

两个常用集成决策树模型原理介绍

[xgboost 原始论文地址]
(<https://arxiv.org/pdf/1603.02754v1.pdf>)
[xgboost 原始 ppt 介绍]
(<https://homes.cs.washington.edu/~tqchen/pdf/BoostedTree.pdf>)
表 5.1 是一个由 15 个样本组成的贷款申请的训练数据。

表 5.1 贷款申请样本数据表

ID	年龄	有工作	有自己的房子	信贷情况	类别
1	青年	否	否	一般	否
2	青年	否	否	好	否
3	青年	是	否	好	是
4	青年	是	是	一般	是
5	青年	否	否	一般	否
6	中年	否	否	一般	否
7	中年	否	否	好	否
8	中年	是	是	好	是
9	中年	否	是	非常好	是
10	中年	否	是	非常好	是
11	老年	否	是	非常好	是
12	老年	否	是	好	是
13	老年	是	否	好	是
14	老年	是	否	非常好	是
15	老年	否	否	一般	否

希望通过所给的训练数据学习一个贷款申请的决策树，用以对未来的贷款申请进行分类，即当新的客户提出贷款申请时，根据申请人的特征利用决策树决定是否批准贷款申请。

CART

分类树的生成

分类树用基尼指数选择最优特征，同时决定该特征的最优二值切分点。

定义 5.4（基尼指数） 分类问题中，假设有 K 个类，样本点属于第 K 类的概率为 p_k ，

则概率分布的基尼指数定义为：

$$Gini(p) = \sum_{k=1}^K p_k(1 - p_k) = 1 - \sum_{k=1}^K p_k^2$$

对于二类分类问题，若样本点属于第 1 个类的概率是 p ，则概率分布的基尼指数为：

$$Gini(p) = 2p(1-p)$$

对于给定的样本集合 D ，其基尼指数为

$$Gini(D) = 1 - \sum_{k=1}^K \left(\frac{|C_k|}{|D|} \right)^2$$

这里， C_k 是 D 中属于第 k 类的样本子集， K 是类的个数。

如果样本集合 D 根据特征 A 是否取某一可能值 a 被分隔成 D_1 和 D_2 两部分，即

$$D_1 = \{(x, y) \in D \mid A(x) = a\}, D_2 = D - D_1$$

则在特征 A 的条件下，集合 D 的基尼指数定义为

$$Gini(D, A) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2)$$

基尼指数 $Gini(D)$ 表示集合 D 的不确定性，基尼指数 $Gini(D, A)$ 表示经 $A = a$ 分割后集合 D 的不确定性。基尼指数值越大，样本集合的不确定性也就越大。

CART 生成算法

输入：训练数据集 D ，停止计算的条件；

输出：CART 决策树。

根据训练数据集，从根结点开始，递归地对每个结点进行以下操作，构建二叉决策树：

(1) 设结点的训练数据集为 D_1 ，计算现有特征对该数据集的基尼指数。此时，对每一个特征 A ，对其可能取的每个值 a ，根据样本点对 $A=a$ 的测试为“是”或“否”将 D 分割成 D_1 和 D_2 两部分，利用式

$$Gini(D, A) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2)$$
 计算 $A=a$ 时的基尼指数。

(2) 在所有可能的特征 A 以及它们所有可能的切分点 a 中，选择基尼指数最小的特征及其对应的切分点作为最优特征与最优切分点。依最优特征与最优切分点，从现结点生成两个子结点，将训练数据集依特征分配到两个子结点中去。

(3) 对两个子结点递归地调用 (1)，(2)，直至满足停止条件。

(4) 生成 CART 决策树。

算法停止计算的条件是结点中的样本个数小于预定阈值，或样本集的基尼指数小于预定阈值（样本基本属于同一类），或者没有更多特征。

根据表 5.1 所给训练数据集，应用 CART 算法生成决策树。

解 首先计算各特征的基尼指数，选择最优特征以及其最优切分点。分别以 A_1, A_2, A_3, A_4 表示年龄、有工作、有自己的房子和信贷情况 4 个特征，并以 1, 2, 3 表示年龄的值为青年、中年和老年，以 1, 2 表示有工作和有自己的房子的值为是和否，以 1, 2, 3 表示信贷情况的值为非常好、好和一般。

求特征 A_1 的基尼指数：

$$Gini(D, A_1 = 1) = \frac{5}{15} \left(2 \times \frac{2}{5} \times \left(1 - \frac{2}{5} \right) \right) + \frac{10}{15} \left(2 \times \frac{7}{10} \times \left(1 - \frac{7}{10} \right) \right) = 0.44$$

$$Gini(D, A_1 = 2) = 0.48$$

$$Gini(D, A_1 = 3) = 0.44$$

由于 $Gini(D, A_1 = 1)$ 和 $Gini(D, A_1 = 3)$ 相等，且最小，所以 $A_1 = 1$ 和 $A_1 = 3$ 都可以选作 A_1 的最优切分。

求特征 A_2 和 A_3 的基尼指数：

$$Gini(D, A_2 = 2) = 0.32$$

$$Gini(D, A_3 = 1) = 0.27$$

由于 A_2 和 A_3 只有一个切分点，所以它们就是最优切分点。

求特征 A_4 的基尼指数：

$$Gini(D, A_4 = 1) = 0.36$$

$$Gini(D, A_4 = 2) = 0.47$$

$$Gini(D, A_4 = 3) = 0.32$$

$Gini(D, A_4 = 3)$ 最小，所以 $A_4 = 3$ 为 A_4 的最优切分点。

在 A_1, A_2, A_3, A_4 几个特征中, $Gini(D, A_3=1)=0.27$ 最小, 所以特征选择 A_3 为最优特征, $A_3=1$ 为其最优的切分点。于是根节点生成两个子结点, 一个是叶结点。对另一个结点继续使用以上方法在 A_1, A_2, A_4 中选择最优特征及其选择最优切分点, 结果是 $A_2=1$, 依此计算得知, 所得结点都是叶结点。

最小二乘回归树生成算法

输入: 训练数据集 D ;

输出: 回归树 $f(x)$ 。

在训练数据集所在的输入空间中, 递归地将每个区域划分为两个子区域并决定每个子区域上的输出值, 构建二叉决策树:

(1) 选择最优切分变量 j 与切分点 s , 求解

$$\min_{j,s} \left[\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2 \right]$$

遍历变量 j , 对固定的切分变量 j 扫描切分点 s , 使用

$$\min_{j,s} \left[\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2 \right] \text{ 达到最小的值对 } (j, s)。$$

(2) 用选定的对 (j, s) 划分区域并决定相应的输出值:

$$R_1(j,s) = \{x | x^{(j)} \leq s\}, R_2(j,s) = \{x | x^{(j)} > s\}$$

$$\hat{c}_m = \frac{1}{N_m} \sum_{x \in R_m} y_i, x \in R_m, m=1,2$$

(3) 继续对两个子区域调用步骤 (1), (2), 直至满足停止条件。

(4) 将输入空间划分为 M 个区域 R_1, R_2, \dots, R_m , 生成决策树:

$$f(x) = \sum_{m=1}^M \hat{c}_m I(x \in R_m)$$

回归树的例子:

x	1	2	3	4	5	6	7	8	9	10
y	5.56	5.70	5.91	6.40	6.80	7.05	8.90	8.70	9.00	9.05

当 $s=1.5$ 时

$$R_1 = \{1\}, R_2 = \{2,3,4,5,6,7,8,9,10\}$$

$$C_1 = 5.56,$$

$$C_2 = \frac{1}{9}(5.70+5.91+6.40+6.80+7.05+8.90+8.70+9.00+9.05) = 7.05$$

$$m(1.5) = 0 + 15.72 = 15.72$$

代码块:

```
(5.56-5.56)**2+ (5.70-7.50)**2+ (5.91-7.50)**2+ (6.40-7.50)**2+
(6.80-7.50)**2+ (7.05-7.50)**2+ (8.90-7.50)**2+ (8.70-7.50)**2+
(9.00-7.50)**2+ (9.05-7.50)**2
```

s	1.5	2.5	3.5	4.5	5.5	6.5	7.5	8.5	9.5
C_1	5.56	5.63	5.72	5.89	6.07	6.24	6.62	6.88	7.11
C_2	7.5	7.73	7.99	8.25	8.54	8.91	8.92	9.03	9.05
m(s)	15.72	12.07	8.36	5.78	3.91	1.93	8.01	11.73	15.74

在表中我们可以发现 $s=6.5$ 时, $c_1 = 6.24, c_2 = 8.91, m(s)$ 最小。因此 $j = x, s = 6.5$,

回归树 $f_1(x)$:

$$f_1(x) = \begin{cases} 6.24, & x \leq 6.5 \\ 8.91, & x > 6.5 \end{cases}$$

对 $x \leq 6.5$ 部分进行划分, 回归树 $f_2(x)$:

$$f_1(x) = \begin{cases} 5.72, & x \leq 3.5 \\ 6.75, & 3.5 < x \leq 6.5 \\ 8.91, & x > 6.5 \end{cases}$$

依此类推 $x > 6.5$ 部分, 之后不断重复直到满足条件 (树的深度、树的叶子个数等都可以作为停止条件)。

XGBOOST

回顾监督学习的关键概念：

$x \in R^d$ 训练集

模型：

线性模型： $\hat{y}_i = \sum_j w_j x_{ij}$

线性回归： \hat{y}_i 预测回归的结果

逻辑回归： $\frac{1}{1 + e^{-\hat{y}_i}}$ 预测出正类的概率。

从中我们需要学习到的参数：

$$\Theta = \{w_j \mid j = 1, \dots, d\}$$

随处可见的目标函数：

$$Obj = L(\Theta) + \Omega(\Theta)$$

$L(\Theta)$ 是训练集的损失函数， $\Omega(\Theta)$ 是正则项。

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

交叉熵损失函数： $l(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2$

逻辑回归损失函数： $l(y, \hat{y}_i) = y_i \ln(1 + e^{-\hat{y}_i}) + (1 - y_i) \ln(1 + e^{\hat{y}_i})$

正则项：

L1 正则项： $\Omega(w) = \lambda \|w\|^2$

L2 正则项： $\Omega(w) = \lambda \|w\|_1$

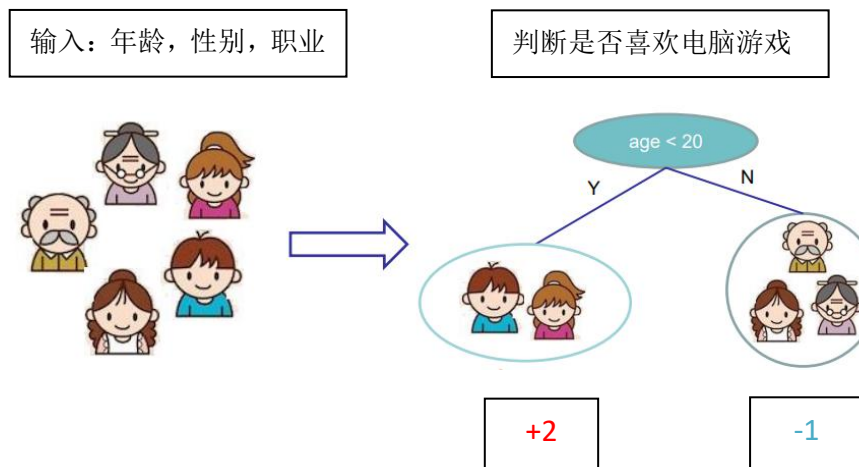
这样交叉熵损失函数为：

$$L2: \sum_{i=1}^n (y_i - w^T x_i)^2 + \lambda \|w\|^2$$

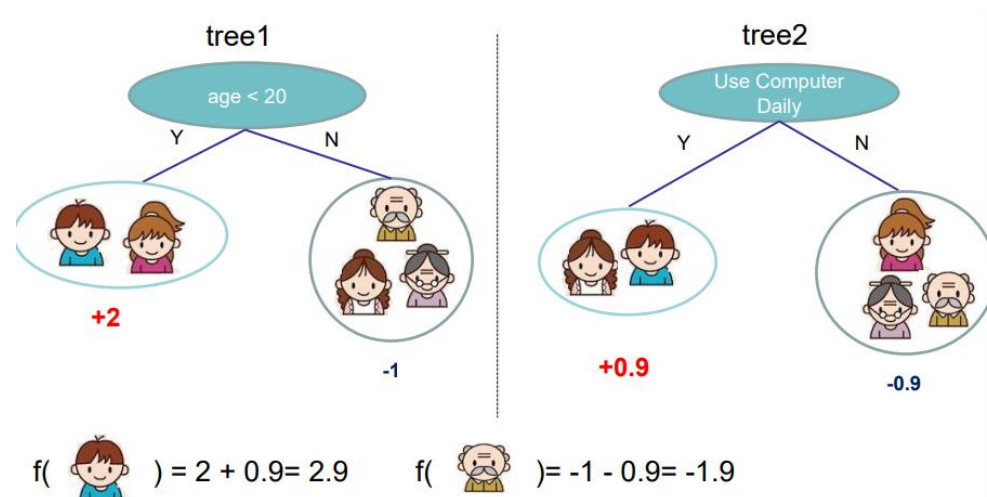
逻辑回归损失函数为：

$$L2: [y_i \ln(1 + e^{-w^T x_i}) + (1 - y_i) \ln(1 + e^{w^T x_i})] + \lambda \|w\|^2$$

CART 分类回归树 (classification and regression tree)。



集层思想：



一个 xgboost ensemble model 使用 K 个累加的函数来预测输出：

这样我们有 K 棵树：

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

F 包含了所有回归树的向量空间。

这样我们就需要求:

$\Theta = \{f_1, f_2, \dots, f_k\}$ 的参数。

目标函数:

$$L = \sum_{i=1}^n l(y_i, \hat{y}_i) + \sum_k \Omega(f_k), f_k \in F$$

$$\text{其中, } \Omega(f) = \gamma T + \frac{1}{2} \lambda \|w\|^2$$

这样, 从常量开始, 我们每次都新增一个函数:

$$\hat{y}_i^{(0)} = 0$$

$$\hat{y}_i^{(1)} = f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i)$$

$$\hat{y}_i^{(1)} = f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i)$$

$$\hat{y}_i^{(t)} = \sum_{k=1}^t f_k(x_i) + f_2(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)$$

如何决定添加哪一个 f 呢, 通过优化目标函数!

在 t 轮的预测为: $\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i)$, 目标函数为:

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

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

考虑使用平方差损失函数:

$$Obj^t = \sum_{i=1}^n (y_i - \hat{y}_i^{(t-1)} + f_t(x_i))^2 + \Omega(f_i) + const$$

$$Obj^t = \sum_{i=1}^n \left[2 \left(\hat{y}_i^{(t-1)} - y_i \right) f_t(x_i) + f_t(x_i)^2 \right] + \Omega(f_i) + const$$

使用泰勒展开式逼近目标函数

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

泰勒展开式：

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

$$\text{定义： } g_i = \partial_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)}), h_i = \partial_{\hat{y}_i^{(t-1)}}^2 l(y_i, \hat{y}_i^{(t-1)})$$

则原目标函数变为：

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

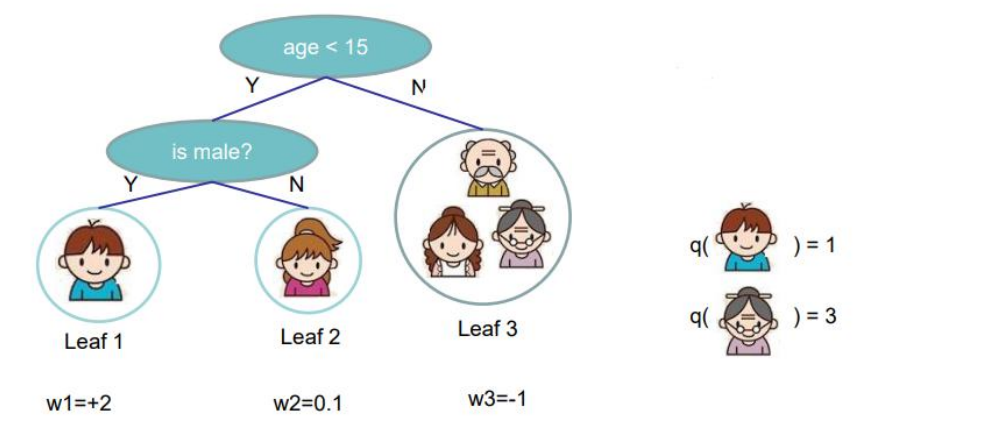
由于上一步的结果也是常数，所以目标函数变为

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

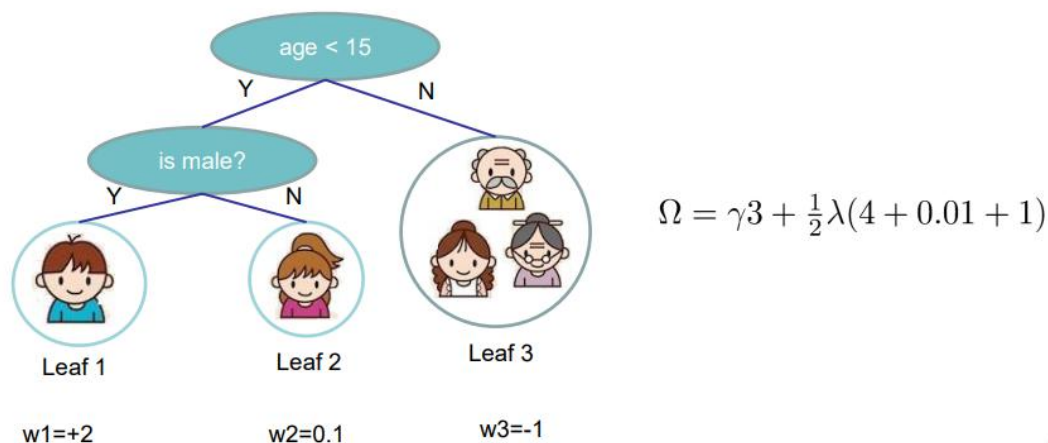
$$\text{其中 } g_i = \partial_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)}), h_i = \partial_{\hat{y}_i^{(t-1)}}^2 l(y_i, \hat{y}_i^{(t-1)})。$$

目标函数经由 g_i , h_i 来优化，所以我们可以将树的生长优化转换为目标函数的优化。

其中， $f_t(x) = w_{q(x)}$, $w \in R^T$, $q: R^d \rightarrow \{1, 2, \dots, T\}$, $w \in R^T$ 叶子的权重, q 为叶子的结构。



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



这样，换成叶子优化，目标函数则为：

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

$$Obj^t \cong \sum_{i=1}^n [g_i w_{q(x_i)} + \frac{1}{2} h_i w_{q(x_i)}^2] + \gamma T + \frac{1}{2} \lambda \sum_{j=1}^T w_j^2$$

$$Obj^t \cong \sum_{j=1}^T [(\sum_{i \in I_j} g_i) w_j + \frac{1}{2} (\sum_{i \in I_j} h_i + \lambda) w_j^2] + \gamma T$$

其中，其中 I 被定义为每个叶子上面样本集合：

$$I_j = \{i \mid q(x_i) = j\}$$

定义: $G_j = \sum_{i \in I_j} g_i$, $H_j = \sum_{i \in I_j} h_i$

G_j : 叶子结点 j 所包含样本的一阶偏导数累加之和, 是一个常量;

H_j : 叶子结点 j 所包含样本的二阶偏导数累加之和, 是一个常量;

则

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






$$Obj^t = \sum_{j=1}^T [G_j w_j + \frac{1}{2} (H_j + \lambda) w_j^2] + \gamma T$$

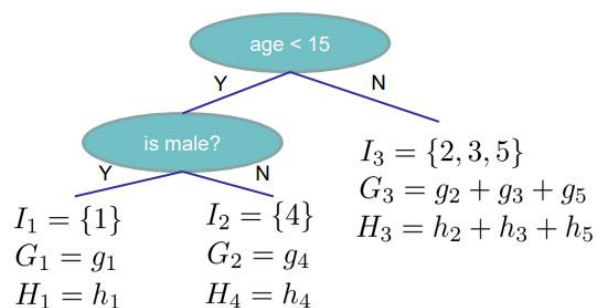
通过对 w_j^* 求导=0, 得

$$w_j^* = \frac{G_j}{H_j + \lambda},$$

将 w_j^* 带入上式中,

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

示例索引	梯度统计
1 	g_1, h_1
2 	g_2, h_2
3 	g_3, h_3
4 	g_4, h_4
5 	g_5, h_5



$$Obj = -\sum_j \frac{G_j^2}{H_j + \lambda} + 3\gamma$$

得分越小, 树结构越好

一棵树的生长:

在实际训练过程中,当建立第 t 棵树时,XGBoost 采用贪心法进行树结点的分裂:

从树深为 0 时开始:

对树中的每个叶子结点尝试进行分裂:

每次分裂后,原来的一个叶子结点继续分裂为左右两个子叶子结点,原叶子结点中的样本集将根据该结点的判断规则分散到左右两个叶子结点中;

新分裂一个结点后,需要检查这次分裂是否会给损失函数带来增益,增益的定义如下:

$$\begin{aligned} Gain &= Obj_{L+R} - (Obj_L + Obj_R) \\ Gain &= \left[-\frac{1}{2} \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} + \gamma \right] - \left[-\frac{1}{2} \left(\frac{(G_L)^2}{H_L + \lambda} + \frac{(G_R)^2}{H_R + \lambda} \right) + 2\gamma \right] \\ Gain &= \frac{1}{2} \left[\frac{(G_L)^2}{H_L + \lambda} + \frac{(G_R)^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} \right] - \gamma \end{aligned}$$

如 $Gain > 0$, 则表示分裂为两个叶子结点, 目标函数下降, 表示可以考虑此次分裂的结果。

而多个分裂点可能会产生一个增益, 需要获得**最优分裂点**:

特征预排序: XGBoost 在训练之前, 会先根据特征值对特征进行预排序, 保存为 **block** 结构, 并在迭代中反复使用, 减少计算量;

分位点近似: 对每个特征按照特征值排序后, 采用类似分位点选取的方式, 仅仅选出常数个特征值作为该特征的候选分割点, 在寻找该特征的最佳分割点时, 从候选分割点中选出最优的一个。

并行查找: 由于各个特性已预先存储为 **block** 结构, XGBoost 支持利用多个线程并行地计算每个特征的最佳分割点, 这不仅大大提升了结点的分裂速度, 也极利于大规模训练集的适应性扩展。

停止生长:

1. 当上式中 $Gain < 0$, 放弃当前分裂
2. 当树达到最大深度, 停止建树, **防止过拟合**
3. 当引入一次分裂后, 重新计算新生成的左、右两个叶子结点的样本权重和。如果任一个叶子结点的样本权重低于某一个阈值, 也会放弃此次分裂。最小样本

权重和，是指如果一个叶子节点包含的样本数量太少也会放弃分裂，防止树分的太细，防止过拟合。

LightGBM-A Highly Efficient Gradient Boosting Decision Tree

- 1) 压缩了数据的数量;
- 2) 压缩了数据的维度;
- 3) 降低训练数据的量。

特点:

特点	备注
Gradient-based One-Side Sampling (GOSS)	保留梯度较大数据：将分裂的特征按绝对值大小降序排序， XGB:保留排序后结果， LGB 不保留取绝对值最大 $a100\%$, 剩余小梯度随机选取 $b100\%$, 且 $(1-a)/b$ 。只使用 $(a+b)\%$ 部分数据计算收益
Exclusive Feature Bundling	特征融合绑定降低特征数量：1) 图着色：每个特征有个图 G 定点，用边连接不相互独立的特征，边权重为两特征总冲突值，如着色一样，变为一个 bundle;2) 对非零值的数量降序排序 (进行步骤 1，判断是否新建 bundle, 特征值中加入偏置常量解决捆绑互斥特征，也就是他们很少同时取非零值
Histogram-based Algorithm	连续变量离散化：连续的特征映射到离散的 buckets 中，组成一个个的 bins
Leaf-wise	深度限制的叶子生长：只需达到设置树的叶子数不加深度生长了 (速度快之一)

