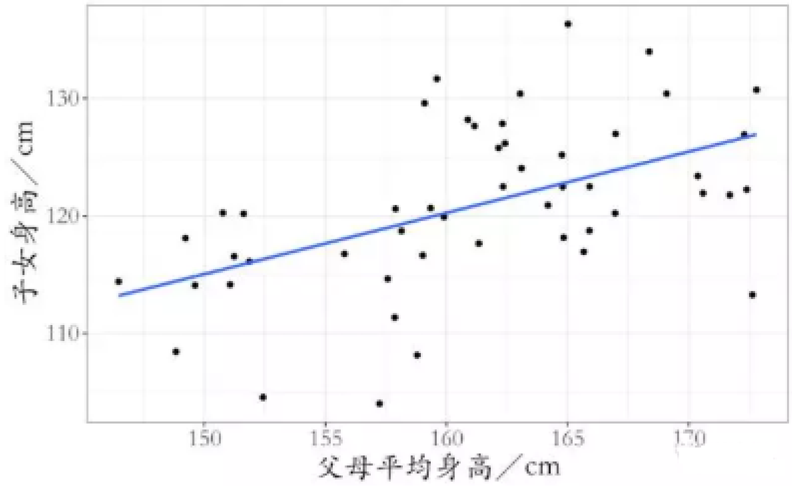
逻辑回归

线性回归、逻辑回归直观感受

我们以身高来举例，直觉告诉我们爸爸妈妈的身高会共同影响子女的身高，为了同时考虑到父母双方的身高的影响，可以取其两者的平均值作为因素进行研究，这里父母的平均身高就是自变量 x ，而我们的身高就是因变量 y ，y 和 x 之间存在线性关系：

$$y = wx + b$$

那我们怎么求出上面的参数w和b呢，就是需要我们收集足够多的 x,y ，然后通过线性回归算法就可以拟合数据帮我们求出参数 w 和 b 。

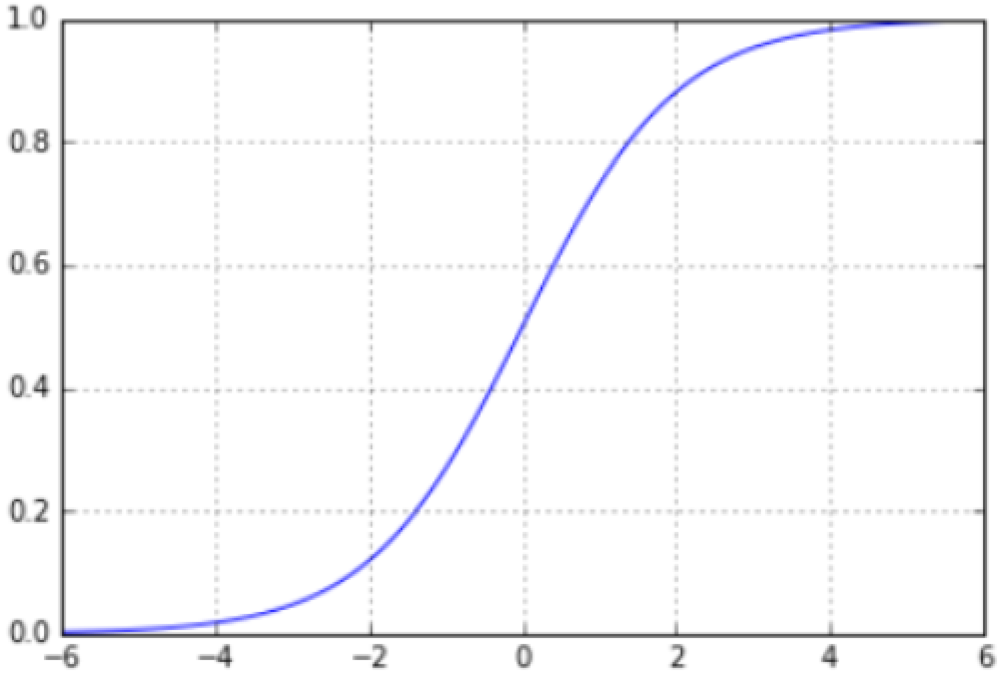


虽然线性回归模型在自变量的种类上面已经没有限制了，因变量只能是连续的数值却是一个很大的制约因素，因为在实际应用中，因变量是分类变量的情形太普遍了。分类变量中最简单、也最常用的情形是二元变量 （binary variable），比如明天会不会下雨，就是二元变量，这正是逻辑回归要解决的问题。

直观上，你可能会这么想，只要把线性回归的连续值想方设法地转换成 0 到 1 之间的数值，这不就变成了逻辑回归了吗，恩，你猜想得没错，有个叫逻辑函数就是做这个事情，如下：

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

这个 g 函数将实数域数据转换到 0，1之间，函数图片如下：



通过这个转换，我们就具备了预测二元变量的能力了。

线性回归推导

极大似然估计

这里我们先介绍一个概念，**极大似然估计**，在机器学习中，这个概念是绕不过去的。

假设一个黑袋子里面有一堆球，有黑球和白球，但是我们不知道具体的分布情况。我们从里面抓 3 个球，2 个黑球，1 个白球。这时候，有人就直接得出了黑球 67%，白球占比 33% 。这个时候，其实这个人使用了**极大似然估计**的思想，通俗来讲，当黑球是 67% 的占比的时候，我们抓 3 个球，出现 2 黑 1 白的概率最大。

这种通过样本，反过来猜测总体的情况，就是**似然**。

再举个例子，有一枚硬币，一般我们认为他出现正面和反面的**概率**是相同的，都是 0.5 。你为了验证这一想法，你抛了 100 次，100 次出现的都是正面，在这样的事实下，我觉得似乎硬币的参数不是公平的，这时候，你修正你的看法，觉得硬币出现正面的概率是 1 ，而出现反而的概率是 0 ，这就是**极大似然估计**，按这个估计，出现 100 次都是正面的概率才最大。

同样，直观感受完之后，我们给出一个比较严谨的定义：

设总体分布为 $f(x,\theta)$，$X_1, X_2, \cdots, X_n$ 为该总体采样得到的样本。因为 $X_1, X_2, \cdots, X_n$ 独立同分布，因此，对于联合密度函数：

$L(X_1, X_2, \cdots, X_n; \theta_1, \theta_2, \cdots, \theta_k) = \prod_{i=1}^{n} f(X_i; \theta_1, \theta_2, \cdots, \theta_k)$

上式中，如果 θ 知道，我们就可以直接求出 L 了，总体分布的参数我们不是上帝，是没办法知道的。所以，我们换一种思路，反过来，因为样本已经存在，可以看成 $X_1, X_2, \cdots, X_n$是固定的，L是关于 θ 的函数，即**似然函数**。求 θ 的值，使得似然函数取极大值，这种方法就是**极大似然估计**。

这说的是同一回事，都是说用样本去估计总体的参数值，这个参数值使得样本出现的概率最大。

线性回归的模型如下：

$h_\theta(x_1, x_2, \cdots, x_n) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n = \sum_{i=0}^{n} \theta_i x_i$

上式中，$x_0 = 1$

使用矩阵表示为：

$h_\theta(X) = \theta^T X$

上式中，X 为 $m \times n$ 维矩阵，m 为样本个数，n 为样本特征数。

我们知道，样本基本是在所求线性回归 $h_\theta(X) = \theta^T X$ 的附近，之间有会一个上下浮动的误差，记为 ε，表示为：

$Y = \theta^T X + \varepsilon$

上式中，ε 是 $m \times 1$ 维向量，代表 m 个样本相对于线性回归方程的上下浮动程度。ε 是独立同分布的，由中心极限定理，ε分布服从均值为 0，方差为σ^2 的正态分布。

最小二乘法推导

结合上面的公式，对每个样本来说，有：

$\varepsilon^{(j)} = y^{(j)} - \theta^T x^{(j)}$

上式中，$j \in (1, 2, \cdots, m)$

ε 分布服从均值为 0，方差为 σ^2 的正态分布，所以：

$f(\varepsilon^{(j)}) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(\varepsilon^{(j)})^2}{2\sigma^2}}$

将$\varepsilon^{(j)} = y^{(j)} - \theta^T x^{(j)}$ 代入上式，有：

$f(y^{(j)} | x^{(j)}; \theta) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y^{(j)} - \theta^T x^{(j)})^2}{2\sigma^2}}$

下面的公式推导用到了如下对数转换公式：

$\log a + \log b = \log ab$

$\log ab = \log a + \log b$

似然函数：

$L(\theta) = \prod_{j=1}^{m} f(y^{(j)} | x^{(j)}; \theta) = \prod_{j=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y^{(j)} - \theta^T x^{(j)})^2}{2\sigma^2}}$

两边取对数，令$l(\theta) = logL(\theta)$：

$l(\theta) = \log \prod_{j=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y^{(j)} - \theta^T x^{(j)})^2}{2\sigma^2}}$ $l(\theta) = \sum_{j=1}^{m} \log \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y^{(j)} - \theta^T x^{(j)})^2}{2\sigma^2}}$
$l(\theta) = m \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{\sigma^2} \cdot \frac{1}{2} \sum_{j=1}^{m} (y^{(j)} - \theta^T x^{(j)})^2$

上式中，去掉常数项，去掉负号，即将求极大似然函数最大值转换为求成本函数最小值：

$J(\theta) = \frac{1}{2} \sum_{j=1}^{m} (y^{(j)} - \theta^T x^{(j)})^2$

即：

$J(\theta) = \frac{1}{2} \sum_{j=1}^{m} (y^{(j)} - h_\theta(x^{(j)}))^2$

到这里，是不是看到经常看到的最小二乘法的味道来了，没错，上式中 $y^{(j)}$ 表示样本实际值，$h_\theta(x^{(j)})$表示线性回归预测值，我们的目的就是求这两个值的差的平方的最小值，这就是最小二乘法的由来。

下面我们就看怎么求解上式中的参数θ 。

我们先将上式改为使用矩阵表示：

$J(\theta)=\frac{1}{2}(X\theta-Y)^T(X\theta-Y)$

$J(\theta)=\frac{1}{2}(\theta^TX^T-Y^T)(X\theta-Y)$

$J(\theta)=\frac{1}{2}(\theta^TX^TX\theta-(\theta^TX^TY-Y^TX\theta+Y^TY)$

对上式求导数：

$\nabla J(\theta)=\frac{1}{2}(2X^TX\theta-X^TY-(Y^TX)^T)$

$\nabla J(\theta)=\frac{1}{2}(2X^TX\theta-X^TY-X^TY)$

$\nabla J(\theta)=\frac{1}{2}(2X^TX\theta-2X^TY)$

$\nabla J(\theta)=X^TX\theta-X^TY$

令上式 =0，即：

$\nabla J(\theta)=X^TX\theta-X^TY=0$

可求得：

$\theta=(X^TX)^{-1}X^TY$

这就是最小二乘法的解法，一步到位，都不用机器学习，直接求解出来。这是一大优势，但可想而知，天下没有免费的午餐，它肯定存在一些劣势，比如：

> 1. 当特征量很大时，最小二乘法计算量太大，计算时间无法忍受或直接算力不足。当特征量小于 1 万时，可以考虑使用最小二乘法，大于 1 万时，还是使用梯度下降法。
> 2. 最小二乘法只适应于线性回归。
> 3. X^TX 不一定都存在逆矩阵。不可逆其实很少发生。有两种不可逆的情况： a. X 里面的 x_1 和 x_2 存在线性关系，比如 $x_1=3.28x_2$ b. m <= n ，这种情况可以用正则化处理，使之可逆，即：
> $\theta=(X^TX+\lambda I)^{-1}X^TY$

梯度下降法推导

下面我们使用梯度下降法来求解线性回归的参数 θ 。上面已经提到成本函数为：

$J(\theta)=\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2$

对 θ 求偏导：

$\frac{\partial}{\partial\theta_j}J(\theta)=\frac{\partial}{\partial\theta_j}\sum_{i=1}^{m}\frac{1}{2m}(h_\theta(x^{(i)})-y^{(i)})^2$

$=\frac{1}{2m}\cdot2\cdot\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})\frac{\partial}{\partial\theta_j}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})$

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

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

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

加入学习率，我们的梯度下降算法可以描述为：

> 重复 $\theta * j = \theta * j - \alpha \frac{1}{m} \sum *i = 1^m (h * \theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$
>
> (更新 θ_j , $j = 0, 1, \cdots, n)$ }

逻辑回归推导

逻辑回归成本函数

逻辑回归的模型如下:

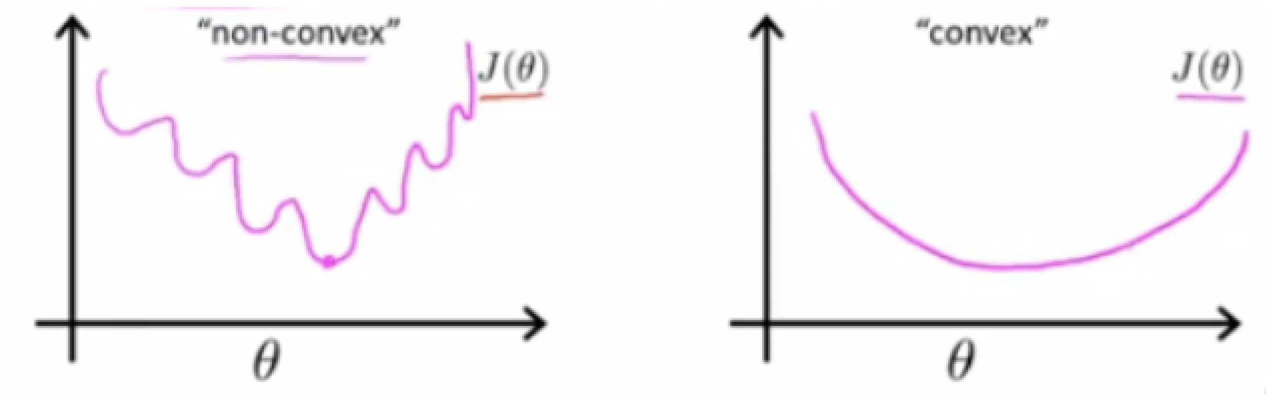
$h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$

$log \frac{p(y=1)}{p(y=0)} = \theta^T x$

那么，它的成本函数能不能像线性回归那样使用平方函数呢，即:

$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$

与线性回归相比，这里面的 $h_\theta(x)$一样了，这个J函数将是个非凸函数，没办法得到全局最优解。



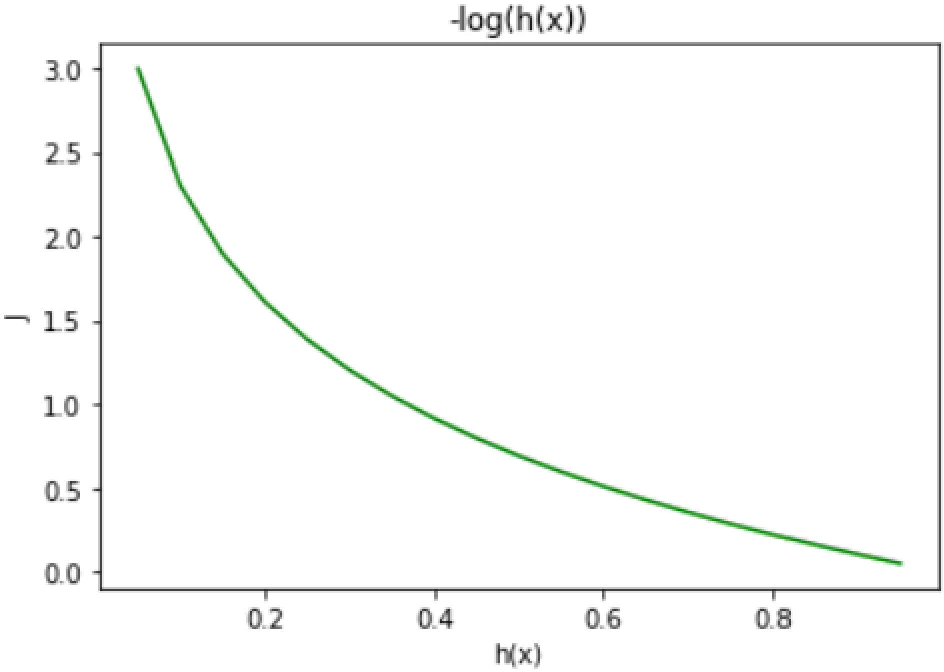
如上图所示，左边的是非凸函数，右边的是凸函数。

那么，我们就另想个方式来定义它的成本函数。

当$y = 1$ 时，我们这样定义它的成本函数:

$J(\theta) = -\log(h_\theta(x))$

它的函数曲线如下:

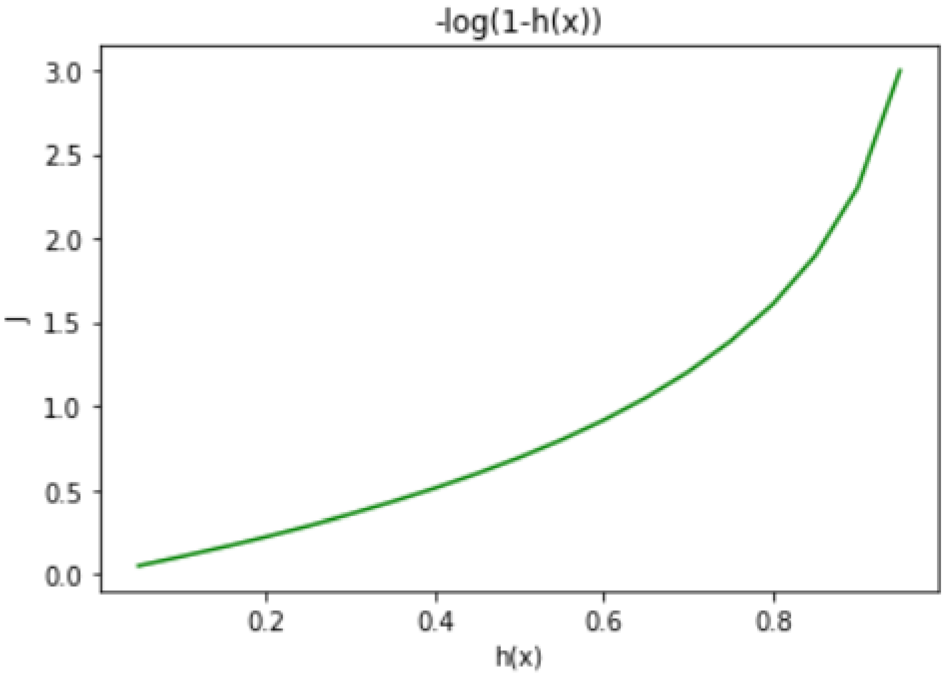


就是预测值$h_\theta(x)$ 越靠近 1，$J(\theta)$就越接近 0 ，因为预测越准确，代价就越小。

当 y = 0 时，我们这样定义它的成本函数：

$J(\theta) = -\log(1 - h_\theta(x))$

它的函数曲线如下：



就是预测值 $h_\theta(x)$越靠近 0 ，$J(\theta)$就越接近 0 ，因为预测越准确，代价就越小。

好了，到这里，你应该很清楚，我们想把上面那两种情况结合起来，写成一个统一的公式：

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

你自己试试看，分别让 y 取 1 和 0 ，是不是能得到上面那两种情况。

逻辑回归梯度下降法推导

现在，我们可以使用梯度下降来求解参数了，对上式求偏导。

由 $(\log x)^{'}=\frac{1}{x}$，得:

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

$=-\frac{1}{m}\sum_{i=1}^{m}(y^{(i)}\frac{1}{g(\theta^T x)}-(1-y^{(i)})\frac{1}{1-g(\theta^T x)})\frac{\partial g(\theta^T x)}{\partial\theta_j}$

由:

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

$g^{'}(x)=g(x)(1-g(x))$

得:

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

$=-\frac{1}{m}\sum_{i=1}^{m}(y^{(i)}(1-g(\theta^T x^{(i)}))-(1-y^{(i)})g(\theta^T x^{(i)}))\cdot x_j^{(i)}$

$=-\frac{1}{m}\sum_{i=1}^{m}(y^{(i)}-g(\theta^T x^{(i)}))\cdot x_j^{(i)}$

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

加入学习率，我们的梯度下降算法可以描述为：

> 重复$\theta_j=\theta_j-\alpha\frac{1}{m}\sum_{i=>1}^{m}(h_\theta(x^{(i)})-y^{(i)})x_j^{(i)}(\text{更新}\ \theta_j\text{，}j=0,1,\cdots,n)$

你会发现，这个跟线性回归模型的梯度下降表达上一模一样，但是，你要知道，其中的 $h_\theta(x)$ 是不一样的。