决策树

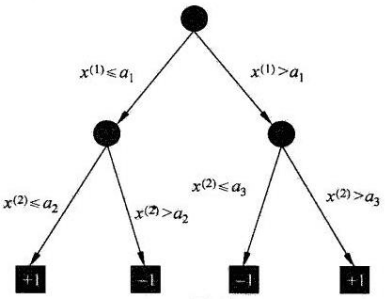
sysu & spf 机器学习研修班系列讲义　3

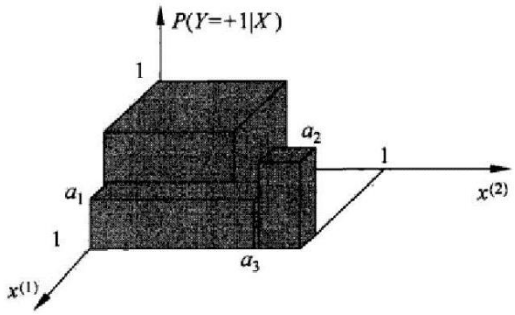
主讲人： 薛轲

中山大学数学学院

决策树综述

决策树（decision tree）是机器学习中基本的分类与回归算法。这次我们的重点在分类问题中，实质上决策树是基于特征对实例的分类（if-then 结构），或者可以理解成定义在特种空间和类空间上的条件概率分布，也就是对空间用超平面进行划分。





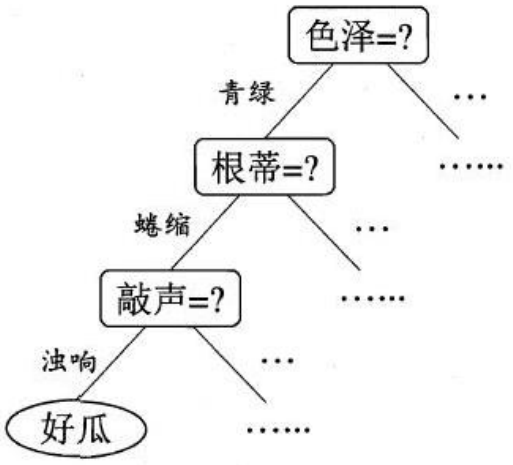
决策树的主要优点：模型的可读性强，计算复杂度不高（分类速度快），输出结果易于理解，对中间值的缺失不敏感，可以处理不相关的特征数据 。主要缺点：容易出现过度匹配，处理连续变量不好

构建决策树时，我们基于损失函数最小化的原则来建立模型，通常分为三步 :特征选择，决策树生成，决策树修剪。常用算法有 ID3，C4.5，CART 等。

实际应用上，在金融领域中，决策树常用来进行用户的分级评估和风险评估；医疗领域中，决策树可以生成辅助诊断处置模型。另外，决策树的衍生算法也有较好的应用，比如随机森林模型（Random Forest）。

决策树的构建与学习

先看一个简单的决策树例子。



决策树构建的本质就是从训练数据集当中归纳出一组分类规则。我们需要的是与训练数据矛盾较小的决策树，并且尽可能拥有较好的泛化能力。从条件概率模型的角度上考虑，我们生成的决策树应该不仅能够对训练数据有很好的拟合，而且对未知数据也能有很好的预测。

这一目标如何用数学语言来描述呢？我们使用损失函数（loss function）来表示，学习策略就是使得损失函数最小化。损失函数通常是正则化的极大似然函数

[启发式方法指人在解决问题时所采取的一种根据经验规则进行发现的方法。其特点是在解决问题时,利用过去的经验,选择已经行之有效的方法，而不是系统地、以确定的步骤去寻求答案。]

决策树构建算法过程：

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

可以看出，决策树三种停止条件（把当前 node 标记为叶节点）：

(1) 当前结点包含的样本全属于同一类别，无需划分；对应第 3 行。

(2) 当前属性集为空，或是所有样本在所有属性上在给定阈值下取值相同，无法划分；对应第 5 行。

(3) 当前结点包含的样本集合为空，不能划分；对应第 12 行。

决策树的生成

决策树生成的关键步骤在于如何选择最优划分属性，也就是如何进行特征选择。特征选择实质上就是决定用哪个特征来划分特征空间。

如何来判断特征的选择优良与否呢？一般有三个标准：信息增益，信息增益比，基尼指数，它们对应着决策树生成的三大算法：ID3，C4.5 与 CART。首先先给出信息熵的定义。

1928 年，Hartley L.V.R. 提出信息量的定义：

$$H=log_2 s^n=nlog_2 s$$

假设某条信息含有 n 个符号，每个符号有 s 种不同的可能，则该条信息的组成一共有 s^n 种情况。若以 s^n 作为信息量的度量，结果将随着信息的长度的增加呈指数增长。为不改变信息量随长度增长或单一符号可能情况增多而递增的特性的同时，降低其递增速度，对 s^n 取对数处理。

Hartley 信息量假设每种符号的出现概率相同，这在实际应用中往往不成立。Shannon（后来被称为信息论之父）在此基础上提出用概率替代可能性，信息量依概率加权计算得出。Shannon 熵的定义为：假设某一随机现象有 n 种可能，各情况对应概率分别为 $p_1,p_2,\cdots p_n$，则有

$$H(p)=\sum_i p_i log_2\frac{1}{p_i}=-\sum_i p_i log_2 p_i$$

从信息熵(Entropy)定义可以看出来，分布越集中，熵值就越低。

1.信息增益与 ID3 算法

了解信息熵以后，我们给出信息增益的定义：

首先定义条件熵，

$$H(Y|X)=\sum_{i=1}^{n}p_i\,H(Y|X=x_i)$$

这里随机变量 X 是某个特定特征，随机变量 Y 是类。n 是某属性x的 n 种取值可能，x_i表示随机变量 X 的每一种可能取值。

接下来定义信息增益，

$$g(D,A)=H(D)-H(D|A)=H(D)-\sum_{i=1}^{K}\frac{|D^i|}{|D|}H(D^i)$$

这里 D 代表数据集，A 代表特征，有 K 个可能的取值$\{A_1,A_2\dots,A_k\}$，D^i代表所有在特征 A 上取值为A_i的样本。

注意到，信息增益定义为集合 D 的经验熵 H(D)与特征 A 在给定条件下 D 的经验条件熵 H(D|A)之差。（由于真正的熵并不知道，这里的熵是根据样本进行极大似然估计出来的）一般地，熵 H(Y)与 H(Y|X)之差称为互信息(mutual information)。决策树学习中的信息增益等价于训练数据集中类与特征的互信息。

一般而言，信息增益越大，就意味着用属性 A 来进行划分的“纯度提升”就越大。ID3 决策树学习算法(Iterative Dichotomiser 3，迭代二分法 3)以信息增益为准则来进行决策树的划分属性的选择。

以下面的数据集为例，讲解一下 ID3 算法的构建过程。

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

在 17 条数据中，正例有 8 例，反例有 9 例，可以计算根节点的信息熵。

$$\mathrm{Ent}(D)=-\sum_{k=1}^{2}p_k\log_2 p_k=-\left(\frac{8}{17}\log_2\frac{8}{17}+\frac{9}{17}\log_2\frac{9}{17}\right)=0.998\ .$$

当前属性集合为｛色泽，根蒂，敲声，纹理，脐部，触感。以”色泽”为例，他有 3 个可能的取值｛青绿，乌黑，浅白｝，如果用色泽属性对 D 进行划分，可以得到 3 个子集，分别

记为D^1、D^2、D^3。

子集 D^1 包含编号为 $\{1, 4, 6, 10, 13, 17\}$ 的 6 个样例, 其中正例占 $p_1 = \frac{3}{6}$, 反例占 $p_2 = \frac{3}{6}$; D^2 包含编号为 $\{2, 3, 7, 8, 9, 15\}$ 的 6 个样例, 其中正、反例分别占 $p_1 = \frac{4}{6}$, $p_2 = \frac{2}{6}$; D^3 包含编号为 $\{5, 11, 12, 14, 16\}$ 的 5 个样例, 其中正、反例分别占 $p_1 = \frac{1}{5}$, $p_2 = \frac{4}{5}$. 根据式(4.1)可计算出用“色泽”划分之后所获得的 3 个分支结点的信息熵为

$$\begin{aligned}
\mathrm{Ent}(D^1) &= -\left(\frac{3}{6}\log_2\frac{3}{6} + \frac{3}{6}\log_2\frac{3}{6}\right) = 1.000\,, \\
\mathrm{Ent}(D^2) &= -\left(\frac{4}{6}\log_2\frac{4}{6} + \frac{2}{6}\log_2\frac{2}{6}\right) = 0.918\,, \\
\mathrm{Ent}(D^3) &= -\left(\frac{1}{5}\log_2\frac{1}{5} + \frac{4}{5}\log_2\frac{4}{5}\right) = 0.722\,,
\end{aligned}$$

那么我们就可以计算出 A=色泽 时的信息增益了。

$$\begin{aligned}
\mathrm{Gain}(D, \text{色泽}) &= \mathrm{Ent}(D) - \sum_{v=1}^{3}\frac{|D^v|}{|D|}\mathrm{Ent}(D^v) \\
&= 0.998 - \left(\frac{6}{17}\times 1.000 + \frac{6}{17}\times 0.918 + \frac{5}{17}\times 0.722\right) \\
&= 0.109\,.
\end{aligned}$$

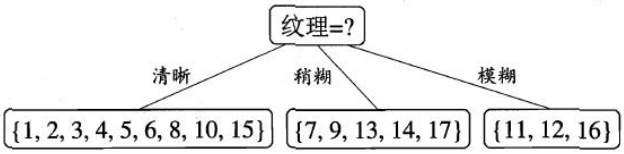
类似的, 我们可计算出其他属性的信息增益:

$\mathrm{Gain}(D, \text{根蒂}) = 0.143$; $\mathrm{Gain}(D, \text{敲声}) = 0.141$;

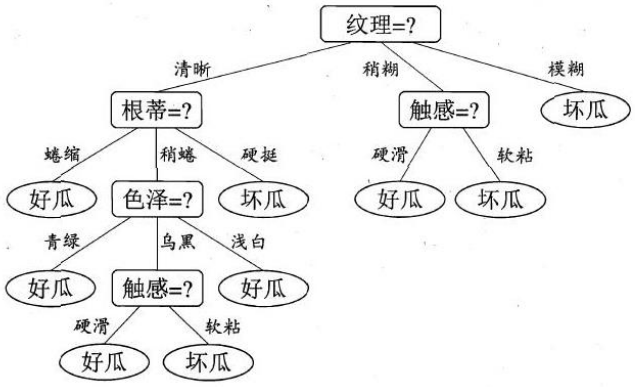
$\mathrm{Gain}(D, \text{纹理}) = 0.381$; $\mathrm{Gain}(D, \text{脐部}) = 0.289$;

$\mathrm{Gain}(D, \text{触感}) = 0.006$.

可以看出纹理的信息增益最高，因此选择纹理作为根节点划分。



对每一个节点进行以上操作，可以得到最终的决策树。



算法（ID3 算法）

输入：训练数据集 D，特征集 A，阈值ε；

输出：决策树 T。

(1)若 D 中所有实例属于同一类C_k，则 T 为单结点树，并将类C_k作为该结点的类标记，返回 T；

(2)若$A = \varnothing$，则 T 为单结点树，并将 D 中实例数最大的类C_k作为该结点的类标记，返回 T；

(3)否则，按算法计算 A 中各特征对 D 的信息增益，选择信息增益最大的特征A_g；

(4)如果A_g的信息增益小于阈值ε，则置 T 为单结点树，并将 D 中实例数最大的类C_k作为该结点的类标记，返回 T；

(5)否则，对A_g的每一可能值a_i，依$A_g = a_i$将 D 分割为若干非空子集D_i，将D_i中实例数最大的类作为标记，构建子结点，由结点及其子结点构成树 T，返回 T；

(6)对第 i 个子结点，以D_i为训练集，以$A - \{A_g\}$为特征集，递归地调用步(1)~步(5)，得到子树T_i，返回T_i。

2.信息增益比与 C4.5 算法

细心的同学可能发现，我们在选择特征空间时，没有将“编号”这一栏纳入到特征选择中，如果将编号纳入考量，仍以信息增益作为评判标准的话，特征的最佳选择就是编号了。事实上，信息增益的实质是提高“纯度”，那么，以信息增益作为划分训练数据集的特征，存在偏向于选择取值较多的特征的问题。使用信息增益比可以对这一问题进行校正。

特征 A 对训练数据集 D 的信息增益比$g_R(\mathrm{D},\mathrm{A})$定义如下：

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

其中，g(D,A)表示信息增益，$H_A(D) = -\sum_{i=1}^{n}\frac{|D_i|}{|D|}\log_2\frac{|D_i|}{|D|}$,n 是特征 A 取值的个数。

与 ID3 类似，C4.5 在生成过程中用信息增益比来选择特征。

3.GINI 指数与 CART 算法

CART 分类树使用 GINI 指数作为选择的最优特征。

分类问题中，假设有 K 个类，样本点属于第 k 类的概率为P_k，则概率分布的基尼指数定义为：

$$\text{Gini}(p)=\sum_{k=1}^{K}p_k(1-p_k)=1-\sum_{k=1}^{K}p_k^2$$

样本集合 D 的基尼指数（CART）

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

特征 A 条件下集合 D 的基尼指数：

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

Gini(D)表示了集合 D 的不确定性，直观来讲，它反映了从数据集 D 中随机抽取两个样本，其类别标志不一致的概率，Gini 指数越小，则数据集 D 的纯度越高。

Gini 指数和熵有什么关系呢？首先看一下他们的表达式。

$$\begin{cases} H(X)=-\sum_{k=1}^{K}p_k\ln p_k \\ \text{Gini}(X)=\sum_{k=1}^{K}p_k(1-p_k) \end{cases}$$

将 f(x)=−lnx 在 x=1 处进行一阶泰勒展开（忽略高阶无穷小）：

$$\begin{aligned} f(x)&=f(x_0)+f'(x_0)(x-x_0)+o(\cdot)\\ &=f(1)+f'(1)(x-1)+o(\cdot)\\ &=1-x \end{aligned}$$

则有如下近似：

$$\begin{aligned} H(X)&=-\sum_{k=1}^{K}p_k\ln p_k=\sum_{k=1}^{K}p_k(-\ln p_k)\\ &\simeq\sum_{k=1}^{K}p_k(1-p_k)=\text{Gini}(X) \end{aligned}$$

CART 分类树的构建算法：

算法 5.6（CART 生成算法）

输入：训练数据集 D，停止计算的条件；

输出：CART 决策树.

根据训练数据集，从根结点开始，递归地对每个结点进行以下操作，构建二叉决策树：

（1）设结点的训练数据集为 D，计算现有特征对该数据集的基尼指数. 此时，对每一个特征 A，对其可能取的每个值 a，根据样本点对 $A=a$ 的测试为“是”或“否”将 D 分割成 D_1 和 D_2 两部分，利用式（5.25）计算 $A=a$ 时的基尼指数.

（2）在所有可能的特征 A 以及它们所有可能的切分点 a 中，选择基尼指数最小的特征及其对应的切分点作为最优特征与最优切分点. 依最优特征与最优切分点，从现结点生成两个子结点，将训练数据集依特征分配到两个子结点中去.

（3）对两个子结点递归地调用（1），（2），直至满足停止条件.

（4）生成 CART 决策树.

CART 生成树的停止条件：

1.结点中的样本个数小于预定阈值

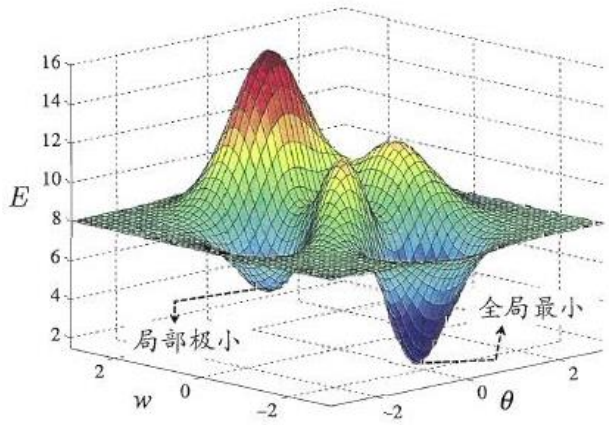
2.样本集的基尼指数小于预定阈值（样本基本属于同一类）

3.没有更多特征

一般情况下我们会选择启发式的方法来确定指标，比如先从划分属性中找出信息增益高于平均水平的，再从中选择增益率最高的。

决策树的剪枝

前面已经提到，决策树的一大缺点就是容易产生过度匹配的情况。过度匹配显然不是我们希望得到的结果，我们训练模型最终还是希望他在新的情境下也能有效，也就是要具备一定的泛化能力。



剪枝的目的：减少过拟合的情况，提高模型泛化能力。具体来说，剪枝是从已经生成的树上裁掉一些子树或者叶节点，并将其根节点或父节点作为新的叶节点，从而简化分类树模型。

剪枝通常情况下通过极小化决策树整体的损失函数或代价函数来实现。

设树 T 的叶结点个数为$|T|$，t 是树 T 的叶结点，该叶结点有N_t个样本点，其中 k 类的样本点有N_{tk}个，$k=1,2,...,K$，，$\alpha \geqslant 0$为参数，则决策树学习的损失函数可以定义如

下：

$$C_\alpha(T) = C(T) + \alpha|T|$$

$$C(T) = \sum_{t=1}^{|T|} N_t H_t(T) = -\sum_{t=1}^{|T|}\sum_{k=1}^{K} N_{tk}\log\frac{N_{tk}}{N_t}$$

C(T)表示模型对训练数据的预测误差，即拟合程度；

|T|表示模型复杂度，参数α>=0 用来控制二者之间的影响。

较大的α倾向于选择较简单的树，较小的α倾向于选择较复杂的树。α=0 时意味着只考虑拟合程度不考虑模型的复杂程度。损失函数表示了模型复杂度和训练数据拟合度之间的平衡。

剪枝做的事情就是在给定的α下，选择损失函数最小的模型。

算法（树的剪枝算法）

输入：生成算法产生的整个树 T，参数α；

输出：修剪后的子树T_α。

(1)计算每个结点的经验熵。

(2)递归地从树的叶结点向上回缩。

设一组叶结点回缩到其父结点之前与之后的整体树分别为T_B与T_A，其对应的损失函数值分别是

$$C_\alpha(T_A) \leqslant C_\alpha(T_B)$$

则进行剪枝，即将父结点变为新的叶结点。

(3)返回(2)，直至不能继续为止，得到损失函数最小的子树T_α。

接下来我们来看 CART 剪枝，也叫 Cost-Complexity Pruning(CCP)，代价复杂度剪枝，事实上剪枝算法很多，这本书只是介绍了两种剪枝方法，后面我们会说到这一点。

CART 剪枝算法和之前的算法有所不同，之前是给定α进行剪枝，而 CART 算法需要自己来寻找α ;相同点在于他们的目标都是让损失函数最小化。仅仅针对这个问题，我们会考虑，如何使得损失函数最小化？

$$C_\alpha(T) = C(T) + \alpha|T|$$

1.降低第一部分的不确定次数。然而对于 CART 剪枝处理的是“完全生长”的决策树。降低不确定次数的办法是再找寻一个特征，或者找寻特征的最优切分点。这在生成决策时就已经决定了，无法改变。

2.进行剪枝操作。剪枝最明显地变化就是叶结点个数变少，也就是复杂度降低。比如对三叉树的剪枝。（这也另一方面说明了剪枝操作的重要性）。

因为α参数给定，所以一次剪枝，损失函数的减少量也是固定的。所有子结点都是三叉树时，我们可以简单认为损失函数的减少量为 2α。假设只有一个子结点发生剪枝，那么该子结点上的叶结点都将全部归并到该结点，由此我们计算此结点的不确定次数。倘若不确定次数增大的量超过 2α，那么不进行剪枝，否则进行剪枝；然后再对下一个节点，进行类似的判断。这里给出一个 Claim，就是对于每一个α，我们都有唯一的最小子树与之对应。那我们现在的想法是找到子树与α区间的一一对应，

这里有两种操作：遍历α，找到每一棵子树，同一棵子树对应的α划分到同一区间，直到找全所有的子树。显而易见这个方法很难实现，但是给了我们一个思路，那就是第二种方法。

显而易见，一棵决策树，它的子树是有限棵的，我们可以对每一颗子树，来找到对应的

α区间。那么怎么得到这些子树呢？我们要从下往上，每次只进行一次剪枝，直到剪枝到根节点为止，得到了一组嵌套的子树序列。接下来通过最优子树来求解参数α。

这里基于一个假设：一定会发生剪枝；剪枝结束后，当前决策树是最优子树。（这很重要，我们要理解清楚哪些是已知的哪些是未知的）（不进行假设，逻辑推不下去）

剪枝后（节点 t 吞并了它的原来的叶子节点）：

$$C_\alpha(t) = C(t) + \alpha$$

剪枝前：

$$C_\alpha(T_t) = C(T_t) + \alpha|T_t|$$

这里，只有α是位置的，根据书上叙述的α增大时两个损失函数的关系，当两个损失函数相等时，我们有：

$$\alpha 1 = \frac{C(t) - C(T_1)}{|T_t - 1|}$$

当且仅当α>=α1 时，发生剪枝，假设成立。

要想证明我们的是最优子树，说白了就是要找 t。

$$t = \arg\min \frac{C(t) - C(T_t)}{|T_t| - 1}$$

有α有 t，可以完成剪枝，得到子树T_1，递归地剪枝下去就可以得到子树序列了。

1.剪枝，形成子树序列。

确定剪枝过程中的损失函数， 对于固定的α，一定有使得损失函数最小的子树 T_α，它在损失函数最小的意义下是最优的。

Breiman 的证明，说明了我们可以用递归的方法，得到一系列嵌套的最优子树序列。

2.在子树序列中交叉验证选取最优子树 T_α

使用独立的测试集，测试最优子树序列各子树的平方误差或基尼指数，平方误差或基尼指数最小的决策树视为最优的决策树，由于每一棵子树都对应于一个参数α，当最优子树确定，α确定，最优决策树也确定。

算法 5.7（CART 剪枝算法）

输入：CART 算法生成的决策树 T_0；

输出：最优决策树 T_α.

（1）设 $k=0$，$T=T_0$.

（2）设 $\alpha=+\infty$.

（3）自下而上地对各内部结点 t 计算 $C(T_t)$，$|T_t|$ 以及

$$g(t) = \frac{C(t) - C(T_t)}{|T_t| - 1}$$

$$\alpha = \min(\alpha, g(t))$$

这里，T_t 表示以 t 为根结点的子树，$C(T_t)$ 是对训练数据的预测误差，$|T_t|$ 是 T_t 的叶结点个数.

（4）自上而下地访问内部结点 t，如果有 $g(t)=\alpha$，进行剪枝，并对叶结点 t 以多数表决法决定其类，得到树 T.

（5）设 $k=k+1$，$\alpha_k=\alpha$，$T_k=T$.

（6）如果 T 不是由根结点单独构成的树，则回到步骤 (4).

（7）采用交叉验证法在子树序列 $T_0,T_1,\cdots,T_n$ 中选取最优子树 T_α. ■

拓展阅读

1.决策树的历史

第一个决策树算法：CLS (Concept Learning System)

[E. B. Hunt, J. Marin, and P. T. Stone's book "Experiments in Induction" published by Academic Press in 1966]

使决策树受到关注并成为机器学习主流技术的算法：ID3

[J. R. Quinlan's paper in a book "Expert Systems in the Micro Electronic Age" edited by D. Michie, published by Edinburgh University Press in 1979]

最常用的决策树算法：C4.5

[J. R. Quinlan's book "C4.5: Programs for Machine Learning" published by Morgan Kaufmann in 1993]

可以用于回归任务的决策树算法：CART (Classification and Regression Tree)

[L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone's book "Classification and Regression Trees" published by Wadsworth in 1984]

基于决策树的(最)强大算法：RF (Random Forest)

[L. Breiman's MLJ'01 paper "Random Forest"]

2.过拟合，欠拟合，正则化

关于过拟合，欠拟合以及正则化的问题，我们在各个机器学习模型的学习中都会遇到，这篇博客叙述的比较好，大家下来可以了解一下，这里就不多赘述了。

http://www.cnblogs.com/jianxinzhou/p/4083921.html

3.关于启发式原则

感兴趣的同学看一下这篇文章，或者可以自己查一些相关的资料了解一下有哪些常用的启发式原则。

https://arxiv.org/abs/1412.6980

3.连续值、缺失值、多变量的处理

在实际的数据处理中，我们处理的数据并不会像给出的例子这样都是离散的、完整的，而是尝尝会遇到连续值，以及数据的缺失。另外，我们并不满足于只生成我们生成的决策树

具体参考 周志华《机器学习》 第四章 决策树 P83-91

4.其他优秀的决策树算法

C5.0

C5.0 和 C4.5 算法的对比：

1.都是通过计算**信息增益率**来划分结点，两者的共同

2.C5.0 算法通过构造多个 C4.5 算法，是一种 **boosting 算法**，可以增加惩罚项而且准确率更高。

3.C5.0 算法运行**速度快**，可以出来例如,C4.5 需要 9 个小时找到森林的规则集,但 C5.0 在 73 秒完成了任务。

4.C5.0 运行**内存小**。C4.5 需要超过 3 GB.,但 C5.0 需要少于 200 mb。

5.C5.0 算法，可以人为的加入**客观规则**

6.C5.0 可以**处理较大的数据集**，特征可以是：数字，时间，日期，名义字段。

QUEST 决策树

quick unbiased efficient statistical tree （快速无偏有效统计树）

QUEST 节点可提供用于构建决策树的二元分类法，此方法的设计目的是减少大型 CART 决策树分析所需的处理时间，同时减小分类树方法中常见的偏向类别较多预测变量的趋势。预

测变量字段可以是数字范围的，但目标字段必须是分类的。所有分割都是二元的。

5.决策树的剪枝

事实上，决策树的剪枝分为预剪枝和后剪枝两部分。《统计学习方法》这本书里说的都是后剪枝，也就是生成树之后再进行剪枝。想要快速了解预剪枝和后剪枝的同学，可以参考周志华《机器学习》P80-83，这里简单介绍一下。

预剪枝

•预剪枝就是在完全正确分类训练集之前，较早地停止树的生长。 具体在什么时候停止决策树的生长有多种不同的方法:

(1) 一种最为简单的方法就是在决策树到达一定高度的情况下就停止树的生长。

(2) 到达此结点的实例具有相同的特征向量，而不必一定属于同一类， 也可停止生长。

(3) 到达此结点的实例个数小于某一个阈值也可停止树的生长。

(4) 还有一种更为普遍的做法是计算每次扩张对系统性能的增益，如果这个增益值小于某个阈值则不进行扩展。

优点

•由于预剪枝不必生成整棵决策树，且算法相对简单，效率很高，适合解决大规模问题。

缺点

•视野效果问题。也就是说在相同的标准下，也许当前的扩展会造成过度拟合训练数据，但是更进一步的扩展能够满足要求，也有可能准确地拟合训练数据。这将使得算法过早地停止决策树的构造。

后剪枝

后剪枝，在已生成过拟合决策树上进行剪枝，可以得到简化版的剪枝决策树。

这里主要介绍四种：

1、REP-错误率降低剪枝

REP(Reduced Error Pruning)方法，对于决策树 T 的每棵非叶子树 S，用叶子替代这棵子树. 如果 S 被叶子替代后形成的新树关于 D 的误差等于或小于 S 关于 D 所产生的误差，则用叶子替代子树 S。

优点：

1.REP 是当前最简单的事后剪枝方法之一。

2.它的计算复杂性是线性的。

3.和原始决策树相比，修剪后的决策树对未来新事例的预测偏差较小。

缺点：

数据量较少的情况下很少应用. REP 方法趋于过拟合(overfitting)，这是因为训练数据集中存在的特性在剪枝过程中都被忽略了，当剪枝数据集比训练数据集小得多时 ，这个问题特别值得注意.。

2、PEP-悲观剪枝

PEP(Pessimistic Error Pruning)方法，为了克服 REP 方法需要独立剪枝数据集的缺点而提出的，它不需要分离的剪枝数据集，为了提高对未来事例的预测可靠性，PEP 方法对误差估计增加了连续性校正(continuity correction)。

优点：

1.PEP 方法被认为是当前决策树事后剪枝方法中精度较高的算法之一

2.PEP 方法不需要分离的剪枝数据集，这对于事例较少的问题非常有利

3.它的计算时间复杂性也只和未剪枝树的非叶节点数目成线性关系 .
缺点：
PEP 是唯一使用自顶向下剪枝策略的事后剪枝方法，这种策略会带来与事前剪枝方法出现的同样问题，那就是树的某个节点会在该节点的子孙根据同样准则不需要剪裁时也会被剪裁。

3、CCP-代价复杂度剪枝

就是我们说的 CART 剪枝
优点：
精度高。
缺点：
生成子树序列 T (α) 所需要的时间和原决策树非叶节点的关系是 $O(n^2)$的，这就意味着这就比其它剪枝方法所需要的时间长得多，因为其它剪枝方法的运行时间和非叶节点的关系是 O(n)的.

4、MEP-最小错误剪枝

MEP(Minimum Error Pruning)方法的基本思路是采用自底向上的方式，对于树中每个非叶节点，首先计算该节点的误差 Er(t) . 然后，计算该节点每个分枝的误差 Er(Tt) ，并且加权相加，权为每个分枝拥有的训练样本比例. 如果 Er(t) 大于 Er(Tt) ，则保留该子树; 否则，剪裁它.
优点：
1.MEP 方法不需要独立的剪枝数据集，无论是初始版本，还是改进版本，在剪枝过程中，使用的信息都来自于训练样本集.
2.它的计算时间复杂性也只和未剪枝树的非叶节点数目成线性关系 .

缺点：
类别平均分配的前提假设现实几率不大&对 K 太过敏感

6.RF 与 GBDT

单决策树 C4.5 由于功能太简单，并且非常容易出现过拟合的现象，于是引申出了许多变种决策树，就是将单决策树进行模型组合，形成多决策树，比较典型的就是迭代决策树 GBRT 和随机森林 RF。在最近几年的 paper 上，有不少文章都是与 Boosting 和随机森林相关的。模型组合+决策树相关算法有两种比较基本的形式：随机森林 RF 与 GBDT，其他比较新的模型组合+决策树算法都是来自这两种算法的延伸。

随机森林是一个用随机方式建立的，包含多个决策树的分类器。其输出的类别是由各个树输出的类别的众数而定。它的随机性主要体现在两个方面：(1) 训练每棵树时，从全部训练样本中选取一个子集进行训练（即 bootstrap 取样)。用剩余的数据进行评测，评估其误差；(2) 在每个节点，随机选取所有特征的一个子集，用来计算最佳分割方式。

随机森林的主要优点：(1) 在大的、高维数据训练时，不容易出现过拟合而且速度较快；(2) 测试时速度很快；(3) 对训练数据中的噪声和错误鲁棒。

GBDT(Gradient Boosting Decision Tree) 是一种迭代的决策树算法，该算法由多棵决策树组成，所有树的结论累加起来做最终结果。它在被提出之初就和 SVM 一起被认为是泛化能力（generalization)较强的算法。近些年更因为被用于搜索排序的机器学习模型而引起大家关注。GBDT 是一个应用很广泛的算法，可以用来做分类、回归。在很多的数据上都有不错的效果。GBDT 这个算法有很多名字，但都是同一个算法。GBDT (Gradient BoostDecision

Tree) 渐进梯度决策树，GBRT (Gradient BoostRegression Tree) 渐进梯度回归树，MART (MultipleAdditive Regression Tree) 多决策回归树，Tree Net 决策树网络

。

7.深度森林

[Zhou Z H, Feng J. Deep Forest: Towards An Alternative to Deep Neural Networks[J]. 2017.]

这是代表论文，周志华教授今年前半年提出的一个理论，有兴趣的同学可以下载论文看一看。

下面是一篇 CSDN 的报道，可以简单地了解一下文章的思路和想法

http://geek.csdn.net/news/detail/160588

“周志华和冯霁提出了一种基于树的新方法——gcForest，用文中的术语说，就是“multi-Grained Cascade forest”，多粒度级联森林。此外，他们还提出了一种全新的决策树集成方法，使用级联结构让 gcForest 做表征学习。实验中，gcForest 使用相同的参数设置，在不同的域中都获得了优异的性能，并且无论是大规模还是小规模的数据，表现都很好。此外，由于是基于树的结构，gcForest 相比神经网络也更容易分析。”

“我们认为，要解决复杂的问题，学习模型也需要往深了走。然而，当前的深度模型全部都是神经网络。这篇论文展示了如何构建深度树（deep forest），为在许多任务中使用深度神经网络之外的方法打开了一扇门。”