西安电子科技大学
XIDIAN UNIVERSITY

数据分析与挖掘

决策树归纳

专业名称	软件工程
姓名	郑健
学号	15130110047
班级	1513011 云

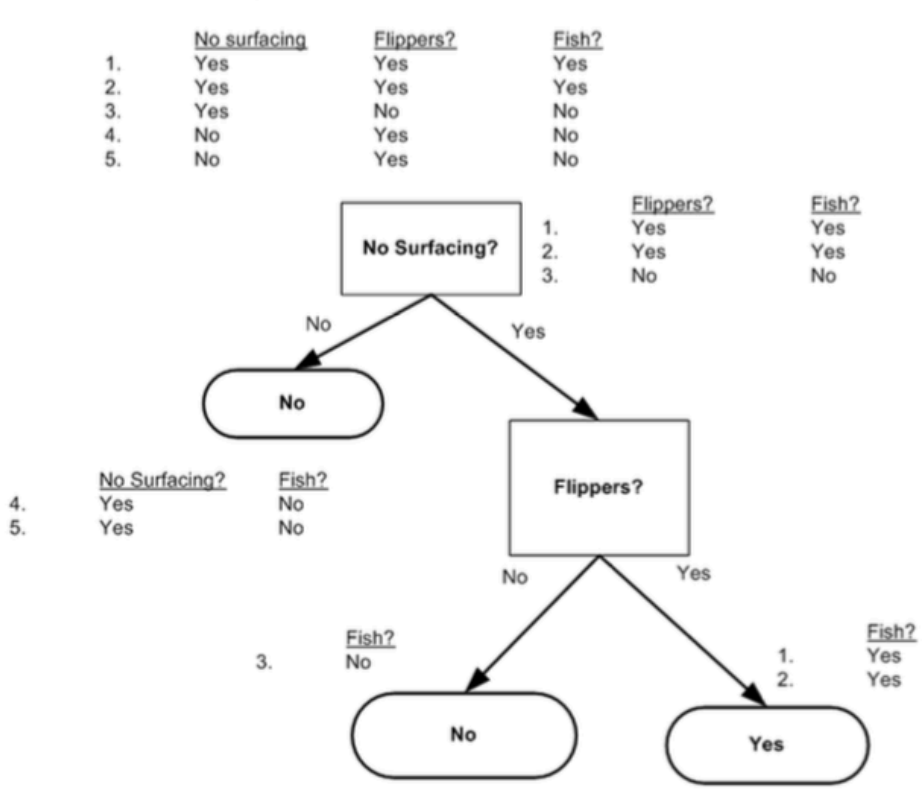
2018 年 11 月

1. 介绍

决策树是一种机器学习的方法。决策树的生成算法有ID3, C4.5和C5.0等。决策树是一种树形结构，其中每个内部节点表示一个属性上的判断，每个分支代表一个判断结果的输出，最后每个叶节点代表一种分类结果。决策树是一种十分常用的监督学习分类方法，监管学习就是给出一堆样本，每个样本都有一组属性和一个分类结果，也就是分类结果已知，那么通过学习这些样本得到一个决策树，这个决策树能够对新的数据给出正确的分类。

这里通过一个简单的例子来说明决策树的构成思路：给出如下的一组数据，一共有五个样本，每个样本有'不浮出水面是否可以生存'，'是否有脚蹼'属性，最后判断这些样本是否是鱼类。最后一列给出了人工分类结果。

序号 ID	不浮出水面是否可以生存No Surfacing?	是否有脚蹼 Flippers?	是否为鱼类 Fish?
1	是 Yes	是 Yes	是 Yes
2	是 Yes	是 Yes	是 Yes
3	是 Yes	否 No	否 No
4	否 No	是 Yes	否 No
5	否 No	是 Yes	否 No



2. 相关基础概念

2.1 信息熵 Entropy

- 概念说明：熵表示混乱的程度，**熵越大，越混乱**，比如一杯浑浊水的熵就比一杯纯净的水熵大.
- 在信息论和概率统计中，设X是一个取有限个值的离散随机变量，其概率分布为：

$$P(X=x_i)=p_i,i=1,2,3,..,n$$

则随机变量X的熵定义为：

$$H(X)=-\sum_{i=1}^{n}p_i\log_2 p_i$$

若$pi=0$，则规定$0log0=0$。需要说明的是，熵只依赖于X的分布，而不依赖于X的值.

- 计算出上面给定的集合D的熵： $H(D)=-\left(-\frac{2}{5}\log_2\frac{2}{5}-\frac{3}{5}\log_2\frac{3}{5}\right)=0.971$
- 推广：多个变量的联合熵，这里给出两个变量X和Y的**联合熵**表达式：

$$H(X,Y)=-\sum_{i=1}^{n}p(x_i,y_i)logp(x_i,y_i)$$

2.2 条件熵 Conditional Entropy

- 概念说明：条件熵 $H(X|Y)$表示在已知随机变量Y的条件下随机变量X的不确定性
- 条件熵的计算公式如下：

$$H(X|Y)=-\sum_{i=1}^{n}p(x_i,y_i)logp(x_i|y_i)=\sum_{i=1}^{n}p(y_i)H(X|Y=y_i)$$

当熵和条件熵中的概率由数据估计得到时，所对应的熵与条件熵分别称为**经验熵**和**经验条件熵**

2.3 信息增益 Information gain

- 概念说明：**信息增益**表示得知特征X的信息而使得类Y的信息的不确定性`减少的程度`。换一个角度解释一下，一杯浑浊的水Y，其熵为$H1$，现在将其中悬浮的一类物质X去除，这杯水的熵下降为$H2$，则物质X对于这杯水的信息增益就为$H1-H2$。
- 特征X对数据集D的信息增益记为$I(D,X)$，计算公式如下：

$$I(D,X)=H(D)-H(D|X)$$

其中$H(D|X)$为特征X给定条件下D的经验条件熵。

- 以X_1表示“不浮出水面是否可以生存”，则

$$\begin{aligned}I(D,X_1)&=H(D)-\left[\frac{3}{5}H(D_1)+\frac{2}{5}H(D_2)\right]\\&=0.971-\left[\frac{3}{5}(-\frac{2}{3}\log_2\frac{2}{3}-\frac{1}{3}\log_2\frac{1}{3})+\frac{2}{5}(-\frac{2}{2}\log_2\frac{2}{2})\right]\\&=0.420\end{aligned}$$

其中D_1,D_2表示D中X_1取“是”和“否”的样本子集。

以X_2表示“是否有脚蹼”，则

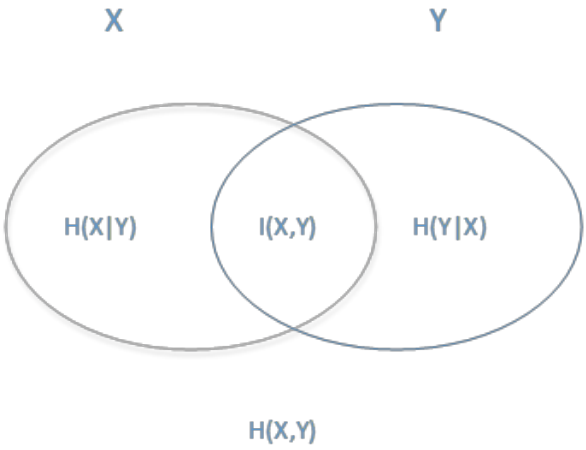
$$
\begin{aligned}
I(D, X_2) &= H(D) - \left[\frac{4}{5}H(D_1) + \frac{1}{5}H(D_2)\right] \\
&= 0.971 - \left[\frac{4}{5}(-\frac{2}{4}\log_2\frac{2}{4} - \frac{2}{4}\log_2\frac{2}{4}) + \frac{1}{5}(-\frac{1}{1}\log_2\frac{1}{1})\right] \\
&= 0.171
\end{aligned}
$$

其中D_1,D_2表示D中X_2取“是”和“否”的样本子集。

比较各个特征的信息增益，X_1的信息增益较大，所以选择X_1作为分类的`最优特征`。

ID3决策树在生成的过程中，根据信息增益来选择特征。

关系梳理：$H(X)$度量了X的不确定性，条件熵$H(X|Y)$度量了我们在知道Y以后X剩下的不确定性，$H(X) - H(X|Y)$为信息增益$I(X, Y)$



左边的椭圆代表$H(X)$,右边的椭圆代表$H(Y)$,中间重合的部分就是我们的互信息或者信息增益$I(X, Y)$, 左边的椭圆去掉重合部分就是$H(X|Y)$,右边的椭圆去掉重合部分就是$H(Y|X)$。两个椭圆的并就是$H(X, Y)$。

2.4 信息增益比 Information gain Ratio

- 引入目的：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。
- 概念说明：信息增益比是信息增益和特征熵的比值。
- 信息增益比计算公式如下：

$$
I_R(D, X) = \frac{I(D, X)}{H_X(D)}
$$

其中D为样本特征输出的集合，X为样本特征，对于特征熵$H_X(D)$, 表达式如下：

$$
H_X(D) = -\sum_{i=1}^{n}\frac{|D_i|}{|D|}\log_2\frac{|D_i|}{|D|}
$$

特征数越多的特征对应的特征熵越大，它作为分母，可以校正信息增益容易偏向于取值较多的特征的问题。

- 以给定的集合D为例，计算信息增益比：

$$H_{X1}(D) = -\left(-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}\right) = 0.971$$

$$I_R(D, X_1) = \frac{I(D, X_1)}{H_{X1}(D)} = \frac{0.420}{0.971} = 0.433$$

$$H_{X2}(D) = -\left(-\frac{4}{5}\log_2\frac{4}{5} - \frac{1}{5}\log_2\frac{1}{5}\right) = 0.722$$

$$I_R(D, X_2) = \frac{I(D, X_2)}{H_{X2}(D)} = \frac{0.171}{0.722} = 0.237$$

根据信息增益比，选择X_1作为分类的最优特征。

C4.5决策树在生成的过程中，根据信息增益比来选择特征。

2.5 基尼指数 GINI index

- 基尼系数代表了模型的不纯度，基尼系数越小，则不纯度越低，特征越好。这和信息增益(比)是相反的。
- 在分类问题中，假设有K个类别，第k个类别的概率为p_k, 则基尼系数的表达式为：

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

如果是`二类`分类问题，计算就更加简单了，如果属于第一个样本输出的概率是p，则基尼系数的表达式为：

$$Gini(p) = 2p(1-p)$$

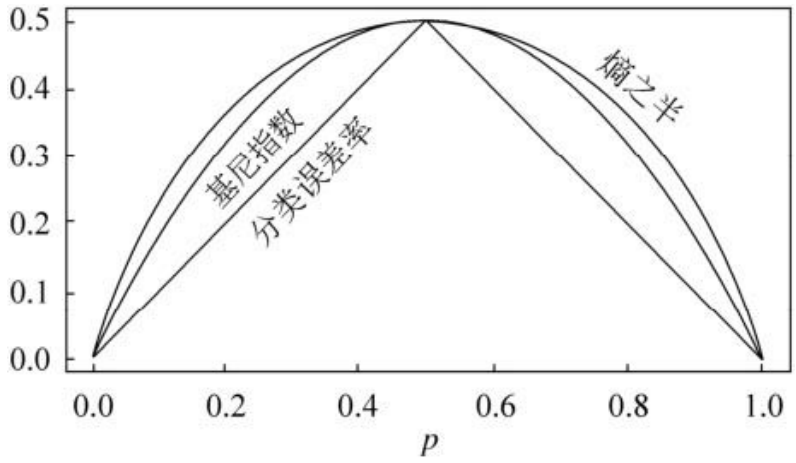
对于个给定的样本D,假设有k个类别, 第k个类别的数量为C_k,则样本D的基尼系数表达式为：

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

特别的，对于样本D,如果根据特征A的某个值a,把D分成D_1和D_2两部分，则在特征A的条件下，D的基尼系数表达式为：

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

和熵模型的度量方式比，基尼系数对应的误差有多大呢？对于二类分类，基尼系数和熵之半的曲线如下：



从上图可以看出，基尼系数和熵之半的曲线非常接近，仅仅在45度角附近误差稍大。因此，基尼系数可以做为熵模型的一个近似替代。而**CART分类树算法就是使用的基尼系数来选择决策树的特征**。同时，为了进一步简化，CART分类树算法每次仅仅对某个特征的值进行二分，而不是多分，这样CART分类树算法建立起来的是二叉树，而不是多叉树。这样一可以进一步简化基尼系数的计算，二可以建立一个更加优雅的二叉树模型。

3. 构建决策树

3.1 决策树生成过程

1. **特征选择**：特征选择是指从训练数据中众多的特征中选择一个特征作为当前节点的分裂标准，如何选择特征有着很多不同量化评估标准标准，从而衍生出不同的决策树算法。
2. **决策树生成**： 根据选择的特征评估标准，从上至下递归地生成子节点，直到数据集不可分则停止决策树停止生长。 树结构来说，递归结构是最容易理解的方式。
3. **剪枝**：决策树容易过拟合，一般来需要剪枝，缩小树结构规模、缓解过拟合.剪枝技术有`预剪枝`和`后剪枝`两种.

在非结束的条件下，首先选择出`合适的特征`，然后根据其分类。分类开始时，记录分类的特征到决策树中，然后在特征标签集中删除该特征，表示已经使用过该特征。`根据选中的特征将数据集分为若干个子数据集`，然后将子数据集作为参数`递归创建决策树`，最终生成一棵完整的决策树。

注：理想条件下，任何到达叶子节点的数据必然属于叶子结点的分类。

3.2 递归结束的条件

- 每个分支下的所有实例都具有`相同的分类`
- 程序遍历完所有划分数据集的`属性`

  遍历完全部属性，划分的数据有可能不全属于一个类，这个时候需要根据**多数表决准则**确定该子数据集的分类

4. 决策树算法

4.1 ID3算法

特点：使用`信息增益`来选择特征，信息增益`大`的优先选择

算法步骤：

1）初始化信息增益的阈值ϵ

2）判断样本是否为同一类输出D_i，如果是则返回单节点树T。标记类别为D_i

3）判断特征是否为空，如果是则返回单节点树T，标记类别为样本中输出类别D实例数最多的类别

4）计算A中的各个特征（一共n个）对输出D的信息增益，选择信息增益最大的特征A_g

5）如果A_g的信息增益小于阈值ϵ，则返回单节点树T，标记类别为样本中输出类别D实例数最多的类别

6）否则，按特征A_g的不同取值A_{gi}将对应的样本输出D分成不同的类别D_i。每个类别产生一个子节点。对应特征值为A_{gi}。返回增加了节点的树T

7）对于所有的子节点，令$D=D_i$,$A=A-\{A_g\}$递归调用2-6步，得到子树T_i并返回

不足：

a）ID3没有考虑`连续特征`，比如长度，密度都是连续值，无法在ID3运用这大大限制了ID3的用途

b）ID3采用信息增益大的特征优先建立决策树的节点。很快就被人发现，在相同条件下，取值比较多的特征比取值少的特征信息增益大（即`用信息增益作为标准容易偏向于取值较多的特征`）。比如一个变量有2个值，各为1/2，另一个变量为3个值，各为1/3，其实他们都是完全不确定的变量，但是取3个值的比取2个值的信息增益大。如果校正这个问题呢?

c）ID3算法对于`缺失值`的情况没有做考虑

d) 没有考虑`过拟合`的问题

4.2 C4.5算法

特点：采用`信息增益比`来选择特征，以减少信息增益容易选择特征值多的特征的问题

算法思路：针对ID3算法的不足，加以改进

a）对于第一个问题，不能处理连续特征， C4.5的思路是将连续的特征离散化。比如m个样本的连续特征A有m个，从小到大排列为$a_1,a_2,\ldots,a_m$,则C4.5取相邻两样本值的平均数，一共取得m-1个划分点，其中第i个划分点T_i表示为：$T_i=\frac{a_i+a_{i+1}}{2}$。对于这m-1个点，分别计算以该点作为二元分类点时的信息增益。选择信息增益最大的点作为该连续特征的二元离散分类点。比如取到的增益最大的点为a_t，则小于a_t的值为类别1，大于a_t的值为类别2，这样我们就做到了连续特征的离散化。要注意的是，与离散属性不同的是，如果当前节点为连续属性，则该属性后面还可以参与子节点的产生选择过程。

b）对于第二个问题，信息增益作为标准容易偏向于取值较多的特征的问题。我们引入一个`信息增益比`的变量$I_R(D,A)$。特征数越多的特征对应的特征熵越大，它作为分母，可以校正信息增益容易偏向于取值较多的特征的问题。

c）对于第三个缺失值处理的问题，主要需要解决的是两个问题，一是在样本某些特征缺失的情况下`选择划分的属性`，二是选定了划分属性，对于在该属性上缺失特征的`样本处理`。

对于第一个子问题，对于某一个有缺失特征值的特征A。C4.5的思路是将数据分成两部分，对每个样本设置一个权重（初始可以都为1），然后划分数据，一部分是有特征值A的数据D_1，另一部分是没有特征A的数据D2。然后对于没有缺失特征A的数据集D_1来和对应的A特征的各个特征值一起计算`加权重后的信息增益比`，最后乘上一个系数，这个系数是无特征A缺失的样本加权后所占加权总样本的比例。

对于第二个子问题，可以将缺失特征的样本同时划分入所有的子节点，不过将该样本的**权重**按各个子节点样本的**数量比例**来分配。比如缺失特征A的样本a之前权重为1，特征A有3个特征值A1,A2,A3。3个特征值对应的无缺失A特征的样本个数为2,3,4.则a同时划分入A1，A2，A3。对应权重调节为2/9,3/9, 4/9。

d）对于第4个问题，C4.5引入了正则化系数进行初步的剪枝。下文介绍CART算法时会详细讨论剪枝的思路。

不足：

- 由于决策树算法非常容易过拟合，因此对于生成的决策树必须要进行剪枝。剪枝的算法有非常多，C4.5的剪枝方法有优化的空间。思路主要是两种，一种是预剪枝，即在生成决策树的时候就决定是否剪枝。另一个是后剪枝，即先生成决策树，再通过交叉验证来剪枝。一般地，常采用"后剪枝加上交叉验证"选择最合适的决策树。
- C4.5生成的是多叉树，即一个父节点可以有多个节点。很多时候，在计算机中二叉树模型会比多叉树运算效率高。如果采用二叉树，可以提高效率。
- C4.5只能用于分类，如果能将决策树用于回归的话可以扩大它的使用范围。
- C4.5由于使用了熵模型，里面有大量的耗时的对数运算,如果是连续值还有大量的排序运算。如果能够加以模型简化可以减少运算强度但又不牺牲太多准确性的话，那就更好了。

4.3 CART算法

特点：使用`基尼系数`来选择特征，简化模型（减少计算量）同时也不至于完全丢失熵模型的优点。

对于连续特征和离散特征处理的改进：

- 对于CART分类树`连续值`的处理问题，其思想和C4.5是相同的，都是将连续的特征离散化。唯一的区别在于在选择划分点时的度量方式不同，C4.5使用的是信息增益比，则CART分类树使用的是基尼系数。
- 对于CART分类树`离散值`的处理问题，采用的思路是不停的二分离散特征。如果某个特征A被选取建立决策树节点，它有A1,A2,A3三种类别，CART分类树会考虑把A分成$\{A_1\}$和$\{A_2, A_3\}$，$\{A_2\}$和$\{A_1, A_3\}$,$\{A_3\}$和$\{A_1, A_2\}$三种情况，找到基尼系数最小的组合，比如$\{A_2\}$和$\{A_1, A_3\}$，然后建立二叉树节点，一个节点是$A2$对应的样本，另一个节点是$\{A1, A3\}$对应的节点。同时，由于这次没有把特征A的取值完全分开，后面我们还有机会在子节点继续选择到特征A来划分A1和A3。这和ID3或者C4.5不同，在ID3或者C4.5的一棵子树中，离散特征只会参与一次节点的建立。

构建CART分类树的算法步骤：算法输入是训练集D，基尼系数的阈值，样本个数阈值；输出是决策树T；从根节点开始，用训练集递归的建立CART树。

1）对于当前节点的数据集为D，如果样本个数小于阈值或者没有特征，则返回决策子树，当前节点停止递归。

2）计算样本集D的基尼系数，如果基尼系数小于阈值，则返回决策树子树，当前节点停止递归。

3）计算当前节点现有的各个特征的各个特征值对数据集D的基尼系数，缺失值的处理和C4.5算法里描述的相同。

4）在计算出来的各个特征的各个特征值对数据集D的基尼系数中，选择基尼系数最小的特征A和对应的特征值a。根据这个最优特征和最优特征值，把数据集划分成两部分D1和D2，同时建立当前节点的左右节点，做节点的数据集D为D1，右节点的数据集D为D2.

5）对左右的子节点递归的调用1-4步，生成决策树。

对于生成的决策树做预测的时候，假如测试集里的样本A落到了某个叶子节点，而节点里有多个训练样本。则对于A的类别预测采用的是这个叶子节点里概率最大的类别（**多数表决准则**）。

构建CART回归树算法：

CART回归树和CART分类树的建立算法大部分是类似的，所以这里我们只讨论CART回归树和CART分类树的建立算法不同的地方。

首先，我们要明白，什么是回归树，什么是分类树。两者的区别在于样本输出，如果样本输出是离散值，那么这是一颗分类树。如果果样本输出是连续值，那么那么这是一颗回归树。

除了概念的不同，CART回归树和CART分类树的建立和预测的区别主要有下面两点：

1)连续值的处理方法不同

2)决策树建立后做预测的方式不同。

对于连续值的处理，我们知道CART分类树采用的是用基尼系数的大小来度量特征的各个划分点的优劣情况。这比较适合分类模型，但是对于回归模型，我们使用了常见的**和方差**的度量方式，CART回归树的度量目标是，对于任意划分特征A，对应的任意划分点s两边划分成的数据集D1和D2，求出使D1和D2各自集合的**均方差最小**，同时D1和D2的**均方差之和最小**所对应的特征和特征值划分点。表达式为：

$$\underbrace{min}_{A,s}\left[\underbrace{min}_{c_1}\sum_{x_i\in D_1(A,s)}(y_i-c_1)^2+\underbrace{min}_{c_2}\sum_{x_i\in D_2(A,s)}(y_i-c_2)^2\right]$$

其中，c1c1为D1数据集的样本输出均值，c2c2为D2数据集的样本输出均值。

对于决策树建立后做预测的方式，上面讲到了CART分类树采用叶子节点里概率最大的类别作为当前节点的预测类别。而回归树输出不是类别，它采用的是用最终叶子的均值或者中位数来`预测输出结果`。

除了上面提到了以外，CART回归树和CART分类树的建立算法和预测没有什么区别。

CART树算法的剪枝

由于决策时算法很容易对训练集`过拟合`，而导致泛化能力差，为了解决这个问题，我们需要对CART树进行剪枝，即类似于线性回归的正则化，来增加决策树的泛化能力。但是，有很多的剪枝方法，我们应该这么选择呢？CART采用的办法是`后剪枝法`，即先生成决策树，然后产生所有可能的剪枝后的CART树，然后使用`交叉验证`来检验各种剪枝的效果，选择泛化能力最好的剪枝策略。

也就是说，CART树的剪枝算法可以概括为两步，第一步是从原始决策树生成`各种剪枝效果`的决策树，第二步是用交叉验证来检验剪枝后的预测能力，选择泛化预测能力最好的、剪枝后的树作为最终的CART树。

首先我们看看剪枝的损失函数度量，在剪枝的过程中，对于任意的一刻子树T,其损失函数为：

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

其中，α为正则化参数，这和线性回归的正则化一样。$C(T_t)$为训练数据的预测误差，分类树是用基尼系数度量，回归树是均方差度量。$|T_t|$是子树T的叶子节点的数量。

当$\alpha = 0$时，即没有正则化，原始的生成的CART树即为最优子树。当$\alpha = \infty$时，即正则化强度达到最大，此时由原始的生成的CART树的根节点组成的单节点树为最优子树。当然，这是两种极端情况。一般来说，α越大，则剪枝剪的越厉害，生成的最优子树相比原生决策树就越偏小。对于固定的α，一定存在使损失函数$C_\alpha(T)$最小的唯一子树。

看过剪枝的损失函数度量后，我们再来看看剪枝的思路，对于位于节点t的任意一颗子树T_t，如果没有剪枝，它的损失是

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

如果将其剪掉，仅仅保留根节点，则损失是

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

当$\alpha = 0$或者α很小时，$C_\alpha(T_t) < C_\alpha(T)$. 当α增大到一定的程度时，$C_\alpha(T_t) = C_\alpha(T)$ 。当α继续增大时不等式反向，也就是说，如果满足下式：$\alpha = \frac{C(T)-C(T_t)}{|T_t|-1}$，$T_t$和$T$有相同的损失函数，但是$T$节点更少，因此可以对子树$T_t$进行剪枝，也就是将它的子节点全部剪掉，变为一个叶子节点T。

最后我们看看CART树的交叉验证策略。上面我们讲到，可以计算出每个子树是否剪枝的阈值α，如果我们把所有的节点是否剪枝的值α都计算出来，然后分别针对不同的α所对应的剪枝后的最优子树做交叉验证。这样就可以选择一个最好的α，有了这个α，我们就可以用对应的最优子树作为最终结果。

CART树的剪枝算法：

- 输入是CART树建立算法得到的原始决策树T

- 输出是最优决策子树T_α

1）初始化$\alpha_{min} = \infty$， 最优子树集合$\omega = \{T\}$。

2）从叶子节点开始自下而上计算各内部节点t的训练误差损失函数$C_\alpha(T_t)$（回归树为均方差，分类树为基尼系数）, 叶子节点数$|T_t|$，以及正则化阈值$\alpha = min\{\frac{C(T)-C(T_t)}{|Tt|-1}, \alpha_{min}\}$, 更新$\alpha_{min} = \alpha$

3) 得到所有节点的α值的集合M。

4）从M中选择最大的值α_k，自上而下的访问子树t的内部节点，如果$\frac{C(T)-C(T_t)}{|T_t|-1} \leq \alpha_k$时，进行剪枝。并决定叶节点t的值。如果是分类树，则是概率最高的类别，如果是回归树，则是所有样本输出的均值。这样得到α_k对应的最优子树T_k

5）最优子树集合$\omega = \omega \cup T_k$， $M = M - \alpha_k$。

6）如果M不为空，则回到步骤4。否则就已经得到了所有的可选最优子树集合ω.

7）采用交叉验证在ω选择最优子树T_α

不足：

- 无论是ID3, C4.5还是CART,在做特征选择的时候都是选择最优的一个特征来做分类决策，但是大

多数，分类决策不应该是由某一个特征决定的，而是应该由一组特征决定的。这样决策得到的决策树更加准确。这个决策树叫做多变量决策树(multi-variate decision tree)。在选择最优特征的时候，多变量决策树不是选择某一个最优特征，而是选择最优的一个特征线性组合来做决策。这个算法的代表是OC1，这里不多介绍。

- 如果样本发生一点点的改动，就会导致树结构的剧烈改变。这个可以通过集成学习里面的随机森林之类的方法解决。

4.4 算法对比

算法	支持模型	树结构	特征选择	连续值处理	缺失值处理	剪枝
ID3	分类	多叉树	信息增益	不支持	不支持	不支持
C4.5	分类	多叉树	信息增益比	支持	支持	支持
CART	分类，回归	二叉树	基尼系数，均方差	支持	支持	支持

5. 小结

决策树优点：

1）简单直观，生成的决策树很直观。

2）基本不需要预处理，不需要提前归一化，处理缺失值。

3）使用决策树预测的代价是$O(log_2 m)$。 m为样本数。

4）既可以处理离散值也可以处理连续值。很多算法只是专注于离散值或者连续值。

5）可以处理多维度输出的分类问题。

6）相比于神经网络之类的黑盒分类模型，决策树在逻辑上可以得到很好的解释

7）可以交叉验证的剪枝来选择模型，从而提高泛化能力。

8） 对于异常点的容错能力好，健壮性高。

决策树缺点：

1）决策树算法易过拟合，导致泛化能力弱。可以通过设置节点最少样本数量\限制决策树深度来改进。

2）决策树会因为样本发生些许改动，便导致树结构的剧烈改变。可以通过集成学习之类的方法解决。

3）寻找最优的决策树是一个NP难的问题，我们一般是通过启发式方法，容易陷入局部最优。可以通过集成学习之类的方法来改善。

4）有些较复杂的关系，决策树很难学习，比如异或。一般这种关系可以换神经网络分类方法来解决。

5）如果某些特征的样本比例过大，生成决策树易偏向于这些特征。可以通过调节样本权重来改善。