第四章 决策树

1. **基本流程**
   - 决策树的生成是一个递归过程，在决策树基本算法中，有三种情况会导致递归返回：（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，或者所有样本在所有属性上取值相同，无法划分；（3）当前节点包含的的样本集合为空，不能划分。
   - 在第（2）中情况下，我们将当前结点标记成叶结点，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节点所含样本最多的类别。注意：以上两种情形的处理实质上不同：（2）是利用当前结点的后验分布，（3）是利用父节点的样本分布作为当前结点的先验分布
2. **划分选择**

   2.1 信息增益

   - 信息熵是度量样本集合纯度最常用的一种指标。假设当前样本集合D中第k类样本所占比例为$p_k(k=1,2,\cdots,|\mathcal{Y}|)$，则D的信息熵定义为：

     $$Ent(D)=-\sum_{i=1}^{|\mathcal{Y}|}p_k\log_2 p_k$$

     Ent(D)的的值越小，则D的纯度越高。熵表达的是不确定性，因此越小，不确定性越低，纯度越高。
   - 假设离散数学a有V个可能的取值$\{a^1,a^2,\cdots,a^V\}$，若使用属性a来对样本集D进行划分，则会产生V个结点，其中第v个结点包含了D中所有在属性a熵取值为a^v的样本，记为D^v，因此我们可以计算出D^v的信息熵，再考虑到不同分支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进行划分得到的信息增益:

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

     一般来说，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，我们可用信息增益来进行决策树的划分属性选择。著名的ID3决策树学习算法就是以信息增益为准则来选择划分属性的。

   2.2 增益率

   - 事实上，信息增益对取值数目较多的属性有所偏好，例如：ID属性，每个样本都有唯一ID，通过信息增益会将其划分为每个样本一个分支，纯度提升最大，但没什么实际意义。
   - 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，而是使用增益率来选择最有划分属性。增益率定义为：

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

     其中

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

     称为属性a的固有值，属性a可能取值数目越多，IV(a)的值通常会越大。
   - 须注意的是，增益率对取值数目少的属性有所偏好，因此，C4.5算法并不是直接选择增益率最大的候选划分属性，而是使用了一个启发式：先从侯选属性中找到信息增益高于平均水平的属性，再从中选择增益率最高的。

2.3 基尼指数

- CART决策树使用“基尼指数”来选择划分属性。数据集D的纯度可用基尼值来度量

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

- 直观上来说，Gini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越小，则数据集的纯度越高。
- 属性a的基尼值数定义为：

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

- 于是在侯选属性集合中，选择那个使得划分候基尼值数最小的属性作用最优划分属性

3. **剪枝处理**

- 剪枝是决策树学习算法对付过拟合的主要方法。在决策树学习中，为了尽可能正确分类训练样本，结合划分过程不断重复，有时会造成决策树分支过多，即训练样本学的太好了，以至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，可通过主动去除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”、“后剪枝“。预剪枝是在决策树生成过程中，对每个结点在划分前先进行评估，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；后剪枝则是先从训练集生成一颗完整的决策树，然后自底向上的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，则将该子树替换成叶结点。

3.1 预剪枝

- 预剪枝使得决策树的很多分支没有展开，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，甚至可能带来泛化性能暂时降低，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。

3.2 后剪枝

- 后剪枝决策树通常比预剪枝决策树保留了更多的分支。一般情况下，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。但后剪枝过程是在完全生成决策树之后进行的，并且要自底向上的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大得多

4. **连续与缺失值**

4.1 连续值处理

- 由于连续属性的可取值数目不再有限，因此， 不能直接根据连续属性的可取值来对结点进行划分。此时，连续属性离散化技术可派上用场。最简单的策略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n个不同的取值，将这些值从小到大进行排序，记为：$a^1, a^2, \cdots, a^n$。基于划分点t可将D划分为子集D_t^-和D_t^+，其中D_t^-包含在那些在属性a上取值不大于t的样本，而D_t^+则包含那些在属性a上取值大于t的样本。显然，对于相邻的属性取值a^i与a^{i+1}来说，t在区间$[a^i, a^{i+1})$中取任意值所产生的划分结果相同。因此，对连续属性a，我们可考察包含n-1个元素的候选划分点集合

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

即把区间$[a^i, a^{i+1})$的中位点$\frac{a^i+a^{i+1}}{2}$作为候选划分点，然后我们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，例如：

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

其中$Gain(D,a,t)$是样本集D基于划分点t二分之后的信息增益。于是，我们就可选择使$Gain(D,a,t)$最大的划分点，

- 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。

4.2 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，仅使用无缺失的样本来进行学习，显然是对数据信息极大的浪费，有必要考虑利用有缺失属性值的训练样例来进行学习。
- 我们需解决两个问题：（1）如何在属性值缺失的情况下进行属性划分？（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中属性a上没有缺失值的样本子集。对于问题（1）,显然我们尽可根据$\tilde{D}$来判断属性a的优劣。假定属性a有V个可取值$\{a^1, a^2, \cdots, a^V\}$，令$\tilde{D}^v$表示$\tilde{D}$中在属性a上取值为$a^v$的样本子集，令$\tilde{D}_k$表示$\tilde{D}$中属于第k类$(k = 1, 2, \cdots, |\mathcal{Y}|)$的样本子集，则显然有$\tilde{D} = \bigcup_{k=1}^{|\mathcal{Y}|} \tilde{D}_k$，$\tilde{D} = \bigcup_{v=1}^{V} \tilde{D}^v$，假设我们为每个样本$x$赋予一个权重$\omega_x$，并定义

$$\begin{aligned} \rho &= \frac{\sum_{x \in \tilde{D}} \omega_x}{\sum_{x \in D} \omega_x} \\ \tilde{p}_k &= \frac{\sum_{x \in \tilde{D}_k} \omega_x}{\sum_{x \in \tilde{D}} \omega_x} \ (1 \leq k \leq |\mathcal{Y}|) \\ \tilde{r}_v &= \frac{\sum_{x \in \tilde{D}^v} \omega_x}{\sum_{x \in \tilde{D}} \omega_x} \ (1 \leq v \leq V) \end{aligned}$$

直观来说，对于属性a，ρ表示无缺失样本所占比例，$\tilde{p}_k$表示无缺失样本中第k类所占比例，$\tilde{r}_v$表示无缺失样本中在属性a上取值为a^v的样本所占比例。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，我们可将信息增益计算式推广为：

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

其中，熵定义为：

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

- 对于问题（2），若样本x在划分属性a上的取值已知，则将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ω_x。若样本x在划分属性a上的取值未知，则将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$\tilde{r}_v \cdot \omega_x$。直观的看，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。
- C4.5算法使用了上述解决方案。

5. **多变量决策树**

- 若我们把每个属性视为坐标空间中的一个坐标轴，则d个属性描述的样本的就对应了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，即它的分类边界由若干个与坐标轴平行的分段组成。

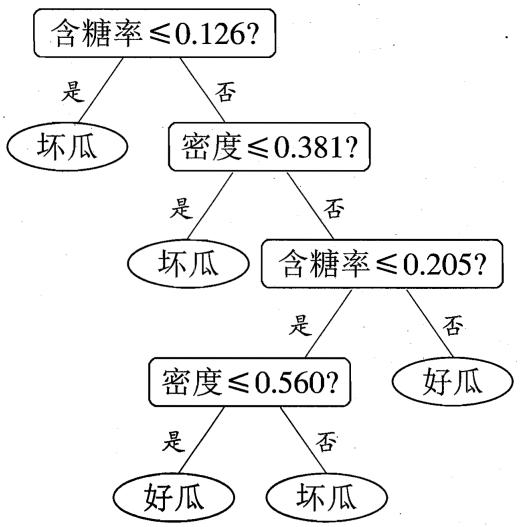

图 4.10 在西瓜数据集 3.0α 上生成的决策树

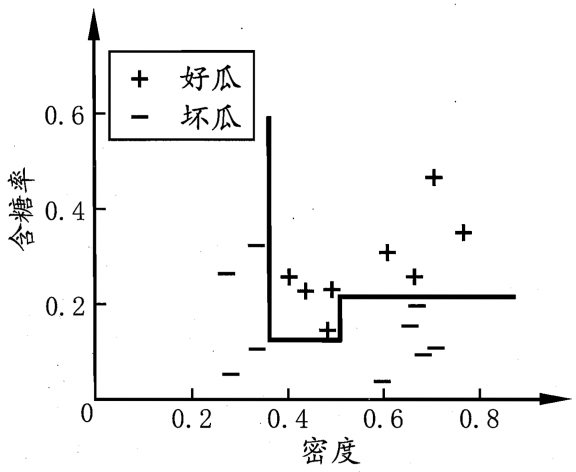

图 4.11 图 4.10 决策树对应的分类边界

- 显然，分类边界的每一段都是与坐标作平行的，这样的分类边界使得学习结果有较好的可解释性，因为每一段划分都直接对应了某个属性取值。但在这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。
  - 若使用斜的划分边界，则决策树模型将大为简化。"多变量决策时"就是能实现这样的"斜划分"甚至更复杂划分的决策树。以实现斜划分的多变量决策树为例，非叶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，每个非叶结点是一个形如$\sum_{i=1}^{d} \omega_i a_i = t$的线性分类器，其中$\omega_i$是属性$a_i$的权重，$\omega_i$和t可在该结点所含的样本集和属性集上学的。
  - 于是，与传统的单变量决策树不同，多变量决策树的学习过程中，不是为每个非叶结点寻找一个最优划分属性，而是试图建立一个合适的线性分类器。
  - 针对与上图决策树相同的数据集西瓜数据3.0，可学习到以下多变量决策树：

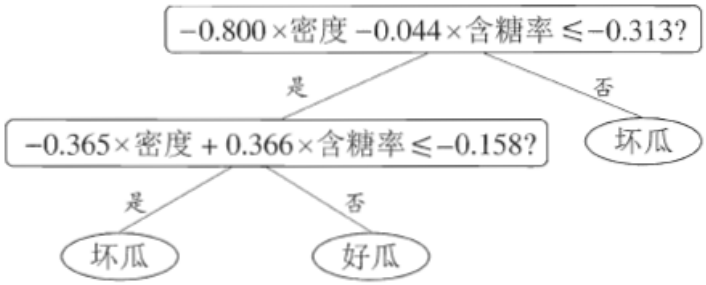

图 4.13 在西瓜数据集 3.0α 上生成的多变量决策树

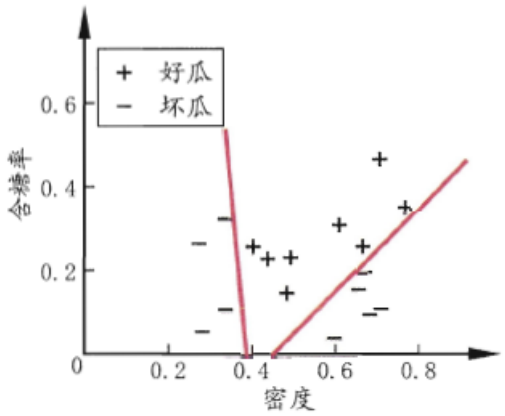

图 4.14 图 4.13 多变量决策树对应的分类边界