The results you need to get in ex2

ex2

必做部分

第一部分 打印数据点：

（1）先导入数据：

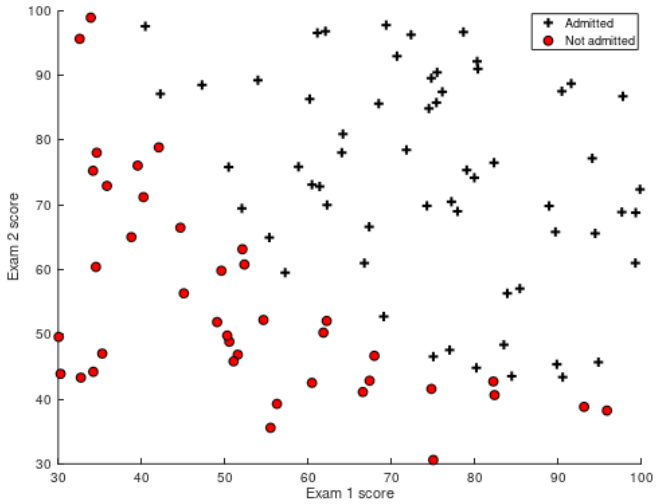
```
data = load('ex2data1.txt');
X = data(:, [1, 2]); y = data(:, 3);
```

（2）这里需要理解PlotData.m文件中的Octave的Plot函数：

```
pos=find(y==1);
neg=find(y==0);
plot(X(pos,1),X(pos,2),'k+','LineWidth',2,'MarkerSize',7);
plot(X(neg,1),X(neg,2),'ko','MarkerFaceColor','r','MarkerSize',7);
```

（3）要实现下图的效果还需要ex2.m文件中的

```
hold on;
% 加标签
xlabel('Exam 1 score')
ylabel('Exam 2 score')
%加图例
legend('Admitted', 'Not admitted')
hold off;
```



第二部分 ：在sigmoid.m文件中实现Sigmoid函数

由于不仅需要能够进行离散数值计算，还需要能够进行向量、矩阵的运算，因此涉及到 "."这个可以进行元素遍历运算的符号

```
g = 1./(1+exp(-z));
```

测试运算结果：

```
>> a=[1 2 3 4 5 6];
>> sigmoid(a)
ans =

   0.73106   0.88080   0.95257   0.98201   0.99331   0.99753
```

第一次提交作业：

```
>> submit
== Submitting solutions | Logistic Regression...
Login (email address): 2455254643@qq.com
Token: UVLlk0M7oghfUutB
==
==                                  Part Name |     Score | Feedback
==                                  --------- |     ----- | --------
==                            Sigmoid Function |   5 /   5 | Nice work!
==                  Logistic Regression Cost |   0 /  30 |
==               Logistic Regression Gradient |   0 /  30 |
==                                   Predict |   0 /   5 |
==       Regularized Logistic Regression Cost |   0 /  15 |
==   Regularized Logistic Regression Gradient |   0 /  15 |
==                                  ---------------------------------
==                                           |   5 / 100 |
==
```

第三部分：实现损失函数和逻辑回归的梯度下降

%中间变量t

t=sigmoid(X*theta);

%损失函数，其值必须为数值而非向量

J=(-y'*log(t)-(1-y)'*log(1-t))/m;

%梯度向量，左侧需要以θ的个数为函数的向量

grad=(X'*(t-y))/m;

测试结果：

```
Cost at initial theta (zeros): 0.693147
Expected cost (approx): 0.693
Gradient at initial theta (zeros):
 -0.100000
 -12.009217
 -11.262842
Expected gradients (approx):
 -0.1000
 -12.0092
 -11.2628

Cost at test theta: 0.218330
Expected cost (approx): 0.218
Gradient at test theta:
 0.042903
 2.566234
 2.646797
Expected gradients (approx):
 0.043
 2.566
 2.647
```

第四部分：使用fminunc高级优化方法来获取参数

fminunc拟牛顿法求多元函数极值点，无需再写迭代和寻找学习速率，只需要损失计算和梯度计算

ex2.m文件中已经执行到:

% 为fminunc设置选项，梯度输出以及最大迭代次数

options = optimset('GradObj', 'on', 'MaxIter', 400);

% 运行fminunc以获取最小损失时的最优θ，initial_theta为n+1维零向量

[theta, cost] = ...

fminunc(@(t)(costFunction(t, X, y)), initial_theta, options);

```
Cost at theta found by fminunc: 0.203498
Expected cost (approx): 0.203
theta:
 -25.161272
 0.206233
 0.201470
Expected theta (approx):
 -25.161
 0.206
 0.201
```

第五部分：查阅plotDecisionBoundary.m文件，理解决策边界线是如何生成的

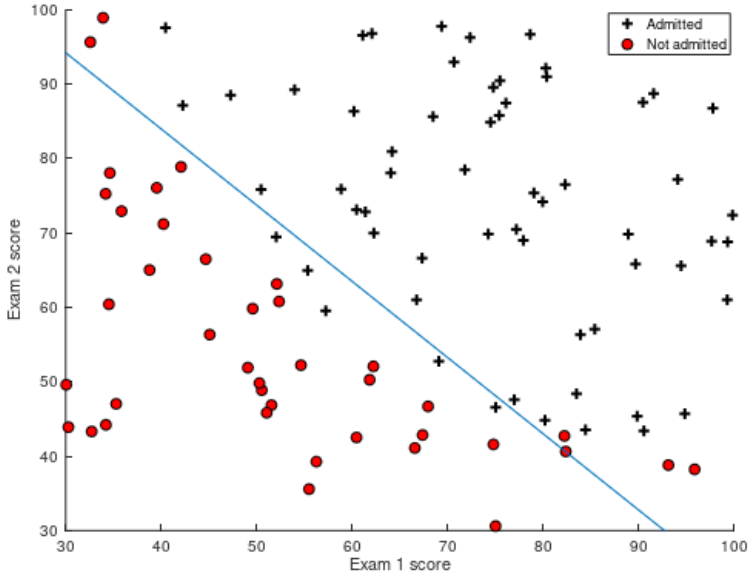
ex2.m文件中实现：

```
% 为决策边界线绘图
plotDecisionBoundary(theta, X, y);
hold on;
% 标签
xlabel('Exam 1 score')
ylabel('Exam 2 score')
% 说明
legend('Admitted', 'Not admitted')
hold off;
```

hold on能够实现图像重叠绘制

plotDecisionBoundary.m文件实现：

```
%绘制数据点
plotData(X(:,2:3), y);
hold on
if size(X, 2) <= 3
   %只需首末两点即可定义一条直线
   plot_x = [min(X(:,2))-2,  max(X(:,2))+2];
   %计算决策边界线上的点
   plot_y = (-1./theta(3)).*(theta(2).*plot_x + theta(1));
   % 绘图，设置轴线
   plot(plot_x, plot_y)
   %说明
   legend('Admitted', 'Not admitted', 'Decision Boundary')
   axis([30, 100, 30, 100])
else...
```



第六部分：使用逻辑回归进行评估（根据成绩判定是否录取学生）

以当前的最优θ值预测某个学生被录取的概率：

```
prob = sigmoid([1 45 85] * theta);
fprintf(['For a student with scores 45 and 85, we predict an admission ' ...
         'probability of %f\n'], prob);
fprintf('Expected value: 0.775 +/- 0.002\n\n');
```

```
For a student with scores 45 and 85, we predict an admission probability of 0.776289
Expected value: 0.775 +/- 0.002
```

完善predict.m文件代码：

```
p=(sigmoid(X*theta)>=0.5);
```

最终ex2.m生成正确率，预期为89%：

```
fprintf('Train Accuracy: %f\n', mean(double(p == y)) * 100);
```

```
Train Accuracy: 89.000000
Expected accuracy (approx): 89.0
```

此时提交：

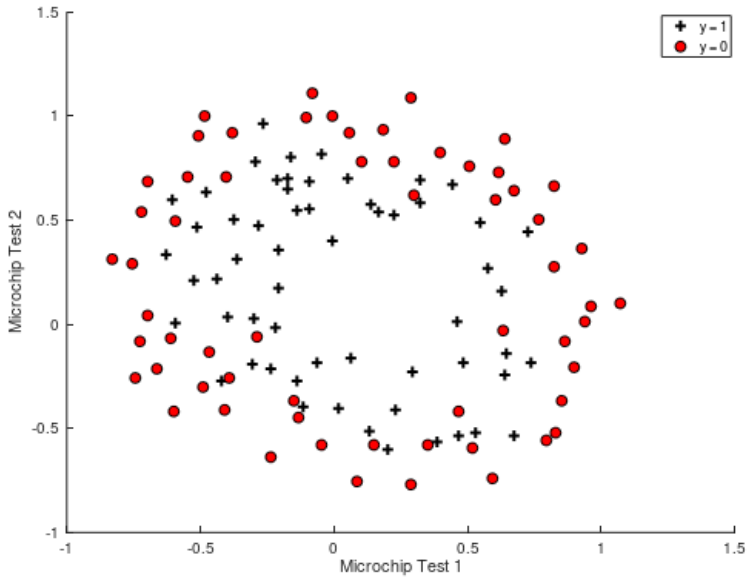
```
==
==                                Part Name |     Score | Feedback
==                                --------- |     ----- | --------
==                          Sigmoid Function |   5 /   5 | Nice work!
==                  Logistic Regression Cost |  30 /  30 | Nice work!
==              Logistic Regression Gradient |  30 /  30 | Nice work!
==                                   Predict |   5 /   5 | Nice work!
==       Regularized Logistic Regression Cost |   0 /  15 |
==   Regularized Logistic Regression Gradient |   0 /  15 |
==                                  ---------------------------------
==                                           |  70 / 100 |
```

第七部分：正则化逻辑回归（预测设备的微芯片能否通过质检）

运行ex2_reg.m文件进行后面的练习

（一）数据可视化

同样运行PlotData.m文件，可以看到样本数据点无法使用直线进行分割



（二）特征映射（即将样本数据进行自扩充）

加入更多特征可以更好地对数据进行拟合，使用mapFeature.m文件，θ特征向量增长至28维，决策边界线将会是更复杂的、非线性的。虽然拥有更好的分类能力，但是过拟合的风险也更大，为降低过拟合需要使用正则化

%输入所有样本的第一次测试成绩和第二次测试成绩

```
function out = mapFeature(X1, X2)

degree = 6;
out = ones(size(X1(:,1)));
for i = 1:degree
   for j = 0:i
      out(:, end+1) = (X1.^(i-j)).*(X2.^j);
   end
end
```

（三）正则化逻辑回归的损失函数与梯度

完成costFunctionReg.m的代码，注意θ0不进行正则化，因此Octave中的theta(1）不进行正则化：

```
t=sigmoid(X*theta);
J=-(y'*log(t)+(1-y)'*log(1-t))/m+lambda/(2*m)*sum(theta(2:end).^2);
theta(1)=0;
grad=1/m*X'*(t-y)+lambda/m*theta;
```

```
Cost at initial theta (zeros): 0.693147
Expected cost (approx): 0.693
Gradient at initial theta (zeros) - first five values only:
 0.008475
 0.018788
 0.000078
 0.050345
 0.011501
Expected gradients (approx) - first five values only:
 0.0085
 0.0188
 0.0001
 0.0503
 0.0115

Program paused. Press enter to continue.

Cost at test theta (with lambda = 10): 3.164509
Expected cost (approx): 3.16
Gradient at test theta - first five values only:
 0.346045
 0.161352
 0.194796
 0.226863
 0.092186
Expected gradients (approx) - first five values only:
 0.3460
 0.1614
 0.1948
 0.2269
 0.0922
```

（四）使用fminunc进行梯度下降获取最优的特征参数θ

计算损失函数

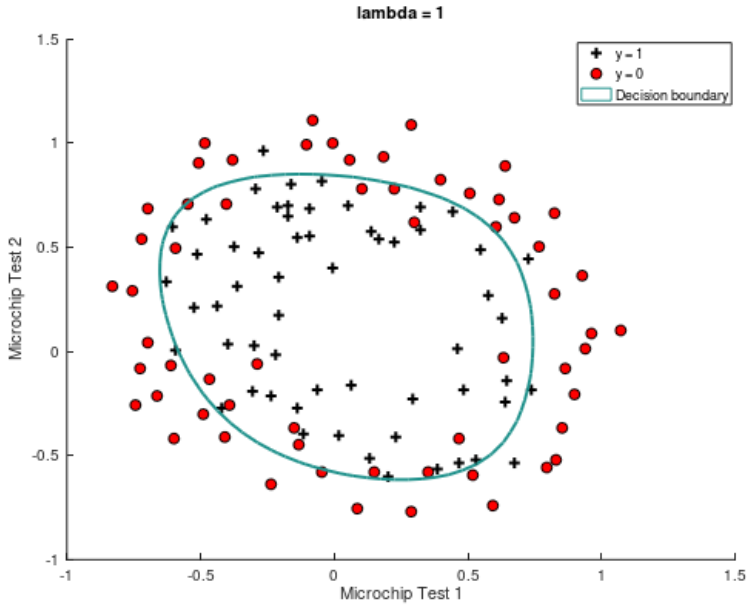
```
initial_theta = zeros(size(X, 2), 1);
%设置λ为1
lambda = 1;
% 设置fminunc梯度下降的选项
options = optimset('GradObj', 'on', 'MaxIter', 400);
% 进行优化
[theta, J, exit_flag] = ...
 fminunc(@(t)(costFunctionReg(t, X, y, lambda)), initial_theta, options);
```

（五）绘制决策边界线

由于特征已经超过三个因此使用ployDecisionBoundary.m的else部分：

```
else
    % Here is the grid range
    u = linspace(-1, 1.5, 50);
    v = linspace(-1, 1.5, 50);
    z = zeros(length(u), length(v));
    % Evaluate z = theta*x over the grid
    for i = 1:length(u)
        for j = 1:length(v)
            z(i,j) = mapFeature(u(i), v(j))*theta;
        end
    end
```

```
    z = z'; % important to transpose z before calling contour
    % Plot z = 0
    % Notice you need to specify the range [0, 0]
    contour(u, v, z, [0, 0], 'LineWidth', 2)
end
hold off
```



（六）计算判断正确率

```
p = predict(theta, X);
fprintf('Train Accuracy: %f\n', mean(double(p == y)) * 100);
```

```
Train Accuracy: 83.050847
Expected accuracy (with lambda = 1): 83.1 (approx)
```

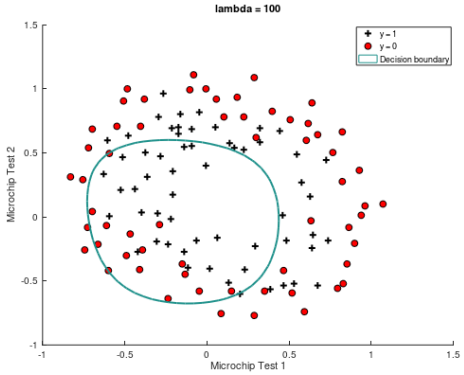
此时提交：

```
==
==                                Part Name |     Score | Feedback
==                                --------- |     ----- | --------
==                          Sigmoid Function |   5 /   5 | Nice work!
==                  Logistic Regression Cost |  30 /  30 | Nice work!
==              Logistic Regression Gradient |  30 /  30 | Nice work!
==                                   Predict |   5 /   5 | Nice work!
==       Regularized Logistic Regression Cost |  15 /  15 | Nice work!
==   Regularized Logistic Regression Gradient |  15 /  15 | Nice work!
==                                  ---------------------------------
==                                           | 100 / 100 |
```

选做部分

尝试对数据集使用不同的正则化参数λ来理解正则化如何削弱过拟合（修改ex2_reg.m文件中的lambda值即可观察不同图像）

设置过大（如100）可能导致欠拟合：



设置过小（如0）可能导致过拟合：