Logistic 回归学习笔记

李向阳　d1142845997@gmail.com

目录

1 基本 Logistic 回归

1.1 数据集

在二分类问题中, 我们的样本是 $\{\boldsymbol{x}, y\}$, 其中 y 是 0 或者 1, $y = 1$ 表示正类, $y = 0$ 表示负类 (也有些文献中用 $y = \pm 1$ 区分正负类), 而 $\boldsymbol{x}$ 表示样本的特征, 设有 n 个特征, 即 $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)^T$, 因此我们也可以将样本表示为 $\{x_1, \cdots, x_n, y\}$, 我们的期待是由这 n 个样本特征来预测样本的类别. 设训练集有 m 个样本, 即 $\{\boldsymbol{x}^{(i)}, y^{(i)}\}, i = 1, 2, \cdots, m$, 其中

$$\boldsymbol{x}^{(i)} = (x_1^{(i)}, x_2^{(i)}, \cdots, x_n^{(i)})^T, i = 1, 2, \cdots, m$$

这里要注意记号的不同, 也有些文献中把粗体的 $\boldsymbol{x}$ 用 $\mathbf{x}$ 来表示, 这里我们仍将粗体倾斜.

关于记号, 这里要多说一点, 这篇文章中我用粗体的 $\boldsymbol{x}$ 表示样本, 它有 n 个分量, 分量用下标表示, 即有 $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)^T$, 用上标表示具体的样本, 即 $\boldsymbol{x}^{(i)}$ 表示第 i 个样本, 并设总共有 m 个样本. 采用这样的记号, 个人觉得比采用双下标要好一些, 但推导时还是略显麻烦, 因此有些文献中会采用这样的记号: 用粗体的 $\boldsymbol{x}$ 表示样本, 同时用带下标 i 的 $\boldsymbol{x}_i$ 表示第 i 个样本, 注意 $\boldsymbol{x}_i$ 实际上有 n 个分量, 但为了推导方便并不写出, 直接理解为带 n 个分量的向量即可 (不过要加粗, 有些文档中甚至不加粗, 这很不好), 然后设样本个数为 N. 这也是一种常见的记号.

1.2 基本模型

所谓 Logistic 回归模型, 也就是假设样本 $\boldsymbol{x}$ 属于正类或者说对应的 $y = 1$ 的概率可以通过 sigmoid 函数来表示:

$$\begin{aligned} P(y = 1|\boldsymbol{x}; \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^T \boldsymbol{x}) = h_{\boldsymbol{\theta}}(\boldsymbol{x}) &= \frac{1}{1 + \exp(-\boldsymbol{\theta}^T \boldsymbol{x})} \\ &= \frac{1}{1 + \exp(-(\theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n))} \end{aligned} \tag{1}$$

注意这里为了记号的方便, 引入 $x_0 = 1$, 即有 $\boldsymbol{x} = (x_0, x_1, \cdots, x_n)^T$, 而 $\boldsymbol{\theta} = (\theta_0, \theta_1, \cdots, \theta_n)^T$, $\boldsymbol{\theta}$ 是模型参数, 或者称为回归系数.

以上的想法是对样本属于正类的概率进行建模, 也就是给样本 $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)^T$ 的每个特征赋予一个"权重"(可正可负), 即 $\theta_1, \theta_2, \cdots, \theta_n$, 然后计算这些特征的加权值 $\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$, 其中 θ_0 为常数. 为了使加权值能够转化为概率, 上面考虑了 sigmoid 函数, 由此导出了 Logistic 回归模型.

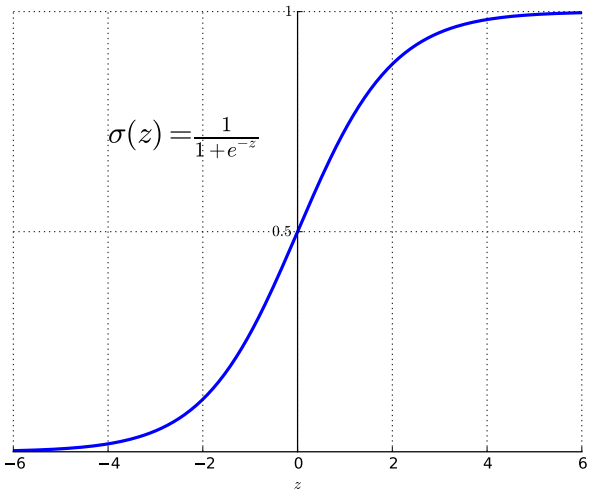


图 1: sigmoid 函数图像

此外, 我们还可以从对数几率的角度来得到 Logistic 回归模型. 一个事件的几率是指其发生的概率与不发生的概率的比值. 如果事件发生的概率是 p, 根据定义, 该事件的几率是 $\frac{p}{1-p}$, 其对数几率或 logit 函数是

$$\text{logit}(p) = \log \frac{p}{1-p} \tag{2}$$

假设样本属于正类的对数几率是样本输入特征 $\boldsymbol{x}$ 的线性函数, 即

$$\log \frac{P(y=1|\boldsymbol{x})}{1-P(y=1|\boldsymbol{x})} = \boldsymbol{\theta}^T\boldsymbol{x} = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n \tag{3}$$

化简可得式 (1), 同样得到了 Logistic 回归模型. 也就是说, 对几率做了对数变换以后, 成了线性模型. 事实上, Logistic 回归本身就是一种广义线性模型, 也可以从广义线性模型的角度推导出来 (可参考 Andrew Ng 的课件).

注 1.1. 上面已经指出, *Logistic* 回归的模型是

$$p(y=1|\boldsymbol{x}) = \frac{1}{1+\exp(-\boldsymbol{\theta}^T\boldsymbol{x})}), \ln \frac{p}{1-p} = \boldsymbol{\theta}^T\boldsymbol{x}$$

Logistic 模型也称之为 *Logit* 模型, 与之类似的有一个模型, 称为 *Probit* 模型, 其假设是

$$p(y=1|\boldsymbol{x}) = \Phi(\boldsymbol{\theta}^T\boldsymbol{x}), \Phi^{-1}(p) = \boldsymbol{\theta}^T\boldsymbol{x}$$

其中 $\Phi(\cdot)$ 是标准正态分布的累积分布函数, 由此可以看出二者的不同. 即 *Logit* 模型是借助函数 $\Phi(\cdot)$ 将 $\boldsymbol{\theta}^T\boldsymbol{x}$ 转化为了概率值, 而 *Logistic* 模型是

借助 *sigmoid* 函数将 $\boldsymbol{\theta}^T\boldsymbol{x}$ 转化为了概率值. 要注意区分. 不过, 由于只是转化函数不同, 因此二者的参数估计是很类似的. 当然, 对于 *Probit regression model,* 更常用的是借助 *EM* 算法估计其参数.

Logistic 回归的任务便是利用训练样本得到 $\boldsymbol{\theta}$ 的参数估计, 不妨记为 $\hat{\boldsymbol{\theta}}$, 于是得到预测模型为

$$P(y=1|\boldsymbol{x}) = \frac{1}{1+\exp(-\hat{\boldsymbol{\theta}}^T\boldsymbol{x})}$$

当得到一个新的样本 $\boldsymbol{x}^{(0)}$ 后, 代入上式可计算该样本属于正类的概率, 若其值大于 0.5, 我们将其归为正类, 反之则归为负类.

1.3 参数估计

这里我们采用最大似然估计方法. 注意在机器学习的一般框架中, 为了估计模型参数, 常常构造称为代价函数 (cost function) 或者损失函数 (loss function) 的东西, 然后将其极小化得到参数的估计, 比如将似然函数取负号作为代价函数, 二者的实质是一样的.

由定义可知, 对于样本 $\boldsymbol{x}$, 其为正类和负类的概率分别为

$$P(y=1|\boldsymbol{x};\boldsymbol{\theta}) = h_{\boldsymbol{\theta}}(\boldsymbol{x}), P(y=0|\boldsymbol{x};\boldsymbol{\theta}) = 1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})$$

实际上这是一个 $0-1$ 分布 (伯努利分布), 其概率函数为

$$p(y|\boldsymbol{x};\boldsymbol{\theta}) = (h_{\boldsymbol{\theta}}(\boldsymbol{x}))^y(1-h_{\boldsymbol{\theta}}(\boldsymbol{x}))^{1-y}$$

由于假定 m 个训练样本独立同分布, 故似然函数为

$$\begin{aligned} L(\boldsymbol{\theta}) &= \prod_{i=1}^m p(y^{(i)}|\boldsymbol{x}^{(i)};\boldsymbol{\theta}) \\ &= \prod_{i=1}^m (h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}))^{y^{(i)}}(1-h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}))^{1-y^{(i)}} \end{aligned}$$

其对数似然函数为

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

最大似然估计就是求使 $l(\boldsymbol{\theta})$ 取得最大值时的 $\boldsymbol{\theta}$, 此时如果用求导的办法求最大值的话, 会发现是无法求得解析解的. 当然, 我们可以用优化中的许多方法进行数值求解.

现在, 我们转向机器学习的框架来考虑参数估计. 构造关于参数 $\boldsymbol{\theta}$ 的代价函数 $J(\boldsymbol{\theta})$, 在这里, 我们将对数似然函数取个符号, 然后再除以常数 m(即为训练样本个数), 即

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

因为乘了一个负的系数 $-1/m$, 所以极小化 $J(\boldsymbol{\theta})$ 与最大化 $l(\boldsymbol{\theta})$ 是等价的.

此外, 代价函数还可由下式定义:

$$J(\boldsymbol{\theta}) = \frac{1}{m} \mathrm{Cost}(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y)$$

其中 $\mathrm{Cost}(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y)$ 为对数损失函数,

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

由此得到

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

注 1.2. 对数损失函数的一般形式为 *(可参考 http://www.csuldw.com/2016/03/26/2016-03-26-loss-function/)*

$$L(y, p(y|\boldsymbol{x})) = -\log p(y|\boldsymbol{x})$$

注 1.3. 我们可以对损失函数做进一步的化简, 即有 *(先略去常数因子 $-1/m$, 参考: http://blog.csdn.net/zouxy09/article/details/20319673)*

$$\begin{aligned} l(\boldsymbol{\theta}) &= \sum_{i=1}^{m} \left[y^{(i)} \log h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) + (1 - y^{(i)}) \log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})) \right] \\ &= \sum_{i=1}^{m} \left[y^{(i)} \log p(y^{(i)} = 1|\boldsymbol{x}^{(i)}) + (1 - y^{(i)}) \log(1 - p(y^{(i)} = 1|\boldsymbol{x}^{(i)})) \right] \\ &= \sum_{i=1}^{m} y^{(i)} \log \frac{p(y^{(i)} = 1|\boldsymbol{x}^{(i)})}{1 - p(y^{(i)} = 1|\boldsymbol{x}^{(i)})} + \sum_{i=1}^{m} \log(1 - p(y^{(i)} = 1|\boldsymbol{x}^{(i)})) \\ &= \sum_{i=1}^{m} y^{(i)} \cdot \boldsymbol{\theta}^T \boldsymbol{x}^{(i)} + \sum_{i=1}^{m} \log \frac{1}{1 + \exp(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)})} \\ &= \sum_{i=1}^{m} y^{(i)} \cdot \boldsymbol{\theta}^T \boldsymbol{x}^{(i)} - \sum_{i=1}^{m} \log(1 + \exp(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)})) \end{aligned}$$

由此可得

$$J(\boldsymbol{\theta}) = \frac{1}{m}\left[\sum_{i=1}^{m}\log(1+\exp(\boldsymbol{\theta}^T\boldsymbol{x}^{(i)})) - \sum_{i=1}^{m} y^{(i)}\cdot\boldsymbol{\theta}^T\boldsymbol{x}^{(i)}\right] \tag{4}$$

此时可直接对 $\boldsymbol{\theta}$ 求导 (下面又用了复杂的方法推导)

$$\begin{aligned}\frac{\partial J(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}} &= \frac{1}{m}\left[\sum_{i=1}^{m}\frac{\exp(\boldsymbol{\theta}^T\boldsymbol{x}^{(i)})}{1+\exp(\boldsymbol{\theta}^T\boldsymbol{x}^{(i)})}\boldsymbol{x}^{(i)} - \sum_{i=1}^{m} y^{(i)}\boldsymbol{x}^{(i)}\right]\\ &= \frac{1}{m}\sum_{i=1}^{m}\left(\sigma(\boldsymbol{\theta}^T\boldsymbol{x}^{(i)}) - y^{(i)}\right)\boldsymbol{x}^{(i)}\end{aligned}$$

如果令导数为 0, 会发现是无法得到解析解的, 所以需要用下面的迭代法进行计算.

注 1.4. 以上的讨论和损失函数都是针对样本类别标签 y 取值为 1 或 0 的情形, 有些资料中使用的是 ±1 来区分正负类, 即 y 的取值为 1 或 -1, 此时的表达方式略有不同. 这里稍微提一下.

如果用 $y=\pm1$ 区分正负类, 那么样本的概率可以表示为

$$p(y|\boldsymbol{x};\boldsymbol{\theta}) = \frac{1}{1+\exp(-y\cdot\boldsymbol{\theta}^T\boldsymbol{x})} = \frac{1}{1+\exp(-yf(\boldsymbol{x}))}$$

其中 $f(\boldsymbol{x}) = \boldsymbol{\theta}^T\boldsymbol{x}$.

类似的, 似然函数为

$$\begin{aligned}l(\boldsymbol{\theta}) &= \log\prod_{i=1}^{m} p(y^{(i)}|\boldsymbol{x}^{(i)};\boldsymbol{\theta})\\ &= \sum_{i=1}^{m}\log p(y^{(i)}|\boldsymbol{x}^{(i)};\boldsymbol{\theta})\\ &= \sum_{i=1}^{m} -\log(1+\exp(-y^{(i)}f(\boldsymbol{x}^{(i)})))\end{aligned}$$

因此损失函数为

$$J(\boldsymbol{\theta}) = -\frac{1}{m}l(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m}\log(1+\exp(-y^{(i)}f(\boldsymbol{x}^{(i)})))$$

当然, 我们也可以直接从损失函数的角度来得到这个表达式 (仍可参考: http://www.csuldw.com/2016/03/26/2016-03-26-loss-function/), 事实上, 在 Logistic 回归中, 我们采用的是对数损失函数

$$L(y, p(y|\boldsymbol{x})) = -\log p(y|\boldsymbol{x})$$

由于

$$p(y|\boldsymbol{x}) = \frac{1}{1+\exp(-yf(\boldsymbol{x}))}$$

因此损失函数为

$$L(y, p(y|\boldsymbol{x})) = \log(1+\exp(-yf(\boldsymbol{x})))$$

整个样本集合的损失函数为

$$J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} L(y^{(i)}, p(y^{(i)}|\boldsymbol{x}^{(i)})) = \frac{1}{m}\sum_{i=1}^{m}\log(1+\exp(-y^{(i)}f(\boldsymbol{x}^{(i)})))$$

即我们直接从损失函数的角度也得到了 $J(\boldsymbol{\theta})$ 的表达式.

这里额外提一点, 就是这里的 $J(\boldsymbol{\theta})$ 的表达式与上面 (4) 的表达式结果相同吗?

事实上是一样的, (4) 式可化为

$$\begin{aligned} J(\boldsymbol{\theta}) &= \frac{1}{m}\left[\sum_{i=1}^{m}\log(1+\exp(f(\boldsymbol{x}^{(i)}))) - \sum_{i=1}^{m} y^{(i)}f(\boldsymbol{x}^{(i)})\right] \\ &= \frac{1}{m}\sum_{i=1}^{m}\log\left[(1+\exp(f(\boldsymbol{x}^{(i)})))\cdot\exp(-y^{(i)}f(\boldsymbol{x}^{(i)}))\right] \end{aligned}$$

因此, 在 (4) 里面令 $y^{(i)}=1$ 与现在令里面令 $y^{(i)}=1$ 的结果相同, 在在 (4) 里面令 $y^{(i)}=0$ 与现在令里面令 $y^{(i)}=-1$ 的结果也相同. 也就是说类标记 y 用 $1,0$ 区分和用 ± 1 区分是几乎一样的.

接下来我们需要求 $J(\boldsymbol{\theta})$ 的最小值, 可以采用梯度下降法 (即最速下降法)、共轭梯度法、拟牛顿法等优化方法. 这里以梯度下降法为例进行分析, 其迭代更新公式为

$$\boldsymbol{\theta} := \boldsymbol{\theta} - \alpha\cdot\nabla J(\boldsymbol{\theta})$$

其中 α 为步长, $\boldsymbol{\theta} = (\theta_0, \theta_1, \cdots, \theta_n)^T$, 写成分量形式即为

$$\theta_j := \theta_j - \alpha\cdot\frac{\partial J(\boldsymbol{\theta})}{\partial\theta_j}$$

计算可得

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

下面说一下计算方法. 为了记号的简便, 我们用 $h'_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})$ 表示 $h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})$ 对 θ_j 求偏导, $j=0,1,\cdots,n$, 注意到

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) = \frac{1}{1+\exp(-\boldsymbol{\theta}^T\boldsymbol{x}^{(i)})} = \frac{1}{1+\exp(-(\theta_0 x_0^{(i)} + \theta_1 x_1^{(i)} + \cdots + \theta_n x_n^{(i)}))}$$

运用复合函数求导法则可得

$$\begin{aligned} h'_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) &= \frac{\partial h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})}{\partial \theta_j} = -h^2_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) \cdot \exp(-\boldsymbol{\theta}^T \boldsymbol{x}^{(i)}) \cdot (-x_j^{(i)}) \\ &= h^2_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) \left(\frac{1}{h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})} - 1 \right) x_j^{(i)} \\ &= h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})) x_j^{(i)} \end{aligned}$$

其实自己在下面推导时是可以暂时把上标 i 忽略掉的, 此外还可以得到 (求梯度与求导很相似)

$$\nabla h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) = h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}))\boldsymbol{x}^{(i)}$$

另一方面

$$\begin{aligned} \frac{\partial J(\boldsymbol{\theta})}{\partial \theta_j} &= -\frac{1}{m} \left(\sum_{i=1}^{m} y^{(i)} \cdot \frac{h'_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})}{h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})} + (1 - y^{(i)}) \cdot \frac{-h'_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})}{1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})} \right) \\ &= -\frac{1}{m} \left(\sum_{i=1}^{m} \frac{y^{(i)}(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}))h'_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) + (y^{(i)} - 1)h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})h'_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})}{h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}))} \right) \\ &= -\frac{1}{m} \left(\sum_{i=1}^{m} \frac{y^{(i)}h'_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) - y^{(i)}h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})h'_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) + y^{(i)}h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})h'_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) - h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})h'_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})}{h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}))} \right) \\ &= -\frac{1}{m} \left(\sum_{i=1}^{m} \frac{(y^{(i)} - h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})) \cdot h'_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})}{h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}))} \right) \end{aligned}$$

代入可得

$$\begin{aligned} \frac{\partial J(\boldsymbol{\theta})}{\partial \theta_j} &= -\frac{1}{m} \left(\sum_{i=1}^{m} \frac{(y - h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})) \cdot h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}))x_j^{(i)}}{h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}))} \right) \\ &= \frac{1}{m} \sum_{i=1}^{m} (h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) - y^{(i)}) x_j^{(i)} \end{aligned}$$

综上我们便得到了参数估计的迭代计算式, 即为

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

注 1.5. 这里的求导结果的形式看似和线性回归的平方损失函数求导形式一样, 但必须明确 *Logistic* 回归的损失函数是对数损失函数, 而不是如下的平方损失函数

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

最后, 考虑编程的方便, 我们需要将参数计算的过程向量化 (vectorization), 为此引入特征设计矩阵 (将 m 个样本值以行的形式并起来) 和类标签向量

$$\boldsymbol{X}=\begin{pmatrix}(\boldsymbol{x}^{(1)})^T\\(\boldsymbol{x}^{(2)})^T\\\vdots\\(\boldsymbol{x}^{(m)})^T\end{pmatrix}=\begin{pmatrix}x_0^{(1)}&x_1^{(1)}&\cdots&x_n^{(1)}\\x_0^{(2)}&x_1^{(2)}&\cdots&x_n^{(2)}\\\vdots&\vdots&\ddots&\vdots\\x_0^{(m)}&x_1^{(m)}&\cdots&x_n^{(m)}\end{pmatrix},\boldsymbol{y}=\begin{pmatrix}y^{(1)}\\y^{(2)}\\\vdots\\y^{(m)}\end{pmatrix}$$

先求 $\boldsymbol{X\theta}$, 并记为 $\boldsymbol{b}$ 作为中间量,

$$b=\boldsymbol{X\theta}=\begin{pmatrix}x_0^{(1)}&x_1^{(1)}&\cdots&x_n^{(1)}\\x_0^{(2)}&x_1^{(2)}&\cdots&x_n^{(2)}\\\vdots&\vdots&\ddots&\vdots\\x_0^{(m)}&x_1^{(m)}&\cdots&x_n^{(m)}\end{pmatrix}\begin{pmatrix}\theta_0\\\theta_1\\\vdots\\\theta_n\end{pmatrix}=\begin{pmatrix}\theta_0x_0^{(1)}+\theta_1x_1^{(1)}+\cdots+\theta_nx_n^{(1)}\\\theta_0x_0^{(2)}+\theta_1x_1^{(2)}+\cdots+\theta_nx_n^{(2)}\\\vdots\\\theta_0x_0^{(m)}+\theta_1x_1^{(m)}+\cdots+\theta_nx_n^{(m)}\end{pmatrix}$$

计算矩阵与向量的乘积或者矩阵与矩阵的乘积可以使用 Python 里 numpy 模块中的 dot 函数, 然后计算 $\boldsymbol{c}=h_{\boldsymbol{\theta}}(\boldsymbol{X})-\boldsymbol{y}=\sigma(\boldsymbol{b})-\boldsymbol{y}=\sigma(\boldsymbol{X\theta})-\boldsymbol{y}$, 即有

$$\boldsymbol{c}=\begin{pmatrix}h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(1)})-y^{(1)}\\h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(2)})-y^{(2)}\\\vdots\\h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(m)})-y^{(m)}\end{pmatrix}$$

注意到 θ_j 的迭代更新过程为

$$\begin{aligned}\theta_j&:=\theta_j-\alpha\cdot\frac{1}{m}\sum_{i=1}^{m}(h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})-y^{(i)})x_j^{(i)}\\&:=\theta_j-\alpha\cdot\frac{1}{m}\cdot(x_j^{(1)},x_j^{(2)},\cdots,x_j^{(m)})\begin{pmatrix}h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(1)})-y^{(1)}\\h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(2)})-y^{(2)}\\\vdots\\h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(m)})-y^{(m)}\end{pmatrix}\\&:=\theta_j-\alpha\cdot\frac{1}{m}\cdot(x_j^{(1)},x_j^{(2)},\cdots,x_j^{(m)})\cdot c\end{aligned}$$

综合起来就是

$$\begin{pmatrix}\theta_0\\\theta_1\\\vdots\\\theta_n\end{pmatrix}:=\begin{pmatrix}\theta_0\\\theta_1\\\vdots\\\theta_n\end{pmatrix}-\alpha\cdot\frac{1}{m}\cdot\begin{pmatrix}x_0^{(1)}&x_0^{(2)}&\cdots&x_0^{(m)}\\x_1^{(1)}&x_1^{(2)}&\cdots&x_1^{(m)}\\\vdots&\vdots&\ddots&\vdots\\x_n^{(1)}&x_n^{(2)}&\cdots&x_n^{(m)}\end{pmatrix}\cdot\boldsymbol{c}$$

写成向量形式即为

$$\boldsymbol{\theta} := \boldsymbol{\theta} - \alpha \cdot \frac{1}{m} \cdot \boldsymbol{X}^T c = \boldsymbol{\theta} - \alpha \cdot \frac{1}{m} \cdot \boldsymbol{X}^T \cdot (h_{\boldsymbol{\theta}}(\boldsymbol{X}) - \boldsymbol{y})$$

其中 $\boldsymbol{X}^T$ 表示特征矩阵 $\boldsymbol{X}$ 的转置.

1.4 决策边界

决策边界 (decision boundary) 由下式确定:

$$P(y = 1|\boldsymbol{x}; \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^T \boldsymbol{x}) = 0.5 \Rightarrow \boldsymbol{\theta}^T \boldsymbol{x} = 0$$

写成分量形式即为

$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n = 0$$

这表示 n 为空间中的一个超平面 (hyper plane), 为了进行直观展示, 我们以二维为例, 此时超平面退化为一条直线, 即有

$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 = 0 \Rightarrow x_2 = -\frac{\theta_1}{\theta_2} x_1 - \frac{\theta_0}{\theta_2}$$

以数据集 ex2data1.txt 为例, 即有

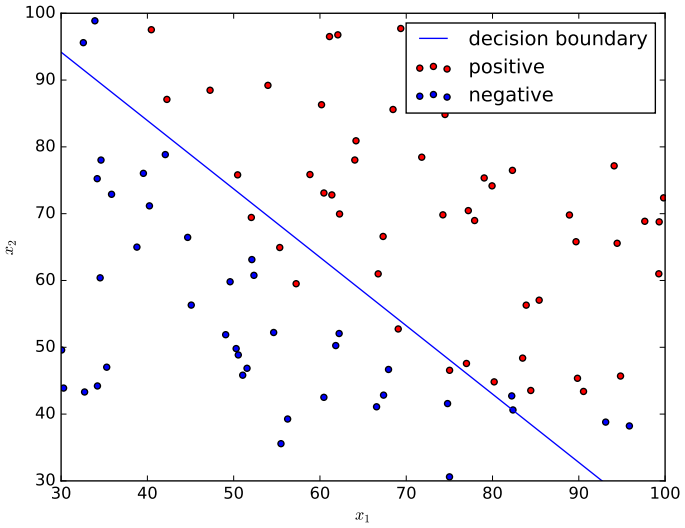


图 2: 线性决策边界

因此, 从另一个角度看, 进行 Logistic 回归, 实际上就是在求这个决策边界的回归方程, 或者说画一个超平面把两类点尽量地区分开来.

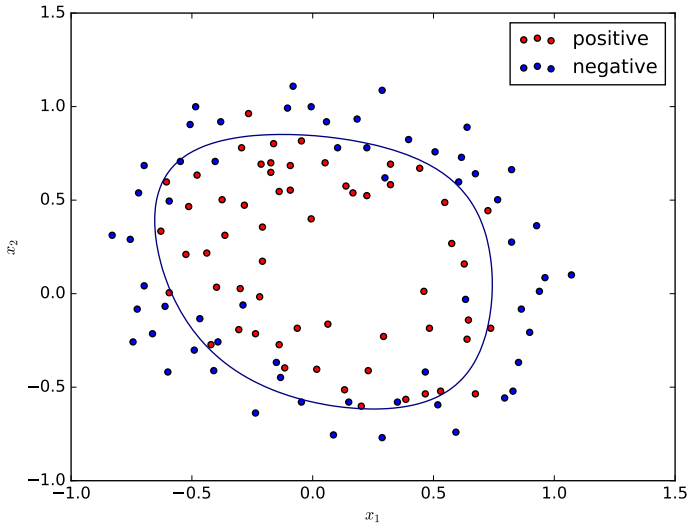


图 3: 非线性决策边界

不过 Logistic 回归的威力不止于此, 如同线性回归一样, 它可以处理更复杂的模型, 比如对数据集 ex2data2.txt, 有

事实上, 如同线性回归一样, 模型看似是关于特征变量 $x_1, x_2, \cdots, x_n$ 线性的, 这里"线性"的本质是关于参数 $\theta_0, \theta_1, \cdots, \theta_n$ 的线性函数, 比如模型如果长这样:

$$\theta_0 + \theta_1 x_1^2 + \theta_2 x_1 x_2 + \theta_3 x_2^4 + \theta_4 x_1^3 x_2^3$$

构造新的变量, 令

$$z_1 = x_1^2, z_2 = x_1 x_2, z_3 = x_2^4, z_4 = x_1^3 x_2^3$$

则模型就变为了关于 $\boldsymbol{z} = (z_0, z_1, \cdots, z_4)^T$ 的线性模型了, 其中 $z_0 = 1$.

$$\boldsymbol{\theta}^T \boldsymbol{z} = \theta_0 + \theta_1 z_1 + \theta_2 z_2 + \theta_3 z_3 + \theta_4 z_4$$

以数据集 ex2data2.txt 为例, 虽然只有 x_1, x_2 两个特征, 但我们可以将其映射到一个高维空间中, 不妨将其称为 MapFeature, 假设映射到 6 次多项式, 即有

$$\text{MapFeature}(\boldsymbol{x}) = (1, x_1, x_2, x_1^2, x_1 x_2, x_2^2, \cdots, x_2^6)$$

这样总共就有 $1 + 2 + 3 + \cdots + 7 = 28$ 个特征, 于是可令 $z_0 = 1, z_1 = x_1, \cdots, z_{28} = x_2^6$, 再应用 Logistic 回归模型即可, 求得回归参数

$\theta_0, \theta_1, \cdots, \theta_{28}$ 以后, 决策边界的曲线方程为

$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1 x_2 + \theta_5 x_2^2 + \cdots + \theta_{28} x_2^6 = 0$$

这是一个隐函数的图像, 可以利用 ContourPlot 函数画出来.

模型越复杂, 相当于曲线越复杂, 理论上越能把两类点区分开来, 但是如同线性回归一样, 复杂的模型常常产生过拟合, 这就需要用下面的正则化来解决.

2 正则化

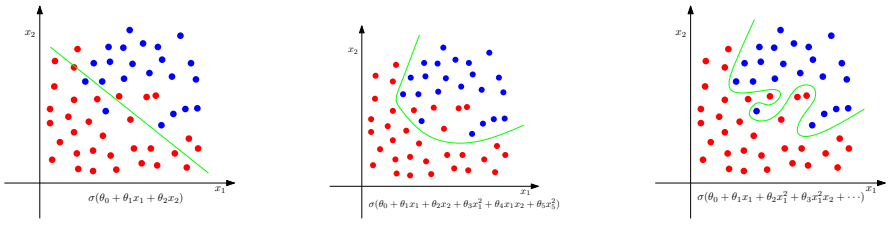
2.1 过拟合与正则化

Logistic 回归也有过拟合 (overfitting) 的问题, 一个复杂的模型可以保证训练误差非常小, 但是其测试误差和预测误差可能非常大, 这样模型就没有很好的预测能力. 为了防止过拟合, 我们通过施加惩罚项来解决这个问题. 如同线性回归一样, 参数 $\boldsymbol{\theta}$ 反映了模型的复杂度, 因此可对参数 $\boldsymbol{\theta}$ 施加惩罚, 比如, 代价函数为

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

注意我们不需要对 θ_0 做惩罚, 所以上式右边的求和中 j 是从 1 到 n, 而不是从 0 到 n(虽然最后实际计算时差别影响不大). 同理可以求出导数为

$$\begin{aligned}
\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_0} &= \frac{1}{m}\sum_{i=1}^{m}(h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) - y^{(i)})x_j^{(i)}, && j = 0 \\
\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_j} &= \frac{1}{m}\left(\sum_{i=1}^{m}(h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) - y^{(i)})x_j^{(i)} + \lambda\theta_j\right), && j \geqslant 1
\end{aligned}$$



(a) 欠拟合 (b) 正常 (c) 过拟合

图 4: 拟合情况

前面提到过, 在未正则化时, 最大似然估计与构造代价函数本质上是一样的, 求得的参数估计均为

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{m} -\log p(y^{(i)}|\boldsymbol{x}^{(i)};\boldsymbol{\theta})$$

略去常数因子, 我们上面对参数 $\boldsymbol{\theta}$ 施加惩罚所得到的估计实际为

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \left(\sum_{i=1}^{m} -\log p(y^{(i)}|\boldsymbol{x}^{(i)};\boldsymbol{\theta}) + \lambda||\boldsymbol{\theta}||_2 \right)$$

其中 $||\boldsymbol{\theta}||_2 = \sum_{j=1}^{n} \theta_j^2$, 求和指标 j 从 1 开始.

对数据集 ex2data2.txt 来说, 图3正是用 MapFeature 结合正则化的方法画出来的.

也就是说, 上面实际上是 L_2 正则化, 当然, 我们也可以进行 L_1 正则化, 即有

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \left(\sum_{i=1}^{m} -\log p(y^{(i)}|\boldsymbol{x}^{(i)};\boldsymbol{\theta}) + \lambda||\boldsymbol{\theta}||_1 \right)$$

2.2 统计学角度看问题

以上大都是从机器学习的角度来看问题, 主要的目的是估计出参数 $\boldsymbol{\theta}$, 至于预测效果与模型选择等, 可由训练误差与测试误差决定.

大多数文献也都止步于此. 然而, 从经典统计学的角度看, 参数 $\boldsymbol{\theta}$ 是一个未知的固定常数, 但是估计量 $\hat{\boldsymbol{\theta}}$ 是一个随机变量, 是随机样本的函数, 因此, 也可以如同线性回归一样, 进一步分析估计量的统计性质, 比如分布等等, 进而寻求参数的区间估计, 或者对我们的结果进行假设检验, 比如模型系数的显著性检验, 以及利用 AIC 准则或者 BIC 准则进行模型选择等等. 这些确实也是可以做到的, 只不过不同与线性回归, 在 Logistic 回归中, 我们并没有得到估计量 $\hat{\boldsymbol{\theta}}$ 的解析表达式, 因此进行这些工作会比较困难. 不过, 在 R 语言和 Python 的 statmodels 包中, 可以给出相应的结果 (注意在 Python 的 sklearn 包中, 是没有系数的显著性的).

另外一方面, 从贝叶斯统计的角度看, 一切未知量都是随机变量, 即参数 $\boldsymbol{\theta}$ 不是一个未知的常数, 它也是一个随机变量, 只要我们给它一个合适的先验分布, 然后计算出后验分布, 也可进行各种统计推断, 而且往往比经典统计学更合理一些, 比如上面的最大似然估计会过拟合, 因此需要加上惩罚项来正则化, 而我们很容易证明, 假如 $\boldsymbol{\theta}$ 的先验分布是 Laplace 分布, 即 $p(\boldsymbol{\theta}) = (\lambda/2)^n \exp(-\lambda||\boldsymbol{\theta}||_1)$, 其中常数 $\lambda > 0$, 则 $\boldsymbol{\theta}$ 的最大后验估计由下式

给出:

$$\min_{\boldsymbol{\theta}} \left(\sum_{i=1}^{m} -\log p(y^{(i)}|\boldsymbol{x}^{(i)};\boldsymbol{\theta}) + \lambda||\boldsymbol{\theta}||_1 \right)$$

这恰好即为 L_1 正则化. 因此, 运用贝叶斯统计的观点来看问题, 能自动避免过拟合, 接下来我们就来看如何用贝叶斯方法进行 Logistic 回归.

3 贝叶斯 Logistic 回归

在贝叶斯统计中, 关键是要求出后验分布, 一般利用核方法, 即

$$p(\boldsymbol{\theta}|\boldsymbol{x}) \propto p(\boldsymbol{\theta}) \cdot p(\boldsymbol{x}|\boldsymbol{\theta}) = p(\boldsymbol{\theta}) \cdot L(\boldsymbol{\theta}|\boldsymbol{x}) = K(\boldsymbol{\theta})$$

其中 $L(\boldsymbol{\theta}|\boldsymbol{x})$ 就是样本似然函数 $L(\boldsymbol{\theta})$, 最大后验估计即要使 $K(\boldsymbol{\theta}) = p(\boldsymbol{\theta}) \cdot L(\boldsymbol{\theta}|\boldsymbol{x})$ 最大. 注意在 Logistic 回归中, 这里的 $p(\boldsymbol{\theta}|\boldsymbol{x})$ 应当理解为 $p(\boldsymbol{\theta}|\boldsymbol{x}, y)$, 因为本质上 $\boldsymbol{x}, y$ 合在一起是一组样本.

前面已经求得

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = L(\boldsymbol{\theta}|\boldsymbol{x}) = L(\boldsymbol{\theta}) = \prod_{i=1}^{m} p(y^{(i)}|\boldsymbol{x}^{(i)};\boldsymbol{\theta})$$

下面来看 $\boldsymbol{\theta}$ 的先验分布 $p(\boldsymbol{\theta})$.

假定 $\boldsymbol{\theta}$ 的先验分布为 Laplace 分布, 即

$$p(\boldsymbol{\theta}) = \left(\frac{\lambda}{2}\right)^n \mathrm{e}^{-\lambda||\boldsymbol{\theta}||_1}$$

其中 $\lambda > 0$ 为常数, 则有

$$K(\boldsymbol{\theta}) = p(\boldsymbol{\theta}) \cdot p(\boldsymbol{x}|\boldsymbol{\theta}) = \left(\frac{\lambda}{2}\right)^n \mathrm{e}^{-\lambda||\boldsymbol{\theta}||_1} \cdot \prod_{i=1}^{m} p(y^{(i)}|\boldsymbol{x}^{(i)};\boldsymbol{\theta})$$

对 $K(\boldsymbol{\theta})$ 取负对数, 可得

$$-\log K(\boldsymbol{\theta}) = -n \log \frac{\lambda}{2} + \lambda||\boldsymbol{\theta}||_1 - \sum_{i=1}^{m} \log p(y^{(i)}|\boldsymbol{x}^{(i)};\boldsymbol{\theta})$$

极大化 $K(\boldsymbol{\theta})$ 与极小化 $-\log K(\boldsymbol{\theta})$ 是一样的, 即等价于

$$\min_{\boldsymbol{\theta}} \left(\sum_{i=1}^{m} -\log p(y^{(i)}|\boldsymbol{x}^{(i)};\boldsymbol{\theta}) + \lambda||\boldsymbol{\theta}||_1 \right)$$

这就是 L_1 正则化, 于是我们就得到了前面所说的结果.

此外, 如果假定 $\boldsymbol{\theta}$ 的先验分布为

$$\theta_j \sim \mathcal{N}(0, s^2) = \frac{1}{\sqrt{2\pi}s} \mathrm{e}^{-\frac{\theta_j^2}{2s^2}}, j = 1, 2, \cdots, n$$

于是可得 $\boldsymbol{\theta}$ 的先验分布为

$$p(\boldsymbol{\theta}) = \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi}s} \mathrm{e}^{-\frac{\theta_j^2}{2s^2}} = \left(\frac{1}{\sqrt{2\pi}s}\right)^n \mathrm{e}^{-\frac{\boldsymbol{\theta}^T\boldsymbol{\theta}}{2s^2}}$$

进一步有

$$K(\boldsymbol{\theta}) = p(\boldsymbol{\theta}) \cdot p(\boldsymbol{x}|\boldsymbol{\theta}) = \left(\frac{1}{\sqrt{2\pi}s}\right)^n \mathrm{e}^{-\frac{\boldsymbol{\theta}^T\boldsymbol{\theta}}{2s^2}} \cdot \prod_{i=1}^{m} p(y^{(i)}|\boldsymbol{x}^{(i)};\boldsymbol{\theta})$$

还是对 $K(\boldsymbol{\theta})$ 取负对数, 可得极小化 $-\log K(\boldsymbol{\theta})$ 即为

$$\min_{\boldsymbol{\theta}} \left(\sum_{i=1}^{m} -\log p(y^{(i)}|\boldsymbol{x}^{(i)};\boldsymbol{\theta}) + \frac{\boldsymbol{\theta}^T\boldsymbol{\theta}}{2s^2} \right)$$

注意到 $\boldsymbol{\theta}^T\boldsymbol{\theta} = ||\boldsymbol{\theta}||_2^2$, 因此这相当于 L^2 正则化.

以上正态分布的假定相当于 $\boldsymbol{\theta} \sim \mathcal{N}(\mathbf{0}, s^2\boldsymbol{I})$. 当然, 我们也可以假定 $\boldsymbol{\theta}$ 的先验分布为其他的形式, 比如 $\boldsymbol{N(\mu, S)}$, 只不过处理会稍微复杂些, 这里就先不讨论了. 此外, 这里假定超参数比如 s 是已知的, 如果 s 未知, 我们也可以给 s 一个分布, 只不过讨论会更麻烦些.

4 变分贝叶斯 Logistic 回归

4.1 变分贝叶斯一般框架

在变分贝叶斯中, 我们将所有观测到的样本记为 $\boldsymbol{X}$, 不妨设有 m 个样本, 即有 $\boldsymbol{X} = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(m)}\}$, 把所有的未知参数和隐变量 (latent variable) 记为 $\boldsymbol{Z}$, 我们的目的是要估计出后验分布 $p(\boldsymbol{Z}|\boldsymbol{X})$, 有了这个后验分布, 我们就可以估计出未知参数了.

但是解析的算出后验分布 $p(\boldsymbol{Z}|\boldsymbol{X})$ 的表达式是几乎不可能的, 我们考虑用一个分布 $q(\boldsymbol{Z})$ 去近似它. 为此先来看一个关系.

$$\begin{aligned}
\ln p(\boldsymbol{X}) &= \ln \frac{p(\boldsymbol{X},\boldsymbol{Z})}{p(\boldsymbol{Z}|\boldsymbol{X})} = \int q(\boldsymbol{Z}) \ln \frac{p(\boldsymbol{X},\boldsymbol{Z})}{p(\boldsymbol{Z}|\boldsymbol{X})} \,\mathrm{d}\boldsymbol{Z} \\
&= \int q(\boldsymbol{Z}) \ln \left(\frac{p(\boldsymbol{X},\boldsymbol{Z})}{p(\boldsymbol{Z}|\boldsymbol{X})} \cdot \frac{q(\boldsymbol{Z})}{q(\boldsymbol{Z})} \right) \mathrm{d}\boldsymbol{Z} \\
&= \int q(\boldsymbol{Z}) \left(\ln \frac{q(\boldsymbol{Z})}{p(\boldsymbol{Z}|\boldsymbol{X})} + \ln \frac{p(\boldsymbol{X},\boldsymbol{Z})}{q(\boldsymbol{Z})} \right) \mathrm{d}\boldsymbol{Z} \\
&= \int q(\boldsymbol{Z}) \cdot \ln \frac{q(\boldsymbol{Z})}{p(\boldsymbol{Z}|\boldsymbol{X})} \,\mathrm{d}\boldsymbol{Z} + \int q(\boldsymbol{Z}) \cdot \ln \frac{p(\boldsymbol{X},\boldsymbol{Z})}{q(\boldsymbol{Z})} \,\mathrm{d}\boldsymbol{Z} \\
&= -\int q(\boldsymbol{Z}) \cdot \ln \frac{p(\boldsymbol{Z}|\boldsymbol{X})}{q(\boldsymbol{Z})} \,\mathrm{d}\boldsymbol{Z} + \int q(\boldsymbol{Z}) \cdot \ln \frac{p(\boldsymbol{X},\boldsymbol{Z})}{q(\boldsymbol{Z})} \,\mathrm{d}\boldsymbol{Z} \\
&= \mathrm{KL}(q||p) + \mathcal{L}(q)
\end{aligned}$$

其中 $\mathrm{KL}(q||p)$ 表示 $q(\boldsymbol{Z})$ 与 $p(\boldsymbol{Z}|\boldsymbol{X})$ 间的 KL 散度.

$$\mathrm{KL}(q||p) = -\int q(\boldsymbol{Z}) \cdot \ln \frac{p(\boldsymbol{Z}|\boldsymbol{X})}{q(\boldsymbol{Z})} \,\mathrm{d}\boldsymbol{Z} \geqslant 0$$

从这个角度看, $\mathrm{KL}(q||p)$ 反映了我们用 $q(\boldsymbol{Z})$ 去近似 $p(\boldsymbol{Z}|\boldsymbol{X})$ 的差距, 这个值越小越好, 而且我们知道 $\mathrm{KL}(q||p) = 0 \Longleftrightarrow q(\boldsymbol{Z}) = p(\boldsymbol{Z}|\boldsymbol{X})$.

在这里, 我们是通过不断最大化 $\mathcal{L}(q)$ 来达到相同的目的, 而 $\mathcal{L}(q)$ 刚好是一个函数 $q(\boldsymbol{Z})$ 的泛函, 变分的名称由此得来.

如同一般的变分法, 这里要对 $q(\boldsymbol{Z})$ 做某些要求, 相当于把 $q(\boldsymbol{Z})$ 限制到某个空间中. 假设 $\boldsymbol{Z}$ 可独立成 M 个部分 $Z_i, i = 1, 2, \cdots, M$, 那么最常用的一个要求是因子分解分布, 即有

$$q(\boldsymbol{Z}) = \prod_{i=1}^{M} q_i(Z_i)$$

在此限定条件下我们来求 $\mathcal{L}(q)$ 的最大值, 总体思路是把 $\mathcal{L}(q)$ 拆分成独立的 $q_j(Z_j)$, 从而对每一项求最大值, 以下推导中将 $q_j(Z_j)$ 简记为 q_j, 于是有

$$\begin{aligned}
\mathcal{L}(q) &= \int q(\boldsymbol{Z}) \cdot \ln \frac{p(\boldsymbol{X}, \boldsymbol{Z})}{q(\boldsymbol{Z})} \,\mathrm{d}\boldsymbol{Z} \\
&= \int \prod_i q_i(Z_i) \left(\ln p(\boldsymbol{X}, \boldsymbol{Z}) - \ln \prod_i q_i(Z_i) \right) \mathrm{d}\boldsymbol{Z} \\
&= \int \prod_i q_i \left(\ln p(\boldsymbol{X}, \boldsymbol{Z}) - \sum_i \ln q_i \right) \mathrm{d}\boldsymbol{Z} \\
&= \int \prod_i q_i \ln p(\boldsymbol{X}, \boldsymbol{Z}) \prod_i \mathrm{d}Z_i - \sum_i \int \prod_j q_j \ln q_i \,\mathrm{d}Z_i \\
&= \int q_j \left(\ln p(\boldsymbol{X}, \boldsymbol{Z}) \prod_{i \neq j} (q_i \,\mathrm{d}Z_i) \right) \mathrm{d}Z_j - \int q_j \ln q_j \,\mathrm{d}Z_j - \sum_{i \neq j} \int q_i \ln q_i \,\mathrm{d}Z_i \\
&= \int q_j \ln \tilde{p}(\boldsymbol{X}, Z_j) \,\mathrm{d}Z_j - \int q_j \ln q_j \,\mathrm{d}Z_j - \sum_{i \neq j} \int q_i \ln q_i \,\mathrm{d}Z_i \\
&= -\int q_j \ln \frac{\tilde{p}(\boldsymbol{X}, Z_j)}{q_j} \,\mathrm{d}Z_j - \sum_{i \neq j} \int q_i \ln q_i \,\mathrm{d}Z_i \\
&= -\mathrm{KL}(q_j||\tilde{p}) - \sum_{i \neq j} \int q_i \ln q_i \,\mathrm{d}Z_i
\end{aligned}$$

其中

$$\ln \tilde{p}(\boldsymbol{X}, Z_j) = \int \ln p(\boldsymbol{X}, \boldsymbol{Z}) \prod_{i \neq j} (q_i \,\mathrm{d}Z_i) = \mathbb{E}_{i \neq j}[\ln p(\boldsymbol{X}, \boldsymbol{Z})]$$

显然, 当 $\mathrm{KL}(q_j||\tilde{p})=0$ 时上式取得最大值, 也即 $q_j(Z_j)=\tilde{p}(\boldsymbol{X},Z_j)$, 因此最优分布 $q_j^*(Z_j)$ 即为

$$\ln q_j^*(Z_j)=\ln\tilde{p}(\boldsymbol{X},Z_j)=\mathbb{E}_{i\neq j}[\ln p(\boldsymbol{X},\boldsymbol{Z})]+\text{const}$$

这里的常数是为了保证分布 $q_j^*(Z_j)$ 的正则性, 直接进行归一化可得

$$q_j^*(Z_j)=\frac{\exp(\mathbb{E}_{i\neq j}[\ln p(\boldsymbol{X},\boldsymbol{Z})])}{\displaystyle\int\exp(\mathbb{E}_{i\neq j}[\ln p(\boldsymbol{X},\boldsymbol{Z})])\,\mathrm{d}Z_j}$$

这便是最优分布的表达式.

关于变分贝叶斯还有很多东西, 这里就不再讨论了.

4.2 变分贝叶斯 Logistic 回归

现在, 我们来运用变分贝叶斯方法进行 Logistic 回归.

注 4.1. 这里推导的结果的表达式仍然是以类标签 y 取值为 1 或 0 的, 这与 *PRML* 上保持的是一致的, 不过推导结果是借鉴的 *Jan Drugowitsch* 的文献, 文献上用的类标签是 ±1, 不过仔细推导可以发现, 二者其实也是一致的, 在得出 $h(\boldsymbol{\theta},\boldsymbol{\xi})$ 的时候其表达式实质上就是相同的. 事实上, 如果用 $y=1,0$ 作为类标签, 那么转化关系 $y^*=2\cdot y-1$, 就将 y^* 化为了 ±1.

设参数 $\boldsymbol{\theta}$ 的先验分布为依赖于超参数 α 的正态分布, 即 $\boldsymbol{\theta}\sim\mathcal{N}(\mathbf{0},\alpha^{-1}\boldsymbol{I})$

$$p(\boldsymbol{\theta}|\alpha)=\left(\frac{\alpha}{2\pi}\right)^{n/2}\exp\left(-\frac{\alpha}{2}\boldsymbol{\theta}^T\boldsymbol{\theta}\right)$$

而 α 服从伽马分布 $\alpha\sim\mathrm{Gam}(a_0,b_0)$, 即有

$$p(\alpha)=\frac{1}{\Gamma(a_0)}b_0^{a_0}\alpha^{a_0-1}\exp(-b_0\alpha)$$

我们的目的是用一个分布 $q(\boldsymbol{\theta},\alpha)$ 去近似后验分布, 首先假设 $q(\boldsymbol{\theta},\alpha)=q(\boldsymbol{\theta})q(\alpha)$, 不过将变分贝叶斯运用到 Logistic 回归中还是会有些困难, 接下来将直接给出结果. 可以求得

$$\begin{aligned}\ln q^*(\boldsymbol{\theta})&=\boldsymbol{\theta}^T\sum_i\frac{2y^{(i)}-1}{2}\boldsymbol{x}^{(i)}-\frac{1}{2}\boldsymbol{\theta}^T\left(\mathbb{E}_\alpha(\alpha)\boldsymbol{I}+2\sum_i\lambda(\xi_i)\boldsymbol{x}^{(i)}(\boldsymbol{x}^{(i)})^T\right)\boldsymbol{\theta}+\text{const}\\&=\ln\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\theta}_m,\boldsymbol{S}_m)\end{aligned}$$

其中

$$\boldsymbol{S}_m^{-1} = \mathbb{E}_\alpha(\alpha)\boldsymbol{I} + 2\sum_i \lambda(\xi_i)\boldsymbol{x}^{(i)}(\boldsymbol{x}^{(i)})^T$$

$$\boldsymbol{\theta}_m = \boldsymbol{S}_m \sum_i \frac{2y^{(i)}-1}{2}\boldsymbol{x}^{(i)}$$

$$\lambda(\xi) = \frac{1}{2\xi}\left(\sigma(\xi) - \frac{1}{2}\right)$$

类似地, 对于 α 有

$$\begin{aligned}\ln q^*(\alpha) &= \left(a_0 - 1 + \frac{n}{2}\right)\ln\alpha - \left(b_0 + \frac{1}{2}\mathbb{E}_{\boldsymbol{\theta}}(\boldsymbol{\theta}^T\boldsymbol{\theta})\right)\alpha + \text{const} \\ &= \ln \text{Gam}(\alpha|a_m, b_m)\end{aligned}$$

其中

$$a_m = a_0 + \frac{n}{2} \qquad b_m = b_0 + \frac{1}{2}\mathbb{E}_{\boldsymbol{\theta}}(\boldsymbol{\theta}^T\boldsymbol{\theta})$$

上述公式中的期望可由下式计算

$$\mathbb{E}_\alpha(\alpha) = \frac{a_m}{b_m} \qquad \mathbb{E}_{\boldsymbol{\theta}}(\boldsymbol{\theta}^T\boldsymbol{\theta}) = \boldsymbol{\theta}_m^T\boldsymbol{\theta}_m + \text{Tr}(\boldsymbol{S}_m)$$

可以看到, 不管是计算 $\boldsymbol{\theta}$ 还是 α 都要依赖地用到对方的信息, 因此可用 EM 算法来进行计算, 其中 ξ 的更新过程为

$$\xi_i^{\text{new}} = \sqrt{(\boldsymbol{x}^{(i)})^T(\boldsymbol{S}_m + \boldsymbol{\theta}_m\boldsymbol{\theta}_m^T)\boldsymbol{x}^{(i)}}$$

最终由 EM 算法即可算得参数 $\boldsymbol{\theta}$ 的值. 此外, 运用变分贝叶斯还可以直接对概率分布作预测, 这里也就不再讨论了.

5 总结

5.1 参考资料

(1) 博客: http://aimotion.blogspot.lt/2011/11/machine-learning-with-python-logistic.html, 有 Python 编程, 但是原理推导不够详细.

(2) 博客: https://www.kunxi.org/notes/Machine_Learning/Logistic_Regression/, 有 Python 编程和部分原理推导

(3) R-bloggers: http://www.r-bloggers.com/interactive-visualization-of-non-linear-logist
一个交互式绘出决策边界的 R 语言实现

(4) StackOverflow: http://stackoverflow.com/questions/22294241/plotting-a-decision-bound
运用 sklearn 画出决策边界

(5) R-bloggers: http://www.r-bloggers.com/machine-learning-ex-5-2-regularized-logistic-
正则化 Logistic 回归的 R 实现

(6) CSDN: http://blog.csdn.net/dongtingzhizi/article/details/15962797
推导很详细

(7) CSDN: http://blog.csdn.net/zouxy09/article/details/20319673
写的较为通俗

(8) 公开课材料: http://openclassroom.stanford.edu/MainFolder/DocumentPage.
php?course=MachineLearning&doc=exercises/ex5/ex5.html Matlab
代码实现

附录