# 机器学习-逻辑回归

# 逻辑回归

建立一个逻辑回归模型，用于预测一个学生是否应该被大学录取。

简单起见，大学通过**两次考试的成绩来确定一个学生是否应该录取**。你有以前数届考生的成绩，可以做为训练集学习逻辑回归模型。每个训练样本包括了考生两次考试的成绩和对应的录取决定。

你的任务是建立一个分类模型，根据两次考试的成绩来估计考生被录取的概率。
本次实验需要实现的函数

- `plot_data` 绘制二维的分类数据。
- `sigmoid` 函数
- `cost_function` 逻辑回归的代价函数
- `cost_gradient` 逻辑回归的代价函数的梯度，无正则化
- `predict` 逻辑回归的预测函数
- `cost_function_reg` 逻辑回归带正则化项的代价函数
- `cost_gradient_reg` 逻辑回归的代价函数的梯度，带正则化

## 数据可视化

在实现机器学习算法前，可视化的显示数据以观察其规律通常是有益的。本次作业中，你需要实现 `plot_data` 函数，用于绘制所给数据的散点图。你绘制的图像应如下图所示，两坐标轴分别为两次考试的成绩，正负样本分别使用不同的标记显示。

## 热身练习：Sigmoid函数

逻辑回归的假设模型为：

$$h_\theta(x) = g(\theta^{\mathrm{T}} x)$$

其中函数 $g(\cdot)$ 是Sigmoid函数，定义为：

$$g(z) = \frac{1}{1 + \exp(-z)}$$

本练习中第一步需要你实现 Sigmoid 函数。在实现该函数后，你需要确认其功能正确。对于输入为矩阵和向量的情况，你实现的函数应当对每一个元素执行Sigmoid 函数。

## 代价函数与梯度

现在你需要实现逻辑回归的代价函数及其梯度。补充完整`cost_function`函数，使其返回正确的代价。补充完整`cost_gradient`函数，使其返回正确的梯度。

逻辑回归的代价函数为:

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

对应的梯度向量各分量为

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

# 预测函数

在获得模型参数后，你就可以使用模型预测一个学生能够被大学录取。如果某学生考试一的成绩为45，考试二的成绩为85，你应该能够得到其录取概率约为0.776。

你需要完成 `predict` 函数，该函数输出“1”或“0”。通过计算分类正确的样本百分数， 我们可以得到训练集上的正确率。

# 使用`scipy.optimize.fmin_cg`学习模型参数

在本次作业中，希望你使用 `scipy.optimize.fmin_cg` 函数实现代价函数 $J(\theta)$ 的优化，得到最佳参数 $\theta^*$ 。

使用该优化函数的代码已经在程序中实现，调用方式示例如下:

```
ret = op.fmin_cg(cost_function,
                 theta,
                 fprime=cost_gradient,
                 args=(X, y),
                 maxiter=400,
                 full_output=True)
theta_opt, cost_min, _, _, _ = ret
```

其中`cost_function`为代价函数， `theta` 为需要优化的参数初始值， `fprime=cost_gradient` 给出了代价函数的梯度， `args=(X, y)` 给出了需要优化的函数与对应的梯度计算所需要的其他参数， `maxiter=400` 给出了最大迭代次数， `full_output=True` 则指明该函数除了输出优化得到的参数 `theta_opt` 外，还会返回最小的代价函数值 `cost_min` 等内容。

对第一组参数，得到的代价约为 0.203 (cost_min)。

# 逻辑回归的正则化形式

# 数据可视化

调用函数 `plot_data` 可视化第二组数据 `LR_data2.txt` 。
正确的输出如下：

LR_data2

# 特征变换

创建更多的特征是充分挖掘数据中的信息的一种有效手段。在函数 map_feature 中，我们将数据映射为其六阶多项式的所有项。

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

# 代价函数与梯度

逻辑回归的代价函数为

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

对应的梯度向量各分量为：

$$\begin{aligned} \frac{\partial J(\theta)}{\partial \theta_0} &= \frac{1}{m} \sum_{i=1}^{m} h\left( h_\theta(\boldsymbol{x}^{(i)}) - y^{(i)} \right) x_0^{(i)} & \text{for } j = 0 \\ \frac{\partial J(\theta)}{\partial \theta_j} &= \frac{1}{m} \sum_{i=1}^{m} h\left( h_\theta(\boldsymbol{x}^{(i)}) - y^{(i)} \right) x_j^{(i)} + \frac{\lambda}{m} \theta_j & \text{for } j \geq 1 \end{aligned}$$

完成以下函数：

- `cost_function_reg()`
- `cost_gradient_reg()`

# 模型训练

如果将参数$\theta$ 初始化为全零值，相应的代价函数约为 0.693。可以使用与前述无正则化项类似的方法实现梯度下降，
获得优化后的参数 $\theta^*$ 。

你可以调用 plot_decision_boundary 函数来查看最终得到的分类面。建议你调整正则化项的系数，分析正则化对分类面的影响!

参考输出图像：