GBDT 模型

Hahabula

2025-02-07

目录

GBDT (Gradient Boosting Decision Tree), 梯度提升树。它是一种基于决策树的集成算法。

- 基本结构：决策树组成的树林
- 学习方式：梯度提升

通过构造一组弱的学习（树），并把多颗决策树的结果累加起来作为最终的预测输出。该算法将决策树与集成思想进行了有效结合。

1 GBDT 详解

接下来将详细介绍 GBDT 模型。

1.1 GBDT 的原理

- 所有弱分类器的结构相加等于预测值
- 每次都以当前预测值为基准，下一个弱分类器去拟合误差函数对预测值的误差
- GBDT 的弱分类器使用的是树模型

1.2 前向分布算法

许多加法模型都有如下形式

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

β_m 是系数, $b(x, \gamma_m)$ 是基函数并带有一组参数 γ_m。这类模型的参数的求解大都采用极小化如下损失函数：

$$\sum_{i=1}^{n} L(y_i, \sum_{m=1}^{M} \beta_m b(x, \gamma_m))$$

但是当样本足够多，问题较为复杂时，如果直接求解，则要估计的参数将会有 $M + \sum_{m=1}^{M} \dim(\gamma_m)$ 个，求解十分耗费算力，于是前向分布算法 (Forward stagewise additive modeling) 被提出了。前向分布算法将原来的问题转换为——一步一步的估计基函数项，当在估计某一基函数项 $\beta_m b(x, \gamma_m)$ 时，其之前的 $\beta_i b(x, \gamma_i), i = 1, 2, ..., m-1$ 将作为定值，不再被改变，故该算法的第 m 步即是取解决如下下问题：

$$\min \sum_{i=1}^{n} L(y_i, f_m(x))$$
$$f_m(x) = f_{m-1}(x) + \beta_m b(x, \gamma_m)$$

其中 $f_{m-1}(x)$ 已知。当我们假定采用平方损失函数时，我们有：

$$\begin{aligned} Loss &= \sum_{i=1}^{n} \frac{1}{2}[y_i - (f_{m-1}(x) + \beta_m b(x, \gamma_m))]^2 \\ &= \sum_{i=1}^{n} \frac{1}{2}[(y_i - f_{m-1}(x)) - \beta_m b(x, \gamma_m)]^2 \\ &= \sum_{i=1}^{n} \frac{1}{2}(r_{mi} - \beta_m b(x, \gamma_m))^2 \end{aligned}$$

$r_{mi} = y_i - f_{m-1}(x_i)$ 是在第 m-1 步拟合后模型的残差，从损失函数得第 m 步的本质在于利用第 m 个基函数对上一步所得残差进行拟合，但这种理解是建立在损失函数为平方损失的情况下。

- 前向的意义："前向" 指的是逐步进行的过程，即算法是从零开始，逐步向解的方向构建和优化。每一步添加一个新的基函数或模型参数，以逐渐逼近最终的目标。与之相对的概念是"后向"（Backward），后向算法通常从一个复杂的模型开始，然后逐步移除不必要的成分，而前向算法则是从简单的模型开始逐步加成。
- 分布的含义："分布" 或" 阶段性"（Stagewise）意味着算法在每一步迭代时，只增加一个新的基函数或参数，而不会对之前的基函数或参数进行重新调整。也就是说，每次迭代仅调整新增的模型成分，已添加的部分在整个过程中保持不变。这种特性使得算法较为保守，每次变化较小，且有利于控制模型的复杂度和避免过拟合。

2 参数估计

假设有一个加法模型：

$$F(\mathbf{x}; \{\beta_m, \mathbf{a}_m\}_1^M) = \sum_{i=1}^{M} \beta_m h(\mathbf{x}; \mathbf{a}_m)$$

$h(\mathbf{x}; \mathbf{a}_m)$ 是一个参数比较简单的基函数；$\mathbf{a} = \{a_1, a_2, ...\}$

2.1 数值优化

需要估计的参数为：

$$\mathbf{P}^* = \arg\min_{\mathbf{P}} \Phi(\mathbf{P})$$

$$\Phi(\mathbf{P}) = E_{y,\mathbf{x}} L(y, F(\mathbf{x}; \mathbf{P}))$$

最优的模型为：

$$F^*(\mathbf{x}) = F(\mathbf{x}; \mathbf{P}^*)$$

参数的组成：

$$\mathbf{P}^* = \sum_{m=0}^{M} \mathbf{p}_m$$

对于参数构成的理解：即所有参数的列向量，每一次拟合模型都相当于往 $\mathbf{P}$ 的某些列上加值/更新值。

2.2 最速下降法 (Steepest-descent)

由前面讨论可得，有如下优化问题：

$$\min_{\mathbf{P}} \Phi(\mathbf{P})$$

则由梯度下降法得到的梯度为：

$$\mathbf{g}_m = \nabla\Phi(\mathbf{P})|_{\mathbf{P}=\mathbf{P}_{m-1}} = [\frac{\partial\Phi(\mathbf{P})}{\partial P_j}|_{\mathbf{P}=\mathbf{P}_{m-1}}]^T$$

由于 $\mathbf{P}_{m-1} = \sum_{i=0}^{m-1} \mathbf{p}_i$, 则得到如下迭代公式：

$$\mathbf{P}_m = \mathbf{P}_{m-1} + \mathbf{p}_m$$

而由最速下降法得:

$$\mathbf{p}_m = -\rho_m \mathbf{g}_m$$

由一维线搜索得步长 ρ_m 有下式决定：由前面讨论可得，有如下优化问题：

$$\min_{\mathbf{P}} \Phi(\mathbf{P})$$

则由梯度下降法得到的梯度为：

$$\mathbf{g}_m = \nabla\Phi(\mathbf{P})|_{\mathbf{P}=\mathbf{P}_{m-1}} = [\frac{\partial\Phi(\mathbf{P})}{\partial P_j}|_{\mathbf{P}=\mathbf{P}_{m-1}}]^T$$

由于 $\mathbf{P}_{m-1} = \sum_{i=0}^{m-1} \mathbf{p}_i$, 则得到如下迭代公式：

$$\mathbf{P}_m = \mathbf{P}_{m-1} + \mathbf{p}_m$$

而由最速下降法得:

$$\mathbf{p}_m = -\rho_m \mathbf{g}_m$$

由一维线搜索得步长 ρ_m 有下式决定:

$$\rho_m = \arg\min_{\rho} \Phi(\mathbf{P}_{m-1} - \rho \mathbf{g}_m)$$

2.3 函数空间上的数值优化

假定某函数由多个基函数相加而成，即加法模型:

$$F^*(x) = \sum_{m=0}^{M} f_m(x)$$

有如下优化问题:

$$\min \Phi(F(x)) = E_y[L(y, F(x))|x]$$

$F_m(x)$ 是迭代的类似于 $x_k = x_{k-1} + \alpha_k d_k$ 则由最速下降法得:

$$f_m(\mathbf{x}) = -\rho_m \mathbf{g}_m(\mathbf{x})$$

$$\mathbf{g}_m(\mathbf{x}) = E_y[\frac{\partial L(y, F(\mathbf{x}))}{\partial F(\mathbf{x})}|\mathbf{x}]_{\mathbf{F}(x)=\mathbf{F}_{m-1}(\mathbf{x})}$$

步长 ρ_m 由下列精确一维线搜索决定:

$$\rho_m = \arg\min_{\rho} E_{y,\mathbf{x}} L(y, F_{m-1}(\mathbf{x}) - \rho \mathbf{g}_m(\mathbf{x}))$$

2.4 有限维数据——梯度提升算法

从参数的角度而言，已知样本数据 $\{y_i, \mathbf{x}_i\}$，模型为加法模型，则有如下参数优化问题:

$$\{\beta_m^*, \mathbf{a}_m^*\}_1^M = \arg\min_{\{\beta_m, \mathbf{a}_m\}_1^M} \sum_{i=1}^{N} L(y_i, \sum_{m=1}^{M} \beta_m h(\mathbf{x}_i; \mathbf{a}_m)) \tag{1}$$

要直接求解上述参数优化问题将会比较棘手，因为需要在一个优化问题中求解 $M \cdot (1 + \dim(\mathbf{a}_m))$ 个参数，参数个数庞大，我们考虑使用 Section **??** 中的前向分布算法，其将一个参数优化问题转换为一个函数优化问题。利用前向分布算法后，此时需要解决函数优化问题，即对该优化问题的每一步有:

$$F^* = \arg\min_F \sum_{i=1}^{N} L(y_i, F(\mathbf{x}_i))$$

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \beta_m h(\mathbf{x}; \mathbf{a}_m)$$

由最速下降法得:

$$h(\mathbf{x}_i; \mathbf{a}_m) = -\mathbf{g}_m(\mathbf{x}_i) = -[\frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}$$

但是 $h(\mathbf{x}; \mathbf{a}_m)$ 是一个在 $\mathcal{D}_\mathbf{x}$ 上均有定义的函数，而 $\mathbf{g}_m(\mathbf{x}_i)$ 仅在 $\{\mathbf{x}_1, \dots, x_N\}$ 处有定义，故上式并不能直接将 $h(\mathbf{x}; \mathbf{a}_m)$ 直接确定下来，需要采用其他方式让 $h(\mathbf{x}; \mathbf{a}_m)$ 去贴合负梯度方向。一种想法是在 $h(\mathbf{x}; \mathbf{a}_m)$ 函数族中找到一个 $h^*(\mathbf{x}; \mathbf{a}_m)$ 使得 $h_m = \{h(x_i; a_m)\}_1^N$ 与 $-g_m$ 最平行，则有如下优化问题:

$$\mathbf{a}_m = \arg\min_a \sum_{i=1}^{N} [-\mathbf{g}_m(\mathbf{x}_i) - \alpha h(\mathbf{x}_i; \mathbf{a})]^2$$

寻找负梯度方向说明

- 此处的思想是回归的思想，但是没有截距项因为我们要的就是尽量平行。两向量平行的数学定义是 $-\mathbf{g}_m = \alpha \mathbf{h}_m$
- α 也是该优化问题所要求解的

一旦确定了 $h(\mathbf{x}; \mathbf{a}_m)$ 后，我们便可利用精确一维线搜索去求解步长 β_m:

$$\beta_m = \arg\min_\beta \sum_{i=1}^{N} L(y_i, F_{m-1}(\mathbf{x}_i + \beta h(\mathbf{x}_i; \mathbf{a}_m)))$$

确定了 $\beta_m, \mathbf{a}_m$ 后便可得到:

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \beta_m h(\mathbf{x}; \mathbf{a}_m)$$

在现实数据中并不是直接去找 Equation **??** 下的 $h(\mathbf{x}; \mathbf{a}_m)$ ，而是将 $h(\mathbf{x}; \mathbf{a}_m)$ 去拟合伪响应 $\{\tilde{y}_i = -\mathbf{g}_m(\mathbf{x}_i)\}_i^N$, 这使得函数优化问题转化为了最小二乘函数优化问题。

算法 1 梯度提升算法

初始化模型 $F_0(\mathbf{x}) = \arg\min_\rho \sum_{i=1}^N L(y_i,\rho)$
for $m=1$ 到 M **do**
    计算伪残差：$\tilde{y}_i = -\left[\frac{\partial L(y_i,F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F=F_{m-1}}$
    拟合一个基学习器 $h_m(\mathbf{x};\mathbf{a}_m)$ 来拟合伪残差 $\tilde{y}_i$: $\mathbf{a}_m = \arg\min_{\mathbf{a}} \sum_{i=1}^N[-\mathbf{g}_m(\mathbf{x}_i) - \alpha h(\mathbf{x}_i;\mathbf{a})]^2$
    计算最佳步长：$\beta_m = \arg\min_\beta \sum_{i=1}^N L(y_i, F_{m-1}(\mathbf{x}_i + \beta h(\mathbf{x}_i;\mathbf{a}_m)))$
    更新模型：$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \beta_m h_m(\mathbf{x};\mathbf{a}_m)$
end for
return 最终模型 $F_M(\mathbf{x})$

3 GBDT 算法

GBDT 算法如下：

- 要拟合的模型是: $f(\mathbf{x}) = f_0(\mathbf{x}) + \sum_{m=1}^M \beta_m \sum_{j=1}^J \Upsilon_{jm} I(\mathbf{x}\in R_{im})$ $\mathbf{x} = (x_1, x_2, \dots, x_p)^T$
- 数据结构：即有 n 个样本，p 个特征 $X = \begin{pmatrix}\mathbf{x}_1\\ \vdots \\ \mathbf{x}_n\end{pmatrix}$

1. 初始化学习器该弱学习器不具有任何参数即 $f_0(\mathbf{x}_i) = c$ 则 $f_0(\mathbf{x}) = \arg\min_c \sum_{i=1}^n L(y_i, c)$ ，一般情况下可以取平方损失即 $L(y_i,c) = \frac{1}{2}(y_i - c)^2$ 求导可得 $\frac{\partial \sum_{i=1}^N (\frac{1}{2}(y_i-c)^2)}{\partial c} = -\sum_{i=1}^N (y_i - c) = 0 \Rightarrow c = \frac{1}{N}\sum_{i=1}^N y_i$，即在损失为平方损失的情况下，初始化的弱学习器将会使得各样本的初始值为它们的均值。
2. 构建后续学习器对 $m = 1, 2, \dots, M$ ，即构建 M 个决策树，假设取平方损失，则对于第 m 步而言损失函数有如下形式：$Loss = \sum_{i=1}^n \frac{1}{2}(y_i - f(\mathbf{x}_i))^2 = L(\mathbf{y}, f(\mathbf{x}))|_{f(\mathbf{x})=f_m(\mathbf{x})} = \frac{1}{2}||\mathbf{y} - F_m(X)||_2^2$ $F_m(X) = \begin{pmatrix} f_m(\mathbf{x}_1)\\ \vdots \\ f_m(\mathbf{x}_n)\end{pmatrix}, B(\mathbf{x},\gamma_m) = \begin{pmatrix} b(\mathbf{x}_1,\gamma_m)\\ \vdots \\ b(\mathbf{x}_n,\gamma_m)\end{pmatrix}, \mathbf{y} = \begin{pmatrix} y_1\\ \vdots \\ y_n\end{pmatrix}$

   第 m 步的模型为: $f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \sum_{j=1}^J \Upsilon_{jm} I(\mathbf{x}\in R_{jm}) = f_{m-1}(\mathbf{x}) + b(\mathbf{x},\gamma_m)$

   由最速下降法得第 m 步的迭代公式为（以 $F_m(X)$ 为整体）：$F_m(X) = F_{m-1}(X) - \alpha_{m-1} g_{m-1}, g_{m-1} = \nabla_{F_{m-1}(X)} L(\mathbf{y}, F(X))$ 令 $\alpha_k = \mathbf{1}_n, k = 1, 2, \dots, m-1$, 则有当 $F_m(X) = F_{m-1}(X) - \frac{\partial L(\mathbf{y}, F(X))}{\partial F(X)}|_{F(X)=F_{m-1}(X)}$

   而由 $f_m(\mathbf{x})$ 的形式可得 $B(\mathbf{x},\gamma_m) = -\frac{\partial L(\mathbf{y},F(X))}{\partial F(X)}|_{F(X)=F_{m-1}(X)}$ 由 Matrix cookbook 书得有：$\frac{\partial \mathbf{x}^T\mathbf{a}}{\partial \mathbf{x}} = \frac{\partial \mathbf{a}^T\mathbf{x}}{\partial \mathbf{x}} = \mathbf{a}$ 和 $\frac{\partial \mathbf{b}^T X^T X \mathbf{c}}{\partial X} = X(\mathbf{b}\mathbf{c}^T + \mathbf{c}\mathbf{b}^T)$ $-\frac{\partial L(\mathbf{y},F(X))}{\partial F(X)}|_{F(X)=F_{m-1}(X)} = -\frac{1}{2}[-2\mathbf{y} + 2F_{m-1}(X)] = \mathbf{y} - F_{m-1}(X)$

也即 $b(\mathbf{x}_i, \gamma_m) = r_{im} = y_i - f_{m-1}(\mathbf{x})$，即第 m 颗决策树需要拟合的是上一颗决策树构建后所得的残差。

3. 用 $(\mathbf{x}_i, r_{im})_{i=1}^n$ 建立第 m 颗树后，需要利用一维线搜索解决如下一维优化问题以计算 β_m：$\beta_m = \arg\min_\beta \sum_{i=1}^N L(y_i, F_{m-1}(\mathbf{x}_i + \beta h(\mathbf{x}_i; \mathbf{a}_m)))$

最终的学习器即为：$f(\mathbf{x}) = f_0(\mathbf{x}) + \sum_{m=1}^M \sum_{j=1}^J \Upsilon_{jm} I(\mathbf{x} \in R_{jm})$