台湾大学林轩田机器学习基石课程学习笔记10 -- Logistic Regression

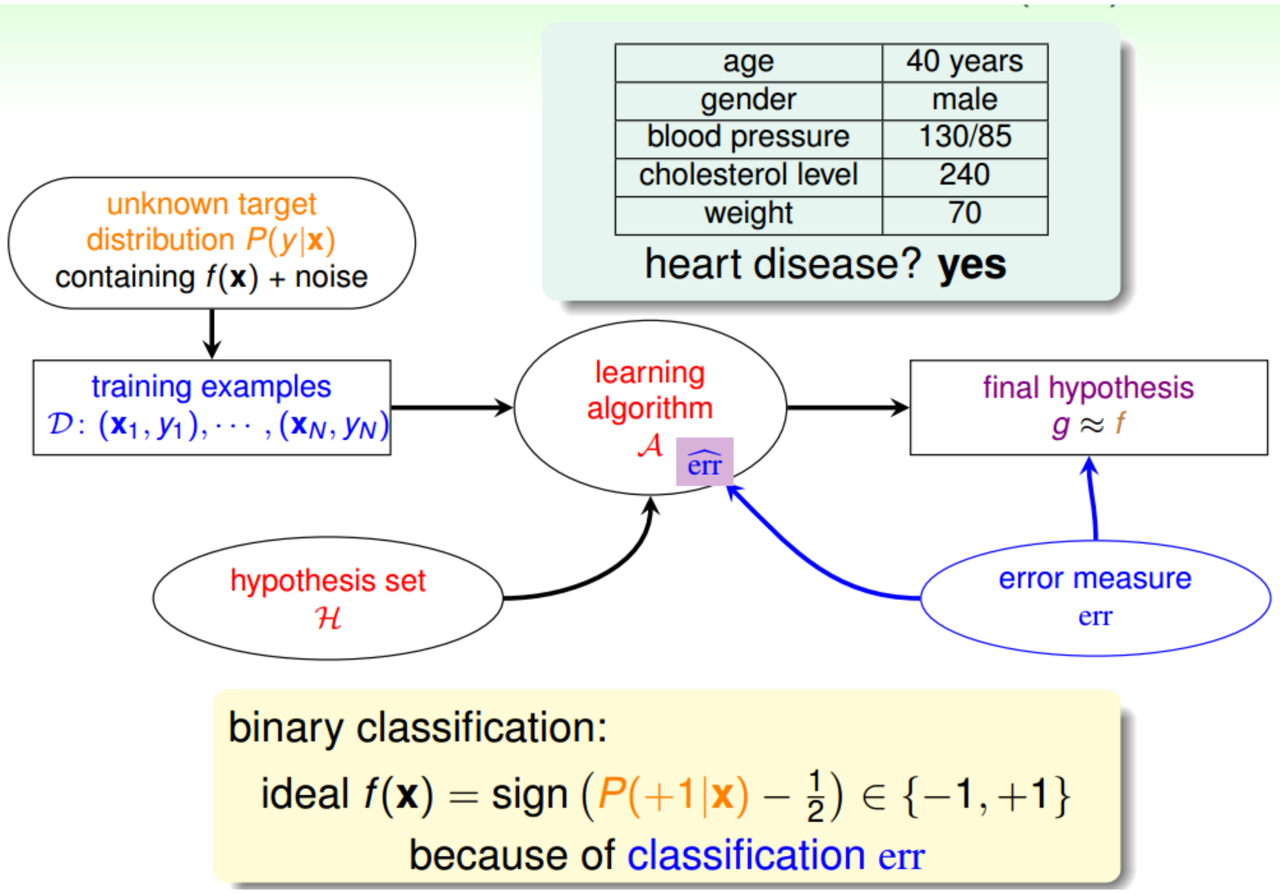
> 作者：红色石头
> 微信公众号：AI有道（ID：redstonewill）

上一节课，我们介绍了Linear Regression线性回归，以及用平方错误来寻找最佳的权重向量w，获得最好的线性预测。本节课将介绍Logistic Regression逻辑回归问题。

一、Logistic Regression Problem

一个心脏病预测的问题：根据患者的年龄、血压、体重等信息，来预测患者是否会有心脏病。很明显这是一个二分类问题，其输出y只有{-1,1}两种情况。

二元分类，一般情况下，理想的目标函数f(x)>0.5，则判断为正类1；若f(x)<0.5，则判断为负类-1。



但是，如果我们想知道的不是患者有没有心脏病，而是到底患者有多大的几率是心脏病。这表示，我们更关心的是目标函数的值（分布在0,1之间），表示是正类的概率（正类表示是心脏病）。这跟我们原来讨论的二分类问题不太一样，我们把这个问题称为软性二分类问题（'soft' binary classification）。这个值越接近1，表示正类的可能

性越大；越接近0，表示负类的可能性越大。

'soft' binary classification:

$$f(\mathbf{x}) = P(+1|\mathbf{x}) \in [0, 1]$$

对于软性二分类问题，理想的数据是分布在[0,1]之间的具体值，但是实际中的数据只可能是0或者1，我们可以把实际中的数据看成是理想数据加上了噪声的影响。

target function $f(\mathbf{x}) = P(+1|\mathbf{x}) \in [0, 1]$

ideal (noiseless) data

$$\begin{aligned}
&(\mathbf{x}_1, y'_1 &= 0.9 &= P(+1|\mathbf{x}_1)) \\
&(\mathbf{x}_2, y'_2 &= 0.2 &= P(+1|\mathbf{x}_2)) \\
&\quad\vdots \\
&(\mathbf{x}_N, y'_N &= 0.6 &= P(+1|\mathbf{x}_N))
\end{aligned}$$

actual (noisy) data

$$\begin{aligned}
&(\mathbf{x}_1, y_1 &= \circ &\sim P(y|\mathbf{x}_1)) \\
&(\mathbf{x}_2, y_2 &= \times &\sim P(y|\mathbf{x}_2)) \\
&\quad\vdots \\
&(\mathbf{x}_N, y_N &= \times &\sim P(y|\mathbf{x}_N))
\end{aligned}$$

same data as hard binary classification,
different **target function**

如果目标函数是$f(x) = P(+1|x) \in [0, 1]$的话，我们如何找到一个好的Hypothesis跟这个目标函数很接近呢？

首先，根据我们之前的做法，对所有的特征值进行加权处理。计算的结果s，我们称之为'risk score'：

- For $\mathbf{x} = (x_0, x_1, x_2, \cdots, x_d)$ 'features of patient', calculate a weighted 'risk score':

$$s = \sum_{i=0}^{d} w_i x_i$$

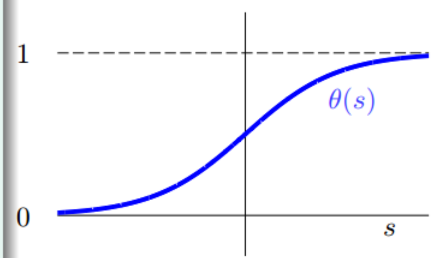
- convert the score to estimated probability by logistic function $\theta(s)$

但是特征加权和$s \in (-\infty, +\infty)$，如何将s值限定在[0,1]之间呢？一个方法是使用sigmoid Function，记为$\theta(s)$。那么我们的目标就是找到一个hypothesis：$h(x) = \theta(w^T x)$。

- For $\mathbf{x} = (x_0, x_1, x_2, \cdots, x_d)$ 'features of patient', calculate a weighted 'risk score':

$$s = \sum_{i=0}^{d} w_i x_i$$

- convert the score to estimated probability by logistic function $\theta(s)$



logistic hypothesis: $h(\mathbf{x}) = \theta(\mathbf{w}^T \mathbf{x})$

Sigmoid Function函数记为$\theta(s) = \frac{1}{1+e^{-s}}$，满足$\theta(-\infty) = 0$，$\theta(0) = \frac{1}{2}$，$\theta(+\infty) = 1$。这个函数是平滑的、单调的S型函数。则对于逻辑回归问题，hypothesis就是这样的形式：

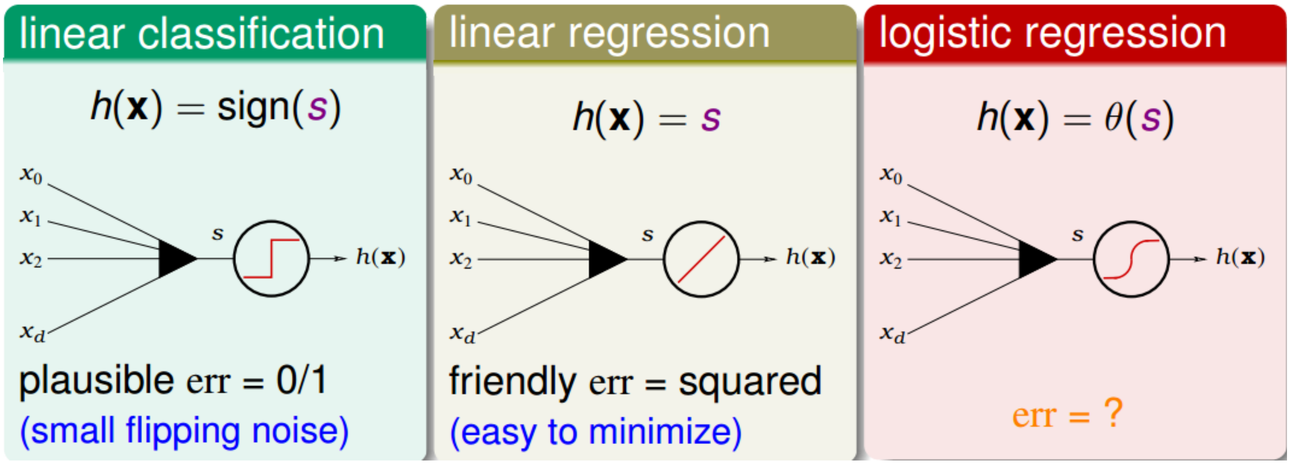
$$h(x) = \frac{1}{1 + e^{-w^T x}}$$

那我们的目标就是求出这个预测函数h(x)，使它接近目标函数f(x)。

二、Logistic Regression Error

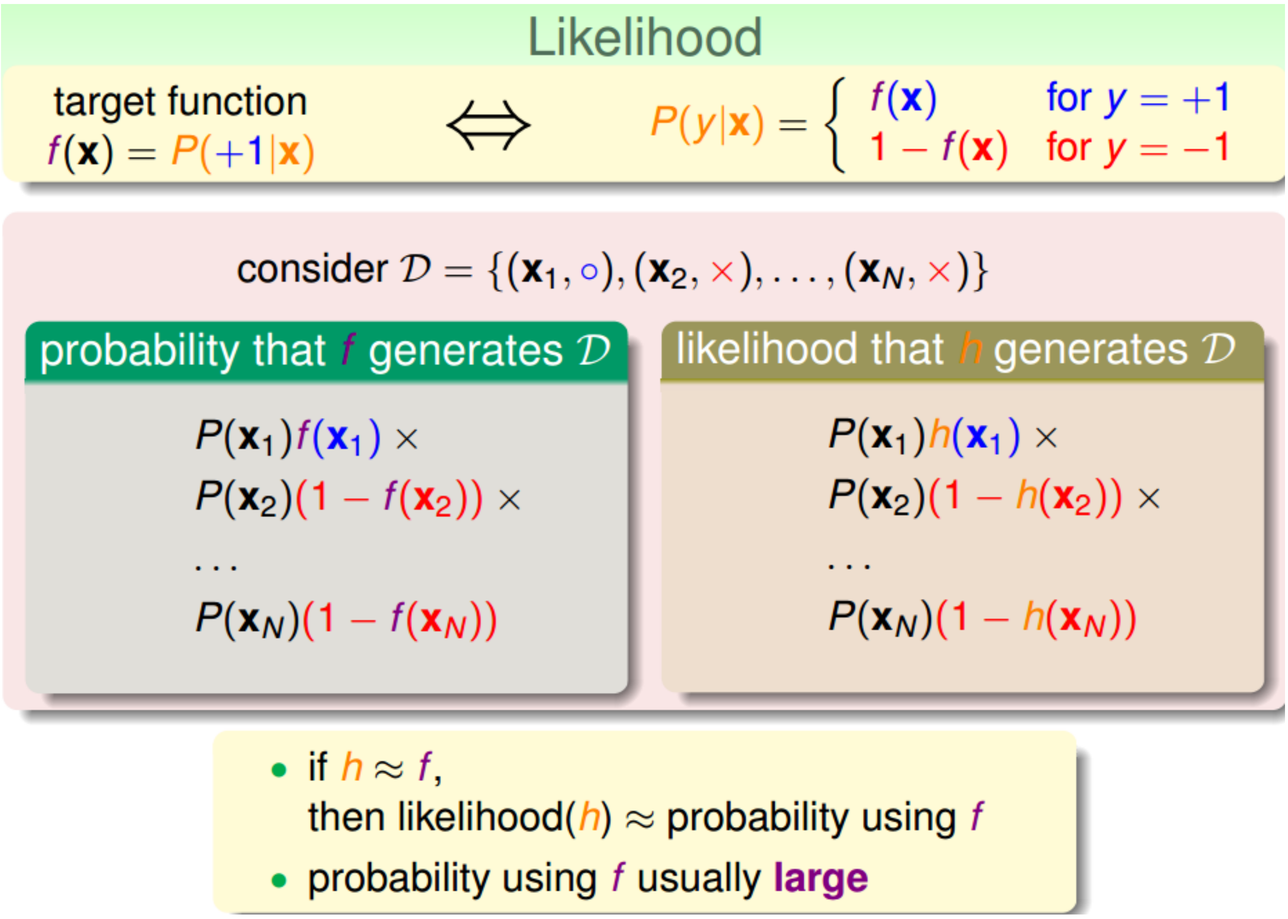
现在我们将Logistic Regression与之前讲的Linear Classification、Linear Regression

做个比较：



这三个线性模型都会用到线性scoring function $s = w^T x$。linear classification的误差使用的是0/1 err；linear regression的误差使用的是squared err。那么logistic regression的误差该如何定义呢？

先介绍一下“似然性”的概念。目标函数$f(x) = P(+1|x)$，如果我们找到了hypothesis很接近target function。也就是说，在所有的Hypothesis集合中找到一个hypothesis与target function最接近，能产生同样的数据集D，包含y输出label，则称这个hypothesis是最大似然likelihood。



logistic function: $h(x) = \theta(w^T x)$满足一个性质：$1 - h(x) = h(-x)$。那么，似然性h:

$$likelihood(h) = P(x_1)h(+x_1) \times P(x_2)h(-x_2) \times \cdots P(x_N)h(-x_N)$$

因为$P(x_n)$对所有的h来说，都是一样的，所以我们可以忽略它。那么我们可以得到logistic h正比于所有的$h(y_n x)$乘积。我们的目标就是让乘积值最大化。

$$\max_{h} \quad \text{likelihood}(\text{logistic } h) \propto \prod_{n=1}^{N} h(y_n \mathbf{x}_n)$$

如果将w代入的话：

$$\max_{\mathbf{w}} \quad \text{likelihood}(\mathbf{w}) \propto \prod_{n=1}^{N} \theta\left(y_n \mathbf{w}^T \mathbf{x}_n\right)$$

为了把连乘问题简化计算，我们可以引入ln操作，让连乘转化为连加：

$$\max_{\mathbf{w}} \quad \ln \prod_{n=1}^{N} \theta\left(y_n \mathbf{w}^T \mathbf{x}_n\right)$$

接着，我们将maximize问题转化为minimize问题，添加一个负号就行，并引入平均数操作$\frac{1}{N}$：

$$\min_{\mathbf{w}} \quad \frac{1}{N} \sum_{n=1}^{N} - \ln \theta\left(y_n \mathbf{w}^T \mathbf{x}_n\right)$$

将logistic function的表达式带入，那么minimize问题就会转化为如下形式：

$$\theta(s) = \frac{1}{1+\exp(-s)} \quad : \quad \min_{\mathbf{w}} \quad \frac{1}{N} \sum_{n=1}^{N} \ln\left(1+\exp(-y_n \mathbf{w}^T \mathbf{x}_n)\right)$$

$$\Longrightarrow \quad \min_{\mathbf{w}} \quad \underbrace{\frac{1}{N} \sum_{n=1}^{N} \mathrm{err}(\mathbf{w}, \mathbf{x}_n, y_n)}_{E_{\text{in}}(\mathbf{w})}$$

至此，我们得到了logistic regression的err function，称之为cross-entropy error交叉熵误差：

$$\mathrm{err}(\mathbf{w}, \mathbf{x}, y) = \ln\left(1 + \exp(-y\mathbf{w}\mathbf{x})\right):$$

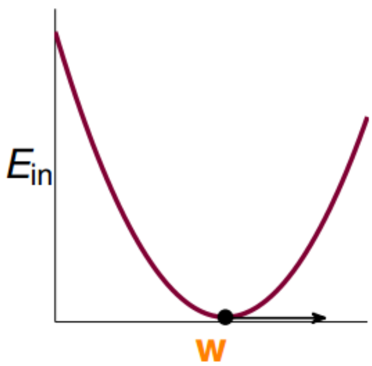
cross-entropy error

三、Gradient of Logistic Regression Error

我们已经推导了E_{in}的表达式，那接下来的问题就是如何找到合适的向量w，让E_{in}最小。

$$\min_{\mathbf{w}} \quad E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \ln\left(1 + \exp(-y_n\mathbf{w}^T\mathbf{x}_n)\right)$$

Logistic Regression的E_{in}是连续、可微、二次可微的凸曲线（开口向上），根据之前Linear Regression的思路，我们只要计算E_{in}的梯度为零时的w，即为最优解。



- $E_{\text{in}}(\mathbf{w})$: continuous, differentiable, twice-differentiable, **convex**
- how to minimize? locate **valley**

$$\text{want } \nabla E_{\text{in}}(\mathbf{w}) = \mathbf{0}$$

对E_{in}计算梯度，学过微积分的都应该很容易计算出来：

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \ln\Bigg(\underbrace{1+\exp(\overbrace{-y_n\mathbf{w}^T\mathbf{x}_n}^{\bigcirc})}_{\square}\Bigg)$$

$$\begin{aligned}
\frac{\partial E_{\text{in}}(\mathbf{w})}{\partial w_i} &= \frac{1}{N}\sum_{n=1}^{N}\left(\frac{\partial \ln(\square)}{\partial \square}\right)\left(\frac{\partial(1+\exp(\bigcirc))}{\partial \bigcirc}\right)\left(\frac{\partial - y_n\mathbf{w}^T\mathbf{x}_n}{\partial w_i}\right) \\
&= \frac{1}{N}\sum_{n=1}^{N}\left(\frac{1}{\square}\right)\Big(\exp(\bigcirc)\Big)\Big(-y_n x_{n,i}\Big) \\
&= \frac{1}{N}\sum_{n=1}^{N}\left(\frac{\exp(\bigcirc)}{1+\exp(\bigcirc)}\right)\Big(-y_n x_{n,i}\Big) = \frac{1}{N}\sum_{n=1}^{N}\theta(\bigcirc)\big(-y_n x_{n,i}\big)
\end{aligned}$$

最终得到的梯度表达式为：

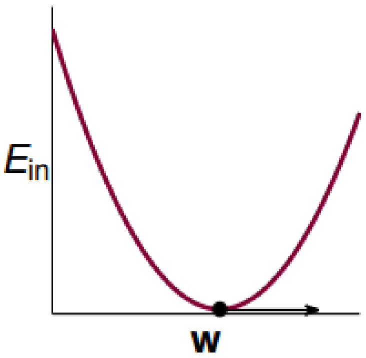
$$\nabla E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\theta\left(-y_n\mathbf{w}^T\mathbf{x}_n\right)\left(-y_n\mathbf{x}_n\right)$$

为了计算E_{in}最小值，我们就要找到让$\nabla E_{in}(w)$等于0的位置。

$$\begin{aligned}
\min_{\mathbf{w}} \quad E_{\text{in}}(\mathbf{w}) &= \frac{1}{N}\sum_{n=1}^{N}\ln\left(1+\exp(-y_n\mathbf{w}^T\mathbf{x}_n)\right) \\
\text{want } \nabla E_{\text{in}}(\mathbf{w}) &= \boxed{\frac{1}{N}\sum_{n=1}^{N}\theta\left(-y_n\mathbf{w}^T\mathbf{x}_n\right)\left(-y_n\mathbf{x}_n\right)} = \mathbf{0}
\end{aligned}$$

上式可以看成$\theta(-y_nw^Tx_n)$是$-y_nx_n$的线性加权。要求$\theta(-y_nw^Tx_n)$与$-y_nx_n$的线性加权和为0，那么一种情况是线性可分，如果所有的权重$\theta(-y_nw^Tx_n)$为0，那就能保证$\nabla E_{in}(w)$为0。$\theta(-y_nw^Tx_n)$是sigmoid function，根据其特性，只要让$-y_nw^Tx_n \ll 0$，即$y_nw^Tx_n \gg 0$。$y_nw^Tx_n \gg 0$表示对于所有的点，y_n与w^Tx_n都是同号的，这表示数据集D必须是全部线性可分的才能成立。

然而，保证所有的权重$\theta(-y_nw^Tx_n)$为0是不太现实的，总有不等于0的时候，那么另一种常见的情况是非线性可分，只能通过使加权和为零，来求解w。这种情况没有closed-form解，与Linear Regression不同，只能用迭代方法求解。



scaled θ-weighted sum of $-y_n\mathbf{x}_n$

- all $\theta(\cdot) = 0$: only if $y_n\mathbf{w}^T\mathbf{x}_n \gg 0$
  —linear separable $\mathcal{D}$
- weighted sum = **0**:
  non-linear equation of **w**

closed-form solution? no :-(

之前所说的Linear Regression有closed-form解，可以说是“一步登天”的；但是PLA算法是一步一步修正迭代进行的，每次对错误点进行修正，不断更新w值。PLA的迭代优化过程表示如下：

PLA: start from some $\mathbf{w}_0$ (say, **0**), and ‘correct’ its mistakes on $\mathcal{D}$

For $t = 0, 1, \ldots$

1. find a mistake of $\mathbf{w}_t$ called $\left(\mathbf{x}_{n(t)}, y_{n(t)}\right)$

$$\text{sign}\left(\mathbf{w}_t^T\mathbf{x}_{n(t)}\right) \neq y_{n(t)}$$

2. (try to) correct the mistake by

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_{n(t)}\mathbf{x}_{n(t)}$$

1. (equivalently) pick some n, and update $\mathbf{w}_t$ by

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \left[\!\left[\text{sign}\left(\mathbf{w}_t^T\mathbf{x}_n\right) \neq y_n\right]\!\right] y_n\mathbf{x}_n$$

when stop, return last **w** as g

w每次更新包含两个内容：一个是每次更新的方向y_nx_n，用v表示，另一个是每次更新的步长η。参数(v,η)和终止条件决定了我们的迭代优化算法。

For $t = 0, 1, \ldots$

1. (equivalently) pick some n, and update $\mathbf{w}_t$ by

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \underbrace{1}_{\eta} \cdot \underbrace{\left([\![\text{sign}\left(\mathbf{w}_t^T \mathbf{x}_n\right) \neq y_n]\!] \cdot y_n \mathbf{x}_n \right)}_{\mathbf{v}}$$

when stop, return last $\mathbf{w}$ as g

choice of $(\eta, \mathbf{v})$ and stopping condition defines
iterative optimization approach

四、Gradient Descent

根据上一小节PLA的思想，迭代优化让每次w都有更新：

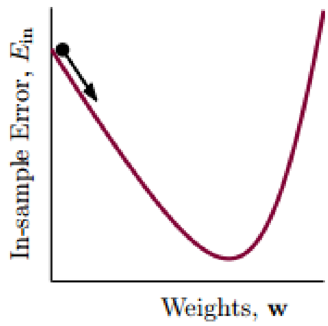
For $t = 0, 1, \ldots$

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

when stop, return last $\mathbf{w}$ as g

我们把$E_{in}(w)$曲线看做是一个山谷的话，要求$E_{in}(w)$最小，即可比作下山的过程。整个下山过程由两个因素影响：一个是下山的单位方向v；另外一个是下山的步长η。

- PLA: $\mathbf{v}$ comes from mistake correction
- smooth $E_{\text{in}}(\mathbf{w})$ for logistic regression: choose $\mathbf{v}$ to get the ball roll '**downhill**'?
  - direction $\mathbf{v}$: (assumed) of unit length
  - step size η: (assumed) positive



a greedy approach for some given $\eta > 0$:

$$\min_{\|\mathbf{v}\|=1} E_{\text{in}}(\underbrace{\mathbf{w}_t + \eta \mathbf{v}}_{\mathbf{w}_{t+1}})$$

利用微分思想和线性近似，假设每次下山我们只前进一小步，即η很小，那么根据泰勒Taylor一阶展开，可以得到：

$$E_{in}(w_t + \eta v) \approx E_{in}(w_t) + \eta v^T \nabla E_{in}(w_t)$$

关于Taylor展开的介绍，可参考我另一篇博客：
多元函数的泰勒(Taylor)展开式

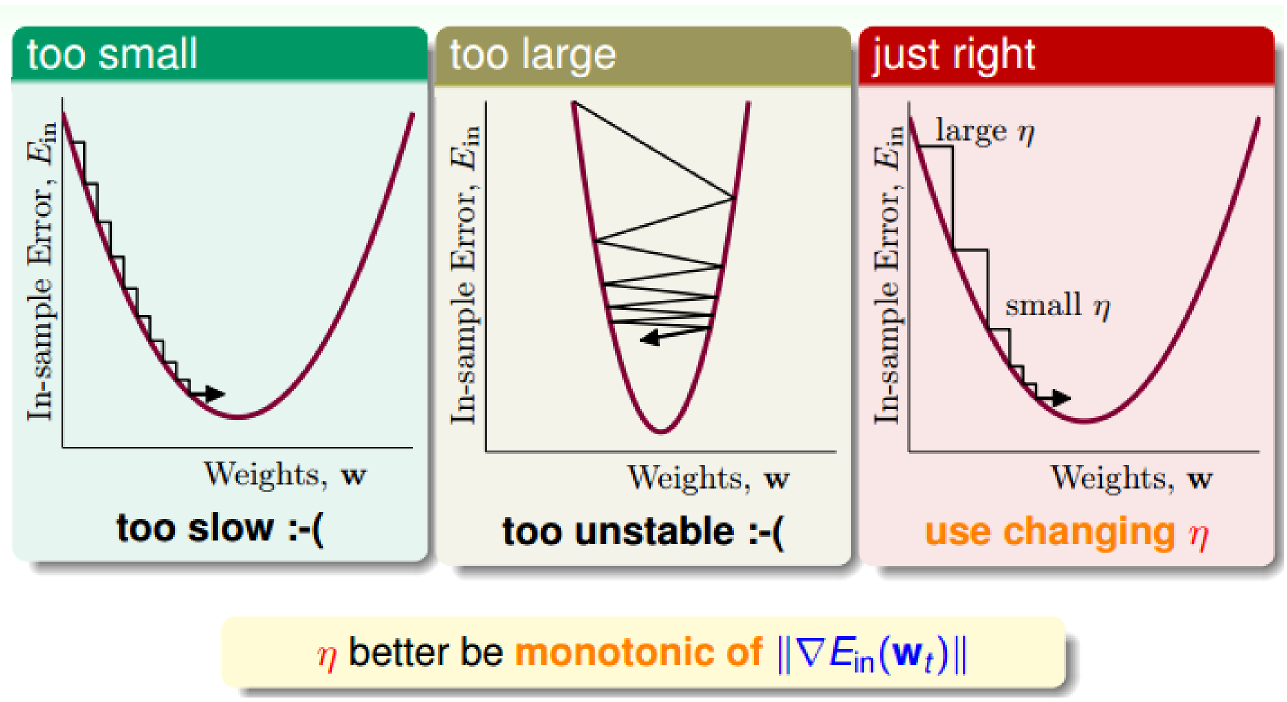
迭代的目的是让E_{in}越来越小，即让$E_{in}(w_t+\eta v)<E_{in}(w_t)$。$\eta$是标量，因为如果两个向量方向相反的话，那么他们的内积最小（为负），也就是说如果方向v与梯度$\nabla E_{in}(w_t)$反向的话，那么就能保证每次迭代$E_{in}(w_t+\eta v)<E_{in}(w_t)$都成立。则，我们令下降方向$v$为：

$$v=-\frac{\nabla E_{in}(w_t)}{||\nabla E_{in}(w_t)||}$$

v是单位向量，v每次都是沿着梯度的反方向走，这种方法称为梯度下降（gradient descent）算法。那么每次迭代公式就可以写成：

$$w_{t+1}\leftarrow w_t-\eta\frac{\nabla E_{in}(w_t)}{||\nabla E_{in}(w_t)||}$$

下面讨论一下η的大小对迭代优化的影响：η如果太小的话，那么下降的速度就会很慢；η如果太大的话，那么之前利用Taylor展开的方法就不准了，造成下降很不稳定，甚至会上升。因此，η应该选择合适的值，一种方法是在梯度较小的时候，选择小的η，梯度较大的时候，选择大的η，即η正比于$||\nabla E_{in}(w_t)||$。这样保证了能够快速、稳定地得到最小值$E_{in}(w)$。



对学习速率η做个更修正，梯度下降算法的迭代公式可以写成：

$$w_{t+1}\leftarrow w_t-\eta'\nabla E_{in}(w_t)$$

其中：

$$\eta'=\frac{\eta}{||\nabla E_{in}(w_t)||}$$

总结一下基于梯度下降的Logistic Regression算法步骤如下：

- **初始化**w_0
- **计算梯度**$\nabla E_{in}(w_t)=\frac{1}{N}\sum_{n=1}^{N}\theta(-y_nw_t^Tx_n)(-y_nx_n)$
- **迭代跟新**$w_{t+1}\leftarrow w_t-\eta\nabla E_{in}(w_t)$
- **满足**$\nabla E_{in}(w_{t+1})\approx 0$**或者达到迭代次数，迭代结束**

五、总结

我们今天介绍了Logistic Regression。首先，从逻辑回归的问题出发，将$P(+1|x)$作为目标函数，将$\theta(w^Tx)$作为hypothesis。接着，我们定义了logistic regression的err function，称之为cross-entropy error交叉熵误差。然后，我们计算logistic regression error的梯度，最后，通过梯度下降算法，计算$\nabla E_{in}(w_t)\approx 0$时对应的$w_t$值。

注明：

文章中所有的图片均来自台湾大学林轩田《机器学习基石》课程