编程练习2：逻辑回归

介绍

在本练习中，你将实现logistic回归并将其应用于两个实例不同的数据集。

开始练习之前，使用Octave中的cd命令来更改此目录。

此练习中包含的文件

ex2.m - Octave的脚本，将有助于逐步通过你的练习

ex2 reg.m - 练习后面部分的Octave脚本

ex2data1.txt - 前半部分的训练集

ex2data2.txt - 后半部分的训练集

mapFeature.m – 函数生成多项式特征

plotDecisionBounday.m - 用于绘制分类器决策边界的函数

plotData.m - 用于绘制二维分类数据的函数

sigmoid.m – Sigmoid函数

costFunction.m - 逻辑回归成本函数

predict.m - 回归预测函数

costFunctionReg.m - 正规化逻辑回归成

指示需要完成的文件

在整个练习中，你将使用脚本ex2.m和ex2 reg.m。这些脚本为问题设置数据集，并调用你编写的函数。你不需要修改它们中的任何一个。你只需按照本作业中的说明修改其他文件中的函数。

1 对数几率回归

在这部分练习中，你将建立一个逻辑回归模型来预测学生是否被大学录取。

假设你是一所大学的系主任，你想根据每个申请者两次考试的成绩来确定他们被录取的机会。你有以前申请者的历史数据，可以作为逻辑回归的训练集。对于每个培训示例，你都有申请人的两次考试成绩和录取决定。

你的任务是建立一个分类模型，根据这两次考试的分数来估计申请人被录取的概率。这个大纲和ex2.m中的框架代码将指导你完成这个练习。

1.1 可视化数据

在开始执行任何学习算法之前，如果可能的话，最好将数据可视化。在ex2.m的第一部分中，代码将加载数据并通过调用函数plotData将其显示在二维绘图上。

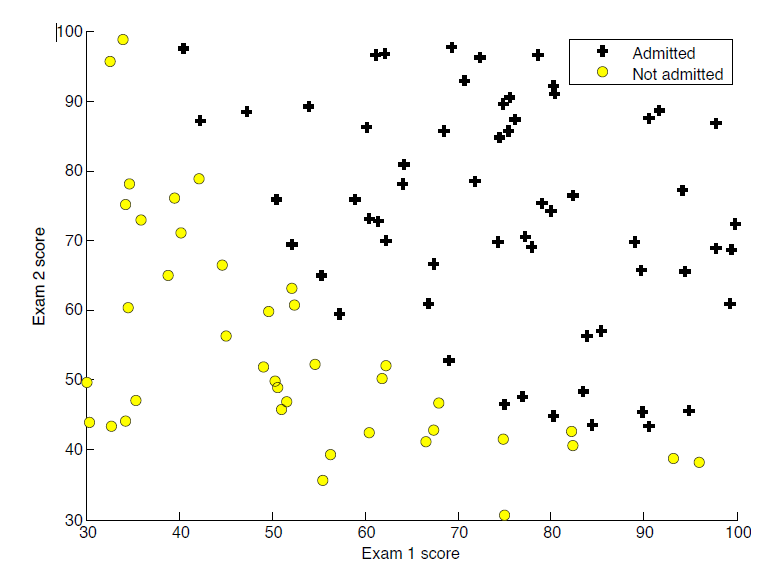
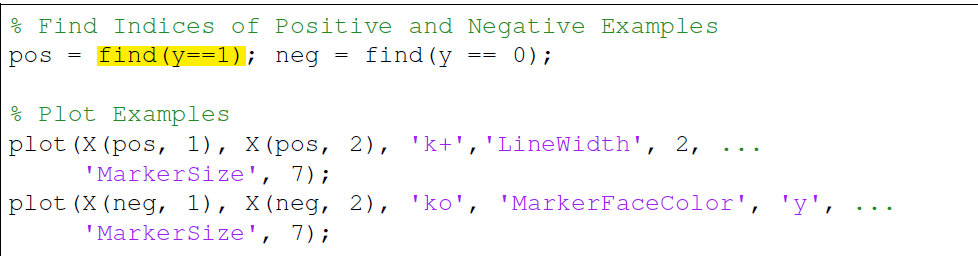
现在，你将在plotData中完成代码，使其显示如图1所示的图形，其中轴是两个考试分数，正示例和负示例用不同的标记显示。

图1：训练数据散点图

为了帮助你更熟悉绘图，我们将plotData.m保留为空，便于你尝试自己实现它。但是，这是一个可选（未分级）练习。我们还提供了下面的实现，以便您可以复制或引用它。如果您选择复制我们的示例，请确保通过查阅Octave文档了解每个命令的作用。



1.2 实施

1.2.1 热身练习：sigmoid函数

在开始实际成本函数之前，回想一下逻辑回归假设的定义是：

，

其中函数g是sigmoid函数。sigmoid函数定义为：

.

第一步是在sigmoid.m中实现这个函数，以便程序的其余部分可以调用它。完成后，在octave命令行调用sigmoid（x）来测试一些值。对于x的正值，sigmoid应该接近1，而对于负值，sigmoid应该接近0。计算sigmoid（0）应该正好得到0.5。你的代码也应该使用向量和矩阵。对于矩阵，函数应该对每个元素执行sigmoid函数。

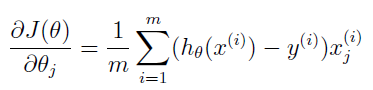
1.2.2 成本函数与梯度

现在您你实现logistic回归的成本函数和梯度。完成costFunction.m中的代码以返回成本和梯度。

回想一下，logistic回归中的成本函数是

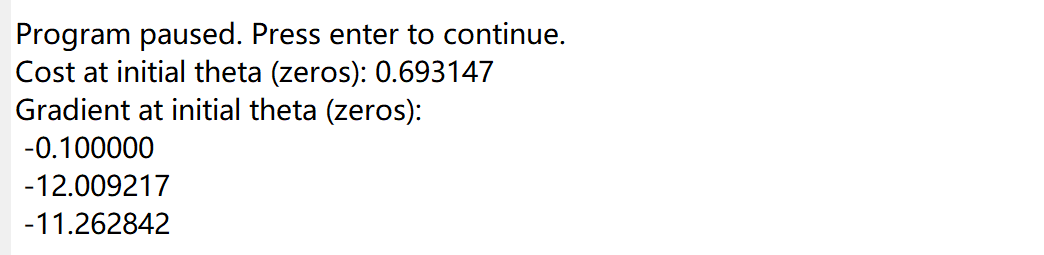
,

费用的梯度是与θ相同长度的向量，其中第j个元素（对于j=0,1，…，n）定义如下：



注意，虽然这个梯度看起来与线性回归梯度相同，但是公式实际上是不同的，因为线性回归和logistic回归对hθ（x）有不同的定义。

完成后，ex2.m将使用θ的初始参数调用costFunction。你应该看到成本大约是0.693。



1.2.3 利用fminunc学习参数

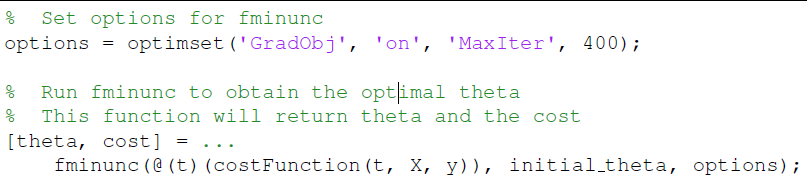
在上一个作业中，你通过实现梯度下降找到了线性回归模型的最佳参数。你已经编写了一个代价函数并计算了它的梯度，然后相应地采取了梯度下降步骤。这一次，你将使用名为fminunc的倍频程内置函数，而不是采取梯度下降步骤。

Octave的fminunc是一个寻找无约束函数最小值的优化解算器。对于logistic回归，需要使用参数θ优化成本函数J（θ）。

具体来说，你将使用fminunc为logistic回归成本函数找到最佳参数θ，给定一个固定的数据集（X和y值）。你将把以下输入传递给fminunc：

* 我们试图优化的参数的初始值。
* 当给定训练集和特定θ时，计算数据集（X，y）的逻辑回归成本和相对于θ的梯度的函数

在ex2.m中，我们已经编写了使用正确参数调用fminunc的代码。

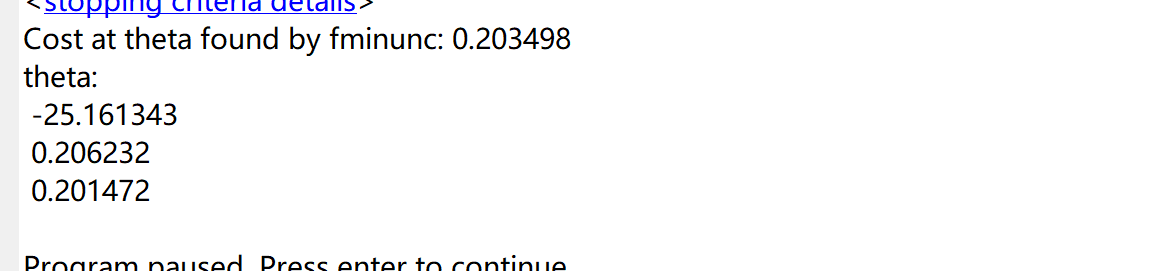


在这个代码段中，我们首先定义了要与fminunc一起使用的选项。具体来说，我们将GradObj选项设置为on，这会告诉fminunc我们的函数同时返回cost和gradient。这允许fminunc在最小化函数时使用梯度。此外，我们将MaxIter选项设置为400，以便fminunc在终止之前最多运行400步。

为了指定我们要最小化的实际函数，我们使用一个“速记”来指定带有@（t）的函数（costFunction（t，X，y））。这将创建一个参数为t的函数，该函数调用costFunction。这允许我们包装costFunction以与fminunc一起使用。

如果你正确地完成了cost函数，fminunc将收敛到正确的优化参数，并返回cost和θ的最终值。注意，通过使用fminunc，您不必自己编写任何循环，也不必像梯度下降那样设置学习速率。这一切都是由fminunc完成的：你只需要提供一个计算成本和梯度的函数。

一旦fminunc完成，ex2.m将使用θ的最佳参数调用costFunction函数。你应该知道费用大约是0.203.



最后的θ值将用于在训练数据上绘制决策边界，从而得到一个类似于图2的图。我们还建议你查看plotDecisionBoundary.m中的代码，了解如何使用θ值绘制这样的边界。

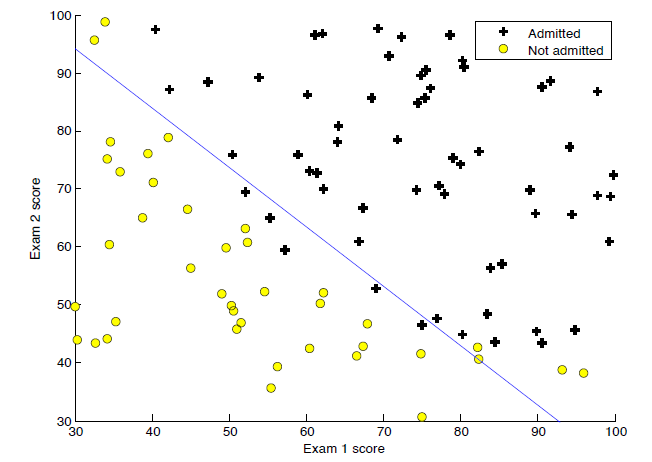


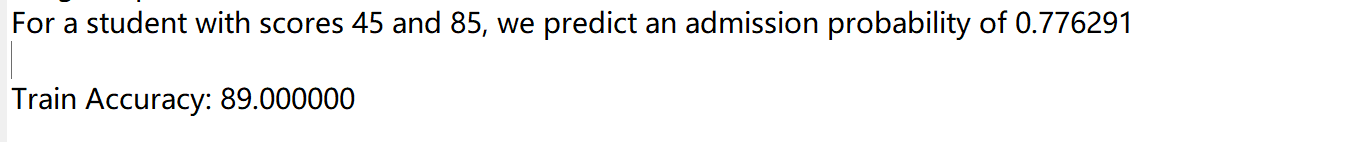
图2：具有决策边界的训练数据

1.2.4 评价logistic回归

在学习了这些参数之后，你可以使用该模型来预测某个学生是否会被录取。对于一次考试成绩为45分，二次考试成绩为85分的学生，你应该预计录取概率为0.776。

评估我们发现的参数质量的另一种方法是查看学习模型在我们的训练集上的预测效果。在这一部分中，你的任务是完成predict.m中的代码。给定一个数据集和一个学习的参数向量θ，predict函数将生成“1”或“0”预测。

在predict.m中完成代码后，ex2.m脚本将继续通过计算正确示例的百分比来报告分类器的训练精度。



2 正则logistic回归

在这部分练习中，你将实现正则化逻辑回归，以预测制造厂的微芯片是否通过质量保证（QA）。在QA过程中，每个微芯片都要经过各种测试，以确保其正常工作。

假设你是工厂的产品经理，在两个不同的测试中得到了一些微芯片的测试结果。从这两个测试中，你可以确定是否应该接受或拒绝微芯片。为了帮助你做出决定，你有一个过去芯片测试结果的数据集，从中你可以建立一个逻辑回归模型。

你将使用另一个脚本ex2 reg.m来完成这部分练习。

2.1 可视化数据

与本练习前面的部分类似，plotData用于生成如图3所示的图形，其中轴是两个测试分数，阳性（y=1，接受）和阴性（y=0，拒绝）示例用不同的标记显示。

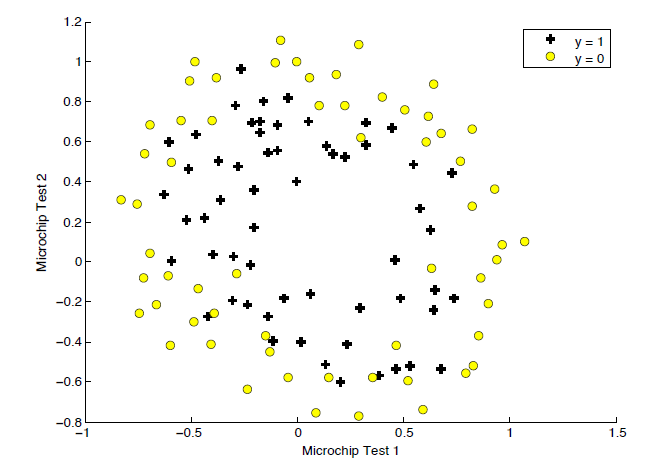
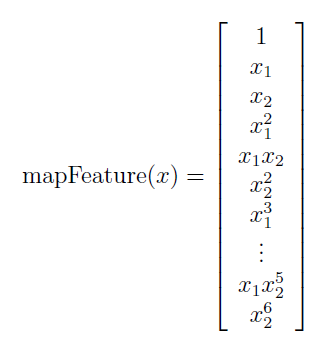


图3：训练数据图

图3显示，我们的数据集不能通过一条直线通过图被分成正反两个例子。因此，logistic回归的直接应用将不能在这个数据集上很好地执行，因为logistic回归只能找到一个线性决策边界。

2.2 特征映射

更好地拟合数据的一种方法是从每个数据点创建更多的特征。在提供的函数mapFeature.m中，我们将把特征映射为x1和x2的所有多项式项，直到六次方。



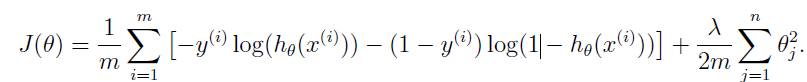
作为这个映射的结果，我们的两个特征向量（两个QA测试的分数）被转换成28维向量。在这个高维特征向量上训练的logistic回归分类器将具有更复杂的决策边界，并且在我们的二维图中绘制时将出现非线性。

虽然特征映射允许我们构建更具表现力的分类器，但它也更容易过度拟合。在练习的下一部分中，你将实现正则化logistic回归来拟合数据，还将亲眼看看正则化如何帮助解决过度拟合问题。

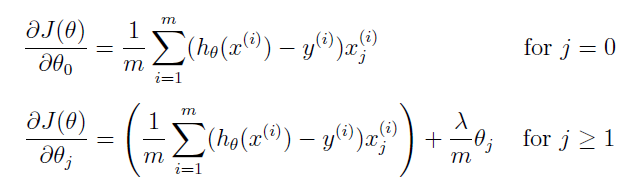
2.3 成本函数与梯度

现在你将实现代码来计算正则logistic回归的代价函数和梯度。完成costFunctionReg.m中的代码以返回成本和梯度。

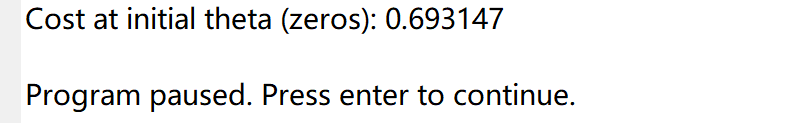
回想一下，logistic回归中的正则化成本函数是

 .

注意，不应该正则化参数θ0。在倍频程中，回想索引从1开始，因此，不应该正则化代码中的θ（1）参数（对应于θ0）。成本函数的梯度是一个向量，其中第j个元素定义如下：



完成后，ex2 reg.m将使用θ的初始值（初始化为全零）调用costFunctionReg函数。你应该看到成本大约是0.693。



2.3.1 利用fminunc学习参数

与前面的部分类似，你将使用fminunc学习最佳参数θ。如果你正确地完成了正则化logistic回归（costFunctionReg.m）的成本和梯度，你应该能够逐步通过ex2 reg.m的下一部分来使用fminunc学习参数θ。

2.4 绘制决策边界

为了帮助你将这个分类器学习到的模型可视化，我们提供了函数plotDecisionBoundary.m，它绘制（非线性）决策边界，将正反两个示例分开。在plotDecisionBoundary.m中，我们通过在均匀分布的网格上计算分类器的预测来绘制非线性决策边界，然后绘制预测从y=0变为y=1的等高线图。

在学习了参数θ之后，exreg.m中的下一步将绘制一个决策边界，类似于图4。

2.5 可选（未分级）练习

在本部分练习中，你将尝试为数据集使用不同的正则化参数，以了解正则化如何防止过度拟合。

请注意，随着λ的变化，决策边界发生了变化。使用一个小的λ，您应该会发现分类器几乎可以正确地获得每个训练示例，但是会绘制一个非常复杂的边界，从而过度拟合数据（图5）。这不是一个好的决策边界：例如，它预测x=（−0.25,1.5）处的一个点是可接受的（y=1），考虑到训练集，这似乎是一个错误的决策。

对于较大的λ，你应该看到一个图，该图显示了一个更简单的判定边界，该边界仍然可以很好地区分正数和负数。但是，如果将λ设置为太高的值，则不会得到很好的拟合，决策边界也不会很好地跟随数据，从而导致数据拟合不足（图6).

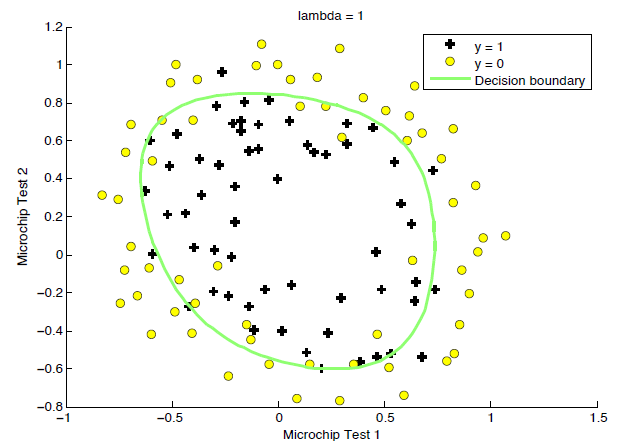


图4：具有决策边界的训练数据（λ=1）

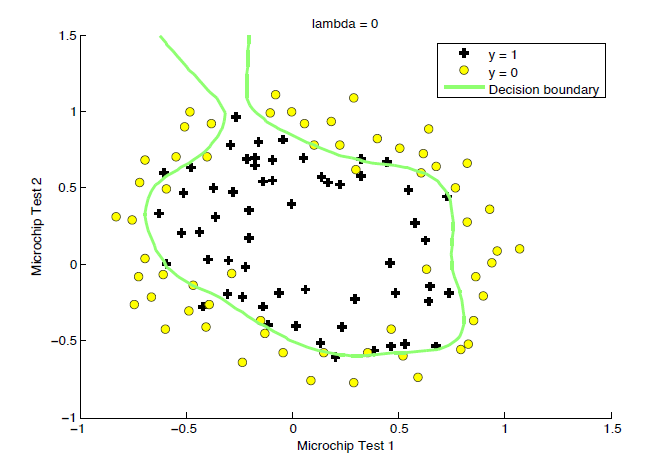


图5：无正则化（过拟合）（λ=0）

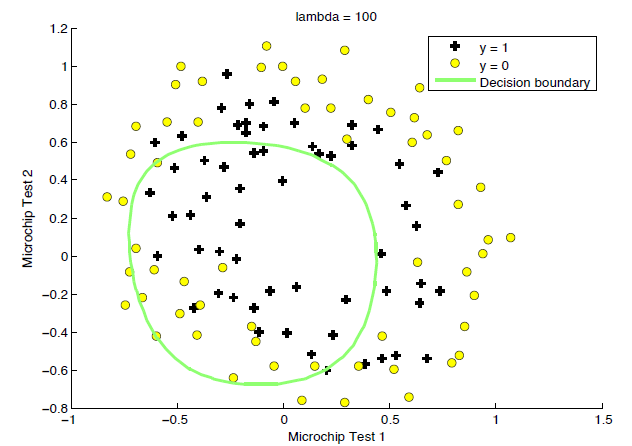


图6：正则化过多（欠拟合）（λ=100）