

XgBoost

核心思想：

- 1) 根据样本构建一个叶子节点，作为 initial guess.
- 2) 计算 residual，然后用树去 predict (拟合) residual.
- 3) 在每个树中创建节点时计算 similarity，计算 gain.
- 用 cover 来决定是否创建节点、如何划分.
- 4) 引入 regularization 次，防止过拟合
- 5) 直到模型效果达到要求 / 树的数目达到限制 |

首先看 regression 问题

Loss function:

$$\text{Loss}(y, \hat{p}) = \left[\sum_{i=1}^n L(y_i, p_i) \right] + \lambda T + \frac{1}{2} \gamma O_{\text{value}}$$

$$\rightarrow = \left[\sum_{i=1}^n L(y_i, p_i^{(0)} + O_{\text{value}}) \right] + \lambda T + \frac{1}{2} \gamma O_{\text{value}}^2$$

$$L(y_i, p_i) = \frac{1}{2} (y_i - p_i)^2$$

- 阶导： $g_i = \frac{d}{dp_i} \frac{1}{2} (y_i - p_i)^2 = -(y_i - p_i)$

- Hessian： $h_i = \frac{d^2}{dp_i^2} \frac{1}{2} (y_i - p_i)^2 = \frac{d}{dp_i} -(y_i - p_i) = 1$

为简化 function. $\sum \lambda = 0$

$$\begin{aligned} \text{Loss}(y_i, p_i^{(0)} + O_{\text{value}}) &\approx L(y_i, p_i) + \left[\frac{d}{dp_i} L(y_i, p_i) \right] O_{\text{value}} \\ &\quad + \frac{1}{2} \left[\frac{d^2}{dp_i^2} L(y_i, p_i) \right] O_{\text{value}}^2 \end{aligned}$$

$$\approx L(y_i, p_i) + g_i O_{\text{value}} + \frac{1}{2} h_i O_{\text{value}}^2$$

$$\therefore \frac{d}{dO_{\text{value}}} \left(L(y_i, p_i) + g_i O_{\text{value}} + \frac{1}{2} h_i O_{\text{value}}^2 \right) = g_i + h_i O_{\text{value}}$$

$$2. \frac{d\text{Loss}(y, p)}{d\text{Ovalue}} = \sum_{i=1}^N g_i + (\sum_{i=1}^N h_i + \lambda) \text{Ovalue} = 0$$

$$\Rightarrow \text{Ovalue} = \frac{-(g_1 + g_2 + \dots + g_N)}{(h_1 + h_2 + \dots + h_N) + \lambda}$$

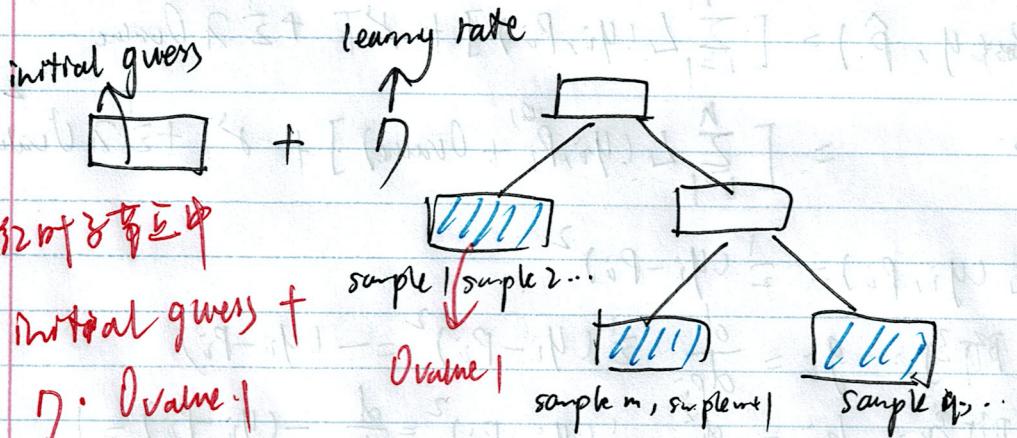
根据之前推导: $-(g_1 + g_2 + \dots + g_N)$

$$-[(p_1 - y_1) + (p_2 - y_2) + \dots + (p_n - y_n)]$$

$$= (y_1 - p_1) + (y_2 - p_2) + \dots + (y_n - p_n)$$

→ sum of residuals

$$h_1 + h_2 + \dots + h_n = N \quad (\text{Number of observations})$$



$\boxed{1111}$ 为最终叶子节点. 分到该叶子节点的 sample

output 为 Ovalue.

之后的一堆推导是为了寻找一个固定值 Ovalue

去让 $\text{Loss}(y, p)$ 最小. 所以让 $\frac{d\text{Loss}(y, p)}{d\text{Ovalue}} = 0$

而直接对其求导很复杂. 则使用泰勒展开.

下面介绍 Xgboost 如何选择树分支。

核心思想：

1) Greedy search. 每个 feature. 每个划分.

选择 Gain 最大的那个部分的 similarity 和划分后 similarity 之和之间的差.

$$\text{similarity} = \frac{(\text{sum of residuals})^2}{\text{Number of observations} + \lambda}$$

2) 根据 Gain 大小来确定是否剪枝.

若 Gain - $\gamma > 0$ 舍

若 Gain - $\gamma < 0$ 剪

3) 根据叶节点的 sample 数来确定是否剪枝

$$\text{cover} = \text{Number of observations}$$

default 是 cover < 1 则剪掉对 regression 无影响

GB 对 classification 有影响.

为什么 similarity score 公式长那样？

现在我们重新回到代价问题上.

我们解出

$$\text{Loss}(y, \hat{p}) = \sum_{i=1}^n L(y_i, \hat{p}_i) + \frac{\lambda}{2} \sum_{i=1}^n O_{value}^2$$

$$\approx \sum L(y_i, p_i) + \sum \frac{d}{dp_i} L(y_i, p_i) \cdot O_{value} + \sum \left[\frac{d^2}{dp_i^2} L(y_i, p_i) \right] O_{value}^2$$

$$\text{其中 } \hat{O}_{value} = \frac{- (g_1 + g_2 + \dots + g_n)}{(h_1 + h_2 + \dots + h_n) + \lambda} \text{ 取最小值. } + \frac{1}{2} \lambda O_{value}^2$$

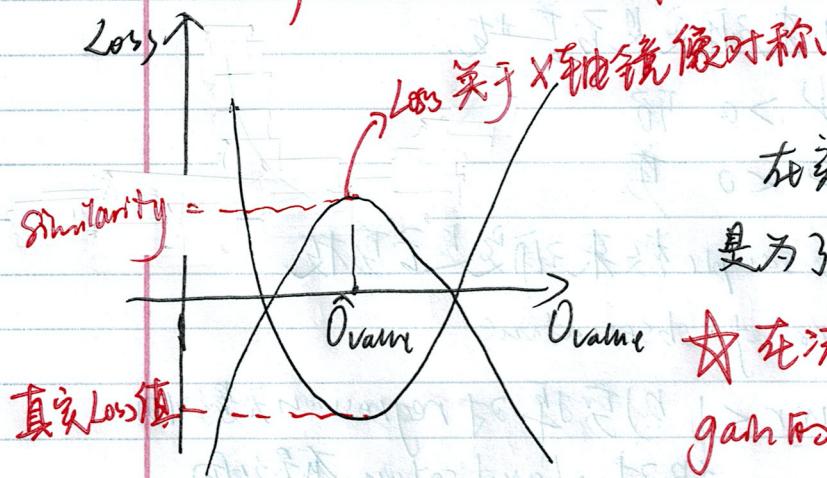
那么将 \hat{O}_{value} 代入 $\text{Loss}(y, \hat{p})$

$$\text{Loss}(y, \hat{p})_{min} = \sum L(y_i, p_i) + \frac{-(\sum_{i=1}^n g_i) \cdot -(\sum_{i=1}^n g_i)}{\sum_{i=1}^n h_i + \lambda}$$

$$\begin{aligned}
 & + \frac{1}{2} \left(\sum_{i=1}^N h_i + \lambda \right) \cdot \frac{\left(\sum_{i=1}^N g_i \right)^2}{\left(\sum_{i=1}^N h_i + \lambda \right)^2} \\
 & = \sum \angle(y_i, p_i) - \frac{\left(\sum_{i=1}^N g_i \right)^2}{\sum_{i=1}^N h_i + \lambda} + \frac{1}{2} \frac{\left(\sum_{i=1}^N g_i \right)^2}{\sum_{i=1}^N h_i + \lambda} \\
 & = \boxed{\sum \angle(y_i, p_i)} - \boxed{\frac{1}{2} \frac{\left(\sum_{i=1}^N g_i \right)^2}{\sum_{i=1}^N h_i + \lambda}}
 \end{aligned}$$

之差 loss predict.

对 residual 的调整.



在实际计算中我们省略 $\lambda/2$

是为了节约计算量,

★ 在决策树中选择最大的 gain 的过程实际上就在选择最小的 Loss !

这里 \lambda 的作用:

在分母加上 \lambda 可以使 gain 的值相对不加 \lambda 变小.

- ① 从而使 \hat{O}_{value} 值偏小. ② 在决策树剪枝过程中.

也更易于使多余的枝被剪掉.

* 对 classification 问题:

这里仅在两互上有不同.

① Loss function 不同.

② 剪枝时 cover 会影响判剪枝结果

$$\rightarrow \text{① } \mathcal{L}(y_i, p_i) = -[y_i \log(\hat{p}_i) + (1-y_i) \log(1-\hat{p}_i)]$$

$$\Rightarrow \mathcal{L}(y_i, \log(\text{odds})_i) = -y_i \log(\text{odds}) + \log(1 + e^{\log(\text{odds})})$$

$$-\frac{\partial}{\partial \log(\text{odds})} \mathcal{L}(y_i, \log(\text{odds})_i) = -y_i + \frac{e^{\log(\text{odds})}}{1 + e^{\log(\text{odds})}} = -(y_i - p_i)$$

$$-\frac{\partial^2}{\partial \log(\text{odds})^2} \mathcal{L}(y_i, \log(\text{odds})_i) = p_i \cdot (1-p_i)$$

$$\rightarrow \text{② } \text{cover} = \sum_{i=1}^N p_i(1-p_i).$$

在其余部分均与 regression 相同

Xgboost 算法优化:

① Approximate greedy algorithm

针对对大数量数据时. 使用 greedy search 太耗时间.

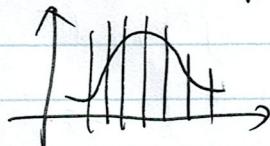
所以采用 approximate greedy algorithm. 用 quantiles

来作为分组界限 (default 一般为 33 quantiles)

< parallel learning

weighted sketches algorithms

原来按每个 sample 分权到 quantile



现在, 按 cover 作为 weight

让每个 quantile 里所有 observation

weight 之和相同.

Intuition: cover 大的意味着对后方块
更不确定 在这样的 quantile 里,

observation ↗

假设 $\hat{P}_i = 0.1 \rightarrow$ 较明朗的分类

$$\text{cover} = 0.1 \times (1 - 0.1) = 0.09 \rightarrow$$
 较小.

$\hat{P}_i = 0.5 \rightarrow$ 分类不是很明朗

$$\text{cover} = 0.5 \times (1 - 0.5) = 0.25 \rightarrow$$
 较大.

因为可以在分 quantiles 时, 将分类明确的组中放进去.

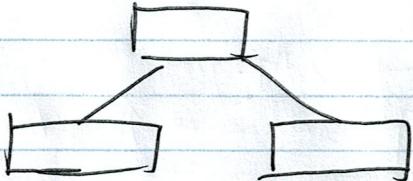
分类不明确的组中 observationⁱ. 从而加快速度.

② 处理 missing value

step 1) 将 dataset 分为两堆

① 该 feature 不缺失

② 该 feature 缺失.



step 2) $\text{fmiss left} = \text{missing} + \text{left}$

$\text{fmiss right} = \text{missing} + \text{right}$.

③ Cache.

Cache gradients/hessians. in cache memory.

从而加速速度.