

模式识别与机器学习

实验报告（一）

班级：计科1505

姓名：陶婉莹

学号：0902150521

指导教师：梁毅雄

**Part I**

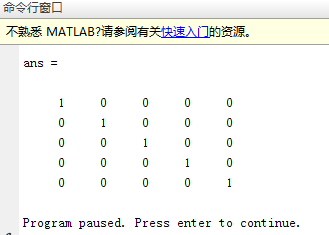
**Linear Regression**

**1 Simple octave function**

完成 warmUpExercise.m ，使得 A 为 5x5 单位矩阵。

补充代码：A = eye(5);

结果：



1. **Linear regression with one variable**

通过以前的数据预测新开的店的收益。ex1data1.txt 提供了所需要的训练组，第一列是城市人口，第二列是对应的收益，为负值则代表亏损。

**Plotting the Data**

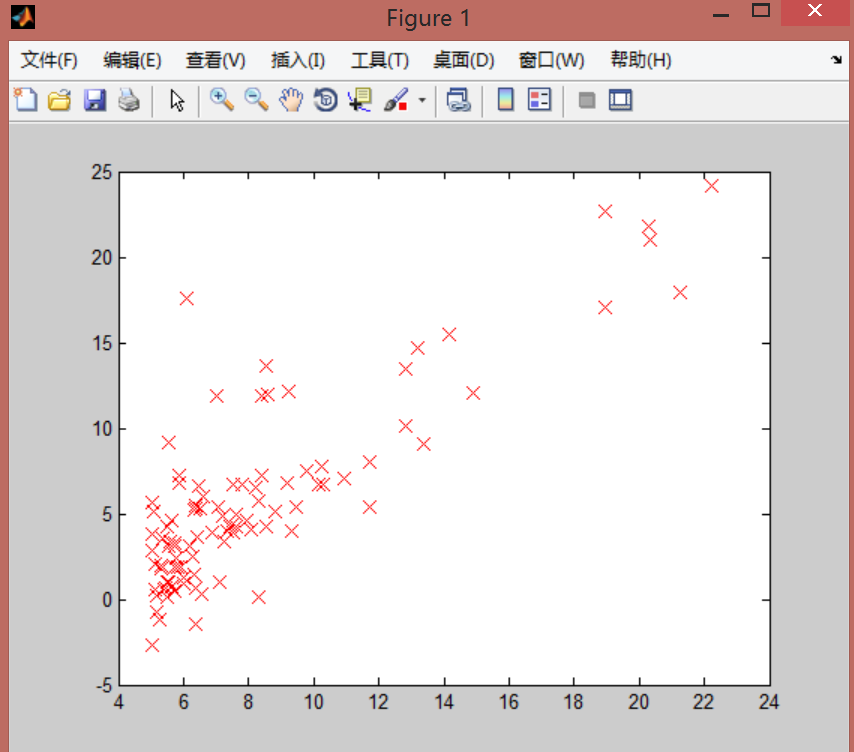
在exm1.m中已经加载了X，y，通过plotData函数将散点图描绘出来。

补充代码：plot(x, y, 'rx', 'MarkerSize', 10); % Plot the data

ylabel('Profit in $10,000s'); % Set the y-axis label

xlabel('Population of City in 10,000s'); % Set the x-axis label

结果：

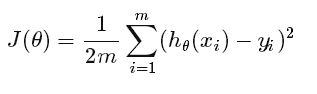


1. **Gradient Descent**

通过梯度下降法计算出合适的线性回归参数。

1. **Update Equations**

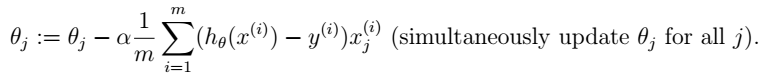
线性回归的目标是将成本函数最小化：



其中线性模型的假设  为：



其中变量是 ，通过调整  的值来最小化成本函数。一种方式就是梯度下降法，每次迭代都会更新 ：



使用梯度下降法时必须在矩阵 X 第一列增加一列 1，因为成本函数的公式中 的系数为1。

1. **Implementation**

初始化参数为0，学习率为0.01，设置迭代次数为1500次。

X = [ones(m, 1), data(:,1)]; % Add a column of ones to x

theta = zeros(2, 1); % initialize fitting parameters

% Some gradient descent settings

iterations = 1500;

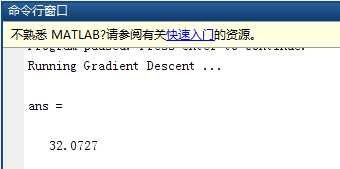
alpha = 0.01;

1. **Computing the cost** 

根据成本函数完成 computeCost.m，注意到X，y是训练组中的数据矩阵而不是非矢量数(scalar values)。当正确完成成本函数后，ex1.m 下一步将 theta 初始化为0，运行成功后会输出结果 32.07 。

补充代码：J = sum((X\*theta-y).^2)/(2\*length(y));

结果：

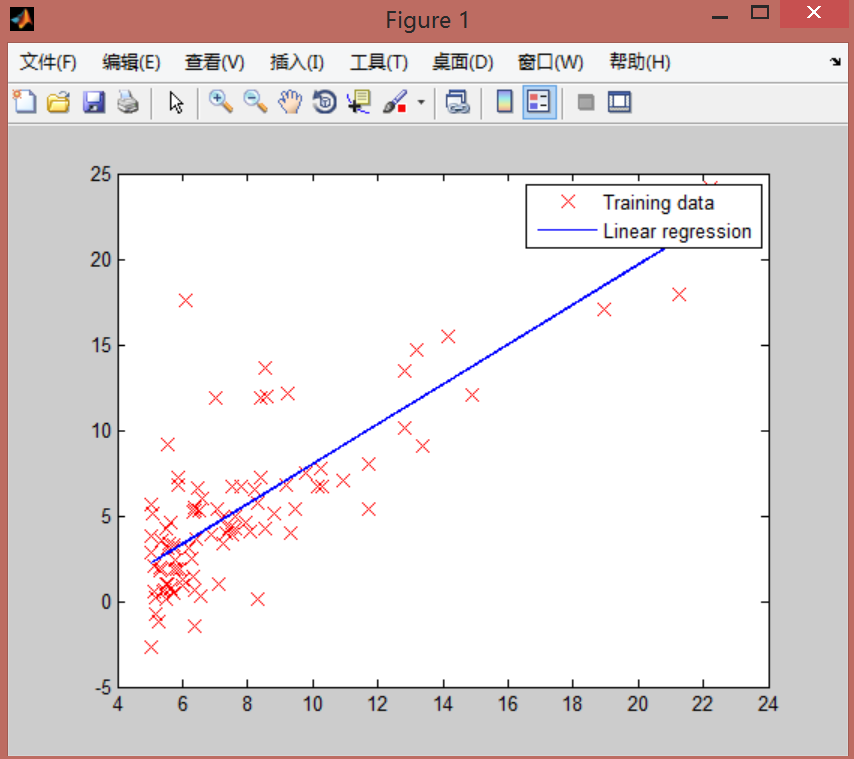


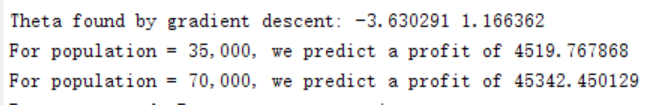
1. **Gradient descent**

完成 gradientDescent.m ，其中循环结构已经写好了，仅需要写每次更新  的函数即可。编程时要明确地知道，成本函数是改变  来最小化的，而不是 X 或者 y。可以通过观察结果是否逐步减小来判断梯度下降法的工作情况。正确的结果应该是不会增大，并且到后面收敛到一个稳定的值。ex1.m将会使用最后计算得到的绘制出一条匹配的直线。最后的值也会被用来在有35,000和70,000人口数量的地区进行预测。

补充代码：theta = theta - X'\*(X\*theta-y)/m\*alpha;

结果：





1. **Debugging**
2. **Visualizing** 

为了更好的理解成本函数，需要构建一个三维的图形。ex1.m 中已经写好了这一部分的代码，但是要理解为什么代码这么写。

% initialize J vals to a matrix of 0's

J vals = zeros(length(theta0 vals), length(theta1 vals));

% Fill out J vals

for i = 1:length(theta0 vals)

for j = 1:length(theta1 vals)

t = [theta0 vals(i); theta1 vals(j)];

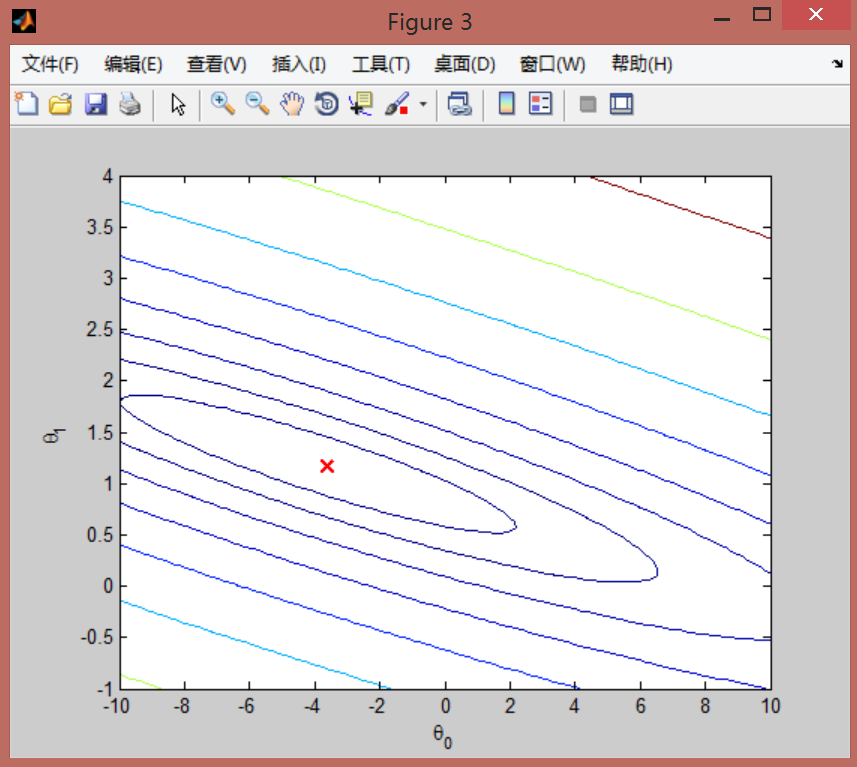
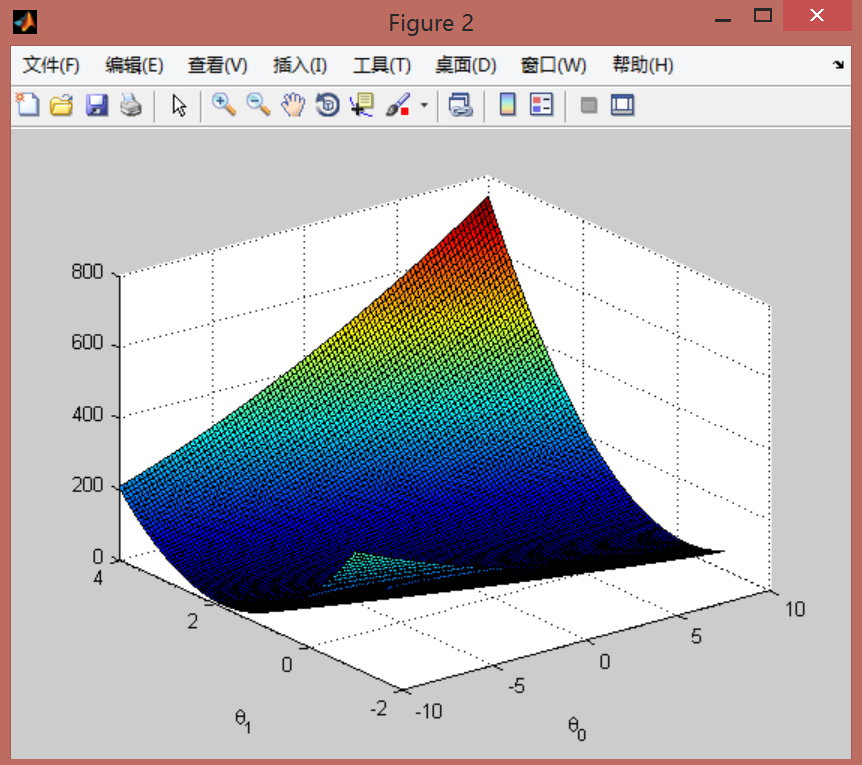
J\_vals(i,j) = computeCost(x, y, t);

end

end

执行完这几行代码后，可以得到有关值的二维数组。运行ex1.m后，通过使用surf和contour命令，构建出相应图形，展示的值如何随着和的变化而变化。最小点是 和的最适点，梯度下降的每一步都会更靠近这个点。

结果：



1. **Linear regression with multiple variable**

利用多元线性回归预测房子的价格。ex1data2.txt 提供了所需要的训练组，第一列是房子的面积，第二列是房间的数量，第三列是房子的价格。

**3.1 Feature Normalization**

完成featureNormalize.m中的代码。

1. 从数据集中计算出每个特征的平均值；
2. 计算完平均值后再计算标准差；
3. 特性缩放；

一种方式是使用标准差，另一种方式是使用差值。

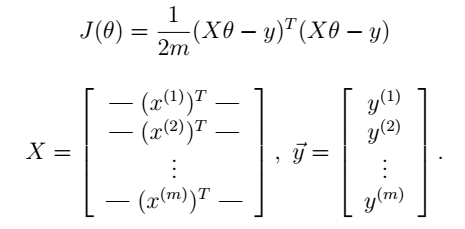
补充代码：mu = mean(X);

sigma = std(X);

X\_norm = (X - mu) ./ sigma;

**3.2 Gradient Descent**

多元线性回归问题和单变量线性回归问题的唯一区别在前者矩阵X的特征不止一个，假设函数和梯度下降的更新策略没有变化。补全computeCostMulti.m和GradientDescentMulti.m中的代码来完成多元线性回归的成本函数和梯度下降。在多变量中，损失函数也可以写成以下矢量形式：



补充代码：computeCostMulti.m

S= y - X\*theta;

for i = 1:m

J = J+S(i).\*S(i);

end

J = J/(2\*m);

GradientDescentMulti.m

n = length(theta);

theta1 = theta;

for i = 1:n

S = 0;

for j = 1:m

S = S + (X(j,:)\*theta-y(j)).\*X(j,i);

end

S = S\*alpha/m;

theta1(i) = theta(i) - S;

end

theta = theta1;

**3.2.1 Selecting learning rates**

在这个练习中，对数据集尝试不同的学习率并找到可以快速收敛的学习率。通过修改ex1\_multi.m中有关学习率的代码部分来改变学习率的值。

可以逐步地改变学习率，例如0.3,0.1,0.03,0.01等等。如果学习率过小，那么梯度下降要过很长一段时间才能收敛到最适值；如果学习率过大，梯度下降可能不收敛甚至会发散。

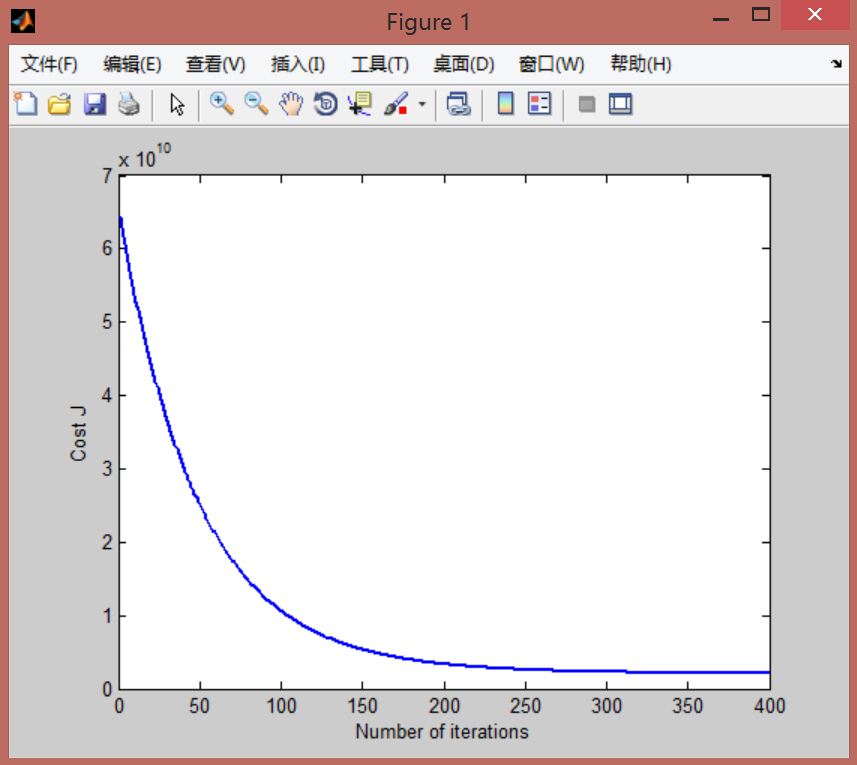
找到最佳学习率后，运行ex1\_multi.m去启动梯度下降直至收敛找到最后的值。然后用这个值去预测一间1650平方英尺、带3个卧室的房子价格，注意不要忘记特征归一化。

补充代码：% Estimate the price of a 1650 sq-ft, 3 br house

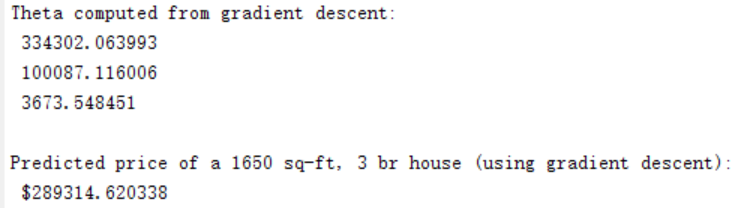
X\_predit = [1, ([1650 3] - mu) ./ sigma];

price = X\_predit \* theta;

结果：（找到了合适的学习率）



预测价格



**3.3 Normal Equations**

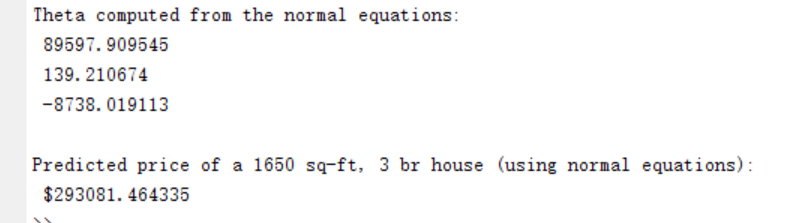
完成normalEqn.m中的代码，使用以下公式去计算：



计算之后可以预测 一间1650 平方英尺、带有 3 个卧室的房子的价格了。

补充代码：theta = pinv(X'\*X)\*X'\*y;

结果：



**Part II**

**Logistic Regression for Classfication**

**4 Logistic Regression**

在这个练习中，建立一个逻辑回归模型去预测一个学生能否被大学录取。可将之前申请者的数据作为逻辑回归的训练集。每个训练样本都有两场考试的成绩和录取结果。

**4.1 Visualizing the Data**

ex2.m中的代码会加载数据并且通过调用函数plotData（）绘制一个二维图,完成plotData.m中的代码。

补充代码：% Find Indices of Positive and Negative Examples

pos = find(y==1); neg = find(y == 0);

% Plot Examples

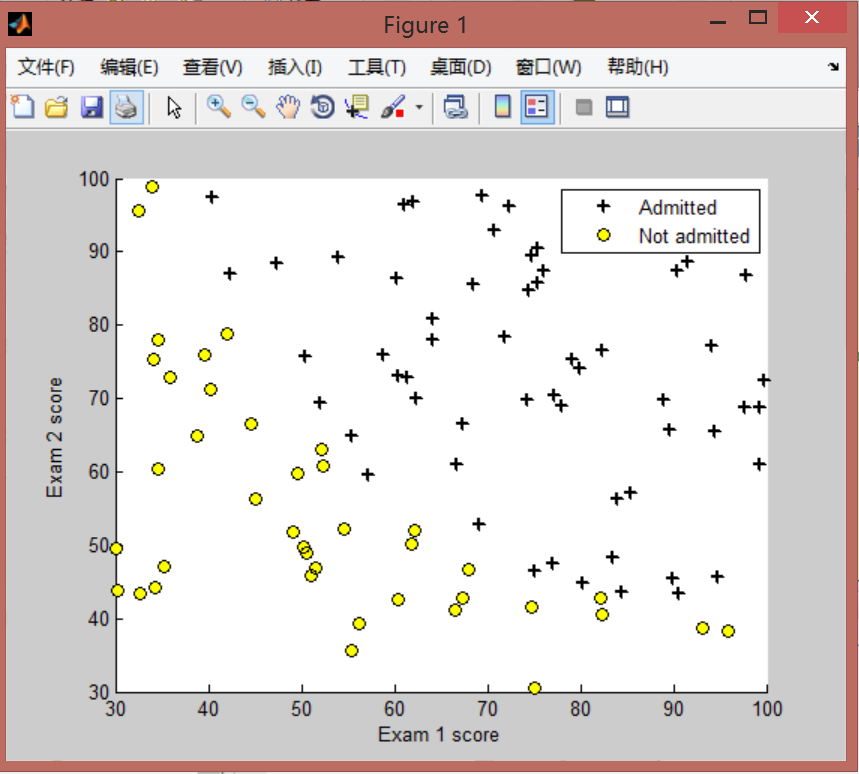
plot(X(pos, 1), X(pos, 2), 'k+','LineWidth', 2, ...

'MarkerSize', 7);

plot(X(neg, 1), X(neg, 2), 'ko', 'MarkerFaceColor', 'y', ...

'MarkerSize', 7);

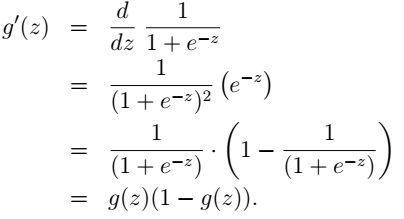
结果：



**4.2 Implementation**

**4.2.1 Warmup Exercise:sigmoid function**

sigmoid函数:



当z趋向于正无穷大时，g(z)趋近于1；当z趋向于负无穷大时，g(z)趋近于0。

首先将sigmoid.m中的代码（代码要同样可以用于矢量和矩阵，对于一个矩阵，函数应该能在每一个元素上使用）补全，以便可以被程序中其余部分所调用。

当x是很大的正数时，sigmoid(x)应该趋近于1；当x是很小的负数时，sigmoid(x)应该趋近于0；而sigmoid(0)的结果应该为0.5。

补充代码：[m,n] = size(z);

for i = 1:m

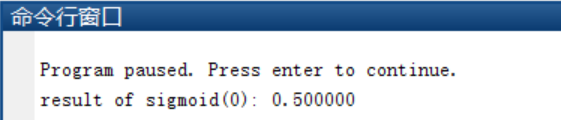
for j= 1:n

g(i,j) = 1/(1+exp(-z(i,j)));

end

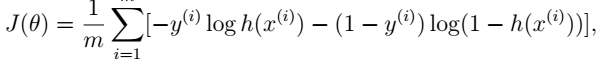
End

测试结果：

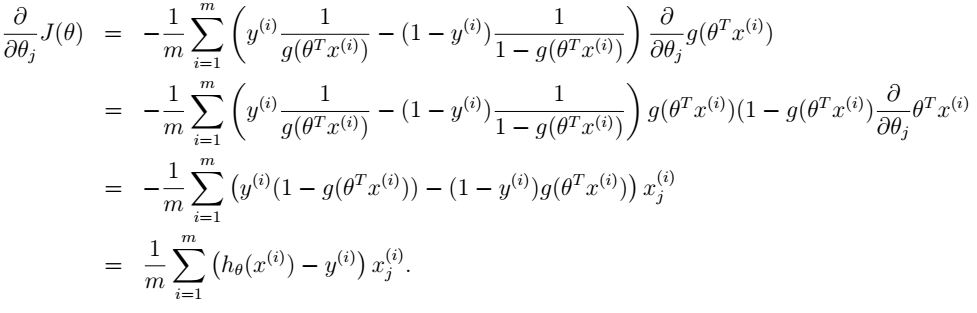


**4.3 Cost function and gradient**

完成costFunction.m中的代码以返回损失和梯度。逻辑回归中的损失函数为：



并且



补全代码后运行ex2.m，应当显示cost大约为0.693。

补充代码：for i = 1:m

h\_thetax = sigmoid(X(i,:)\*theta);

J = J - y(i)\*log(h\_thetax) -(1-y(i))\*log(1-h\_thetax);

end

J = J/m;

n = size(theta,1);

for j =1:n

sum = 0;

for i = 1:m

h\_thetax = sigmoid(X(i,:)\*theta);

sum = sum + (h\_thetax - y(i))\*X(i,j);

end

grad(j) = sum/m;

end

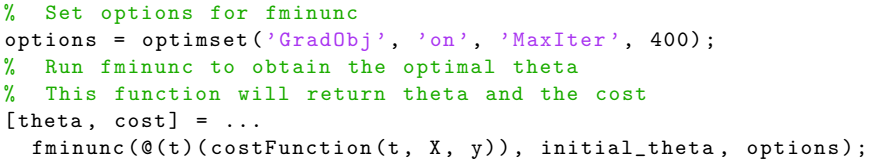
结果：



**4.4 Learning parameters using fminunc**

fminunc是一个用来找无约束函数最小值的很好的办法。在逻辑回归中，通过来优化。

具体地，就是用fminunc找到对逻辑回归损失函数来说最好的参数。在ex2.m中，已经给出如下正确代码去调用fminunc:

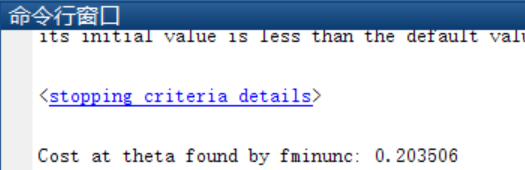


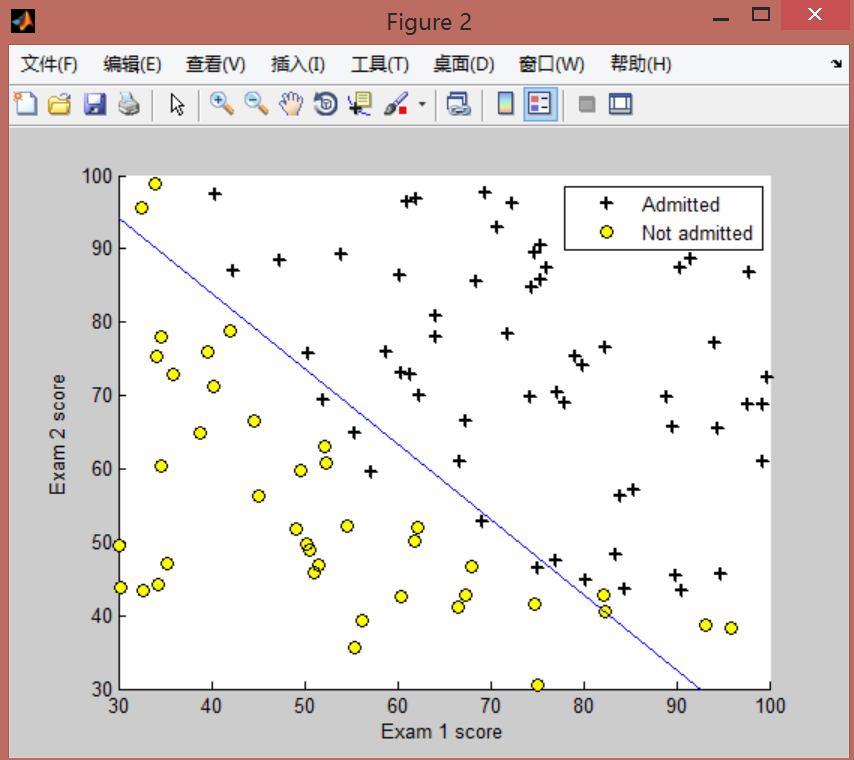
完成costFunction中的代码后，fminunc会在正确的优化参数上收敛并且返回最终的损失值和。

运行ex2.m，调用costFuntion函数，会看到损失值大约为0.203。

最后的值会被用来在训练数据上描绘决定边界（如何描绘一条边界的代码可见plotDecisionBoundary.m）。

结果：





**4.5 Evaluating logistic regression**

现在可以用搭建的模型来预测一个学生是否会被录取了。对一个第一次考试成绩为45、第二次考试成绩为85的学生来说，预测结果应为0.776。

完成predict.m中的代码，看建立的模型在训练集上工作地怎么样，这是检测找到的参数质量的另一种方法，预测函数会根据给定的数据集和theta产生结果“0”或者“1”。运行ex2.m，可以通过计算样例预测的正确的百分比来查看分类器的准确性。

补充代码：p = sigmoid(X\*theta);

for i = 1:m

if(p(i)>=0.5)

p(i)=1;

else

p(i)=0;

end

End

结果：

