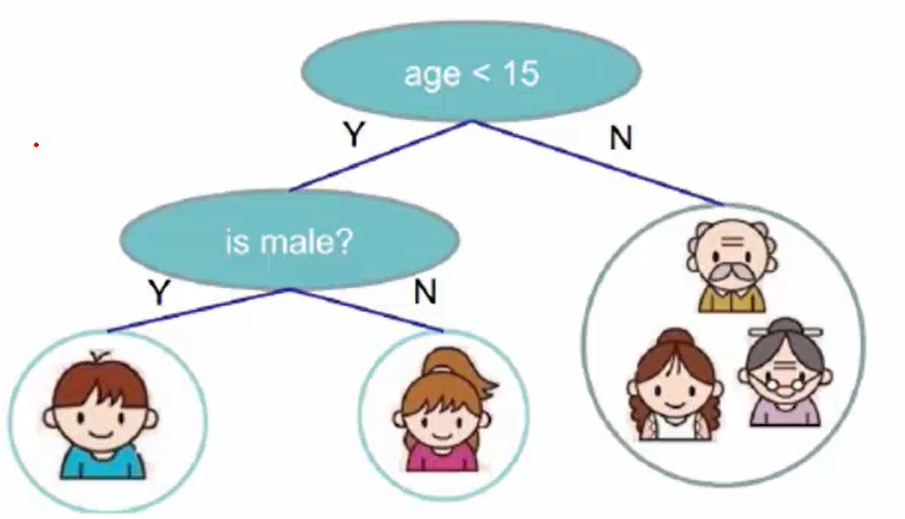
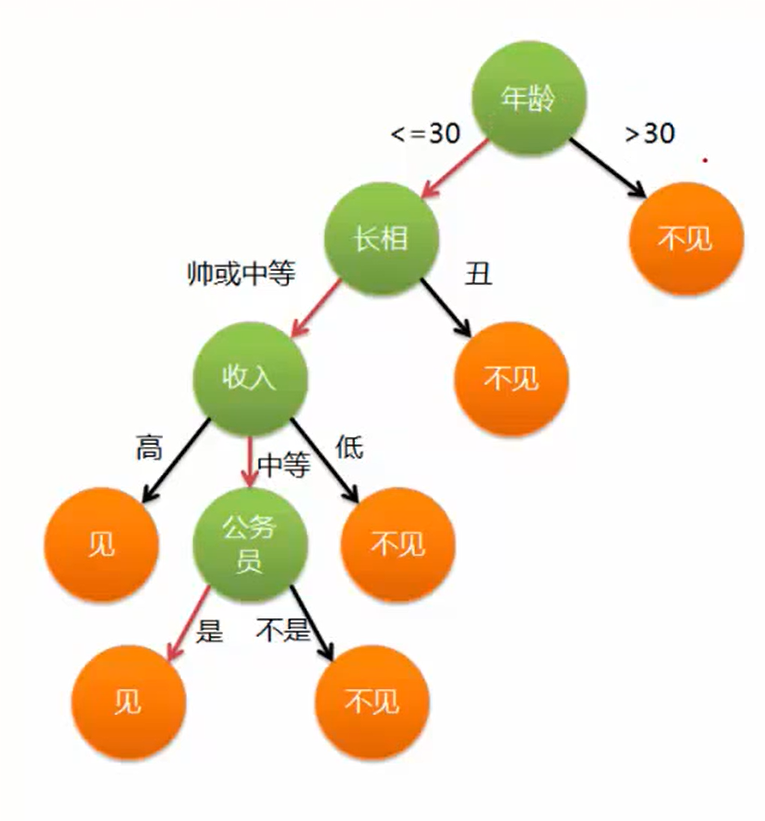
决策树与随机森林（原理）





[决策树](https://www.cnblogs.com/molieren/articles/10664954.html)

在现实生活中，我们会遇到各种选择，不论是选择男女朋友，还是挑选水果，都是基于以往的经验来做判断。如果把判断背后的逻辑整理成一个结构图，你会发现它实际上是一个树状图，这就是我们今天要讲的决策树。



信息熵和gini系数

熵，gini系数 都是在表达数据的混乱程度，只不过它们的计算方式不同

熵，gini系数 不纯度 是同增减的

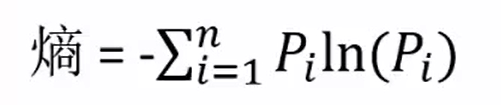
A[1,2,3,1,,2,1] 不纯度高，熵值大

B[1,1,1,1,1,2] 纯度高

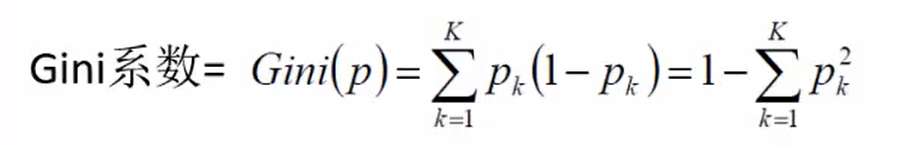
H（X），H（Y）当成它们发生的不确定性

P（X）越大->H（X）值越小 如：今天正常上课

P（X）越小->H（X）值越大

熵： 对这个式子求导，知道随着概率的增大，熵值在递减

Gini系数：

对这个式子求导，知道随着概率的增大，熵值在递减

信息熵和gini系数的提出是为了

根节点的选择：

构造树的基本想法是随着树深度的增加，节点的熵值迅速地降低。熵值降低的速度越快越好，这样我们有望得到一棵高度最矮的决策树。

决策树的工作方式是以一种贪婪（greedy）的方式迭代式地将数据分成不同的子集。其中回归树（regression tree）的目的是最小化所有子集中的 MSE（均方误差）或 MAE（平均绝对误差）；而分类树（classification tree）则是对数据进行分割，以使得所得到的子集的熵或基尼不纯度（Gini impurity）最小。

从信息论的知识中我们知道：信息嫡越大，从而样本纯度越低，。ID3算法的核心思想就是以信息增益来度量特征选择，选择信息增益最大的特征进行分裂。算法采用自顶向下的贪婪搜索遍历可能的决策树空间（C4.5也是贪婪搜索）。其大致步骤为：

1.初始化特征集合和数据集合；

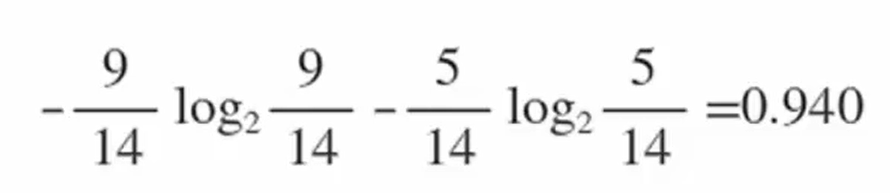
2.计算数据集合信息熵和所有特征的条件熵，选择信息增益最大的特征作为当前决策节点；

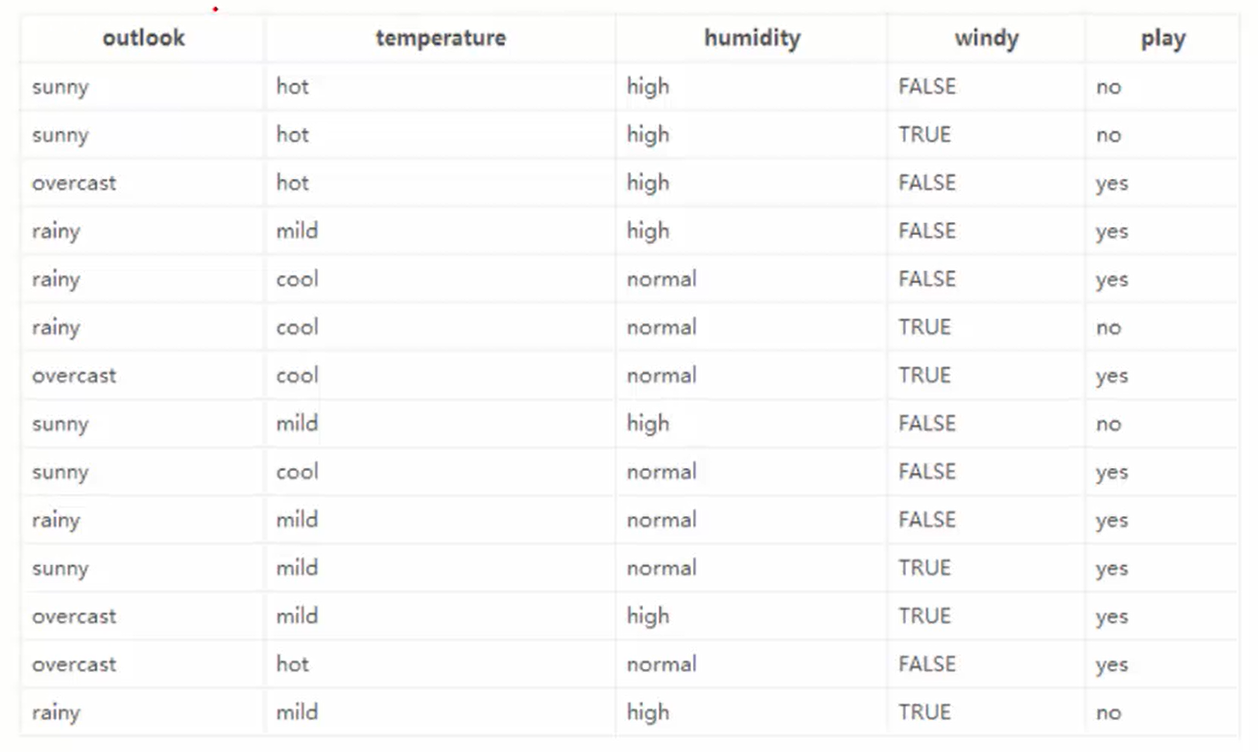
3.更新数据集合和特征集合（删除上一步使用的特征，并按照特征值来划分不同分支的数据集合）；

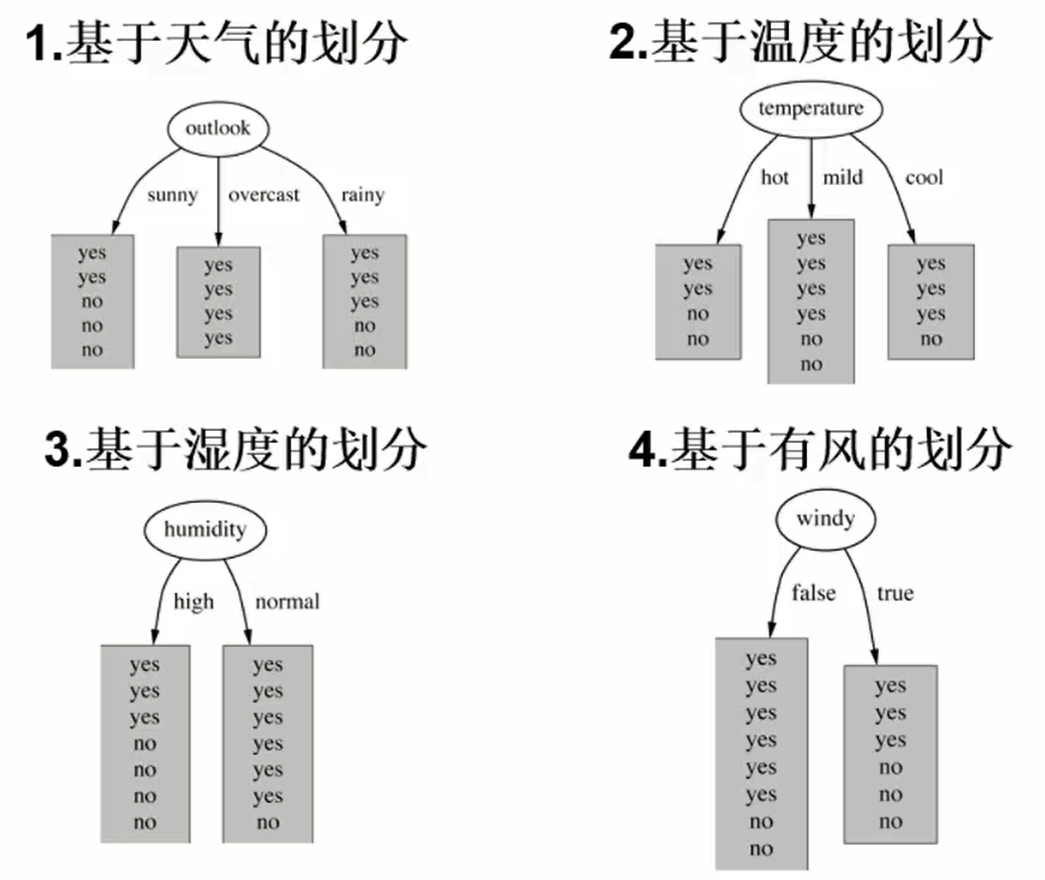
4.重复2，3两步，若子集值包含单一特征，则为分支叶子节点。

例子：是否去打球

在没有给定任何天气信息时，根据历史数据，我们只知道新的一天打球的概率是9/14，不打球的概率是5/14。此时的熵为：

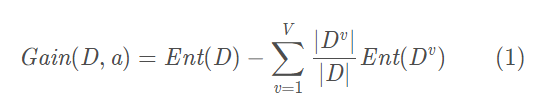






选择哪个作为第一个节点？（目的是让这棵树最矮，也就是更快做出决策）

信息增益的公式



特征有4个：outook，temperature，humidity，windy。我们首先要决定哪个特征作为树的根节点。

对每项指标分别统计：在不同的取值下打球和不打球的次数。

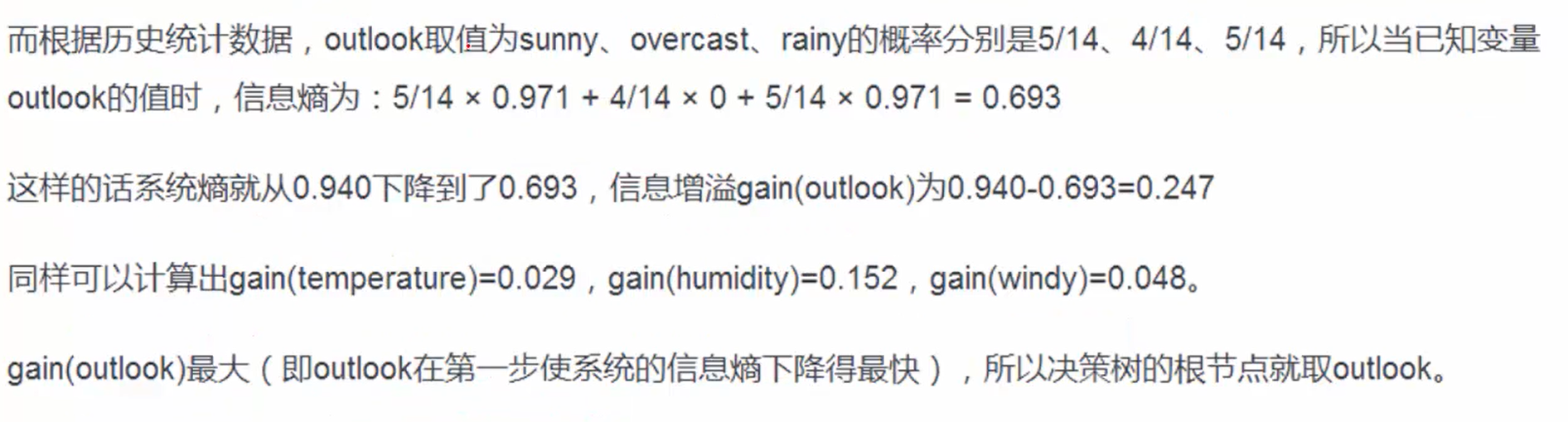
下面我们计算当已知变量outlook的值时，信息熵为多少。

（5）outlook=sunny时，2/5的概率打球，3/5的概率不打球。entropy=0.971

（4）outlook=overcast时，entropy=0 都出去打球了，纯度很高，熵值很低

P（打球）= 1 -> 熵值 = 0

（5）outlook=rainy时，entropy=0.971



而根据历史统计数据，outlook取值为sunny、overcast、rainy的概率分别是5/14、4/14、5/14，所以当节点选择outlook进行决策时，信息熵为：5/14×0.971+4/14×0+5/14×0.971=0.693（加权，选择每个分支的概率\*每个分支的熵值）

这样的话系统熵就从0.940下降到了0.693，信息增溢gain（outlook）为0.940-0.693=0.247同样可以计算出gain（temperature）=0.029，gain（humidity）=0.152，gain（windy）=0.048。

gain（outlook）最大（即outlook作为根节点使系统的信息熵下降得最快），所以决策树的根节点就取outlook。

决策树每进行一次分支，使得熵值下降，纯度上升，每次都选用不纯度下降最快（信息增益最大）的那个作为分支节点

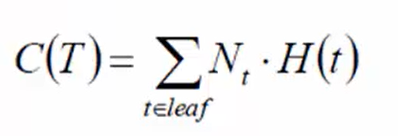
然后递归的构造其他节点 三种情况导致递归返回

1. 当前节点包含的样本完全属于同一类别，无需划分
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
3. 当前节点包含样本集合为空，不能划分

ID3：根据信息增益

C4.5：信息增益率

CART:Gini系数

评价函数：（希望它越小越好，类似损失函数了）

N 是划分给这个叶子节点的样本数

H(t) 这个叶子节点的熵值

ID3的缺点：

如果一类特征的可能取值很多，但是每个可能取值对应的样本又很少，得到的信息增益也很大。

假如我们误把样本的编号也当成了一组特征，那么计算这组特征（实际上是样本的ID，与样本无关）的信息熵，我们会发现是0，信息增益最大，这显然是不对的。（写一组式子）

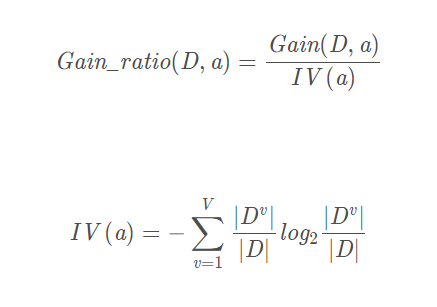
·ID3没有剪枝策略，容易过拟合；

·信息增益准则对可取值数目较多的特征有所偏好，类似“编号”的特征其信息增益接近于1；

·只能用于处理离散分布的特征；

·没有考虑缺失值。

**因此提出C4.5 根据信息增益率去选择需要分支的节点：**



称为特征a的固有值，特征a的可能取值数目越多（即V越大）,则IV（a）的值通常会越大

信息增益率 = 分支后信息增益 / 特征自身的信息熵

A[1,2,3…14] 是ID，式子， 特征自身的信息熵比较大，最后的信息增益率比较小，有效遏制了特征可能性很多，但每种可能性对应的样本又很少这一类特征。

这里需要注意，信息增益率对可取值较少的特征有所偏好（分母越小，整体越大），因此 C4.5 并不是直接用增益率最大的特征进行划分，而是使用一个启发式方法：先从候选划分特征中找到信息增益高于平均值的特征，再从中选择增益率最高的。

C4.5 采用的悲观剪枝方法，用递归的方式从低往上针对每一个非叶子节点，评估用一个最佳叶子节点去代替这课子树是否有益。如果剪枝后与剪枝前相比其错误率是保持或者下降，则这棵子树就可以被替换掉。C4.5 通过训练数据集上的错误分类数量来估算未知样本上的错误率。

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。但同时其训练时间会大的多。

缺点

* 剪枝策略可以再优化；
* C4.5 用的是多叉树，用二叉树效率更高；
* C4.5 只能用于分类；
* C4.5 使用的熵模型拥有大量耗时的对数运算，连续值还有排序运算；
* C4.5 在构造树的过程中，对数值属性值需要按照其大小进行排序，从中选择一个分割点，所以只适合于能够驻留于内存的数据集，当训练集大得无法在内存容纳时，程序无法运行。

## 3. CART（Classification and Regression Tree，分类回归树）

ID3 和 C4.5 虽然在对训练样本集的学习中可以尽可能多地挖掘信息，但是其生成的决策树分支、规模都比较大，CART 算法的二分法可以简化决策树的规模，提高生成决策树的效率。

### 3.1 思想

CART 包含的基本过程有分裂，剪枝和树选择。

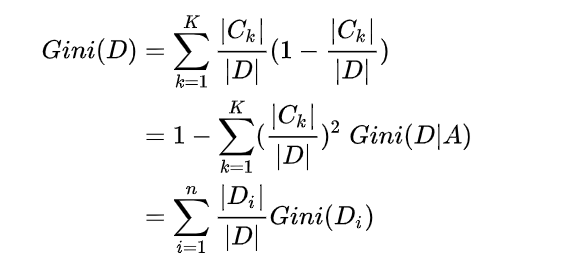
* **分裂：**分裂过程是一个二叉递归划分过程，其输入和预测特征既可以是连续型的也可以是离散型的，CART 没有停止准则，会一直生长下去；
* **剪枝：**采用**代价复杂度剪枝**，从最大树开始，每次选择训练数据熵对整体性能贡献最小的那个分裂节点作为下一个剪枝对象，直到只剩下根节点。CART 会产生一系列嵌套的剪枝树，需要从中选出一颗最优的决策树；
* **树选择：**用单独的测试集评估每棵剪枝树的预测性能（也可以用交叉验证）。

CART 在 C4.5 的基础上进行了很多提升。

* C4.5 为多叉树，运算速度慢，CART 为二叉树，运算速度快；
* C4.5 只能分类，CART 既可以分类也可以回归；
* CART 使用 Gini 系数作为变量的不纯度量，减少了大量的对数运算；
* CART 采用代理测试来估计缺失值，而 C4.5 以不同概率划分到不同节点中；
* CART 采用“基于代价复杂度剪枝”方法进行剪枝，而 C4.5 采用悲观剪枝方法。

### 3.2 划分标准

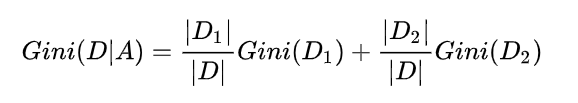
熵模型拥有大量耗时的对数运算，基尼指数在简化模型的同时还保留了熵模型的优点。基尼指数代表了模型的不纯度，基尼系数越小，不纯度越低，特征越好。这和信息增益（率）正好相反。



其中 k 代表类别。

基尼指数反映了从**数据集中随机抽取两个样本，其类别标记不一致的概率**。因此基尼指数越小，则数据集纯度越高。基尼指数偏向于特征值较多的特征，类似信息增益。基尼指数可以用来度量任何不均匀分布，是介于 0~1 之间的数，0 是完全相等，1 是完全不相等，

此外，当 CART 为二分类，其表达式为：



我们可以看到在平方运算和二分类的情况下，其运算更加简单。当然其性能也与熵模型非常接近。

针对过拟合（数据中有噪声）的策略：

预剪枝：在构建决策树的过程时，提前停止。 比如：层数>3 or 样本>50 自动停止

在节点划分前来确定是否继续增长，及早停止增长的主要方法有：

* 节点内数据样本低于某一阈值；
* 所有节点特征都已分裂；
* 节点划分前准确率比划分后准确率高。

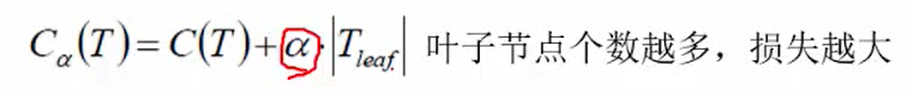
预剪枝不仅可以降低过拟合的风险而且还可以减少训练时间，但另一方面它是基于“贪心”策略，会带来欠拟合风险。

后剪枝：决策树构建好后，然后才开始裁剪。

在已经生成的决策树上进行剪枝，从而得到简化版的剪枝决策树。

例：C4.5 采用的悲观剪枝方法，用递归的方式从低往上针对每一个非叶子节点，评估用一个最佳叶子节点去代替这课子树是否有益。如果剪枝后与剪枝前相比其错误率是保持或者下降，则这棵子树就可以被替换掉。C4.5 通过训练数据集上的错误分类数量来估算未知样本上的错误率。

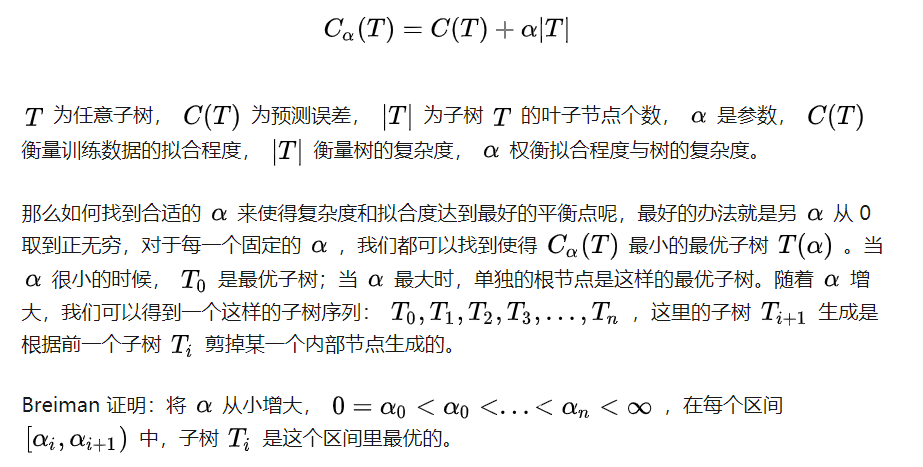
后剪枝决策树的欠拟合风险很小，泛化性能往往优于预剪枝决策树。但同时其训练时间会大的多。

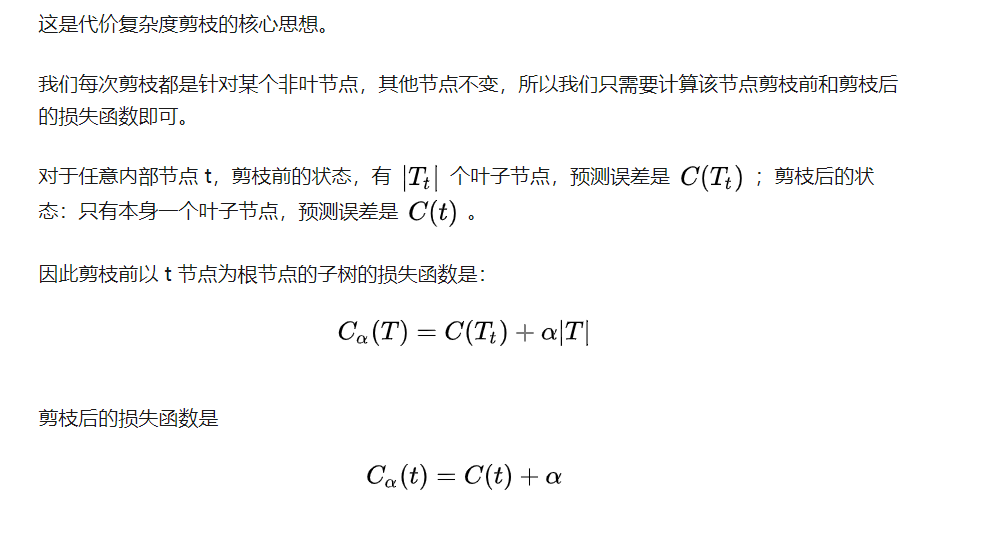
 加入惩罚项，因为叶子节点越多，模型越复杂，越容易过拟合。（加上李航书中的图）

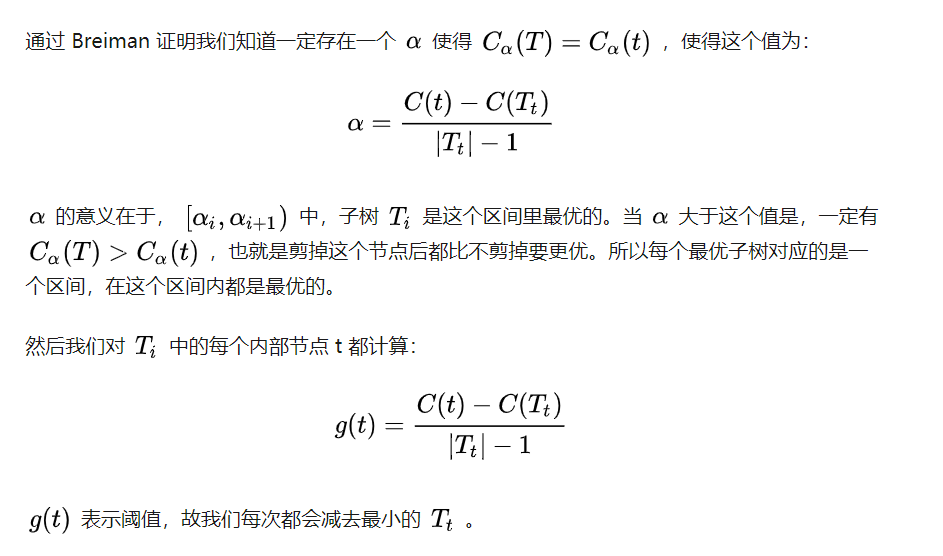
采用一种“基于代价复杂度的剪枝”方法进行后剪枝，这种方法会生成一系列树，每个树都是通过将前面的树的某个或某些子树替换成一个叶节点而得到的，这一系列树中的最后一棵树仅含一个用来预测类别的叶节点。然后用一种成本复杂度的度量准则来判断哪棵子树应该被一个预测类别值的叶节点所代替。这种方法需要使用一个单独的测试数据集来评估所有的树，根据它们在测试数据集熵的分类性能选出最佳的树。

我们来看具体看一下代价复杂度剪枝算法：

首先我们将最大树称为  ，我们希望减少树的大小来防止过拟合，但又担心去掉节点后预测误差会增大，所以我们定义了一个损失函数来达到这两个变量之间的平衡。损失函数定义如下：







总结

最后通过总结的方式对比下 ID3、C4.5 和 CART 三者之间的差异。

除了之前列出来的划分标准、剪枝策略、连续值确实值处理方式等之外，我再介绍一些其他差异：

* 划分标准的差异：ID3 使用信息增益偏向特征值多的特征，C4.5 使用信息增益率克服信息增益的缺点，偏向于特征值小的特征，CART 使用基尼指数克服 C4.5 需要求 log 的巨大计算量，偏向于特征值较多的特征。
* 使用场景的差异：ID3 和 C4.5 都只能用于分类问题，CART 可以用于分类和回归问题；ID3 和 C4.5 是多叉树，速度较慢，CART 是二叉树，计算速度很快；
* 样本数据的差异：ID3 只能处理离散数据且缺失值敏感，C4.5 和 CART 可以处理连续性数据且有多种方式处理缺失值；从样本量考虑的话，小样本建议 C4.5、大样本建议 CART。C4.5 处理过程中需对数据集进行多次扫描排序，处理成本耗时较高，而 CART 本身是一种大样本的统计方法，小样本处理下泛化误差较大 ；
* 样本特征的差异：ID3 和 C4.5 层级之间只使用一次特征，CART 可多次重复使用特征；
* 剪枝策略的差异：ID3 没有剪枝策略，C4.5 是通过悲观剪枝策略来修正树的准确性，而 CART 是通过代价复杂度剪枝。

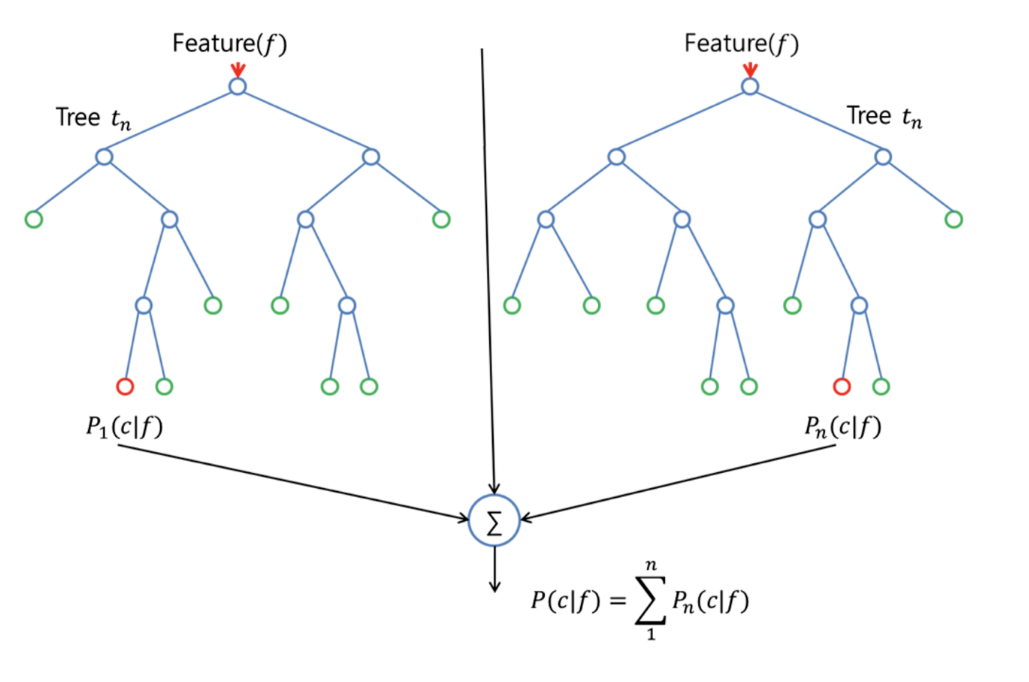
**随机森林**：随机体现在两个方面，1.数据（训练集中的样本）选择随机（有放回的抽样）2.特征选择随机（无放回的抽样）

构建很多树，让每棵树都参与决策，然后综合最后的结果（回归，分类 怎么综合）

**随机森林**是一个监督学习算法。 就像你已经看到它的名字一样，它创建了一个森林，并以某种方式使它成为随机的。 它构建的“森林”是决策树的集合体，大部分时间都是用“装袋（bagging）”方法训练的。 装袋方法的总体思路是，多个模型通过组合可以显著的比单个模型要好, 可以用来增加整体效果。

简单来说：随机森林建立多个决策树并将它们合并在一起以获得更准确和稳定的预测。

随机森林的一大优势是，它可以应用于分类和回归问题，目前大多数机器学习系统都是围绕这两个问题进行的。 我将在分类问题中讨论随机森林，因为分类问题有时被认为是机器学习的基石。 在下面你可以看到两棵树构成随机森林的样子：



## 决策树和随机森林之间的区别：

就像我之前提到的，**随机森林是决策树的集合，但是有一些区别。**

如果您将具有特征和标签的训练数据集输入到决策树中，它将制定一些规则集，这些规则被用于预测。

例如，如果您想预测某人是否会点击在线广告，则可以收集该广告的过去点击人员以及描述其决定的某些功能。如果将特征和标签放入决策树中，它将生成节点和一些规则。然后你可以预测广告是否会被点击。当决策树生成节点和规则时，它通常使用信息增益和基尼指数计算。相比之下，随机森林是随机的。

另一个区别是“深度”**决策树可能会因过拟合而受到影响。随机森林可防止大部分过拟合**，方法是创建随机的特征子集并使用这些子集构建较小的子树。之后，它组合这些子树。请注意，这也会使计算速度变慢，这取决于随机森林构建的子树数量

## 总结

随机森林是一个很好的算法，可以在模型开发初期了解它是如何执行的，并且由于其简单性，很难构建“坏”随机森林。如果您需要在短时间内开发模型，该算法也是一个很好的选择。最重要的是，它可以只是您的特征的重要性。

随机森林在性能方面也很难被击败。当然，你可能总能找到一个性能更好的模型，比如神经网络，但是这些模型通常需要更多的时间来开发。最重要的是，它们可以处理许多不同的特征类型，如二元的，数字的。

总的来说，随机森林是一个（基本上）快速，简单和灵活的工具，尽管它有其局限性

用ppt画个图

<https://zhuanlan.zhihu.com/p/85731206>