逻辑斯蒂回归

赵海涛

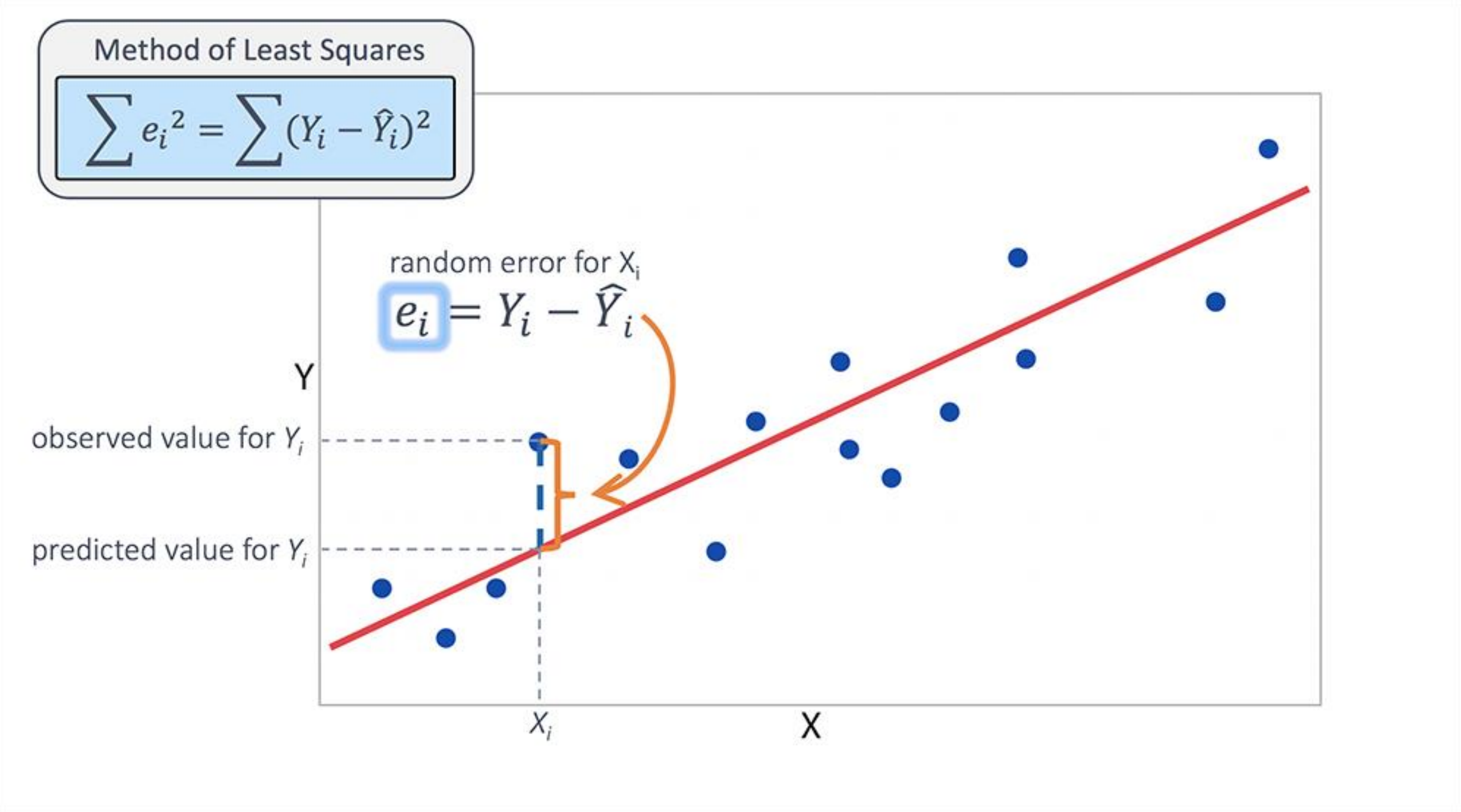
haitaozhao@ecust.edu.cn

大纲

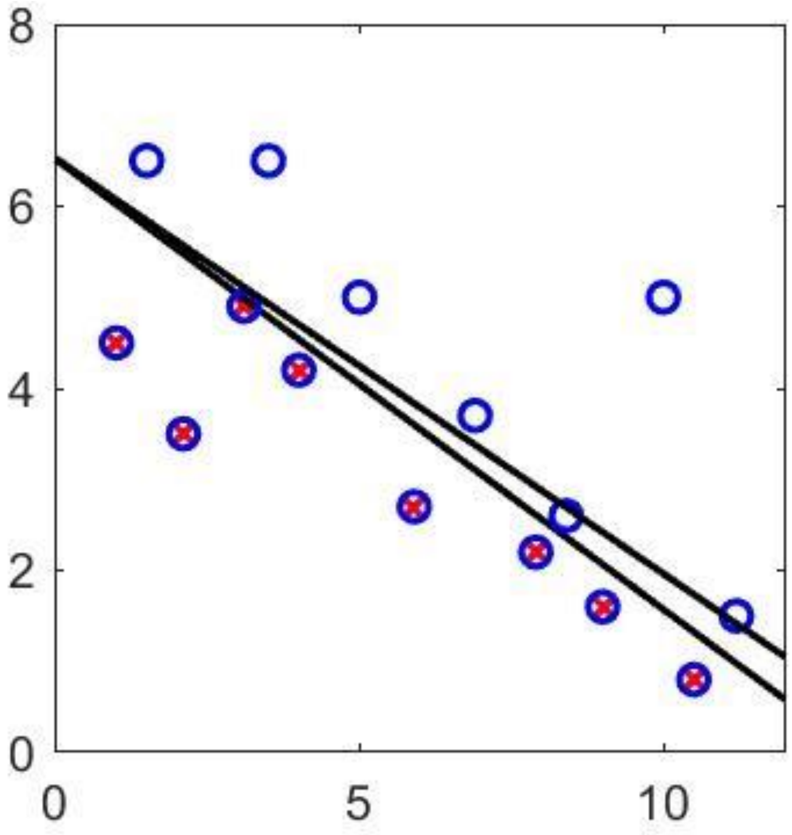
- Logistic回归的模型
- Logistic回归的策略
- Logistic回归的算法
- 优化算法

线性回归 （Linear Regression）

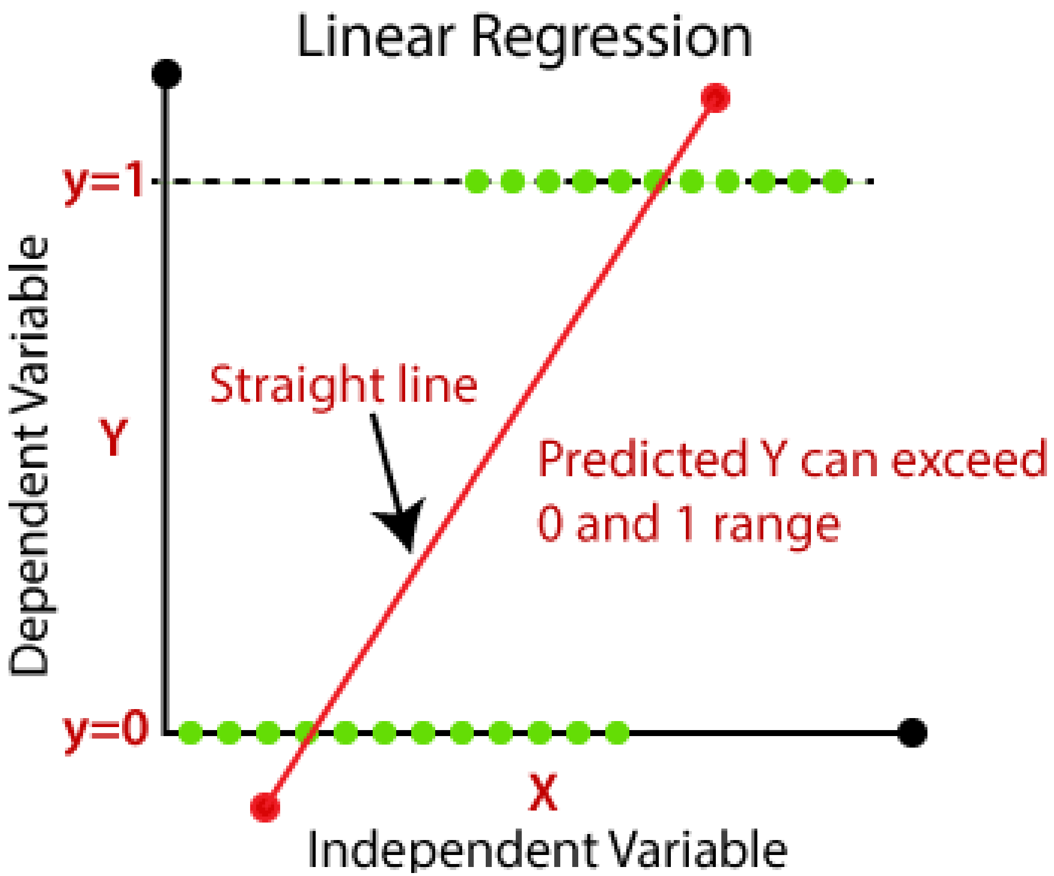
面积 (m^2)	销售价钱（万元）
123	250
150	320
87	160
102	220
…	…

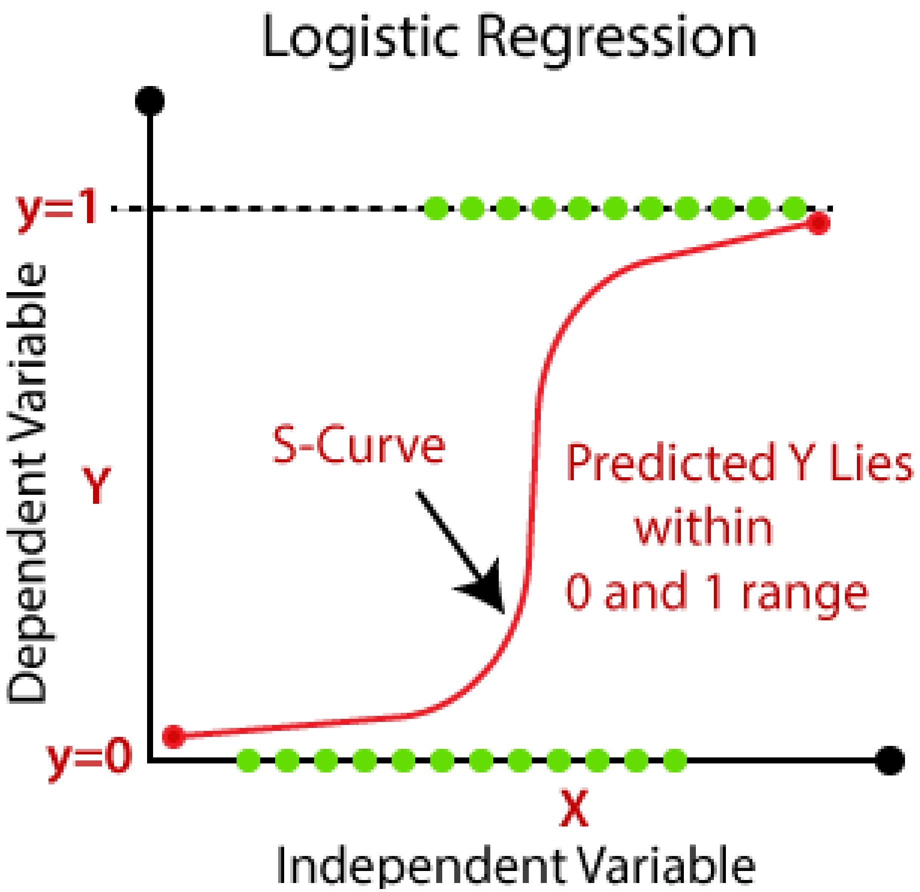


线性回归 （Linear Regression）



线性回归与逻辑斯蒂回归

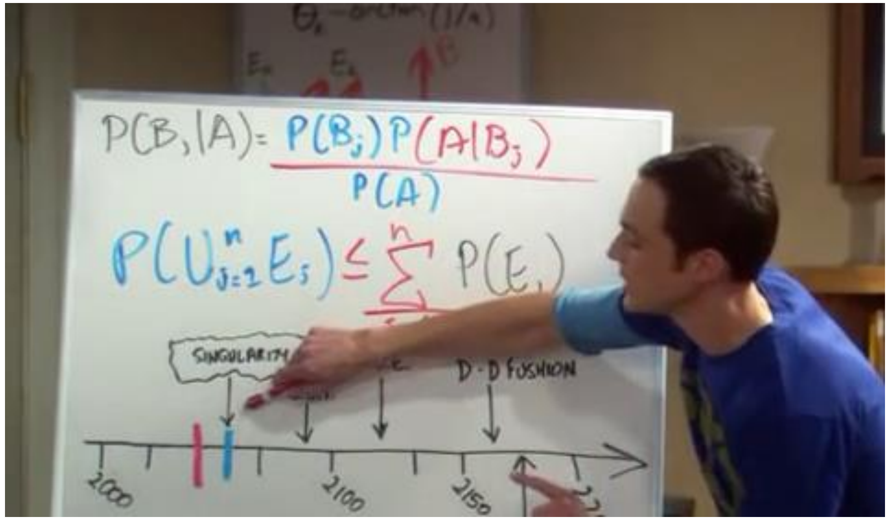




Logistic 回归的模型

针对二分类问题（$y_i \in \{-1,1\}$），当判别函数$g_i(x) > 0.5$, 即可认为$y_i = 1$，$g_i(x) < 0.5$, 即可认为$y_i = -1$。取后验概率作为判别函数$g_i(x)$:

$$P(\omega_i|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\omega_i)P(\omega_i)}{\sum_{j=1}^{c} p\left(\boldsymbol{x}|\omega_j\right)P\left(\omega_j\right)}$$



Bayes' rule

Logistic 回归的模型

针对二分类问题，只有ω_1和ω_2两类，$P(\omega_1|\boldsymbol{x})+P(\omega_2|\boldsymbol{x})=1$，假设两类样本个数相同，即$P(\omega_1)=P(\omega_2)$，可得

$$P(\omega_i|\boldsymbol{x})=\frac{p(\boldsymbol{x}|\omega_i)P(\omega_i)}{p(\boldsymbol{x}|\omega_1)P(\omega_1)+p(\boldsymbol{x}|\omega_2)P(\omega_2)} \quad (i=1,2)$$

$$=\frac{p(\boldsymbol{x}|\omega_i)}{p(\boldsymbol{x}|\omega_1)+p(\boldsymbol{x}|\omega_2)} \quad (i=1,2)$$

Logistic 回归的模型

取d维多元正态密度函数作为概率密度函数：

$$p(\boldsymbol{x}|\omega_i) = \mathcal{N}(\boldsymbol{x};\ \mu_i, \sigma)$$

$$p(\boldsymbol{x}|\omega_i) = \frac{1}{\left(\sqrt{2\pi\sigma}\right)^d} \exp\left(-\frac{(\boldsymbol{x}-\boldsymbol{\mu}_i)^T(\boldsymbol{x}-\boldsymbol{\mu}_i)}{2\sigma^2}\right)$$

$$P(\omega_1|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\omega_1)}{p(\boldsymbol{x}|\omega_1)+p(\boldsymbol{x}|\omega_2)}$$ （将$p(\boldsymbol{x}|\omega_i)$代入）

$$= \frac{\frac{1}{\left(\sqrt{2\pi\sigma}\right)^d}\exp\left(-\frac{(x-\boldsymbol{\mu}_1)^T(x-\boldsymbol{\mu}_1)}{2\sigma^2}\right)}{\frac{1}{\left(\sqrt{2\pi\sigma}\right)^d}\exp\left(-\frac{(x-\boldsymbol{\mu}_1)^T(x-\boldsymbol{\mu}_1)}{2\sigma^2}\right)+\frac{1}{\left(\sqrt{2\pi\sigma}\right)^d}\exp\left(-\frac{(x-\boldsymbol{\mu}_2)^T(x-\boldsymbol{\mu}_2)}{2\sigma^2}\right)}$$

Logistic 回归的模型

$$= \frac{\exp\left(-\frac{x^T x - 2\mu_1{}^T x + \mu_1{}^T \mu_1}{2\sigma^2}\right)}{\exp\left(-\frac{x^T x - 2\mu_1{}^T x + \mu_1{}^T \mu_1}{2\sigma^2}\right) + \exp\left(-\frac{x^T x - 2\mu_2{}^T x + \mu_2{}^T \mu_2}{2\sigma^2}\right)}$$

$$= \frac{1}{1 + \exp\left[\left(-\frac{x^T x - 2\mu_2{}^T x + \mu_2{}^T \mu_2}{2\sigma^2}\right) - \left(-\frac{x^T x - 2\mu_1{}^T x + \mu_1{}^T \mu_1}{2\sigma^2}\right)\right]}$$

$$= \frac{1}{1 + \exp\left(-\frac{(\mu_1{}^T - \mu_2{}^T)x}{\sigma^2} + \frac{\mu_1{}^T \mu_1 - \mu_2{}^T \mu_2}{2\sigma^2}\right)}$$

$$= \frac{1}{1 + \exp(-(w^T x + b))}$$

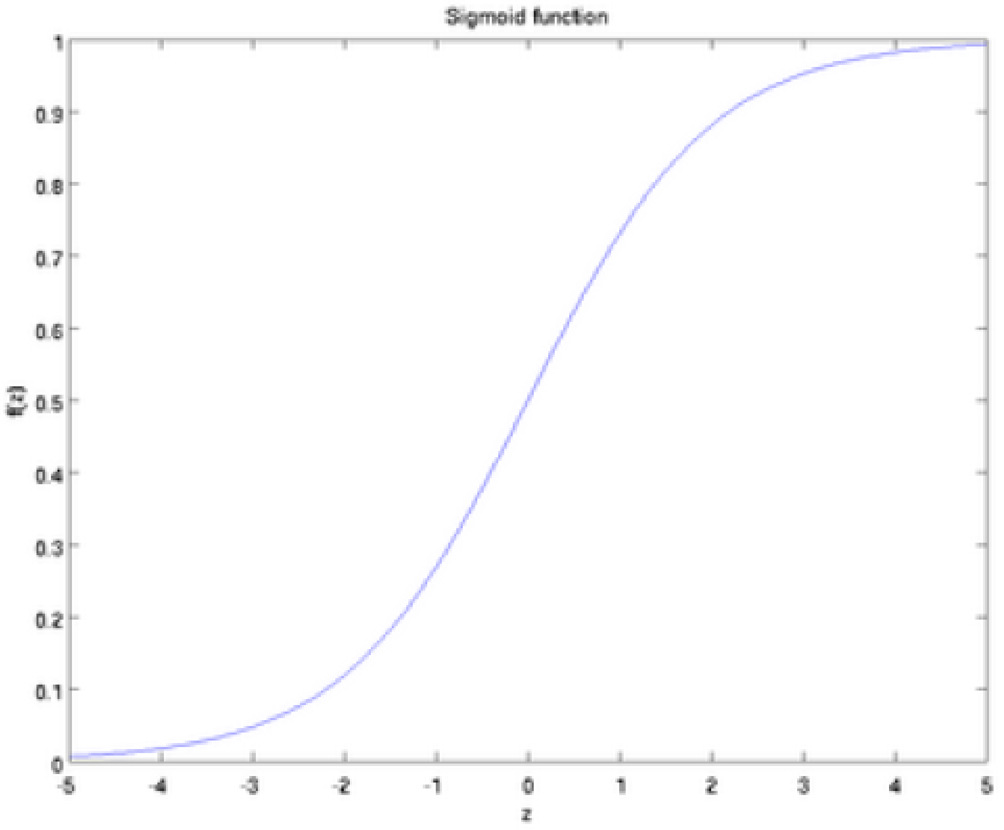
$$= \frac{1}{1 + \exp(-z)} = \sigma(z)$$

Logistic 回归的模型

Sigmoid:

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

$$\sigma'(z) = \sigma(z)\left(1 - \sigma(z)\right)$$



Logistic 回归的模型

$$P(\omega_1|\boldsymbol{x}) = P(y=1|\boldsymbol{x}) = \frac{1}{1+\exp(-\boldsymbol{W}^T\boldsymbol{x})}$$

$$P(\omega_2|\boldsymbol{x}) = P(y=-1|\boldsymbol{x}) = 1 - P(\omega_1|\boldsymbol{x})$$

$$= 1 - \frac{1}{1+\exp(-\boldsymbol{W}^T\boldsymbol{x})}$$

$$= \frac{\exp(-\boldsymbol{W}^T\boldsymbol{x})}{1+\exp(-\boldsymbol{W}^T\boldsymbol{x})}$$

$$= \frac{1}{1+\exp(\boldsymbol{W}^T\boldsymbol{x})}$$

其中$\boldsymbol{x} = \begin{bmatrix}\boldsymbol{x}\\1\end{bmatrix}$，$\boldsymbol{W} = \begin{bmatrix}\boldsymbol{w}\\b\end{bmatrix}$。根据结果可得$P(\omega_i|\boldsymbol{x}) = \frac{1}{1+\exp(-y_i\boldsymbol{W}^T\boldsymbol{x})}$

Logistic 回归的策略

- logistic分类器是由一组权值系数组成的，最关键的问题就是如何获取这组权值，通过极大似然函数估计获得
- 似然函数是统计模型中参数的函数。给定输出$\boldsymbol{x}$时，关于参数$\boldsymbol{\theta}$的似然函数$L(\boldsymbol{\theta}|\boldsymbol{x})$（在数值上）等于给定参数$\boldsymbol{\theta}$后变量$X$的概率：

$$L(\boldsymbol{\theta}|\boldsymbol{x}) = P(X = \boldsymbol{x}; \boldsymbol{\theta})$$

- 似然函数的重要性不是它的取值，而是当参数变化时概率密度函数到底是变大还是变小。
- 最大似然估计：似然函数取得最大值表示相应的参数能够使得统计模型最为合理

最大似然估计

- N noisy measurements are made to observe the constant μ:

$$z_i = \mu + v_i \quad i = 1, 2, \cdots, N \quad with \quad i.i.d.\ v_i \sim N(0, \sigma^2)$$

- For $\theta = (\mu, \sigma^2)$,
$p(z_1, z_2, \cdots, z_N|\theta) = \frac{1}{(2\pi)^{N/2}\sigma^N} \exp(-\frac{1}{2\sigma^2}\sum_{i=1}^{N}(z_i - \mu)^2)$
- MLE:

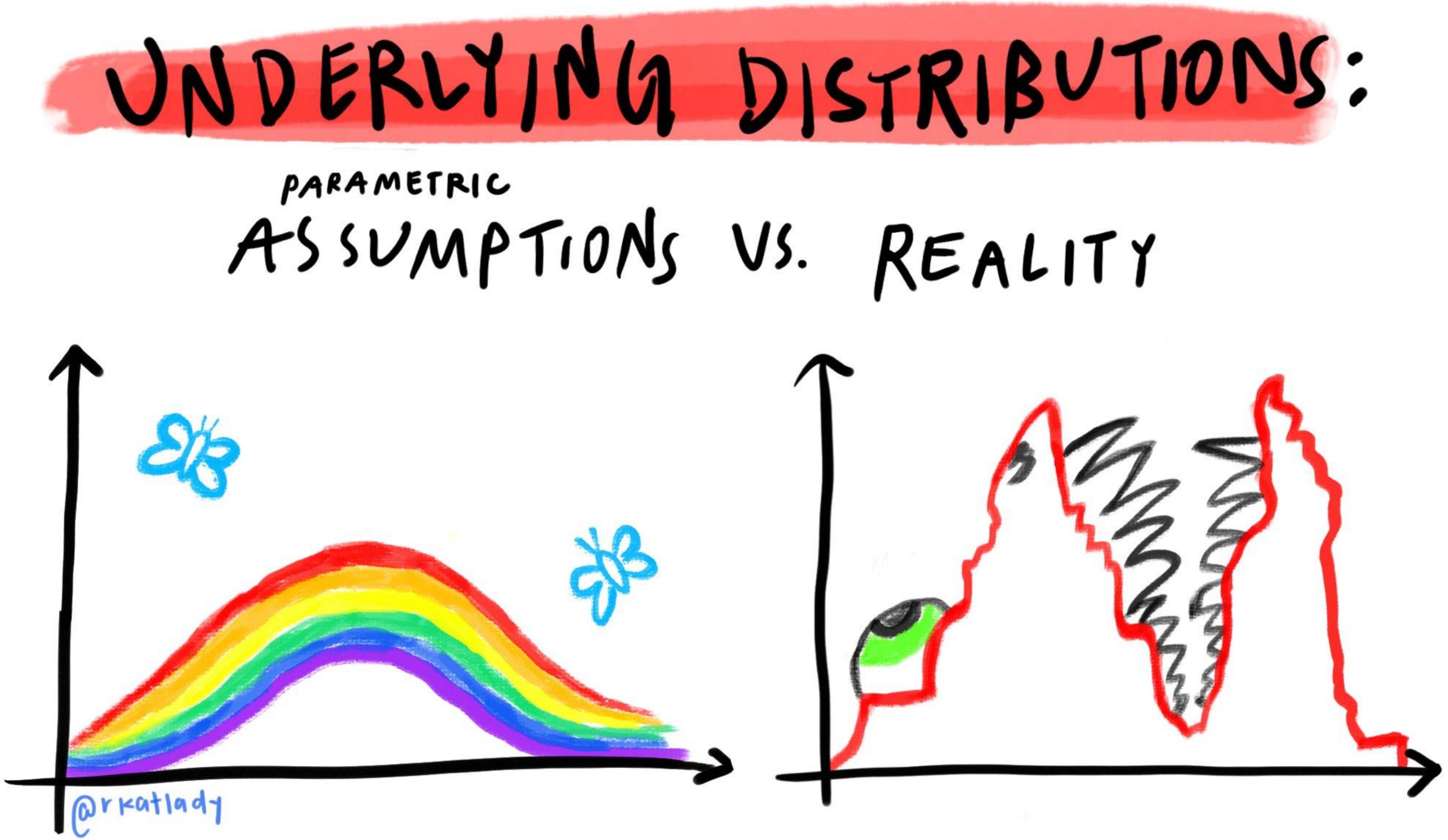
$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} z_i = sample\ mean$$

$$\hat{\sigma^2} = \frac{1}{N}\sum_{i=1}^{N}(z_i - \hat{\mu})^2 = sample\ variance$$

最大似然估计

- Result of tossing a coin is $\in$ {Heads,Tails}
- Random var $X \in \{1,0\}$
  - Bernoulli: $P(X = x) = p_0^x (1 - p_0)^{(1-x)}$
- Sample: $X = \{x^t\}_{t=1}^N$
- $\ln p(D|\theta) = \ln \prod_{t=1}^N p_0^{x^t} (1 - p_0)^{(1-x^t)}$
- MLE $\hat{p}_0 = \frac{\sum_{i=1}^N x^i}{N}$

最大似然估计


UNDERLYING DISTRIBUTIONS:
PARAMETRIC
ASSUMPTIONS VS. REALITY
@rkatlady

Logistic 回归的目标函数

- 由于$\boldsymbol{x}_i$是独立同分布（i.i.d.），似然函数为：

$$L(W)=\prod_{i=1}^{N}p(\omega_i|\boldsymbol{x}_i)=\prod_{i=1}^{N}\frac{1}{1+\exp(-y_iW^T\boldsymbol{x}_i)}$$

- 一般使用似然函数的负对数函数（Negative Log Likelihood，NLL）来作为Logistic回归的损失函数：

$$NLL(W)=-\ln L(W)=-\sum_{i=1}^{N}\ln\frac{1}{1+\exp(-y_iW^T\boldsymbol{x}_i)}=-\sum_{i=1}^{N}\ln\sigma(z_i)$$

$$(z_i=y_iW^T\boldsymbol{x}_i)$$

Logistic 回归的算法

- Logistic回归就是要求W,使得$NLL(W)$最小。
- 用梯度下降法来求解：

$$\frac{\partial NLL}{\partial W} = -\sum_{i=0}^{\mathrm{N}} \frac{1}{\sigma(z_i)} \sigma(z_i)[1-\sigma(z_i)] y_i \boldsymbol{x}_i$$

$$= -\sum_{i=1}^{N} \frac{\exp(-y_i W^T \boldsymbol{x}_i)}{1+\exp(-y_i W^T \boldsymbol{x}_i)} y_i \boldsymbol{x}_i$$

$$= -\sum_{i=1}^{N} \frac{1}{1+\exp(y_i W^T \boldsymbol{x}_i)} y_i \boldsymbol{x}_i$$

Logistic 回归的算法

- 用牛顿法来求解：

$$\frac{\partial^2 NLL}{\partial W^T \partial W} = \sum_{i=1}^{N} \frac{\exp(y_i W^T \boldsymbol{x}_i)\boldsymbol{x}_i {\boldsymbol{x}_i}^T}{(1+\exp(y_i W^T \boldsymbol{x}_i))^2}$$

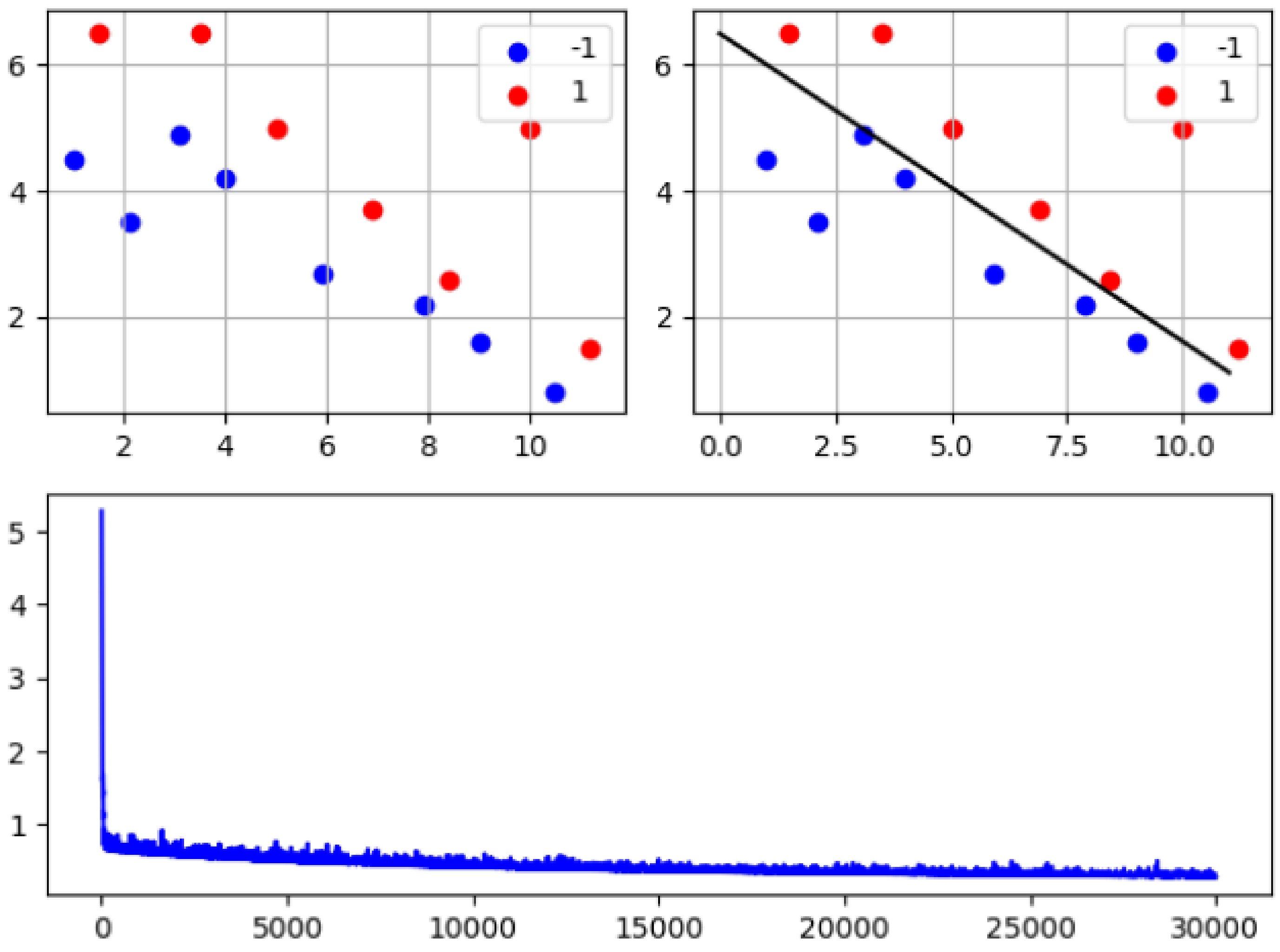
$$= X diag[\sigma(-y_i W^T \boldsymbol{x}_i)(1-\sigma(-y_i W^T \boldsymbol{x}_i))]X^T$$

$$= \ XSX^T$$

（ X =[$\boldsymbol{x}_1, \boldsymbol{x}_2, \dots \boldsymbol{x}_N$] , S是一个对角阵）

可以发现 XSX^T 半正定。

Logistic 回归的例子



梯度下降法

- 梯度下降法(gradient descent)
- 最速下降法(steepest descent)
- 梯度下降法是一种迭代算法。选取适当的初值x(0)，不断迭代，更新x的值，进行目标函数的极小化，直到收敛。由于负梯度方向是使函数值下降最快的方向，在迭代的每一步，以负梯度方向更新x的值，从而达到减少函数值的目的。

梯度下降法

- 假设$f(x)$具有一阶连续偏导数的函数：$\min\limits_{x\in \mathbf{R}^n} f(x)$
- 一阶泰勒展开：　$f(x)=f(x^{(k)})+g_k^{\mathrm{T}}(x-x^{(k)})$
- $f(x)$在$x^{(k)}$的梯度值：$g_k=g(x^{(k)})=\nabla f(x^{(k)})$

$$x^{(k+1)} \leftarrow x^{(k)}+\lambda_k p_k$$

- 负梯度方向：

$$p_k=-\nabla f(x^{(k)})$$

$$f(x^{(k)}+\lambda_k p_k)=\min_{\lambda\geq 0} f(x^{(k)}+\lambda p_k)$$

无约束最优化问题

- 牛顿法（Newton method）
- 拟牛顿法（quasi Newton method）
- 有收敛速度快的优点。

- 牛顿法是迭代算法，每一步需要求解目标函数的海赛矩阵的逆矩阵，计算比较复杂。
- 拟牛顿法通过正定矩阵近似海赛矩阵的逆矩阵或海赛矩阵，简化了这一计算过程。

牛顿法

- 无约束最优化问题：$\min\limits_{x\in\mathbf{R}^n} f(x)$
- 假设$f(x)$具有二阶连续偏导数，若第k次迭代值为$x^{(k)}$，则可将$f(x)$在$x^{(k)}$附近进行二阶泰勒展开:

$$f(x)=f(x^{(k)})+g_k^{\mathrm{T}}(x-x^{(k)})+\frac{1}{2}(x-x^{(k)})^{\mathrm{T}}H(x^{(k)})(x-x^{(k)}) \qquad \text{B.2}$$

- $g_k=g(x^{(k)})=\nabla f(x^{(k)})$ 是$f(x)$的梯度向量在$x^{(k)}$的值。
- $H(x^{(k)})$ 是$f(x)$的海塞矩阵 在点$x^{(k)}$的值$H(x)=\left[\frac{\partial^2 f}{\partial x_i \partial x_j}\right]_{n\times n}$

牛顿法

- 函数$f(x)$有极值的必要条件是:在极值点处一阶导数为0, 即梯度向量为0。
- 特别是当$H(x^{(k)})$是正定矩阵时，函数$f(x)$的极值为极小值。
- 利用条件： $\nabla f(x)=0$
- 设迭代从$x^{(k)}$开始，求目标函数的极小点

$$\nabla f(x^{(k+1)})=0$$

$$\nabla f(x)=g_k+H_k(x-x^{(k)})$$

$$H_k=H(x^{(k)}) \quad g_k+H_k(x^{(k+1)}-x^{(k)})=0$$

牛顿法

$$x^{(k+1)} = x^{(k)} - H_k^{-1} g_k$$

$$x^{(k+1)} = x^{(k)} + p_k$$

$$H_k p_k = -g_k$$

牛顿法

- 算法步骤：

输入：目标函数 $f(x)$，梯度 $g(x)=\nabla f(x)$

海赛矩阵 $H(x)$，精度要求 ε

输出：$f(x)$ 的极小点 x^*.

（1）取初始点 $x^{(0)}$，置 $k=0$

（2）计算 $g_k=g(x^{(k)})$

（3）若 $\|g_k\|<\varepsilon$，则停止计算，得近似解 $x^*=x^{(k)}$

（4）计算 $H_k=H(x^{(k)})$，并求 p_k

$$H_k p_k=-g_k$$

求逆

（5）置 $x^{(k+1)}=x^{(k)}+p_k$

（6）置 $k=k+1$，转（2）.

拟牛顿法

- 考虑用一个n阶矩阵$G_k = G(x^{(k)})$来近似代替 $H_k^{-1} = H^{-1}(x^{(k)})$

$$\nabla f(x) = g_k + H_k(x - x^{(k)})$$

$$g_{k+1} - g_k = H_k(x^{(k+1)} - x^{(k)})$$

记 $y_k = g_{k+1} - g_k$， $\delta_k = x^{(k+1)} - x^{(k)}$

- 拟牛顿条件： $\boxed{y_k = H_k \delta_k}$ $\boxed{H_k^{-1} y_k = \delta_k}$
- 由：

$$x = x^{(k)} + \lambda p_k = x^{(k)} - \lambda H_k^{-1} g_k$$

$$f(x) = f(x^{(k)}) + g_k^{\mathrm{T}}(x - x^{(k)}) + \frac{1}{2}(x - x^{(k)})^{\mathrm{T}} H(x^{(k)})(x - x^{(k)})$$

$$f(x) = f(x^{(k)}) - \lambda g_k^{\mathrm{T}} H_k^{-1} g_k$$

拟牛顿法

- 如果H_k是正定的， H_k^{-1}也是正定的，那么可以保证牛顿法搜索方向P_k是下降方向，因为搜索方向 $p_k = -\lambda g_k$

- 由B.8得： $x = x^{(k)} + \lambda p_k = x^{(k)} - \lambda H_k^{-1} g_k$

- 由B.2得： $f(x) = f(x^{(k)}) - \lambda g_k^{\mathrm{T}} H_k^{-1} g_k$

  因 H_k^{-1} 正定，故有 $g_k^{\mathrm{T}} H_k^{-1} g_k > 0$

  当 λ 为一个充分小的正数时，总有 $f(x) < f(x^{(k)})$

- 将G_k作为 H_k^{-1} 的近似 $G_{k+1} y_k = \delta_k$ ，拟牛顿条件

拟牛顿法

- 在每次迭代中可以选择更新矩阵

$$G_{k+1} = G_k + \Delta G_k$$

- Broyden类优化算法：
  - DFP(Davidon-Fletcher-Powell)算法(DFP algorithm)
  - BFGS(Broyden-Fletcher-Goldfarb-Shanno)算法(BFGS algorithm)
  - Broyden类算法(Broyden's algorithm)

DFP(Davidon-Fletcher-Powell)算法

- 假设G_{k+1}由G_k加上两个附加项构成：

$$G_{k+1} = G_k + P_k + Q_k \qquad G_{k+1} y_k = G_k y_k + P_k y_k + Q_k y_k$$

- 为使G_{k+1}满足拟牛顿条件，可使P和Q满足

$$P_k y_k = \delta_k \qquad Q_k y_k = -G_k y_k$$

- 得： $P_k = \dfrac{\delta_k \delta_k^{\mathrm{T}}}{\delta_k^{\mathrm{T}} y_k} \qquad Q_k = -\dfrac{G_k y_k y_k^{\mathrm{T}} G_k}{y_k^{\mathrm{T}} G_k y_k}$

- G_k正定 $G_{k+1} = G_k + \dfrac{\delta_k \delta_k^{\mathrm{T}}}{\delta_k^{\mathrm{T}} y_k} - \dfrac{G_k y_k y_k^{\mathrm{T}} G_k}{y_k^{\mathrm{T}} G_k y_k}$

DFP(Davidon-Fletcher-Powell)算法

输入：目标函数 $f(x)$，梯度 $g(x)=\nabla f(x)$，精度要求 ε；

输出：$f(x)$ 的极小点 x^*.

（1）选定初始点 $x^{(0)}$，取 G_0 为正定对称矩阵，置 $k=0$

（2）计算 $g_k=g(x^{(k)})$. 若 $\|g_k\|<\varepsilon$，则停止计算，

得近似解 $x^*=x^{(k)}$；否则 转 (3)

（3）置 $p_k=-G_k g_k$

（4）一维搜索：求 λ_k 使得

$$f(x^{(k)}+\lambda_k p_k)=\min_{\lambda\geq 0} f(x^{(k)}+\lambda p_k)$$

（5）置 $x^{(k+1)}=x^{(k)}+\lambda_k p_k$

（6）计算 $g_{k+1}=g(x^{(k+1)})$，若 $\|g_{k+1}\|<\varepsilon$，则停止计算，得近似解 $x^*=x^{(k+1)}$

BFGS(Broyden-Fletcher-Goldfarb-Shanno)算法

- 可以考虑用G_k逼近海赛矩阵的逆矩阵H^{-1}，也可以考虑用B_k逼近海赛矩阵H, 这时，相应的拟牛顿条件是：$B_{k+1}\delta_k = y_k$

- 用同样的方法得到另一迭代公式。首先令

$$B_{k+1}\delta_k = B_k\delta_k + P_k\delta_k + Q_k\delta_k \qquad B_{k+1} = B_k + P_k + Q_k$$

- 考虑使P_k和Q_k满足：$P_k\delta_k = y_k \qquad Q_k\delta_k = -B_k\delta_k$

- B_{k+1}的迭代公式：

$$B_{k+1} = B_k + \frac{y_k y_k^{\mathrm{T}}}{y_k^{\mathrm{T}}\delta_k} - \frac{B_k\delta_k\delta_k^{\mathrm{T}}B_k}{\delta_k^{\mathrm{T}}B_k\delta_k} \qquad \text{B.30}$$

BFGS(Broyden-Fletcher-Goldfarb-Shanno)算法

- 我们可以从BFGS算法矩阵 $\boldsymbol{B}_k$ 的迭代公式(B.30)得到BFGS算法关于 G_k 的迭代公式。
- 事实上，记 $G_k = B_k^{-1}$，$G_{k+1} = B_{k+1}^{-1}$
- 对B.30两次应用Sherman-Morrison公式,即得

$$G_{k+1} = \left(I - \frac{\delta_k y_k^{\mathrm{T}}}{\delta_k^{\mathrm{T}} y_k} \right) G_k \left(I - \frac{\delta_k y_k^{\mathrm{T}}}{\delta_k^{\mathrm{T}} y_k} \right)^{\mathrm{T}} + \frac{\delta_k \delta_k^{\mathrm{T}}}{\delta_k^{\mathrm{T}} y_k}$$

- 称为BFGS算法关于Gk的迭代公式
- 由DFP算法得到的公式，和BFGS得到的公式线性组合：

$$G_{k+1} = \alpha G^{\mathrm{DFP}} + (1-\alpha) G^{\mathrm{BFGS}}$$