XGBoost

Decision Tree + Boosting ⟶ BDT ⟶ GBDT ⟶ XGBoost

一、原理

1. Boosting
   - 加法模型 (additive model)
   - 前向分步算法 (Forward Stagewise Algorithm)

加法模型

一般性：基函数没有限制

$$f(x) = \sum_{m=1}^{M} \beta_m b(x; \gamma_m)$$

其中，$b(x;\gamma_m)$ 是基函数，γ_m 是基函数的参数，β_m 为基函数的系数

给定训练集 $\{(x_i,y_i)\}_{i=1}^{N}$ 及 Loss Function $L(y,f(x))$ 的条件下：

$$\min_{\beta_m,\gamma_m} \sum_{i=1}^{N} L\left(y_i, \sum_{m=1}^{M} \beta_m b(x_i;\gamma_m)\right)$$

为了优化此问题，采用前向分步算法。因为是加法模型，每一步只学习一个基函数及其参数，逐步优化目标函数表达式。具体地，每一步只需优化：

$$\min_{\beta,\gamma} \sum_{i=1}^{N} L(y_i, \beta b(x_i;\gamma))$$

前向分步算法

算法1.1 前向分步算法

输入：训练数据集 $T=\{(x_1,y_1),(x_2,y_2),\ldots,(x_N,y_N)\}$；损失函数 $L(y,f(x))$；基函数集合 $\{b(x;\gamma)\}$；

输出：加法模型 $f(x)$

(1) 初始化 $f_0(x)=0$

(2) 对 $m=1,2,\ldots,M$

(a) 极小化损失函数

$$(\beta_m,\gamma_m)=\arg\min_{\beta,\gamma}\sum_{i=1}^{N}L(y_i,f_{m-1}(x_i)+\beta b(x_i;\gamma))$$

得到参数 β_m，γ_m

(b) 更新

$$f_m(x)=f_{m-1}(x)+\beta_m b(x;\gamma_m)$$

---

(3) 得到加法模型

$$f(x)=f_M(x)=\sum_{m=1}^{M}\beta_m b(x;\gamma_m)$$

前向分步算法将同时求解从 $m=1$ 到 M 所有参数 β_m,γ_m 的优化问题简化为逐次求解各个 β_m,γ_m 的优化问题。

2. BDT (Boosting Decision Tree)

Definition: 以决策树为基函数，权重系数为1的提升方法

$$f_M = \sum_{m=1}^{M} T(x;\theta_m)$$

where $T(x;\theta_m)$ 表示决策树；θ_m 为决策树的参数，M为树的个数

依然用 Forward Stagewise Algorithm 训练

$$\begin{cases} f_0(x)=0 \\ f_m = f_{m-1} + T(x;\theta_m) \end{cases}$$

其中，$f_{m-1}(x)$ 为当前模型，通过经验风险极小化(ERM)来确定下一棵决策树的参数 θ_m

$$\hat{\theta} = \arg\min_{\theta_m} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + T(x_i;\theta_m))$$

Remarks: 已知训练集 $T=\{(x_1,y_1),(x_2,y_2)\cdots(x_n,y_n)\}$，$x_i \in \mathcal{X} \in R^n$，$\mathcal{X}$ 为输入空间，$y_i \in \mathcal{Y} \in R$，$\mathcal{Y}$ 为输出空间。如果将输入空间 $\mathcal{X}$ 划分为 J 个互不相交的区域 $R_1, R_2, \ldots, R_J$，并且在每一个区域上确定输出的常量 c_j，则决策树可表示为 $T(x,\theta) = \sum_{j=1}^{J} c_j I(x \in R_j)$。

其中参数 $\theta = \{(R_1,c_1),(R_2,c_2),\cdots,(R_J,c_J)\}$ 表示决策树的区域划分和各区域上的常量值。J 是决策树的复杂度，即叶子结点个数。

提升决策树使用以下前向分步算法：

$$f_0(x) = 0$$

$$f_m(x) = f_{m-1}(x) + T(x;\Theta_m), \quad m = 1,2,\ldots,M$$

$$f_M(x) = \sum_{m=1}^{M} T(x;\Theta_m)$$

在前向分步算法的第 m 步，给定当前模型 $f_{m-1}(x)$，需要求解

$$\hat{\Theta}_m = \arg\min_{\Theta_m} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + T(x_i;\Theta_m))$$

上一个模型与当前模型拟合问题

得到 $\hat{\Theta}_m$，即第 m 棵树的参数。

当采用平方误差损失函数时，

$$L(y, f(x)) = (y - f(x))^2$$

其损失变为

$$\begin{aligned} L(y, f_{m-1}(x) + T(x;\Theta_m)) &= [y - f_{m-1}(x) - T(x;\Theta_m)]^2 \\ &= [r - T(x;\Theta_m)]^2 \end{aligned}$$

当前模型和残差的拟合问题

其中，

$$r = y - f_{m-1}(x)$$

上个模型没学好的部分

是当前模型拟合数据的残差（residual）。对回归问题的提升决策树，只需要简单地拟合当前模型的残差。

回归问题的GDT算法

算法2.1 回归问题的提升决策树算法

输入：训练数据集$T=\{(x_1,y_1),(x_2,y_2),\ldots,(x_N,y_N)\}$；

输出：提升决策树$f_M(x)$

（1）初始化$f_0(x)=0$

（2）对$m=1,2,\ldots,M$

（a）按照式（2.5）计算残差

$$r_{mi}=y_i-f_{m-1}(x_i),\quad i=1,2,\ldots,N$$

(b)拟合残差r_{mi}学习一个回归树，得到$T(x;\Theta_m)$

（c）更新$f_m(x)=f_{m-1}(x)+T(x;\Theta_m)$

（3）得到回归提升决策树

$$f_M(x)=\sum_{m=1}^{M}T(x;\Theta_m)$$

3. GBDT (Gradient Boosting Decision Tree)

GBDT 使用损失函数的负梯度在当前模型的值 $-\left[\frac{\partial L(y,f(x_i))}{\partial f(x_i)}\right]_{f(x)=f_{m-1}(x)}$ 作为回归问题GBDT算法中残差的近似值，拟合一个回归树。

GBDT

算法3.1 梯度提升算法

输入：训练数据集$T=\{(x_1,y_1),(x_2,y_2),\ldots,(x_N,y_N)\}$；损失函数$L(y,f(x))$

输出：梯度提升决策树$\hat{f}(x)$

（1）初始化

$$f_0(x)=\arg\min_c\sum_{i=1}^{N}L(y_i,c)$$

（2）对$m=1,2,\ldots,M$

（a）对$i=1,2,\ldots,N$，计算

$$r_{mi}=-\left[\frac{\partial L(y,f(x_i))}{\partial f(x_i)}\right]_{f(x)=f_{m-1}(x)}$$

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

（c）对$j=1,2,\ldots,J$，计算

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

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

（3）得到回归梯度提升决策树

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

4. XGBoost (extreme Gradient Boosting)

对GBDT的改进 ① 二阶项

② 加上了 Regulazation

总的指导原则：把样本分配到叶子结点会对应一个obj，优化过程就是obj优化，也就是分裂节点到叶子不同的组合，不同的组合对应不同的obj，所有的优化围绕这个思想展开.

假设数据集 $D=\{(x_i, y_i)\}$，其中 $x_i \in R^m$，$y \in R$，$|D|=n$

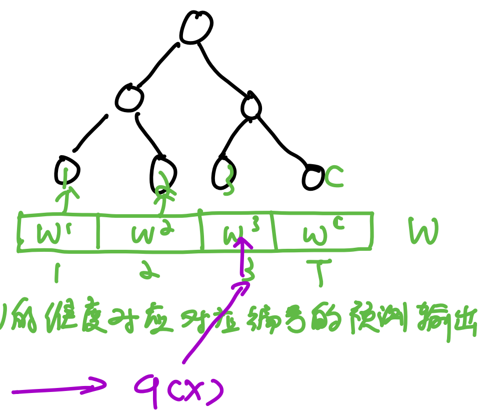
△ 决策树模型 $f(x)=W_{q(x)}$

$q(x)$ 表示样本 x 在某个节点上，$W_{q(x)}$ 是该节点的打分，即该样本的模型预测值.

其中，$q: R^m \to \{1, \cdots, T\}$ 是由输入 x 向叶子结点编号的映射. $w \in R^T$ 是叶子结点向量，

T为决策树叶子节点数.

$$W = [w^1, w^2, \cdots, w^T]^T$$



XGBoost 模型预测输出：

$$\hat{y} = \phi(x_i) = \sum_{i=1}^{K} f_k(x_i)$$，其中 $f_k(x)$ 为第k棵决策树

正则化目标函数：

$$L(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$$

正则化项，Ω指模型的复杂程度

其中，$\Omega(f) = \gamma T + \frac{1}{2}\lambda \|W\|^2 = \underbrace{\gamma T}_{\text{叶结点的个数}} + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$ → 预测输出值

第t轮目标函数：

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

Remarks：泰勒展开式 $f(x^t) = f(x^{t-1} + \Delta x)$

$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f^{(2)}(a)}{2!}(x-a)^2 + \cdots + \frac{f^{(n)}(a)}{n!}(x-a)^n + R_n(x)$

$$= \underbrace{f(x^{t-1})}_{a} + \underbrace{f'(x^{t-1})\Delta x}_{b} + \underbrace{f''(x^{t-1}) \cdot \frac{\Delta x^2}{2}}_{c}$$

第t轮目标函数 $L^{(t)}$ 在 $\hat{y}^{(t-1)}$ 处二阶泰勒展开

$$L^{(t)} = \sum_{i=1}^{n} \Big[\underbrace{l(y_i, \hat{y}^{(t-1)})}_{a} + \underbrace{\partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}) \cdot f_t(x_i)}_{b} + \frac{1}{2} \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}) f_t^2(x_i) \Big] + \Omega(f_t)$$ → c

记 $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$

$h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$

则原式有 $L^{(t)} = \sum_{i=1}^{n}\left[l(y_i, \hat{y}^{(t-1)}) + g_i \cdot f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)\right] + \Omega(f_t)$

移除关于 $f_t(x_i)$ 常数项

$$l^{(t)} = \sum_{i=1}^{n}\left[g_i \cdot f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)\right] + \Omega(f_t)$$

$$l^{(t)} = \boxed{\sum_{i=1}^{n}\left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)\right] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2}$$

△定义叶结点 j 上的样本的下标集合 $I_j = \{i \mid q(x_i) = j\}$，则目标函数可表示为按叶结点累加的形式

→△表示集元素是否包含在某叶子结点下面。

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

$f(x) = W_{q(x)}$ 树的定义

当前叶子结点内每一个元素

△ I_j 被定义为每个叶节点 j 上面样本下标的集合 $I_j = \{i \mid q(x_i) = j\}$，每个样本值都能通过函数 $q(x_i)$ 映射到树上的某个叶子节点，从而把 $\sum_{i=1}^{n} \Rightarrow \sum_{j=1}^{T}$

由于 $w_j^* = \arg\min_{w_j} \hat{l}^{(t)}$，令 $\frac{\partial \hat{l}^{(t)}}{\partial w_j} = 0$，

得到每个叶结点的最优分数为 $w_j = -\frac{(\sum_{i\in I_j} g_i)^2}{\sum_{i\in I_j} h_i + \lambda}$

代入每个叶结点 j 的最优分数，得到最优化目标函数值：

代入每个叶结点 j 的最优分数，得到最优化目标函数值

$$\tilde{\mathcal{L}}^{(t)}(q) = -\frac{1}{2}\sum_{j=1}^{T}\frac{\left(\sum_{i\in I_j} g_i\right)^2}{\sum_{i\in I_j} h_i + \lambda} + \gamma T$$

叶子结点如何分裂？（思想同CART）

假设 I_L 和 I_R 分别为分裂后左右结点的实例集，令 $I = I_L \cup I_R$，则分裂后损失减少量由下式得出

$$\mathcal{L}_{split} = \frac{1}{2}\left[\frac{\left(\sum_{i\in I_L} g_i\right)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{\left(\sum_{i\in I_R} g_i\right)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{\left(\sum_{i\in I} g_i\right)^2}{\sum_{i\in I} h_i + \lambda}\right] - \gamma$$

← 加入新叶子节点引入的复杂度代价

左子树分数　　右子树分数　　不可分割可以得到的分数

用以评估待分裂结点。

算法4.1 分裂查找的精确贪婪算法

输入：当前结点实例集 I；特征维度 d

输出：根据最大分值分裂

(1) $gain \leftarrow 0$

(2) $G \leftarrow \sum_{i\in I} g_i$，$H \leftarrow \sum_{i\in I} h_i$

(3) for $k = 1$ to d do

(3.1) $G_L \leftarrow 0$，$H_L \leftarrow 0$

(3.2) for j in sorted(I, by $\mathbf{x}_{jk}$) do

(3.2.1) $G_L \leftarrow G_L + g_j$，$H_L \leftarrow H_L + h_j$

(3.2.2) $G_R \leftarrow G - G_L$，$H_R = H - H_L$

(3.2.3) $score \leftarrow \max\left(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda}\right)$

(3.3) end

(4) end

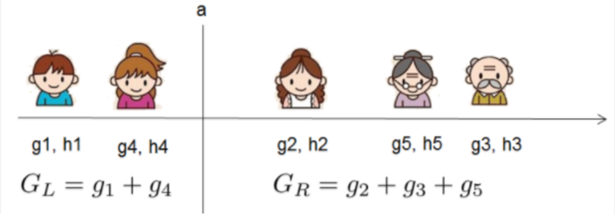
e.g.

第一个值得注意的事情是“对于某个特征，先按照该特征里的值进行排序”，这里举个例子。

比如设置一个值a，然后枚举所有x < a、a < x这样的条件（x代表某个特征比如年龄age，把age从小到大排序：假定从左至右依次增大，则比a小的放在左边，比a大的放在右边），对于某个特定的分割a，我们要计算a左边和右边的导数和。

比如总共五个人，按年龄排好序后，一开始我们总共有如下4种划分方法：
①把第一个人和后面四个人划分开
②把前两个人和后面三个人划分开
③把前三个人和后面两个人划分开
④把前面四个人和后面一个人划分开

接下来，把上面4种划分方法全都各自计算一下Gain，看哪种划分方法得到的Gain值最大则选取哪种划分方法，经过计算，发现把第2种划分方法“前面两个人和后面三个人划分开”得到的Gain值最大，意味着在一分为二这个第一层层面上这种划分方法是最合适的。



换句话说，对于所有的特征x，我们只要做一遍从左到右的扫描就可以枚举出所有分割的梯度和GL和GR。然后用计算Gain的公式计算每个分割方案的分数就可以了。

然后后续则依然按照这种划分方法继续第二层、第三层、第四层、第N层的分裂。

第二个值得注意的事情就是引入分割不一定会使得情况变好，所以我们有一个引入新叶子的惩罚项。优化这个目标对应了树的剪枝， 当引入的分割带来的增益小于一个阀值γ的时候，则忽略这个分割。

换句话说，当引入某项分割，结果分割之后得到的分数 - 不分割得到的分数得到的值太小（比如小于我们的最低期望阀值γ），但却因此得到的复杂度过高，则相当于得不偿失，不如不分割。即做某个动作带来的好处比因此带来的坏处大不了太多，则为避免复杂 多一事不如少一事的态度，不如不做。

相当于在我们发现“分”还不如“不分”的情况下后（得到的增益太小，小到小于阈值γ），会有2个叶子节点存在同一棵子树上的情况。