東莞理工學院
DONGGUAN UNIVERSITY OF TECHNOLOGY

人工智能概论

第四章：决策树

丁烨，计算机科学与技术学院
dingye@dgut.edu.cn

目录

决策树基本原理

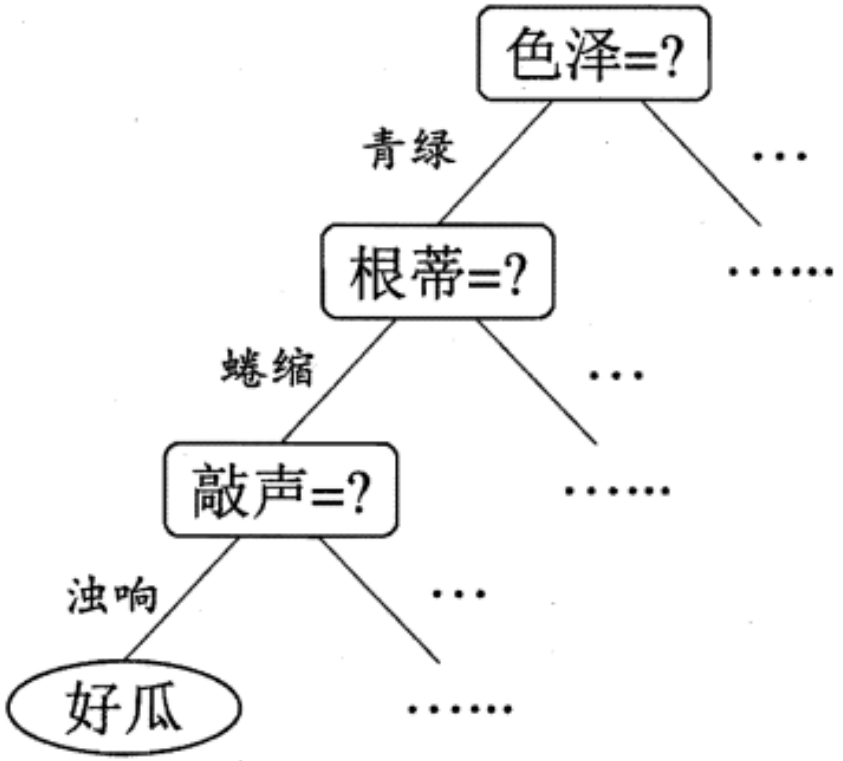
基本概念

- 决策树（Decision Tree）
- 又称判定树，是一个基于树结构进行决策的常见机器学习方法
- 决策树是人类在面临决策问题时一种很自然的处理机制
- 决策树是一个分类算法

决策树基本原理

基本概念

- 主问题：“这是好瓜吗？”
- 子问题：“它是什么颜色？”青绿
- 子问题：“它的根蒂是什么形态？”蜷缩
- 子问题：“它敲起来是什么声音？”浊响
- 最终决策：这是个好瓜



决策树基本原理

基本概念

- 一般的，一棵决策树包含：
- 一个根结点（Root）
- 若干个内部结点
- 若干个叶结点（Leaf）
- 叶结点对应于决策结果，内部结点对应于一个特征测试
- 每个结点包含的样本集合根据特征测试的结果被划分到子结点中
- 从根结点到每个叶结点的路径对应了一个判定测试序列

决策树基本原理

基本概念

- 决策树学习的目的是为了产生一棵决策树模型
- 该模型的泛化能力，即处理未见示例的能力要足够强
- 其基本流程遵循简单且直观的策略:
- “分而治之（Divide-and-conquer）”

决策树基本原理

算法分析

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

- 输入：
- 训练集：
- 特征（属性）的值
- 类别
- 特征（属性）集：
- 特征（属性）的名称

决策树基本原理

算法分析

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

- 生成一个新节点
- 根据训练集的情况来决定节点的类型

决策树基本原理

算法分析

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

- 情况一：当前结点包含的样本全属于同一类别
- 生成一个叶节点并标识为这个类别
- 今后当判定测试序列走到这个节点时：
- 所有验证样本都会被判定为该类别

决策树基本原理

算法分析

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

- 情况二：当前特征集为空，或所有样本在所有特征上取值相同
- 则生成一个叶节点
- 其取值为训练集中样本数最多的类别

决策树基本原理

算法分析

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 [illegible]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

- 情况三：正常情况
- 找到一个最优划分特征
- 生成一个分支：
- 如果符合该特征的样本为空，则视为情况二
- 否则，在这个特征上展开分支，持续递归，直到其符合前两种情况

决策树基本原理

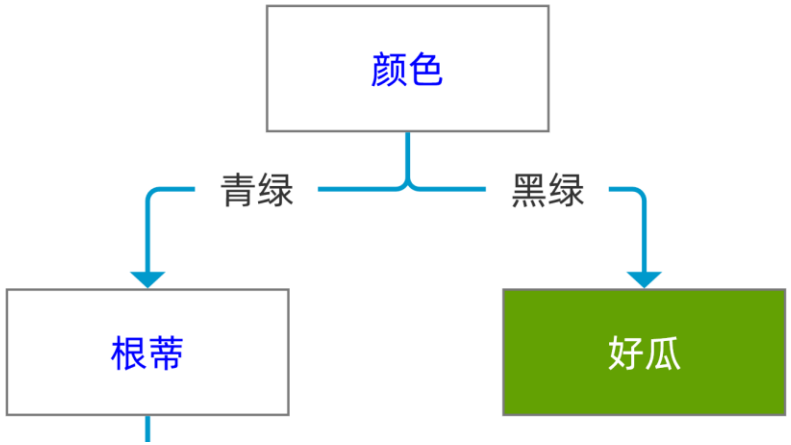
算法分析

- 例如，我们获取了这些样本数据：
- 青绿、蜷缩、浊响：好瓜
- 青绿、坚硬、清脆：坏瓜
- 青绿、蜷缩、清脆：坏瓜
- 黑绿、坚硬、浊响：好瓜
- 对应的特征集为：颜色、根蒂、声音

决策树基本原理

算法分析

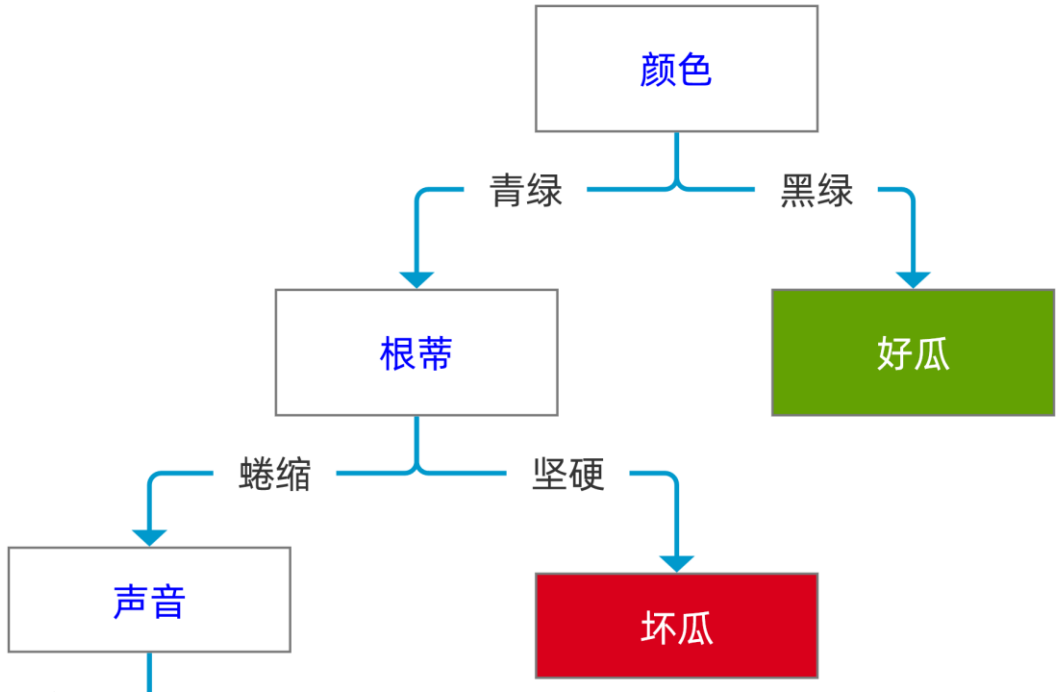
- 青绿、蜷缩、浊响：好瓜
- 青绿、坚硬、清脆：坏瓜
- 青绿、蜷缩、清脆：坏瓜
- 黑绿、坚硬、浊响：好瓜



决策树基本原理

算法分析

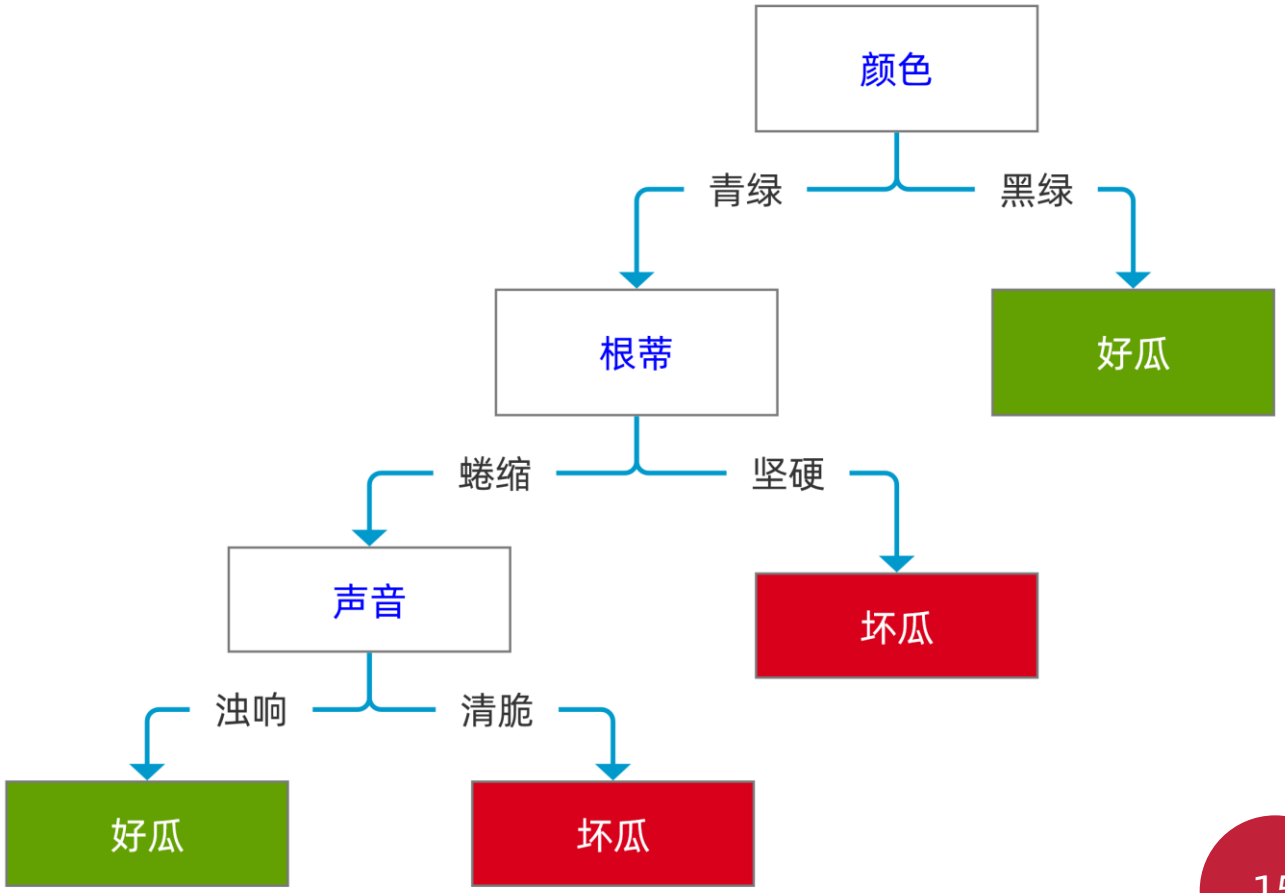
- 青绿、蜷缩、浊响：好瓜
- 青绿、坚硬、清脆：坏瓜
- 青绿、蜷缩、清脆：坏瓜
- 黑绿、坚硬、浊响：好瓜



决策树基本原理

算法分析

- 青绿、蜷缩、浊响：好瓜
- 青绿、坚硬、清脆：坏瓜
- 青绿、蜷缩、清脆：坏瓜
- 黑绿、坚硬、浊响：好瓜



决策树基本原理

算法分析

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 = ∅ OR D 中样本在 A 上[illegible]同 then
6:     将 node 标记为叶结点, 其类[illegible]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

- 核心问题：
- 如何确定这个“最优化分特征（属性）”？
- 在刚才的例子里
- 我们是按照特征集的自然顺序选择的

决策树基本原理

划分选择

- 如何选择最优划分特征？
- 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

决策树基本原理

划分选择

- 常见的划分选择包括：
- 信息增益（Information Gain）
- 增益率（Gain Ratio）
- 基尼指数（Gini Index）

决策树基本原理

信息增益

- 信息熵（Information Entropy）
- 度量样本集合纯度最常用的一种指标
- 假定当前样本集合 D 中第 k 类样本所占的比例为：p_k
- 则 D 的信息熵定义为：

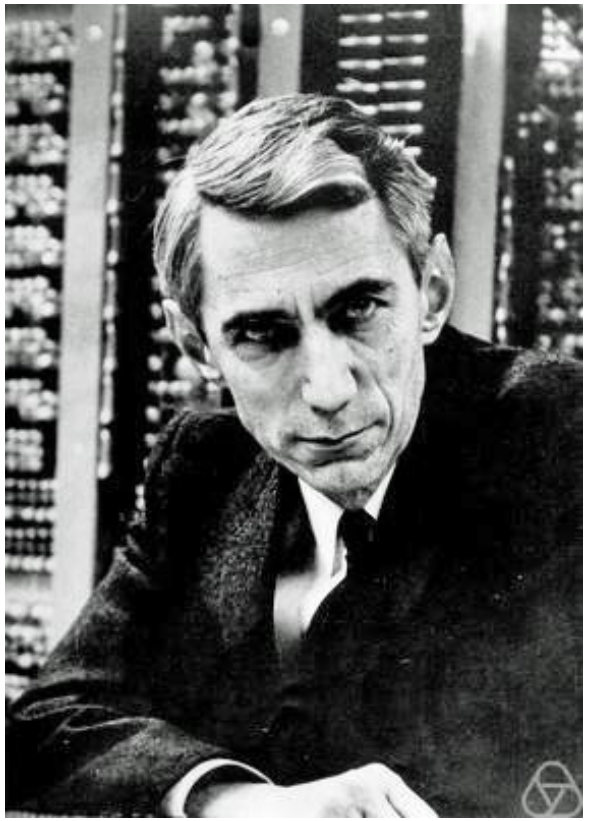
$$H(D) = -\sum_{k=1}^{|y|} p_k \log_2 p_k$$

- H 的值越小，则 D 的纯度越高

决策树基本原理

信息增益

- 香农（Claude Shannon）
- 美国数学家、电子工程师和密码学家
- 被誉为信息论的创始人
- 1948 年，香农发表了划时代的论文：
- 《通信的数学理论》
- 奠定了现代信息论的基础



决策树基本原理

信息增益

- 假定离散特征 a 有 V 个可能的取值 $\{a^1, a^2, \cdots, a^V\}$
- 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 中所有在特征 a 上取值为 a^v 的样本
- 记为 D^v

决策树基本原理

信息增益

- 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
- 于是可计算出用特征 a 对样本集 D 进行划分所获得的
- “信息增益（Information Gain）”

$$\mathrm{Gain}(D,\mathrm{a}) = H(D) - \sum_{v=1}^{V} \frac{|D^v|}{|D|} H(D^v)$$

决策树基本原理

信息增益

- 一般而言，信息增益越大
- 则意味着使用特征 a 来进行划分所获得的“纯度提升”越大
- 因此，我们可用信息增益来进行决策树的划分特征选择
- 著名的 ID3 决策树学习算法（Quinlan，1986）
- 就是以信息增益为准则来选择划分特征

决策树基本原理

信息增益

表 4.1　西瓜数据集 2.0

编号	色泽	根蒂	敲声	纹理	脐部	触感	好瓜
1	青绿	蜷缩	浊响	清晰	凹陷	硬滑	是
2	乌黑	蜷缩	沉闷	清晰	凹陷	硬滑	是
3	乌黑	蜷缩	浊响	清晰	凹陷	硬滑	是
4	青绿	蜷缩	沉闷	清晰	凹陷	硬滑	是
5	浅白	蜷缩	浊响	清晰	凹陷	硬滑	是
6	青绿	稍蜷	浊响	清晰	稍凹	软粘	是
7	乌黑	稍蜷	浊响	稍糊	稍凹	软粘	是
8	乌黑	稍蜷	浊响	清晰	稍凹	硬滑	是
9	乌黑	稍蜷	沉闷	稍糊	稍凹	硬滑	否
10	青绿	硬挺	清脆	清晰	平坦	软粘	否
11	浅白	硬挺	清脆	模糊	平坦	硬滑	否
12	浅白	蜷缩	浊响	模糊	平坦	软粘	否
13	青绿	稍蜷	浊响	稍糊	凹陷	硬滑	否
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	否
15	乌黑	稍蜷	浊响	清晰	稍凹	软粘	否
16	浅白	蜷缩	浊响	模糊	平坦	硬滑	否
17	青绿	蜷缩	沉闷	稍糊	稍凹	硬滑	否

决策树基本原理

信息增益

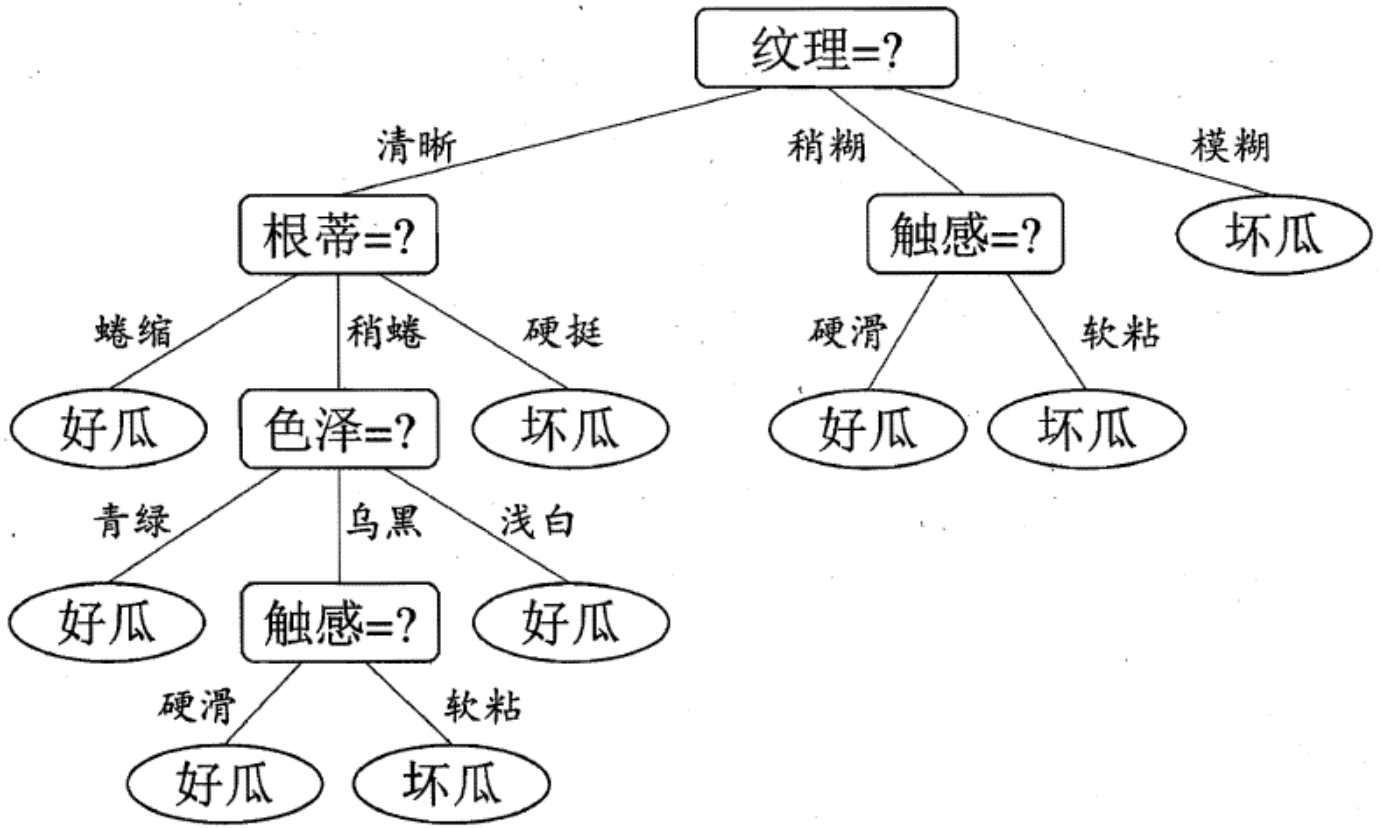


图 4.4 在西瓜数据集 2.0 上基于信息增益生成的决策树

决策树基本原理

增益率

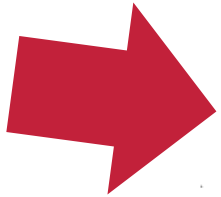


表 4.1　西瓜数据集 2.0

编号	色泽	根蒂	敲声	纹理	脐部	触感	好瓜
1	青绿	蜷缩	浊响	清晰	凹陷	硬滑	是
2	乌黑	蜷缩	沉闷	清晰	凹陷	硬滑	是
3	乌黑	蜷缩	浊响	清晰	凹陷	硬滑	是
4	青绿	蜷缩	沉闷	清晰	凹陷	硬滑	是
5	浅白	蜷缩	浊响	清晰	凹陷	硬滑	是
6	青绿	稍蜷	浊响	清晰	稍凹	软粘	是
7	乌黑	稍蜷	浊响	稍糊	稍凹	软粘	是
8	乌黑	稍蜷	浊响	清晰	稍凹	硬滑	是
9	乌黑	稍蜷	沉闷	稍糊	稍凹	硬滑	否
10	青绿	硬挺	清脆	清晰	平坦	软粘	否
11	浅白	硬挺	清脆	模糊	平坦	硬滑	否
12	浅白	蜷缩	浊响	模糊	平坦	软粘	否
13	青绿	稍蜷	浊响	稍糊	凹陷	硬滑	否
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	否
15	乌黑	稍蜷	浊响	清晰	稍凹	软粘	否
16	浅白	蜷缩	浊响	模糊	平坦	硬滑	否
17	青绿	蜷缩	沉闷	稍糊	稍凹	硬滑	否

决策树基本原理

增益率

- 在上面的例子中，我们有意忽略了表 4.1 中的“编号”这一列
- 若把“编号”也作为一个候选划分特征
- 则可计算出它的信息增益为：0.998
- 远大于其他候选划分特征
- 这很容易理解，“编号”将产生 17 个分支
- 每个分支结点仅包含一个样本，这些分支结点的纯度已达最大
- 然而，这样的决策树显然不具有泛化能力
- 无法对新样本进行有效预测

决策树基本原理

增益率

- 实际上，信息增益准则对可取值数目较多的特征有所偏好
- 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（Quinlan，1993）不直接使用信息增益
- 而是使用“增益率（Gain Ratio）”来选择最优划分特征

$$\text{GainRatio}(D,a) = \frac{\text{Gain}(D,a)}{\text{IV}(a)}$$

决策树基本原理

增益率

- 其中：

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

- 称为特征 a 的“固有值（Intrinsic Value）”
- 特征 a 的可能取值数目越多（即 V 越大）
- 则 $\mathrm{IV}(a)$ 的值通常也会越大

决策树基本原理

增益率

- 增益率准则对可取值数目较少的特征有所偏好
- 因此，C4.5 算法并不是直接选择增益率最大的候选划分特征
- 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特征
- 再从中选择增益率最高的

决策树基本原理

基尼指数

- 基尼指数（Gini Index）
- CART 决策树（Breiman，1984）使用的一种划分选择

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

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

目录

优化决策树算法

剪枝处理

- 剪枝（Pruning）
- 决策树算法对付“过拟合”的主要手段
- 在决策树中，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
- 这时就可能因训练样本学得“太好”了
- 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
- 因此，可通过主动去掉一些分支来降低过拟合的风险

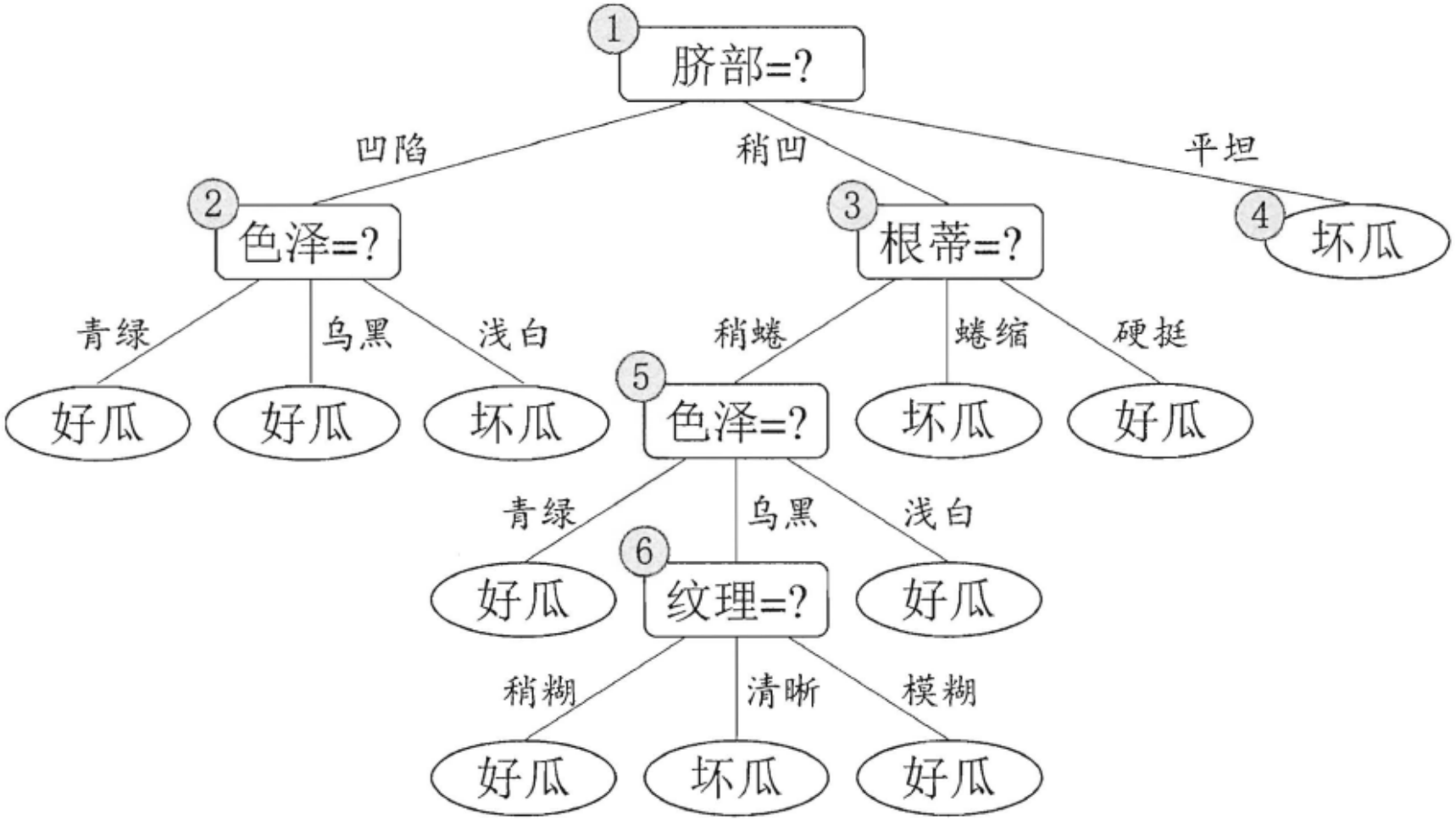
优化决策树算法

剪枝处理

- 预剪枝（Pre-pruning）
- 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

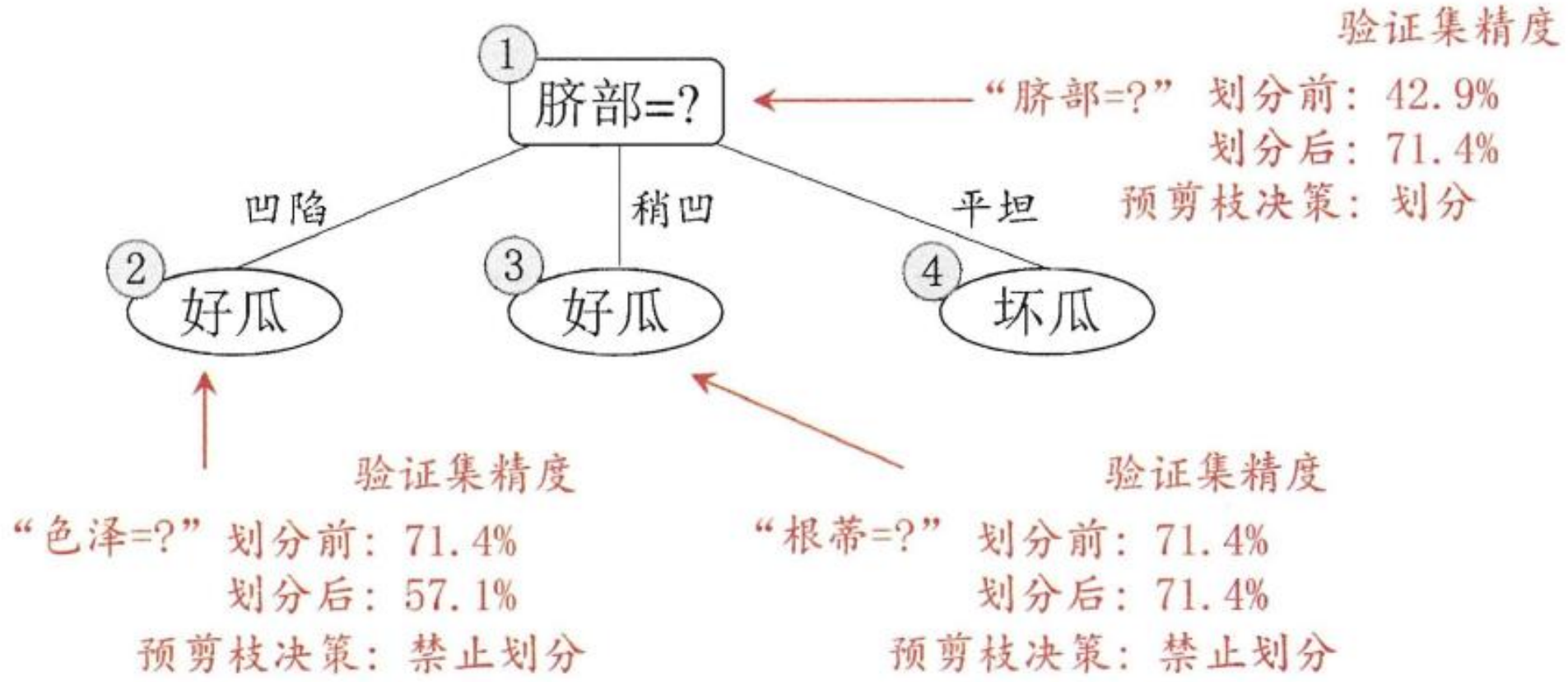
优化决策树算法

剪枝处理



优化决策树算法

剪枝处理



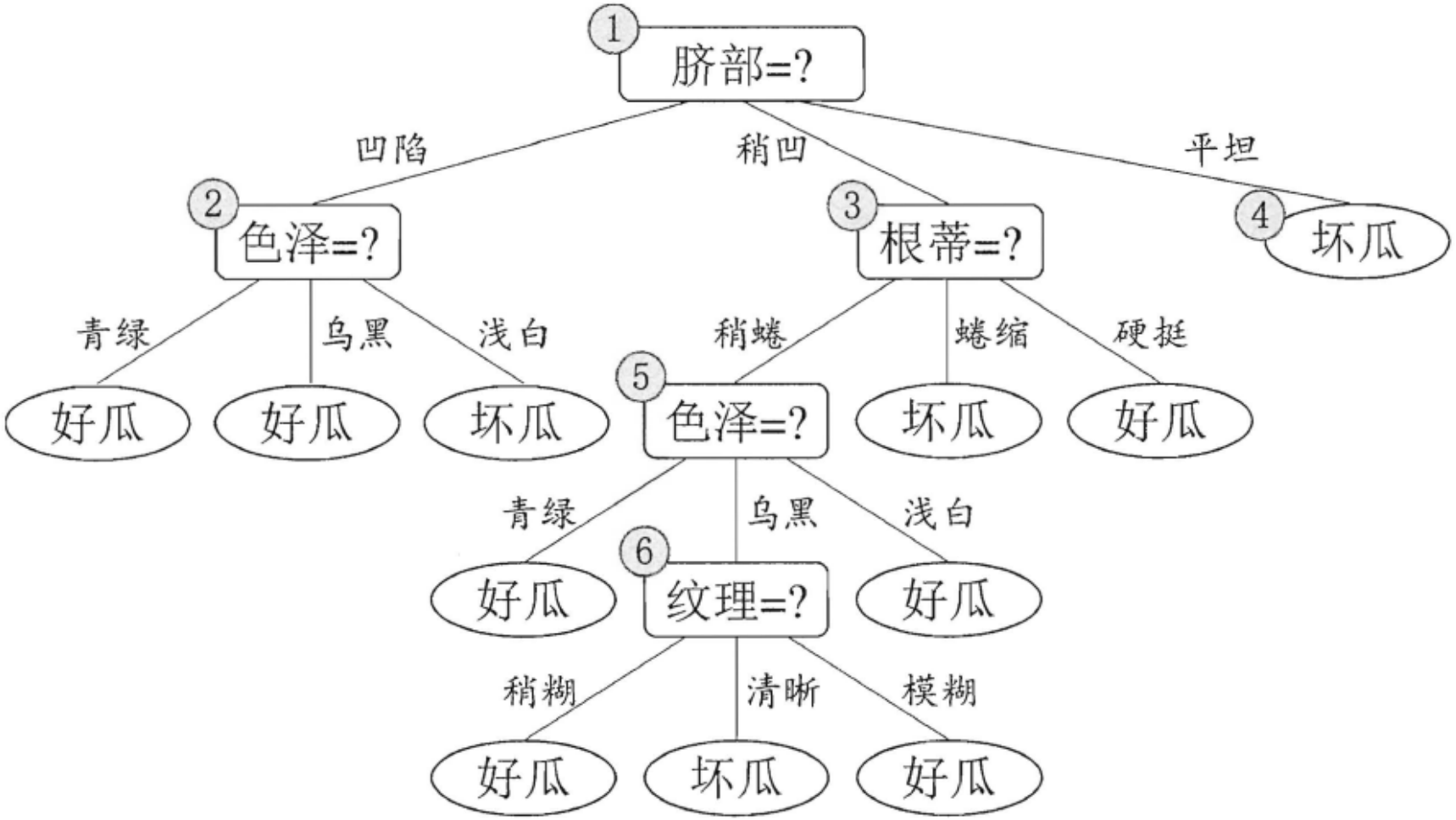
优化决策树算法

剪枝处理

- 后剪枝（Post-pruning）
- 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

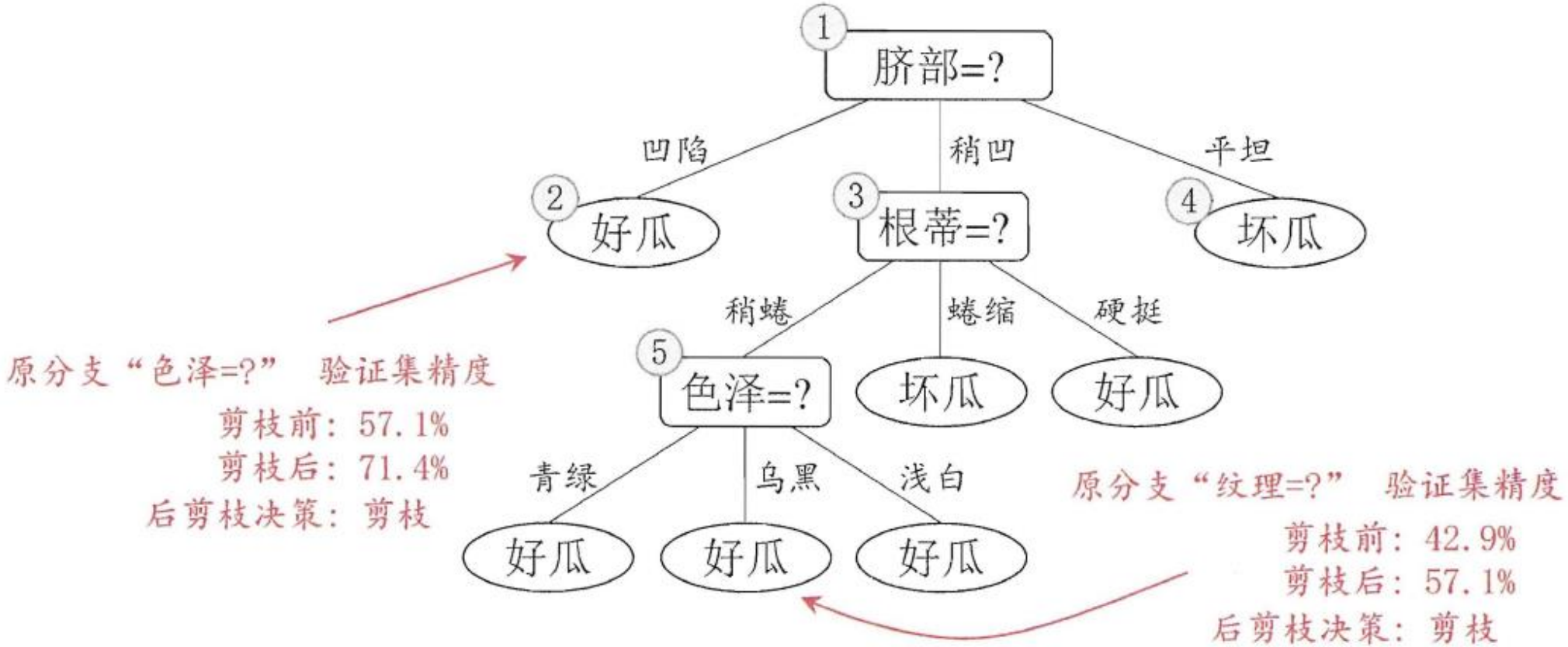
优化决策树算法

剪枝处理



优化决策树算法

剪枝处理



优化决策树算法

剪枝处理

- 预剪枝使得决策树的很多分支都没有“展开”
- 给决策树带来了欠拟合的风险
- 后剪枝决策树通常比预剪枝决策树保留了更多的分支
- 欠拟合风险很小，泛化性能往往优于预剪枝决策树
- 但后剪枝过程是在生成完全决策树之后进行的
- 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

优化决策树算法

连续与缺失值

- 连续值处理
- 到目前为止我们仅讨论了基于离散特征来生成决策树
- 而连续特征的可取值数目不再有限，不能直接进行划分
- 此时离散化技术可派上用场

优化决策树算法

连续与缺失值

- 决策树中最简单的策略是：二分法（Bi-partition）
- 这正是 C4.5 决策树算法中采用的机制（Quinlan，1993）
- 二分法把区间的中位点作为候选划分点：

$$T_a = \left\{ \frac{a^i + a^{i+1}}{2} \middle| 1 \leq i \leq n-1 \right\}$$

优化决策树算法

连续与缺失值

编号	色泽	根蒂	敲声	纹理	脐部	触感	密度	含糖率	好瓜
1	青绿	蜷缩	浊响	清晰	凹陷	硬滑	0.697	0.460	是
2	乌黑	蜷缩	沉闷	清晰	凹陷	硬滑	0.774	0.376	是
3	乌黑	蜷缩	浊响	清晰	凹陷	硬滑	0.634	0.264	是
4	青绿	蜷缩	沉闷	清晰	凹陷	硬滑	0.608	0.318	是
5	浅白	蜷缩	浊响	清晰	凹陷	硬滑	0.556	0.215	是
6	青绿	稍蜷	浊响	清晰	稍凹	软粘	0.403	0.237	是
7	乌黑	稍蜷	浊响	稍糊	稍凹	软粘	0.481	0.149	是
8	乌黑	稍蜷	浊响	清晰	稍凹	硬滑	0.437	0.211	是
9	乌黑	稍蜷	沉闷	稍糊	稍凹	硬滑	0.666	0.091	否
10	青绿	硬挺	清脆	清晰	平坦	软粘	0.243	0.267	否
11	浅白	硬挺	清脆	模糊	平坦	硬滑	0.245	0.057	否
12	浅白	蜷缩	浊响	模糊	平坦	软粘	0.343	0.099	否
13	青绿	稍蜷	浊响	稍糊	凹陷	硬滑	0.639	0.161	否
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	0.657	0.198	否
15	乌黑	稍蜷	浊响	清晰	稍凹	软粘	0.360	0.370	否
16	浅白	蜷缩	浊响	模糊	平坦	硬滑	0.593	0.042	否
17	青绿	蜷缩	沉闷	稍糊	稍凹	硬滑	0.719	0.103	否

优化决策树算法

连续与缺失值

编号	色泽	根蒂	敲声	纹理	脐部	触感	密度	含糖率	好瓜
1	青绿	蜷缩	浊响	清晰	凹陷		0.697	0.460	是
2	乌黑	蜷缩	沉闷		凹陷		0.774	0.376	是
3	乌黑	蜷缩	浊响		陷	硬滑	0.634	0.264	是
4	青绿	蜷缩	沉闷			硬滑	0.608	0.318	是
5	浅白	蜷缩	浊			硬滑	0.556	0.215	是
6	青绿	稍蜷	浊			软粘	0.403	0.237	是
7	乌黑	稍蜷	浊响		凹	软粘	0.481	0.149	是
8	乌黑	稍蜷	浊响		稍凹	硬滑	0.437	0.211	是
9	乌黑	稍蜷	沉闷	稍糊	稍凹	硬滑	0.666	0.091	否
10	青绿	硬挺	清脆	清晰	平坦	软粘	0.243	0.267	否
11	浅白	硬挺	清脆	模糊	平坦	硬滑	0.245	0.057	否
12	浅白	蜷缩	浊响	模糊	平坦	软粘	0.343	0.099	否
13	青绿	稍蜷	浊响	稍糊	凹陷	硬滑	0.639	0.161	否
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	0.657	0.198	否
15	乌黑	稍蜷	浊响	清晰	稍凹	软粘	0.360	0.370	否
16	浅白	蜷缩	浊响	模糊	平坦	硬滑	0.593	0.042	否
17	青绿	蜷缩	沉闷	稍糊	稍凹	硬滑	0.719	0.103	否

0.381

优化决策树算法

连续与缺失值

- 缺失值处理
- 现实任务中常会遇到不完整样本
- 例如由于诊测成本、隐私保护等因素，患者的医疗数据在某些特征上的取值（如 HIV 测试结果）未知
- 尤其是在特征数目较多的情况下，往往会有大量样本出现缺失值

表 4.4　西瓜数据集 2.0α

编号	色泽	根蒂	敲声	纹理	脐部	触感	好瓜
1	–	蜷缩	浊响	清晰	凹陷	硬滑	是
2	乌黑	蜷缩	沉闷	清晰	凹陷	–	是
3	乌黑	蜷缩	–	清晰	凹陷	硬滑	是
4	青绿	蜷缩	沉闷	清晰	凹陷	硬滑	是
5	–	蜷缩	浊响	清晰	凹陷	硬滑	是
6	青绿	稍蜷	浊响	清晰	–	软粘	是
7	乌黑	稍蜷	浊响	稍糊	稍凹	软粘	是
8	乌黑	稍蜷	浊响	–	稍凹	硬滑	是
9	乌黑	–	沉闷	稍糊	稍凹	硬滑	否
10	青绿	硬挺	清脆	–	平坦	软粘	否
11	浅白	硬挺	清脆	模糊	平坦	–	否
12	浅白	蜷缩	–	模糊	平坦	软粘	否
13	–	稍蜷	浊响	稍糊	凹陷	硬滑	否
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	否
15	乌黑	稍蜷	浊响	清晰	–	软粘	否
16	浅白	蜷缩	浊响	模糊	平坦	硬滑	否
17	青绿	–	沉闷	稍糊	稍凹	硬滑	否

优化决策树算法

连续与缺失值

- 如何在特征值缺失的情况进行划分特征选择？
- 给定划分特征，若样本在该特征上的值缺失，如何对样本进行划分？
- 解决两个问题的基本思想为：样本赋权，权重划分

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

目录

决策树算法进阶

多变量决策树

- 若我们把每个特征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了 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
- 决策树所形成的分类边界有一个明显的特点:
- 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

决策树算法进阶

多变量决策树

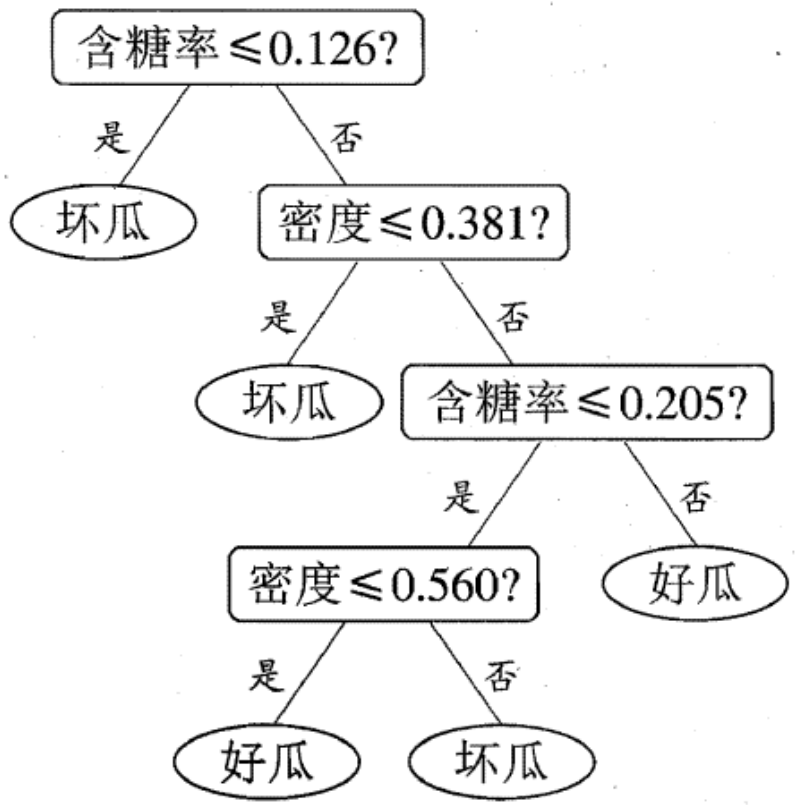


图 4.10 在西瓜数据集 3.0α 上生成的决策树

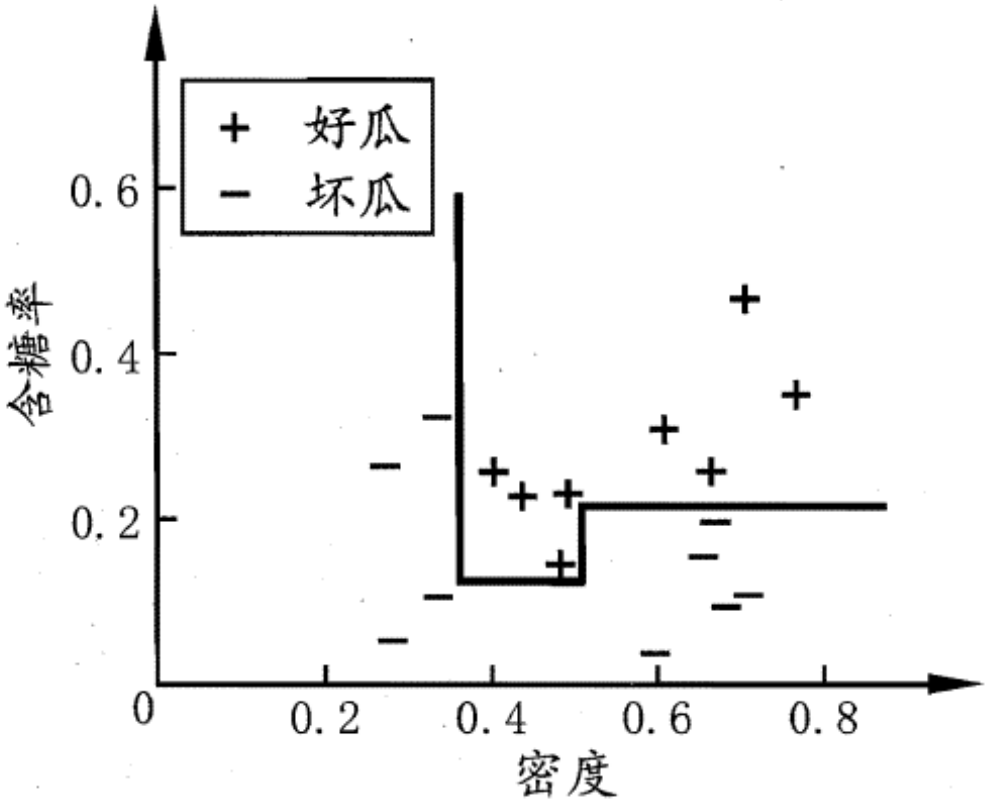


图 4.11 图 4.10 决策树对应的分类边界

决策树算法进阶

多变量决策树

- 传统的单变量决策树
- 优点：分类边界有较好的解释性，每一段划分都对应某个特征取值
- 缺点：学习任务的真实分类边界比较复杂时，需要多段划分才能获得较好的模型，训练过程较长

决策树算法进阶

多变量决策树

- 多变量决策树（Multivariate Decision Tree）
- 非叶结点不再是某个特征，而是特征的线性组合
- 分类边界不必与坐标轴平行，实现“斜划分”甚至更复杂的划分
- 更复杂的“混合决策树”甚至可以嵌入神经网络或其他非线性模型

决策树算法进阶

多变量决策树

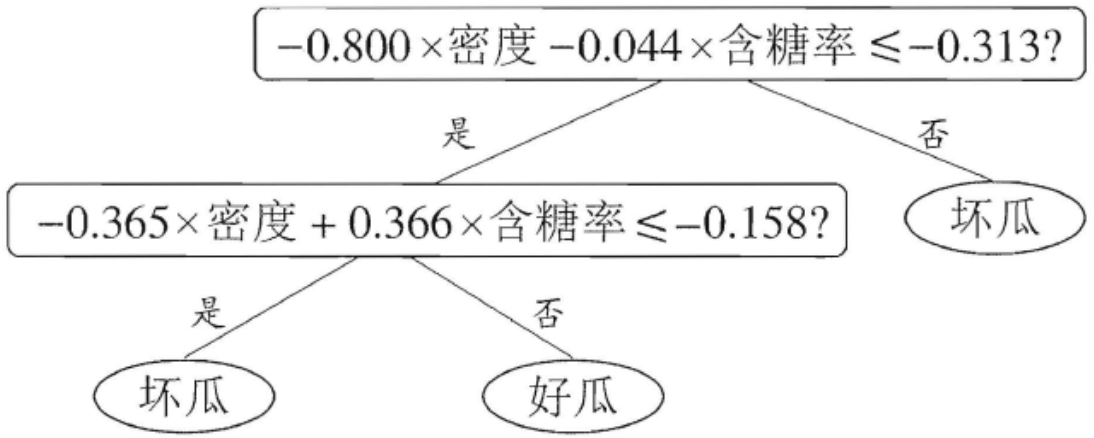


图 4.13 在西瓜数据集 3.0α 上生成的多变量决策树

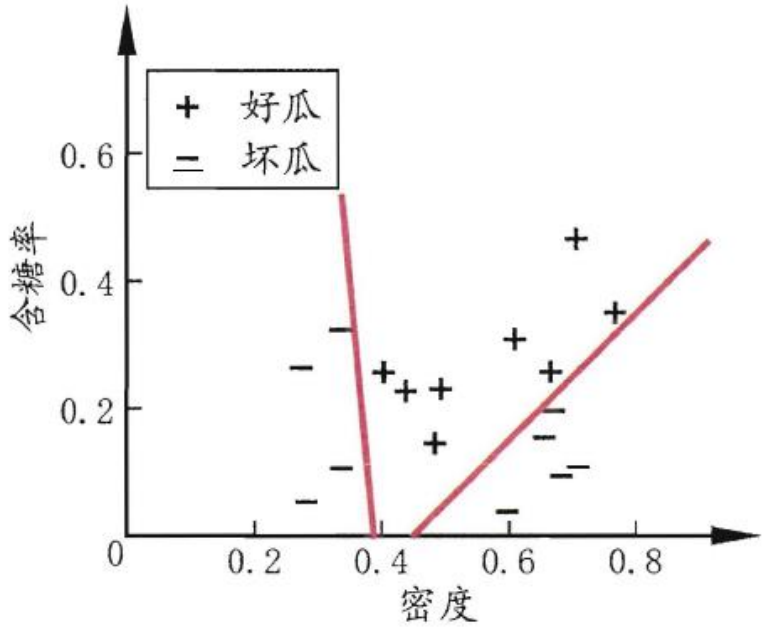


图 4.14 图 4.13 多变量决策树对应的分类边界

总结

- 「西瓜书」周志华《机器学习》，清华大学出版社
- https://item.jd.com/12762673.html

- 「南瓜书」谢文睿、秦州《机器学习公式详解》，人民邮电出版社
- https://github.com/datawhalechina/pumpkin-book/

总结

- Scikit-Learn
- https://scikit-learn.org

- TensorFlow
- https://www.tensorflow.org

東莞理工學院
DONGGUAN UNIVERSITY OF TECHNOLOGY

Thank You!

丁烨，计算机科学与技术学院
dingye@dgut.edu.cn