斯坦福大学公开 Machine Learning 学习笔记

第三章 逻辑回归（Logistic Regression）

之前第一章学习过监督学习时将其简单分为线性回归和逻辑回归，而逻辑回归一般都是用来解决离散的问题，即适用于 y 取离散值的情况。逻辑回归属于分类（Classification）算法的一种，后面学习的神经网络和支持向量机算法都属于分类算法。
对于二分类问题，y=0 或 y=1，线性回归的$h_\theta(x)$可以取到>1 或<0，而对于逻辑回归算法 0≤ $h_\theta(x) \leq 1$。

3.1 逻辑回归模型

逻辑回归的目的是要找出一个假设函数$h_\theta(x) = \mathrm{g}(\Theta^T \mathrm{x})$，其取值范围为 0≤ $h_\theta(x) \leq 1$，用来预测一个未知的样本属于已知类的哪个类。g 函数称为为 sigmoid 函数或 Logistic 函数，公式为：

$$\mathrm{g}(\mathrm{z}) = \frac{1}{1+e^{-z}} \qquad \text{(公式 1)}$$

将$\Theta^T \mathrm{X}$代入到 Sigmoid 函数得出假设函数的公式:

$$h_\theta(x) = \frac{1}{1+e^{-\Theta^T \mathrm{X}}} \qquad \text{(公式 2)}$$

其中，sigmoid 函数的图形如下图：

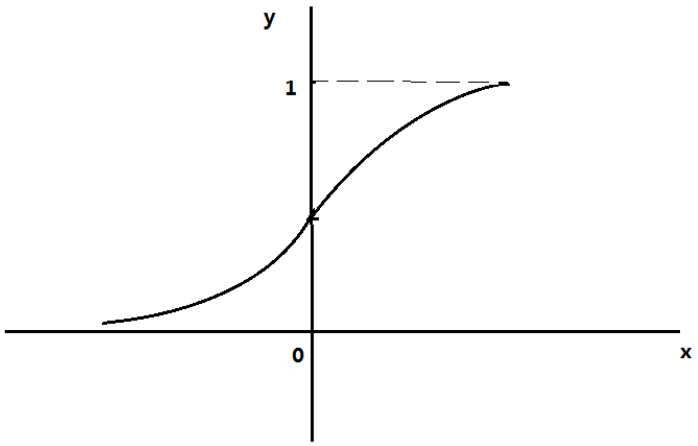

图 3-1 sigmoid 函数的图形

从图中我们可以看出，所谓的$h_\theta(x)$的意义便是：估计（预测）新数据 x 对应 y=1 的概率值。例如在癌症预测中，如果$\mathrm{x} = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ tumorsize \end{bmatrix}$，预测出$h_\theta(x) = 0.7$，那么就可以告得出结论，有 70%的概率癌细胞是恶性的。
以上用概率公式可表示为：$h_\theta(x) = P(y = 1|x;\theta)$，而且，$P(y = 0|x;\theta) + P(y = 1|x;\theta) = 1$。

3.2 决策边界（Decision Boundary）

从图 3-1 中我们应该会有个疑惑，$h_\theta(x)$计算出来的是个大于等于 0 小于等于 1 的数，但是我们期望 y 的结果是 1 或 0，这个时候就要求我们做出决定，到底当$h_\theta(x)$大于等于多少时取 y=1，小于多少 y=0。举癌症预测的例子，当$h_\theta(x)$的概率多大时，我们认为癌细胞是恶性的，小于多少是良性的，假定我们取值 n，且假设当Θ^Tx=N 是$h_\theta(x) = n$，则 x 对应的值就是假设函数的决策边界，当Θ^Tx $\geq$ N，认为 y=1，其他 y=0。对于决策边界的取值取决于实际中期望的效果。例如癌症预测中，决策边界取决于是宁愿将良性的误预测为恶性而不漏过可能的恶性，还是要求高准确率的预测恶性肿瘤。

3.3 代价函数（cost function）及逻辑回归算法

代价函数

有了假设函数，接下来的工作和线性回归的差不多，就是通过训练样本训练出最优的特征向量。和线性回归相同，训练中也需要代价函数。

在线性回归中，代价函数为：J(θ) = $\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$，从中我们可以提取出单个样本的代价函数为：$\text{cost}(h_\theta(\text{x}),\text{y}) = \frac{1}{2}(h_\theta(x) - y)^2$，但是将这样的代价函数用在逻辑回归中，得出的代价函数是非凸的（non-convex），而我们希望得出一个凸（convex）的代价函数，这样有利于我们使用梯度下降法或者其他的算法来求得代价函数的全局最小值最优解，但是如果代价函数是非凸的，那么代价函数就无法达到全局最优解，有可能是局部最优解，图 3-2 给出了非凸和凸函数的区别。

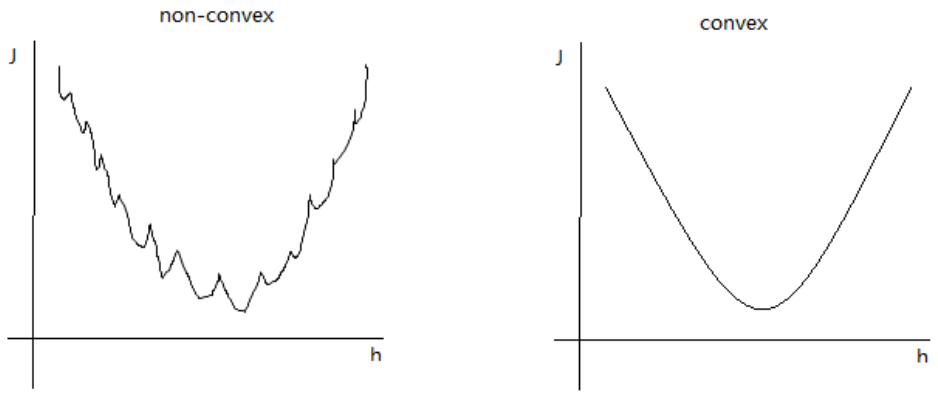

图 3-2 凸函数和非凸函数的图形比较

基于上述原因，我们使用下面的代价函数公式：

$$\text{cost}(h_\theta(\text{x}),\text{y}) = \begin{cases} -\log(h_\theta(x)), & \text{y} = 1 \\ -\log(1 - h_\theta(x)), & \text{y} = 0 \end{cases} \quad \text{（公式 3）}$$

图 3-3 为 y=1 和 y=0 时$\text{cost}(h_\theta(\text{x}),\text{y})$函数的图形。

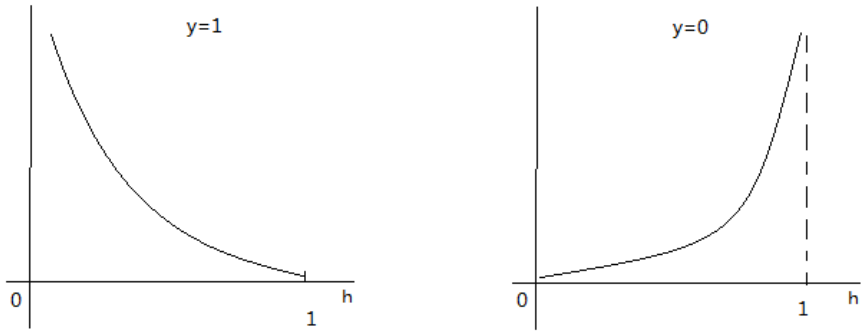

图 3-3 y=1 和 y=0 时$\text{cost}(h_\theta(\text{x}),\text{y})$函数的图形

对于 y=1 的情况，当$h_\theta(\text{x})$=1 时，cost=0；当$h_\theta(\text{x})$->0 时，cost->∞。直观的感受就是如果$h_\theta(\text{x})$=0，但实际上 y=1,此时的预测是完全错误的，所以代价是非常大的。同理对于 y=0 的情况，当$h_\theta(\text{x})$=0 时，cost=0；当$h_\theta(\text{x})$->1 时，cost->∞。

简化代价函数

由于公式 3 是两个公式，所以不适合得出统一的代价函数，故将公式 3 简化为一个统一的公式，如下：

$$\text{cost}(h_\theta(\text{x}),\text{y}) = -\text{y}\log\big(h_\theta(x)\big) - (1-\text{y})\log\big(1-h_\theta(x)\big) \qquad \text{（公式 4）}$$

因为 y 只取 0 或 1，所以公式 3 和公式 4 是完全等价的，最后可得出逻辑回归的代价函数公式：

$$\text{J}(\theta) = \frac{1}{m}\sum_{i=1}^{m}\text{cost}(h_\theta(\text{x}),\text{y})$$

$$= -\frac{1}{m}\sum_{i=1}^{m}\big(y^{(\text{i})}\log\big(h_\theta(x)\big) - (1-y^{(\text{i})})\log\big(1-h_\theta(x)\big)\big) \text{（公式 5）}$$

得到代价函数后的工作就和线性回归更加类似了，直接给出基于梯度下降的逻辑回归算法，重复执行以下公式，同时更新所有的θ_j：

$$\theta_j = \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta) \qquad \text{（公式 6）}$$

$$\theta_j = \theta_j - \frac{\alpha}{m}\sum_{i=1}^{m}\big(h_\theta(x^{(i)}) - y^{(i)}\big)x_j^{(i)} \qquad \text{（公式 7）}$$

公式 6 拆开和线性回归公式相同（公式 7），只是对于线性回归来说，$h_\theta(x)=\Theta^{\text{T}}\text{X}$，而逻辑回归$h_\theta(x) = \frac{1}{1+e^{-\Theta^T\text{x}}}$。

3.4 高级优化算法（Advanced Optimization）

相比梯度下降法，下面是一些更为高级的训练算法。

1. Conjugate gradient
2. BFGS(共轭梯度法)
3. L-BFGS(限制共轭梯度法)

使用高级的训练算法需要提供下面公式的代码：$\frac{-J(\theta)}{\frac{-\partial}{\partial\theta_j}J(\theta)}$（for j=0, 1, 2, …, n）

高级优化算法的优缺点：
优点：不用人工选择学习速率α；一般比梯度下降法要快
缺点：更加复杂

3.5 多类别分类（multiclass classification one-vs-all）

将两类问题推广到多个类别：对于每个类别 i 训练一个逻辑回归分类器$h_\theta^{(i)}(x)$，从而用来预测 y=i 的可能性，也就是说有 n 个类别就有 n 个分类器。预测时，迭代类别个数（n）次，挑选出可能性最大值（$\max_i h_\theta^{(i)}(x)$ ）对应的类别 i，新输入的（x_1, x_2）便属于该类。

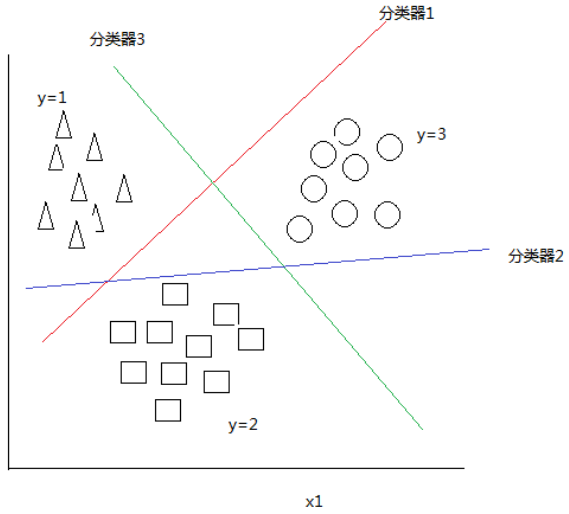

图 3-4 多类别逻辑回归分类示意图

3.6 过度拟合问题（The Problem of Overfitting）与正则化（regularization）

1.过度拟合概念：当我们使用过多的特征值时，则假设函数将含有高阶多项式，这个函数可能会很好地拟合训练集（代价值几乎为 0），甚至拟合所有的训练数据，但是不能很好地泛化到需要预测的新数据上，如图 3-5 所示，黑色线表示的是过度拟合的假设函数，虽然它很好地拟合了所有的训练数据，但是没能很好地预测新数据，过度拟合具有高方差（high variance）。相对应的有欠拟合和恰好拟合，图中的蓝色线便是欠拟合的假设函数，它既不能很好的拟合训练数据，也不能预测新数据，欠拟合具有高偏差（high bias）。我们期望得到的好的假设函数应该类似红色线表示的假设函数，基本上能拟合训练数据和预测新数据。

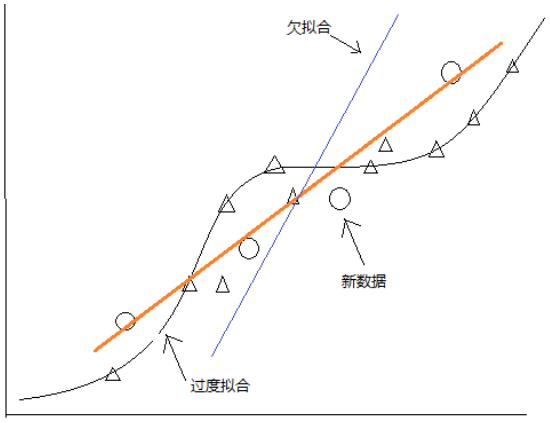

图 3-5 过度拟合、欠拟合与恰好（just right）拟合

2. 解决过度拟合——正则化

方法 1：

减少特征值数量

方法 2：

正则化，正则化具有两个比较好的特点：保持了所有的特征值，但是减少了每个特征值向量θ_j的规模（影响力）；当需要很多特征值的时候表现良好，从而使得每个特征值都对预测做出了一些贡献。所以一般会使用正则化来解决过拟合。

3. 正则化后的代价函数（cost function）

为了减少每个每个特征值向量θ_j的影响力，那么就需要在训练假设函数的代价函数中引入正则化，引入后的代价函数如公式 8。

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n}\theta_j^2\right] \quad \text{（公式 8）}$$

其中，λ为正则化参数。

4. 正则化的线性回归梯度训练算法

repeat{

$$\theta_0 := \theta_0 - \frac{\alpha}{m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i)$$

$$\theta_1 := \theta_1 - \frac{\alpha}{m}\left[\sum_{i=1}^{m}(h_\theta(x_i) - y_i)\cdot x_i^1 + \lambda\cdot\theta_1\right]$$

$$= \theta_1\left(1 - \alpha\frac{\lambda}{m}\right) - \frac{\alpha}{m}\left[\sum_{i=1}^{m}(h_\theta(x_i) - y_i)\cdot x_i^1\right]$$

…

$$\theta_n := \theta_n - \frac{\alpha}{m}\left[\sum_{i=1}^{m}(h_\theta(x_i) - y_i)\cdot x_n^i + \lambda\cdot\theta_i\right]$$

$$= \theta_n\left(1 - \alpha\frac{\lambda}{m}\right) - \frac{\alpha}{m}\left[\sum_{i=1}^{m}(h_\theta(x_i) - y_i)\cdot x_i^n\right]$$

}

其中 $1 - \alpha\frac{\lambda}{m} < 1$, 即把$\theta_j$向 0 压缩了一点，一般比 1 小一点点，如 0.99。

正则化后的正规方程：

$$\Theta = (X^T X + \lambda \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_{(n+1)*(n+1)})^{-1} X^T y \quad \text{（公式 9）}$$

正则化也解决了正规化方程中$X^T X$不可逆的问题。

5. 正则化的逻辑回归梯度训练算法

代价函数为:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \text{cost}(h_\theta(x), y)$$

$$= -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)} \log\left(h_\theta(x)\right) - (1-y^{(i)})\log\left(1-h_\theta(x)\right)\right] + \lambda \sum_{j=1}^{n} \theta_j^2 \quad \text{（公式 10）}$$

梯度算法与线性回归相同：

$$\theta_j := \theta_j - \frac{\alpha}{m}\left[\sum_{i=1}^{m}(h_\theta(x_i) - y_i) \cdot x_j^i + \lambda \cdot \theta_j\right]$$

$$= \theta_j\left(1 - \alpha\frac{\lambda}{m}\right) - \frac{\alpha}{m}\left[\sum_{i=1}^{m}(h_\theta(x_i) - y_i) \cdot x_j^i\right]$$