4.1 基本流程

4.1.1 决策树的基本概念

决策树 (decision tree) 是一类常见的机器学习方法. 在二分类中, 学得一个模型用以对新示例进行分类, 整个把样本分类的任务, 可以堪称对"当前样本属于正类吗?"这个问题的"决策"或"判定"过程. 如在对"这是好瓜吗?"这个问题决策时, 通常会进行一系列的判断或"子决策".

决策过程的最终结论对应了我们所希望的判定结果, 如"是"或"不是"好瓜; 决策过程中提出的每个判定问题都是对某个属性的"测试"; 每个测试的结果或是导出最终结论, 或是导出进一步的判定问题(其考虑的范围是上次决策结果的限定范围之内).

4.1.2 根结点, 叶结点和内部结点

1 三类结点的概念

1. 根结点: 第一个判定的集合, 根结点包含样本全集
2. 叶结点: 叶结点对应于决策结果
3. 内部结点: 中间的结点, 需要进一步进行属性测试的结点

总结:

1. 一般的，一棵决策树包含一个根结点、若干个内部结点和若干个叶结点;
2. 叶结点对应于决策结果, 其他每个结点则对应于一个属性测试(包括根结点和内部结点)
3. 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
4. 根结点包含样本全集
5. 从根结点到每个叶结点的路径对应了一个判定测试序列

决策树学习的目的是为了产生一棵**泛化能力强**，即处理未见示例能力强的决策树，其基本流程遵循简单且直观的"**分而治之**" (divide-and-conquer)策略

4.1.3 决策树的基本算法流程

1 决策树的基本算法

输入: 训练集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
　　属性集 $A = \{a_1, a_2, \ldots, a_d\}$.
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** $A = \varnothing$ **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, $A \setminus \{a_*\}$)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

图 4.2　决策树学习基本算法

3 具体递归过程

决策树的生成是一个递归过程, 在决策树基本算法中, 有**三种情形**会导致递归返回:

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;
3. 当前结点包含的样本集合为空, 不能划分.

> 在第 (2) 种情形下, 把当前结点标记为叶结点, 并将其类别设定为该结点所含样本最多的类别;
>
> 在第 (3) 种情形下, 同样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.

4 递归过程的详细解析

1. 这是最理想的情况, 也即是**子集只包含同一类别的样本**. 若递归划分过程中某子集已经只含有某一类别的样本 (例如只含好瓜或坏瓜) , 那么此时划分的目的已经达到了, 无需再进行划分, 这种情况就是**递归返回的情形1**
2. 递归划分时每次选择一个属性, 并且**划分依据属性不能重复使用** (从候选依据属性中将当前使用的依据属性剔除, 这是因为根据某属性划分之后, 产生的各个子集在该属性上的取值相同). **但样本的属性是有限的**, 因此划分次数不超过属性个数; **若所有属性均已被作为划分依据, 此时子集中仍含有多类样本** (例如仍然同时含有好瓜和坏瓜) .但是因已无属性可用作划分依据 (即子集中样本在各属性上取值均相同,但却无法达到子集只包含同一类 别的样本). 此时只能少数服从多数, 以**此子集中样本数最多的类为标记**, 此即为**递归返回的情形 2 中的** $A = \varnothing$
3. 每递归一次, 候选的属性个数就减少一个. 假如现在还剩下两个属性, 触感和色泽. 且此时样本是多类的(即好瓜和坏瓜). 但是剩下的样本触感和色泽都是一样的, **即当前样本集合在任何候选属性上的取值相同, 无法再通过属性**

划分了.(两个以上属性的情况类似). 此时也只能少数服从多数, 以**此子集中样本数最多的类为标记**, 此即为**递归返回情形 2 中的 "D 中样本在 A 上取值相同"**

4. **根据某属性进行划分时, 若该属性多个属性值中的某个属性值不包含任何样本**. 如当前子集 D_v 以"色泽"属性来划分, "色泽"的属性值有: "青绿", "乌黑"和"浅白". 可发现没有样本的属性值为"浅白"的, 只有"青绿"和"乌黑". 此时对于取值为"浅白"的分支, 因为没有样本落入, 将其标记为叶结点, 其类别标记为 D_v 中样本最多的类.(注意, 此分支必须保留, 因为在测试时, 可能会有样本落入该分支). 此种情况即为**递归返回的情形 3.**

4.2 划分选择

决策树学习的关键点在于: 如何选择最优划分属性.

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.

4.2.1 信息增益

1 信息熵

"信息熵"(information entropy)是对量样本集合纯度最常用的一种指标. 它的定义如下:

假定当前样本集合 D 中第 k 类所占的比例为 $p_k(k=1,2,\ldots,|\mathcal{Y}|)$, 则 D 的信息熵定义为:

$$\operatorname{Ent}(D)=-\sum_{k=1}^{|\mathcal{Y}|}p_k\log_2 p_k \tag{4.1}$$

$Ent(D)$ 的值越小, 则 D 的纯度越高.

> 注: 这里的 k 就是西瓜里的"好瓜"和"坏瓜"这**两个类别**

2 信息增益

假定离散属性 a 有 V 个可能的取值 $\{a^1,a^2,\ldots,a^V\}$, 若使用 a 来对样本集 D 进行划分, 则会产生 V 个分支结点, 其中第 v 个分支结点包含了 D 中所有在属性 a 上取值为 a^v 的样本, 记为 D^v . 可根据 (4.1) 计算出 D^v 的信息熵, 同时再考虑到不同分支结点所包含的样本数不同, 给分支结点赋予权重 $|D^v|/|D|$, 即样本数越多的分支结点的影响越大, 于是可计算出**用属性 a 对样本集 D 进行划分所获得的"信息增益"**(information gain)

$$\operatorname{Gain}(D,a)=\operatorname{Ent}(D)-\sum_{v=1}^{V}\frac{|D^v|}{|D|}\operatorname{Ent}(D^v) \tag{4.2}$$

一般而言, **信息增益越大**, 则意味着使用属性 a 来进行划分所获得的**"纯度提升"越大**. 因此,我们可以用信息增益来进行决策树的划分属性选择, 即在图4.2算法第 8 行选择属性 $a_*=\arg\max_{a\in A}\operatorname{Gain}(D,a)$. $ID3$ **决策树学习算法**就是基于信息增益来进行属性划分的.

> 注: 信息增益准则对**可取值数目较多**的属性有**偏好**

4.2.2 增益率

信息增益准则对可取值数目较多的属性有所偏好, 为减少这种偏好可能带来的不利影响, 著名的 $C4.5$ **决策树算法** 不直接使用信息增益, 而是使用**"增益率"** (gain ratio) 来选择最优划分属性.采用与式 (4.2) 相同的符号表示, **增益率**定义为:

$$\text{Gain\_ratio}(D,a)=\frac{\text{Gain}(D,a)}{\text{IV}(a)} \tag{4.3}$$

其中,

$$\text{IV}(a)=-\sum_{v=1}^{V}\frac{|D^v|}{|D|}\log_2\frac{|D^v|}{|D|} \tag{4.4}$$

$\text{IV}(a)$ 称为属性 a 的"固有值". 属性 a 的可能取值数目越多 (即 V 越大) , 则 $\text{IV}(a)$ 的值通常会越大.

> 注: 增益率准则对可取值数目较少的属性有所偏好.

$C4.5$**算法划分原则:**

$C4.5$ 算法并不是直接选择增益率最大的候选划分属性, 而是使用了一个启发式: **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

$CART$ **决策树**使用"基尼指数" (Gini index) 来选择划分属性.采用与式 (4.1) 相同的符号, **数据集** D **的纯度可用基尼值**来度量:

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

直观来说, $Gini(D)$ 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. 因此, $Gini(D)$ 越小, 则数据集 D 的纯度越高.

属性 a 的基尼指数定义为 :

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

我们在候选属性集合 A 中, 选择那个使得划分后基尼指数最小的属性作为最优划分属性, 即
$a_* = \underset{a \in A}{\arg\min}\ \text{Gini\_index}\ (D, a)$

4.3 剪枝处理

4.3.1 剪枝的基本概念

剪枝 (pruning)是决策树学习算法对付**"过拟合"**的主要手段. 在决策树学习中, 有时会造成决策树分支过多, 把训练样本学得"太好", 以致于**把训练集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**. 因此, 可通过**主动去掉一些分支**来**降低过拟合**的风险.

决策树剪枝的基本策略有**"预剪枝"** (prepruning)和**"后剪枝 "**(post-pruning).

1. 预剪枝是指在决策树生成过程中，对每个结点在**划分前先进行估计**，若当前结点的划分**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**
2. 后剪枝则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 (*也就是把该结点和下面的子树整体替换成叶结点*) 能带来决策树泛化性能提升, 则将该子树替换为叶结点.

如何判断决策树泛化性能是否提升:

假定采用留出法, 即预留一部分数据用作"验证集"以进行性能评估

4.3.2 预剪枝

见西瓜书P81-P82例子

4.3.3 后剪枝

见西瓜书P82-P83例子

4.4 连续与缺失值

4.4.1 连续值处理

目前讨论的都是离散属性, 对于连续属性来说, 需要把连续属性离散化. 最简单的就是采用二分法对连续属性进行处理. 这正是 $C4.5$ 决策树算法中采用的机制.

给定样本集 D 和连续属性 a , 假定 a 在 D 上出现了 n 个不同的取值, 将这些值从小到大进行排序, 记为 $\left\{a^1, a^2, \ldots, a^n\right\}$. 基于划分点 t 可将 D 分为子集 D_t^- 和 D_t^+ , 其中 D_t^- 包含那些在属性 a 上取值不大于 t 的样而 D_t^+ 则包含那些在属性 a上取值大于 t 的样本. 显然, 对相邻的属性取值 a^i 与 a^{i+1} 来说 , t 在区间 $\left[a^i, a^{i+1}\right)$ 中取任意值所产生的划分结果相同. 因此, 对于连续属性 a ,考察包含 $n-1$ 个元素的候选划分点集合

$$T_a=\left\{\frac{a^i+a^{i+1}}{2} \mid 1 \leqslant i \leqslant n-1\right\} \tag{4.7}$$

即把区间 $\left[a^i, a^{i+1}\right)$ 的中位点 $\frac{a^i+a^{i+1}}{2}$ 作为划分点.

这样, 就可以像离散属性值一样来考察这些划分点, 选取最优的划分点进行样本集合的划分.

如, 对式(4.2)改造可得:

$$\begin{aligned} \operatorname{Gain}(D, a) & =\max _{t \in T_a} \operatorname{Gain}(D, a, t) \\ & =\max _{t \in T_a} \operatorname{Ent}(D)-\sum_{\lambda \in\{-,+\}} \frac{\left|D_t^\lambda\right|}{|D|} \operatorname{Ent}\left(D_t^\lambda\right) \end{aligned} \tag{4.8}$$

其中 $\operatorname{Gain}(D, a, t)$ 是样本集 D 基于划分点 t 二分后的信息增益.于是, 我们就可选择使 $\operatorname{Gain}(D, a, t)$ 最大化的划分点.

> 注意: 与离散属性不同, 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.

4.4.2 缺失值处理

现实任务中常会遇到不完整样本, 即**样本的某些属性值缺失**. 如果简单地放弃不完整样本, 仅使用无缺失值的样本来进行学习, 显然是对数据信息极大的浪费. 显然, 有必要考虑利用有缺失属性值的训练、样例来进行学习.

需解决**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?

(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?

给定训练集 D 和属性 a， 令 $\tilde{D}$ 表示 D 中在属性 a 上没有缺失值的样本子集. **对问题(1)**, 显然我们仅可根据 $\tilde{D}$ 来判断属性 a 的优劣. 假定属性 a 有 V 个可取值 $\left\{a^1, a^2, \ldots, a^V\right\}$, 令 $\tilde{D}^v$ 表示 $\tilde{D}$ 中在属性 a 上取值为 a^v 的样本子集 , $\tilde{D}_k$ 表示 $\tilde{D}$ 中属于第 k 类 $(k=1,2, \ldots,|\mathcal{Y}|)$ 的样本子集， 则显然有 $\tilde{D}=\bigcup_{k=1}^{|\mathcal{Y}|} \tilde{D}_k$, $\tilde{D}=\bigcup_{v=1}^V \tilde{D}^v$, 假定为每个样本 $\boldsymbol{x}$ 赋予一个权重 $w_{\boldsymbol{x}}$,

$$\rho = \frac{\sum_{\boldsymbol{x} \in \tilde{D}} w_{\boldsymbol{x}}}{\sum_{\boldsymbol{x} \in D} w_{\boldsymbol{x}}} \tag{4.9}$$

$$\tilde{p}_k = \frac{\sum_{\boldsymbol{x} \in \tilde{D}_k} w_{\boldsymbol{x}}}{\sum_{\boldsymbol{x} \in \tilde{D}} w_{\boldsymbol{x}}} \quad (1 \leqslant k \leqslant |\mathcal{Y}|) \tag{4.10}$$

$$\tilde{r}_v = \frac{\sum_{\boldsymbol{x} \in \tilde{D}^v} w_{\boldsymbol{x}}}{\sum_{\boldsymbol{x} \in \tilde{D}} w_{\boldsymbol{x}}} \quad (1 \leqslant v \leqslant V) \tag{4.11}$$

参数的含义:

对属性 a， ρ 表示无缺失值样本所占的比例, $\tilde{p}_k$ 表示无缺失值样本中第 k 类所占的比例, $\tilde{r}_v$则表示无缺失值样本中在属性 a 上取值 a^v 的样本所占的比例. 显然， $\sum_{k=1}^{|\mathcal{Y}|} \tilde{p}_k = 1, \sum_{v=1}^{V} \tilde{r}_v = 1$.

基于上述定义, 我们可将信息增益的计算式 (4.2) 推广为:

$$\begin{aligned} \operatorname{Gain}(D, a) &= \rho \times \operatorname{Gain}(\tilde{D}, a) \\ &= \rho \times \left(\operatorname{Ent}(\tilde{D}) - \sum_{v=1}^{V} \tilde{r}_v \operatorname{Ent}\left(\tilde{D}^v\right) \right) \end{aligned} \tag{4.12}$$

其中由 (4.1), 有

$$\operatorname{Ent}(\tilde{D}) = -\sum_{k=1}^{|\mathcal{Y}|} \tilde{p}_k \log_2 \tilde{p}_k$$

> 注: 1. w_x 开始时一般取1(书中的例子)
>
> 2. 为什么要乘上 ρ ? 按说按照(4.9)的理解, 不应该是除以 ρ 吗? 不是很理解. 星球上的一种解释是" 这里有一些样本在属性 a 上是有缺失值的, 所以不同的属性, 有不同的缺失比例, 所以计算增益时乘以 ρ".正不正确?

对问题 (2) , 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， 且样本权值在于结点中保持为 $w_{\boldsymbol{x}}$. 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值 a^v 对应的子结点中调整为 $\tilde{r}_v \cdot w_{\boldsymbol{x}}$. 直观地看, 这就是让同一个样本**以不同的概率划入到不同的子结点中**去.