实验 2：Logistic回归

实验目的

1、实现Logistic回归算法
2、实现和sigmoid函数、基于交叉熵的损失函数和梯度计算
3、调用最小化函数实现梯度下降算法

实验数据

1. ex2data1.txt-用于线性分类的数据集（高校录取预测）
2. ex2data2.txt-用于非线性分类的数据集（芯片质量预测）

实验步骤

1. 线性分类问题

在这部分的练习中，你将建立一个Lgistic回归模型来预测一个学生是否被大学录取。假设你是一个大学某院系的管理者，你想根据每个申请人的两次考试成绩来确定他们是否由能够入学。你有以前申请人的历史数据，可以作为训练Logistic模型的训练集。

在文件ex2data1.txt 中包含了我们本次线性分类实验的数据集，数据共三列：每行表示一个申请人的历史数据，前两列为申请人的两个成绩；第三列为标签，1表示能够入学，0表示不能入学。

1.1 读取数据

首先你需要做的是将ex2data1.txt 文件中的数据进行读取，使用的方法是numpy.loadtxt()，具体要求如下：

1. 使用loadtxt函数读取数据存于变量ex2data1，注意指定分隔符参数。

2. 使用变量X储存ex2data1的前两列数据（申请人的两科成绩）。

3. 使用变量y储存ex2data1的第三列数据（标签，1表示能够入学，0表示不能入学）,存为列向量。

4. 使用变量m储存样本数量。

代码：

```
ex2data1 = np.loadtxt('.\ex2data1.txt',delimiter=',')
X = ex2data1[:,0:2]
y = ex2data1[:,[2]]#注意存为列向量的写法
m = np.shape(y)[0]
```

cell[2]正确输出：(100, 2) (100, 1)

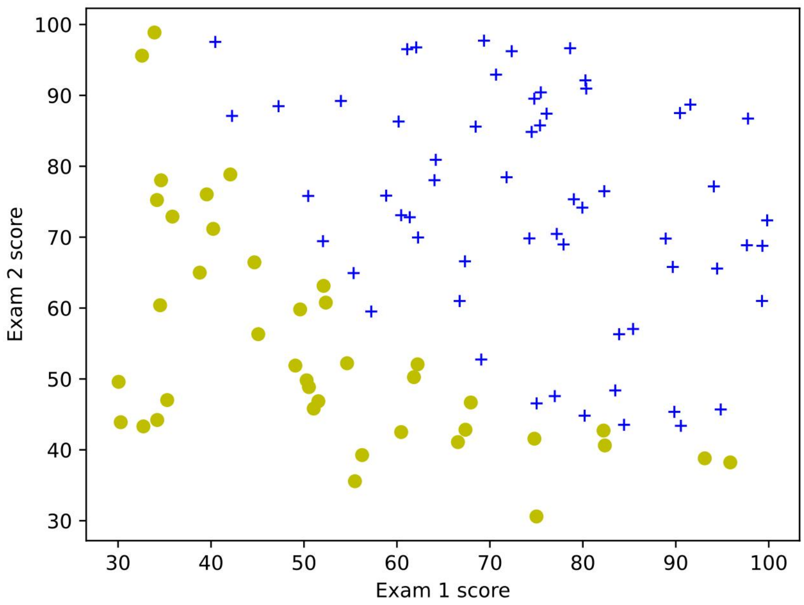
1.2 可视化数据

对数据进行可视化有助于更好的理解数据集的分布，对于本次实验的数据，可以通过plt.plot()函数绘制散点图。具体要求为：分别以两次考试的成绩为x、y轴绘制散点图，并使用不同颜色和形状的散点区分正例和反例。比如被录取（y标签为1）则点显示为蓝色“+”点，未被录取（y标签为0）则点显示为黄色“o”点。

代码：

```
plt.xlabel('Exam 1 score')
plt.ylabel('Exam 2 score')
for i in range(m):
  if y[i] == 1: #正例
    plt.plot(X[i,0], X[i,1], '+b')
  elif y[i] == 0: #反例
    plt.plot(X[i,0], X[i,1], 'oy')
```

cell[3]正确输出：



1.3 训练Logistic回归模型

该部分你要完成数据预处理、定义sigmoid函数、定义损失函数和计算梯度，然后本次实验将尝试使用scipy.optimize.minimize()函数自动求取theta最优解。

1.3.1 数据预处理：准备输入数据、标签，初始化θ

首先需要进行数据的准备，包括用于训练的样本x和标签y，以及初始化输出theta数组。具体步骤为：

1. 前面已经使用m存放样本个数，X存放成绩信息，y存放标签。
2. 使用np.ones创建变量名为x0的数组，规模为m*1。
3. 使用np.hstack函数将x0和X进行合并存放于变量x。
4. 使用np.zeros初始化theta,规模为3*1（每个样本有两个特征值）。

Cell[4]正确输出：(100, 3)　[[0]
[0]
[0]]

1.3.2 定义sigmoid函数、损失函数、梯度

在这部分需要你完成sigmoid()、costFunction()、gradient()三个函数。

1.sigmoid函数的公式为：

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

sigmoid()函数对输入数组z中每行元素按公式做计算后返回一个数组g。以e为底的指数计算可以使用numpy.exp(n)实现（该语句为计算e^n的值）。

2.costFunction()函数用于计算基于交叉熵的损失函数，公式为：

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

其中，$h_\theta(\boldsymbol{x}^{(i)}) = g(\boldsymbol{\theta^T x}) = \frac{1}{1+e^{-\boldsymbol{\theta^T x}}}$，计算时注意矩阵乘法的维度要求。

计算代价时可提出负号后将sigma运算分为$\sum_{i=1}^{m}[y^{(i)}\,log(h_\theta(\boldsymbol{x}^{(i)})]$和$\sum_{i=1}^{m}[(1-y^{(i)})\,log(1-h_\theta(\boldsymbol{x}^{(i)}))]$两部分相加。第一部分可由存放标签的y数组和取对数后的$h_\theta(\boldsymbol{x}^{(i)})$做矩阵乘法实现，后一部分类似。对数操作可以通过numpy.log(n)实现（该语句为计算log n的值）。

代码中先对theta做了reshape操作，最后对代价值J做了flatten操作，均为适应op.minimize()函数对参数的要求，暂可忽略。

3. 计算梯度函数gradient()函数实现，梯度计算公式为：

$$grad = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(\boldsymbol{x}^{(i)}) - y^{(i)})\,x_j^{(i)}$$

其中，$h_\theta(\boldsymbol{x}^{(i)}) = g(\boldsymbol{\theta^T x}) = \frac{1}{1+e^{-\boldsymbol{\theta^T x}}}$，sigma运算可通过残差（$h_\theta(\boldsymbol{x}^{(i)}) - y^{(i)}$）数组与x做矩阵乘法实现，计算时注意矩阵乘法的维度要求。该步计算得到的grad为3*1的数组，代码中先对theta做了reshape操作，最后对代梯度grad做了flatten操作，均为适应op.minimize()函数对参数的要求，暂可忽略。

cell[5]正确输出：　对初始零向量theta求得的cost为 [0.69314718]
梯度为 [-0.1　　-12.00921659 -11.26284221]

1.3.3 使用scipy.optimize.minimize求最小损失对应的参数theta

在这部分，你将学习使用scipy.optimize.minimize()自动求取最优参数。使用时主要需要传入的参数为minimize(fun, x0, args=(), method, jac)，minimize()函数对参数要求很严格，具体要求如下：

1.fun为进行优化的目标函数，传入需调用的函数名（不需要加括号），此处为fun=costFunction。需注意调用的函数第一个参数（theta）和返回值（J）必须为一维数组。

2.x0即theta需传入一维数组。

3.args传入fun需要的其他参数，需用tuple传入。

4.method指定优化算法，此处我们使用method='TNC'。

5.jac调用梯度计算函数传入参数需与fun调用函数完全相同，且返回值为一维数组。此处为jac=gradient

高维数组a调整为一维可以使用a.flatten()，该函数会产生一个副本，不会直接改变a的维度

在运行优化函数前首先验证要传入的参数是否符合维度要求。

Cell[7]正确输出：1 1 1

Cell[8]正确输出：

fun: 0.203497701589474366
jac: array([8.75697940e-09, 6.43645929e-08, 4.71900562e-07])
message: 'Local minimum reached (|pg| ~= 0)'
nfev: 36
nit: 17
status: 0
success: True
x: array([-25.16131869,   0.20623159,   0.20147149])

1.4 评估Logistic回归模型

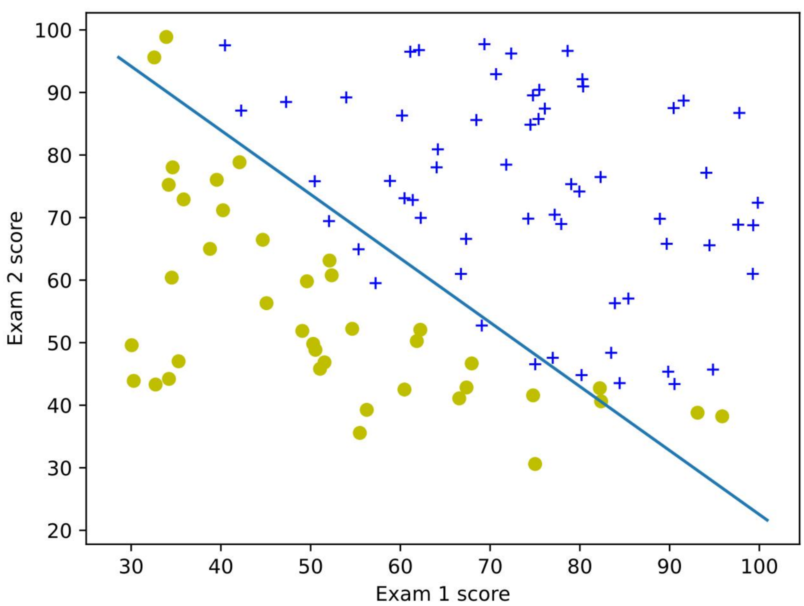
1.4.1 绘制决策边界

通过训练你已经得到最佳的参数，存放于theta_star中。现在可以利用theta_star绘制决策边界。决策边界的方程为：$\theta_0+\theta_1x_1+\theta_2x_2=0$

首先我们依然画出数据集的散点图，然后通过带入两个x_1的值计算得到x_2，由于我们的图像是以x_1和x_2为x、y轴做出，因此相当于得到了图像上的两个点，连线画出决策边界即可。注意两个x_1的选取尽量在x_1的最大/最小值以外，这样使用plot连线的长度足以区分所有散点。

画出直线可以使用函数plt.plot((x0,x1),(y0,y1))。

cell正确输出：



1.4.2 计算模型准确率

在这部分你需要编写函数计算模型在训练集上的正确率，注意预测值$h_\theta(x^{(i)})$>0.5则为正例,反之则为负例。正确结果为89%。

1.4.3 使用训练得到模型进行预测

用训练得到的theta_star，预测一个学生在考试1中获得45分，在考试2中获得85分，该生被录取的概率。Sigmoid函数的返回值介于0和1之间，可以代表概率，即返回$h_\theta(x^{(i)}) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$。正确结果为0.7762906253511527。

2. 非线性分类问题（见下周实验报告）