# Lecture 10　Logistic Regression

整理者: LobbyBoy* 2020年2月24日

## 1. Logistic Regression Problem

### 1) 概述

考虑“二元分类+有noise+选择0/1误差作为错误度量”的情形下，我们的ideal-mini target是：

$$f(\mathbf{x}) = \underset{y\in\{+1,-1\}}{argmax}\ \ P(y|\mathbf{x})$$

或者借助sign function写作：

$$f(\mathbf{x}) = sign\Big(P(+1|\mathbf{x}) - \frac{1}{2}\Big)$$

我们希望通过某个hypothesis $g$去逼近ideal-mini target $f(\mathbf{x})$，这个hypothesis的输出要么是+1，要么是-1，所以叫做“二元分类”，是一种“硬”(hard)分类。例如，输入某个病人的病理资料，输出的是该病人“是否”患病。然而，很多时候我们并不想仅仅拿到这种“硬硬的”分类结果，而是希望拿到一个位于0到1之间的数，告诉我们患病的“可能性”是多少。也就是说，我们希望能够学到条件分布$P(+1|\mathbf{x})$，而不是ideal-mini target。这种问题我们将其称为“软二元分类”(soft binary classification)。In summary, 普通二元分类回答的是“是非”问题，输出空间为$\{-1,+1\}$；软性二元分类回答的是“有多少可能性”的问题，输出空间是$[0,1]$。

### 2) 假说的形式

我们依然采用对输入进行线性加权平均的形式构造hypothesis。然而，$\mathbf{w}^T\mathbf{x} \in R$，但我们希望的输出表示一个概率值，应该位于$[0,1]$。因此，我们引入sigmoid函数(又称为logistic函数)$\theta(s)$：

*本笔记根据台湾大学林轩田教授于线上教育平台Coursera开设的“机器学习基石”课程整理而成(课程内容见: https://www.coursera.org/learn/ntumlone-mathematicalfoundations/home/welcome)。笔记内的大多数图片来自于林老师的课程slides。感谢林老师能够将如此精彩的课程通过线上平台同所有人分享，thanks!

$$\theta(s) = \frac{1}{1+e^{-s}}$$

其图像为：

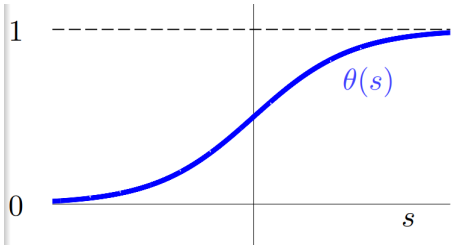


图 1: sigmoid函数

我们将加权值$\mathbf{w}^T\mathbf{x}$传入sigmoid函数便可以得到一个0至1间的值。因此，我们的hypothesis为：

$$h(\mathbf{x}) = \theta\Big(\mathbf{w}^T\mathbf{x}\Big) = \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}$$

## 2. Logistic Regression Error

下面我们将通过最大似然估计法(MLE)定义软二元分类问题中错误度量的方式。注意到，我们有假设：

$$P(+1|\mathbf{x}) = h(\mathbf{x};\mathbf{w}) = \theta\Big(\mathbf{w}^T\mathbf{x}\Big)$$

从统计学出发，我们面临的问题是：利用样本(训练数据)$\mathcal{D} = \Big((\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), \cdots, (\mathbf{x_N}, y_N)\Big)$估计参数$\mathbf{w}$。构造似然函数：

$$
\begin{aligned}
L(\mathbf{w}|\mathcal{D}) &= P(\mathbf{x_1}, y_1) \cdot P(\mathbf{x_2}, y_2) \cdot \cdots \cdot P(\mathbf{x_N}, y_N) \\
&= \prod_{n=1}^{N} P(y_n|\mathbf{x_n}) P(\mathbf{x_n}) \\
&= \prod_{n=1}^{N} P(y_n|\mathbf{x_n}) \prod_{n=1}^{N} P(\mathbf{x_n}) \\
&= \prod_{n_1=1}^{N_1} P(+1|\mathbf{x_{n_1}}) \prod_{n_2=1}^{N_2} P(-1|\mathbf{x_{n_2}}) \prod_{n=1}^{N} P(\mathbf{x_n}) \\
&= \prod_{n_1=1}^{N_1} P(+1|\mathbf{x_{n_1}}) \prod_{n_2=1}^{N_2} \Big(1 - P(+1|\mathbf{x_{n_2}})\Big) \prod_{n=1}^{N} P(\mathbf{x_n}) \\
&= \prod_{n_1=1}^{N_1} h(\mathbf{x_{n_1}}; \mathbf{w}) \prod_{n_2=1}^{N_2} \Big(1 - h(\mathbf{x_{n_2}}; \mathbf{w})\Big) \prod_{n=1}^{N} P(\mathbf{x_n})
\end{aligned}
$$

注意到：

$$h(\mathbf{x}; \mathbf{w}) + h(-\mathbf{x}; \mathbf{w}) = 1$$

因此：

$$
\begin{aligned}
L(\mathbf{w}|\mathcal{D}) &= \prod_{n_1=1}^{N_1} h(\mathbf{x_{n_1}}; \mathbf{w}) \prod_{n_2=1}^{N_2} h(-\mathbf{x_{n_2}}; \mathbf{w}) \prod_{n=1}^{N} P(\mathbf{x_n}) \\
&= \prod_{n=1}^{N} h(y_n \mathbf{x_n}; \mathbf{w}) \prod_{n=1}^{N} P(\mathbf{x_n})
\end{aligned}
$$

因为：①$\prod_{n=1}^{N} P(\mathbf{x_n})$相当于常数；②取自然对数不影响optimal solution；③max等价于min加上负号。因此$max\ L(\mathbf{w}|\mathcal{D})$等价于：

$$\min_{\mathbf{w}} \quad -ln\Big(\prod_{n=1}^{N} h(y_n \mathbf{x_n}; \mathbf{w})\Big)$$

化简得到：

$$\min_{\mathbf{w}} \quad \sum_{n=1}^{N} ln\Big(1 + exp(-y_n \mathbf{w}^T \mathbf{x_n})\Big)$$

我们定义：

$$
\begin{aligned}
E_{in}^{CE}(\mathbf{w}) &= \frac{1}{N} \sum_{n=1}^{N} ln\Big(1 + exp(-y_n \mathbf{w}^T \mathbf{x_n})\Big) \\
&= \frac{1}{N} \sum_{n=1}^{N} err(\mathbf{w}, \mathbf{x_n}, y_n)
\end{aligned}
$$

并将$err(\mathbf{w}, \mathbf{x}, y) = ln(1 + exp(-y\mathbf{w}^T\mathbf{x}))$称为“交叉熵”(cross-entropy)。

## 3. Gradient of LR Error & Gradient Descent

上一节我们得到了软二元分类问题中经验误差的形式，因此我们现在的问题转化为如何最小化以交叉熵为错误度量的经验误差：

$$\min_{\mathbf{w}} \quad \frac{1}{N}\sum_{n=1}^{N} ln\Big(1 + exp(-y_n\mathbf{w}^T\mathbf{x_n})\Big)$$

可以证明$E_{in}^{CE}(\mathbf{w})$是凸函数，因此尝试令梯度为零：

$$\begin{aligned}\nabla_{\mathbf{w}} &= \frac{1}{N}\sum_{n=1}^{N}\Big[\theta(-y_n\mathbf{w}^T\mathbf{x_n})\cdot(-y_n\mathbf{x_n})\Big] \\ &= 0\end{aligned}$$

然而，该方程并没有像线性回归那样的closed-form solution。回忆PLA演算法：PLA从任意起点开始，通过不断地修正，渐渐得到最优解。这种最优化的方法称为iterative optimization approach，我们可以将其概括为：

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta\cdot\mathbf{v}$$

其中，$\eta > 0$表示“步长”(step)，$\mathbf{v} : ||\mathbf{v}|| = 1$表示“方向”(direction)。上式表示，我们将从某一个$\mathbf{w}_0$进行一系列迭代，每一次迭代都是向某一方向$\mathbf{v}$走$\eta$步。这里我们将“一步”分解称为了“方向”与“步长”，因此会有方向向量长度为1、步长大于0这两个条件。假设我们现在位于$\mathbf{w}_t$，首先将步长看作是确定的，来决定方向——方向是下降最多的方向：

$$\min_{||\mathbf{v}||=1} \quad E_{in}\Big(\mathbf{w}_t + \eta\cdot\mathbf{v}\Big)$$

对目标函数进行泰勒展开到一阶线性(注意，$\eta$要很小)：

$$E_{in}\Big(\mathbf{w}_t + \eta\cdot\mathbf{v}\Big) \approx E_{in}\Big(\mathbf{w}_t\Big) + \eta\Big(\nabla_{\mathbf{w_t}}^T\mathbf{v}\Big)$$

因此：

$$\min_{||\mathbf{v}||=1} \quad E_{in}\Big(\mathbf{w}_t\Big) + \eta\Big(\nabla_{\mathbf{w_t}}^T\mathbf{v}\Big)$$

而其中的$E_{in}(\mathbf{w}_t)$, $\eta$, $\nabla_{\mathbf{w}}$都是确定的，因此$\mathbf{v}$比与梯度呈180度反向，并考虑到其长度为1，故有：

$$\mathbf{v} = -\frac{\nabla_{\mathbf{w_t}}}{||\nabla_{\mathbf{w_t}}||}$$

综上，迭代的更新式为：

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta \cdot \frac{\nabla_{\mathbf{w_t}}}{||\nabla_{\mathbf{w_t}}||}$$

刚刚我们假设步长$\eta$是确定的，也就是说，步长是一个需要手动设置的参数。下图展示了不同的$\eta$对学习过程的影响：

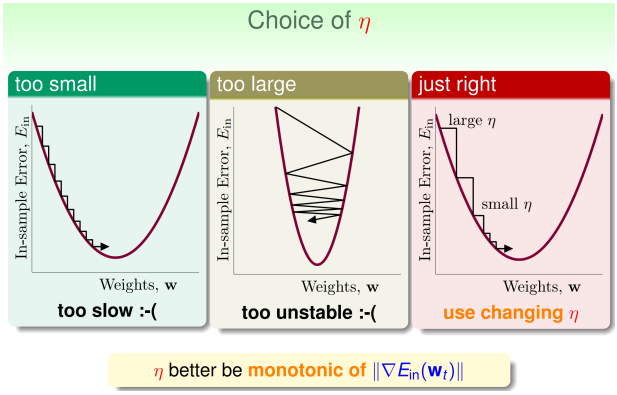


图 2: Choice of $\eta$

可见，步长最好是与梯度大小呈正比的：坡度大时步子迈得大一些(离谷底较远)，坡度小时步子迈得小一点(快到谷底了)。假设我们选这样一种步长：

$$\eta_t = \lambda||\nabla_{\mathbf{w_t}}||$$

其中$\lambda$为常数。这种步长就是上面我们说的“坡大步大，坡小步小”的步长。我们将其带入迭代式：

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \lambda||\nabla_{\mathbf{w_t}}|| \cdot \frac{\nabla_{\mathbf{w_t}}}{||\nabla_{\mathbf{w_t}}||} = \mathbf{w}_t - \lambda\nabla_{\mathbf{w_t}}$$

可见，我们其实仅需要确定$\lambda$，然后根据上式来进行迭代更新，即可达到“适应性步长”的效果。我们将上式中的$\lambda$换成$\eta$，得到“梯度下降”(gradient descent)的表达式：

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta\nabla_{\mathbf{w_t}}$$

其中的$\eta$我们称之为fixed learning rate。一个经验法则是取0.1常常会得到差不多的效果。因此，对于logistic regression问题，我们可以用梯度下降来迭代求解：

**Logistic Regression Algorithm**

initialize $\mathbf{w}_0$

For $t = 0, 1, \cdots$

1. compute

$$\nabla E_{\text{in}}(\mathbf{w}_t) = \frac{1}{N}\sum_{n=1}^{N} \theta\left(-y_n \mathbf{w}_t^T \mathbf{x}_n\right)\left(-y_n \mathbf{x}_n\right)$$

2. update by

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta \nabla E_{\text{in}}(\mathbf{w}_t)$$

...until $\nabla E_{\text{in}}(\mathbf{w}_{t+1}) = 0$ or enough iterations

return last $\mathbf{w}_{t+1}$ as $g$

图 3: Logistic Regression Algorithm