1. XGBoost原理

$$\begin{aligned} XGBoost &= eXtreme + GBDT \\ &= eXtreme + (Gradient + BDT) \\ &= eXtreme + Gradient + (Boosting + DecisionTree) \end{aligned}$$

$$DecisionTree + Boosting \rightarrow BDT \rightarrow GBDT \rightarrow XGBoost$$

1.1. 提升方法

（Boosting）

1.1.1. 定义

提升方法使用加法模型 + 前向分步算法。

1.1.2. 加法模型

预测模型

$$f(x) = \sum_{m=1}^{M} \beta_m b(x; \gamma_m) \tag{1.1}$$

其中，$b(x; \gamma_m)$为基函数，比如线性回归，逻辑回归，决策树，神经网络等；γ_m为基函数的参数，β_m为基函数的系数。

模型求解

在给定训练数据$\{(x_i, y_i)\}_{i=1}^{N}$及损失函数$L(y, f(x))$的条件下，学习加法模型$f(x)$成为经验风险极小化问题：

$$\min_{\beta_m, \gamma_m} \sum_{i=1}^{N} L\left(y_i, \sum_{m=1}^{M} \beta_m b(x_i; \gamma_m)\right) \tag{1.2}$$

模型问题

因为加法模型是由很多个模型累加而成的，比较难以优化，所以我们利用前向分步算法求解这一优化问题。其思路是：因为学习的是加法模型，可以从前向后，每一步只学习一个基函数及其系数，逐步逼近优化目标函数式（1.2），则可以简化优化复杂度。具体地，每步只需优化如下损失函数：

$$\min_{\beta, \gamma} \sum_{i=1}^{N} L(y_i, \beta b(x_i; \gamma)) \tag{1.3}$$

1.1.3. 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，以得到参数β_m，γ_m

$$(\beta_m, \gamma_m) = \arg\min_{\beta, \gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma)) \tag{1.4}$$

　（b）更新

$$f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m) \tag{1.5}$$

（3）得到加法模型

$$f(x) = f_M(x) = \sum_{m=1}^{M} \beta_m b(x; \gamma_m) \tag{1.6}$$

总结：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。

1.2. 提升决策树

（BDT，Boosting Decision Tree）

1.2.1. 定义

以**决策树为基函数（基函数权重系数 = 1）**的提升方法为提升决策树。提升决策树模型可以表示为决策树的加法模型：

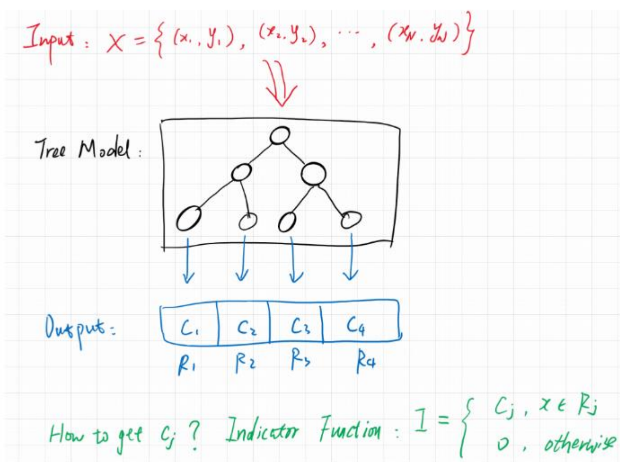
$$f_M = \sum_{m=1}^{M} T(x;\Theta_m) \tag{2.1}$$

其中，$T(x;\Theta_m)$表示决策树；Θ_m为决策树的参数；M为树的个数。

已知训练数据集$T=\{(x_1,y_1),(x_2,y_2),\ldots(x_N,y_N)\}$，$x_i\in\mathcal{X}\subseteq\mathbb{R}^n$，$\mathcal{X}$为输入空间，$y_i\in\mathcal{Y}\subseteq\mathbb{R}$，$\mathcal{Y}$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$R_1,R_2,\ldots,R_J$，并且在每个区域上确定输出的常量$c_j$，那么决策树可表示为

$$T(x;\Theta) = \sum_{j=1}^{J} c_j I\left(x\in R_j\right) \tag{2.4}$$

其中，参数$\Theta=\{(R_1,c_1),(R_2,c_2),\ldots,(R_J,c_J)\}$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。



1.2.2. 前向分步算法

提升决策树采用前向分步算法。首先确定初始提升决策树$f_0(x)=0$（只有一个节点的树），第m步的模型是

$$f_m(x) = f_{m-1}(x) + T(x;\Theta_m) \tag{2.2}$$

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

$$\hat{\Theta}_m = \arg\min_{\Theta_m} \sum_{i=1}^{N} L\left(y_i, f_{m-1}(x_i) + T(x_i;\Theta_m)\right) \tag{2.3}$$

提升决策树使用以下前向分步算法：

$$\begin{aligned} f_0(x) &= 0 \\ f_m(x) &= f_{m-1}(x) + T(x;\Theta_m), \quad m=1,2,\ldots,M \\ f_M(x) &= \sum_{m=1}^{M} T(x;\Theta_m) \end{aligned}$$

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

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

得到$\hat{\Theta}_m$，即第m棵树的参数。

1.2.3. 回归问题的提升决策树

当采用平方误差(MSE)损失函数时，

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

其损失变为

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

其中，

$$r = y - f_{m-1}(x) \tag{2.5}$$

r是当前模型拟合数据的残差（residual），也是上一个模型没有学好的部分。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。

1.2.4. 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

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

1.3. 梯度提升决策树 （GBDT，Gradient Boosting Decision Tree）

1.3.1. 定义

梯度提升算法使用损失函数的负梯度在当前模型的值

$$-\left[\frac{\partial L(y, f(x_i))}{\partial f(x_i)}\right]_{f(x)=f_{m-1}(x)} \tag{3.1}$$

作为回归问题提升决策树算法中残差的近似值，拟合一个回归树。

1.3.2. 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$f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J} c_{mj} I\left(x \in R_{mj}\right)$

（3）得到回归梯度提升决策树

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

1.4. 极限梯度提升（XGBoost，eXtreme Gradient Boosting）

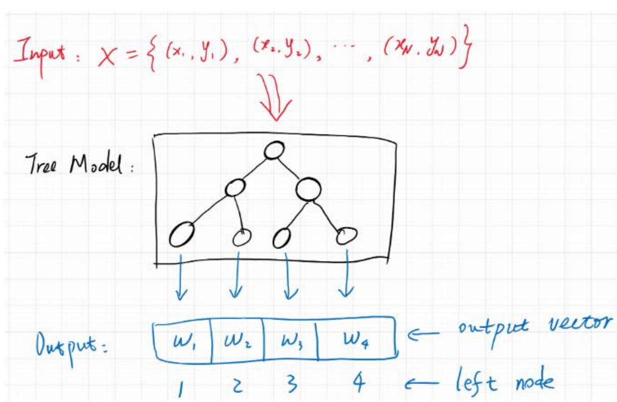
1.4.1. 定义

训练数据集$D = \{(\mathbf{x}_i, y_i)\}$, 其中$\mathbf{x}_i \in \mathbb{R}^m, y_i \in \mathbb{R}, x_i$和$y_i$是列向量, $|D| = n$。

决策树模型

$$f(\mathbf{x}) = w_{q(\mathbf{x})} \tag{4.1}$$

其中，$q : \mathbb{R}^m \rightarrow \{1, \ldots, T\}$是有输入 $\mathbf{x}$ 向叶子结点编号的映射，$w \in \mathbb{R}^T$是叶子结点向量，T为决策树叶子节点数。



提升决策树模型预测输出

$$\hat{y}_i = \phi(\mathbf{x}_i) = \sum_{k=1}^{K} f_k(\mathbf{x}_i) \tag{4.2}$$

其中，$f_k(\mathbf{x})$为第k棵决策树。

1.4.2. 正则化目标函数

$$\mathcal{L}(\phi) = \sum_i l\left(\hat{y}_i, y_i\right) + \sum_k \Omega(f_k) \tag{4.3}$$

其中，$\Omega(f) = \gamma T + \frac{1}{2}\lambda\|w\|^2 = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$，限制叶子节点个数 T，和模型输出 w。

第t轮目标函数

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i)\right) + \Omega(f_t) \tag{4.4}$$

1.4.3. 二阶泰勒展开

第t轮目标函数$\mathcal{L}^{(t)}$在$\hat{y}^{(t-1)}$处的二阶泰勒展开

$$\begin{aligned}\mathcal{L}^{(t)} &\simeq \sum_{i=1}^{n}\left[l\left(y_i, \hat{y}^{(t-1)}\right) + \partial_{\hat{y}^{(t-1)}} l\left(y_i, \hat{y}^{(t-1)}\right) f_t\left(\mathbf{x}_i\right) + \frac{1}{2}\partial^2_{\hat{y}^{(t-1)}} l\left(y_i, \hat{y}^{(t-1)}\right) f_t^2\left(\mathbf{x}_i\right)\right] + \Omega\left(f_t\right) \\ &= \sum_{i=1}^{n}\left[l\left(y_i, \hat{y}^{(t-1)}\right) + g_i f_t\left(\mathbf{x}_i\right) + \frac{1}{2} h_i f_t^2\left(\mathbf{x}_i\right)\right] + \Omega\left(f_t\right)\end{aligned} \tag{4.5}$$

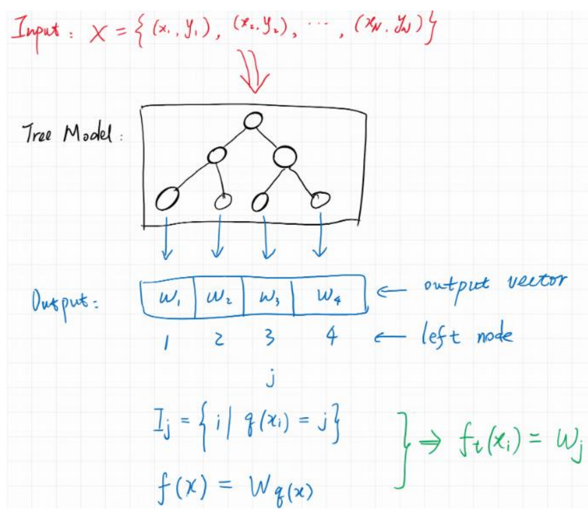
其中，一阶导数：$g_i = \partial_{\hat{y}^{(t-1)}} l\left(y_i, \hat{y}^{(t-1)}\right)$，二阶导数：$h_i = \partial^2_{\hat{y}^{(t-1)}} l\left(y_i, \hat{y}^{(t-1)}\right)$。

第t轮目标函数$\mathcal{L}^{(t)}$的二阶泰勒展开移除关于 $f_t\left(\mathbf{x}_i\right)$常数项

$$\begin{aligned}\tilde{\mathcal{L}}^{(t)} &= \sum_{i=1}^{n}\left[g_i f_t\left(\mathbf{x}_i\right) + \frac{1}{2} h_i f_t^2\left(\mathbf{x}_i\right)\right] + \Omega\left(f_t\right) \\ &= \sum_{i=1}^{n}\left[g_i f_t\left(\mathbf{x}_i\right) + \frac{1}{2} h_i f_t^2\left(\mathbf{x}_i\right)\right] + \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T} w_j^2\end{aligned} \tag{4.6}$$

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

$$\tilde{\mathcal{L}}^{(t)} = \sum_{j=1}^{T}\left[\left(\sum_{i\in I_j} g_i\right) w_j + \frac{1}{2}\left(\sum_{i\in I_j} h_i + \lambda\right) w_j^2\right] + \gamma T \tag{4.7}$$



$$w_j^* = \arg\min_{w_j} \tilde{\mathcal{L}}^{(t)}$$

可令

$$\frac{\partial \tilde{\mathcal{L}}^{(t)}}{\partial w_j} = 0$$

得到每个叶结点j的最优分数为

$$w_j^* = -\frac{\sum_{i\in I_j} g_i}{\sum_{i\in I_j} h_i + \lambda} \tag{4.8}$$

代入每个叶结点j的最优分数，得到最优化目标函数值

$$\tilde{\mathcal{L}}^{(t)}(q) = -\frac{1}{2}\sum_{j=1}^{T}\frac{\left(\sum_{i\in I_j} g_i\right)^2}{\sum_{i\in I_j} h_i + \lambda} + \gamma T \tag{4.9}$$

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

$$\mathcal{L}_{split} = \frac{1}{2}\left[\frac{\left(\sum_{i\in I_L} g_i\right)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{\left(\sum_{i\in I_R} g_i\right)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{\left(\sum_{i\in I} g_i\right)^2}{\sum_{i\in I} h_i + \lambda}\right] - \gamma \tag{4.10}$$

用以评估待分裂结点。

1.4.4. 分裂查找的精确贪婪算法

输入：当前结点实例集$\boldsymbol{I}$;特征维度$\boldsymbol{d}$

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($\boldsymbol{I}$, 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