第七章　决策树

张皓

https://haomood.github.io/homepage/

zhangh0214@gmail.com

摘要

本章介绍决策树算法，包括单变量决策树 *(C4.5、ID3、CART)*、多变量决策树、以决策树为集学习器的 *Bagging* 集成 *(*随机森林*)* 和以决策树为集学习器的 *Boosting* 集成 *(GBDT、XGBoost、LightGBM)*.

本系列文章有以下特点：*(a)*. 为了减轻读者的负担并能使尽可能多的读者从中收益，本文试图尽可能少地使用数学知识，只要求读者有基本的微积分、线性代数和概率论基础，并在第一节对关键的数学知识进行回顾和介绍. *(b)*. 本文不省略任何推导步骤，适时补充背景知识，力图使本节内容是自足的，使机器学习的初学者也能理解本文内容. *(c)*. 机器学习近年来发展极其迅速，已成为一个非常广袤的领域. 本文无法涵盖机器学习领域的方方面面，仅就一些关键的机器学习流派的方法进行介绍. *(d)*. 为了帮助读者巩固本文内容，或引导读者扩展相关知识，文中穿插了许多问题，并在最后一节进行问题的“快问快答”.

1　准备知识

1.1　信息论基础

熵 (entropy) 是对信息中不确定性的一种度量. 如果在计算机通信中传输的信息是无损编码的，熵提供了一个理论上信息的最短二进制编码. 信息论中常用术语对应表如表 1 所示，熵、条件熵和互信息之间的关系如图 1 所示.

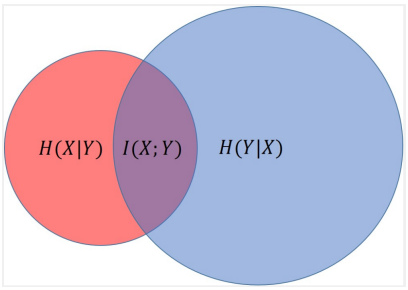

Figure 1: 熵、条件熵和互信息之间的关系. 其中，红色圆圈代表 $H(X)$，蓝色圆圈代表 $H(Y)$. 本图源于 [18] .

引理 1. 若 X 有 n 个取值，$H(X) \in [0, \lg n]$.

Proof. 当存在某个 i 使得 $p(X = x_i) = 1$ 时，$H(X) = 0$. 当 $p(X)$ 是一个离散均匀分布时，$H(X) = \lg n$. □

引理 2. (X, Y) 中包含的信息量是 X 中的信息量以及 Y 中不依赖 X 的那部分 *(*即 Y 中除去了 X 影响的那部分*)* 的信息量之和

$$H(X, Y) = H(X) + H(Y \mid X). \tag{1}$$

Proof.

$$\begin{aligned} -\mathbb{E}[\lg p(X)] - \mathbb{E}[\lg p(Y \mid X)] &= -\mathbb{E}[\lg p(X)p(Y \mid X)] \\ &= -\mathbb{E}[\lg p(X, Y)]. \end{aligned} \tag{2}$$

□

推论 3. 通常条件熵是不对称的 $H(Y \mid X) \neq H(X \mid Y)$. 然而，互信息是对称.

$$I(X, Y) = H(X) - H(X \mid Y) = H(Y) - H(Y \mid X) = I(Y, X). \tag{3}$$

Table 1: 信息论中常用术语对应表.

术语	符号	定义	离散	连续
熵	$H(X)$	$-\mathbb{E}[\lg p(X)]$	$-\sum_x p(x)\lg p(x)$	$-\int_{-\infty}^{\infty} p(x)\lg p(x)\,\mathrm{d}x$
联合熵	$H(X,Y)$	$-\mathbb{E}[\lg p(X,Y)]$	$-\sum_x\sum_y p(x,y)\lg p(x,y)$	$-\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} p(x,y)\lg p(x,y)\,\mathrm{d}x\,\mathrm{d}y$
条件熵	$H(Y\mid X)$	$-\mathbb{E}[\lg p(Y\mid X)]$	$-\sum_x\sum_y p(x,y)\lg p(y\mid x)$	$-\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} p(x,y)\lg p(y\mid x)\,\mathrm{d}x\,\mathrm{d}y$
互信息	$I(X,Y)$	$\mathbb{E}\left[\lg\frac{p(X,Y)}{p(X)p(Y)}\right]$	$\sum_x\sum_y p(x,y)\lg\frac{p(x,y)}{p(x)p(y)}$	$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} p(x,y)\lg\frac{p(x,y)}{p(x)p(y)}\,\mathrm{d}x\,\mathrm{d}y$
KL 散度	$\mathrm{KL}(p\,\|\,q)$	$\mathbb{E}\left[\lg\frac{p(x)}{q(x)}\right]$	$\sum_x p(x)\lg\frac{p(x)}{q(x)}$	$\int_x p(x)\lg\frac{p(x)}{q(x)}\,\mathrm{d}x$

互信息可以看做是 X (或 Y) 中的信息量去除 X 中不依赖 Y (或 Y 中不依赖 X) 的那部分信息剩余的 X 和 Y 共有的信息量.

Proof. 根据引理 2 ,

$$H(X,Y)=H(X)+H(Y\mid X)=H(Y)+H(X\mid Y)\,, \tag{4}$$

调整最后一个等号左右两边即得推论中间等号. 通过

$$\begin{aligned}-\mathbb{E}[\lg p(X)]+\mathbb{E}[\lg p(X\mid Y)]&=\mathbb{E}\left[\lg\frac{p(X\mid Y)}{p(X)}\right]\\&=\mathbb{E}\left[\lg\frac{p(X,Y)}{p(X)p(Y)}\right].\end{aligned} \tag{5}$$

可得推论第一个等号. 推论最后一个等号同理. □

引理 4. *KL 散度 (KL divergence) 度量了两个分布之间的差异. KL 散度不是对称的. KL 散度是非负的, 当且仅当 p 和 q 是同一个分布时 $\mathrm{KL}(p\,\|\,q)=0$.*

推论 5. *互信息是联合分布和两个边缘分布之间的 KL 散度 $I(X,Y)=\mathrm{KL}(p(x,y)\,\|\,p(x)p(y))$, 因此, 互信息也是非负的, 当且仅当 X 和 Y 独立时 $I(X,Y)=0$.*

1.2 决策树的基本结构

决策树的基本结构? 一颗决策树 (decision tree) 包括若干内部结点 (对应于属性测试) 和若干叶结点 (对应于决策结果). 根结点包含训练集全集, 每个内部结点包含的样本集合根据属性测试的结果被划分到子结点中用以进一步的属性测试. 从根结点到每个叶结点的路径对应了一个判定测试序列. 因此, 决策树设计的关键是如何设计属性测试方法, 或属性划分准则.

决策树生成过程的停止条件? 决策树的生成过程采用分治法. 有三种情形会停止结点递归, 该结点将作为叶结点:

- 当前结点包含的样本全属于同一类别, 无需划分. 将该结点的标记设为这个同一类别.
- 当前结点包含的属性集为空, 或是当前结点包含的样本在所有属性上取值相同, 无法划分. 将该结点的标记设为该结点所含样本最多的类别 (实质上利用了当前结点的后验分布).
- 当前结点包含的样本集合为空, 不能划分. 将该结点的标记设为其父结点所含样本最多的类别 (实质上是把父结点的样本分布作为当前结点的先验分布).

2 单变量决策树

2.1 划分选择

属性如何选择最优的划分属性? 一般而言, 随着划分过程不断进行, 我们希望决策树的分支结点所包含的样本尽可能属于同一类别, 即结点的纯度 (purity) 越来越高. 两种最长度刻画纯度的方式是基于信息熵和基于基尼值, 从中产生了三种最常用的划分准则: 信息增益、增益率和基尼指数, 如表 2 所示. 另外, 对连续属性, 最简单的策略是使用二分法.

对离散属性, 不同的层使用不同的属性进行结点划分, 即在整个决策树中, 每个属性最多被用于一次结点划分. 然而, 若当前结点划分属性为连续属性, 该属性还可作为其后代结点的划分属性.

信息增益 (infromation gain).

计算方法包括如下三步.

1. 计算样本集合 D 中第 c 类样本所占的比例

$$p_c:=\frac{1}{m}\sum_{i=1}^{m}\mathbb{I}(y_i=c)\,. \tag{6}$$

Table 2: 常见单变量决策树使用的属性划分准则.

学习算法	属性划分准则
ID3 (iterative dichotomiser) [11]	信息增益
C4.5 [12]	信息增益 + 增益率, 二分法
CART (classification and regression tree) [2]	基尼指数

我们定义样本集合 D 的信息熵为*

$$\operatorname{ent} D := -\sum_{c=1}^{C} p_c \lg p_c \,. \tag{7}$$

2. 假设某属性有 K 个可能的取值 $a_1, a_2, \ldots, a_K$, 若使用该属性来对样本集 D 进行划分, 则会产生 K 个分支结点, 我们记 D_{a_k} 为所有在属性上取值为 a_k 的样本. 类似于公式 6 和 7 , 我们可以计算 D_{a_k} 的信息熵 $\operatorname{ent} D_{a_k}$.
3. 考虑到不同的分支结点所包含的样本数不同, 给分支结点赋予权重 $\frac{|D_{a_k}|}{|D|}$, 即样本数越多的分支结点影响越大, 于是可计算出用该属性对样本集 D 进行划分所获得的信息增益

$$IG_a := \operatorname{ent} D - \sum_{k=1}^{K} \frac{|D_{a_k}|}{|D|} \operatorname{ent} D_{a_k} \,. \tag{8}$$

一般而言, 信息增益越大, 使用该属性来进行划分所获得的纯度提升越大. 因此, 我们选择能最大化信息增益的属性.

增益率 (gain ratio).

信息增益对可取值数较多的属性有所偏好. 属性取值数越多, 每个分支结点包含的样本越少, 则这些结点包含的纯度也越大. 为了减少这种偏好可能带来的不利影响, 增益率将信息增益除以固有值 (intrinsic value)

$$GR := \frac{IG}{IV} \,, \tag{9}$$

其中

$$IV := -\sum_{k=1}^{K} \frac{|D_{a_k}|}{|D|} \lg \frac{|D_{a_k}|}{|D|} \,. \tag{10}$$

属性的可能取值数越多, 固有值通常越大. 但是, 增益率对可取值数较少的属性有所偏好. 因此, C4.5 采用了一个启发式的划分策略: 先从候选划分属性中找出信息增益高于平均水平的属性, 再从中选择增益率最高的.

*为了避免和假设函数 h 的符号混淆, 这里使用 ent 表示熵.

基尼指数 (Gini index).

类似于信息增益, 基尼指数的计算方法包括如下三步.

1. 计算样本集合 D 中第 c 类样本所占的比例 p_c, 以及样本集合 D 的基尼值

$$\operatorname{gini} D := \sum_{c=1}^{C} \sum_{c' \neq c} p_c p_{c'} = \sum_{c=1}^{C} p_c(1-p_c) = 1 - \sum_{c=1}^{C} p_c^2 \,. \tag{11}$$

基尼值反映了从数据集 D 中随机抽取两个样本, 其类别标记不一致的概率. 基尼值越小, 数据集 D 的纯度越高.

2. 计算每个分支 D_{a_k} 的基尼值 $\operatorname{gini} D_{a_k}$.
3. 用该属性对样本集 D 进行划分所对应的基尼指数是

$$GI := \sum_{k=1}^{K} \frac{|D_{a_k}|}{|D|} \operatorname{gini} D_{a_k} \,. \tag{12}$$

我们选择能最小化基尼指数的属性.

二分法 (bi-partition).

假设某属性在 D 上出现了 K 个不同的值, 将这些值按从小到大进行排序, 记为 $a_1, a_2, \ldots, a_K$. 基于划分点 t 可将 D 分为子集 D_t^- 和 D_t^+. 其中, D_t^- 包含那些在该属性上取值不大于 t 的样本, 而 D_t^+ 则包含那些在该属性上取值大于 t 的样本. 我们考察 $K-1$ 个候选划分点

$$\left\{ \frac{a_k + a_{k+1}}{2} \mid 1 \leq k < K \right\}. \tag{13}$$

然后, 可以像离散属性值一样考察这些划分点. 例如, 可对信息增益稍加改造,

$$IG := \max_t \ \operatorname{ent} D - \frac{|D_t^+|}{|D|} \operatorname{ent} D_t^+ - \frac{|D_t^-|}{|D|} \operatorname{ent} D_t^- \,. \tag{14}$$

如何评价这些属性划分准则? 现在已有许多准则用于决策树属性划分. 本质上, 各种特征选择方法均可用于决策树的分化属性选择. 然而, 有实验研究表明 [9] , 这些准则虽然对决策树的尺寸有较大影响, 但对泛化性

能的影响很有限. 理论分析指出 [13] , 信息增益和基尼系数仅在 2% 的情况下会有所不同. 相比之下, 剪枝方法和程度对决策树泛化性能的影响相当显著. 有实验研究表明 [8] , 在数据带有噪声时通过剪枝甚至可将决策树的泛化性能提高 25%.

2.2 剪枝

为什么要进行剪枝? 剪枝 (pruning) 是决策树学习算法应对过拟合的主要手段. 在决策树学习过程中, 为了尽可能正确分类训练样本, 结点划分过程将不断重复, 有时会造成决策树分支过多, 以致于把训练集的自身的一些特点当作所有数据都具有的一般性质而导致过拟合.

剪枝的基本策略? 剪枝的基本策略有两种, 预剪枝 (prepruning) 和后剪枝 (postpruning) [12] . 预剪枝是在决策树生成过程中, 对每个结点在划分前进行估计, 若当前结点的划分不能带来决策树泛化性能的提升 (如验证集性能提升), 则停止划分并将当前结点标记为叶结点. 后剪枝是先从训练集生成一棵完整的决策树, 然后自底向上地对非叶结点进行考察, 若将该结点对应的子树替换为叶结点能带来决策树泛化性能的提升或保持不变, 则将该子树替换为叶结点.

预剪枝的利弊? 预剪枝使得决策树很多分支都没有展开, 这不仅降低了过拟合的风险, 还显著减少了决策树的训练时间开销和测试时间开销. 但另一方面, 有些分支的当前划分虽不能提升泛化性能, 甚至可能导致泛化性能暂时下降, 但在其基础上进行的后续划分却有可能导致性能显著提高. 预剪枝基于贪心本质禁止这些分支展开, 给预剪枝决策树带来了欠拟合的风险.

后剪枝的利弊? 后剪枝决策树通常比预剪枝决策树保留了更多的分支. 一般情形下, 后剪枝决策树的欠拟合风险很小, 泛化性能往往优于预剪枝决策树. 但后剪枝决策树是在生成完全决策树之后进行的, 并且要自底向上地对树中的所有非叶结点进行逐一考察, 因此其训练时间开销比未剪枝决策树和预剪枝决策树都要大得多.

2.3 缺失值处理

现实任务中常常会遇到样本的某些属性值缺失. 面对缺失值, 决策树需要解决两个问题:

(1). *如何在属性值缺失的情况下进行划分属性选择?* 给定训练集 D 和某属性 a, 令 $\tilde{D}_a$ 为 D 中在属性 a 上没有缺失值的样本子集. 我们可以根据 $\tilde{D}_a$ 判断该属性的优劣. 假设我们为每个样本赋予权重 w_i (其学习时初始值为 1), 无缺失值样本 $\tilde{D}_a$ 中第 c 类所占的比例定义为

$$\tilde{p}_{ac} := \frac{\sum_{i=1}^{m} \mathbb{I}((\boldsymbol{x}_i, y_i) \in \tilde{D}_a \wedge y_i = c) w_i}{\sum_{i=1}^{m} \mathbb{I}((\boldsymbol{x}_i, y_i) \in \tilde{D}_a) w_i} . \tag{15}$$

无缺失值样本 $\tilde{D}_a$ 的信息熵定义为

$$\operatorname{ent} \tilde{D}_a := -\sum_{c=1}^{C} \tilde{p}_{ac} \lg \tilde{p}_{ac} . \tag{16}$$

无缺失值样本所占的比例定义为

$$\rho_a := \frac{\sum_{i=1}^{m} \mathbb{I}((\boldsymbol{x}_i, y_i) \in \tilde{D}_a) w_i}{\sum_{i=1}^{m} \mathbb{I}((\boldsymbol{x}_i, y_i) \in D) w_i} . \tag{17}$$

无缺失值样本 $\tilde{D}_a$ 中在属性 a 上取值为 a_k 的样本所占的比例为

$$r_{a_k} := \frac{\sum_{i=1}^{m} \mathbb{I}((\boldsymbol{x}_i, y_i) \in \tilde{D}_a \wedge x_j = a_k) w_i}{\sum_{i=1}^{m} \mathbb{I}((\boldsymbol{x}_i, y_i) \in \tilde{D}_a) w_i} . \tag{18}$$

我们可以将信息增益的计算公式推广为

$$IG_a := \rho_a \left(\operatorname{ent} \tilde{D}_a - \sum_{k=1}^{K} r_{a_k} \operatorname{ent} \tilde{D}_{a_k} \right) . \tag{19}$$

(2). *给定划分属性, 若样本在该属性上的值缺失, 如何对样本进行划分?* 若样本 $\boldsymbol{x}$ 在划分属性 a 上的取值已知, 则将 $\boldsymbol{x}$ 划入与其取值对应的子结点, 且样本权值在子结点中保持为 w_i. 若样本 $\boldsymbol{x}$ 在划分属性 a 上的取值未知, 则将 $\boldsymbol{x}$ 同时划入所有子结点, 且样本权值在与属性 a_k 对应的子结点中调整为 $r_{a_k} w_i$. 直观上看, 这就是让同一个样本以不同的概率划入不同的子结点中去.

3 多变量决策树

单变量决策树的利弊? 单变量决策树在输入空间的分类边界是轴平行的 (axis-parallel), 即它的分类边界由若干个与坐标轴平行的分段组成, 如图 2 所示. 因为每一段划分都直接对应了某个属性取值, 这样的分类边界使得学习结果有较好的可解释性. 例如, C4.5Rule [12] 将 C4.5 决策树转化为符号规则, 但 C4.5Rule 在转化过

Table 3: 多变量决策树代表算法.

学习算法	内部结点划分准则
OC1 [10]	先贪心地寻找每个属性的最优权值, 在局部优化基础上再对分类边界进行随机扰动以找到更好的边界
[3]	线性分类器学习的最小二乘法
感知机树 (perceptron tree) [15]	感知机
[6]	多层神经网络

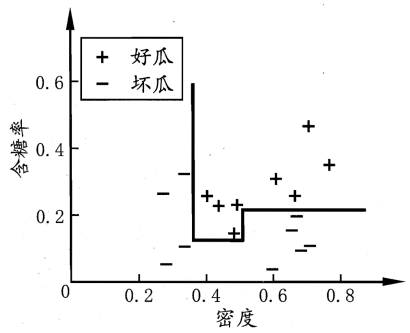

Figure 2: 单变量决策树对应的分类边界, 其对复杂真实分类边界进行分度近似. 本图源于 [19] .

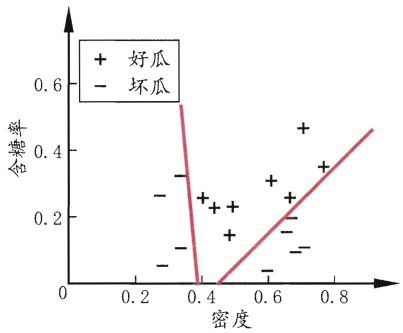

Figure 3: 多变量决策树对应的分类边界. 本图源于 [19] .

程中会进行规则前件合并、删减等操作, 因此最终规则集的泛化性能甚至可能优于原决策树. 但在学习任务的真实分类边界比较复杂时, 必须使用很多段划分才能获得较好的近似, 此时决策树会相当复杂. 由于要进行大量的属性测试, 预测时间开销也会很大.

多变量决策树 (multivariate decision tree), 或斜决策树 (oblique decision tree), 的内部结点不再是仅对某个属性, 而是对属性的线性组合进行测试. 换言之, 每个内部结点是一个形如 $\text{sign}(\boldsymbol{w}^\top \boldsymbol{x} + b)$ 的线性分类器, 以实现斜的划分边界, 如图 3 所示. 三个多变量决策树的代表算法如表 3 所示.

决策树的增量学习. 有些决策树算法可以进行增量学习, 主要机制是通过调整分支路径上的划分属性次序来对树进行部分重构. 代表性算法有 ID4 [14] 、ID5R [16] 、ITI [17] 等. 增量学习可有效地降低每次接收到新样本后的训练时间开销, 但多步增量学习后的模型会与基于全部数据训练而得的模型有较大差别.

定义 1 (增量学习 (incremental learning))**.** 在学得模型后, 再接收到训练样例时, 仅需要根据新样例对模型进行更新, 不必重新训练整个模型, 并且先前学得的有效信息不会被冲掉. 在线学习 (online learning) 是增量学习的特例, 是指每获得一个新样本就进行一次模型更新.

4 随机森林

随机森林的基本思路. 随机森林 (random forest) 是在以决策树为集学习器构建 Bagging 集成 (参见第 6 章) 的基础上, 进一步在单变量决策树的训练过程中引入了随机属性选择. 具体地说, 经典单变量决策树是在选择划分属性时是在当前结点的属性集合 (假设有 d 个属性) 中选择一个属性, 而在随机森林中, 对基决策树的每个结点, 先从该结点的属性集合中随机选择一个包含 $\tilde{d}$ 个属性的子集, 然后再从这个子集中选择一个最优属性用于划分. 这里的参数 $\tilde{d}$ 控制了随机性的引入程度: 若 $\tilde{d} = d$, 基决策树的构建和经典单变量决策树相同; 若 $\tilde{d} = 1$, 则是随机选一个属性用于划分. 一般情况下, 推荐值 $\tilde{d} := \lg d$ [1] .

随机森林简单、容易实现、计算开销小, 令人惊奇的是, 它在很多现实任务中展现出强大的性能, 被誉为代表集成学习技术水平的方法. 可以看出, 随机森林除了 Bagging 中多样性仅来自样本扰动还增加了属性扰动. 这使得最终集成的泛化性能可通过个体学习器之间

差异度的增加而进一步提升.

随机森林的收敛性和 Bagging 相似. 随机森林的起始性能往往相对较差, 特别是在集成中只包含一个集学习器时. 这是因为通过引入属性扰动, 随机森林中个体学习器的性能往往有所降低. 然而, 随着个体学习器数目的增加, 随机森林通常会收敛到更低的泛化误差. 值得一提的是, 随机森林的训练效率常优于 Bagging, 这是因为在个体决策树的构建过程中, Bagging 使用的是确定型决策树, 在选择划分属性时要对结点的所有属性进行考察, 而随机森林使用的随机型决策树只需考察一个属性子集.

5 GBDT、XGBoost、LightGBM

GBDT (gradient boosting descision tree) [5] , 也称为 GBM (gradient boosting machine), MART (multiple additive regression tree), 是在以决策树 (通常是 CART 回归树) 为集学习器构建 Gradient Boosting 集成 (参见第 6 章) 的基础上以平方损失

$$\ell(H) = \frac{1}{2}(H(\boldsymbol{x}) - y)^2 \tag{20}$$

作为 Gradient Boosting 中使用的损失函数.

定理 6. *GBDT 的 h_t 最优值是对任意 $i = 1, 2, \ldots, m$,*

$$h_t(\boldsymbol{x}_i) \approx y_i - H_{t-1}(\boldsymbol{x}_i)\,. \tag{21}$$

即用 h_t 拟合真实标记 y 和 H_{t-1} 之间的残差, 也就是说, h_t 用于弥补 H_{t-1} 对 y 拟合的不足. 具体拟合算法可以采用线性回归.

Proof. 根据第 6 章的定理 8 ,

$$h_t(\boldsymbol{x}_i) \approx -\frac{\partial \ell}{\partial H(\boldsymbol{x})}\Big|_{H(\boldsymbol{x})=H_{t-1}(\boldsymbol{x}_i)} = -(H_{t-1}(\boldsymbol{x}_i) - y)\,. \tag{22}$$

□

定理 7. *XGBoost (extreme gradient boosting) [4] 的 h_t 最优值是对任意 $i = 1, 2, \ldots, m$,*

$$h_t(\boldsymbol{x}_i) \approx y_i - H_{t-1}(\boldsymbol{x}_i)\,. \tag{23}$$

即用 h_t 拟合真实标记 y 和 H_{t-1} 之间的残差.

Proof. 根据第 6 章的定理 9 ,

$$\begin{aligned} h_t(\boldsymbol{x}_i) &\approx -\left(\frac{\partial^2 \ell}{\partial H(\boldsymbol{x})^2}\Big|_{H(\boldsymbol{x})=H_t(\boldsymbol{x}_i)}\right)^{-1} \frac{\partial \ell}{\partial H(\boldsymbol{x})}\Big|_{H(\boldsymbol{x})=H_{t-1}(\boldsymbol{x}_i)} \\ &= -(H_{t-1}(\boldsymbol{x}_i) - y)\,. \end{aligned} \tag{24}$$

□

XGBoost 为什么要对模型进行正则化? XGBoost 优化时还考虑了模型的结构风险, 对决策树叶结点数量和每个叶结点输出值的 ℓ_2 范数进行正则化. 这种正则化的动机是使 h_t 的拟合能力不要过强, XGBoost 希望用多个拟合能力比较弱的基学习器, 而不是少数拟合能力强的基学习器. 这样某个基学习器的误差对整体的影响比较小. 此外, 为了进一步缓解过拟合风险, 类似于 Bagging 的思想, XGBoost 每次只使用一部分样本拟合残差.

GBDT 和 XGBoost 的相同和不同点? 相同点: GBDT 和 XGBoost 都属于 Gradient Boosting 算法. 不同点: (1). XGBoost 对损失函数进行二阶泰勒展开近似. (2). XGBoost 使用了正则化约束模型复杂度和样本采样以缓解过拟合. (3). XGBoost 在工程上的创新更大, 力图将计算资源利用到极致 (并行、分布式、稀疏数据处理、缓存优化等). (4). XGBoost 可以自动处理缺失值. (5). XGBoost 支持基学习器使用线性分类器和自定义损失函数.

GBDT 和 XGBoost 的超参数比较多, 实践中需要仔细调节, 如下两篇文章分别介绍了 GBDT 和 XGBoost 的调参细节:

- https://www.analyticsvidhya.com/blog/2016/02/complete-guide-parameter-tuning-gradient-boosting-gbm-python/.
- https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/.

XGBoost 和 LightGBM 的相同和不同点? 相同点: XGBoost 和 LightGBM [7] 都属于 Gradient Boosting 算法. 不同点: (1). XGBoost 在划分结点计算信息增益时需要遍历所有样本, LightGBM 在划分时忽略那些梯度小的样本, 只在剩余的很小的样本集合中计算增益. (2). 属性通常是稀疏且互斥的 (不同时取非零值), LightGBM 组合考虑这些互斥的属性, 进一步降低计算资源消耗.

6 快问快答

*ID3、C4.5、CART、随机森林、GBDT、XGBoost 和 LightGBM 的区别是什么？*答案见上文.

*决策树模型是否需要对特征进行规范化？*通常不需要. 对基于梯度下降优化的学习算法特征规范化对收敛影响较大, 而决策树在划分结点时采用信息增益等准则受不同属性数值范围不同的影响较小.

*为什么集成学习算法 (如随机森林、GBDT) 偏好决策树模型作为基学习器？*集成学习的关键是要使得基学习器具有多样性. Bagging 通过自助法采样生成不同的训练数据, 如果基学习器使用稳定学习器, 对数据集的扰动不敏感, 对这些基学习器的集成不会提升泛化性能. 而决策树是一种不稳定学习器, 适合作为 Bagging 的基学习器. 对 Boosting 来说, 需要每个基学习器都优于随机猜测, 在不对每个基学习器分别调参的条件下, 决策树相比其他学习算法更容易做到这一点. 决策树可以通过控制深度控制模型复杂度. 另外决策树训练起来比较快, 使得 Boosting 整体训练效率较高.

References

[1] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. 5

[2] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984. 3

[3] C. E. Brodley and P. E. Utgoff. Multivariate decision trees. *Machine Learning*, 19(1):45–77, 1995. 5

[4] T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 785–794, 2016. 6

[5] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001. 6

[6] H. Guo and S. B. Gelfand. Classification trees with neural network feature extraction. *IEEE Transactions on Neural Networks*, 3(6):923–933, 1992. 5

[7] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T. Liu. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems 30 (NIPS)*, pages 3149–3157, 2017. 6

[8] J. Mingers. An empirical comparison of pruning methods for decision tree induction. *Machine Learning*, 4:227–243, 1989. 4

[9] J. Mingers. An empirical comparison of selection measures for decision-tree induction. *Machine Learning*, 3:319–342, 1989. 3

[10] S. K. Murthy, S. Kasif, and S. Salzberg. A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2:1–32, 1994. 5

[11] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986. 3

[12] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993. 3, 4

[13] L. E. Raileanu and K. Stoffel. Theoretical comparison between the Gini index and information gain criteria. *Annals of Mathematics and Artificial Intelligence*, 41(1):77–93, 2004. 4

[14] J. C. Schlimmer and D. H. Fisher. A case study of incremental concept induction. In *Proceedings of the 5th National Conference on Artificial Intelligence (AAAI)*, pages 496–501, 1986. 5

[15] P. E. Utgoff. Perceptron trees: A case study in hybrid concept representations. In *Proceedings of the 7th National Conference on Artificial Intelligence (AAAI)*, pages 601–606, 1988. 5

[16] P. E. Utgoff. Incremental induction of decision trees. *Machine Learning*, 4:161–186, 1989. 5

[17] P. E. Utgoff, N. C. Berkman, and J. A. Clouse. Decision tree induction based on efficient tree restructuring. *Machine Learning*, 29(1):5–44, 1997. 5

[18] 吴建鑫. 模式识别. 机械工业出版社, 2019. 1

[19] 周志华. 机器学习. 清华大学出版社, 2016. 5