一、ID3算法(Iterative Dichotomiser 3 迭代二叉树三代)

基于Occam's Razor(奥卡姆剃刀原理)，越是小型的决策树越优越于大的决策树（也不总是生成最小的树形结构，二是一种启发式算法）。在信息论中，期望信息越小，那么信息增益就越大，从而纯度就越高。ID3的核心思想就是以信息增益来度量属性的选择，选择分裂后信息增益最大的属性进行分裂。该算法采用自顶向下的贪婪搜索遍历可能的决策空间。

ID3核心思想： 在决策树的每个非叶子节点划分之前，先计算每一个属性所带来的信息增益，选择最大信息增益的属性来划分，因为信息增益越大，区分样本的能力就越强，越具有代表性，这是一种自顶向下的贪心策略。

公式：

clipboard.png

1.自顶向下的贪婪搜索遍历可能的决策树空间构造决策树；

2.从“哪一个属性将在树的根节点被测试”开始；

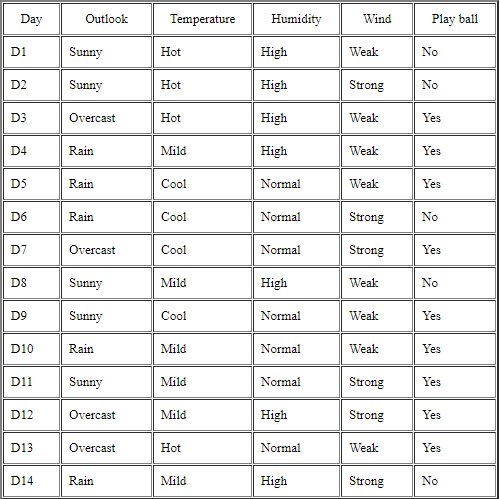
3.使用统计测试来确定每一个属性单独分类训练的能力，把分类能力最好的属性作为根节点（信息增益最大的）；

4.然后为根节点属性的每个可能值产生一个分支，并把训练样例排列到适当的分支（样例的该属性值对应的分支）之下；

5.重复这个过程，用每个分支节点关联的训练样例来选取在该节点被测试的最佳属性。

举例：

最初数据集：



14个样本中9个YES，5个NO，计算出总体的信息熵为：

Entropy(S) = -(9/14)Log2(9/14)-(5/14)Log2(5/14)=0.940286

计算各个属性的信息增益（通过信息熵计算）：

Gain(S,Outlook)=Entropy(S)-Entropy(S|T);

Gain(S,Temperature)=Entropy(S)-Entropy(S|T);

Gain(S,Humidity)=Entropy(S)-Entropy(S|T);

Gain(S,Wind)=Entropy(S)-Entropy(S|T);

其中outlook={sunny,overcast,rain};

temperature={Hot,Mid,Cool};

humidity={High,Normal};

wind={weak,strong};

首先计算Outlook的Entropy(S|T) = Entropy(sunny)+Entropy(overcast)+Entropy(rain)

Entropy(sunny) = -(3/5)Log2(3/5)-(2/5)Log2(2/5)=0.970951

Entropy(overcast) = -(4/4)Log2(4/4)-0\*Log2(0)=0

Entropy(rain) = -(3/5)Log2(3/5)-(2/5)Log2(2/5)=0.970951

=>

Entropy(S|T) = (5/14)\*Entropy(sunny)+(0/14)\*Entropy(overcast)+(5/14)\*Entropy(rain)

=0.693536

=>Gain(S,Outlook) = Entropy(S)-Entropy(S|T) = 0.940286-0.693536=0.246

同理，Temperature的Entropy(S|T) = Entropy(hot)+Entropy(mid)+Entropy(cool)

Entropy(hot) = -(2/4)Log2(2/4)-(2/4)Log2(2/4)=1

Entropy(mid) = -(4/6)Log2(4/6)-(2/6)Log2(2/6)=0.9183

Entropy(cool) = -(3/4)Log2(3/4)-(1/4)Log2(1/4)=0.8113

=>

Entropy(S|T) = (4/14)\*Entropy(hot)+(6/14)\*Entropy(6/14)+(4/14)\*Entropy(4/14)=0.2857+0.3936+0.2318=0.9111

=>Gain(S,Temperature) = Entropy(S)-Entropy(S|T) = 0.940286-0.9111=0.029

同理，Humidity的Entropy(S|T) = Entropy(high)+Entropy(normal)

Entropy(high) = -(3/7)Log2(3/7)-(4/7)Log2(4/7)=0.9852

Entropy(normal) = -(6/7)Log2(6/7)-(1/7)Log2(1/7)=0.5917

=>

Entropy(S|T)=(7/14)\*Entropy(high)+(7/14)\*Entropy(normal)=0.78845

=>Gain(S,Humidity) = Entropy(S)-Entropy(S|T) = 0.152

同理，wind的Entropy(S|T) = Entropy(weak)+Entropy(strong)

Entropy(weak) = -(6/8)Log2(6/8)-(3/6)Log2(3/6)=0.8113

Entropy(strong) = -(3/6)Log2(3/6)-(3/6)Log2(3/6)=1

=>

Entropy(S|T)=(8/14)\*Entropy(weak)+(6/14)\*Entropy(strong)=0.8922

=>Gain(S,wind) = Entropy(S)-Entropy(S|T)=0.048

对比增益：

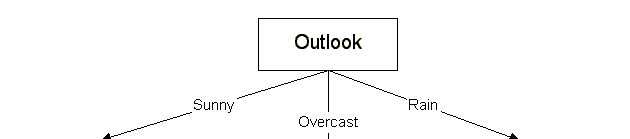
Gain(S,Outlook) = 0.246

Gain(S,Temperature) = 0.029

Gain(S,Humidity) = 0.152

Gain(S,wind) = 0.048

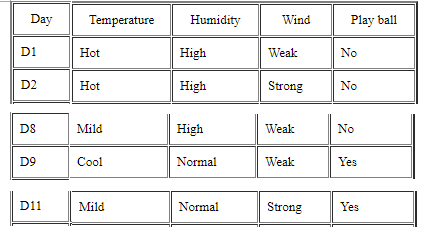
得出信息增益最大的属性为Outlook，所以以Outlook为根节点：



那么第二层的节点怎么确定呢？ 以每个分支的数据集为样本，一次求每个分支下的数据集中剩下几个属性的增益，以sunny分支为例：

S(Sunny) = {D1,D2,D8,D9,D11}

也就是：



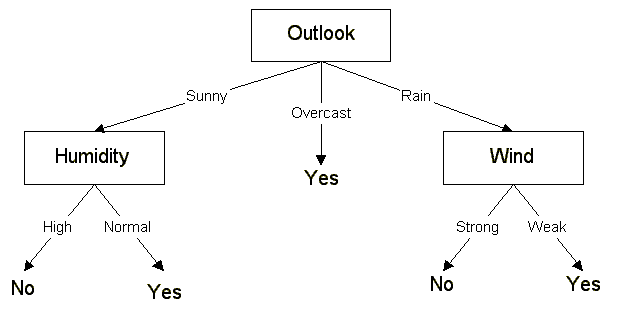
同理按照上述方法求出在这个分支下，剩下三个属性的增益依次为：

Gain(Sunny, Humidity) = 0.970

Gain(Sunny, Temperature) = 0.570

Gain(Sunny, Wind) = 0.019

所以这个分支下的分支节点应该为Humidity. 然后依次递归，继续计算下一层的节点。



参考：https://www.cise.ufl.edu/~ddd/cap6635/Fall-97/Short-papers/2.htm

<https://blog.csdn.net/acdreamers/article/details/44661149>

<https://blog.csdn.net/acdreamers?t=1>

二、C4.5算法

基于ID3算法，是对ID3算法的改进。

<1>用信息增益率来选择分裂属性（克服了ID3中用信息增益来选择会倾向于属性值多的属性的不足）；

<2>能处理连续型的属性（将连续型属性进行离散化处理）；

<3>构造决策树之后进行剪枝操作；

<4>能够处理具有缺失属性值的训练数据。

首先，信息增益率：InfoGainRation(S,A)=Gain(S,A)/SplitInfoA(S)

其中信息增益（与上面ID3计算方法一样）：

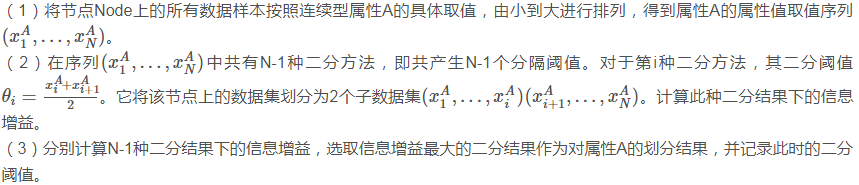
Gain(S,A) = Entropy(S)-Entropy(S|T)

其中属性A的分裂信息：

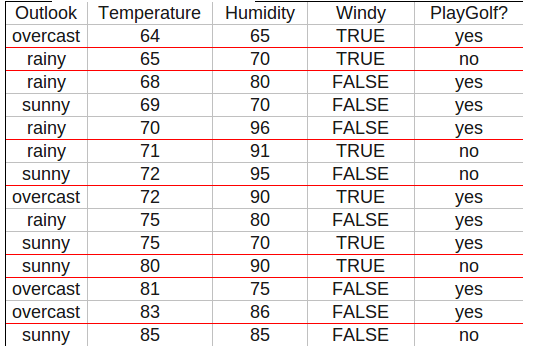
lip_image002.png

其次，连续属性的离散化处理：

离散化的方式有很多，例如 方法1：将属性A的N个属性按升序排序，通过二分法将属性A的所有属性值分成两部分（共有N-1种划分方法，二分的阈值为相邻两个属性的中间值），计算每种划分法对应的信息增益，选取信息增益大的阈值作为属性A的二分阈值：



方法二：对于连续属性进行排序，只有在决策属性发生改变的地方才需要切开，比如对于Temperature：



然后，剪枝：

决策树过于依赖样本，叫做过拟合，因此需要将复杂的决策树进行简化，即去掉一些节点解决过拟合的问题，这个过程就是剪枝；

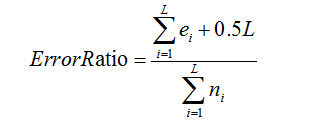
主要有预剪枝和后剪枝两大类；

预剪枝：在构建决策树的过程中，提前终止树的生长，从而避免过多的节点产生。（简单但实用性不强，很难精确判断何时终止树的生长）；

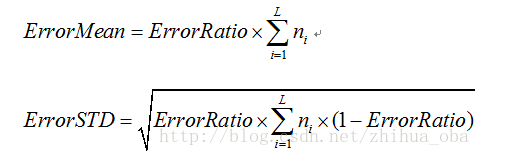
后剪枝：在决策树构建完成后，对那些置信度不达标的节点子树用叶子节点代替，该叶子节点的类标号用该节点子树中频率最高的类标记。（分两种：一类是把训练数据集分成树的生长集和剪枝集；另一类，使用同一数据集进行决策树生长和剪枝）。 常见后剪枝方法：CCP(Cost Compexity Pruning), REP(Reduced Error Pruning),PEP(Pessimistic Error Pruning),MEP(Minimun Error Pruning)

C4.5使用的是PEP剪枝：

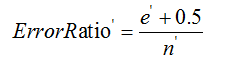
对于一个叶子节点，它覆盖了n个样本，其中有e个错误，那么该叶子节点的错误率就是(e+0.5)/n，其中0.5是一个惩罚因子，对于一棵树，假设有L个叶子节点，那么该树的误判率为：



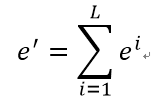
其中ei是第i个叶子节点错误分类的样本数量，ni表示子树第i个叶子节点中样本的总数量。假设一个子树若正确分类一个样本就取1，错误就取0，那么误判次数就是一个伯努利分部（0-1分部），所以子树误判次数的均值和标准差是：

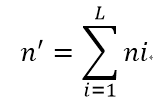


把子树换成叶子节点后，叶子节点的误判率为：



其中





同时叶子节点的误判次数也是一个伯努利分布，因此该叶子节点误判次数均值为：

clipboard.png

剪枝的条件就是：

clipboard.png

满足剪枝条件时，则将所得叶子节点替换该子树，这就是剪枝操作。

缺失属性处理：

在构建决策树的过程中如果训练样本集中有些样本缺少一部分属性，可以这样处理：

计算某属性的信息增益率时忽略掉缺失了此属性的样本；或者通过此属性的样本中出现频率最高的属性值，赋值给缺失了此属性的样本。

当已经选择了某一属性进行分裂，样本该根据该属性的值来进行分支，但对于那些该属性的值为未知的样本，这样处理：

简单的忽略它们；或者根据属性A的其他样本的取值来对未知样本进行分支；或者单独为这样的样本创建一个新的分支（会是模型更复杂）。

在决策树构建完成后，如果待分类样本中有些属性值缺失，这样处理：

待分类样本在到达属性A的分支节点时结束分类过程；或者把待分类的样本属性A分配一个最常见的值，然后继续分类。

参考：<http://www.cnblogs.com/zhangchaoyang/articles/2842490.html>

<https://blog.csdn.net/zhihua_oba/article/details/70632622>

三、K-Means算法

四、Support Vector Machines(SVM)

五、The Apriori algrithm

六、最大期望算法(EM)

七、PageRank

八、AdaBoost

九、KNN: k-nearest neighbor classification

十、Naive Bayes

十一、CART: 分类与回归树