

# 1. 决策树

Collected by Sizhe Zhou  
2022. 4. 6.

- 优点：可读性，分类速度快

## 1.1 决策树模型与学习

### 1.1.1 决策树模型

#### · 定义：

- 由 node 与 directed edge 组成
- node {
  - internal node : 一个特征
  - leaf node : 一个类

### 1.1.2 决策树与 if-then 规则

- 可看成一个 if-then 规则的集合：
  - 根(if)  $\rightarrow$  叶(then)，一条路径一条规则
  - 互斥且完备

### 1.1.3 决策树与条件概率分布

- 可看 feature space 的一个 partition 上的 条件概率分布：  
 $P(Y|X)$

决策树分类时将该结点的实例强行分到概率大的一类。

### 1.1.4 决策树学习

- 训练数据集： $D = \{(x_i, y_i)\}_{i=1}^N$  with  $x_i = (x_i^{(1)}, x_i^{(2)}, \dots, x_i^{(n)})^T$   
 $y_i \in \{1, 2, \dots, K\}$

- 目标：1. 不矛盾 2. 泛化能力
- 从所有可能中选最优  $\rightarrow$  NP 完全问题  $\rightarrow$  启发式算法  
 $\rightarrow$  sub-optimal

• (Often) 将所有数据放到根结点，递归地：

1. 选择最优特征

2. 分割成子集

3. 若这些子集正确分类，构建叶结点

4. 否则重复以上步骤

直至所有子集被正确分类或无合适特征。(考虑局部最优)

再对已生成的树自下而上剪枝：去掉过于细分的叶结点，使其回退到父结点，甚至更高结点，并将其改为新的叶结点。(考虑全局最优)

## 1.2 特征选择

### 1.2.1 特征选择问题

· 信息增益 / 信息增益比

### 1.2.2 信息增益

·  $X$  为取有限个值的离散随机变量：

$$P(X = x_i) = p_i, \quad i = 1, 2, \dots, n \quad (\text{unit: bit/nat})$$

$$\text{熵 } H(X) = -\sum_{i=1}^n p_i \log p_i \quad (\text{定义 } 0 \log 0 = 0)$$

·  $H(X)$  越大，不确定性越大：  $0 \leq H(p) \leq \log n$

$$\cdot \text{条件熵 } H(Y|X) = \sum_{i=1}^n p_i H(Y|X = x_i)$$

· 信息增益 (information gain)：得知特征  $X$  的信息而使得类  $Y$  的信息的不确定减少的程度

· 定义：特征  $A$ ，训练数据集  $D$ ，信息增益  $g(D, A)$ ，  
 $H(\cdot)$  表示经验熵 (由数据估计)

$$g(D, A) = H(D) - H(D|A)$$

· 互信息:  $H(Y) - H(Y|X)$

· 信息增益算法:

训练数据集为  $D$ , 类  $C_k$  with  $k=1, 2, \dots, K$

$|C_k|$  为属于类  $C_k$  的样本个数,  $\sum_{k=1}^K |C_k| = |D|$

设特征  $A$  有  $n$  个不同取值  $\{a_1, a_2, \dots, a_n\}$ , 并根据  $A$  将  $D$  划分为  $n$  个子集  $D_1, D_2, \dots, D_n$ .  $\sum_{i=1}^n |D_i| = |D|$

记  $D_i$  中属于类  $C_k$  的样本集合为  $D_{ik} = D_i \cap C_k$

· 算法 1.1

输入:  $D$  与  $A$

输出:  $g(D, A)$

$$(1) \quad H(D) = - \sum_{k=1}^K \frac{|C_k|}{|D|} \log_2 \frac{|C_k|}{|D|}$$

$$(2) \quad H(D|A) = \sum_{i=1}^n \frac{|D_i|}{|D|} H(D_i|A) = - \sum_{i=1}^n \frac{|D_i|}{|D|} \sum_{k=1}^K \frac{|D_{ik}|}{|D_i|} \log_2 \frac{|D_{ik}|}{|D_i|}$$

$$\qquad \qquad \qquad H(D_i)$$

$$(3) \quad g(D, A) = H(D) - H(D|A)$$

### 1.2.3 信息增益比

· 信息增益  $\rightarrow$  bias towards 取值较多的 feature

$$\cdot \text{定义: } g_R(D, A) = \frac{g(D, A)}{H_A(D)}$$

$$\text{with } H_A(D) = - \sum_{i=1}^n \frac{|D_i|}{|D|} \log_2 \frac{|D_i|}{|D|}$$

$n$  为  $A$  取值个数

## 1.3 决策树的生成

### 1.3.1 ID3 算法

· 算法 1.2：

输入：  $D, A, \text{ 固定 } \varepsilon$

输出： 决策树  $T$

(1)  $D$  中所有实例属于  $C_k$ ,  $T$  为单结点树, 将类  $C_k$  作为该结点

(2) 若  $A = \emptyset$ ,  $T$  为单结点树, 并将  $D$  中实例数最大的类  $C_k$  作为该结点的类标记, 返回  $T$ ;

(3) 否则, 按算法 1.1 计算  $g(D, A)$ , 选择  $A_g = \arg \max_A g(D, A)$

(4) 若  $g(D, A_g) < \varepsilon$ , 置  $T$  为单结点树, 并将  $D$  中实例数最大的类  $C_k$  作为该结点的类标记, 返回  $T$ ;

(5) 否则, 对  $A_g$  中每一可能值  $a_i$ , 依  $A_g = a_i$  将  $D$  分割为若干非空子集  $D_i$ , 将  $D_i$  中实例数最大的类作为该结点的类标记, 构建子结点, 由结点及其子结点构成树  $T$ , 返回  $T$ ;

(6) 对第  $i$  个子结点, 以  $D_i$  为训练数据集, 以  $A - \{A_g\}$  为特征集, 递归地调用 (1) ~ (5), 得到子树  $T_i$ , 返回  $T_i$ .

### 1.3.2 C4.5 的生成算法

· 算法 1.3

输入:  $D$ ,  $A$ , 固定值  $\varepsilon$

输出: 决策树  $T$

(1)  $D$  中所有实例属于  $C_k$ ,  $T$  为单结点树, 将类  $C_k$  作为该结点

(2) 若  $A = \emptyset$ ,  $T$  为单结点树, 并将  $D$  中实例数最大的类  $C_k$  作为该结点的类标记, 返回  $T$ ;

(3) 否则, 计算  $g_R(D, A)$ , 选择  $A_g = \arg \max_A g(D, A)$

(4) 若  $g_R(D, A_g) < \varepsilon$ , 置  $T$  为单结点树, 并将  $D$  中实例数最大的类  $C_k$  作为该结点的类标记, 返回  $T$ ;

(5) 否则, 对  $A_g$  中每一可能值  $a_i$ , 依  $A_g = a_i$  将  $D$  分割为若干非空子集  $D_i$ , 将  $D_i$  中实例数最大的类作为该结点的类标记, 构建子结点, 由结点及其子结点构成树  $T$ , 返回  $T$ ;

(6) 对第  $i$  个子结点, 以  $D_i$  为训练数据集, 以  $A - \{A_g\}$  为特征集, 递归地调用 (1) ~ (5), 得到子树  $T_i$ , 返回  $T_i$ .

## 1.4 决策树的剪枝

- 极小化整体 loss function 或 cost function.

- 设  $T$  的叶结点个数为  $|T|$ ,  $t$  是  $T$  的叶结点, 该叶结点有  $N_t$  个样本点, 其中  $k$  类的样本点有  $N_{tk}$  个,  $k = 1, 2, \dots, K$ .

$H_t(T)$  为  $t$  上的经验熵,  $\alpha \geq 0$  为参数:

定义学习损失函数:

$$C_\alpha(T) = \sum_{t=1}^{|T|} N_t H_t(T) + \alpha |T|$$

with

$$H_t(T) = - \sum_k \frac{N_{tk}}{N_t} \log \frac{N_{tk}}{N_t}$$

$$C(T) := \sum_{t=1}^{|T|} N_t H_t(T) = - \sum_{t=1}^{|T|} \sum_{k=1}^K N_{tk} \log \frac{N_{tk}}{N_t}$$

$$\Rightarrow C_\alpha(T) = C(T) + \alpha |T|$$

↑                   ↑  
对 D      模型复杂度

$\alpha$  越大  $\rightarrow$  选择较简单的树

- $\min C_\alpha(T) \Leftrightarrow$  正则化的 MLE

#### 算法 1.4

输入:  $T, \alpha$

输出: 修剪后的子树  $T_2$

(1) 计算每个结点的经验熵

(2) 递归地从树的叶结点向上回缩:

设一组叶结点回缩至其父结点前后整体树为  $T_B, T_A$ ,  
若  $C_\alpha(T_A) \leq C_\alpha(T_B)$ , 则剪枝, 即将父结点变为新的叶结点  
(可由动态规划实现)

(3) 返回(2). 直至不能继续, 返回  $T_2$

#### 1.5 CART 算法:

(classification and regression tree)

- 生成
- 剪枝

- 假设决策树为二叉树，内部结点特征的取值为是和否。  
左分支为是的分支，右为否。

## 1.5.1 CART 生成

### 1. 回归树的生成

- 输入  $X$ ，输出  $y$  为连续变量， $D = \{(x_i, y_i)\}_{i=1}^N$

### 算法 1.5 (最小二乘回归树生成算法)

Input:  $D$

Output: 回归树  $f(x)$

(1) 选择最优切分变量  $j$  与切分点  $s$

$$\text{使 } \min_{j,s} \left[ \min_{C_1} \sum_{x_i \in R_1(j,s)} (y_i - C_1)^2 + \min_{C_2} \sum_{x_i \in R_2(j,s)} (y_i - C_2)^2 \right]$$

(2) 用  $(j,s)$  划分区域，并决定输出值：

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

$$\hat{C}_m = \frac{1}{N_m} \sum_{x_i \in R_m(j,s)} y_i, \quad x \in R_m, \quad m=1,2$$

(3) 继续对  $R_1(j,s), R_2(j,s)$  用 (1), (2)，直至满足停止条件

(4) 将输入空间划分为  $M$  个区域  $R_1, R_2, \dots, R_M$ ,

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

### 2. 分类树的生成

- 定义(基尼指数)：  $K$  个类， $P_k := P(\text{样本点} \in \text{第 } k \text{ 类})$

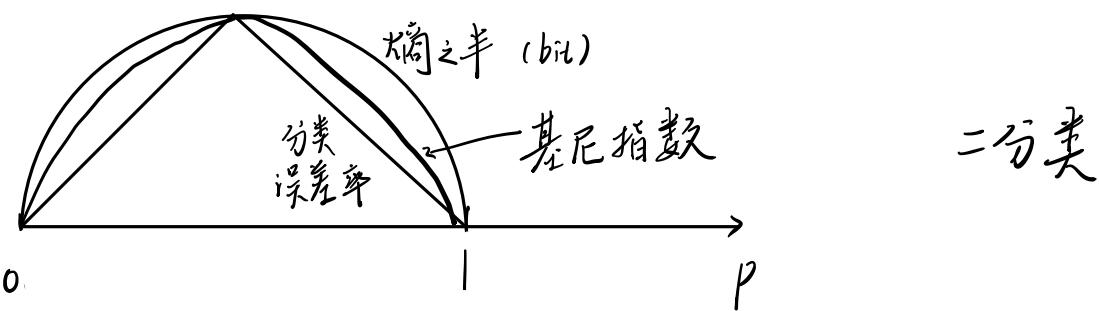
$$\text{Gini}(P) = \sum_{k=1}^K P_k(1-P_k) = 1 - \sum_{k=1}^K P_k^2$$

$$\cdot D \rightarrow \begin{cases} D_1 = \{(x, y) \in D \mid A(x) = a\} \\ D_2 = D - D_1 \end{cases}$$

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

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

· 基尼指数越大，样本集合不确定性越大



### · 算法 1.6 (CART 生成)

*Input*:  $D$ , 停止计算的条件

*Output*: 决策树

(1) 设结点的训练数据集为  $D$ , 计算每个特征  $A$  的每个可能值  $a$  下,  $D \rightarrow D_1, D_2$  的 Gini Index.

(2) 选择  $a = \arg \min Gini\ Index$ . 从现结点生成两个子结点, 将  $D$  依特征  $\overset{a}{\sim}$  分到其中去

(3) 对两个子结点递归调用 1), 2). 直至满足停止条件

(4) 返回决策树

· 停止条件: 结点中样本个数 < 预定阈值, 或样本集的 Gini Index < 阈值, 或无更多特征

## 1.5.2 CART 剪枝

1. 剪枝，形成一个子树序列：

- $C_\alpha(T) = C(T) + \alpha|T|$

$C(T)$  可为 Gini Index

- 易证，最优子树唯一

- Breiman et al. 证明：可以用递归的方法对树进行剪枝。

将  $\alpha$  从小增大， $0 = \alpha_0 < \alpha_1 < \dots < \alpha_n < +\infty$ ，产生一系列区间

$[\alpha_i, \alpha_{i+1})$ ， $i = 0, 1, \dots, n$ 。剪枝得到的子树序列对应着

区间  $\alpha \in [\alpha_i, \alpha_{i+1})$ ， $i = 0, \dots, n$  的最优子树序列  $\{T_0, T_1, \dots, T_n\}$ ，序列中的子树是嵌套的。

2. 在子树序列  $T_0, T_1, \dots, T_n$  中通过 Cross Validation 选取最优子树  $T_\alpha$

- 利用独立验证数据集

- 当最优子树  $T_k$  确定时， $\alpha_k$  也确定

• 算法 1.7 (CART 剪枝)

Input: CART 算法生成的  $T_0$

Output:  $T_\alpha$

(1) 设  $k = 0$ ,  $T = T_0$

(2) 设  $\alpha = +\infty$

(3) 自下而上地对各内部结点 $t$ 计算  $C(T_t)$ ,  $|T_t|$  以及

$$g(t) = \frac{C(t) - C(T_t)}{|T_t| - 1}$$

$$\alpha = \min(\alpha, g(t))$$

这里  $T_t$  表以  $t$  为根结点的子树

(4) 对  $g(t) = \alpha$  的内部结点 $t$  以多数表决法决定其类, 得到树  $T$

(5) 设  $k = k+1$ ,  $\alpha_k = \alpha$ ,  $T_k = T$

(6) 如果  $T_k$  不是由根结点及 2 个叶结点构成的树, 则回到

(2); 否则令  $T_k = T_n$

(7) 采用 Cross Validation 在  $T_0, T_1, \dots, T_n$  中选取最优子树  $T_\alpha$