实验一：Logistic 回归

介绍：

在本练习中，您将实现逻辑回归，并将其应用于两个不同的数据集。

实验文件说明：

ex2.py - 可以帮助您逐步完成练习的 Python 脚本
ex2 reg.py -完成后面部分的 Python 脚本
ex2data1.txt – 数据集 1
ex2data2.txt - 数据集 2
mapFeature.py -生成多项式函数
plotDecisionBounday.py 绘制分类器的判定边界的函数
[*] plotData.py- 二维分类数据绘图函数
[*] sigmoid.py – Sigmoid 函数
[*] costFunction.py - Logistic 回归代价函数
[*] predict.py - Logistic 回归预测函数
[*] costFunctionReg.py 正则 Logistic 回归代价函数
带*的函数有部分代码缺少需要你按照提示将代码补充完整。

在整个实验中，需要使用 python 脚本 ex2.py 和 ex2_reg.py。这两个 Python 脚本会调用数据集和你将要补充完整的函数。你不需要修改它们中的任何一个。你只需按照本文档中的说明补充完整其它函数的代码。

1 逻辑回归

在本部分的实验中，你将构建一个逻辑回归模型来预测学生是否被大学录取。

假设你是一个大学部门的管理员，你希望根据每个申请人的两次考试成绩来确定他们的录取机会。你有以前的申请人的历史数据，你可以用它作为逻辑回归的训练集。

你的任务是建立一个分类模型，根据这两门考试的分数来估计申请人被录取的概率。这个文档中的 ex2.py 代码将指导你完成这项练习。

1.1 数据可视化

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。ex2.py 的第一部分时，代码将加载数据 ex2data1.txt 并通过调用函数 plotData 将其显示在一个二维图形上。

现在，你需要将在 plotData 中完成代码，以便显示如图 1 所示的图形，其中 x 和 y 轴代表的是两个考试分数，正示例和负示例用不同的标记显示。

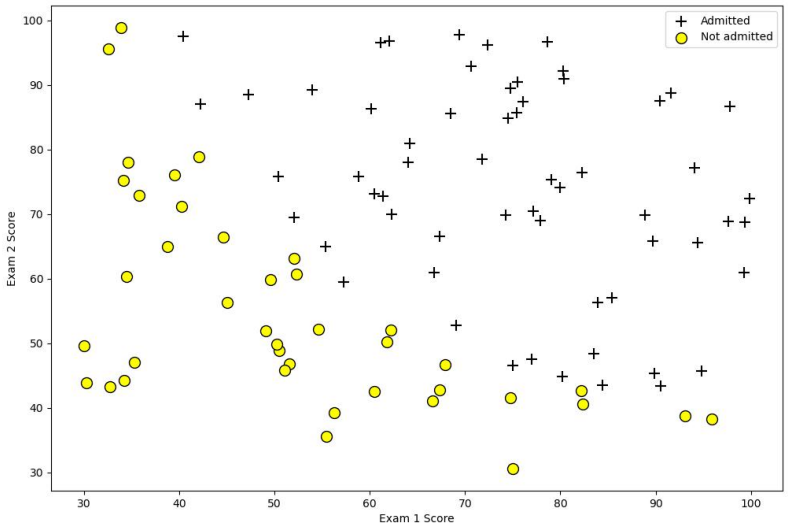

图 1:训练数据散点图

为了帮助你更熟悉 Python 绘图，你需要完成 plotData.py 中的空白部分。

1.2 实现

1.2.1 sigmoid 函数

逻辑回归假设被定义为:

$$h_\theta(x) = g(\theta^T x),$$

其中g函数为 sigmoid 函数。sigmoid 函数定义为:

$$g(z) = \frac{1}{1+e^{-z}}.$$

你的第一步是在 sigmoid.py 中实现这个函数,它可以被你程序的其余部分调用。当你完成之后，通过在 ex2.py 中调用 sigmoid(x)来尝试测试一些值。对于较大的正值 x，sigmoid(x)应该接近 1，而对于较大的负值，sigmoid(x)应该接近 0。求 sigmoid(0)应该正好是 0.5。你的代码也应该能处理向量和矩阵。对于一个矩阵，你的函数应该对每个元素执行 sigmoid 函数。

1.2.2 成本函数和梯度

你需要实现逻辑回归的成本函数和梯度。在 costFunction.py 中完成代码。返回代价和梯度。逻辑回归中的代价函数是

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log(h_\theta(x^{(i)})) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right],$$

代价的梯度是一个与θ长度相同的向量，其中第 j 个元素(j = 0,1，…， n)的定义如下:

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

注意，虽然这个梯度看起来与线性回归梯度相同，但公式实际上是不同的，因为线性回归和逻辑回归对$h_\theta(x)$有不同的定义。调用 costFunction.py，计算出的代价大约是 0.693。

1.2.3 使用 fmin_tnc 学习参数

在之前的作业中，你通过实现梯度下降找到了线性回归模型的最佳参数。你写了一个代价函数，计算了它的梯度，然后相应地做了一个梯度下降步骤。这一次，ex2.py 中使用了一个名为 fmin_tnc 的 Python 内置函数，而不是采用梯度下降步骤。

具体地说，你将使用 fmin_tnc 为逻辑回归函数找到最佳参数θ，给定一个固定的数据集(X 和 y 值)。你将传递给 fmin_tnc 以下输入:

•试图优化的参数的初始值。

•一个函数，当给定训练集和一个特定的θ时，计算数据集(X, y)关于θ的逻辑回归成本和梯度。

在 ex2.py 中已经编写了调用 fmin_tnc 的代码。

```
t = np.zeros(3).reshape(1, -1)  # 提供初始值

theta, _, _ = opt.fmin_tnc(func=costFunction, x0=t, args=(X, y))
cost, _ = costFunction(theta, X, y)
```

fmin_tnc 完成后，ex2.py 将使用的最优参数θ调用 costFunction 函数。你应该看到成本大约是 0.203。

这个最终的θ值将被用来在训练数据上绘制决策边界，得到类似于图 2 的图。我们也鼓励你看看 plotDecisionBoundary.py 中的代码,看看如何用θ值画出这样的边界。

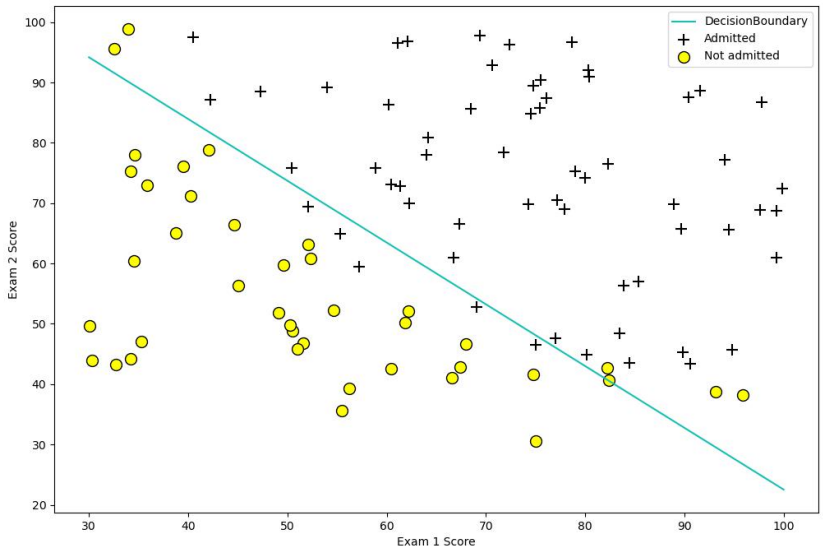

图 2:具有决策边界的训练数据集

1.2.4 logistic 回归评估

学习了这些参数之后，你就可以用这个模型来预测某个学生是否会被录取。对于一个考试一得分 45 分，考试二得分 85 分的学生，你应该会看到 0.776 的录取概率。

另一种评估参数质量的方法是看看学习的模型在我们的训练集上预测得有多好。在这一部分中，需要完善 predict.py 中的代码。给定一个数据集和一个学习过的参数向量θ时，预测函数将产生“1”或“0”预测。

在完成了 predict.py 中的代码之后。Python 将通过计算正确示例的百分比来报告分类器的训练准确性。

2 正则逻辑回归

在实验的这一部分中，将实现正则逻辑回归，以预测来自制造厂的微芯片是否通过质量保证(QA)。在质量保证过程中，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，你在两个不同的测试中获得了一些微芯片的测试结果。从这两个测试中，你想要确定微芯片应该被接受还是被拒绝。为了帮助你做出决定，您有一个过去微芯片测试结果的数据集，你可以从中构建一个逻辑回归模型。

你将使用另一个 Python 脚本 ex2_reg.py,来完成实验的这一部分。

2.1 数据可视化

与本练习的前面部分类似，plotData.py 用于生成如图 3 所示的图，其中 x 轴和 y 轴表示的是两个测试分数，阳性(y = 1，接受)和阴性(y = 0，拒绝)的示例用不同的标记显示。

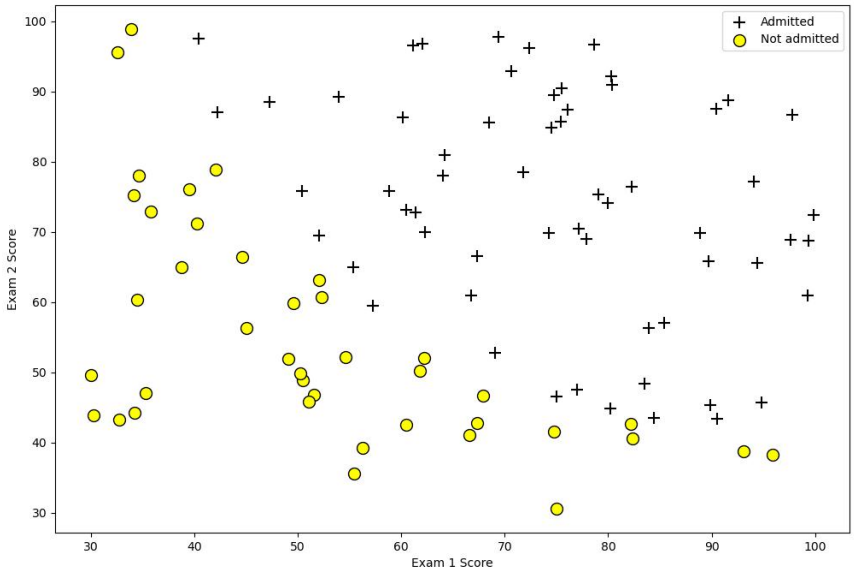

图 3:训练数据图

因为逻辑回归只能找到一个线性的决策边界，因此，直接应用逻辑回归将不适用于这个数据集。

2.2 特征映射

更好地适应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.py 中，将 x1 和 x2 利用多项式映射到到六次幂。

$$\text{mapFeature}(x) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_1 x_2 \\ x_2^2 \\ x_1^3 \\ \vdots \\ x_1 x_2^5 \\ x_2^6 \end{bmatrix}$$

通过这种映射，我们的两个特征向量(两个 QA 测试的分数)被转换成一个 28 维向量。在这个高维特征向量上训练的逻辑回归分类器将有一个更复杂的决策边界，并显示为非线性边界。

虽然特征映射允许我们构建一个更具表现力的分类器，但它也更容易出现过拟合。在本实验的下一部分中，你将实现正则逻辑回归来拟合数据，并亲眼看看正则化如何帮助解决过拟合问题

2.3 代价函数和梯度

完善 costFunctionReg.py 中的代码计算正则逻辑回归的代价函数和梯度，返回成本和梯度。logistic 回归中的正则化代价函数是：

$$J(\theta) = \left[\frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log(h_\theta(x^{(i)})) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right]\right] + \frac{\lambda}{2m}\sum_{j=2}^{n}\theta_j^2.$$

代价函数的梯度是一个向量，其中第 j 个元素定义如下:

$$\frac{\partial J(\theta)}{\partial \theta_0} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} \qquad \text{for } j = 0$$

$$\frac{\partial J(\theta)}{\partial \theta_j} = \left(\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}\right) + \frac{\lambda}{m}\theta_j \quad \text{for } j \geq 1$$

完善 costFunctionReg.py 中的代码后，在 ex2_reg.py 中调用函数 costFunctionReg.py 计算出的 cost 大约是 0.693。

2.3.1 使用 fmin_tnc 学习参数

与前面部分类似，您将使用 fmin_tnc 函数来学习最优参数θ。如果你已经正确地完成了正则化逻辑回归的成本和梯度(costFunctionReg.py)，你应该能够使用 ex2_reg.py 的下一部分用 fmin_tnc 学习参数θ。

2.4 绘制决策边界

为了帮助你可视化这个分类器学习的模型，函数 plotDecisionBoundary.py 绘制出(非线性的)决策边界，该边界将正例和负例分开。在学习了参数θ后，ex_reg.py 将可以绘制类似于图 4 的决策边界。

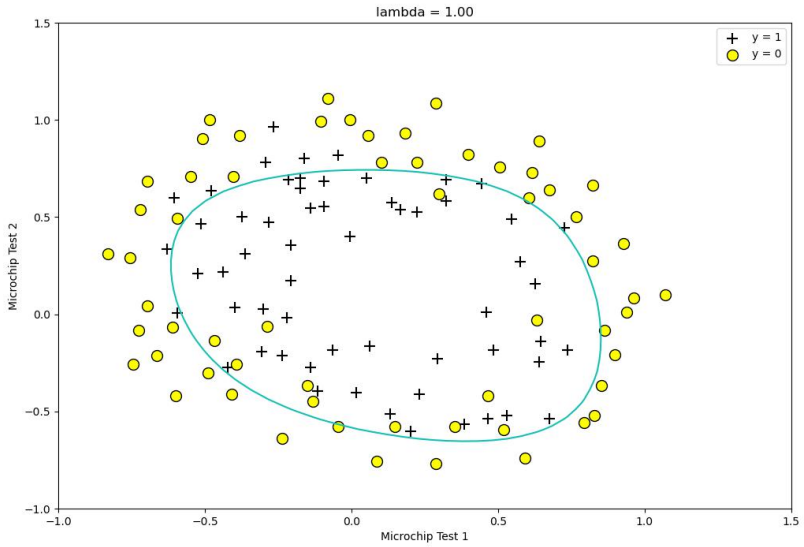

图 4:具有决策边界的训练数据(λ=1)

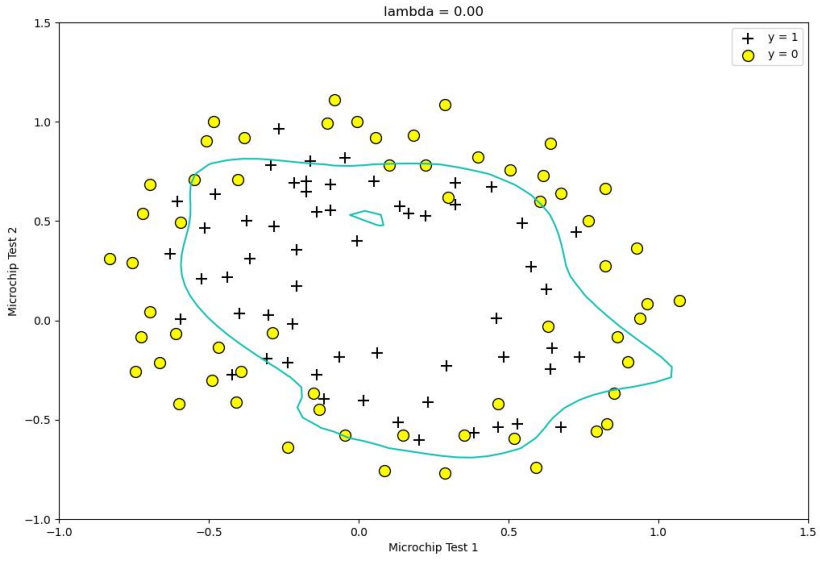

图 5:无正则化(过拟合)(λ = 0)

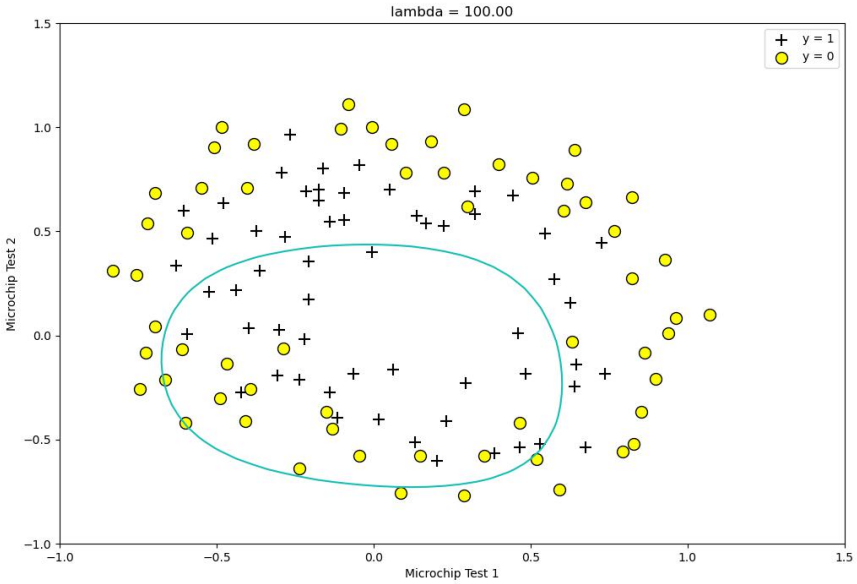

图 6:过度正则化(欠拟合)(λ=100)