前言

逻辑回归是分类当中极为常用的手段，因此，掌握其内在原理是非常必要的。我会争取在本文中尽可能简明地展现逻辑回归(logistic regression)的整个推导过程。

什么是逻辑回归

逻辑回归在某些书中也被称为对数几率回归，明明被叫做回归，却用在了分类问题上，我个人认为这是因为逻辑回归用了和回归类似的方法来解决了分类问题。

假设有一个二分类问题，输出为$y \in \{0,1\}$，而线性回归模型产生的预测值为$z=w^Tx+b$是实数值，我们希望有一个理想的阶跃函数来帮我们实现z值到0/1值的转化。

$$\phi(z)=\begin{cases}0 & \text{if } z<0\\ 0.5 & \text{if } z=0\\ 1 & \text{if } z>0\end{cases}$$

然而该函数不连续，我们希望有一个单调可微的函数来供我们使用，于是便找到了Sigmoidfunction来替代。

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

两者的图像如下图所示（图片出自文献2）

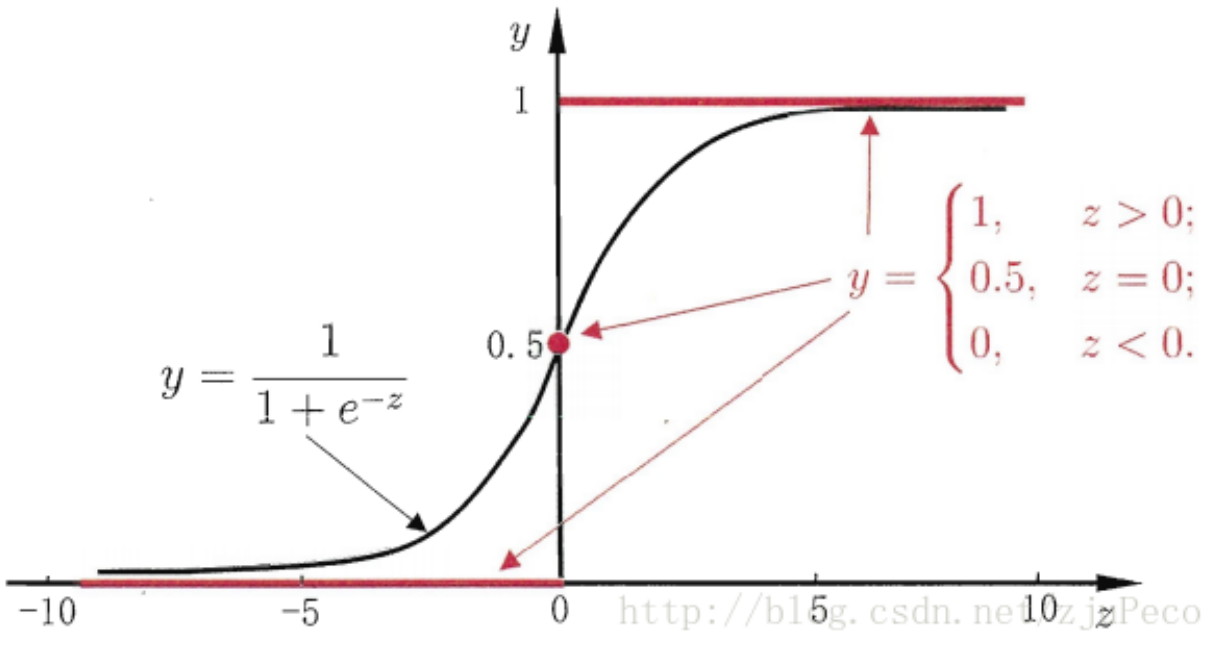

图1：sigmoid & step function

有了Sigmoidfuction之后，由于其取值在[0,1]，我们就可以将其视为类1的后验概率估计p(y=1|x)。说白了，就是如果有了一个测试点x，那么就可以用Sigmoidfuction算出来的结果来当做该点x属于类别1的概率大小。

于是，非常自然地，我们把Sigmoidfuction计算得到的值大于等于0.5的归为类别1，小于0.5的归为类别0。

$\hat{y}=\{1 \text{ if } \phi(z) \geq 0.5 \text{ otherwise}$

同时逻辑回归与自适应线性网络非常相似，两者的区别在于逻辑回归的激活函数是Sigmoidfunction而自适应线性网络的激活函数是y=x，两者的网络结构如下图所示（图片出自文献1）。

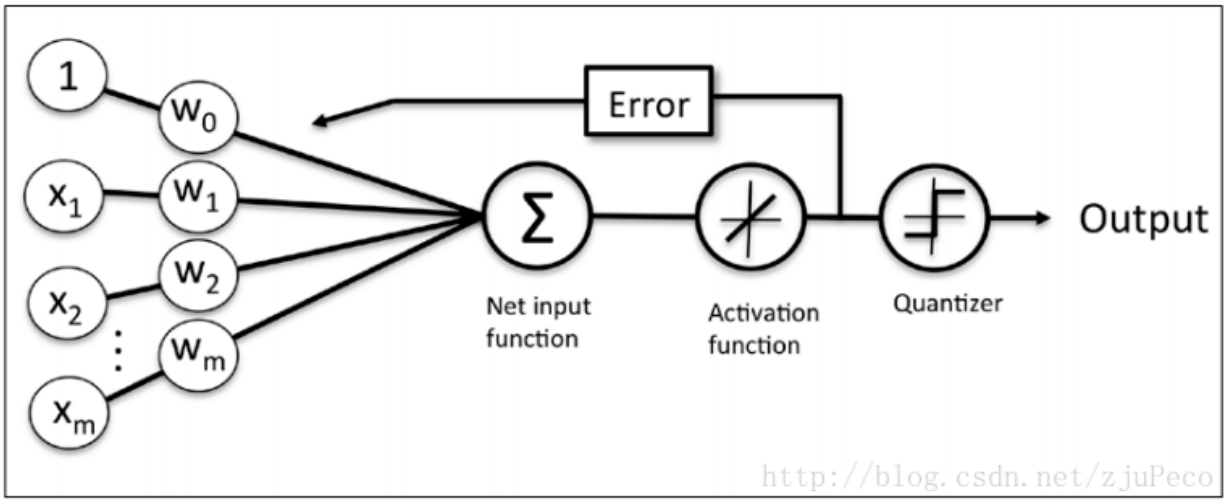

图2：自适应线性网络

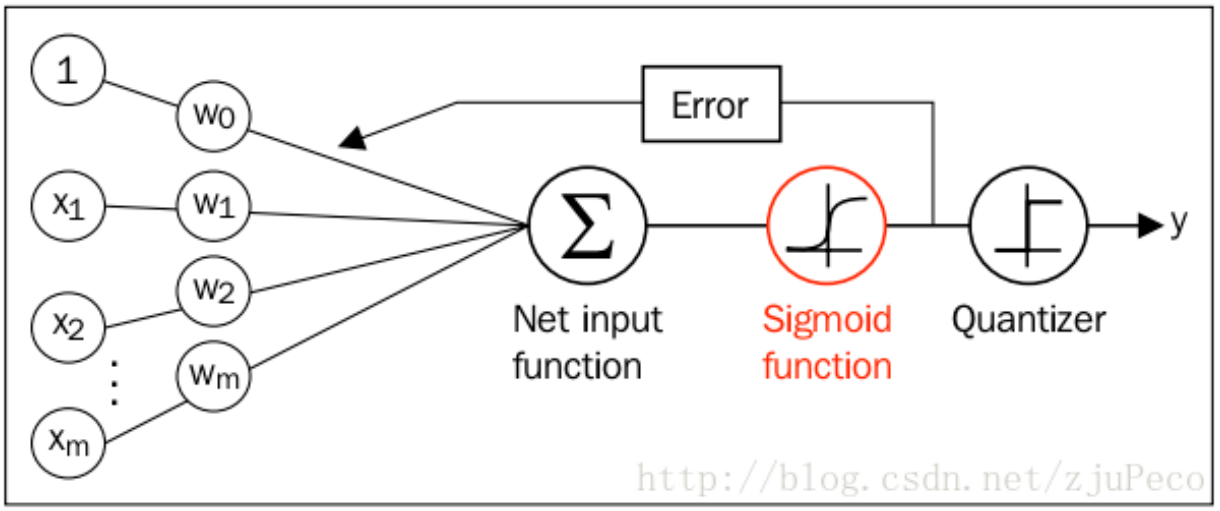

图3：逻辑回归网络

逻辑回归的代价函数

好了，所要用的几个函数我们都有了，接下来要做的就是根据给定的训练集，把参数w给求出来了。要找参数w，首先就是得把代价函数（cost function）给定义出来，也就是目标函数。

我们第一个想到的自然是模仿线性回归的做法，利用误差平方和来当代价函数。

$$J(w)=\sum_i \frac{1}{2}(\phi(z^{(i)})-y^{(i)})^2$$

其中，$z^{(i)}=w^Tx^{(i)}+b$，i表示第i个样本点，$y^{(i)}$表示第i个样本的真实值，$\phi(z^{(i)})$表示第i个样本的预测值。
这时，如果我们将$\phi(z^{(i)})=\frac{1}{1+e^{-z^{(i)}}}$代入的话，会发现这是一个非凸函数，这就意味着代价函数有着许多的局部最小值，这不利于我们的求解。

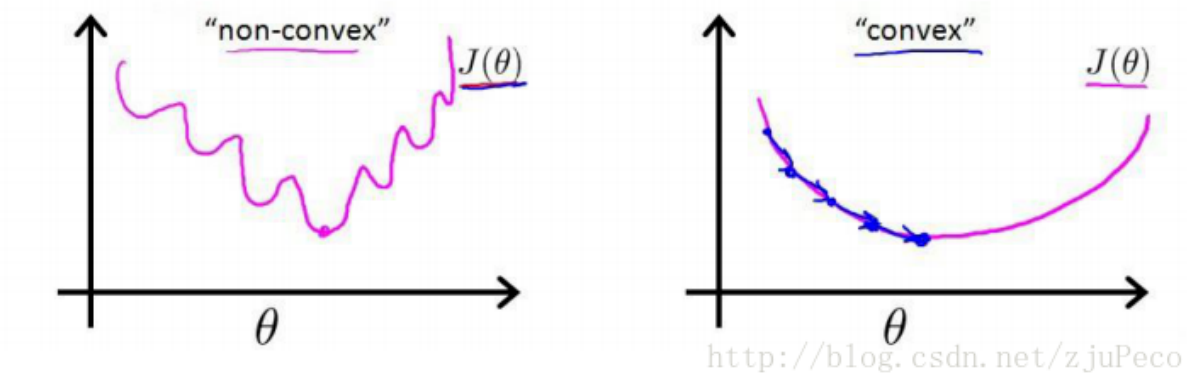

图4：凸函数和非凸函数

那么我们不妨来换一个思路解决这个问题。前面，我们提到了$\phi(z)$可以视为类1的后验估计，所以我们有

$$p(y=1|x;w)=\phi(w^Tx+b)=\phi(z)$$

$$p(y=0|x;w)=1-\phi(z)$$

其中，$p(y=1|x;w)$表示给定w，那么x点y=1的概率大小。
上面两式可以写成一般形式

$$p(y|x;w)=\phi(z)^{y}(1-\phi(z))^{(1-y)}$$

接下来我们就要用极大似然估计来根据给定的训练集估计出参数w。

$$L(w)=\prod_{i=1}^{n}p(y^{(i)}|x^{(i)};w)=\prod_{i=1}^{n}(\phi(z^{(i)}))^{y^{(i)}}(1-\phi(z^{(i)}))^{1-y^{(i)}}$$

为了简化运算，我们对上面这个等式的两边都取一个对数

$$l(w)=\ln L(w)=\sum_{i=1}^{n}y^{(i)}\ln(\phi(z^{(i)}))+(1-y^{(i)})\ln(1-\phi(z^{(i)}))$$

我们现在要求的是使得$l(w)$最大的w。没错，我们的代价函数出现了，我们在$l(w)$前面加个负号不就变成就最小了吗？不就变成我们代价函数了吗？

$$J(w)=-l(w)=-\sum_{i=1}^{n}y^{(i)}\ln(\phi(z^{(i)}))+(1-y^{(i)})\ln(1-\phi(z^{(i)}))$$

为了更好地理解这个代价函数，我们不妨拿一个例子的来看看

$$J(\phi(z),y;w)=-y\ln(\phi(z))-(1-y)\ln(1-\phi(z))$$

也就是说

$$J(\phi(z),y;w)=\begin{cases}-\ln(\phi(z)) & \text{if } y=1\\ -\ln(1-\phi(z)) & \text{if } y=0\end{cases}$$

我们来看看这是一个怎么样的函数

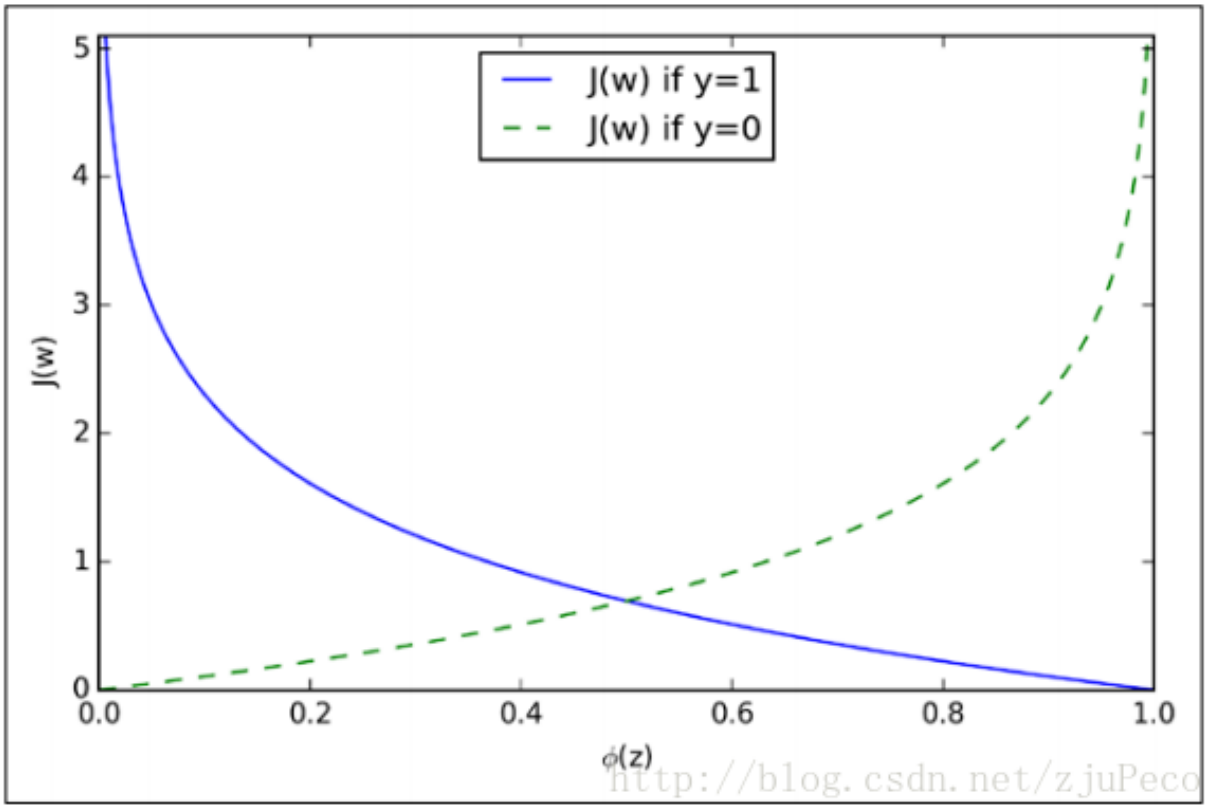

图5：代价函数

从图中不难看出，如果样本的值是1的话，估计值$\phi(z)$越接近1付出的代价就越小，反之越大；同理，如果样本的值是0的话，估计值$\phi(z)$越接近0付出的代价就越小，反之越大。

利用梯度下降法求参数

在开始梯度下降之前，要这里插一句，sigmoidfunction有一个很好的性质就是

$$\phi'(z)=\phi(z)(1-\phi(z))$$

下面会用到这个性质。
还有，我们要明确一点，梯度的负方向就是代价函数下降最快的方向。什么？为什么？好，我来说明一下。借助于泰特展开，我们有
$$f(x+\delta)-f(x)\approx f'(x)\cdot\delta$$

其中，$f'(x)$和δ为向量，那么这两者的内积就等于
$$f'(x)\cdot\delta=||f'(x)||\cdot||\delta||\cdot\cos\theta$$

当$\theta=\pi$时，也就是δ在$f'(x)$的负方向上时，取得最小值，也就是下降的最快的方向了~
okay？好，坐稳了，我们要开始下降了。
$$w:=w+\Delta w,\ \Delta w=-\eta\nabla J(w)$$

没错，就是这么下降。没反应过来？那我再写详细一些
$$w_j:=w_j+\Delta w_j,\ \Delta w_j=-\eta\frac{\partial J(w)}{\partial w_j}$$

其中，w_j表示第j个特征的权重；η为学习率，用来控制步长。
重点来了。
$$\frac{\partial J(w)}{w_j}=-\sum_{ni=1}(y^{(i)}1\phi(z^{(i)})$$
$$-(1-y^{(i)})11-\phi(z^{(i)}))\partial\phi(z^{(i)})\partial w_j=-\sum_{ni=1}(y^{(i)}1\phi(z^{(i)})$$
$$-(1-y^{(i)})11-\phi(z^{(i)}))\phi(z^{(i)})(1-\phi(z^{(i)}))\partial z^{(i)}\partial w_j=-\sum_{ni=1}(y^{(i)}$$
$$(1-\phi(z^{(i)}))-(1-y^{(i)})\phi(z^{(i)}))x^{(i)}{}_j=-\sum_{ni=1}(y^{(i)}-\phi(z^{(i)}))x^{(i)}{}_j$$

所以，在使用梯度下降法更新权重时，只要根据下式即可

$w_j := w_j + \eta \sum_{i=1}^{n} (y^{(i)} - \phi(z^{(i)})) x_j^{(i)}$

此式与线性回归时更新权重用的式子极为相似，也许这也是逻辑回归要在后面加上回归两个字的原因吧。
当然，在样本量极大的时候，每次更新权重会非常耗费时间，这时可以采用随机梯度下降法，这时每次迭代时需要将样本重新打乱，然后用下式不断更新权重。
$w_j := w_j + \eta (y^{(i)} - \phi(z^{(i)})) x_j^{(i)}, \text{for i in range(n)}$

也就是去掉了求和，而是针对每个样本点都进行更新。

结束语

以上就是我参考了基本书中的说法之后对逻辑回归整个推到过程的梳理，也不知道讲清楚没有。

如有不足，还请指正~

参考文献

[1] Raschka S. Python Machine Learning[M]. Packt Publishing, 2015.

[2] 周志华. 机器学习 : = Machine learning[M]. 清华大学出版社, 2016.