什么是逻辑回归？

Logistic 回归与多重线性回归实际上有很多相同之处，最大的区别就在于它们的因变量不同，其他的基本都差不多。正是因为如此，这两种回归可以归于同一个家族，即广义线性模型（generalizedlinear model）。

这一家族中的模型形式基本上都差不多，不同的就是因变量不同。

- 如果是连续的，就是多重线性回归；
- 如果是二项分布，就是 Logistic 回归；
- 如果是 Poisson 分布，就是 Poisson 回归；
- 如果是负二项分布，就是负二项回归。

Logistic 回归的因变量可以是二分类的，也可以是多分类的，但是二分类的更为常用，也更加容易解释。所以实际中最常用的就是二分类的 Logistic 回归。

Logistic 回归的主要用途：

- 寻找危险因素：寻找某一疾病的危险因素等；
- 预测：根据模型，预测在不同的自变量情况下，发生某病或某种情况的概率有多大；
- 判别：实际上跟预测有些类似，也是根据模型，判断某人属于某病或属于某种情况的概率有多大，也就是看一下这个人有多大的可能性是属于某病。

Logistic 回归主要在流行病学中应用较多，比较常用的情形是探索某疾病的危险因素，根据危险因素预测某疾病发生的概率，等等。例如，想探讨胃癌发生的危险因素，可以选择两组人群，一组是胃癌组，一组是非胃癌组，两组人群肯定有不同的体征和生活方式等。这里的因变量就是是否胃癌，即"是"或"否"，自变量就可以包括很多了，例如年龄、性别、饮食习惯、幽门螺杆菌感染等。自变量既可以是连续的，也可以是分类的。

常规步骤

Regression 问题的常规步骤为：

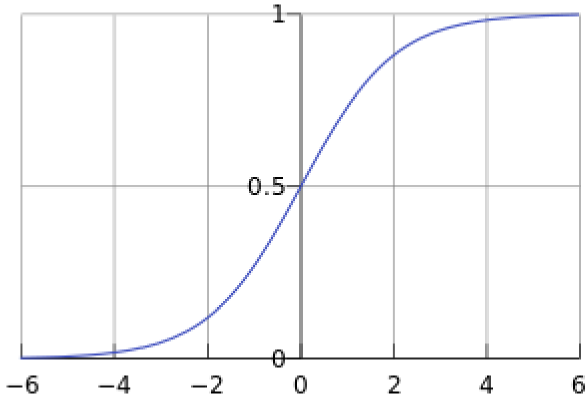
1. 寻找 h 函数（即 hypothesis）；
2. 构造 J 函数（损失函数）；
3. 想办法使得 J 函数最小并求得回归参数（θ）

构造预测函数 h

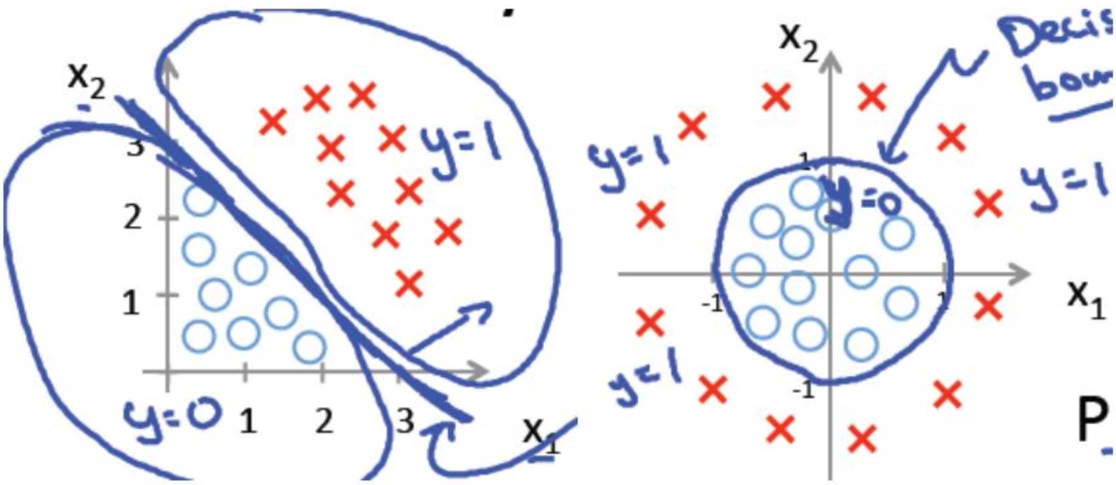
Logistic 回归虽然名字里带"回归"，但是它实际上是一种分类方法，主要用于两分类问题（即输出只有两种，分别代表两个类别），所以利用了 Logistic 函数（或称为 Sigmoid 函数），函数形式为：

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

Sigmoid 函数在有个很漂亮的"S"形，如下图所示（引自维基百科）：



下面左图是一个线性的决策边界，右图是非线性的决策边界。



对于线性边界的情况，边界形式如下：

$$\theta_0 + \theta_1 x_1 +, \ldots, + \theta_n x_n = \sum_{i=1}^{n} \theta_i x_i = \theta^T x$$

构造预测函数为：

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

函数 $h_\theta(x)$ 的值有特殊的含义，它表示结果取 1 的概率，因此对于输入 x 分类结果为类别 1 和类别 0 的概率分别为：

$$\begin{aligned} P(y=1 \mid x;\theta) &= h_\theta(x) \\ P(y=0 \mid x;\theta) &= 1 - h_\theta(x) \end{aligned} \qquad (1)$$

构造损失函数 J

Cost 函数和 J 函数如下，它们是基于最大似然估计推导得到的。

$$Cost(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) \ if \ y = 1 \\ -\log(1 - h_\theta(x)) \ if \ y = 0 \end{cases}$$

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{n} Cost(h_\theta(x_i), y_i) = -\frac{1}{m}\left[\sum_{i=1}^{n} y_i \log h_\theta(x_i) + (1 - y_i)\log(1 - h_\theta(x_i))\right]$$

下面详细说明推导的过程：

（1）式综合起来可以写成：

$$P(y \mid x;\theta) = (h_\theta(x))^{y}(1 - h_\theta(x))^{1-y}$$

取似然函数为：

$$L(\theta) = \prod_{i=1}^{m} P(y_i \mid x_i;\theta) = \prod_{i=1}^{m}(h_\theta(x_i))^{y_i}(1 - h_\theta(x_i))^{1-y_i}$$

对数似然函数为：

$$l(\theta) = \log L(\theta) = \sum_{i=1}^{m}\left(y_i \log h_\theta(x_i) + (1 - y_i)\log(1 - h_\theta(x_i))\right)$$

最大似然估计就是求使 $l(\theta)$ 取最大值时的 θ，其实这里可以使用梯度上升法求解，求得的 θ 就是要求的最佳参数。但是，在 Andrew Ng 的课程中将 $J(\theta)$ 取为下式，即：

$$J(\theta) = -\frac{1}{m} l(\theta)$$

因为乘了一个负的系数-1/m，所以取 $J(\theta)$ 最小值时的 θ 为要求的最佳参数。

梯度下降法求的最小值

θ 更新过程：

$$\theta_j := \theta_j - \alpha \frac{\delta}{\delta_{\theta_j}} J(\theta)$$

$$\frac{\delta}{\delta\theta_j}J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left(y_i\frac{1}{h_\theta(x_i)}\frac{\delta}{\delta\theta_j}h_\theta(x_i) - (1-\mathrm{y_i})\frac{1}{1-h_\theta(x_i)}\frac{\delta}{\delta\theta_j}h_\theta(x_i)\right)$$

$$= -\frac{1}{m}\sum_{i=1}^{m}\left(y_i\frac{1}{g(\theta^{\mathrm{T}}x_i)} - (1-\mathrm{y_i})\frac{1}{1-g(\theta^{\mathrm{T}}x_i)}\right)\frac{\delta}{\delta\theta_j}g(\theta^{\mathrm{T}}x_i)$$

$$= -\frac{1}{m}\sum_{i=1}^{m}\left(y_i\frac{1}{g(\theta^{\mathrm{T}}x_i)} - (1-\mathrm{y_i})\frac{1}{1-g(\theta^{\mathrm{T}}x_i)}\right)g(\theta^{\mathrm{T}}x_i)(1-g(\theta^{\mathrm{T}}x_i))\frac{\delta}{\delta\theta_j}\theta^{\mathrm{T}}x_i$$

$$= -\frac{1}{m}\sum_{i=1}^{m}\left(y_i(1-g(\theta^{\mathrm{T}}x_i)) - (1-\mathrm{y_i})g(\theta^{\mathrm{T}}x_i)\right)x_i^j$$

$$= -\frac{1}{m}\sum_{i=1}^{m}\left(y_i - g(\theta^{\mathrm{T}}x_i)\right)x_i^j$$

$$= \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x_i) - y_i\right)x_i^j$$

θ 更新过程可以写成：

$$\theta_j := \theta_j \text{ - } \alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x_i) - y_i\right)x_i^j$$

向量化 Vectorization

Vectorization 是使用矩阵计算来代替 for 循环，以简化计算过程，提高效率。

如上式，Σ(...)是一个求和的过程，显然需要一个 for 语句循环 m 次，所以根本没有完全的实现 vectorization。

下面介绍向量化的过程：

约定训练数据的矩阵形式如下，x 的每一行为一条训练样本，而每一列为不同的特称取值：

$$x=\begin{bmatrix}x_1\\ \dots\\ x_m\end{bmatrix}=\begin{bmatrix}x_{11} & \cdots & x_{1n}\\ \vdots & \ddots & \vdots\\ x_{m1} & \cdots & x_{mn}\end{bmatrix},y=\begin{bmatrix}y_1\\ \dots\\ y_m\end{bmatrix},\theta=\begin{bmatrix}\theta_0\\ \dots\\ \theta_n\end{bmatrix}$$

$$A=x\bullet\theta=\begin{bmatrix}x_{10} & \cdots & x_{1n}\\ \vdots & \ddots & \vdots\\ x_{m1} & \cdots & x_{mn}\end{bmatrix}\bullet\begin{bmatrix}\theta_0\\ \dots\\ \theta_n\end{bmatrix}=\begin{bmatrix}\theta_0x_{10}+\theta_1x_{11}+\dots+\theta_0x_{1n}\\ \dots\\ \theta_0x_{m0}+\theta_1x_{m1}+\dots+\theta_0x_{mn}\end{bmatrix}$$

$$E=h_\theta(x)-y=\begin{bmatrix}g(A_1)-y_1\\ \dots\\ g(A_m)-y_m\end{bmatrix}=\begin{bmatrix}e_1\\ \dots\\ e_m\end{bmatrix}=g(A)-y$$

g(A)的参数 A 为一列向量，所以实现 g 函数时要支持列向量作为参数，并返回列向量。由上式可知

$h\theta(x)-y$ 可由 $g(A)-y$ 一次计算求得。

θ 更新过程可以改为：

$$\theta_j:=\theta_j-\alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x_i)-y_i)x_i^j=\theta_j-\alpha\frac{1}{m}\sum_{i=1}^{m}e_ix_i^j=\theta_j-\alpha\frac{1}{m}x^TE$$

综上所述，Vectorization 后 θ 更新的步骤如下：

（1）求 $A=x\bullet\theta$；

（2）求 $E=g(A)-y$；

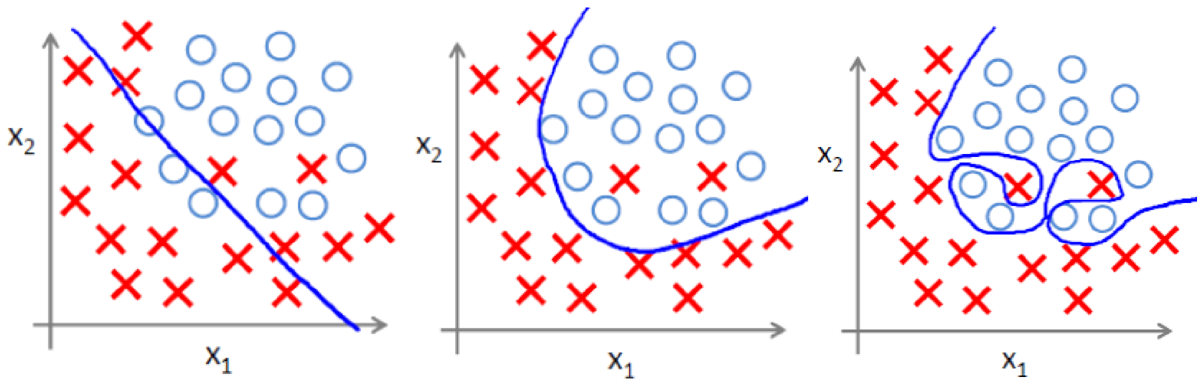
（3）求 $\theta:=\theta-\alpha x^TE$。

正则化 Regularization

过拟合问题

对于线性回归或逻辑回归的损失函数构成的模型，可能会有些权重很大，有些权重很小，导致过拟合（就是过分拟合了训练数据），使得模型的复杂度提高，泛化能力较差（对未知数据的预测能力）。

下面左图即为欠拟合，中图为合适的拟合，右图为过拟合。



问题的主因

过拟合问题往往源自过多的特征。

解决方法

1）减少特征数量（减少特征会失去一些信息，即使特征选的很好）

- 可用人工选择要保留的特征；
- 模型选择算法；

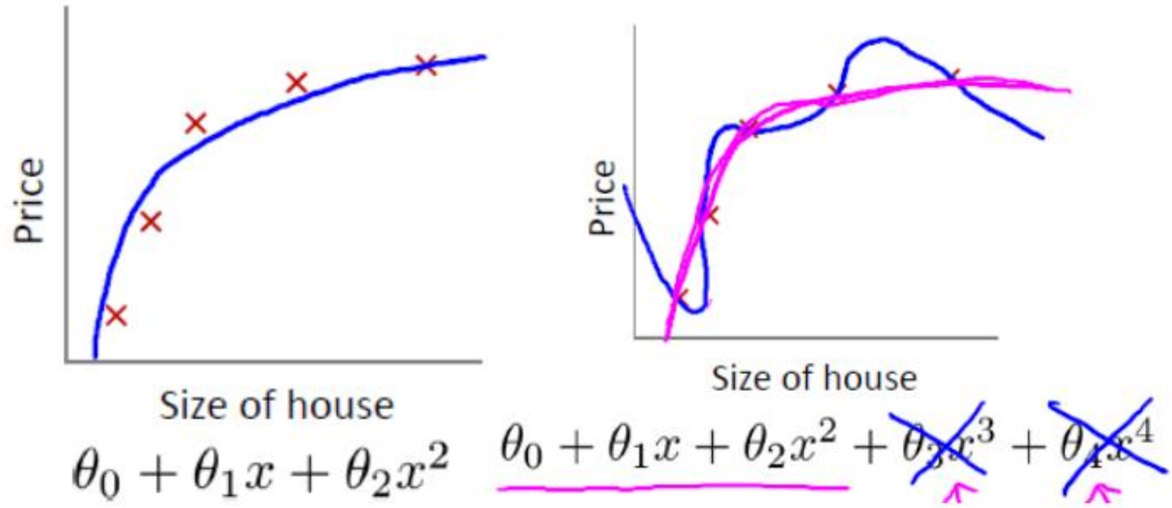
2）正则化（特征较多时比较有效）

- 保留所有特征，但减少 θ 的大小

正则化方法

正则化是结构风险最小化策略的实现，是在经验风险上加一个正则化项或惩罚项。正则化项一般是模型复杂度的单调递增函数，模型越复杂，正则化项就越大。

从房价预测问题开始，这次采用的是多项式回归。左图是适当拟合，右图是过拟合。



直观来看，如果我们想解决这个例子中的过拟合问题，最好能将 x^3, x^4 的影响消除，也就是让 $\theta 3 \approx 0, \theta 4 \approx 0$。假设我们对 θ_3, θ_4 进行惩罚，并且令其很小，一个简单的办法就是给原有的 Cost 函数加上两个略大惩罚项，例如：

$$\min_{\theta} \frac{1}{2m} \sum_{i=1}^{n} (h_\theta(x_i) - y_i)^2 + 1000\theta_3^2 + 1000\theta_4^2$$

这样在最小化 Cost 函数的时候，$\theta 3 \approx 0, \theta 4 \approx 0$。

正则项可以取不同的形式，在回归问题中取平方损失，就是参数的 L2 范数，也可以取 L1 范数。取平方损失时，模型的损失函数变为：

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{n} (\mathrm{h}_\theta(\mathrm{x}_i) - y_i)^2 + \lambda \sum_{j=1}^{n} \theta_j^2$$

lambda 是正则项系数：

- 如果它的值很大，说明对模型的复杂度惩罚大，对拟合数据的损失惩罚小，这样它就不会过分拟合数据，在训练数据上的偏差较大，在未知数据上的方差较小，但是可能出现欠拟合的现象；
- 如果它的值很小，说明比较注重对训练数据的拟合，在训练数据上的偏差会小，但是可能会导致过拟合。

正则化后的梯度下降算法 θ 的更新变为：

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

正则化后的线性回归的 Normal Equation 的公式为：

$$\theta = \left(X^T X + \lambda \begin{bmatrix} 0 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix} \right)^{-1} X^T Y$$

其他优化算法

- Conjugate gradient method(共轭梯度法)
- Quasi-Newton method(拟牛顿法)
- BFGS method
- L-BFGS(Limited-memory BFGS)

后二者由拟牛顿法引申出来，与梯度下降算法相比，这些算法的优点是：

- 第一，不需要手动的选择步长；
- 第二，通常比梯度下降算法快；

但是缺点是更复杂。

多类分类问题

对于多类分类问题，可以将其看做成二类分类问题：保留其中的一类，剩下的作为另一类。

对于每一个类 i 训练一个逻辑回归模型的分类器 $h_\theta^{(i)}(x)$，并且预测 y = i 时的概率；对于一个新的输入变量 x，分别对每一个类进行预测，取概率最大的那个类作为分类结果：

$$\max_i h_\theta^{(i)}(x)$$