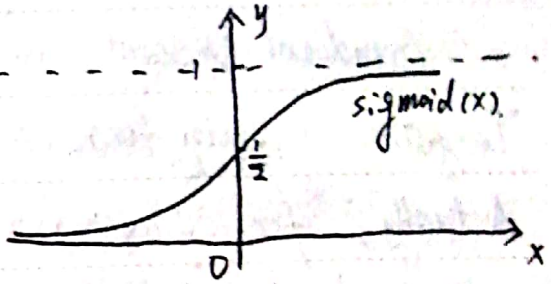
§1.3. 逻辑回归——不是回归的回归.

模型背景：$z=f(y)$，$y=w^Tx+b$. $z\in\{0,1\}$

从 y 的实值到 z 的 0-1 值，套用函数.



① f: unit-step function: $f=\begin{cases}0 & z<0\\ 0.5 & z=0\\ 1 & z>0\end{cases}$. 不可导原点

② f: sigmoid function: $f=\frac{1}{1+e^{-x}}$. 连续可导，原点导值大变化快

$\Rightarrow$ 模型 $z=\frac{1}{1+e^{-(w^Tx+b)}}$. $\Leftrightarrow$ $\ln\frac{z}{1-z}=w^Tx+b$.

2. 模型意义：$z=0$：样本标记为负类 negative $z=1$：样本标记为正类 positive.

所以 logstic regression 不是解决回归问题，而是分类哒！（DIO厨狂喜）

3. 模型求解的描述：

假使说 z 数值代表了一个后验概率 $p(z=1|x)$. 那么 $\ln\frac{p(y=1|x)}{p(y=0|x)}=w^Tx+b$

$$p(y=1|x)=\frac{e^{w^Tx+b}}{1+e^{w^Tx+b}}\qquad p(y=0|x)=\frac{1}{1+e^{w^Tx+b}}$$

那么我们试着用极大似然法求解：（极大似然函数是把 y 的概率全乘再取对数）

$L(w,b)=\sum_{i=1}^{m}\ln p(y_i|x_i)$. 与线性回归一样，记 $\beta=(w;b)^T$ $X=\begin{pmatrix}x_1^T & 1\\ \vdots & \vdots\\ x_m^T & 1\end{pmatrix}$

$p(y_i|x_i)=y_i p_1(x_i;\beta)+(1-y_i)p_0(x_i;\beta)$.

$\Rightarrow$

$$L(w,b)=\sum_{i=1}^{m}\ln[y_i p_1(x_i;\beta)+(1-y_i)p_0(x_i;\beta)].$$

解似然函数最大值

4. 求解模型：

求解方法有牛顿法和梯度下降法.

解 $L(\beta)$ 的最大值本质上也就是 $\frac{\partial L}{\partial\beta}=0$. 方程. 牛顿法可用于方程根的数值解

① 牛顿法：$\beta^{t+1}=\beta^t-\left(\frac{\partial^2L(\beta)}{\partial\beta\,\partial\beta^T}\right)^{-1}\frac{\partial L(\beta)}{\partial\beta}$.

其中：$\frac{\partial L(\beta)}{\partial\beta}=-\sum_{i=1}^{m}\hat{x}_i(y_i-p_1(\hat{x}_i;\beta))$ $\frac{\partial^2L(\beta)}{\partial\beta\,\partial\beta^T}=\sum_{i=1}^{m}\hat{x}_i\hat{x}_i^T p_1(\hat{x}_i;\beta)(1-p_1(\hat{x}_i;\beta))$

② 梯度下降法：$\beta_{i+1}=\beta_i-\eta\frac{\partial L}{\partial\beta}$. η 称为学习率.

补充：梯度下降

Gradient Descent is a common way to solve Optimization problem.

Target: $\min_{x \in R^n} f(x)$. x^* is the best solution.

Actually, $f(x) = f(x^{(k)}) + grad_k^T (x - x^{(k)})$ ①.

$grad_k = \nabla f(x^{(k)})$, is called the gradient of $f(x)$.

⇒ if determind a vector $P_k = \{P_1, P_2, \cdots, P_n\}$ and astep λ.

① can be rewrited as: $X^{(k+1)} = X^{(k)} + \lambda P_k$.

Eh, we often say the step λ as learning rate, and use η instead

and P_k, yes, as its name indicates, is associated with $grad(x^{(k)})$

actually, $P_k = -\nabla f(X^{(k)})$. $(-grad(x^{(k)}))$

So we have the algorithm below:

(1). initialize $X^{(0)} \in R^n$, $k=0$.

(2). compute $f(x^{(k)})$ & $grad(X^{(k)})$.

(3). if $\|grad(X^{(k)})\| \leq \varepsilon$, then break;

else: $P_k = -grad(x^{(k)})$, s.t. $f(x^{(k)} + \eta p^{(k)}) = \min_{\eta \geq 0}$

(4). calculate $X^{(k+1)} = X^{(k)} + \eta P_k$. then $f(x^{(k+1)})$.

(5). if $\|X^{(k+1)} - X^{(k)}\| < \varepsilon$ or $\|f(x^{(k+1)}) - f(x^{(k)})\| < \varepsilon$, break, and $X^* = X$

else, $i+1$, go to step (3) and recurrent.

Here, λ: learning rate; ε: error