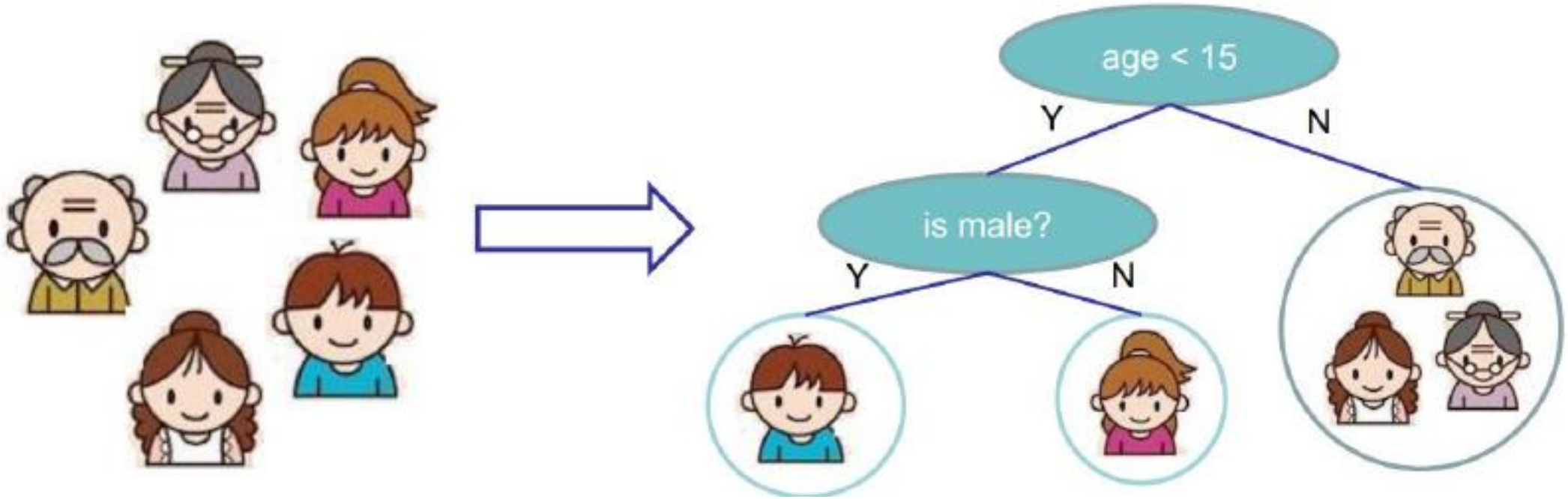
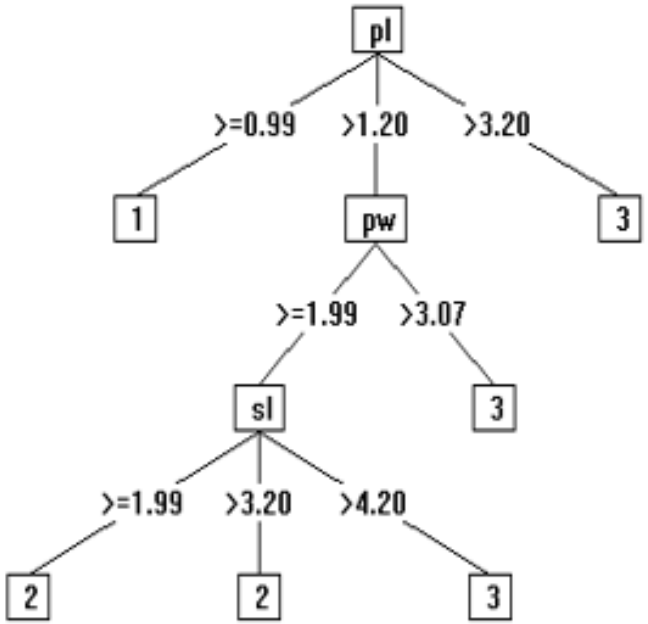
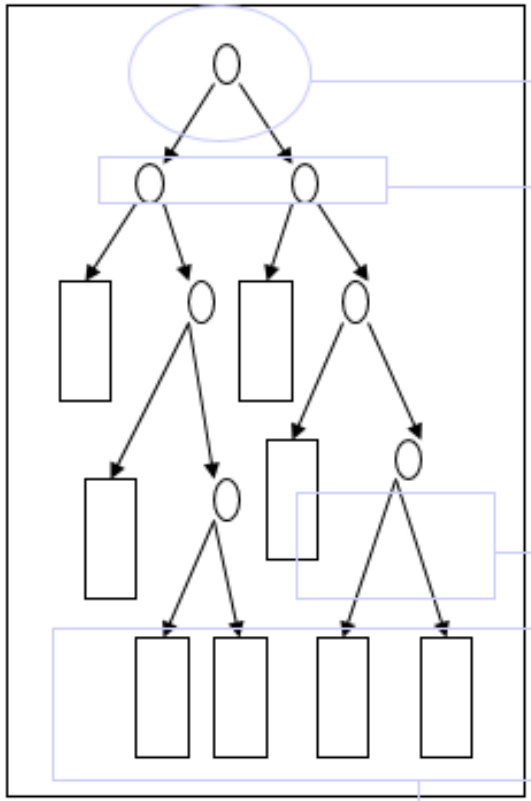
决策树





决策树算法以树状结构表示数据分类的结果。每个决策点实现一个具有离散输出的测试函数，记为分支。

根节点
非叶子节点（决策点）
叶子节点
分支



根部节点(root node)

非叶子节点(non-leaf node)

(代表测试的条件，对数据属性的测试)

分支(branches)(代表测试的结果)

叶节点(leaf node)

(代表分类后所获得的分类标记)

1. 训练阶段

从给定的训练数据集DB，构造出一棵决策树

$$class = DecisionTree(\ DB\ )$$

2. 分类阶段

从根开始，按照决策树的分类属性逐层往下划分，直到叶节点，获得概念（决策、分类）结果。

$$y = DecisionTree(\ x\ )$$

决策树-熵

P(X,Y) = P(X)*P(Y)　X和Y两个事件相互独立　Log(XY) = Log(X)+Log(Y)

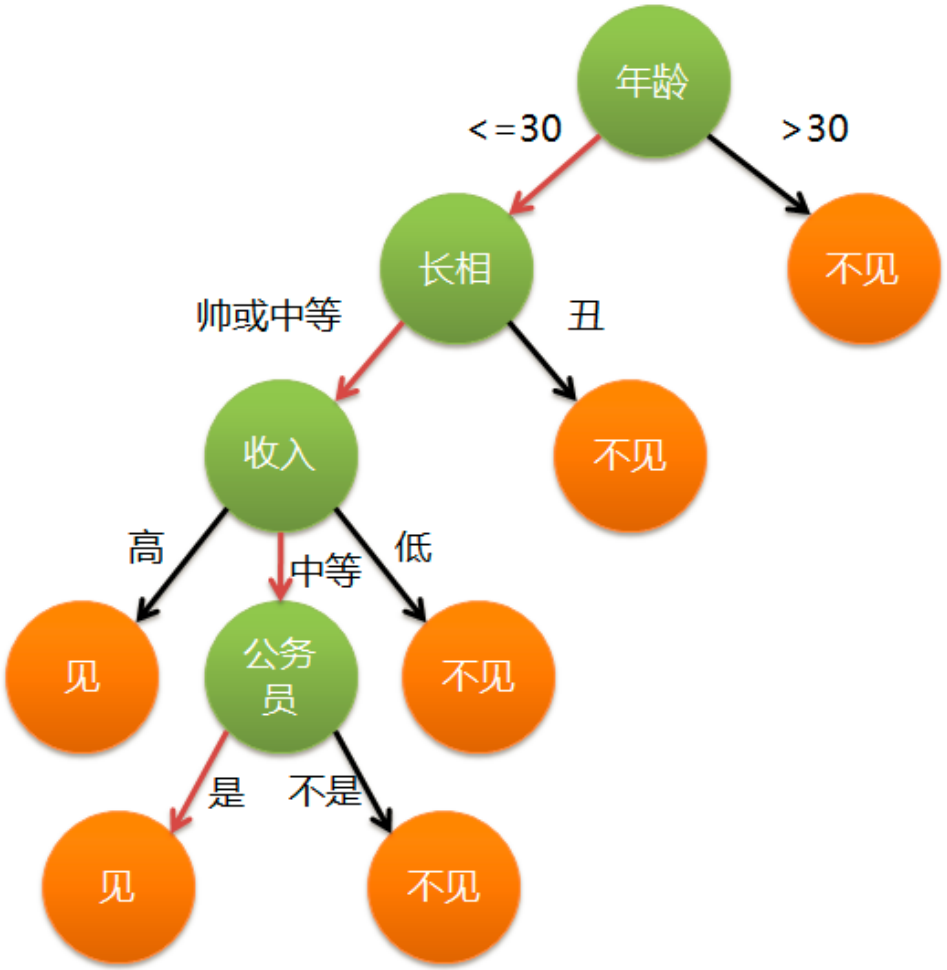
H(X),H(Y)　当成它们发生的不确定性

P(几率越大)->H(X)值越小　如：今天正常上课
P(几率越小)->H(X)值越大　如：今天没翻车

熵 = $-\sum_{i=1}^{n} P_i \ln(P_i)$

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

决策树



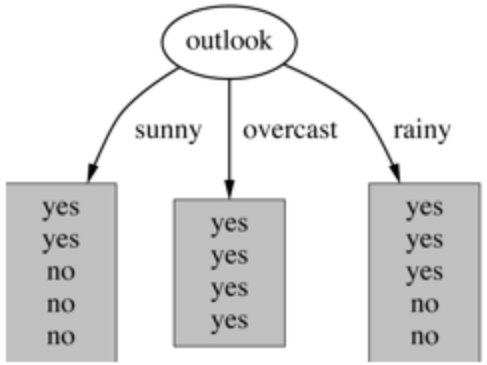
谁当根节点呢？

决策树

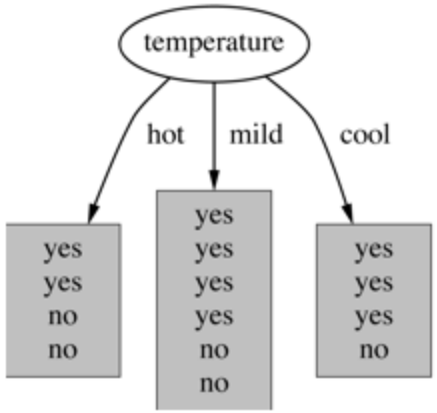
outlook	temperature	humidity	windy	play
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

(14行数据，每个数据4个特征
outlook，temperature，humidity，windy)

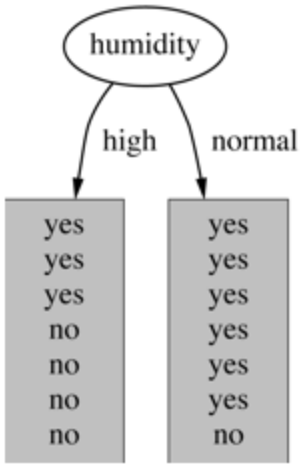
1.基于天气的划分



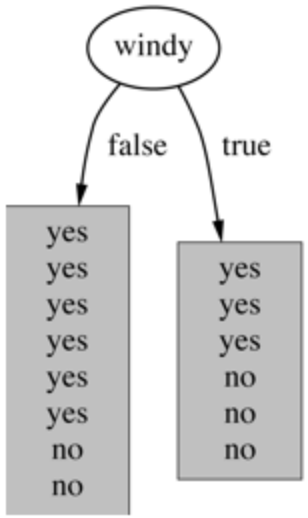
2.基于温度的划分



3.基于湿度的划分



4.基于有风的划分



决策树

熵是无序性（或不确定性）的度量指标。假如事件A的全概率划分是（A1,A2,...,An），每部分发生的概率是(p1,p2,...,pn)，那信息熵定义为：

$$entropy(p_1,p_2,\cdots p_n)=-p_1\log_2 p_1-p_2\log_2 p_2-\cdots-p_n\log_2 p_n$$

决策树

构造树的基本想法是随着树深度的增加，节点的熵迅速地降低。熵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。

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，不打的概率是5/14。此时的熵为：

$$-\frac{9}{14}\log_2\frac{9}{14}-\frac{5}{14}\log_2\frac{5}{14}=0.940$$

决策树

属性有4个：outlook，temperature，humidity，windy。我们首先要决定哪个属性作树的根节点。

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。

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

outlook=overcast时，entropy=0

outlook=rainy时，entropy=0.971

决策树

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，所以当已知变量outlook的值时，信息熵为：5/14 × 0.971 + 4/14 × 0 + 5/14 × 0.971 = 0.693

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在第一步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

接下来要确定N1取temperature、humidity还是windy?在已知outlook=sunny的情况，根据历史数据，我们作出类似table 2的一张表，分别计算gain(temperature)、gain(humidity)和gain(windy)，选最大者为N1。

依此类推，构造决策树。当系统的信息熵降为0时，就没有必要再往下构造决策树了，此时叶子节点都是纯的--这是理想情况。最坏的情况下，决策树的高度为属性（决策变量）的个数，叶子节点不纯（这意味着我们要以一定的概率来作出决策）。

决策树

ID3：信息增益

C4.5：信息增益率

CART：Gini系数

评价函数：$C(T)=\sum_{t\in leaf} N_t \cdot H(t)$ (希望它越小越好,类似损失函数了)

- C4.5算法是ID3算法的扩展
- 能够处理连续型的属性。首先将连续型属性离散化，把连续型属性的值分成不同的区间，依据是比较各个分裂点Gian值的大小。
- 缺失数据的考虑：在构建决策树时，可以简单地忽略缺失数据，即在计算增益时，仅考虑具有属性值的记录。

选取(连续值的)哪个分界点？

- 贪婪算法！

1. 排序

60 70 75 85 90 95 100 120 125 220

若进行“二分”，则可能有9个分界点。

例子：

60 70 75 85 90 95 100 120 125 220

分割成TaxIn<=80 和 TaxIn>80

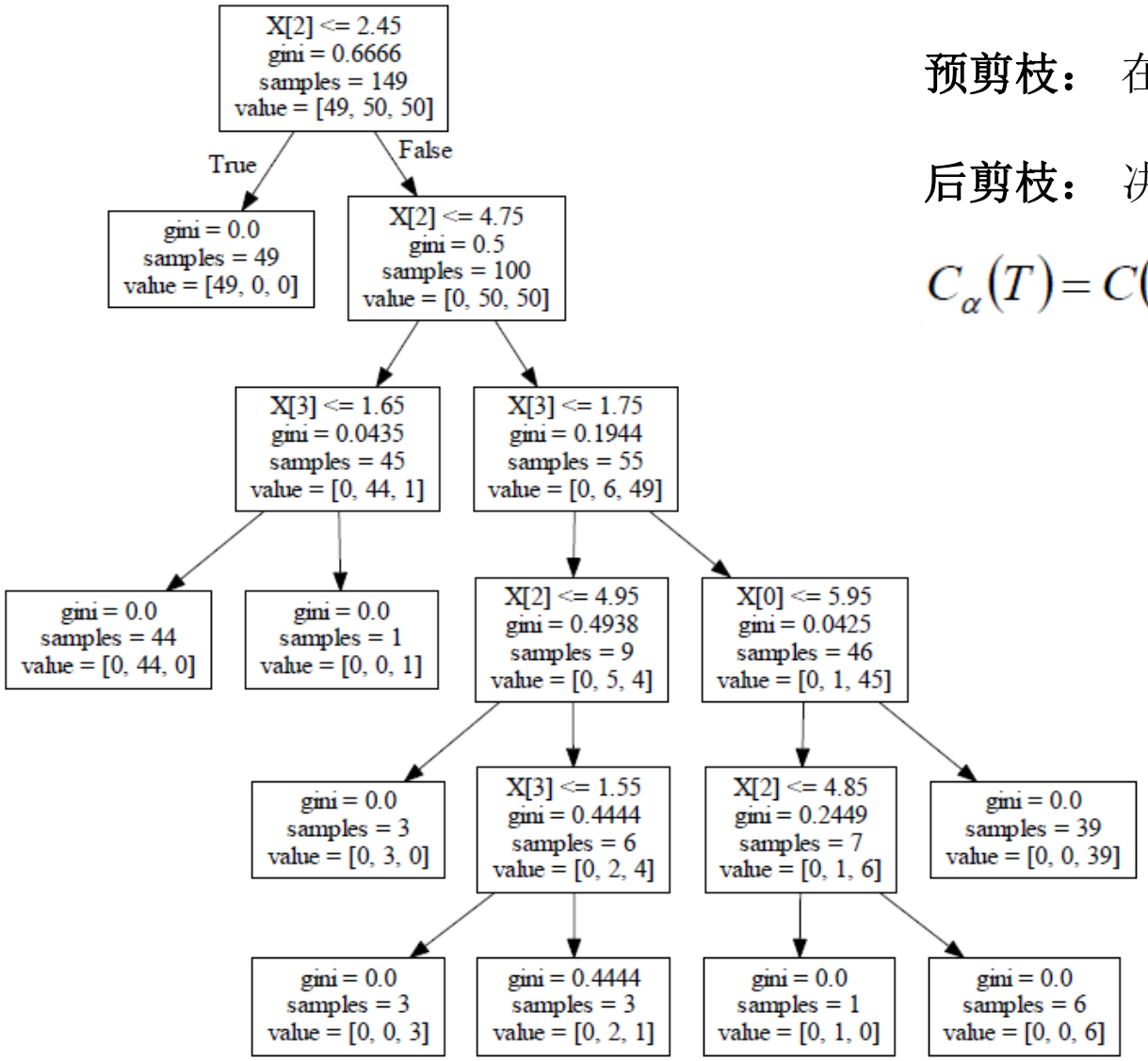
60 70 75 85 90 95 100 120 125 220

分割成TaxIn<=97.5 和 TaxIn>97.5

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- 实际上，这就是“离散化”过程

决策树



预剪枝： 在构建决策树的过程时，提前停止。

后剪枝： 决策树构建好后，然后才开始裁剪。

$C_\alpha(T) = C(T) + \alpha \cdot |T_{leaf}|$ 叶子节点个数越多，损失越大

```
dot -Tpdf iris.dot -o iris.pdf
```

决策树

Bootstraping:有放回采样
Bagging: 有放回采样n个样本一共建立分类器