XGBoost

参考：b站：贪心学院 讲得超级清楚！

Bagging VS Boosting

- Bagging 是一种并行集成方法，通过在原始数据集上进行多次有放回的抽样生成多个子数据集，训练多个基模型，最终通过投票或平均的方法结合这些基模型的预测结果。
  - 并行训练多个基模型，每个模型独立工作。
  - 每个基模型使用的数据集是通过有放回抽样得到的不同子集。
  - 使用多个弱分类器是为了防止**过拟合**。（可以把每个分类器看作一个专家）
- Boosting 是一种序列集成方法，通过逐步训练基模型，每个基模型试图纠正其前一个模型的错误。最终，通过加权投票或加权平均的方法结合这些基模型的预测结果。
  - 串行训练多个基模型，每个模型依赖于前一个模型的结果。
  - 每个基模型使用的数据集是调整过权重的同一数据集。
  - 使用多个弱分类器是为了防止**欠拟合**。（可以把每个分类器看作一个小学生）

提升树——基于残差的训练

最终预测 = 模型1的预测 + 模型2的预测 + 模型3的预测 + ...

年龄	工作年限	收入（K）
20	2	10
22	3	13
25	6	15
24	2	13
28	3	18
23	2	12
25	5	16

模型1		模型2		模型3		最终预测
9		1		0		10
11		3		-0.5		13.5
10	+	4	+	0.5	=	14.5
11		3		0		14
12		5		2		19
12		1		0		13
18		1		-2		17

Xgboost的原理

1、如何构造目标函数?

2、目标函数直接优化很难，如何近似？用泰勒级数

3、如何把树的结构引入到目标函数?

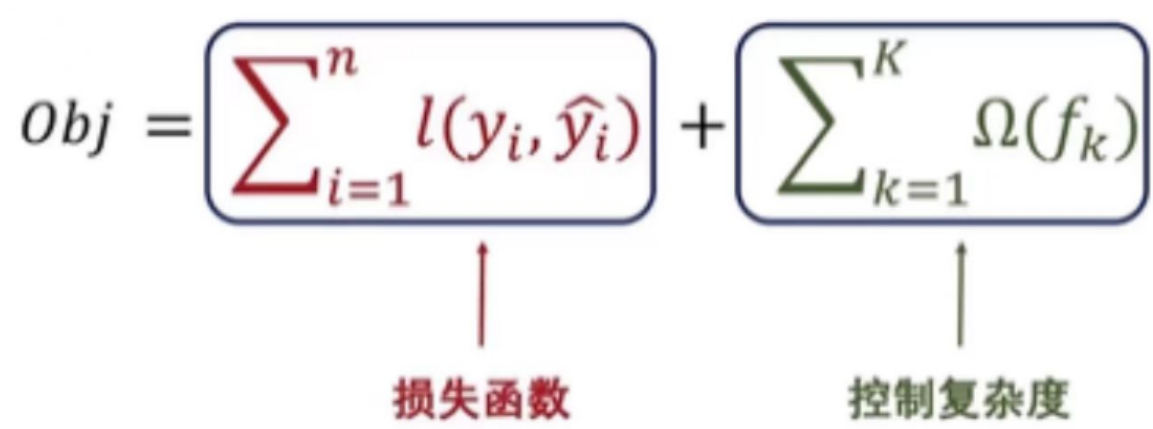
4、仍然难以优化，要不要使用贪心算法?

1、如何构造目标函数?

假设已经训练了k棵树，那么对于第i个样本最终的预测值为：

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

目标函数：损失函数+惩罚项



> 训练决策树时如何控制树的复杂度?
>
> 通过调参的方式控制:
>
> -叶节点个数
>
> -树的深度
>
> -叶节点的值
>
> -.....

Additive Training （叠加式训练）的方式训练第k棵树

2、使用泰勒级数近似目标函数

在 XGBoost 中，目标函数的优化是通过泰勒级数展开来实现的。这种方法使得 XGBoost 能够高效地优化目标函数并处理大规模数据。具体来说，XGBoost 使用了目标函数的二阶泰勒展开（包括一阶和二阶导数）来近似目标函数，从而进行加速计算和优化。

泰勒级数展开

假设损失函数为 L（可以是交叉熵损失、MSE等），对于每一个样本 i，定义预测值为 $\hat{y}_i$。在第 k 轮迭代中，有:

$\hat{y}_i^{(k)} = \hat{y}_i^{(k-1)} + f_k(x_i)$

其中，$f_k(x_i)$ 是第 k 棵树的输出。

目标函数 Obj 对于所有样本的损失可以表示为:

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

这里，$\Omega(f_k)$是正则化项，用于控制模型复杂度。

为了优化这个目标函数，使用二阶泰勒展开，将其展开到二阶导数项。具体地，对于每个样本 i，损失函数 $L(y_i, \hat{y}_i^{(k)})$ 在 $\hat{y}_i^{(k-1)}$处的二阶泰勒展开为:

$L(y_i, \hat{y}_i^{(k)}) \approx L(y_i, \hat{y}_i^{(k-1)}) + g_i f_k(x_i) + \frac{1}{2} h_i f_k(x_i)^2$

其中:

- $g_i = \frac{\partial L(y_i, \hat{y}_i)}{\partial \hat{y}_i}\Big|_{\hat{y}_i = \hat{y}_i^{(k-1)}}$是一阶导数。
- $h_i = \frac{\partial^2 L(y_i, \hat{y}_i)}{\partial \hat{y}_i^2}\Big|_{\hat{y}_i = \hat{y}_i^{(k-1)}}$ 是二阶导数。

优化目标函数

将二阶泰勒展开代入原始的目标函数中，得到：

$$\text{Obj} \approx \sum_{i=1}^{n} \left[L(y_i, \hat{y}_i^{(k-1)}) + g_i f_k(x_i) + \frac{1}{2} h_i f_k(x_i)^2 \right] + \Omega(f_k)$$

由于 $L(y_i, \hat{y}_i^{(t-1)})$是常数项，不影响优化过程，可以将其去掉。于是目标函数变为：

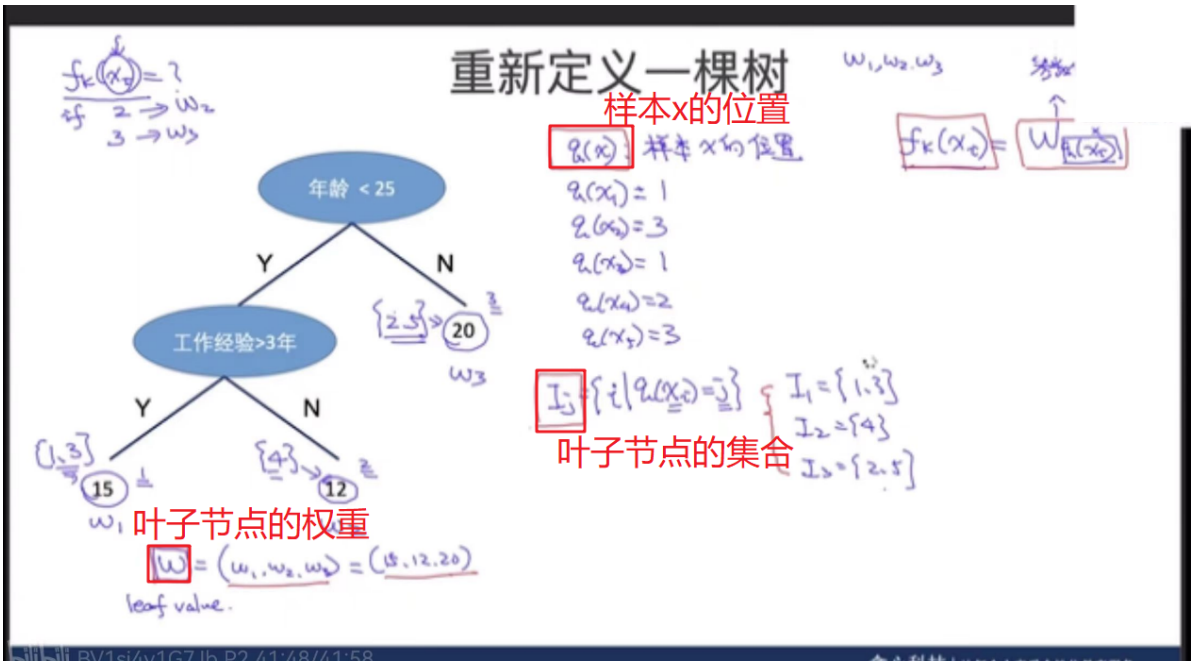
$$\text{Obj} \approx \sum_{i=1}^{n} \left[g_i f_k(x_i) + \frac{1}{2} h_i f_k(x_i)^2 \right] + \Omega(f_k)$$

接下来要考虑的问题就是如何把$f_k(x_i)$ 、$\Omega(f_k)$参数化。

3、如何把树的结构引入到目标函数?

如何去构造一棵树？我们先不考虑这个问题

先假定我们已经有一棵树了，那我们怎么用参数的方式把它表示出来?



我们希望将 $f_k(x)$ 用每个叶节点的权重 w_j 表示出来。假设树结构已经确定，样本 x_i 所属的叶节点为 $j(i)$。因此，模型输出可以表示为：
$f_k(x_i) = w_{j(i)}$

树的复杂度 = 叶节点个数 + 叶节点权重

XGBoost 的复杂度项（正则化项）通常定义为：

$$\Omega(f_k) = \gamma T + \frac{1}{2} \lambda \sum_{j=1}^{T} w_j^2$$

其中：

- T 是叶节点的数量。
- w_j 是第 j个叶节点的权重。
- γ和 λ是超参数，用于控制正则化强度。

新的目标函数

最终，目标函数可以写成：

$$\begin{aligned} \text{Obj} &= \sum_{i=1}^{n} \left[g_i f_k(x_i) + \frac{1}{2} h_i f_k(x_i)^2 \right] + \gamma T + \frac{1}{2} \lambda \sum_{j=1}^{T} w_j^2 \\ &= \sum_{i=1}^{n} \left[g_i w_{j(i)} + \frac{1}{2} h_i w_{j(i)}^2 \right] + \gamma T + \frac{1}{2} \lambda \sum_{j=1}^{T} w_j^2 \end{aligned}$$

为了进一步简化和优化这个目标函数，我们对其进行分组。由于每个叶节点 j 上的权重 w_j 只影响那些落在该叶节点的样本，我们可以将样本按照叶节点进行分组。设 I_j 表示落在叶节点 j的样本集合，则目标函数可以重写为：

$$\text{Obj} = \sum_{j=1}^{T} \left[\sum_{i \in I_j} \left(g_i w_j + \frac{1}{2} h_i w_j^2 \right) \right] + \gamma T + \frac{1}{2} \lambda \sum_{j=1}^{T} w_j^2$$
$$\text{Obj} = \sum_{j=1}^{T} \left[G_j w_j + \frac{1}{2} H_j w_j^2 \right] + \gamma T + \frac{1}{2} \lambda \sum_{j=1}^{T} w_j^2$$

其中：

- $G_j = \sum_{i \in I_j} g_i$是叶节点 j 上所有样本的一阶导数之和。
- $H_j = \sum_{i \in I_j} h_i$是叶节点 j 上所有样本的二阶导数之和。

将这部分目标函数合并，我们得到：
$$\text{Obj} = \sum_{j=1}^{T} \left[G_j w_j + \frac{1}{2} (H_j + \lambda) w_j^2 \right] + \gamma T$$

求解叶节点权重

我们对每个叶节点 j 的权重 w_j 进行优化，使得目标函数最小化。对于每个 j，优化目标是：
$$\text{Obj}_j = G_j w_j + \frac{1}{2} (H_j + \lambda) w_j^2$$

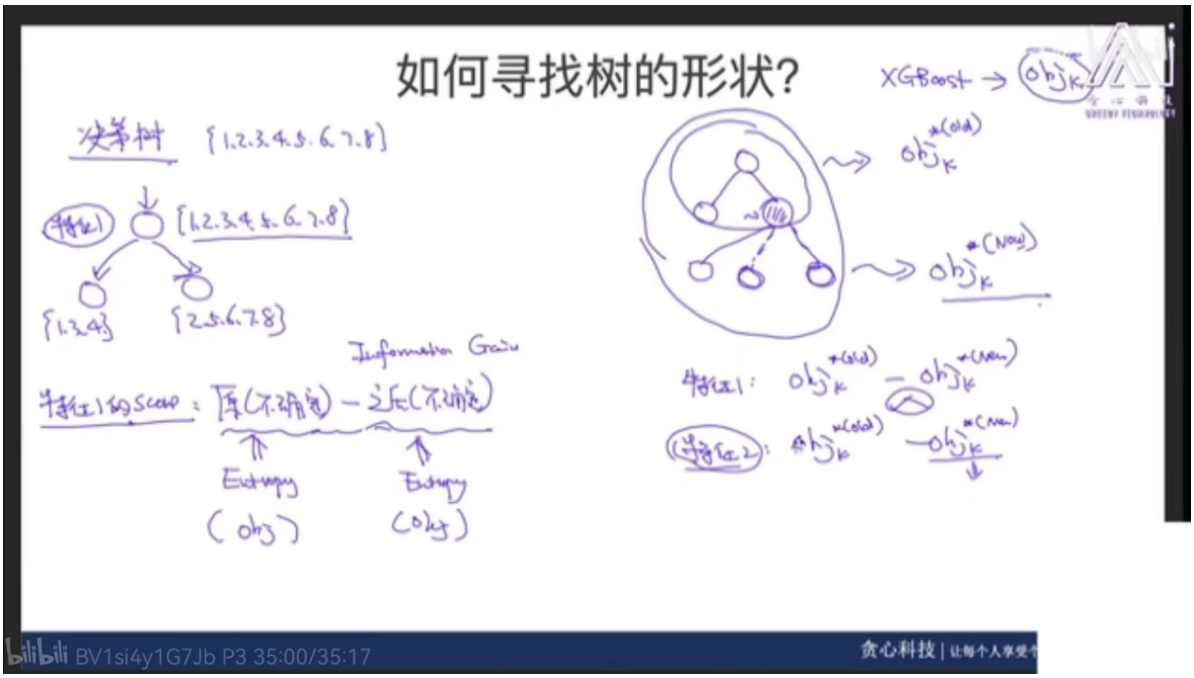
对 w_j 求导并设导数为零，得到最优权重 w_j 的闭式解：
$$\frac{\partial \text{Obj}_j}{\partial w_j} = G_j + (H_j + \lambda) w_j = 0$$

解得：
$$w_j^* = -\frac{G_j}{H_j + \lambda}$$

将最优权重 w_j^*代入目标函数，得到最终的最小化的目标函数值：
$$\text{Obj}_{\min} = -\frac{1}{2} \sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma T$$

4、如何寻找树的形状？贪心

和决策树的构造类似，只不过把熵换成了Obj



具体实现步骤

1. **计算一阶和二阶导数**：对于每个样本，计算损失函数的一阶导数g_i 和二阶导数 h_i （已知）。
2. **构建树结构**：根据导数信息，构建新的树。选择最佳分裂点，分裂节点以最大化**目标函数的增益**。
3. **计算叶节点权重**：对于每个叶节点，计算其最优的权重 w_j，使得目标函数最小。
4. **更新预测值**：将新树的输出添加到当前的预测值中。