GBDT 的原理 [1] [2]

July 1, 2020

1 如何在不改变原有模型的结构上提升模型的拟合能力

假设现在你有样本集 $(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)$，然后你用一个模型，如 $F(x)$ 去拟合这些数据，使得这批样本的平方损失函数（即 $\frac{1}{2}\sum_0^n (y_i - F(x_i))^2$）最小。但是你发现虽然模型的拟合效果很好，但仍然有一些差距，比如预测值 $F(x_1) = 0.8$，而真实值 $y_1 = 0.9$，另外你不允许更改原来模型 $F(x)$ 的参数，那么你有什么办法进一步来提高模型的拟合能力呢。

既然不能更改原来模型的参数，那么意味着必须在原来模型的基础之上做改善，那么直观的做法就是建立一个新的模型 $f(x)$ 来拟合 $F(x)$ 未完全拟合真实样本的残差，即 $y - F(x)$。所以新模型需要拟合的样本集就变成了：$(x_1, y_1 - F(x_1)), (x_2, y_2 - F(x_2)), \cdots, (x_n, y_n - F(x_n))$

2 基于残差的 GBDT

在第一部分，$y_i - F(x_i)$ 被称为残差，这一部分也就是前一模型 $(F(x_i))$ 未能完全拟合的部分，所以交给新的模型来完成。

我们知道 GBDT 的全称是 Gradient Boosting Decision Tree，其中 Gradient 被称为梯度，更一般的理解，可以认为是一阶导，那么这里的残差与梯度是什么关系呢。在第一部分，我们提到了一个叫做平方损失函数的东西，具体形式可以写成 $\frac{1}{2}\sum_0^n (y_i - F(x_i))^2$，熟悉其他算法的原理应该知道，**这个损失函数主要针对回归类型的问题，分类则是用熵值类的损失函数**。具体到平方损失函数的式子，你可能已经发现它的一阶导其实就是残差的形式，所以基于残差的 GBDT 是一种特殊的 GBDT 模型，它的损失函数是平方损失函数，常用来处理回归类的问题。具体形式可以如下表示：

损失函数： $L(y, F(x)) = \frac{1}{2}(y_i - F(x))^2$

我们希望最小化： $J = \frac{1}{2}\sum_0^n (y_i - F(x_i))^2$

损失函数的一阶导：

$$\frac{\partial J}{\partial F(x_i)} = \frac{\partial \sum_i L(y_i, F(x_i))}{\partial F(x_i)} = \frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} = F(x_i) - y_i$$

正好残差就是负梯度：

$$y_i - F(x_i) = -\frac{\partial J}{\partial F(x_i)}$$

3 为什么基于残差的 GBDT 不是一个好的选择

基于残差的 gbdt 在解决回归问题上不算是一个好的选择，一个比较明显的缺点就是对异常值过于敏感。所以一般回归类的损失函数会用绝对损失或者 Huber 损失函数来代替平方损失函数：

- 绝对值 (absolute loss)：$L(y, F) = |y - F|$
- Huber 损失 (huber loss)：$L(y, F) = \begin{cases} \frac{1}{2}(y-F)^2 & |y-F| <= \delta \\ \delta|y - f(x)| - \frac{1}{2}\delta^2 & |y-F| > \delta \end{cases}$

4 Boosting 的加法模型

如前面所述，GBDT 模型可以认为是是由 k 个基模型组成的一个加法运算式：

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), f_k \in F$$

其中 F 是指所有基模型组成的函数空间。

那么一般化的损失函数是预测值 $\hat{y}$ 与真实值 y 之间的关系，如我们前面的平方损失函数，那么对于 n 个样本来说，则可以写成：

$$L = \sum_{i=1}^{n} l(y_i, \hat{y}_i)$$

更一般的，我们知道一个好的模型，在偏差和方差上有一个较好的平衡，而算法的损失函数正是代表了模型的偏差面，最小化损失函数，就相当于最小化模型的偏差，但同时我们也需要兼顾模型的方差，所以目标函数还包括抑制模型复杂度的正则项，因此目标函数可以写成：

$$Obj = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

其中 $\Omega(f_k)$ 代表了基模型的复杂度，若基模型是树模型，则树的深度、叶子节点数等指标可以反应树的复杂程度。

对于 Boosting 来说，它采用的是前向优化算法，即从前往后，逐渐建立基模型来优化逼近目标函数，具体过程如下：

$$
\begin{aligned}
&\hat{y}_i^0 = 0 \\
&\hat{y}_i^1 = f_1(x_i) = \hat{y}_i^0 + f_1(x_i) \\
&\hat{y}_i^2 = f_1(x_i) + f_2(x_i) = \hat{y}_i^1 + f_2(x_i) \\
&\dots \\
&\hat{y}_i^t = \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{t-1} + f_t(x_i)
\end{aligned}
$$

那么，在每一步，如何学习一个新的模型呢，答案的关键还是在于 GBDT 的目标函数上，即新模型的加入总是以优化目标函数为目的的。

我们以第 t 步的模型拟合为例，在这一步，模型对第 i 个样本 x_i 的预测为：

$$\hat{y}_i^t = \hat{y}_i^{t-1} + f_t(x_i)$$

其中 $f_t(x_i)$ 就是我们这次需要加入的新模型，即需要拟合的模型，此时，目标函数就可以写成：

$$Obj^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^t) + \sum_{i=i}^{t} \Omega(f_i) = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{t-1} + f_t(x_i)) + \Omega(f_t) + constant$$

即此时最优化目标函数，就相当于求得了 $f_t(x_i)$

5 什么是 GBDT 的目标函数

根据泰勒公式推导二阶导（GBDT 是一阶导，xgboost 是二阶导）：

$$f(x + \Delta x) \approx f(x) + f'(x)\Delta x + \frac{1}{2} f''(x)\Delta x^2$$

建立 Obj 表达式和二阶泰勒展开的对应关系：

- $l(y_i, \hat{y}_i^{t-1})$ 对应泰勒公式中的 $f(x)$
- $\hat{y}_i^{t-1}$ 对应泰勒公式中的 x
- $f_t(x_i)$ 对应泰勒公式中的 Δx
- $l(y_i, \hat{y}_i^{t-1} + f_t(x_i))$ 对应泰勒公式中的 $f(x + \Delta x)$

那么，对 $l(y_i, \hat{y}_i^{t-1} + f_t(x_i))$ 进行二阶泰勒展开后，可以得到：

$$Obj^{(t)} = \sum_{i=1}^{n}[l(y_i, \hat{y}_i^{t-1}) + g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)] + \Omega(f_t) + constant$$

其中，

- g_i 是损失函数的一阶导，对应泰勒公式中的 $f'(x)$
- h_i 是损失函数的二阶导，对应泰勒公式中的 $f''(x)$

注意是对 $\hat{y}_i^{t-1}$ 求导。以平方损失函数为例：

$$\begin{aligned}
&\sum_{i=1}^{n}(y_i - (\hat{y}_i^{t-1} + f_t(x_i)))^2 \\
&g_i = \partial\frac{(\hat{y}_i^{t-1} - y_i)^2}{\hat{y}^{t-1}} = 2(\hat{y}^{t-1} - y_i) \\
&h_i = \partial^2\frac{(\hat{y}_i^{t-1} - y_i)^2}{\hat{y}^{t-1}} = 2
\end{aligned}$$

由于在第 t 步 $\hat{y}_i^{t-1}$ 是是一个已知的值，所以 $l(y_i, \hat{y}_i^{t-1})$ 是一个常数，其对函数优化不会产生影响，因此，可以将 $Obj^{(t)}$ 改写为

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

所以我么只要求出每一步损失函数的一阶和二阶导的值（由于前一步的 $\hat{y}_i^{t-1}$ 是已知的，所以这两个值就是常数）代入上述等式，然后最优化目标函数，就可以得到每一步的 $f(x)$ ，最后根据加法模型得到一个整体模型。

6 未完待续：如何生成一颗新的树

References

[1] “机器学习-一文理解 gbdt 的原理-20171001.” [Online]. Available: https://zhuanlan.zhihu.com/p/29765582

[2] “Gbdt 的原理和应用.” [Online]. Available: https://zhuanlan.zhihu.com/p/30339807