Regression Tree

Regression Tree算法核心思想

CART树(Classification And Regression Trees)：包括分类树和回归树。假设决策树是二叉树，内部结点特征取值为**是和否**，右分支是取值为**是**的分支，左分支是取值为**否**的分支。递归地二分每个特征，将特征空间划分为有限个单元，并在这些单元上确定预测的概率分布。

构建Regression Tree的过程

1. 考虑数据集D上的所有特征j，遍历每一个特征下所有可能的取值或者切分点s，将数据集D划分为$D_1(j,s)=\{x|x_j<s\}$和$D_2(j,s)=\{x|x_j\geq s\}$。
2. 分别计算D_1和D_2的平方误差和，选择最小的平方误差对应的特征和分割点，生成两个子结点。$RSS=\sum\limits_{x_i\in D_1(j,s)}(y_i-\overline{y}_{D_1})^2+\sum\limits_{x_i\in D_2(j,s)}(y_i-\overline{y}_{D_2})^2$，$\overline{y}_{D_1}=average(y_i|x_i\in D_1(j,s))$，$\overline{y}_{D_2}$同理。
3. 对上述得到的两个子结点递归调用步骤1步骤2，直到满足条件。

> 回归树是将输入空间划分为M个单元，每个区域输出该区域内的所有点y值的平均值：
>
> $$f(x)=\sum_{m=1}^{M}\overline{y}_m\cdot I(x\in D_m)$$

过拟合与正则化

过拟合处理

1. 约束控制树的过度生长
    - 限制树的深度
    - 分类误差法：当树继续生长无法得到**客观的分类误差减小**，停止生长
    - 叶子结点最小数据量限制：一个叶子结点数据量过小，停止生长
2. 剪枝
    - 后剪枝算法：REP、PEP、CCP、EBP等

正则化

> $\sum_{m=1}^{|T|}\sum_{x_i\in D_m}(y_i-\overline{y}_{D_m})^2+\alpha|T|$，这里的$\alpha$就是超参数，$|T|$是树的结点数。
>
> 最佳的树就是使得上述式子最小的树。

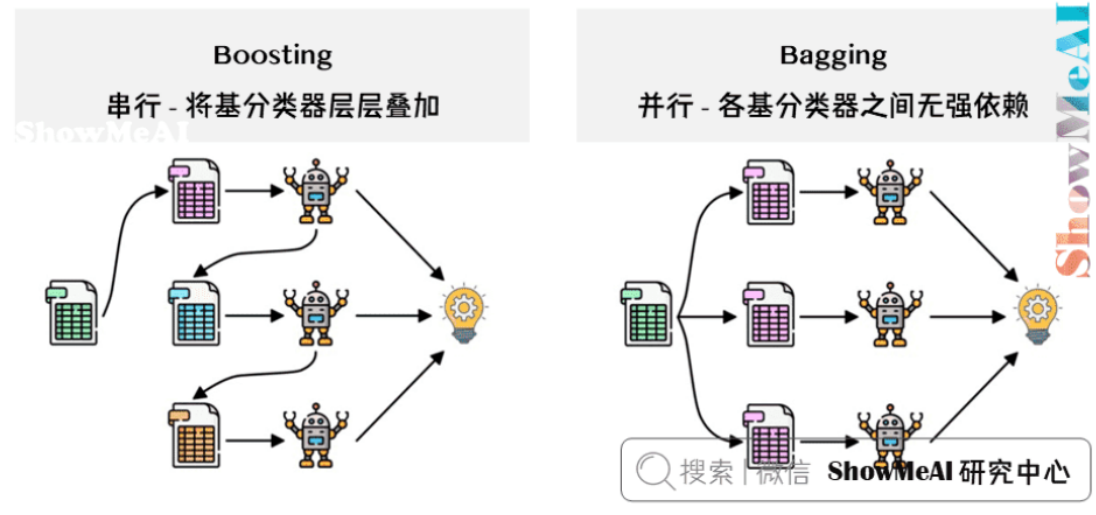
Gradient Boosting Decision Tree

GBDT算法核心

梯度提升决策树(GBDT)：通过构造一组弱的学习器，并把多棵决策树的结果累加起来作为最终的预测输出。

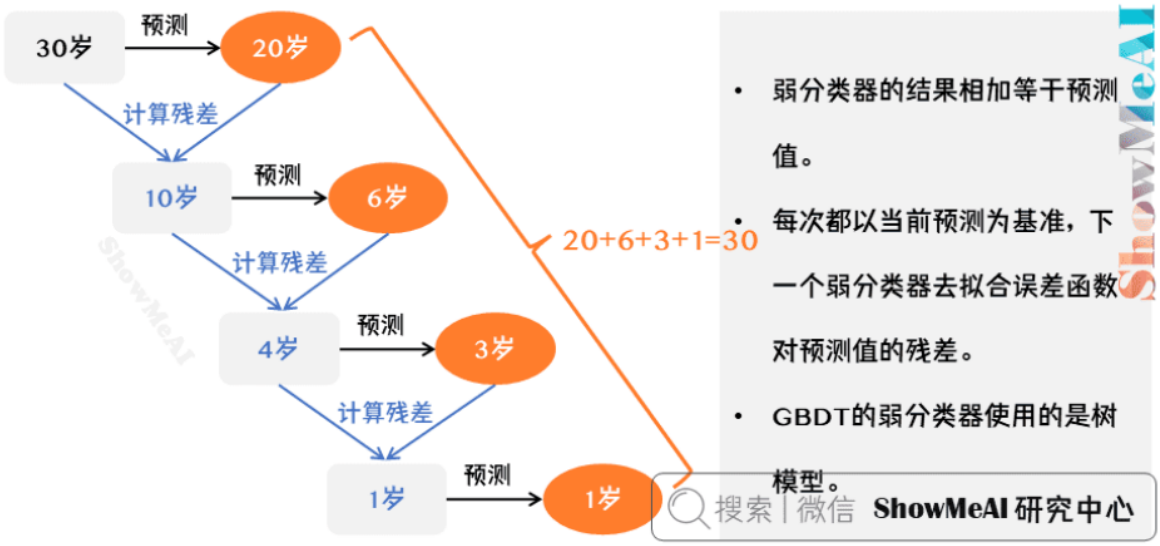
Boosting

采用串行方式训练基分类器，各个分类器之间有依赖。每一层在训练时对前一层基分类器分错的样本给予更高的权重；测试时根据各层分类器的结果加权得到最终结果。

> 与Bagging的区别：
>
> 
>

GBDT原理

1. 所有弱分类器结果相加等于预测值
2. 每次以当前预测为基准，下一个弱分类器去拟合误差函数对预测值的残差



> 回归任务中，GBDT在每一轮迭代对每个样本都有一个预测值，误差函数：$l(y_i,\hat{y}_i)=\frac{1}{2}(y_i-\hat{y}_i)^2$，负梯度为$-[\frac{\partial l(y_i,\hat{y}_i)}{\partial \hat{y}_i}]=(y_i-\hat{y}_i)$，刚好是我们要预测的目标值减去预测值的结果

梯度提升与梯度下降

> 两者都是在每轮迭代中，利用损失函数相对于模型的负梯度方向的信息来对当前模型进行更新

梯度提升(不等同于梯度上升)

在 函数空间F(学习函数f，包括函数的结构和其中的权重) ，不需要进行参数化表示。$F=F_{t-1}-\rho_t\nabla_F\mathbf{L}|_{F=F_{t-1}}$，$\mathbf{L}=\sum_i l(y_i,F(x_i))$

> 计算梯度的目的是为了下一轮的迭代

梯度下降

在 参数空间W(学习模型中的权重) ，$w=w_{t-1}-\rho_t\nabla_w\mathbf{L}|_{w=w_{t-1}}$，$\mathbf{L}=\sum_i l(y_i,f_w(x_i))$

> 计算梯度的目的是为了调整参数

eXtreme Gradient Boosting

XGBoost算法核心思想

Boosted Tree

模型：假设有K棵树，$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), f_k \in F$，$F$包含所有回归树的函数空间。

目标函数：$Obj = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$，第一项是成本函数，第二项是正则化项(代表树的复杂程度，树越复杂正则化项的值越高)

启发式：

- 通过信息增益来做分裂→训练损失
- 修剪树枝→对结点正则化
- 树最大深度→函数空间的约束
- 平滑叶片值→L2正则化对叶片的权重

学习过程

> 初始化一个预测值，每次迭代添加一个新函数(f)：
>
> - $\hat{y}_i^{(0)} = 0$
> - $\hat{y}_i^{(1)} = \hat{y}_i^{(0)} + f_1(x_i)$
> - $\hat{y}_i^{(2)} = \hat{y}_i^{(1)} + f_2(x_i)$
> - $\cdots$
> - $\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i)$
>
> 其中$\hat{y}_i^{(t)}$是第t次迭代的预测值，$f_t(x_i)$是第t棵树

目标函数变换

目的就是找到f_t使得目标函数最小：

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

l为平方误差并引入泰勒展开式化简后为：

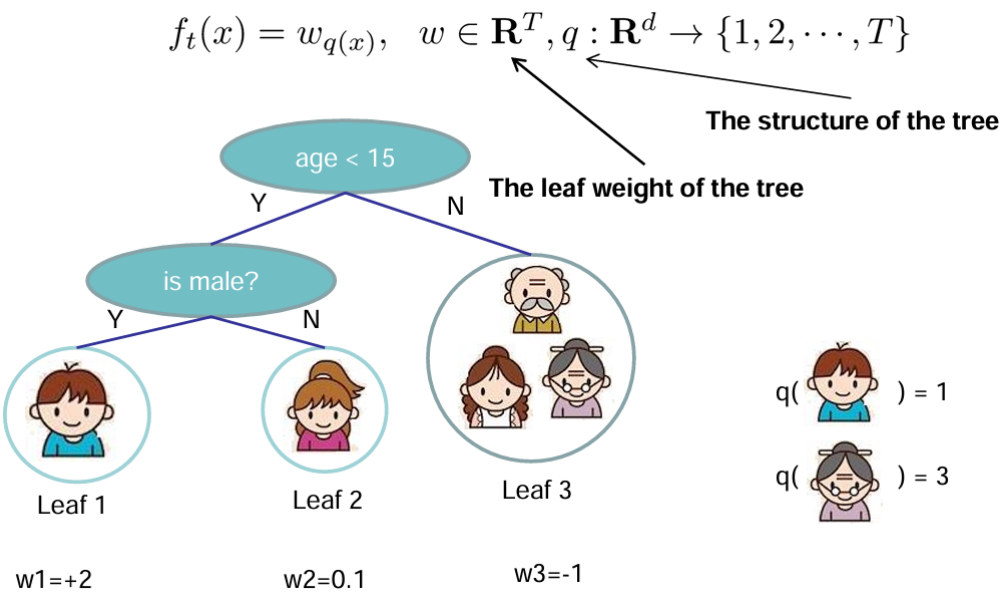
$Obj^{(t)} = \sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t(x_i)^2] + \Omega(f_t) + constant$，$g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$，

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

重新定义树

> 通过叶子结点中的分数向量和实例映射到叶子结点的索引映射函数来定义树：

$f_t(x)$代表一棵树，$f_t(x) = w_q(x), w \in R^T, q : R^d \to \{1, 2, 3, \cdots, T\}$，其中$w$代表树中叶子结点的权重，$q$代表树的结构。



定义树的复杂度(正则化项)

$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$，其中$\lambda T$代表了叶子结点的个数，$\mathbf{w}$为所有叶结点输出回归值构成的向量，$\sum_{j=1}^{T} w_j^2$代表叶子结点分数的**L2 Norm**

重写目标函数(数学转换处理)

在叶子结点j中的实例集合为：$I_j = \{i|q(x_i) = j\}$，$q(x_i)$为将样本映射到叶结点上的索引函数

目标函数：

$$Obj^{(t)} = \sum_{i=1}^{n}[g_i f_t(x_i) + \frac{1}{2}h_i f_t(x_i)^2] + \Omega(f_t) + constant = \sum_{j=1}^{T}[(\sum_{i \in I_j} g_i)w_j + \frac{1}{2}(\sum_{i \in I_j} h_i + \lambda)w_j^2]$$

$$+\gamma T + constant$$

计算叶子结点的值

定义$G_j = \sum_{i \in I_j} g_i$，$H_j = \sum_{i \in I_j} h_i$则有$Obj^{(t)} = \sum_{j=1}^{T}[G_j w_j + \frac{1}{2}(H_j + \lambda)w_j^2] + \gamma T + constant$

> 假设树的结构$q(x)$是固定的

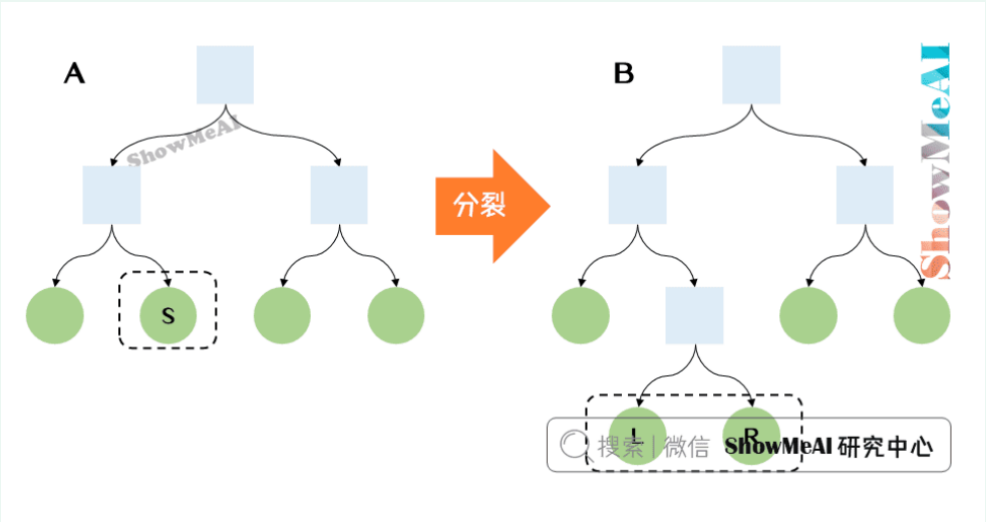
由二次函数极值：

那么每个叶子结点的权重最优值为$w_j^\star = -\frac{G_j}{H_j+\lambda}$，目标函数的最优值为

$Obj = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j+\lambda} + \gamma T$

贪婪算法生成树

1. 生成一个深度为0的树
2. 对于每棵树的每个叶子结点，尝试做分裂，增加分裂之后的目标函数前后变化为：$Gain = \frac{1}{2}(\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}) - \gamma$

> 其中$\frac{G_L^2}{H_L+\lambda}$是左叶子的值，$\frac{G_R^2}{H_R+\lambda}$是右叶子的值，$\frac{(G_L+G_R)^2}{H_L+H_R+\lambda}$是未分裂前的值，$\gamma$是引入多一个叶子结点增加的复杂度
>
> 
>

分裂一个结点的方法：

1. 对每个结点枚举所有特性
2. 对每个特性按特征值排序
3. 使用从左到右线性扫描来决定该特征的最佳分裂点

分类型变量的处理

采用one-hot的方式处理：$z_j = \begin{cases} 1 & \text{if x is in category j} \\ 0 & \text{otherwise} \end{cases}$

修剪和正则化

1. 前停止(Pre-stopping)：最佳分裂是负数时，停止分裂
2. 后剪枝(Post-pruning)：把一颗树生长到最大深度，递归修剪所有分裂为负增益的叶子

列采样和学习率

1. 列采样：与随机森林做法类似，每次结点分裂并不是用全部特征作为候选集，而是一个子集
2. 学习率：步长，在每个子模型前(每个叶结点的回归值上)乘上该系数