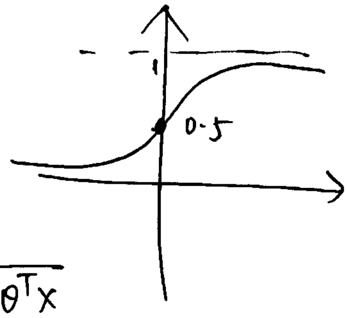
logistic regression

- Sigmoid function: $g(z) = \frac{1}{1+e^{-z}}$



$\Updownarrow$

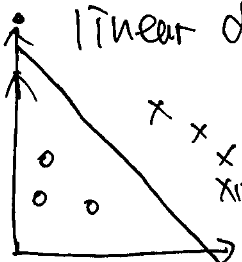
$$h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$$

= estimated prob for $y=1$, based on input x

eg. $X = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ sex \\ height \\ weight \end{pmatrix}$ 若 $h_\theta(x) = 0.7 \Leftrightarrow p(y=1|x,\theta) = 0.7$

这个人有 70% chance 有好身材

- linear decision boundray



$x_1+x_2>3$ (predict $y=1$)

$x_1+x_2<3$ (predict $y=0$) $x_1+x_2=3 \to$ decision boundray

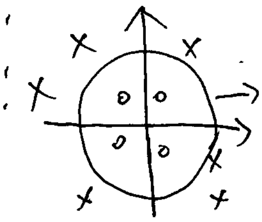
eg. $h_\theta(x) = g(\theta_0+\theta_1 x+\theta_2 x)$

$\theta_0 = -3$ $\theta_1 = 1$ $\theta_2 = 1$

即 $h_\theta(x) = x_1+x_2-3$

predict $y=1$ if $-3+x_1+x_2 \geq 0$

- non-linear decision boundray



$\to x_1^2+x_2^2=1 \to$ decision boundray

$$h_\theta(x) = g(\theta_0+\theta_1 x+\theta_2 x + \theta_3 x_1^2 + \theta_4 x_2^2)$$

$$= g(x_1^2+x_2^2)$$

predict $y=1$ ~~if $1+x_1^2+x_2^2$~~

if $-1+x_1^2+x_2^2 \geq 0$

- core: how to fit $\theta = [\theta_0 \cdots \theta_n]$? input: $\{(x_1,y_1) \cdots (x_m,y_m)\}$ $x \in R^{n+1}$

不能用最小二乘. $\because J(\theta) = \frac{1}{m}\sum_1^m \frac{1}{2}(h_\theta(x_i)-y_i)^2$

$\because J(\theta)$ 会是 non-convex.

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

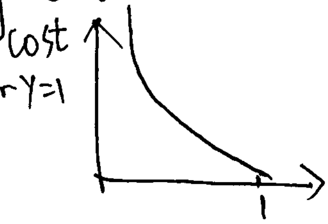
① 若 label=1, 而 $h_\theta(x)=0$, ← y=1 时 prob

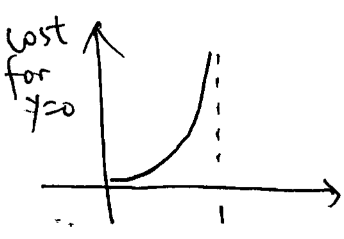
则会给一个很大的 penalty

② 若 label=0, 而 $h_\theta(x)=1$

则会给一个很大 penalty

- why choose this?





再加入正则项得损失函数 $J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y_i \log(h_\theta(x_i)) + (1-y_i)\log(1-h_\theta(x_i))\right] + \lambda/2m\sum_{j=1}^{n}\theta_j^2$

$\lambda/2m\sum_{j=1}^{n}\theta_j^2$

用 gd fit θ, s.t. $\min_\theta J(\theta)$

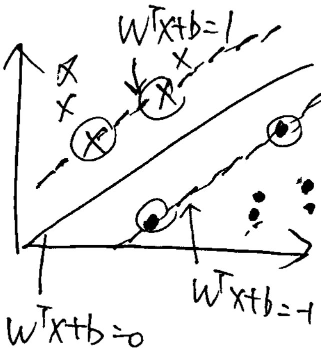
repeat: $\theta_j = \theta_j - \alpha \cdot \frac{\partial J(\theta)}{\partial \theta_j}$

α ↑ 步长.

$\boxed{\text{SVM}}$ 给定 $D=\{(x_1,y_1)\cdots(x_m,y_m)\}$, $y_i \in \{-1,1\}$, 找到超平面 $(W^Tx+b=0)$ 将不同类别分开, 正确分类条件 $\begin{cases} W^Tx_i+b>0 & y_i=1 \\ W^Tx_i+b<0 & y_i=-1 \end{cases}$

SVM 长了个 lu 目股、条件变为 $\begin{cases} W^Tx_i+b\geq 1 & y_i=1 \\ W^Tx_i+b\leq -1 & y_i=-1 \end{cases}$

间隔带宽 $d=\frac{2}{\|W\|}$



so. 求 $\max_{w,b}\frac{2}{\|W\|}$, s.t. $y_i(W^Tx_i+b)\geq 1$ $i=1..m$

也就是 $\min_{w,b}\frac{\|W\|^2}{2}$, s.t. $y_i(W^Tx_i+b)\geq 1$ $i=1...m$. — primal

∵无法直接解 用对偶问题解 $(w,b$, 引入拉格朗日函数 $L(w,b,\alpha) = \frac{1}{2}\|w\|^2 + \sum_{i=1}^{m}\alpha_i(1-y_i(w^Tx_i+b))$

$\frac{\partial L(w,b,\alpha)}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{m}\alpha_i y_i x_i$

$\frac{\partial L(w,b,\alpha)}{\partial b} = 0 \Rightarrow b = \sum_{i=1}^{m}\alpha_i y_i$

} 代入 $L(w,b,\alpha)$ ~~primal~~ 得 dual 问题

dual: $\max_\alpha \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j x_i^T x_j$ s.t. $\sum_{i=1}^{m}\alpha_i y_i = 0$. $\alpha_i \geq 0$ $i=1\cdots m$

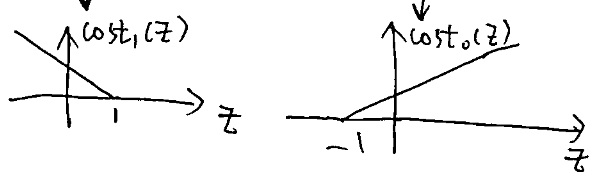
此时 objective: $\arg\max_\alpha \left(\sum_i \alpha_i - \frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j y_i y_j x_i^T x_j\right)$ s.t. $\sum_i^m \alpha_i y_i = 0$

找出 α^* 后

$\Rightarrow W^* = \sum_i \alpha_i^* y_i x_i$

SVM 的 cost function: $\min_\theta C\sum_{i=1}^{m}[y_i \text{cost}_1(\theta^Tx_i) + (1-y_i)\text{cost}_0(\theta^Tx_i)] + \frac{1}{2}\sum_i^m\theta_j^2$

hinge loss: 分类对 损失为0



C: 软间隔 若希望分错点越少越好, 则大C.

若C很小, 对outlier不理会.

∴ SVM 对outlier 不敏感. LR 敏感

梯度下降： 以 $h_\theta = \sum_{j=0}^{n} \theta_j x_j$ 为例.

损失函数 $J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$

① BGD ：批量梯度下降. 目的：求解weights，使误差函数尽可能小

先随机化 weights. 不断更新 weights s.t. J减小.

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

↑ learning rate ：每次朝着J最陡峭的方向迈步

对点(x, y)

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{\partial \left(\frac{1}{2}(h_\theta(x) - y)^2\right)}{\partial \theta_j} = 2 \cdot \frac{1}{2}(h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j}(h_\theta(x) - y)$$

$$= (h_\theta(x) - y) \frac{\partial \left(\sum_{i=0}^{n} \theta_i x_i - y\right)}{\partial \theta_j} = (h_\theta(x) - y) x_j$$

则对all data points: 偏导求和 $\frac{\partial J(\theta)}{\partial \theta_j} = -\frac{1}{m}\sum_{i=1}^{m} (y^i - h_\theta(x^i)) x_j^i$

repeat {

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

for every j=0…n #：有n个参数

}

• 每次训练都用到全部data 很耗时.

• 但迭代次数较少

② SGD ：随机梯度下降

(为了解决 BGD 太慢)

for i=1…m

$\theta_j' = \theta_j + (y^i - h_\theta(x^i)) x_j^i$ (for j=0……n)

sgd：通过每个样本来更新迭代一次. 但噪音比bgd多，s.t. sgd 不是每次朝着整体最优方向. 次数比bgd多.

缺点：更新方向完全依赖当前batch，更新不稳定. ∴可以引入 momentum (更新时一定程度保留之前更新的方向)

③ min_batch gd = MBGD

(BGD和SGD的折衷) 平衡最好!

s.t. 快且 high acc.

Repeat { for i=1, 11, 21 … 991 {

$\theta_j' = \theta_j - \alpha \cdot \frac{1}{10}\sum_{k=i}^{i+9} \left[h_\theta(x^{(k)}) - y^{(k)}\right] x_j^{(k)}$

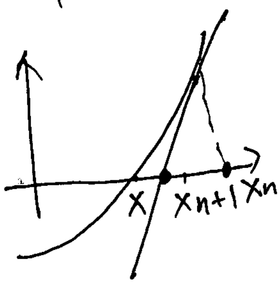
for every j=0…n)

}

}

缺点：不能保证很好的收敛性.

牛顿法 应用 $\begin{cases} \text{求方程根} \\ \text{求最优化办法} \end{cases}$



1) 迭代求根. $f(x)=0$

$$f(x_0+\Delta x)=f(x_0)+f'(x_0)*\Delta x \quad 即 \quad f(x)=f(x_0)+f'(x_0)(x-x_0)$$

一阶 Talyor

令 $f(x)=0$ 得迭代公式: $x=x_0-f(x_0)/f'(x_0)$

$$\Downarrow$$

$$\underline{x_{n+1}=x_n-f(x_n)/f'(x_n)}$$ 逐步迭代 求最优解

2) 扩展到非线性最优化问题

2阶Talayer: $f(x)=f(x_0)+f'(x_0)(x-x_0)+\frac{1}{2}f''(x_0)(x-x_0)^2+\underbrace{\cancel{o(x(x-x_0))}}_{\text{ignore}}$ $o((x-x_0)^3)$

代入 $x=x_0+\Delta x$ 得 $f(x)=f(x_0+\Delta x)=f(x_0)+f'(x_0)\cdot\Delta x+\frac{1}{2}f''(x_0)(\Delta x)^2$

$f'(x)=f'(x_0+\Delta x)=0 \Rightarrow [f(x_0)+f'(x_0)(x-x_0)+\frac{1}{2}f''(x_0)(x-x_0)^2]'=0$

$\Rightarrow f'(x_0)+f''(x_0)(x-x_0) \quad +\frac{1}{2}f''(x_0)\cdot 2(x-x_0)=0$

$\Rightarrow f'(x_0)+f''(x_0)(x-x_0)=0 \quad \Rightarrow x=x_0-f'(x_0)/f''(x_0) \quad \Rightarrow x_{n+1}=x_n-\frac{f'(x_n)}{f''(x_n)}$

用了二阶导, 收敛更快

$\boxed{\text{Hessian}}$ 对于多变量问题, 牛顿法演变为 $x_{n+1}=x_n-\frac{J_f(x_n)}{H(x_n)}$

$$=x_n-H^{-1}(x_n)\cdot J_f(x_n)$$

J: 雅克比矩阵, $J_f(x_n)=\begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \vdots & & \\ \frac{\partial y_m}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix}$

H: Hessian矩阵. 起到了控制步长的 $\cancel{H_f(x_n)}$ $H(f)=\begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 x_n} \\ \vdots & & & \\ \frac{\partial^2 f}{\partial x_n x_1} & \cdots & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$

作用: 特征值 更新步长

$H=E\begin{bmatrix}\lambda_1 & & \\ & \ddots & \\ & & \lambda_n\end{bmatrix}E^T$ E是单位特征向量矩阵

H 非正定时, 无法收敛

H 矩阵维度过大, 带来巨大计算量

~~例 f'' >0 若 $\frac{\partial^2 f}{\partial x^2}>0$ 则~~

例如 一元函数 $f(x)=x^2$. $f'(x)=2x$, $f''(x)=2$ ∵ $f'(x)=0$ $x=0$时, 而 $f''(x)>0$ ∴ $x=0$ 为极小值

类似的 ∴ $|H|=0$ 时, 无法确定函数是否取得极值, So $\begin{cases} \text{正定: 临界点 极小} & (\lambda_i 全>0) \\ \text{负定: 临界点 极大} & (\lambda_i<0) \\ \text{不定: 鞍点} & (\text{o.w.}) \end{cases}$

Momentum. 通过加入 γV_{t-1}，加速 sgd，且抑制震荡

$V_t = \gamma V_{t-1} + \eta \nabla_\theta J(\theta)$ ，$\theta = \theta - V_t$ ，例如 ① 从山顶滚下来，有阻力滚得慢

加入的这一项可以使梯度方向不变的维度速度变快，梯度方向改变的维度更新速度变慢，这样可以加速并↓震荡。 $\gamma = 0.9$

Adagrad：对低频参数做较大更新，高频的较小更新，∴对稀疏数据表现好

Adadelta：对 Adagrad 改进

手推公式

xgboost

- 如何集成：

基础：decision tree

集成：一个个加的

$\hat{y}_i^{(0)} = 0$

$\hat{y}_i^{(1)} = f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i)$

$\hat{y}_i^{(2)} = f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i)$

$\cdots$

$\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)$ 加入一个新函数预测

$\hat{y}_i^{(t-1)}$：保留 t-1 轮的模型预测

$\hat{y}_i^{(t)}$：第t轮的预测模型

- 每一轮加入的模型如何构造？ 方案：优化了我的 objective

$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$ （T：叶子个数）

$$obj^{(t)} = \sum_{i=1}^{n} L(y_i, \hat{y}_i^{(t)}) + \sum_{i=1} \Omega f(i)$$

$$= \sum_{i=1}^{n} L(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) + C$$

防止权重太大：限制叶子个数，权重正则化

- 目标函数：

① 保证 bias 小 ② 保证树模型最精简

Talyor 展开：$f(x+\Delta x) \approx f(x) + f'(x)\Delta x + \frac{1}{2} f''(x) \Delta x^2$ （Δx：新增；$f(x)$：$L(y_i, \hat{y}^{(t-1)})$）

$g_i =$ 一阶导 $= \partial_{\hat{y}^{(t-1)}} L(y_i, \hat{y}^{(t-1)})$ ，$h_i =$ 二阶导 $= \partial^2_{\hat{y}^{(t-1)}} L(y_i, \hat{y}^{(t-1)})$

$$obj^{(t)} \approx \sum_{i=1}^{n} [L(y_i, \hat{y}_i^{(t-1)}) + g_i \cdot f_t(x_i) + \frac{1}{2} h_i \cdot f_t^2(x_i)] + \Omega(f_t) + C = \sum_{i}^{n} (g_i \cdot f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)) + \Omega(f_t)$$

$n \to$ 样本上遍历

$$Obj^{(t)} \simeq \sum_{i=1}^{n}[g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)] + \Omega(f_t)$$

$$= \sum_{1}^{n}[g_i \cdot w_q(x_i) + \frac{1}{2}h_i w_q^2(x_i)] + \gamma T + \lambda \cdot \frac{1}{2}\sum_{j=1}^{T} w_j^2$$

统计每个叶子节点有多少样本

$$= \sum_{j=1}^{T}[(\sum_{i\in I_j} g_i) w_j + \frac{1}{2}(\sum_{i\in I_j} h_i + \lambda) w_j^2] + \gamma T$$

叶子结点上遍历. ∵我们只关心最后结果

$$G_j = \sum_{i\in I_j} g_i$$

$$H_j = \sum_{i\in I_j} h_i$$

$$Obj^{(t)} = \sum_{j=1}^{T}[G_j \cdot w_j + \frac{1}{2}(H_j+\lambda) w_j^2] + \gamma T$$

求导. $\frac{\partial J(f_t)}{\partial w_j} = G_j + (H_j+\lambda)w_j = 0$

$\Rightarrow w_j = -\frac{G_j}{H_j+\lambda}$

代入

$$\therefore \underbrace{Obj = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j+\lambda} + \gamma T}_{\text{structure score}} \quad ☆$$

model performance

分数越小. 效果越好

$$Gain = \frac{1}{2}\Big[\underbrace{\frac{G_L^2}{H_L+\lambda}}_{\text{左子树分数}} + \underbrace{\frac{G_R^2}{H_R+\lambda}}_{\text{右子树分数}} - \underbrace{\frac{(G_L+G_R)^2}{H_L+H_R+\lambda}}_{\text{不分割我们可以拿到的分数}}\Big] - \gamma$$

γ — 加入新叶子节点引入的复杂度代价

$\boxed{\text{GBDT}}$：前一轮学习器为 $f_{t-1}(x)$、对应的损失函数为 $L(y, f_{t-1}(x))$

新一轮迭代：找到一个弱分类器 $h_t(x)$、s.t. $L(y, f_{t-1}(x) + h_t(x))$ 达到最小

用 loss function 的负梯度在当前模型的值. $-\left[\frac{\partial L(y, f(x_i))}{\partial f(x_i)}\right]_{f(x)=}$

作为回归问题中提升树算法残差的近似值.

① 初始化弱分类器，估计使 Loss function 极小化的一个常数值，此时仅有一个根节点. $f_0(x) = \arg\min_c \sum_{i=1}^{N} L(y_i, c)$

迭代轮数 1, 2 … M

对 $i=1 \ldots N$，计算 loss function 的负梯度值在当前模型的值，$r_{mi} = -\left.\frac{\partial L(y, f(x_i))}{\partial f(x_i)}\right|_{f(x)}$

② 对 r_{mi} 拟合一个回归树，得到第 m 棵树叶结点区域 R_{mj} . $j=1 \ldots J$

对 $j=1 \ldots J$ 计算 $c_{mj} = \arg\min_c \sum_{x_i \in R_{mj}} L(y_i, f_{m-1}(x_i) + c)$

更新回归树 $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J} c_{mj} I(x \in R_{mj})$

③ 输出模型：$\hat{f}(x) = f_M(x) = \sum_{m=1}^{M}\sum_{j=1}^{J} c_{mj} I(x \in R_{mj})$