

2 Logistic回归

Logistic回归的起源：<http://blog.csdn.net/kejiaming/article/details/64439664>

Logistic回归是一个用在分类任务的线性分类器。Logistic回归也是（深度）神经网络的基础，可以看作是只包含输入层和输出层的两层网络。同第一周课程中线性回归类似，我们也从经验风险最小、正则、优化、模型评估和模型选择等方面展开讨论。如果大家能参照课件，将其中的公式推导一遍，则圆满完成任务。

Logistic回归（LR）虽然这个算法从名字上来看是回归算法，但其实际上是一个分类算法，亦被称为 logit regression, maximum-entropy classification (MaxEnt)。在机器学习分类算法中，LR回归是其中最常用的一个。线性回归（Linear Regression）也简写为LR，在不引起歧义的情况下，二者我们都用LR缩写，在有歧义的情况下用全称。

LR回归是在线性回归模型的基础上，使用sigmoid函数（logistic分布的累计分布函数），将线性模型 $\mathbf{w}^T \mathbf{x}$ 的输出压缩到[0,1]之间，使其能表示概率。LR本质仍然是一个线性模型，实现相对简单。在广告计算和推荐系统中使用频率极高，是点击率（CTR）预估模型的基本算法（在CTR部分会再介绍LR的在线学习方法）。同时，LR模型也是深度学习的基本组成单元（两层网络就是LR）。

LR回归属于概率性判别式模型。之所以是概率性模型，是因为LR模型是有概率意义的（LR可以得到后验概率 $p(y|\mathbf{x})$ ；而非概率模型如SVM，模型本身并没有概率意义）；之所以是判别式模型，是因为LR回归并没有对数据的分布 $p(\mathbf{x}, y)$ 进行建模，也就是说，LR模型并不知道数据的具体分布，而是直接将判别函数（这里是后验概率），或者说是分类超平面求解了出来。

判别式模型 vs. 产生式模型

有些分类分类算法通过求解 $p(y = c|\mathbf{x})$ 实现分类，即对于一个新的样本 \mathbf{x} ，计算其条件概率 $p(y = c|\mathbf{x})$ ，即后验概率。

后验概率可以基于贝叶斯公式得到 $p(y = c|\mathbf{x}) = \frac{p(y=c)p(\mathbf{x}|y=c)}{\sum_{c=1}^C p(y=c')p(\mathbf{x}|y=c')}$ ： 其中 $p(\mathbf{x}|y = c)$ 是类

条件概率密度， $p(y = c)$ 是类的先验概率。若采用这种方法的模型，称为是产生式模型。之所以被称为产生式模型，是因为在产生式模型中有 $p(\mathbf{x}|y = c)$ 和 $p(y = c)$ ，可得到数据的分布 $p(\mathbf{x}, y = c) = p(y = c)p(\mathbf{x}|y = c)$ ，从而可以从分布中产生数据 $p(\mathbf{x}, y = c)$ （如随机采样）。

分类算法也可以直接对后验概率进行建模，如LR模型中我们假设 $p(y = 1|\mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x})$ ，而无需知道类先验概率和类条件概率。若采用这种方法的模型，称为是判别式模型。

因为有了后验概率后，分类算法可以根据最大后验概率，将输入空间划分成许多不相交的区域，这些区域之间的分隔面被称为判别函数（也称为分类面），有了判别函数，就可以进行分类了。

判别式模型直接对后验概率进行建模，从而得到判别函数。产生式模型，最终也是为了得到判别函数。还有一些模型（如SVM），直接对判别函数进行求解，得到判别面，也被称为判别式法。

2.1 Logistic 分布

LR回归是在线性回归模型的基础上，再用sigmoid函数得到概率。这里就先介绍一下sigmoid函数。

首先，需要对 logistic分布进行说明，这个分布的概率密度函数 (probability density function, pdf) 为：

$$p(x; \mu, s) = \frac{e^{-(x-\mu)/s}}{s(1 + e^{-(x-\mu)/s})}$$

累积分布函数 (cumulative distribution function, CDF) 为：

$$F(x; \mu, s) = \frac{1}{1 + e^{-(x-\mu)/s}}$$

其 μ 位置参数， s 是形状参数。

下图为不同的 μ 和 s 的情况下，Logistic分布的概率密度函数的图形：

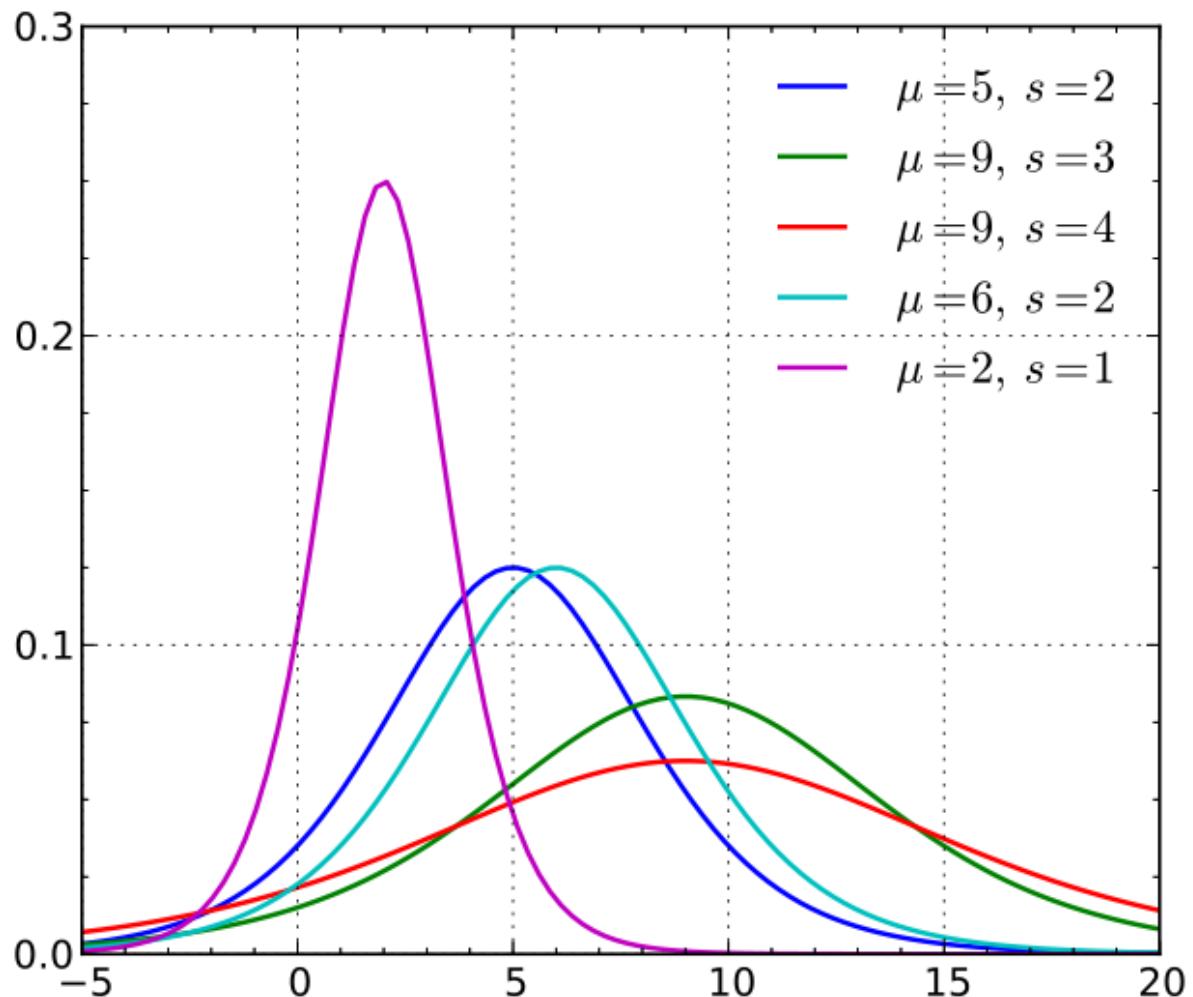


图1: Logistic分布的概率密度函数 (来自wiki)

下图为不同的 μ 和 s 的情况下， Logistic分布的累积分布函数的图形：

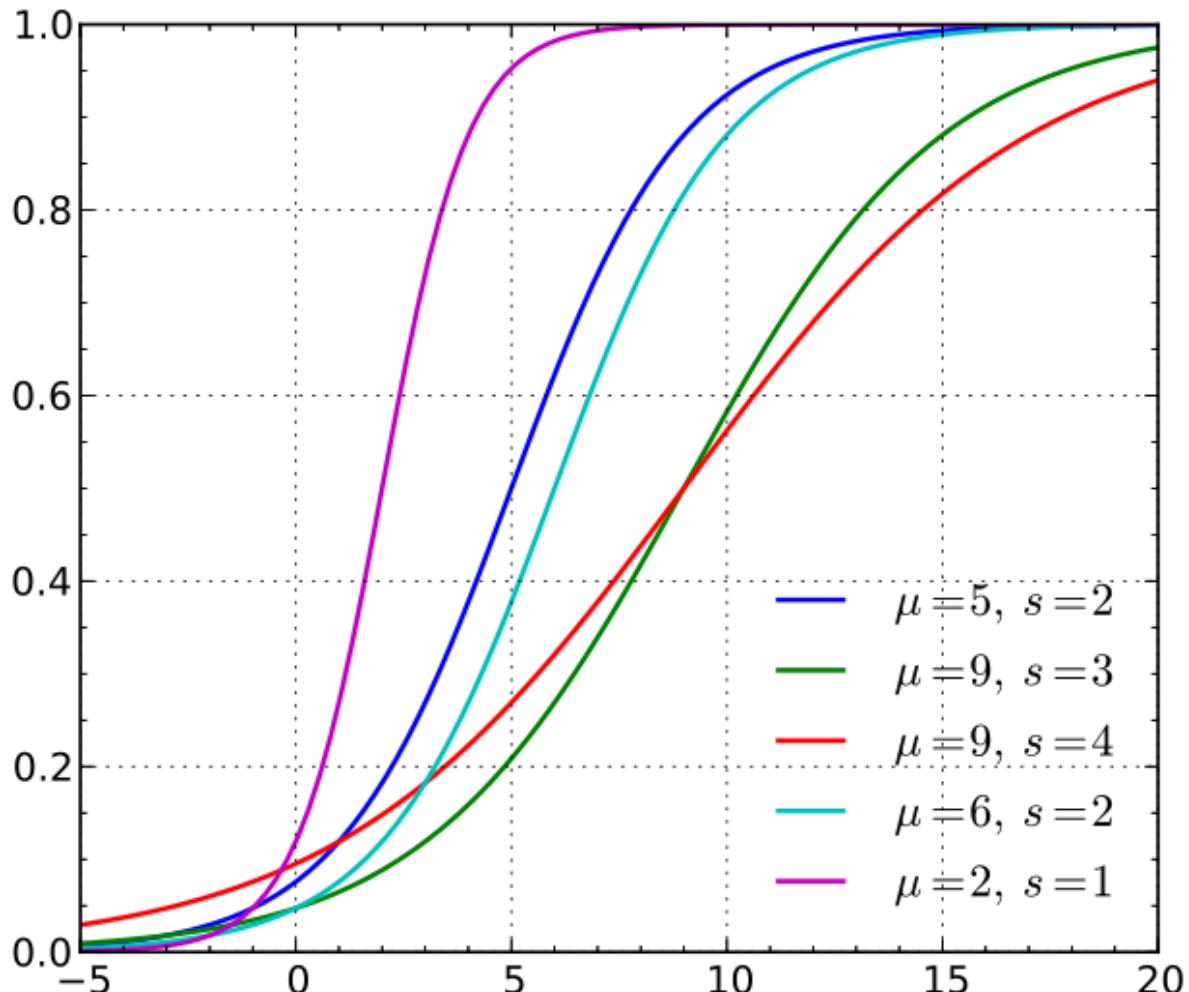


图1: Logistic分布的累积概率函数 (来自wiki)

由图可以看出， Logistic 分布的形状与正态分布的形状相似。但是Logistic 分布的尾部更长。特别注意 Logistic分布的概率分布函数分布函数图形是一条S形曲线，在中心附近增长速度较快，而在两端的增长速度相对较慢。该曲线以点 $(\mu, \frac{1}{2})$ 位中心对称， 即满足 $F(-x + \mu) - \frac{1}{2} = F(x + \mu) + \frac{1}{2}$

当 $\mu = 0, s = 1$ 时， Logistic分布的概率分布函数就是我们常说的**sigmoid函数**：

$$\sigma(a) = \frac{1}{1 + e^{-a}}$$

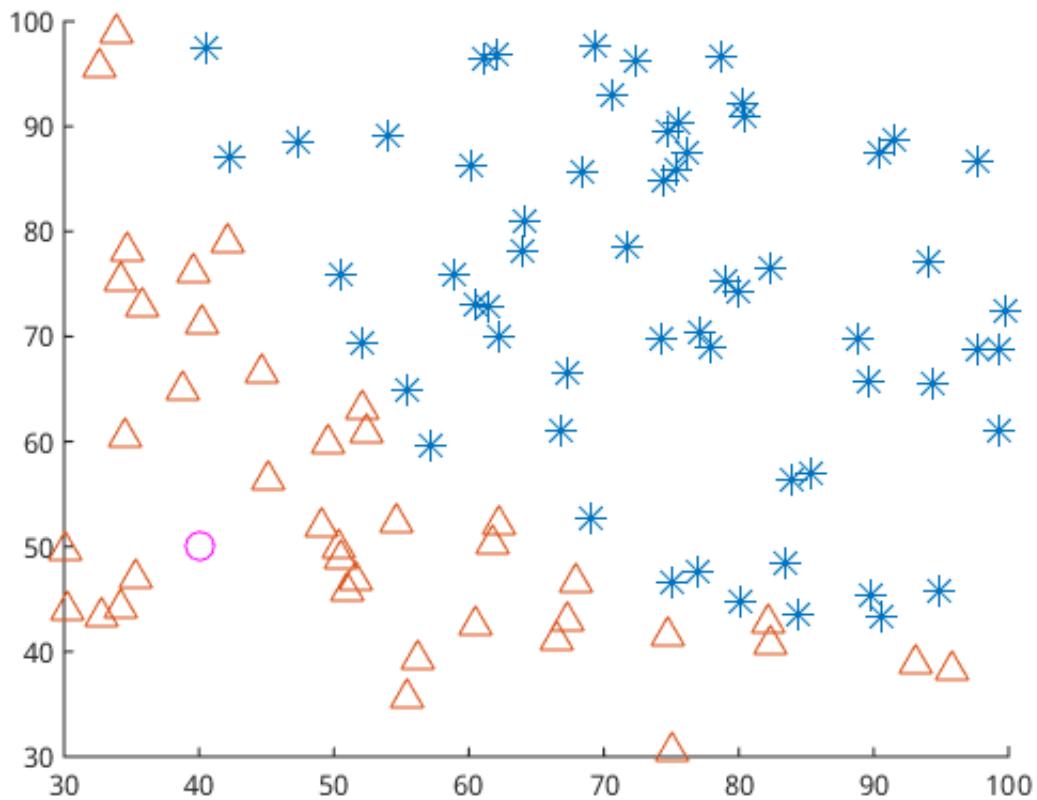
且其导数为：

$$\frac{d\sigma}{da} = \frac{-e^{-a}}{(1 + e^{-a})^2} = \sigma(1 - \sigma)$$

这是一个非常好的特性，并且这个特性在后面的推导中将会被用到。

2.2 从Logistic分布 到 Logistic回归模型

Logistic回归用于分类任务。下面是一个二维平面上两类分类的例子。其中橙色的三角和蓝色的米星分别是两类样本的训练样本，我们需要对一个未知类别的样本紫色圆圈进行分类。



Logistic回归模型用当 $\mu = 0, s = 1$ 的Logistic分布的概率分布函数（sigmoid函数）对后验概率 $p(y = c|\mathbf{x})$ 进行建模。对两类分类问题，logistic回归模型为：

$$p(y = 1|\mathbf{x}) = \sigma(a) = \frac{1}{1 + e^{-a}}$$

$$p(y = 0|\mathbf{x}) = 1 - p(y = 1|\mathbf{x}) = 1 - \sigma(a) = \frac{e^{-a}}{1 + e^{-a}}$$

一个事件的几率(odds)是指该事件发生的概率与不发生的概率的比值。两个概率相除，得到事件的几率为：

$$\frac{p(y = 1|\mathbf{x})}{p(y = 0|\mathbf{x})} = e^a$$

两边取log运算，得到该事件发生的对数几率(log odds)或logit函数是

$$a = \ln \frac{p(y = 1|\mathbf{x})}{p(y = 0|\mathbf{x})}$$

周志华老师的“机器学习”书中采用意译，称Logistic回归为对数几率回归。在此我们直接用英文单词Logistic。

所以当 $a > 0$, $p(y = 1|\mathbf{x}) > p(y = 0|\mathbf{x})$, 如果取最大后验概率, \mathbf{x} 的类别 $y = 1$;

如果 $a < 0$, 则 $p(y = 1|\mathbf{x}) < p(y = 0|\mathbf{x})$, 此时取最大后验概率, \mathbf{x} 的类别取 $y = 0$;

当 $a = 0$, 则 $p(y = 1|\mathbf{x}) = p(y = 0|\mathbf{x})$, \mathbf{x} 的类别 $y = 0$ 和 $y = 1$ 的概率相等, 此时 \mathbf{x} 位于决策面上, 可将 \mathbf{x} 任意分类到某一类, 或拒绝作出判断。

因此即 a 就是我们在分类算法中的决策面。

所谓分类器, 一般是将输入空间 X , 根据需要划分的类别, 将输入空间划分为一些互不相交的区域, 这些区域的边界一般叫做决策面 (**decision boundaries**)。预测函数的形式不同, 会使得决策面或者光滑, 或者粗糙。其中有一种比较特别的就是判别面是参数的线性函数的, 称为线性决策面, 形成的分类器就是线性分类器。

分类器为每个类别分配一个判别函数, 根据判别函数来判断一个新的样本是否是这个类别的。比如, 假设有 C 个类别, 那么分类器肯定会得到 C 个判别函数 $\delta_c(\mathbf{x}); c \in [1, 2, \dots, C]$ 。如果有一个新的样本 \mathbf{x} , 那么一般是找到最大的 $\delta_c(\mathbf{x})$, 就可以认为, 新的样本属于第 c 类。

在一般的分类器中, 判别式函数 $\delta_c(\mathbf{x})$, 和后验概率 $p(y = c|\mathbf{x})$ 是对应的, 能够使判别式的结果最大, 同样也是能够使得样本 \mathbf{x} 在类别 c 下的后验概率最大。

判别式函数 $\delta_c(\mathbf{x})$ 和 $\delta_k(\mathbf{x})$ 相等的点集, 就是我们常说的决策面:

$$\delta_c(\mathbf{x}) = \delta_k(\mathbf{x})$$

由于LR回归是一个线性分类算法, 所以

$$a(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$$

其中参数向量 \mathbf{w} 为线性组合的权重。即Logistic回归模型中, 输出 $y = 1|\mathbf{x}$ 的对数几率是输入 \mathbf{x} 的线性函数。

将上述两步综合在一起, Logistic回归模型可以重写为下面这个形式:

$$p(y = 1|\mathbf{x}) = \sigma(a(\mathbf{x})) = \frac{1}{1 + e^{-a(\mathbf{x})}} = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}$$

2.3 负log似然损失/极大似然估计

对于一个两类分类问题, $y \in [0, 1]$, 所以给定 \mathbf{x} , $p(y|\mathbf{x})$ 可以用Bernoulli分布表示。在Bernoulli分布 $y|\mathbf{x} \sim Ber(\theta)$ 中, 参数 θ 表示在给定 \mathbf{x} 的情况下, $y = 1$ 的概率。后验概率的概率函数为:

$$p(y|\mathbf{x}) = \theta^y (1 - \theta)^{1-y} = \begin{cases} \theta & \text{if } y = 1 \\ 1 - \theta & \text{if } y = 0 \end{cases}$$

这里请注意Logistic回归中后验分布 $p(y|\mathbf{x})$ 的概率分布为Bernoulli分布, 而Bernoulli分布中的参数 θ 又用logit函数 $\theta = \sigma(a) = \sigma(\mathbf{w}^T \mathbf{x})$ 表示, 即 $y|\mathbf{x} \sim Ber(\sigma(\mathbf{w}^T \mathbf{x}))$ 。

Bernoulli分布又被称为两点分布, 是一个是一个离散型概率分布, 为纪念瑞士科学家雅各布·伯努利 (Jakob I. Bernoulli) 而命名。若伯努利试验成功, 则伯努利随机变量取值为1。若伯努利试验失败, 则伯努利随机变量 X 取值为0。记其成功概率为 $\theta(0 \leq \theta \leq 1)$, 则失败的概率为 $(1 - \theta)$, 记为 $X \sim Ber(\theta)$

$$p(x) = \theta^x (1 - \theta)^{1-x} = \begin{cases} \theta & \text{if } x = 1 \\ 1 - \theta & \text{if } x = 0 \end{cases}$$

给定训练样本的情况下，我们可以用极大似然估计求解模型参数。

- 极大似然估计 (Maximize Likelihood Estimator, MLE) 定义为 (即给定参数 θ 的情况下，数据 D 出现的概率为 p ，则MLE取使得 p 最大的参数 θ)

$$\hat{\theta} = \arg \max_{\theta} p(D|\theta)$$

由于我们假设训练数据 (\mathbf{x}_i, y_i) 是独立同分布 (Independent Identical Distribution, IID) 的样本，所以数据 D 出现的概率为各训练样本出现的概率相乘 (独立随机向量的联合分布等于各随机变量的分布相乘)，即似然函数表示在参数为 θ 的情况下，数据 $D = \{\mathbf{x}_i, y_i\}_{i=1}^N$ 出现的概率：

$$L(\theta) = p(D|\theta) = \prod_{i=1}^N p(y_i|\mathbf{x}_i, \theta)$$

取log运算 (通常log运算后更简单)，得到**log似然函数**为

$$l(\theta) = \log p(D|\theta) = \sum_{i=1}^N \log p(y_i|\mathbf{x}_i, \theta)$$

极大似然估计为选择数据出现概率最大的参数：

$$\hat{\theta} = \arg \max_{\theta} l(\theta)$$

由于Bernoulli分布中，参数 $\theta = \sigma(a) = \sigma(\mathbf{w}^T \mathbf{x})$ 表示Bernoulli分布的期望。为了书写简洁，下面用 μ 表示 $\mu(\mathbf{x}) = \sigma(a(\mathbf{x})) = \frac{1}{1+e^{-a(\mathbf{x})}} = \frac{1}{1+e^{-\mathbf{w}^T \mathbf{x}}}$ 。在Logistic回归中，令

$$\begin{aligned} p(y=1|\mathbf{x}) &= \mu(\mathbf{x}) \\ p(y=0|\mathbf{x}) &= 1 - \mu(\mathbf{x}) \end{aligned}$$

将Bernoulli分布代入**log似然函数**计算公式，得到

$$\begin{aligned} l(\theta) &= \sum_{i=1}^N \log p(y_i|\mathbf{x}_i, \theta) \\ &= \sum_{i=1}^N \log (\mu(\mathbf{x}_i)^{y_i} (1 - \mu(\mathbf{x}_i))^{1-y_i}) \\ &= \sum_{i=1}^N [y_i \log(\mu(\mathbf{x}_i)) + (1 - y_i) \log(1 - \mu(\mathbf{x}_i))] \end{aligned}$$

log似然极大，等价于损失函数取负**log似然**，即**负log损失函数**为

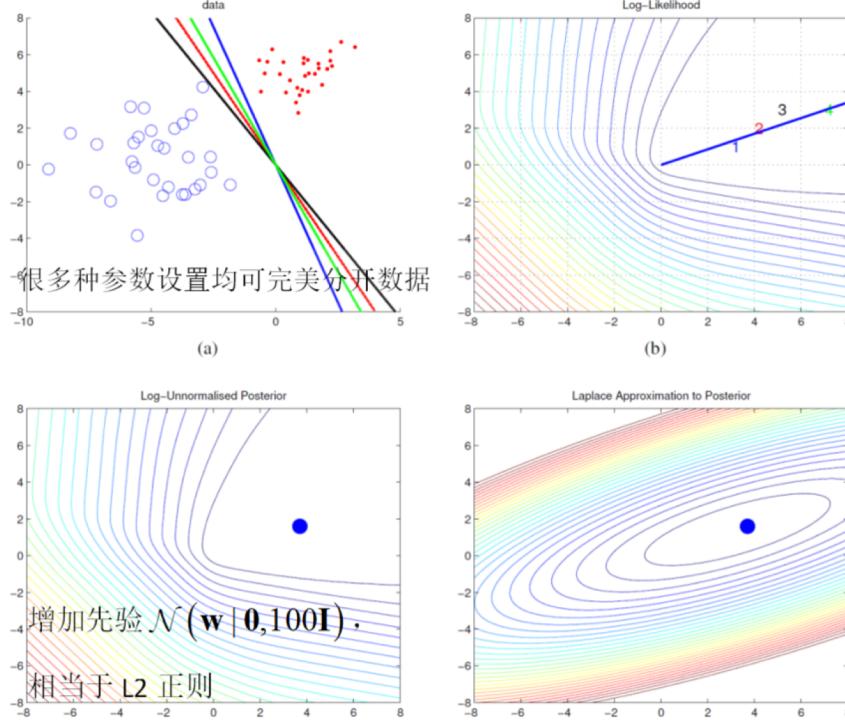
$$\mathcal{L}(y, \mu(\mathbf{x})) = NLL(y, \mu(\mathbf{x})) = -y \log(\mu(\mathbf{x})) - (1 - y) \log(1 - \mu(\mathbf{x}))$$

然后取训练集上的平均损失最小，即最小化目标函数

$$J(\mathbf{w}) = \sum_{i=1}^N \mathcal{L}(y_i, \mu(\mathbf{x}_i)) = - \sum_{i=1}^N [y_i \log(\mu(\mathbf{x}_i)) + (1 - y_i) \log(1 - \mu(\mathbf{x}_i))]$$

2.4 带正则的Logistic回归

同带正则的线性回归类似，Logistic回归也可以带正则惩罚项。事实上，同线性回归可以不带正则（最小二乘线性回归）不同，Logistic回归必须带正则项。请看下面完全线性可分的例子。



似然函数的最大值为 ∞ ，
因为当数据完美可分时，
 $\|\mathbf{w}\|$ 越大，sigmoid 函数越
陡，从而似然值越大

从图中可以，分界面1、2、3和4均可以将两类数据完全分开，但4的似然值最大。事实上对组合权重系数 \mathbf{w} 的每个元素乘以一个无穷大的标量会使得似然值更大，因此必须加正则项限制权重系数 \mathbf{w} 无限制增大但模型的推广性不好（过拟合）。

Logistic回归可以带L2正则和L1正则。关于L2正则和L1正则的详细说明请参看线性回归部分。

L2惩罚的Logistic回归：

$$\begin{aligned} J(\mathbf{w}) &= \sum_{i=1}^N NLL(y_i, \mu(\mathbf{x}_i)) + \lambda \|\mathbf{w}\|_2^2 \\ &= - \sum_{i=1}^N [y_i \log(\mu(\mathbf{x}_i)) + (1 - y_i) \log(1 - \mu(\mathbf{x}_i))] + \lambda \|\mathbf{w}\|_2^2 \end{aligned}$$

L1惩罚的Logistic回归：

$$\begin{aligned} J(\mathbf{w}) &= \sum_{i=1}^N NLL(y_i, \mu(\mathbf{x}_i)) + \lambda \|\mathbf{w}\|_1 \\ &= - \sum_{i=1}^N [y_i \log(\mu(\mathbf{x}_i)) + (1 - y_i) \log(1 - \mu(\mathbf{x}_i))] + \lambda \|\mathbf{w}\|_1 \end{aligned}$$

其中 λ 为正则参数，用于调节目标函数和惩罚项之间关系。 λ 越大，惩罚力度越大，所得到的最优解越趋近于0，或者说参数向量越稀疏； λ 越小，惩罚力度越小，模型和训练数据拟合得越好。

2.5 梯度下降

我们先讨论损失函数部分的梯度，带目标函数中的正则部分的梯度稍后讨论。

损失函数为：

$$J_1(\mathbf{w}) = \sum_{i=1}^N \mathcal{L}(y_i, \mu(\mathbf{x}_i)) = -\sum_{i=1}^N [y_i \log(\mu(\mathbf{x}_i)) + (1 - y_i) \log(1 - \mu(\mathbf{x}_i))]$$

则梯度为：

$$\begin{aligned}\nabla_{\mathbf{w}} = g(\mathbf{w}) &= \frac{\partial}{\partial \mathbf{w}} J_1(\mathbf{w}) = -\sum_{i=1}^N [y_i \times \frac{1}{\mu(\mathbf{x}_i)} - (1 - y_i) \times \frac{1}{1 - \mu(\mathbf{x}_i)}] \times \frac{\partial}{\partial \mathbf{w}} \mu(\mathbf{x}_i) \\ &= -\sum_{i=1}^N [y_i \times \frac{1}{\mu(\mathbf{x}_i)} - (1 - y_i) \times \frac{1}{1 - \mu(\mathbf{x}_i)}] \times \frac{d\sigma(a_i)}{da_i} \frac{\partial a_i}{\partial \mathbf{w}} \\ &= -\sum_{i=1}^N [y_i \times \frac{1}{\mu(\mathbf{x}_i)} - (1 - y_i) \times \frac{1}{1 - \mu(\mathbf{x}_i)}] \times \mu(\mathbf{x}_i)(1 - \mu(\mathbf{x}_i))^{-1} \\ &= -\sum_{i=1}^N [y_i \times \frac{1}{\mu(\mathbf{x}_i)} - (1 - y_i) \times \frac{1}{1 - \mu(\mathbf{x}_i)}] \times \mu(\mathbf{x}_i)(1 - \mu(\mathbf{x}_i))^{-1} \times \mathbf{x}_i \\ &= -\sum_{i=1}^N [y_i \times (1 - \mu(\mathbf{x}_i)) - (1 - y_i) \times \mu(\mathbf{x}_i)] \times \mathbf{x}_i \\ &= -\sum_{i=1}^N [y_i - \mu(\mathbf{x}_i)] \times \mathbf{x}_i \\ &= \sum_{i=1}^N [\mu(\mathbf{x}_i) - y_i] \times \mathbf{x}_i\end{aligned}$$

如果将 $\mu(\mathbf{x}_i)$ 看成是对 y_i 的预测，则 $\mu(\mathbf{x}_i) - y_i$ 可以看出是预测残差，Logistic回归损失函数与线性回归损失函数得到的梯度形式相同。事实上所有线性模型的梯度形式均是如此。

得到梯度之后，那么就可以和线性回归模型一样，采用标准的梯度下降法求解参数（这里暂未考虑正则项的梯度）：

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla_{\mathbf{w}}$$

梯度下降法实现相对简单，但是其收敛速度往往不尽人意，可以考虑使用随机梯度下降法来解决收敛速度的问题和大样本问题（CTR预估部分讲解）。

但上面两种优化方法在最小值附近，都存在以一种曲折的慢速逼近方式来逼近最小点的问题。所以在LR回归的实际算法中，用到的是牛顿法或拟牛顿法（DFP、BFGS、L-BFGS）。当然样本数特别大时，随机梯度下降是最佳选择。

由于求解最优解的问题，其实是一个凸优化的问题，这些数值优化方法的区别仅仅在于选择什么方向走向最优解，而这个方向通常是优化函数在当前点的一阶导数（梯度）或者二阶导数（Hessian矩阵）决定的。比如梯度下降法用的就是一阶导数，而牛顿法和拟牛顿法用的就是二阶导数。

2.6 牛顿法

2.6.1 一元函数的牛顿法

牛顿法的最初提出是用来求解方程的根的。我们假设点 x^* 为函数 $f(x)$ 的根，那么有 $f(x^*) = 0$ 。现在我们将函数 $f(x)$ 在点 $x^{(t)}$ 处一阶泰勒展开有：

$$f(x) = f(x^{(t)}) + f'(x^{(t)})(x - x^{(t)})$$

其中这里 f' 表示函数 f 的导数。

那么假设点 $x^{(t+1)}$ 为该方程的根，则有

$$f(x^{(t+1)}) = f(x^{(t)}) + f'(x^{(t)})(x^{(t+1)} - x^{(t)}) = 0$$

那么就可以得到

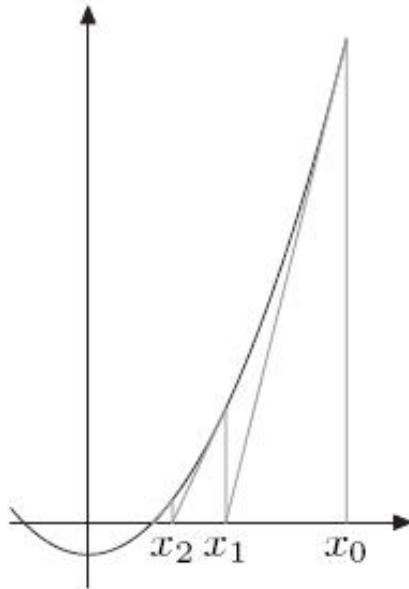
$$x^{(t+1)} = x^{(t)} - \frac{f(x^{(t)})}{f'(x^{(t)})}$$

这样我们就得到了一个递归方程，我们可以通过迭代的方式不断的让 x 趋近于 x^* 从而求得方程 $f(x) = 0$ 的解。

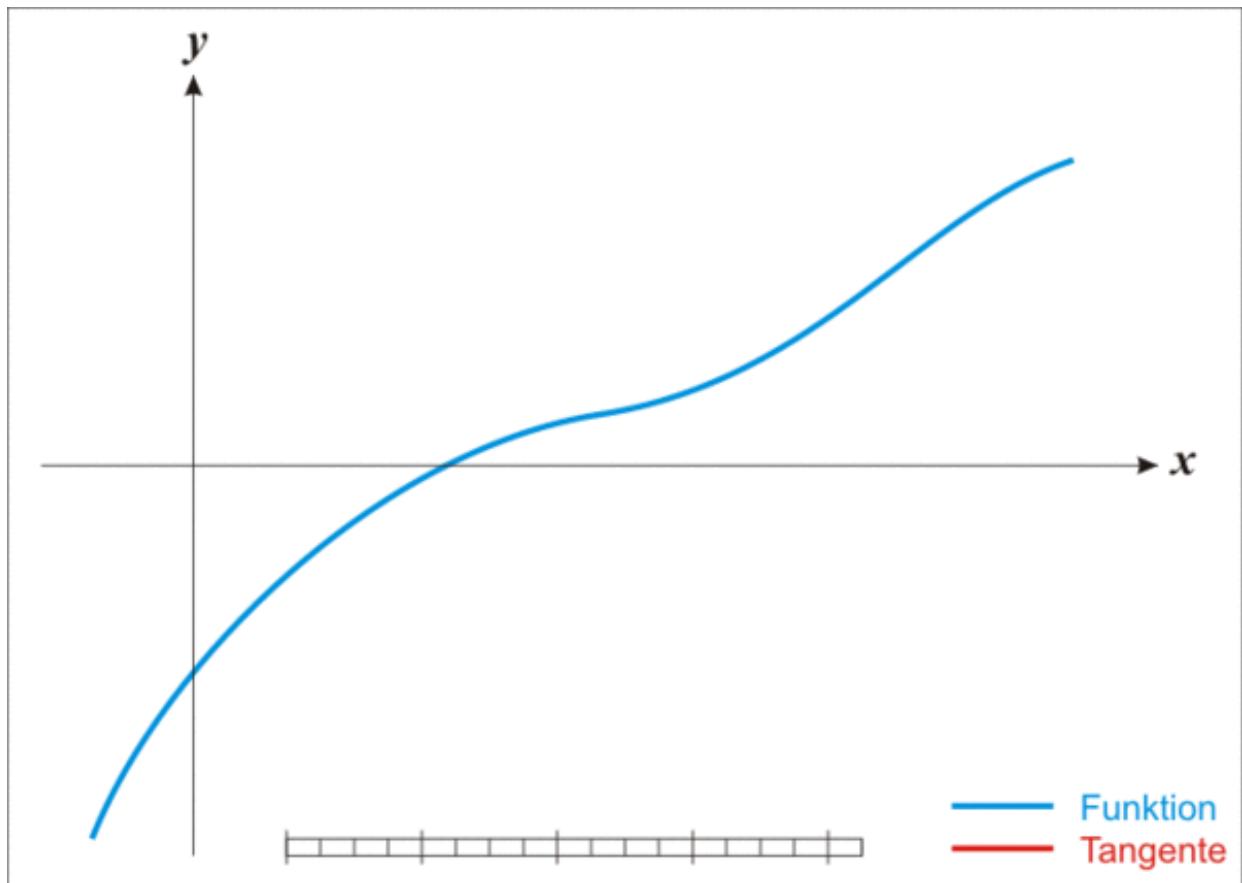
迭代公式用图示表示如下图。选择一个接近函数 $f(x)$ 零点的 x^0 ，计算相应的 $f(x^0)$ 和切线斜率 $f'(x^0)$ 。然后我们计算穿过点 $(x^0, f(x^0))$ ，并且斜率为 $f'(x^0)$ 的直线和 x 轴的交点的 x 坐标，也就是求如下方程的解

$$0 = f(x^{(0)}) + f'(x^{(0)})(x - x^{(0)})$$

我们将新求得的点的 x 坐标命名为 $x^{(1)}$ ，更接近方程 $f(x) = 0$ 的解。因此我们现在可以利用 $x^{(1)}$ 开始下一轮迭代。



下图是来自Wiki上一个牛顿法求解函数的根的动画（蓝色线为函数 $f(x)$ ，红色线为当前点的切线。图中用下标表示迭代索引，类似我们上文中的上标 t ），可以很清楚地看到牛顿法的迭代过程。



2.6.2 牛顿法用于优化求解

对于最优化问题，其极值点处的特性之一是在极值点处函数的一阶导数为0。因此我们可以在一阶导数处利用牛顿法通过迭代的方式来求得最优解，即相当于求一阶导数对应函数的根。

将2.6.1节中的 $f(x)$ 换成其一阶导数 $g(x) = f'(x)$ ，得到

$$\begin{aligned} \mathbf{x}^{(t+1)} &= \mathbf{x}^{(t)} - \frac{g(\mathbf{x}^{(t)})}{g'(\mathbf{x}^{(t)})} \\ &= \mathbf{x}^{(t)} - \frac{f'(\mathbf{x}^{(t)})}{f''(\mathbf{x}^{(t)})} \end{aligned}$$

这样我们就得到了一个不断更新 \mathbf{x} 迭代求得最优解的方法。

上面我们讨论的都是 \mathbf{x} 为标量的情形。那么对于高维函数，其一阶导数为梯度，

$$\mathbf{g}(\mathbf{x}) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_D} \right)^T$$

二阶导数就变为了一个海森 (Hessian) 矩阵，记为

$$\mathbf{H}(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_D} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_D} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_D \partial x_1} & \frac{\partial^2 f}{\partial x_D \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_D^2} \end{bmatrix}$$

那么迭代公式就变为了

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \mathbf{H}^{-1} \mathbf{g}(\mathbf{x}^{(t)})$$

牛顿法求最值的步骤如下：

1. 随机选取起始点 $\mathbf{x}^{(0)}$ ；
2. 计算目标函数 $f(\mathbf{x})$ 在该点 $\mathbf{x}^{(0)}$ 的一阶导数 $\mathbf{g}(\mathbf{x}^{(t)})$ 和海森矩阵 $\mathbf{H}(\mathbf{x}^{(t)})$ ；
3. 依据迭代公式 $\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \mathbf{H}^{-1} \mathbf{g}(\mathbf{x}^{(t)})$ 更新 \mathbf{x} 值
4. 如果 $(f(\mathbf{x}^{(t+1)}) - f(\mathbf{x}^{(t)})) < \epsilon$ ，则收敛返回，否则继续步骤2,3直至收敛

由于牛顿法用到了函数的二阶导数，亦被称为二阶优化方法。可以看出，与梯度下降法相比，牛顿法用的特点是收敛速度快，迭代次数少。

2.6.3 牛顿法用于Logistic回归的MLE求解 (Iteratively reweighted least squares, IRLS)

我们暂时忽略正则项，对logistic回归的MLE（目标函数只取损失函数部分）采用牛顿法求解。

根据2.6.1节中的结论，Logistic回归的MLE解的梯度为

$$\nabla_{\mathbf{w}} = g(\mathbf{w}) = \sum_{i=1}^N [\mu(\mathbf{x}_i) - y_i] \times \mathbf{x}_i = \mathbf{X}^T (\mu - \mathbf{y})$$

则Hessian矩阵为：

$$\begin{aligned}\mathbf{H}(\mathbf{w}) &= \frac{\partial}{\partial \mathbf{w}}(g(\mathbf{w})^T) = \sum_{i=1}^N (\nabla_{\mathbf{w}} \mu(\mathbf{x}_i)) \mathbf{x}_i^T = \sum_{i=1}^N \mu(\mathbf{x}_i)(1 - \mu(\mathbf{x}_i)) \mathbf{x}_i \mathbf{x}_i^T \\ &= \mathbf{X}^T \mathbf{S} \mathbf{X}\end{aligned}$$

其中 $\mathbf{S} \triangleq \text{diag}(\mu(\mathbf{x}_i)(1 - \mu(\mathbf{x}_i)))$, 由于 $0 \leq \mu(\mathbf{x}_i) \leq 1$, $0 \leq (1 - \mu(\mathbf{x}_i)) \leq 1$, 所以 \mathbf{H} 为正定矩阵, Logistic回归有唯一的全局最优解。

$$\begin{aligned}\mathbf{w}^{(t+1)} &= \mathbf{w}^{(t)} - (\mathbf{H}^{(t)}(\mathbf{w}))^{-1} \mathbf{g}(\mathbf{w}^{(t)}) \\ &= \mathbf{w}^{(t)} - (\mathbf{X}^T \mathbf{S}^{(t)} \mathbf{X})^{-1} \mathbf{X}^T (\mu - \mathbf{y}) \\ &= (\mathbf{X}^T \mathbf{S}^{(t)} \mathbf{X})^{-1} [(\mathbf{X}^T \mathbf{S}^{(t)} \mathbf{X}) \mathbf{w}^{(t)} - \mathbf{X}^T (\mu - \mathbf{y})] \\ &= (\mathbf{X}^T \mathbf{S}^{(t)} \mathbf{X})^{-1} \mathbf{X}^T [(\mathbf{S}^{(t)} \mathbf{X}) \mathbf{w}^{(t)} + (\mathbf{y} - \mu)] \\ &= (\mathbf{X}^T \mathbf{S}^{(t)} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{S}^{(t)} [(\mathbf{X} \mathbf{w}^{(t)} - (\mathbf{S}^{(t)})^{-1} (\mathbf{y} - \mu))]\end{aligned}$$

令 $\mathbf{z}^{(t)} = \mathbf{X} \mathbf{w}^{(t)} - (\mathbf{S}^{(t)})^{-1} (\mathbf{y} - \mu)$, 则

$$\mathbf{w}^{(t+1)} = (\mathbf{X}^T \mathbf{S}^{(t)} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{S}^{(t)} \mathbf{z}^{(t)}$$

对比最小二乘线性回归的权重求解公式

$$\mathbf{w}_{OLS} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

可以看出, Logistic回归的权重迭代相当于对数据 \mathbf{X} 施加权重 $(\mathbf{S}^{(t)})^{\frac{1}{2}}$ (OLS公式中的 \mathbf{X} 用 $(\mathbf{S}^{(t)})^{\frac{1}{2}} \mathbf{X}$ 代入, \mathbf{y} 用 $(\mathbf{S}^{(t)})^{\frac{1}{2}} \mathbf{y}$ 代入即可得到Logistic回归的权重迭代公式), 然后再进行最小二乘求解。因此Logistic回归的牛顿法求解亦被称为迭代再加权最小二乘 (Iteratively reweighted least squares, IRLS), 其中权重矩阵 \mathbf{S} 依赖参数向量 \mathbf{w} 。

综上所述, IRLS算法如下:

Iteratively reweighted least squares(IRLS)

1 $\mathbf{w} = \mathbf{0}_D$

2 $w_0 = \log(\bar{y} / (1 - \bar{y}))$

3 repeat

4 $\eta_i = w_0 + \mathbf{w}^T \mathbf{x}_i$

5 $\mu_i = \text{sigm}(\eta_i)$

6 $s_i = \mu_i(1 - \mu_i)$

7 $z_i = \eta_i + \frac{y_i - \mu_i}{s_i}$

8 $\mathbf{S} = \text{diag}(\mathbf{s}_{1:N})$

9 $\mathbf{w} = (\mathbf{X}^T \mathbf{S} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{S} \mathbf{z}$

10 until converged

$$\mathbf{S} = \text{diag}\left(\mu_1(1 - \mu_1), \dots, \mu_N(1 - \mu_N)\right)$$

$$\mathbf{z}_i = \mathbf{w}^T \mathbf{x}_i + \frac{y_i - \mu_i}{\mu_i(1 - \mu_i)}$$

Weighted least square

2.6.4 拟牛顿法

由于牛顿法中需要求解二阶偏导，计算量较大。随着数据规模的增大 (D 增大)，那么 Hessian 矩阵 ($D \times D$) 会越大，需要的存储空间会增多，计算量也会增大，有时候大到不可计算，所以针对海量数据的计算，牛顿法不再适用。拟牛顿法是一些算法的总称，它们的目标是通过某种方式来近似表示森海矩阵(或者它的逆矩阵)，避免每次迭代都计算Hessian矩阵的逆，它的收敛速度介于梯度下降法和牛顿法之间。

拟牛顿法跟牛顿法一样，也是不能处理太大规模的数据，因为计算量和存储空间会开销很多。拟牛顿法虽然每次迭代不像牛顿法那样保证是最优化的方向，但是近似矩阵始终是正定的，因此算法始终是朝着最优化的方向在搜索。

下面我们用 \mathbf{B} 表示 Hessian 矩阵 \mathbf{H} 的近似，用 \mathbf{D} 表示 Hessian 矩阵的逆矩阵 \mathbf{H}^{-1} 的近似。

我们先来看看用 \mathbf{B} 表示 Hessian 矩阵 \mathbf{H} 近似的条件，即拟牛顿条件。

假设经过 $t + 1$ 次迭代后，将目标函数在 $\mathbf{f}(\mathbf{x})$ 在点 $\mathbf{x}^{(t+1)}$ 进行 Taylor 展开，则有

$$\mathbf{f}(\mathbf{x}) \approx \mathbf{f}(\mathbf{x}^{(t+1)}) + \nabla \mathbf{f}(\mathbf{x}^{(t+1)})(\mathbf{x} - \mathbf{x}^{(t+1)}) + \frac{1}{2}(\mathbf{x} - \mathbf{x}^{(t+1)})^T \nabla^2 \mathbf{f}(\mathbf{x}^{(t+1)})(\mathbf{x} - \mathbf{x}^{(t+1)})$$

对上式两边同取梯度算子 ∇ ，那么就可以得到

$$\nabla \mathbf{f}(\mathbf{x}) \approx \nabla \mathbf{f}(\mathbf{x}^{(t+1)}) + \nabla^2 \mathbf{f}(\mathbf{x}^{(t+1)})(\mathbf{x} - \mathbf{x}^{(t+1)})$$

上式取 $\mathbf{x} = \mathbf{x}^{(t)}$ ，得到

$$\begin{aligned}\nabla f(\mathbf{x}^{(t)}) &\approx \nabla f(\mathbf{x}^{(t+1)}) + \nabla^2 f(\mathbf{x}^{(t+1)})(\mathbf{x}^{(t)} - \mathbf{x}^{(t+1)}) \\ \mathbf{g}(\mathbf{x}^{(t)}) &\approx \mathbf{g}(\mathbf{x}^{(t+1)}) + \mathbf{H}^{(t+1)}(\mathbf{x}^{(t)} - \mathbf{x}^{(t+1)}) \\ \mathbf{g}(\mathbf{x}^{(t+1)}) - \mathbf{g}(\mathbf{x}^{(t)}) &\approx \mathbf{H}^{(t+1)}(\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)})\end{aligned}$$

令 $\mathbf{s}^{(t)} = \mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}$, $\mathbf{y}^{(t)} = \mathbf{g}(\mathbf{x}^{(t+1)}) - \mathbf{g}(\mathbf{x}^{(t)})$, 得到拟牛顿条件为:

$$\begin{aligned}\mathbf{y}^{(t)} &\approx \mathbf{H}^{(t+1)}\mathbf{s}^{(t)} \\ \mathbf{s}^{(t)} &\approx (\mathbf{H}^{(t+1)})^{-1}\mathbf{y}^{(t)}\end{aligned}$$

拟牛顿条件对迭代过程中的海森矩阵 $\mathbf{H}^{(t+1)}$ 进行约束。因此对Hessian矩阵的近似 $\mathbf{B}^{(t+1)}$ 和逆Hessian矩阵的近似 $\mathbf{D}^{(t+1)}$, 有

$$\begin{aligned}\mathbf{y}^{(t)} &= \mathbf{B}^{(t+1)}\mathbf{s}^{(t)} \\ \mathbf{s}^{(t)} &= \mathbf{D}^{(t+1)}\mathbf{y}^{(t)}\end{aligned}$$

2.6.5 BFGS

BFGS是一种拟牛顿法, 它是由四个发明人 (Broyden, Fletcher, Goldfarb, and Shanno) 的首字母组合命名, 是求解无约束非线性优化问题最常用的方法之一。

BFGS目标是用迭代的方式逼近海森矩阵 \mathbf{H} 假。设逼近值为 $\mathbf{B}^t \approx \mathbf{H}^t$, 那么希望通过计算 $\mathbf{B}^{t+1} = \mathbf{B}^t + \Delta\mathbf{B}^t$ 达到目的 (其中 \mathbf{B}^0 通常取单位矩阵)。

假设 $\Delta\mathbf{B}^t = \alpha\mathbf{u}\mathbf{u}^T + \beta\mathbf{v}\mathbf{v}^T$, 并代入拟牛顿条件 $\mathbf{y}^{(t)} = \mathbf{B}^{(t+1)}\mathbf{s}^{(t)}$, 得到

$$\begin{aligned}\mathbf{y}^{(t)} &= (\mathbf{B}^t + \Delta\mathbf{B}^t)\mathbf{s}^{(t)} \\ &= \mathbf{B}^t\mathbf{s}^{(t)} + \Delta\mathbf{B}^t\mathbf{s}^{(t)} \\ &= \mathbf{B}^t\mathbf{s}^{(t)} + (\alpha\mathbf{u}\mathbf{u}^T + \beta\mathbf{v}\mathbf{v}^T)\mathbf{s}^{(t)} \\ &= \mathbf{B}^t\mathbf{s}^{(t)} + (\alpha\mathbf{u}^T\mathbf{s}^{(t)})\mathbf{u} + (\beta\mathbf{v}^T\mathbf{s}^{(t)})\mathbf{v}\end{aligned}$$

令 $\alpha\mathbf{u}^T\mathbf{s}^{(t)} = 1$, $\beta\mathbf{v}^T\mathbf{s}^{(t)} = -1$, 以及 $\mathbf{u} = \mathbf{y}^{(t)}$, $\mathbf{v} = \mathbf{B}^{(t)}\mathbf{s}^{(t)}$, 可计算得到

$$\begin{aligned}\alpha &= \frac{1}{(\mathbf{y}^{(t)})^T\mathbf{s}^{(t)}}, \\ \beta &= -\frac{1}{(\mathbf{s}^{(t)})^T\mathbf{B}^{(t)}\mathbf{s}^{(t)}}\end{aligned}$$

综上, 得到

$$\begin{aligned}\Delta\mathbf{B}^{(t)} &= \alpha\mathbf{u}\mathbf{u}^T + \beta\mathbf{v}\mathbf{v}^T \\ &= \frac{\mathbf{y}^{(t)}(\mathbf{y}^{(t)})^T}{(\mathbf{y}^{(t)})^T\mathbf{s}^{(t)}} + \frac{\mathbf{B}^{(t)}\mathbf{s}^{(t)}(\mathbf{s}^{(t)})^T(\mathbf{B}^{(t)})^T}{(\mathbf{s}^{(t)})^T\mathbf{B}^{(t)}\mathbf{s}^{(t)}}\end{aligned}$$

牛顿法中需要计算Hessian矩阵的逆矩阵, 根据Sherman-Morrison公式, 可得到

$$(\mathbf{B}^{(t+1)})^{-1} = \mathbf{D}^{(t+1)} = \left(\mathbf{I} - \frac{\mathbf{s}^{(t)}(\mathbf{y}^{(t)})^T}{(\mathbf{y}^{(t)})^T\mathbf{s}^{(t)}}\right)\mathbf{D}^{(t)} + \left(\mathbf{I} - \frac{\mathbf{y}^{(t)}(\mathbf{s}^{(t)})^T}{(\mathbf{y}^{(t)})^T\mathbf{s}^{(t)}}\right) + \frac{\mathbf{s}^{(t)}(\mathbf{s}^{(t)})^T}{(\mathbf{y}^{(t)})^T\mathbf{s}^{(t)}}$$

BFGS更新参数的流程如下：

1. 初始化变量 $\mathbf{w}^{(0)}, \mathbf{D}^{(0)} = \mathbf{I}, t = 0$
2. 确定搜索方向： $d^{(t)} = -\mathbf{D}^{(t)} \times \mathbf{g}^{(t)}$
3. 更新参数： $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \eta d^{(t)}$
4. 计算： $\Delta \mathbf{g} = \mathbf{g}^{(t+1)} - \mathbf{g}^{(t)}$
5. 计算： $\mathbf{D}^{(t+1)} = \left(\mathbf{I} - \frac{\mathbf{s}^{(t)}(\mathbf{y}^{(t)})^T}{(\mathbf{y}^{(t)})^T \mathbf{s}^{(t)}} \right) \mathbf{D}^{(t)} + \left(\mathbf{I} - \frac{\mathbf{y}^{(t)}(\mathbf{s}^{(t)})^T}{(\mathbf{y}^{(t)})^T \mathbf{s}^{(t)}} \right) + \frac{\mathbf{s}^{(t)}(\mathbf{s}^{(t)})^T}{(\mathbf{y}^{(t)})^T \mathbf{s}^{(t)}}$

这个更新方法跟牛顿法的区别是在更新参数 \mathbf{w} 之后更新一下近似森海矩阵的值，而牛顿法是在更新 \mathbf{w} 之前完全的计算一遍森海矩阵。还有一种从计算上改进BFGS的方法称为L-BFGS (limited memory BFGS)，不直接存储森海矩阵，而是通过存储计算过程中产生的 $\mathbf{s}^{(t)}, \mathbf{y}^{(t)}$ ，从而减少参数存储所需空间。

2.7 L2正则的Logistic回归求解

L2正则的的Logistic回归的目标函数为：

$$\begin{aligned} J(\mathbf{w}) &= \sum_{i=1}^N NLL(y_i, \mu(\mathbf{x}_i)) + \lambda \|\mathbf{w}\|_2^2 \\ &= -\sum_{i=1}^N [y_i \log(\mu(\mathbf{x}_i)) + (1 - y_i) \log(1 - \mu(\mathbf{x}_i))] + \lambda \|\mathbf{w}\|_2^2 \end{aligned}$$

根据2.6.3的结论，其中MLE部分 $J_1(\mathbf{w}) = \sum_{i=1}^N NLL(y_i, \mu(\mathbf{x}_i))$ 的梯度为

$$\mathbf{g}(\mathbf{w}) = \sum_{i=1}^N [\mu(\mathbf{x}_i) - y_i] \times \mathbf{x}_i = \mathbf{X}^T (\mu - \mathbf{y})$$

Hessian矩阵为：

$$\mathbf{H}(\mathbf{w}) = \sum_{i=1}^N \mu(\mathbf{x}_i)(1 - \mu(\mathbf{x}_i)) \mathbf{x}_i \mathbf{x}_i^T = \mathbf{X}^T \mathbf{S} \mathbf{X}$$

L2正则项的梯度和Hessian矩阵非常好计算，再分别加到在MLE部分的梯度和Hessian矩阵，从而得到Logistic回归的梯度为：

$$\mathbf{g}_{L2}(\mathbf{w}) = \mathbf{g}(\mathbf{w}) + \lambda \mathbf{w} = \mathbf{X}^T (\mu - \mathbf{y}) + \lambda \mathbf{w}$$

Hessian矩阵为：

$$\mathbf{H}_{L2}(\mathbf{w}) = \mathbf{H}(\mathbf{w}) + \lambda \mathbf{I} = \mathbf{X}^T \mathbf{S} \mathbf{X} + \lambda \mathbf{I}$$

得到梯度和Hessian矩阵后，L2正则的Logistic回归求解和MLE求解类似，可用（随机）梯度下降、牛顿法或拟牛顿法求解。

2.8 L1正则的Logistic回归求解

L1惩罚的Logistic回归：

$$\begin{aligned} J(\mathbf{w}) &= \sum_{i=1}^N NLL(y_i, \mu(\mathbf{x}_i)) + \lambda \|\mathbf{w}\|_1 \\ &= -\sum_{i=1}^N [y_i \log(\mu(\mathbf{x}_i)) + (1 - y_i) \log(1 - \mu(\mathbf{x}_i))] + \lambda \|\mathbf{w}\|_1 \end{aligned}$$

与L2正则不同，L1正则项的梯度和Hessian矩阵计算不方便，但可以得到系数解。这里我们仅讨论L1正则的牛顿法（IRLS）求解，梯度下降部分请自行推导。在线学习（随即梯度下降）求解我们在CTR预估部分再讨论。

根据2.6.3的结论，IRLS求解公式为：

$$\mathbf{w}^{(t+1)} = (\mathbf{X}^T \mathbf{S}^{(t)} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{S}^{(t)} \mathbf{z}^{(t)}$$

其中 $\mathbf{S} \triangleq \text{diag}(\mu(\mathbf{x}_i)(1 - \mu(\mathbf{x}_i)))$ ， $\mu(\mathbf{x}_i) = \sigma(\mathbf{w}^T \mathbf{x}_i)$ ， $\mathbf{z}^{(t)} = \mathbf{X} \mathbf{w}^{(t)} - (\mathbf{S}^{(t)})^{-1} (\mathbf{y} - \mu)$ 。这相当于一个加权的最小二乘估计，即

$$\mathbf{w}^{(t+1)} = \arg \max_{\mathbf{w}} \|(\mathbf{S}^{(t)})^{\frac{1}{2}} \mathbf{X} \mathbf{w} - (\mathbf{S}^{(t)})^{\frac{1}{2}} \mathbf{z}^{(t)}\|_2^2$$

对比最小二乘线性回归的权重求解公式

$$\mathbf{w}_{OLS} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

可以看出，Logistic回归的权重迭代相当于对数据 \mathbf{X} 施加权重 $(\mathbf{S}^{(t)})^{\frac{1}{2}}$ （OLS公式中的 \mathbf{X} 用 $(\mathbf{S}^{(t)})^{\frac{1}{2}} \mathbf{X}$ 代入， \mathbf{y} 用 $(\mathbf{S}^{(t)})^{\frac{1}{2}} \mathbf{y}$ 代入即可得到Logistic回归的权重迭代公式）

因此在每次迭代中，L1正则的Logistic回归求解可转换为一个加权的Lasso问题求解：

$$\mathbf{w}^{(t+1)} = \arg \max_{\mathbf{w}} \|(\mathbf{S}^{(t)})^{\frac{1}{2}} \mathbf{X} \mathbf{w} - (\mathbf{S}^{(t)})^{\frac{1}{2}} \mathbf{z}^{(t)}\|_2^2, \text{ s.t. } \|\mathbf{w}\|_1 \leq t$$