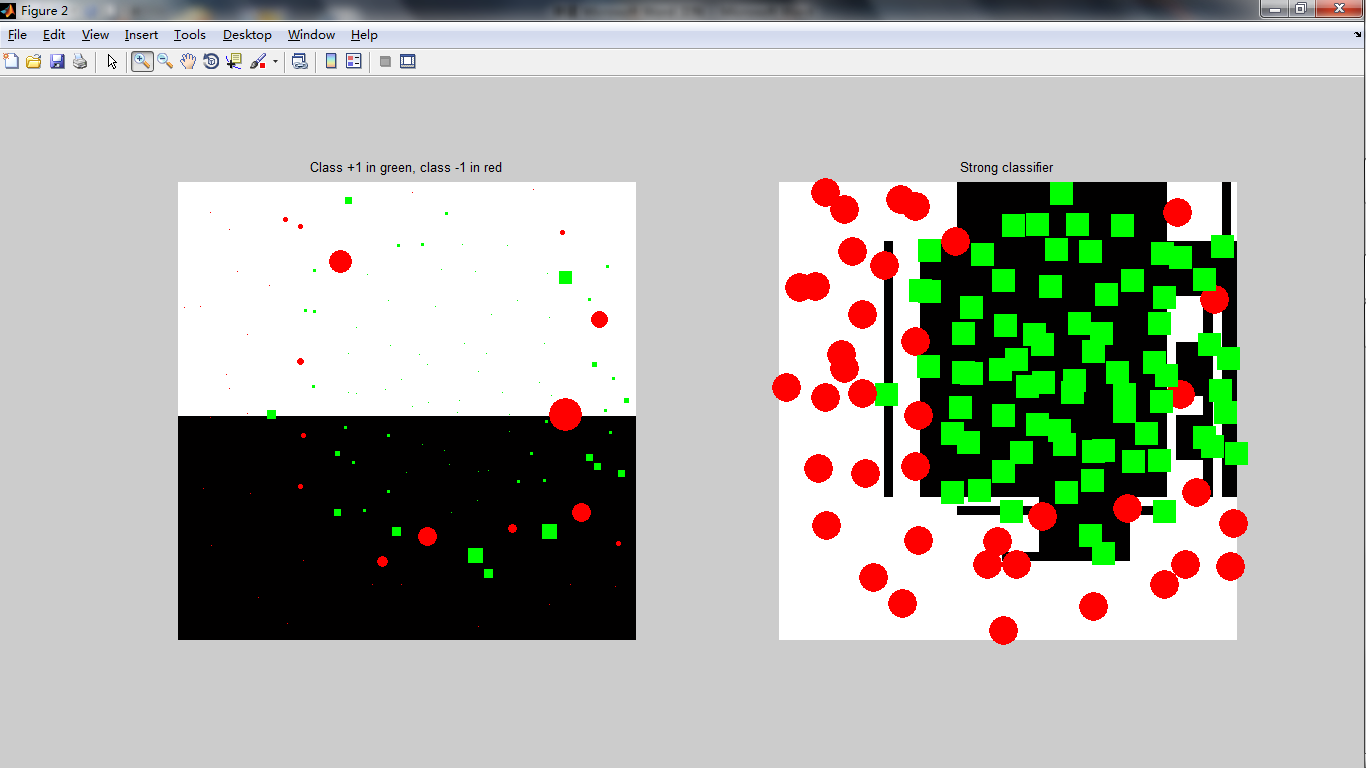
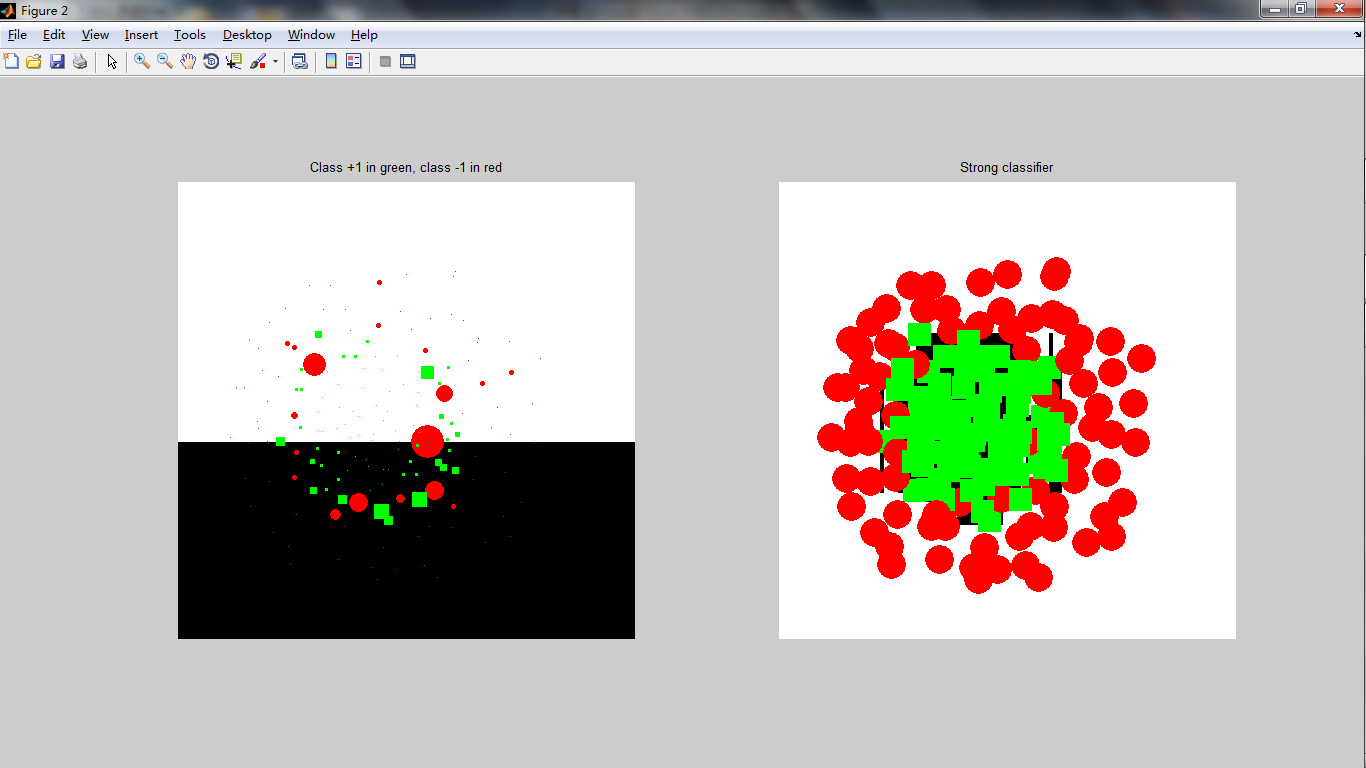


图 1.11

10） 运行createDatabase.m文件，显示识别好的图，有汽车和显示屏。如下图：

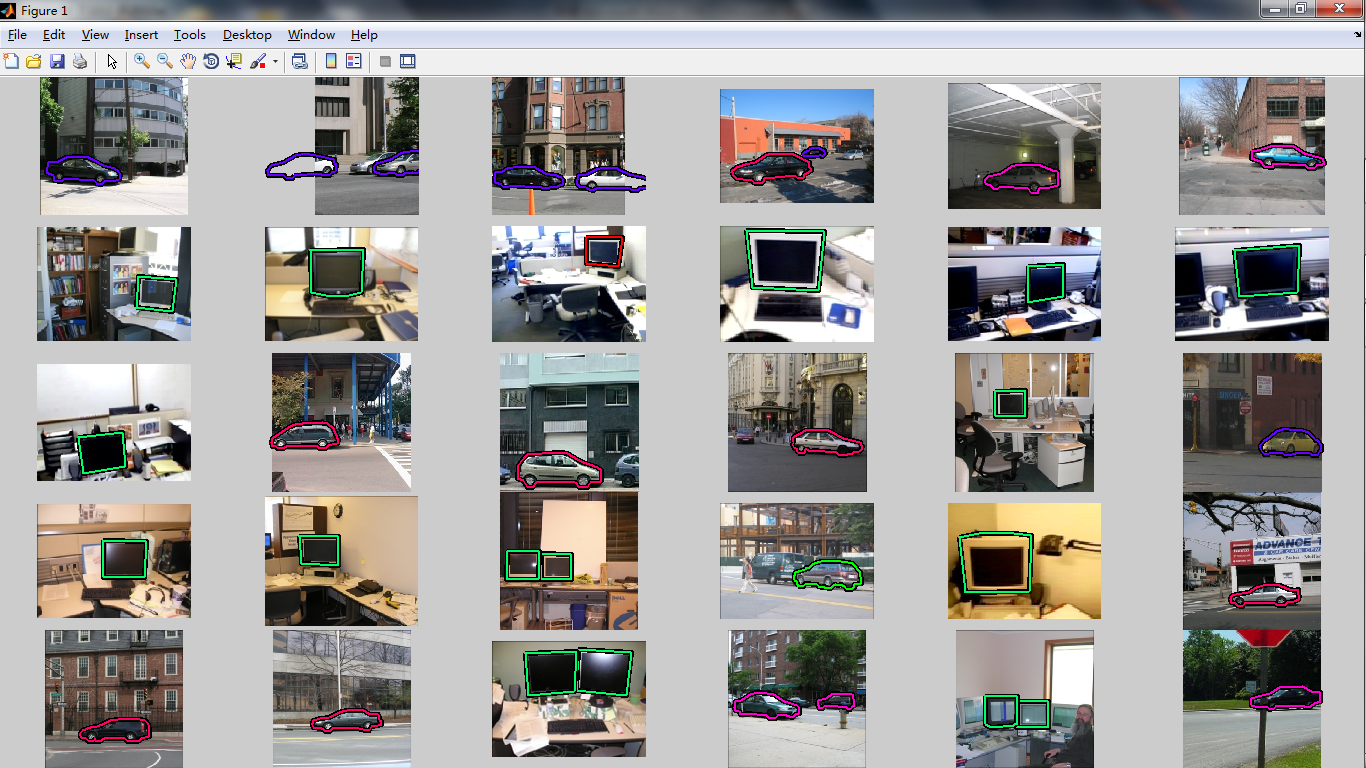
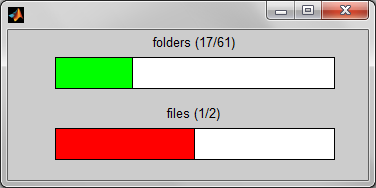


图 1.12

11） 运行createDictionary.m文件，为特征建立库。分离出每个物体。如下图：

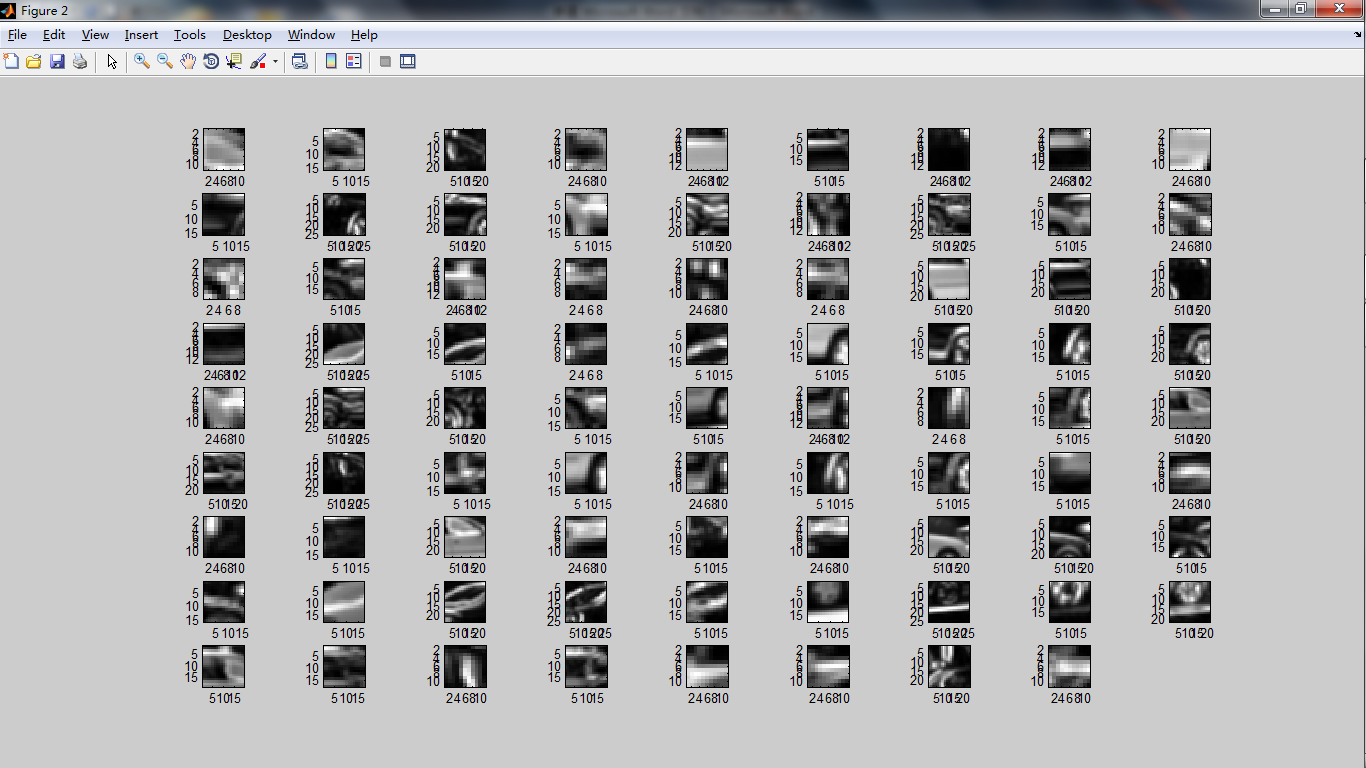
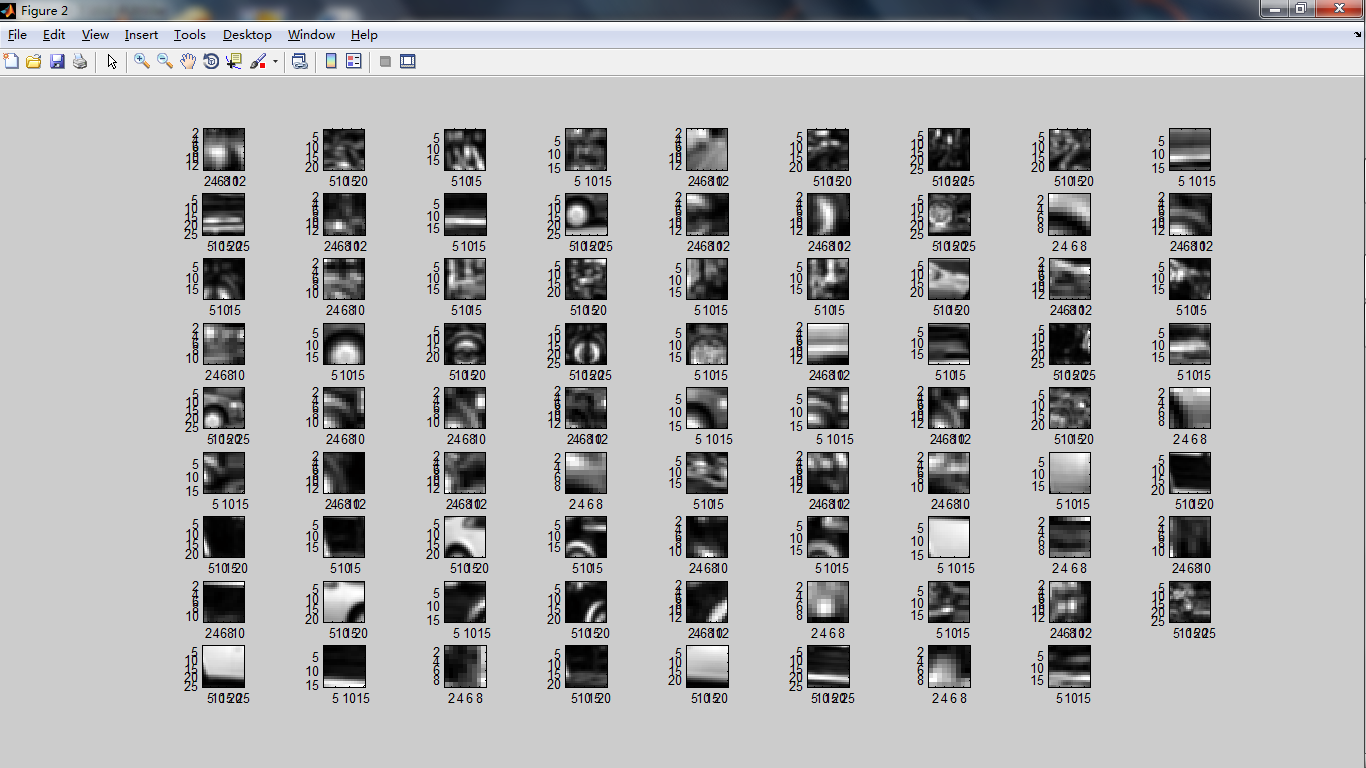


图 1.13

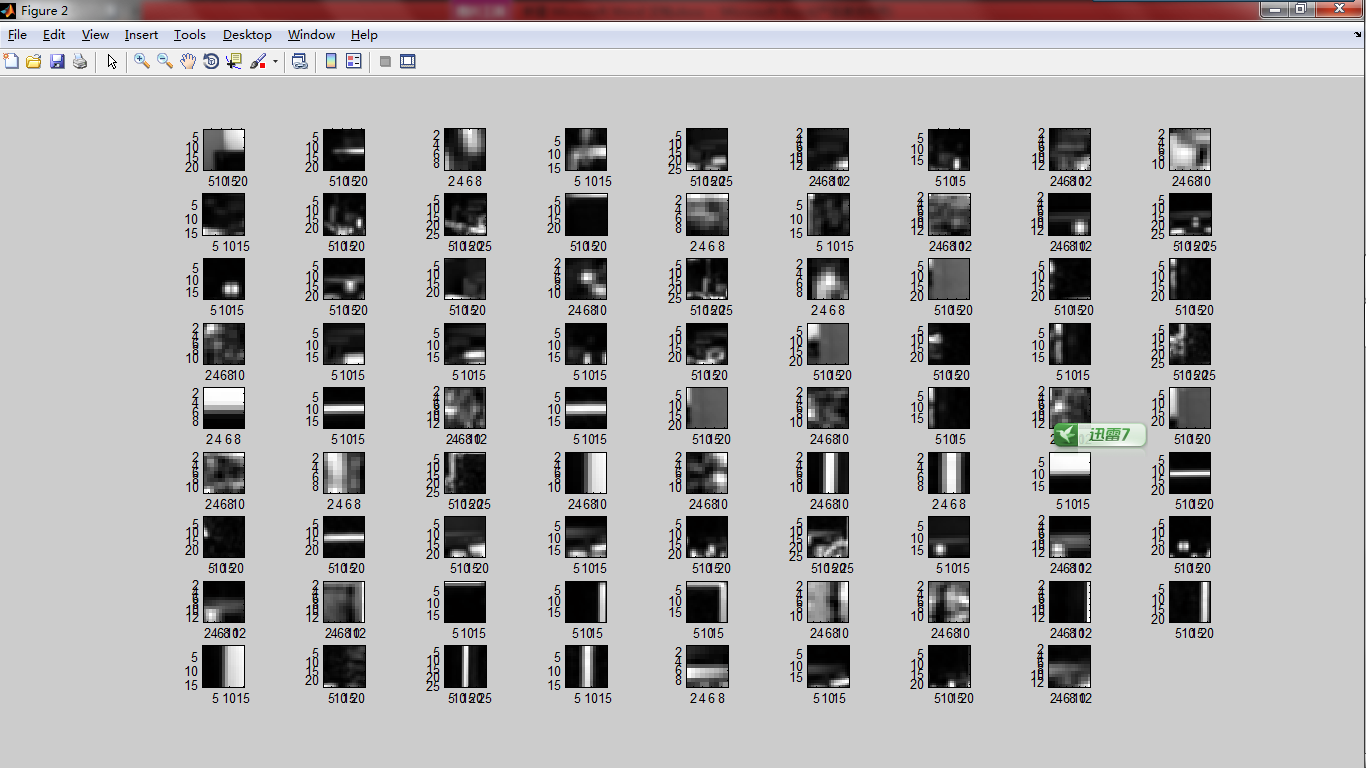
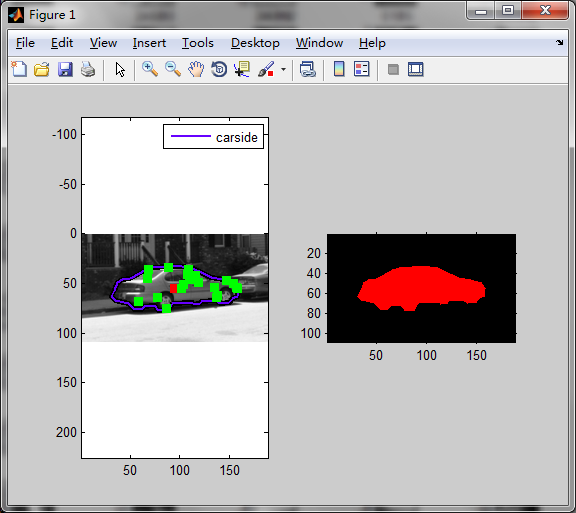


图 1.14

12） 运行computeFeatures.m文件，该程序计算样本集中每个图片的特征点，分离出物体。如下图：

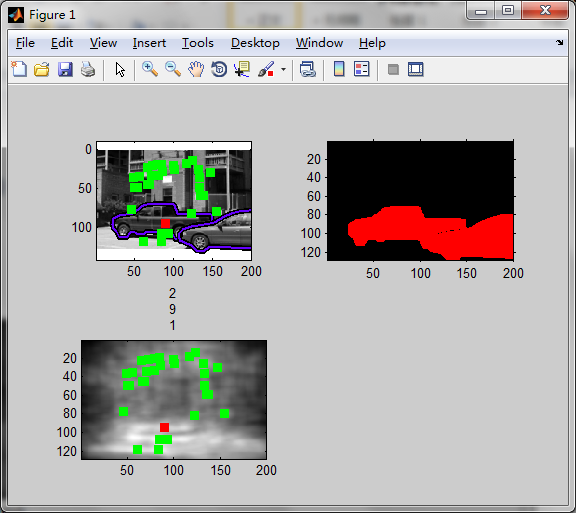
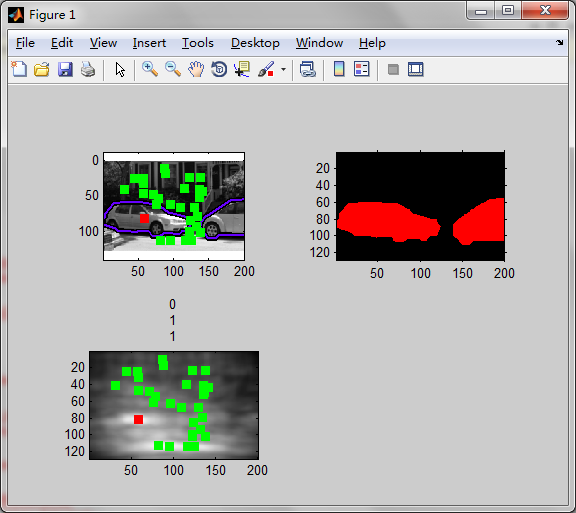


图 1.15

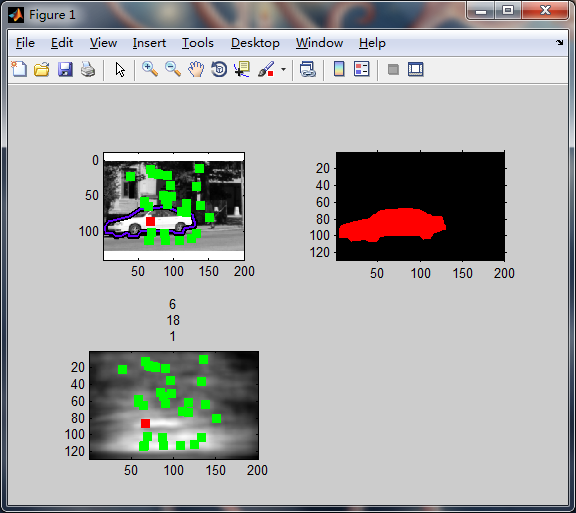
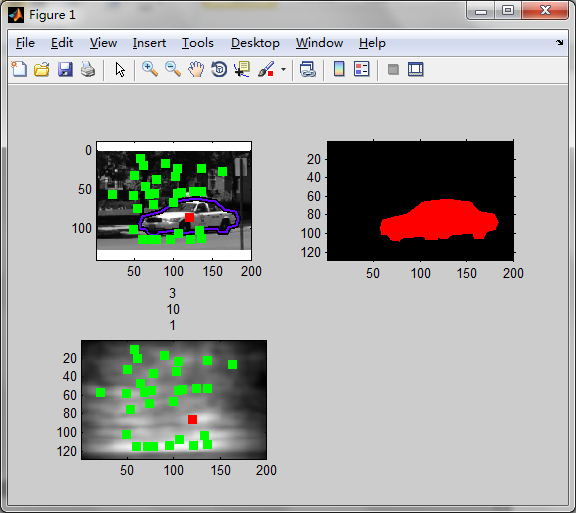


图 1.16

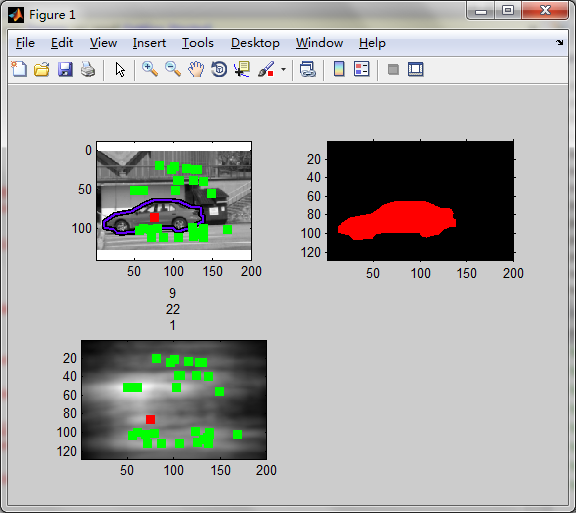
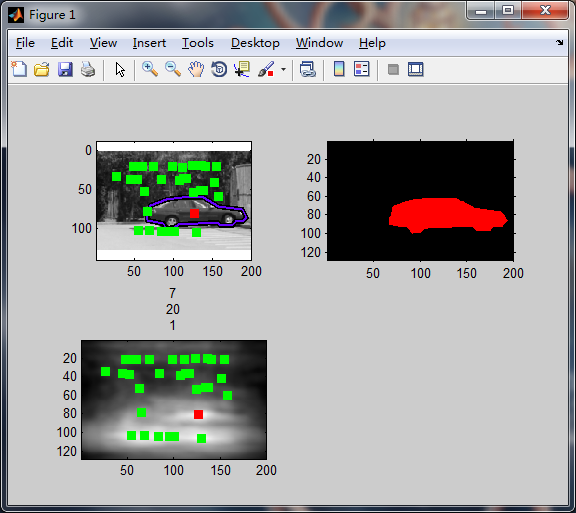


图 1.17

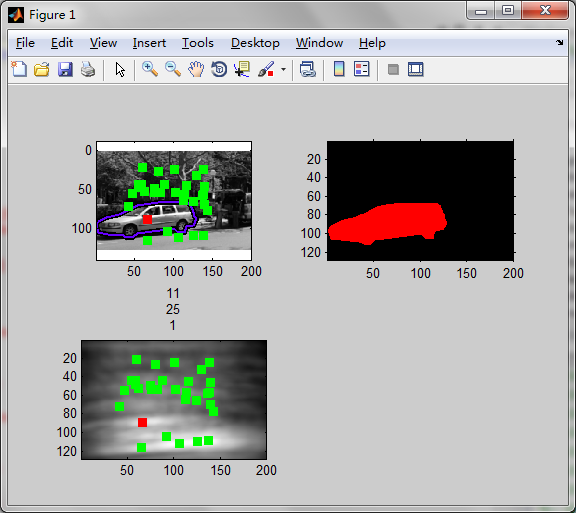
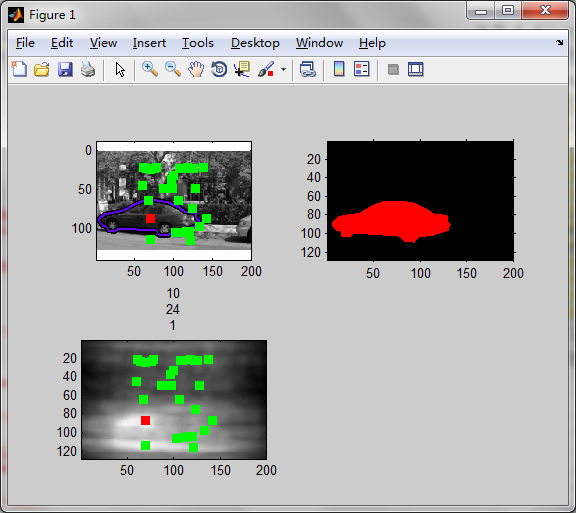


图 1.18

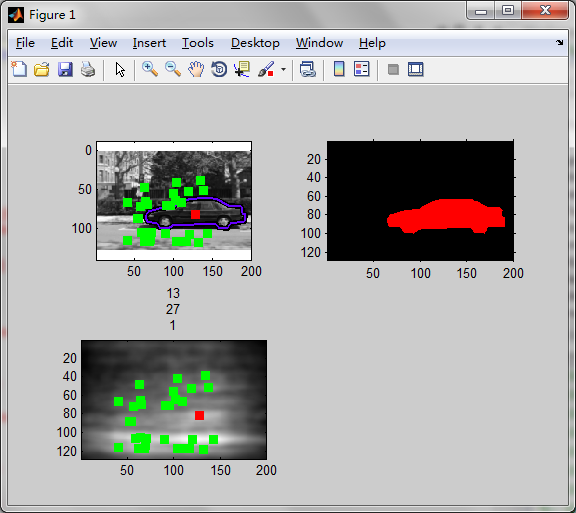
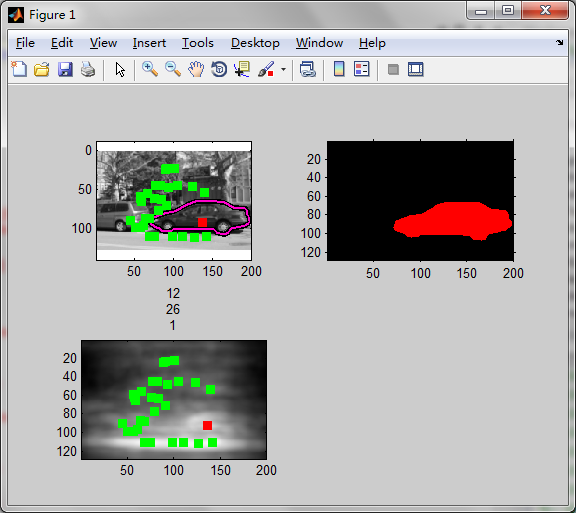


图 1.19

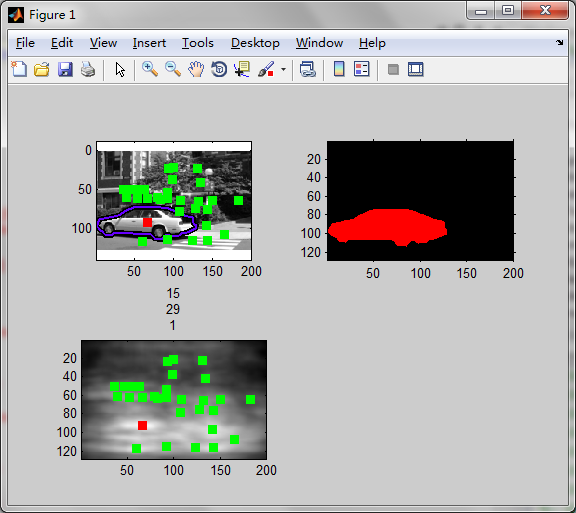
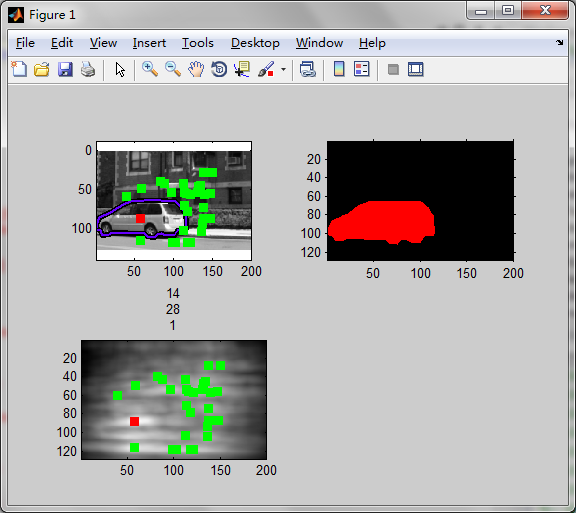


图 1.20

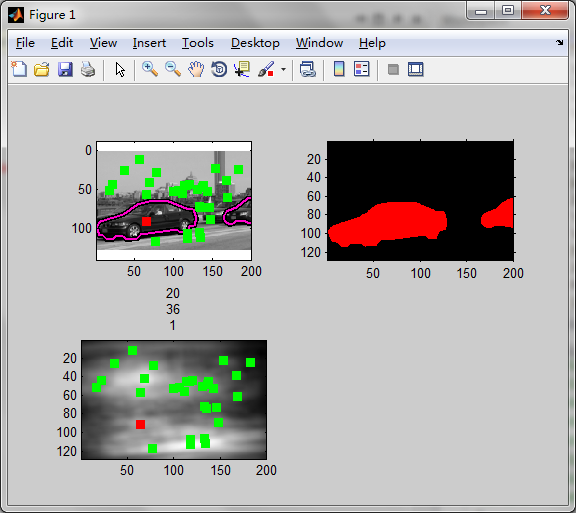
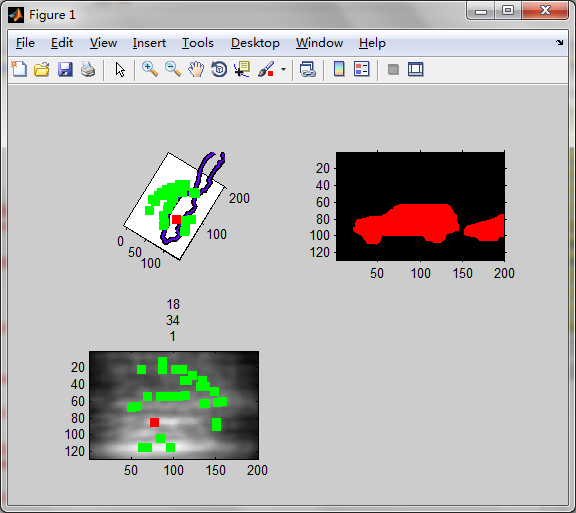


图 1.21

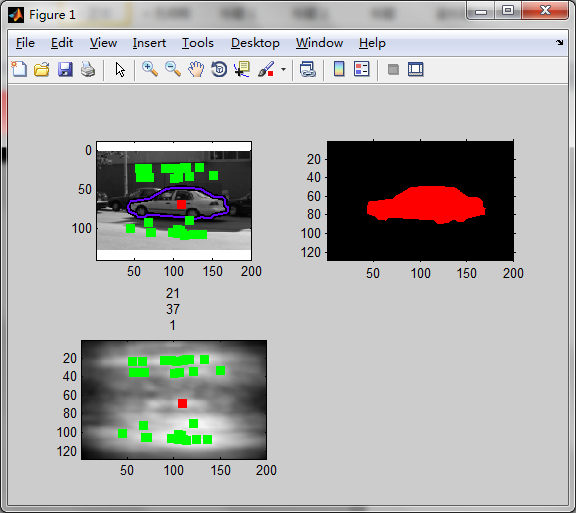
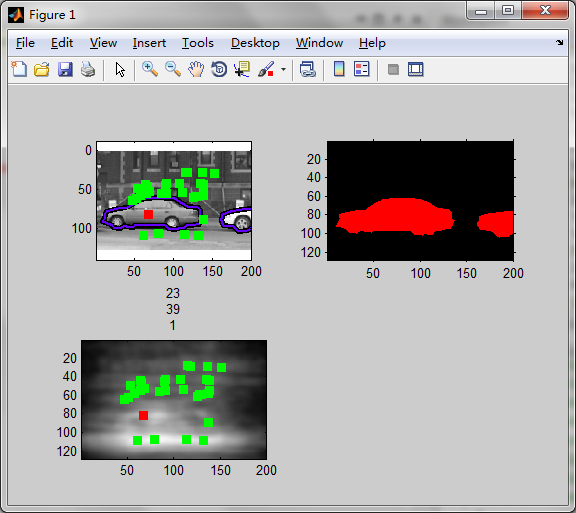


图 1.22

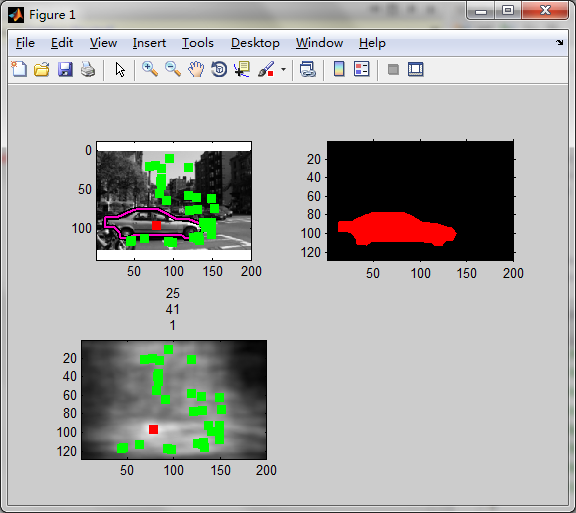
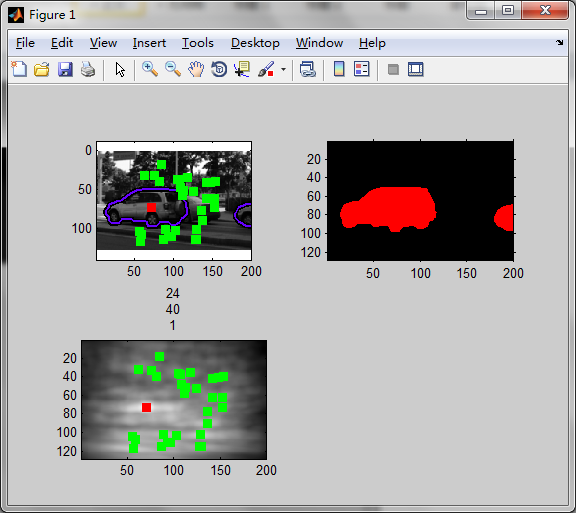


图 1.23

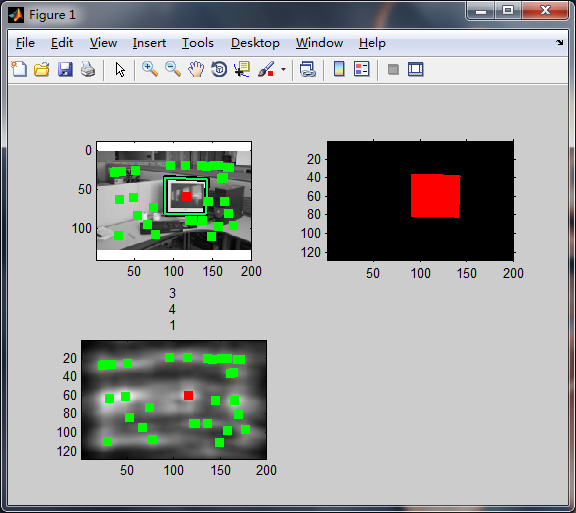
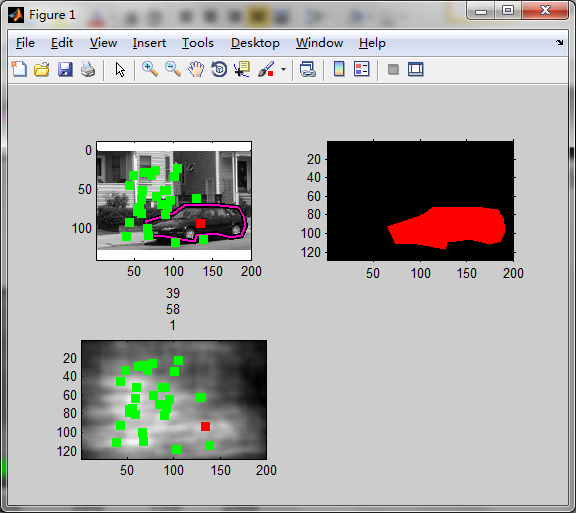


图 1.24

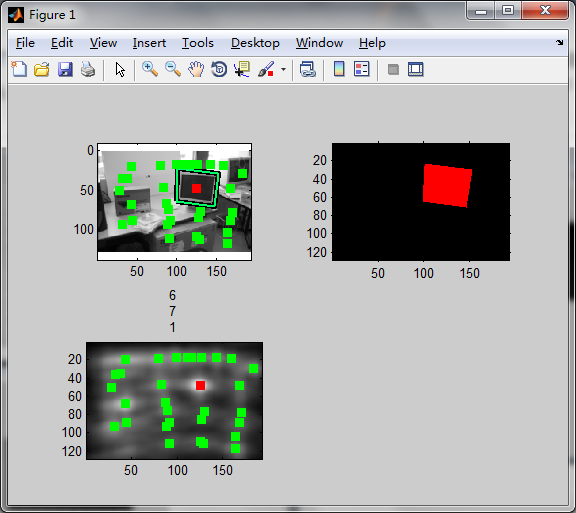


图 1.25

3. 关键代码举例说明：

computeFeature.m:

if min(X{1})>0 & min(Y{1})>0 & max(X{1})<ncols& max(Y{1})<nrows

imageUsed = 1;

通过限制检测图片边缘大小，检测物体是否在图片内，如果在图片内，使用该图片；如果不在该图片内则被使用，否则丢弃。

cx = round((min(X{1})+max(X{1}))/2);

cy = round((min(Y{1})+max(Y{1}))/2);

[y, x] = find(imregionalmax(score)); rr = sqrt((x-cx).^2 + (y-cy).^2);

[minr, jm] = min(rr);

ifminr< 9;

cx = x(jm);

cy = y(jm);

end

在物体中心存特征。

demoGentleBoost.m

[k, th, a , b] = selectBestRegressionStump(X, Y, w);

弱分类器聚集

fm = (a \* (X(k,:)>th) + b);

Fx = Fx + fm;

更新和计算训练集的分类器输出

fmT = (a \* (xt(k,:)>th) + b);

FxT = FxT + fmT;

更新和计算测试集的分类器输出

4. 实验结果分析：

实验结果更证实了GentleBoost的算法核心思想。

在gentleboostDemo.m中观察了弱分类器和强分类器的每一步迭代的变化，分错的点都会被增加权重从而继续分，到最后的决策分类器可以较好地分出红点和绿点。体现了GentleBoost根据每次训练集中每个样本分类正确与否以及上次的总体分类的准确率，来确定每个样本的权值。将修改过权值的新数据集送给下层弱分类器进行训练，最后将每次训练得到的分类器融合起来，作为最后的决策分类器。

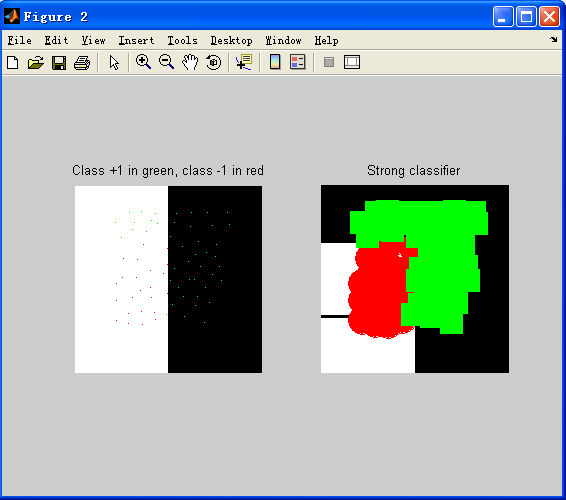
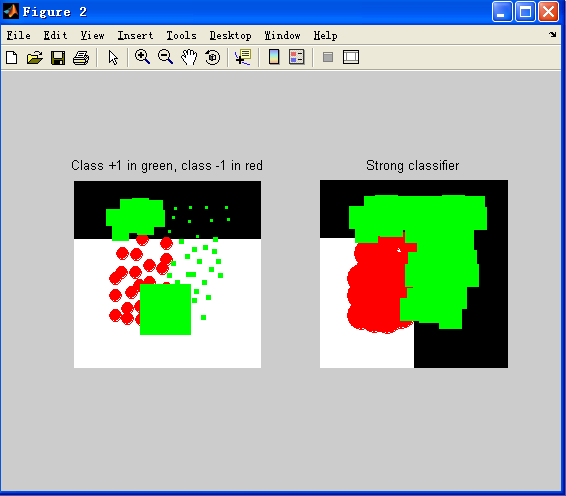
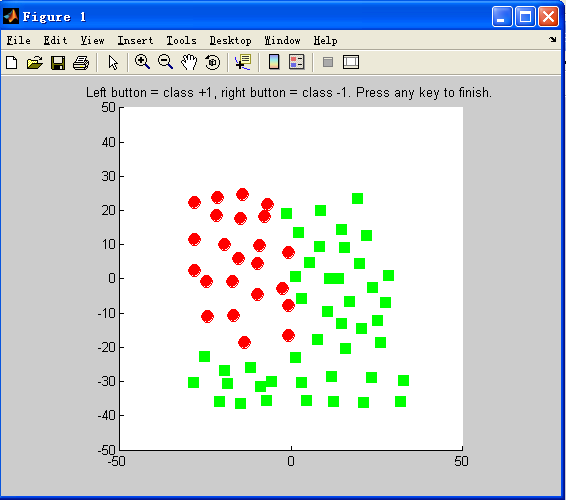
在rundetector.m中，观察到多次迭代，使检测物体的正确率逐步接近100%。这些实验结果都证实了GentleBoost算法的优越性。

（敦婧瑜）  
Gentle Boost

1. 对算法关键点的理解

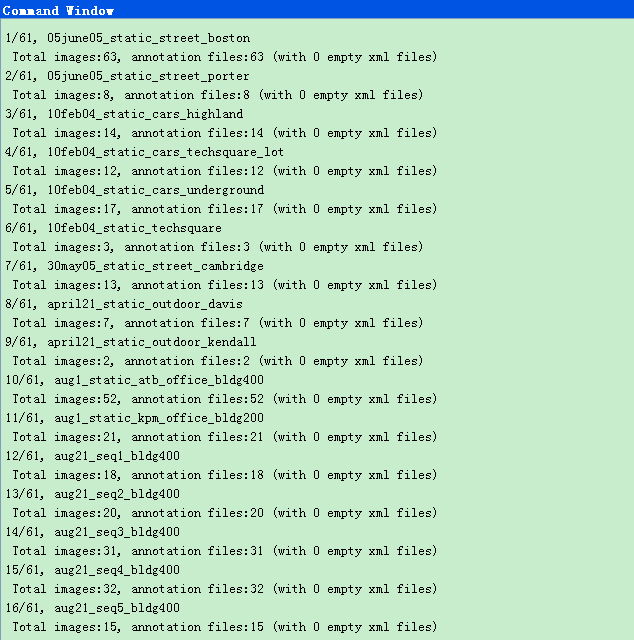
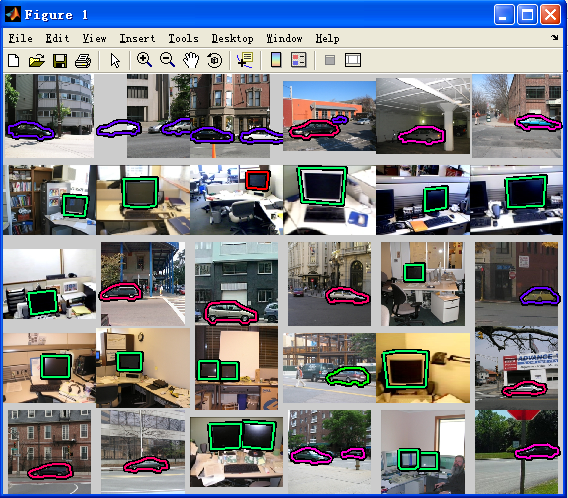
GentleBoost，是基于AdaBoost产生的变种算法.它和Adaboost不同之处是它较少的强调难以分类的样本。分错的样本的权重值会增大，分对的样本的样本的权重值相对下降。这样通过反复迭代，每次迭代都产生一个弱分类器。让所有符合错误率小于50%的弱分类器投票即可决定最后的强分类器。错误率越小的弱分类器，相应的投票比重就大。

1. 实验数据的详细描述
2. 执行demoGentleBoost.m进行一次小的demo测试



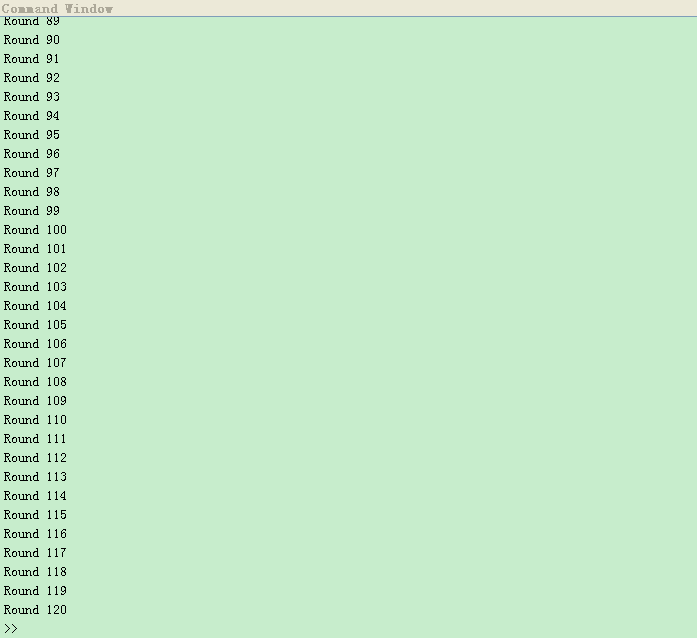
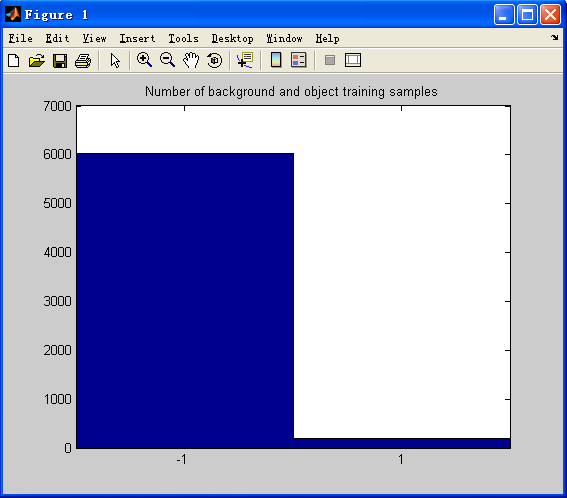
图表 1 自定义要分类的物体 图表 2 第2轮的强弱分类器情况 图表 3 第100轮的强弱分类器情况

1. 创建数据库



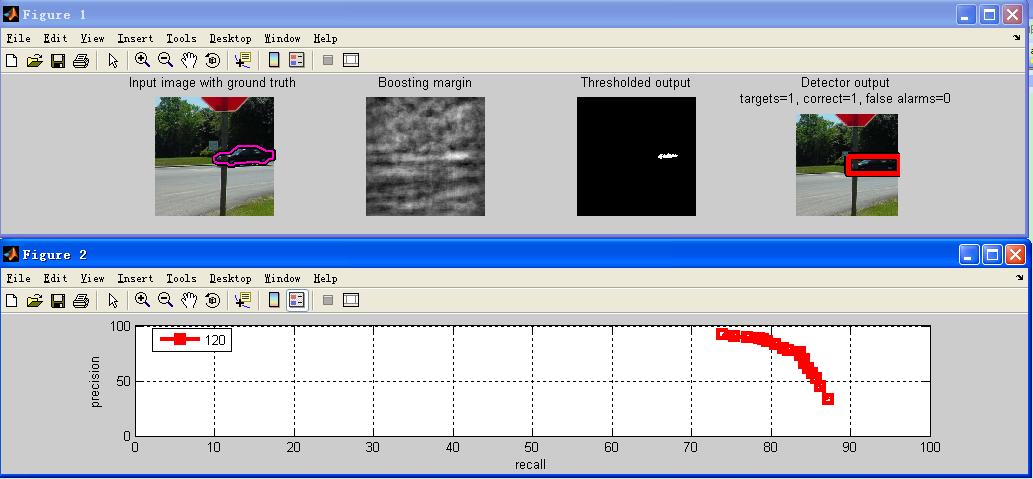
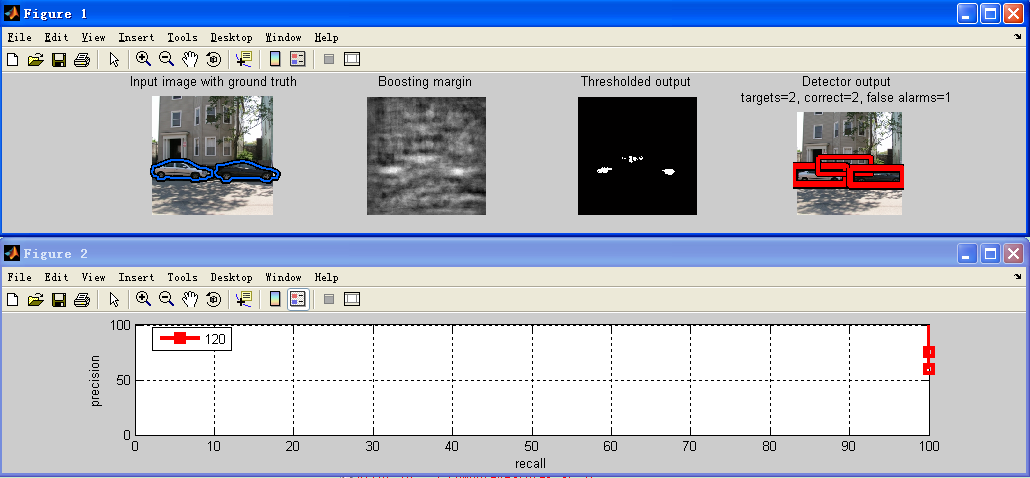
图表 4 数据库创建好后 图表 5 数据库创建时的命令窗口

1. 运行trainDetector，训练正负样本



图表 6 运行trainDetector的结果 图表 7 运行trainDetector的命令窗口

1. 运行runDetector（对车进行检测）



图表 8 运行runDetector的初始阶段 图表 9 运行runDetector得到的最后状态

1. 实验方法和流程

使用GentleBoost用于物体分类识别的方法和流程：

* 1. 首先进行初始设置。将initPath.m中的labelMe toolbox的路径改为LabelMeToolbox在电脑中解压后的位置，将Common functions的路径改为tools所在的目录位置，最后把boosting的路径改为gentleBoost所在的位置。
  2. 将parameter.m中的HOMEIMAGES和HOMEANNOTATIONS分别指向images和annotations所在的路径，即用这两个变量指明其位置，以方便后面的执行文件进行调用。
  3. 执行demoGentleBoost.m进行一次小的demo测试。这个demo可由自己设定两类物体的位置，然后会使用GentleBoost算法对两个物体进行识别分类。具体数据参见第2步的数据描述图。
  4. 为所有图片创建数据库。对图片中的相应物体进行标识，作为训练的对象和测试时的正样本。
  5. 运行trainDetector，训练正负样本。本次一共进行了120轮的训练。得到的训练结果参见第2步的数据描述图。
  6. 运行runDetector。在runDetector的代码中设置对汽车图片中的汽车进行检测。检测结果参见第2步的数据描述图。

1. 关键代码举例说明

gentleBoost.m中的关键代码：

for m = 1:Nrounds

disp(sprintf('Round %d', m))

% Weak regression stump: It is defined by four parameters (a,b,k,th)

% f\_m = a \* (x\_k > th) + b

[k, th, a , b] = selectBestRegressionStump(x, y, w);

% update parameters classifier

classifier(m).featureNdx = k;

classifier(m).th = th;

classifier(m).a = a;

classifier(m).b = b;

% Updating and computing classifier output on training samples

fm = (a \* (x(k,:)>th) + b); % evaluate weak classifier

Fx = Fx + fm; % update strong classifier

% Reweight training samples

w = w .\* exp(-y.\*fm);

% Plot results on a 2D space

if size(plotstyle,1)>0;

plotw(x(1,:), x(2,:), w, y, plotstyle);

pause;

end

end

说明：

以上代码即为gentleBoost算法的核心代码。从上述代码可以看到，总共会进行Nround次的迭代，若分类器由一个弱回归线函数来拟合。每轮先更新回归线的参数值，然后对弱分类器进行计算。新得到的弱分类器由Fx = Fx + fm 来对强分类器进行更新。然后重新计算各个样本点的权重（即增大分错的样本的权重，减小分对的样本的权重）。

1. 实验结果分析
2. 首先，对于demoGentleBoost.m的测试结果，我们在实验过程中观察到，由于两个物体比较容易被划分出来（见图表1），事实上到第40轮的时候，弱分类器的分类状况跟后面的几轮的分类状况已经没有什么区别了。实质上也就是说，这个分类过程只需要进行40次就可以得到满意的结果，不需要进行100轮。但是，如果换一个分布更加复杂的两个物体的混合场景，可能进行完100轮分类，都不一定能够得到一个令人十分满意的结果。
3. 其次对于testing的结果，我们是对有车的图片进行了检测。从执行过程中可以看到，boosting margin和threshold output中的高亮部分就是算法检测到的认为是目标物体的区域。而对于不同的阈值，所得到的precision-recall的图像也是不同的。

（王桢）

利用openCV 实现adaboost算法：

1. 对算法关键点的理解

Adaboost 是 adaptive boost 的缩写，它是一种算法，根据这个算法可以由许多不同种类的弱分类器得出一个错误率较低的强分类器。

它的基本过程如下：

1. 根据对训练集某个特征进行学习确定一个弱分类器。
2. 对于在上一个弱分类器被分错的元素加大他的权重，在此基础上进行第二次学习，根据相同的或者不同的特征对训练集进行第二次分类，从而确定第二个弱分类器。
3. 不断重复上述过程，直到达到某个条件，如错误率低于某个值，或者已达到设定的迭代次数则停止迭代。
4. 最终根据弱分类器不同的错误率分配权重，最终的强分类器就是这些弱分类器的组合，一个元素到底属于哪一类，就由这些弱分类器投票决定。

用相应的数学公式表示adaboost的基本过程[1]：



输入：一组训练集： ，其中 为样本描述， 为样本标识， ；



其中0，1分别表示正例子和反例。初始化：初始化训练样本的权重，D(i),其中 或者  （正例样本和反例样本初始化为不同的值，m为反例总数，l为正例样本总数)。

对 t= 1,2,3……T，循环执行下面的步骤：



1. 归一化权重：
2. 对每个特征f，训练一个弱分器 ；计算所有特征的弱分类器的加权错误率



3)选取最佳弱分类器 ，按最小错误率。



4)按照这个最佳弱分类器，调整权重：

 ， 

其中 表示被正确分类，表示被错误分类。

5) 最后的强分类器为：



其中：



2实验数据详细描述：

正样本源是含有汽车的256\*256的图片：



图10

负样本源图片：



图11

训练后生成的正样本为：

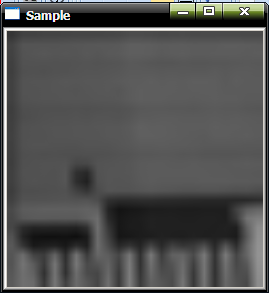
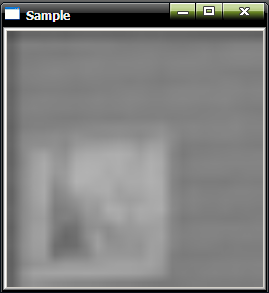
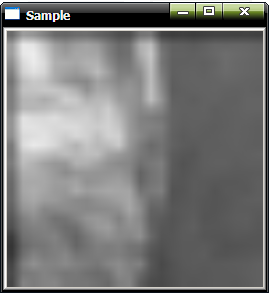


图12

3实验方法和流程:

1. 目标检测分为三个步骤
2. 样本的创建
3. 训练分类器
4. 利用训练好的分类器进行目标检测。

2）样本创建  
       负样本由背景描述文件来描述。背景描述文件是一个文本文件，每一行包含了一个负样本图片的文件名（基于描述文件的相对路径）。该文件必须手工创建。  
     本次实验背景描述文件bg.txt的内容为：  
 Nsample/comproom.jpg

Nsample/frontroom.jpg

Nsample/groginoffice.jpg

Nsample/flexyplan-factors.jpg

Nsample/grotto.jpg

Nsample/mvc-001f.jpg

Nsample/ordinateur.jpg

正样本  
  正样本由程序opencv\_createsamplesd.exe程序来创建。

  创建样本命令为：

 cd E:\zuoye\opencv\bin

opencv\_createsamplesd.exe -vec E:\目前作业\视觉\Cvadaboost\a.vec -info E:\目前作业\视觉\Cvadaboost\info.dat -bg E:\目前作业\视觉\Cvadaboost\bg.txt -num 11 -show -w 256 -h 256

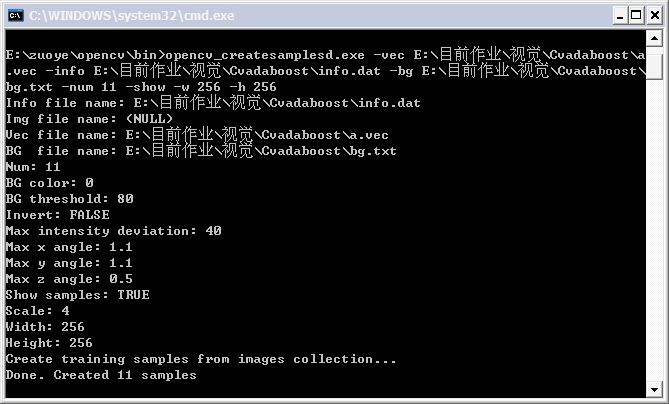


图13 训练正样本命令

生成a.vec文件在E:\目前作业\视觉\Cvadaboost目录下  
3）训练分类器  
    样本创建之后，接下来要训练分类器，这个过程是由opencv\_haartrainingd.exe程序来实现的。

命令为：

cd E:\zuoye\opencv\bin

opencv\_haartrainingd.exe -data E:\目前作业\视觉\Cvadaboost\data

-vec E:\目前作业\视觉\Cvadaboost\a.vec

-bg E:\目前作业\视觉\Cvadaboost\bg.txt

-npos 5

-nneg 2

-w 256

-h 256

在训练分类器的过程中出现了内存溢出的错误，由于没有完全弄清楚openCV的内部机制，没有找到出错的原因，实验失败.

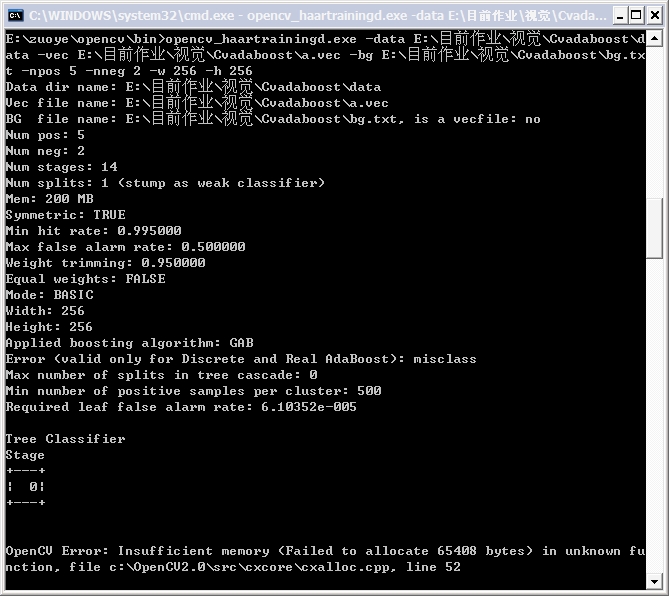


图14 错误代码框 图16 训练分类器命令

**小组信息：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 姓名 | 学号 | 班级 | 手机 | 电子邮箱 |
| 敦婧瑜 | 200830633427 | 08（2）班 | 13560161421 | 943314434@qq.com |
| 郑允芯 | 200830635346 | 08（2）班 | 15017506061 | 316748611@qq.com |
| 王桢（组长） | 200830635100 | 08（2）班 | 13560170948 | 774801861@qq.com |