*李涪 Li Hao 2022.11.29*

# 04决策树

## 4.1基本流程

- 结构
  - 一个根节点 —— 包含样本全集
  - 若干个内部结点 —— 每个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
  - 若干个叶结点 —— 对应于决策结果，其他每个结点对应于一个属性测试
- 目的 —— 为了产生一棵泛化能力强，即处理未见示例能力强的决策树
- 策略 —— 分而治之(divide-and conquer)
- 算法

(图4.2)

```
输入：训练集 D = {(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)};
      属性集 A = {a_1, a_2, ..., a_d}.
过程：函数 TreeGenerate(D, A)
1: 生成结点 node;
2: if D 中样本全属于同一类别 C then
3:     将 node 标记为 C 类叶结点; return
4: end if
5: if A = ∅ OR D 中样本在 A 上取值相同 then
6:     将 node 标记为叶结点，其类别标记为 D 中样本数最多的类; return
7: end if
8: 从 A 中选择最优划分属性 a_*;
9: for a_* 的每一个值 a_*^v do
10:    为 node 生成一个分支; 令 D_v 表示 D 中在 a_* 上取值为 a_*^v 的样本子集;
11:    if D_v 为空 then
12:        将分支结点标记为叶结点，其类别标记为 D 中样本最多的类; return
13:    else
14:        以 TreeGenerate(D_v, A \ {a_*}) 为分支结点
15:    end if
16: end for
输出：以 node 为根结点的一棵决策树
```

(1)当前结点包含的样本全属于同一类别，无需划分

(2)当前属性集为空，或是所有样本在所有属性上取值相同，无法划分 —— 把当前结点标记为叶结点，并将其类别设定为该结点所包含样本最多的类别

(3)当前结点包含的样本集合为空，不能划分 —— 把当前结点标记为叶结点，但将其类别设定为其父结点所含样本最多的类别

递归返回的三种情况

## 4.2划分选择

### 4.2.1信息增益

- 纯度(purity):决策树的分支结点所包含的样本属于同一类别的程度
- 信息熵(information entropy):度量样本集合纯度最常用的一种指标。值越小，样本集合的纯度越高。 (4.1) $\mathrm{Ent}(D) = -\sum_{k=1}^{|\mathcal{Y}|} p_k \log_2 p_k$
- 信息增益(information gain)(互信息):已知一个随机变量的信息后使得另一个随机变量的不确定性减少的程度。信息增益越大，意味着使用属性a来进行划分所获得的"纯度提升"越大。对可取值数目较多的属性有所偏好。 (4.2) $\mathrm{Gain}(D,a) = \mathrm{Ent}(D) - \sum_{v=1}^{V} \frac{|D^v|}{|D|} \mathrm{Ent}(D^v)$
- ID3决策树算法：以信息增益为准则来选择划分属性

### 4.2.2增益率

(4.3) 其中 $\mathrm{Gain\_ratio}(D,a) = \frac{\mathrm{Gain}(D,a)}{\mathrm{IV}(a)}$ ,

$\mathrm{IV}(a) = -\sum_{v=1}^{V} \frac{|D^v|}{|D|} \log_2 \frac{|D^v|}{|D|}$

- IV称为属性a的固有值(intrinsic value)
- 属性a的可能取值数目越多，固有值通常越大

增益率准则对可取值数目较少的属性有所偏好，可减少这种偏好可能带来的不利影响

C4.5决策树算法：不是直接选择增益率最大的候选划分属性，使用一个启发式：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

### 4.2.3基尼指数

用基尼值衡量数据集纯度

(4.5) $\begin{aligned}\mathrm{Gini}(D) &= \sum_{k=1}^{|\mathcal{Y}|} \sum_{k' \neq k} p_k p_{k'} \\ &= 1 - \sum_{k=1}^{|\mathcal{Y}|} p_k^2 .\end{aligned}$

(4.6) $\mathrm{Gini\_index}(D,a) = \sum_{v=1}^{V} \frac{|D^v|}{|D|} \mathrm{Gini}(D^v)$

CART决策树：使用基尼指数(Gini index)选择划分属性。基尼指数越小，数据集纯度越高。

## 4.3剪枝处理

剪枝(pruning):决策树学习算法对付"过拟合"的主要手段，通过主动去掉一些分支来降低过拟合的风险。

### 4.3.1预剪枝

预剪枝(prepruning):在决策树生成过程中，对每个结点在划分前先进行估计，当当前结点的划分不能带来决策树泛化性能提升，则停止划分并将当前结点标记为叶结点。

决策树桩(decision stump):一棵仅有一层划分的决策树。

优点：降低了过拟合的风险，显著减少了决策树的训练时间开销和测试时间开销。

缺点：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欠拟合的风险。

### 4.3.2后剪枝

后剪枝(postpruning):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提升，则将该子树替换为叶结点 $\overline{\text{换}}$。

优点：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。

缺点：训练时间开销比未剪枝决策树和预剪枝决策树要大得多。

## 4.4连续与缺失值

### 4.4.1连续值处理

连续属性的可取值数目不再有限，不能直接根据连续属性的可取值来对结点进行划分。

C4.5决策树算法：采用二分法(bi-partition)对连续属性进行处理。

(4.8) $\begin{aligned}\mathrm{Gain}(D,a) &= \max_{t \in T_a} \mathrm{Gain}(D,a,t) \\ &= \max_{t \in T_a} \mathrm{Ent}(D) - \sum_{\lambda \in \{-,+\}} \frac{|D_t^\lambda|}{|D|} \mathrm{Ent}(D_t^\lambda)\end{aligned}$

选择使Gain最大化的划分点

若当前结点划分属性为连续值，该属性还可作为其后代结点的划分属性。

### 4.4.2缺失值处理

现实任务中常会遇到不完整样本，简单地放弃不完整样本，仅使用无缺失值样本进行学习是对数据信息极大的浪费。

对每一个样本赋予一个权重，并定义

无缺失值样本所占比例(4.9) $\rho = \frac{\sum_{\boldsymbol{x} \in \tilde{D}} w_{\boldsymbol{x}}}{\sum_{\boldsymbol{x} \in D} w_{\boldsymbol{x}}}$

无缺失值样本中第k类所占比例(4.10) $\tilde{p}_k = \frac{\sum_{\boldsymbol{x} \in \tilde{D}_k} w_{\boldsymbol{x}}}{\sum_{\boldsymbol{x} \in \tilde{D}} w_{\boldsymbol{x}}} \quad (1 \leqslant k \leqslant |\mathcal{Y}|)$

无缺失值样本中在属性a上取值a^v的样本所占比例(4.11) $\tilde{r}_v = \frac{\sum_{\boldsymbol{x} \in \tilde{D}^v} w_{\boldsymbol{x}}}{\sum_{\boldsymbol{x} \in \tilde{D}} w_{\boldsymbol{x}}} \quad (1 \leqslant v \leqslant V)$

信息熵的增益(4.12)

$\begin{aligned}\mathrm{Gain}(D,a) &= \rho \times \mathrm{Gain}(\tilde{D},a) \\ &= \rho \times \left( \mathrm{Ent}\left(\tilde{D}\right) - \sum_{v=1}^{V} \tilde{r}_v \, \mathrm{Ent}\left(\tilde{D}^v\right) \right) .\end{aligned}$

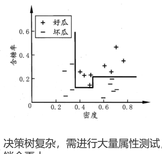
其中由式(4.1), 有 $\mathrm{Ent}(\tilde{D}) = -\sum_{k=1}^{|\mathcal{Y}|} \tilde{p}_k \log_2 \tilde{p}_k .$

## 4.5多变量决策树

多变量决策：把每个属性视为坐标空间中的一个坐标轴，则d个属性描述的样本就对应d维空间中的一个数据点，对样本分类则意味着在这个坐标空间中寻找不同类样本之间的分类边界。

特点：决策树所形成的分类边界轴平行(axis-parallel)，即它的分类边界由若干个与坐标轴平行的分段组成。

(图4.11)




决策树复杂，需进行大量属性测试，预测时间开销会更大。

多变量决策树(multivariate decision tree):实现"斜划分"甚至更复杂划分的决策树。决策树对复杂分类边界进行分段近似。

(图4.12)