## 4、决策树（Decision Tree）

-决策树（decision tree）是一种基本的分类与回归方法。
-决策树模型呈树形结构，在分类问题中，表示基于特征对实例进行分类的过程。它可以认为是 if-then 规则的集合，也可认为是定义在特征空间与类空间上的条件概率分布。
--损失函数：正则化的极大似然函数，策略：最小化损失函数
-优点：可读性、分类速度快
-基本流程：
--学习时，利用训练数据，根据损失函数最小化的原则建立决策树模型。
--预测时，对新的数据，利用决策树模型进行分类。
--决策树学习通常包括三个步骤：特征选择、决策树生成、决策树修剪。

本章概要

1.分类决策树模型是表示特征对实例进行分类的树形结构。决策树可以转换为一个 if-then 规则的集合，也可以看作是定义在特征空间划分上的类的条件概率分布。

2.·决策树旨在构建一个与训练数据拟合很好，并且复杂度小的决策树。因为从可能的决策树中直接选取最优决策树是 NP 完全问题。现实中采用启发式学习次优的决策树。决策树算法包括 3 部分：特征选择、树的生成和树的剪枝。常用的算法有 ID3、C4.5 和 CART。

3．特征选择的目的在于选取对训练数据能够分类的特征。特征选择的关键是其准则。常用的准则如下：

（1）样本集合 D 对特征 A 的信息增益（ID3）

特征 A 对训练数据集 D 的信息增益 g(D,A)，定义为集合 D 的经验熵 H(D)与特征 A 给定条件下 D 的经验条件熵 H(D|A)之差，即

$$g(D,A)=H(D)-H(D\mid A)$$
$$H(D)=-\sum_{k=1}^{K}\frac{|C_k|}{|D|}\log_2\frac{|C_k|}{|D|}$$
$$H(D\mid A)=\sum_{i=1}^{n}\frac{|D_i|}{|D|}H(D_i)$$

其中，H(D)是数据集 D 的熵，H(Di)是数据集 Di 的熵，H(D|A)是数据集 D 对特征 A 的条件熵。 Di 是 D 中特征 A 取第 i 个值的样本子集，Ck 是 D 中属于第 k 类的样本子集。n 是特征 A 取值的个数，K 是类的个数。

（2）样本集合 D 对特征 A 的信息增益比（C4.5）

特征 A 对训练数据集 D 的信息增益比 gR(D,A)定义为其信息增益 g(D,A)与训练数据集 D 的经验熵 H(D)之比

$$g_R(D,A)=\frac{g(D,A)}{H(D)}$$

其中，g(D,A)是信息增益，H(D)是数据集 D 的熵。

（3）样本集合 D 的基尼指数（CART）

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

基尼指数 Gini(D)表示集合 D 的不确定性，基尼指数 Gini(D,A)表示经 A = a 分割后集合 D 的不确定性。

特征 A 条件下集合 D 的基尼指数：

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

4．决策树的生成。通常使用信息增益最大、信息增益比最大或基尼指数最小作为特征选择的准则。决策树的生成往往通过计算信息增益或其他指标，从根结点开始，递归地产生决策树。这相当于用信息增益或其他准则不断地选取局部最优的特征，或将训练集分割为能够基本正确分类的子集。

5．决策树的剪枝。由于生成的决策树存在过拟合问题，需要对它进行剪枝，以简化学到的决策树。决策树的剪枝，往往从已生成的树上剪掉一些叶结点或叶结点以上的子树，并将其父结点或根结点作为新的叶结点，从而简化生成的决策树。

## 4.1 决策树基本概念

顾名思义，决策树是基于树结构来进行决策的，在网上看到一个例子十分有趣，放在这里正好合适。现想象一位捉急的母亲想要给自己的女娃介绍一个男朋友，于是有了下面的对话：

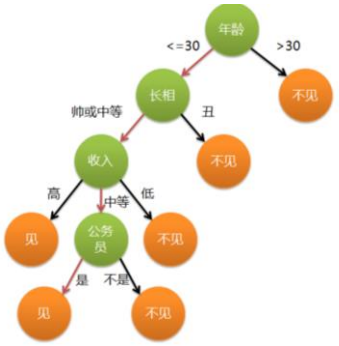
---

女儿：多大年纪了？
母亲：26。
女儿：长的帅不帅？
母亲：挺帅的。
女儿：收入高不？

母亲：不算很高，中等情况。
女儿：是公务员不？
母亲：是，在税务局上班呢。
女儿：那好，我去见见。

---

这个女孩的挑剔过程就是一个典型的决策树，即相当于通过年龄、长相、收入和是否公务员将男童鞋分为两个类别：见和不见。假设这个女孩对男人的要求是：30 岁以下、长相中等以上并且是高收入者或中等以上收入的公务员，那么使用下图就能很好地表示女孩的决策逻辑（即一颗决策树）。

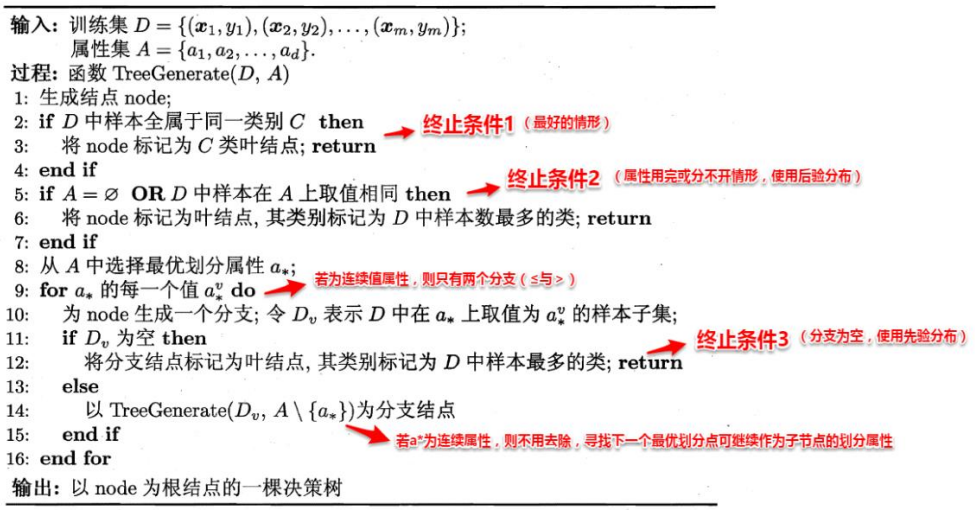



在上图的决策树中，决策过程的每一次判定都是对某一属性的“测试”，决策最终结论则对应最终的判定结果。一般一颗决策树包含：一个根节点、若干个内部节点和若干个叶子节点，易知：

* 每个非叶节点表示一个特征属性测试。
* 每个分支代表这个特征属性在某个值域上的输出。
* 每个叶子节点存放一个类别。
* 每个节点包含的样本集合通过属性测试被划分到子节点中，根节点包含样本全集。

## 4.2 决策树的构造

决策树的构造是一个递归的过程，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，这时直接将该节点标记为叶节点，并设为相应的类别；(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，这时将该节点标记为叶节点，并将其类别设为该节点所含样本最多的类别；(3) 当前结点包含的样本集合为空，不能划分，这时也将该节点标记为叶节点，并将其类别设为父节点中所含样本最多的类别。算法的基本流程如下图所示：




可以看出：决策树学习的关键在于如何选择划分属性，不同的划分属性得出不同的分支结构，从而影响整颗决策树的性能。属性划分的目标是让各个划分出来的子节点尽可能地“纯”，即属于同一类别。因此下面便是介绍量化纯度的具体方法，决策树最常用的算法有三种：ID3，C4.5 和 CART。

### 4.2.1 ID3 算法

ID3 算法使用信息增益为准则来选择划分属性，“信息熵”(information entropy)是度量样本结合纯度的常用指标，假定当前样本集合 D 中第 k 类样本所占比例为 pk，则样本集合 D 的信息熵定义为：

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

值越大表示越混乱，易知只有一个类别时，信息熵为0

假定通过属性划分样本集 D，产生了 V 个分支节点，v 表示其中第 v 个分支节点，易知：分支节点包含的样本数越多，表示该分支节点的影响力越大。故可以计算出划分后相比原始数据集 D 获得的“信息增益”（information gain）。

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

信息增益越大，表示使用该属性划分样本集 D 的效果越好，因此 ID3 算法在递归过程中，每次选择最大信息增益的属性作为当前的划分属性。

### 4.2.2 C4.5 算法

ID3 算法存在一个问题，就是偏向于取值数目较多的属性，例如：如果存在一个唯一标识，这样样本集 D 将会被划分为|D|个分支，每个分支只有一个样本，这样划分后的信息熵为零，十分纯净，但是对分类毫无用处。因此 C4.5 算法使用了“增益率”（gain ratio）来选择划分属性，来避免这个问题带来的困扰。首先使用 ID3 算法计算出信息增益高于平均水平的候选属性，接着 C4.5 计算这些候选属性的增益率，增益率定义为：

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

其中

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

当α属性的取值越多时IV(α)值越大

### 4.2.3 CART 算法

CART 决策树使用“基尼指数”（Gini index）来选择划分属性，基尼指数反映的是从样本集 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)越小越好，基尼指数定义如下：

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

任取两个样本类标不一致的概率

越小表示集合越纯

进而，使用属性 α 划分后的基尼指数为：

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

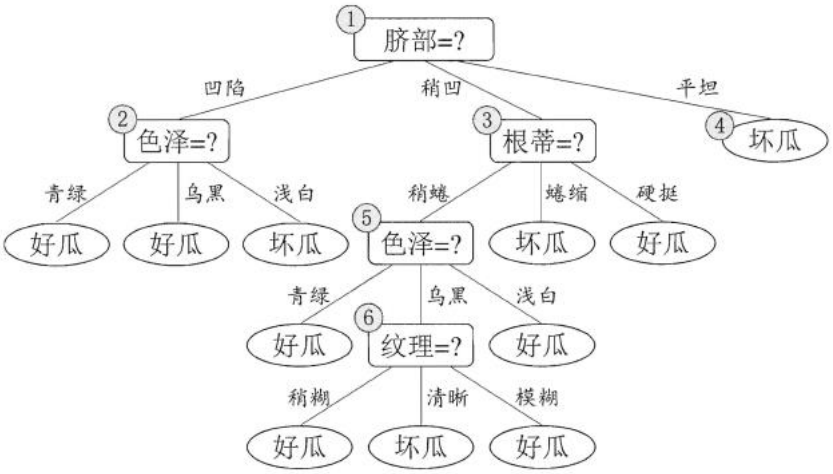
故选择基尼指数最小的划分属性

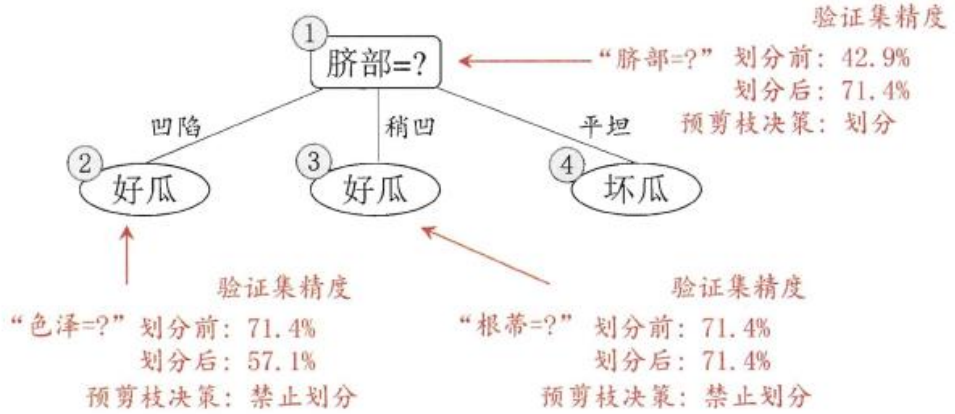
## 4.3 剪枝处理

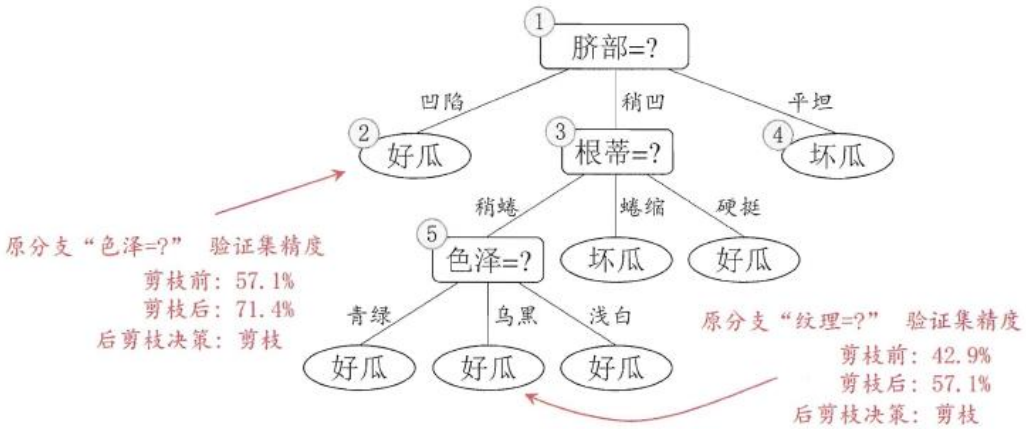
从决策树的构造流程中我们可以直观地看出：不管怎么样的训练集，决策树总是能很好地将各个类别分离开来，这时就会遇到之前提到过的问题：过拟合（overfitting），即太依赖于训练样本。剪枝（pruning）则是决策树算法对付过拟合的主要手段，剪枝的策略有两种如下：

```
* 预剪枝（prepruning）：在构造的过程中先评估，再考虑是否分支。
* 后剪枝（post-pruning）：在构造好一颗完整的决策树后，自底向上，评估分支的必要性。
```

评估指的是性能度量，即决策树的泛化性能。之前提到：可以使用测试集作为学习器泛化性能的近似，因此可以将数据集划分为训练集和测试集。预剪枝表示在构造数的过程中，对一个节点考虑是否分支时，首先计算决策树不分支时在测试集上的性能，再计算分支之后的性能，若分支对性能没有提升，则选择不分支（即剪枝）。后剪枝则表示在构造好一颗完整的决策树后，从最下面的节点开始，考虑该节点分支对模型的性能是否有提升，若无则剪枝，即将该节点标记为叶子节点，类别标记为其包含样本最多的类别。



上图分别表示不剪枝处理的决策树、预剪枝决策树和后剪枝决策树。预剪枝处理使得决策树的很多分支被剪掉，因此大大降低了训练时间开销，同时降低了过拟合的风险，但另一方面由于剪枝同时剪掉了当前节点后续子节点的分支，因此预剪枝“贪心”的本质阻止了分支的展开，在一定程度上带来了欠拟合的风险。而后剪枝则通常保留了更多的分支，因此采用后剪枝策略的决策树性能往往优于预剪枝，但其自底向上遍历了所有节点，并计算性能，训练时间开销相比预剪枝大大提升。

## 4.4 连续值与缺失值处理

对于连续值的属性，若每个取值作为一个分支则显得不可行，因此需要进行离散化处理，常用的方法为二分法，基本思想为：给定样本集 D 与连续属性 α，二分法试图找到一个划分点 t 将样本集 D 在属性 α 上分为≤t 与 > t。

* 首先将 α 的所有取值按升序排列，所有相邻属性的均值作为候选划分点（n-1 个，n 为 α 所有的取值数目）。
* 计算每一个划分点划分集合 D（即划分为两个分支）后的信息增益。
* 选择最大信息增益的划分点作为最优划分点。

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

划分为两个分支

现实中常会遇到不完整的样本，即某些属性值缺失。有时若简单采取剔除，则会造成大量的信息浪费，因此在属性值缺失的情况下需要解决两个问题：(1) 如何选择划分属性。(2) 给定划分属性，若某样本在该属性上缺失值，如何划分到具体的分支上。假定为样本集中的每一个样本都赋予一个权重，根节点中的权重初始化为 1，则定义：

$$\rho=\frac{\sum_{\boldsymbol{x}\in\tilde{D}} w_{\boldsymbol{x}}}{\sum_{\boldsymbol{x}\in D} w_{\boldsymbol{x}}},$$ 样本子集所占比例

$$\tilde{p}_k=\frac{\sum_{\boldsymbol{x}\in\tilde{D}_k} w_{\boldsymbol{x}}}{\sum_{\boldsymbol{x}\in\tilde{D}} w_{\boldsymbol{x}}}\quad (1\leqslant k\leqslant|\mathcal{Y}|),$$ 样本子集每个类别的比例

$$\tilde{r}_v=\frac{\sum_{\boldsymbol{x}\in\tilde{D}^v} w_{\boldsymbol{x}}}{\sum_{\boldsymbol{x}\in\tilde{D}} w_{\boldsymbol{x}}}\quad (1\leqslant v\leqslant V).$$ 每个分支所含样本比例

对于 (1)：通过在样本集 D 中选取在属性 α 上没有缺失值的样本子集，计算在该样本子集上的信息增益，最终的信息增益等于该样本子集划分后信息增益乘以样本子集占样本集的比重。即：

$$\mathrm{Gain}(D,a) = \rho \times \mathrm{Gain}(\tilde{D},a)$$ 表示在属性α上无缺失的样本子集

无缺失样本子集所占比重

$$= \rho \times \left( \mathrm{Ent}\left(\tilde{D}\right) - \sum_{v=1}^{V} \tilde{r}_v \, \mathrm{Ent}\left(\tilde{D}^v\right) \right)$$

对于（2）：若该样本子集在属性 α 上的值缺失，则将该样本以不同的权重（即每个分支所含样本比例）划入到所有分支节点中。该样本在分支节点中的权重变为：

$\omega_x = \omega_x * \tilde{r}_v$ 即以不同权重划分到所有分支节点中