北京航空航天大學
BEIHANG UNIVERSITY

人工智能理论与应用实践

监督学习—分类

刘禹 计算机学院
buaa_liuyu@buaa.edu.cn

本讲重点

- 分类问题
- 问题转化的思想
- KNN算法
- 决策树算法

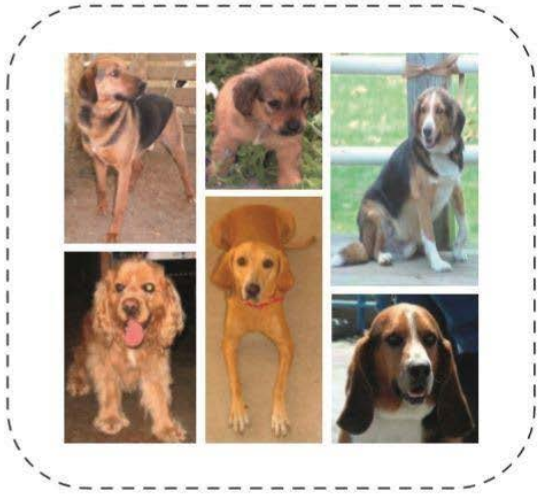
分类问题

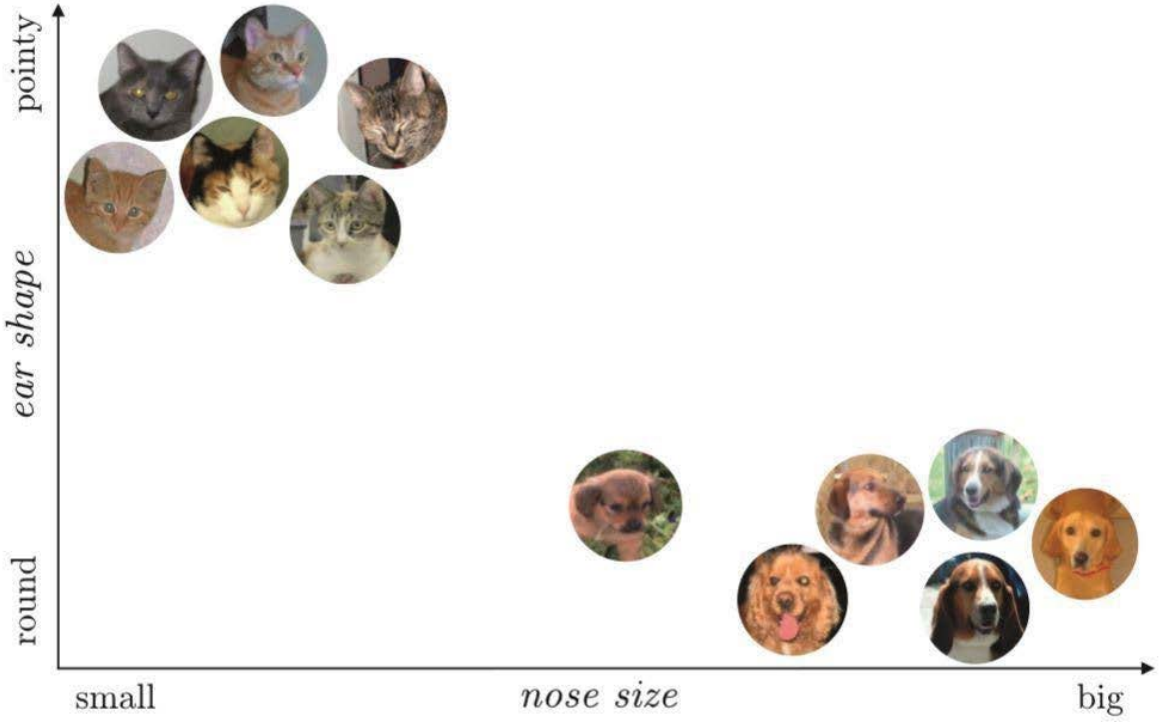
找一个函数判断输入数据所属类别的任务就称为分类。可以是二类别问题（是/不是），也可以是多类别问题。

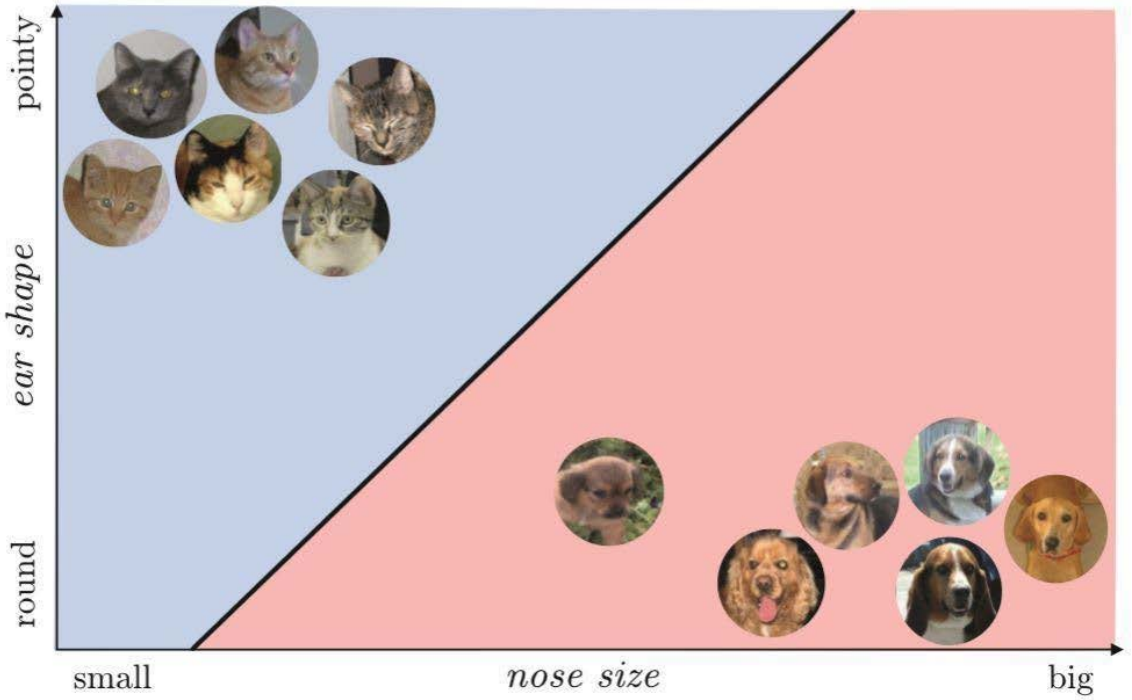
分类问题的输出是离散值，用来指定其属于哪个类别

实际分类任务

计算机区分猫和狗

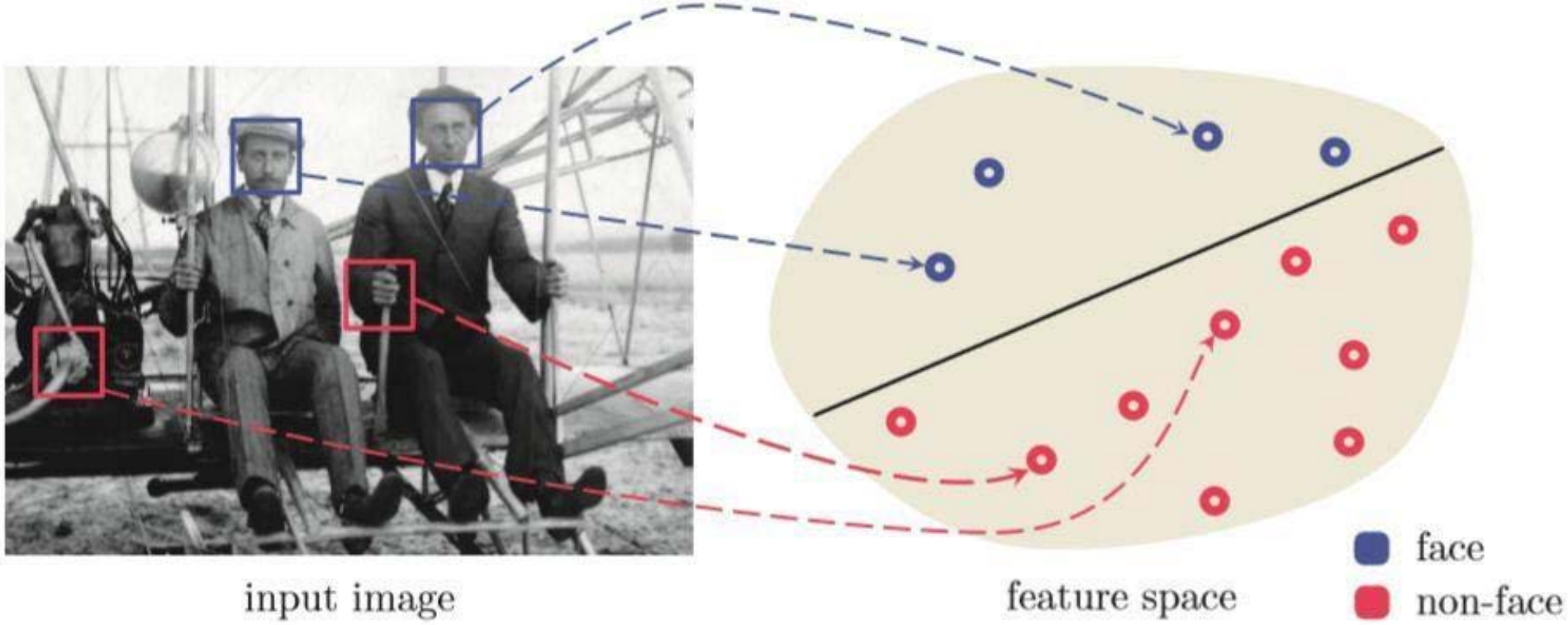






实际分类任务

计算机区分人脸和非人脸



莱特兄弟

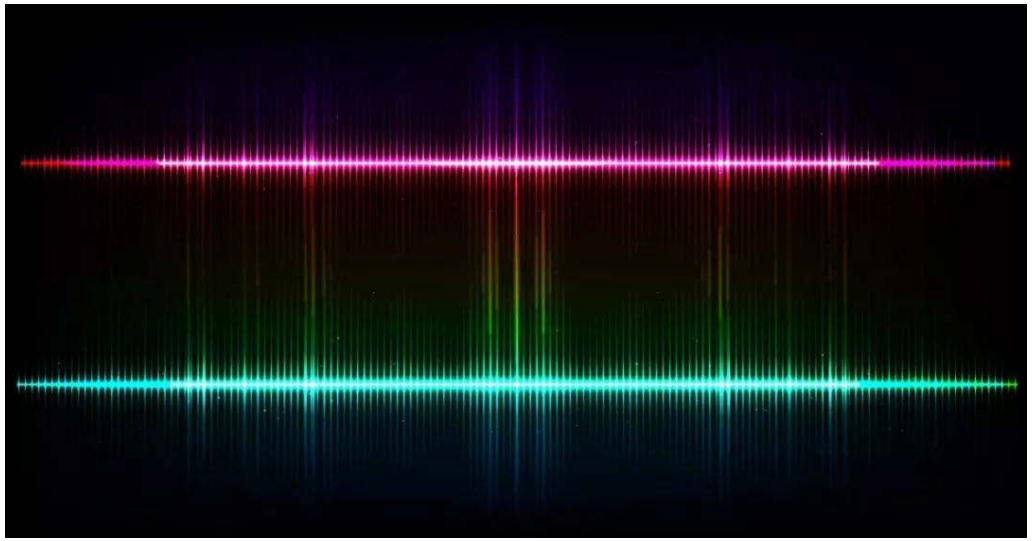
实际分类任务

情感分析

分类模型通常用于情感分析，学习辨别消费者的正面或负面情绪数据。

医学诊断工具

利用人脑功能性核磁共振成像（fMRI），基于fMRI扫描就能区分具有特定神经系统疾病的患者和没有此疾病的患者。

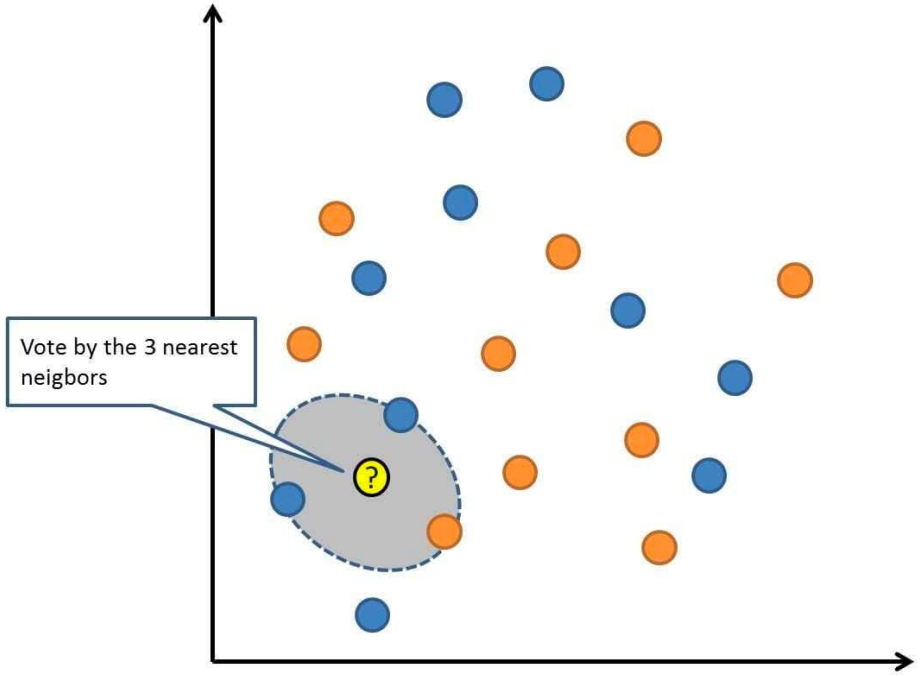


什么是KNN算法

近朱者赤，近墨者黑！

KNN（K-NearestNeighbor）算法的输入为实例的特征向量，对应于特征空间的点；输出为实例的类别。

KNN算法假设给定一个训练数据集，其中的实例类别已定。分类时，对新的实例，根据其 k 个最近邻的训练实例的类别，通过多数表决等方式进行预测



KNN算法执行步骤

1. 将数据分为训练数据和测试数据

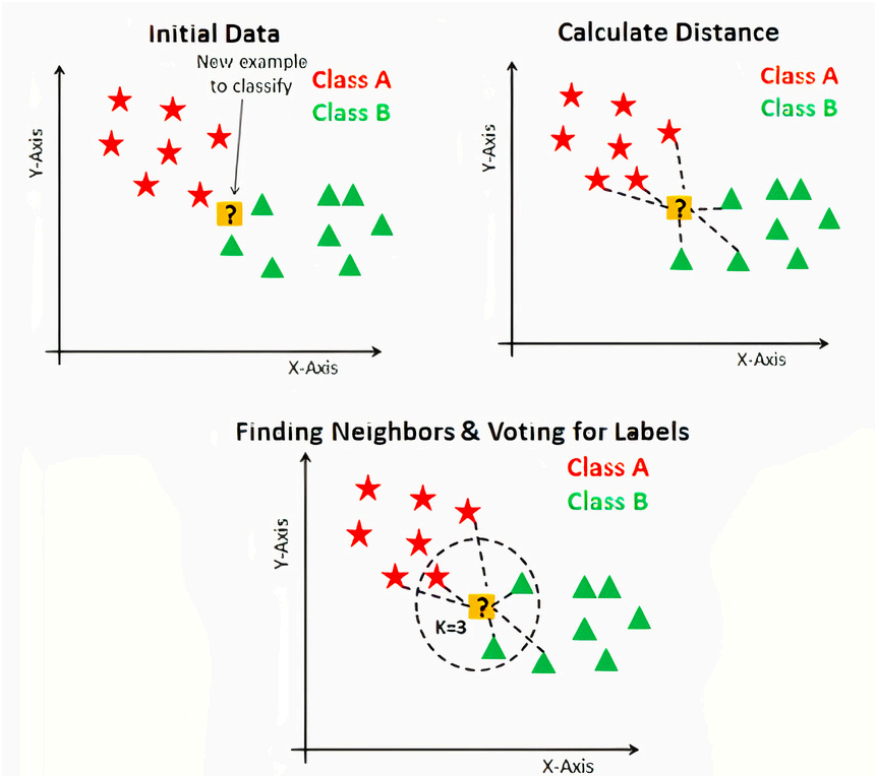
2. 确定要使用的距离函数

3. 选择一个值K

4. 从测试数据中选择一个需要分类的样本，
计算到它的n个训练样本的距离

5. 对获得的距离进行排序，并取k-nearest数据样本

6. 根据其k个邻居的多数票将测试类分配给该类



KNN算法分类示例

项目概述

背景：海伦使用约会网站寻找约会对象。经过一段时间之后，她发现曾交往过三种类型的人:

- 魅力一般的人（分类1）
- 极具魅力的人（分类2）
- 不喜欢的人（分类3）

目标：海伦收集到了一些约会网站未曾记录的数据信息，希望根据这些信息将可能的约会对象

分为以上三类，以实现:

- 工作日与魅力一般的人约会
- 周末与极具魅力的人约会
- 不喜欢的人则直接排除掉

KNN算法分类示例

以下为8名对象的里程数（km/年）游戏时间百分比（%）消费冰淇淋公升数（L/周）类型（enum）数据集

序号	里程数	游戏时间	消费冰淇淋	类型
1	40920	8.326976	0.953952	3
2	14488	7.153469	1.673904	2
3	26052	1.441871	0.805124	1
4	75136	13.147394	0.428964	1
5	38344	1.669788	0.134296	1
6	72993	10.141740	1.032955	1
7	35948	6.830792	1.213192	3
8	42666	13.276369	0.543880	3

测试集

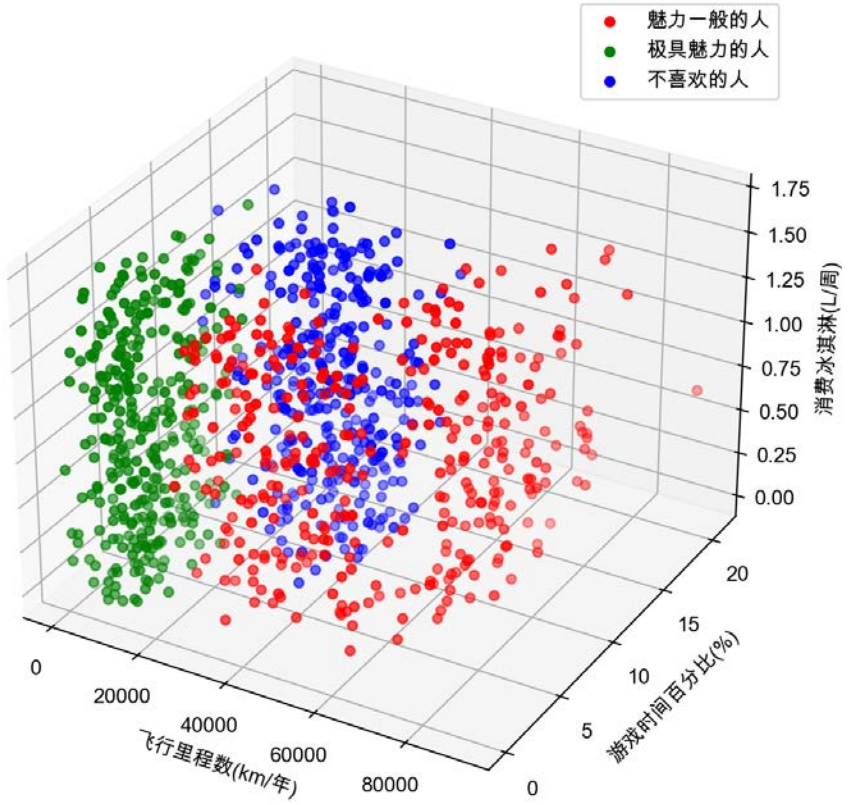
- 魅力一般的人（分类1）
- 极具魅力的人（分类2）
- 不喜欢的人（分类3）

一般需要将全部数据集分为训练集（80%）和测试集（20%），在本例中可将数据集分为6名对象的训练集和2名对象的测试集

KNN算法分类示例

目标：建立预测模型，给出一个人的飞行里程数、游戏时间百分比和每周消费的冰淇淋公升数，
给出海伦眼里该对象的魅力类型

如图，图中点的坐标为该对象的
每年飞行里程数，游戏时间百分比和
每周消费的冰淇淋公升数，不同颜色的点
代表不同魅力类型的对象



KNN算法距离度量

欧几里得距离：两点之间最短直线的长度

$$d(x,y)=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}$$

曼哈顿距离：绝对差之和

$$d(x,y)=\sum_{i=1}^{n}|x_i-y_i|$$

闵可夫斯基距离：p=1时是曼哈顿距离
p=2时是欧氏距离

$$d(x,y)=\left(\sum_{i=1}^{n}|x_i-y_i|^p\right)^{\frac{1}{p}}$$

余弦相似度：两向量夹角余弦值

$$d(x,y)=\cos(\Theta)=\frac{\sum_{i=1}^{n}x_iy_i}{\sqrt{\sum_{i=1}^{n}x_i^2}\sqrt{\sum_{i=1}^{n}y_i^2}}$$

归一化

什么是归一化：将一列数据变化到某个固定区间（范围）中，通常，这个区间是[0,1]，广义的讲，可以是各种区间，比如映射到[0,1]一样可以继续映射到其他范围，图像处理中可能会映射到[0,255],其他情况可能映射到[-1,1]

为什么要归一化： 为了消除特征间单位和尺度差异的影响，以对每维特征同等看待

如何归一化：

min-max归一化：

$$x' = \frac{x - min(x)}{max(x) - min(x)}$$

means归一化：

$$x' = \frac{x - mean(x)}{max(x) - min(x)}$$

其中mean(x),min(x),max(x)分别是样本数据的平均值，最小值和最大值

归一化示例

以示例数据集为例，对象每年的飞行里程数均在10e5量级，而游戏时间百分比在10e1量级，每周消费的冰淇淋在10e-1量级。可见三个维度由于计量单位不同，各自数据的量级差别太大，若用原始数据计算欧氏距离，则飞行里程数会对距离产生主导影响，从而导致模型精度下降

归一化前

里程数	游戏时间	消费冰淇淋
40920	8.326976	0.953952
14488	7.153469	1.673904
26052	1.441871	0.805124
75136	13.147394	0.428964
…	…	…

归一化后

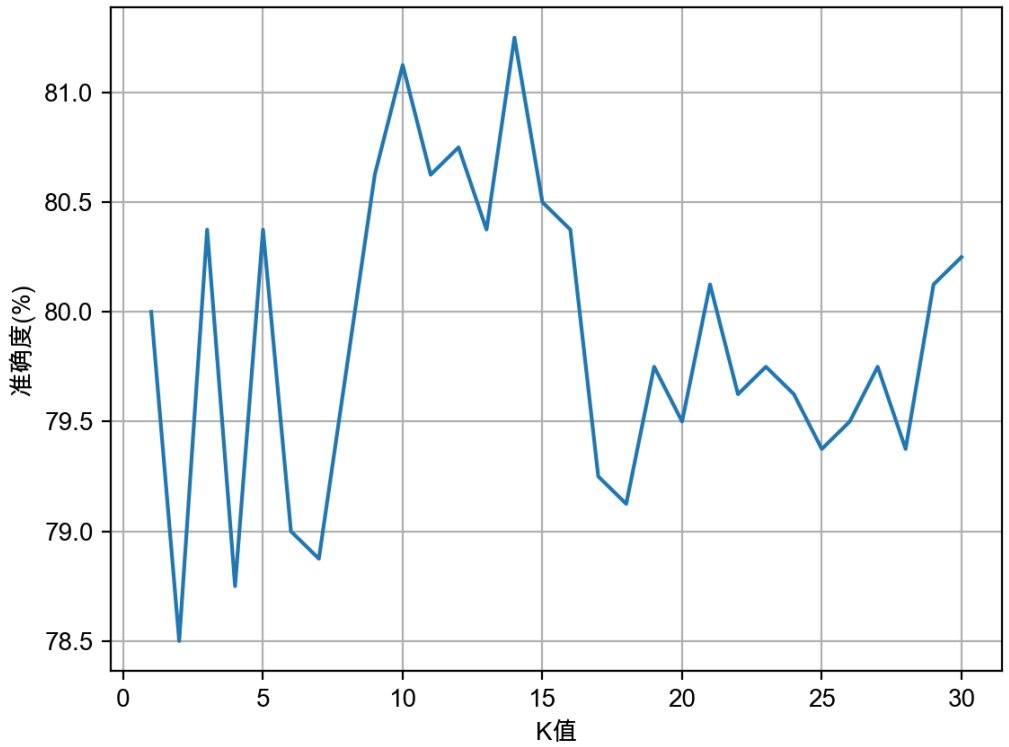
里程数	游戏时间	消费冰淇淋
0.4358264	0.5817826	0.53238
0.0	0.482623	1.0
0.190674	0.0	0.4357135
1.0	0.9891018	0.19139
…	…	…

归一化效果

归一化前

准确度普遍较低

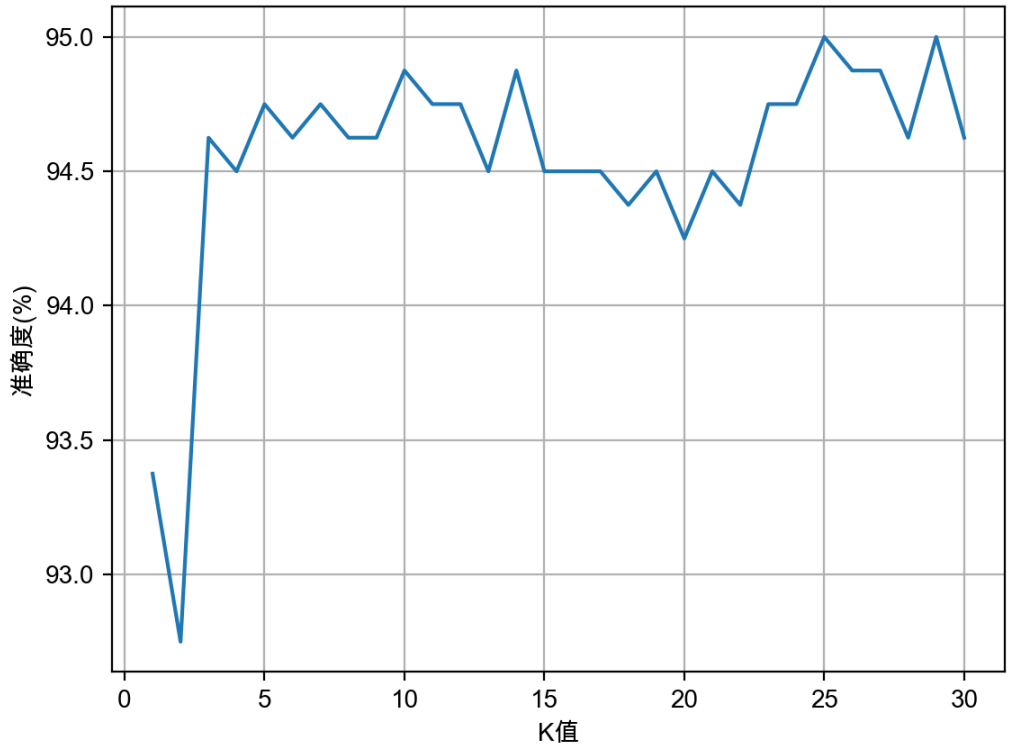
准确度波动幅度大



归一化后

准确度普遍较高

准确度波动幅度小



KNN算法示例

以7号对象（0.353845, 0.460374, 0.70076）为例，计算出它到训练集中所有点的欧式距离

序号	里程数	游戏时间	消费冰淇淋	类型
1	0.4358264	0.5817826	0.53238	3
2	0.0	0.482623	1.0	2
3	0.190674	0.0	0.4357135	1
4	1.0	0.9891018	0.19139	1
5	0.393352	0.01947	0.0	1
6	0.964665	0.7432277	0.5836934	1

distance(1,7) = 0.2231786369316776

distance(2,7)

$= \sqrt{(0.353845-0.0)^2+(0.460374-0.482623)^2+(0.70076-1.0)^2}$

= 0.4639459878326354 6

distance(3,7) = 0.5557145330826341

distance(4,7) = 0.9780220689830267

distance(5,7) = 0.8288677324308144

distance(6,7) = 0.6832363257608965

KNN算法示例

1-6号对象按照与7号对象的距离升序排列可得：1（类型3），2（类型2），3（类型1），6（类型1），5（类型1），4（类型1）

此时若取k=1，则预测7号对象为类型3；

若取k=3，则预测7号对象为类型3；

若取k=5，则预测7号对象为类型1；

…

K 应取何值?

KNN算法K的正确取值

K=1或较小值时，只有与输入实例较近的（相似的）训练实例才会对预测结果起作用。
但缺点是预测结果对近邻的实例点非常敏感。如果邻近的实例点恰巧是噪声，预测就会出错。
换句话说，k值的减小就意味着整体模型变得复杂，容易发生过拟合。

K很大的话，在训练集中占比很大的数据贡献会比一般数据大，存在以偏概全的缺陷

建议K值取不大于20的奇数

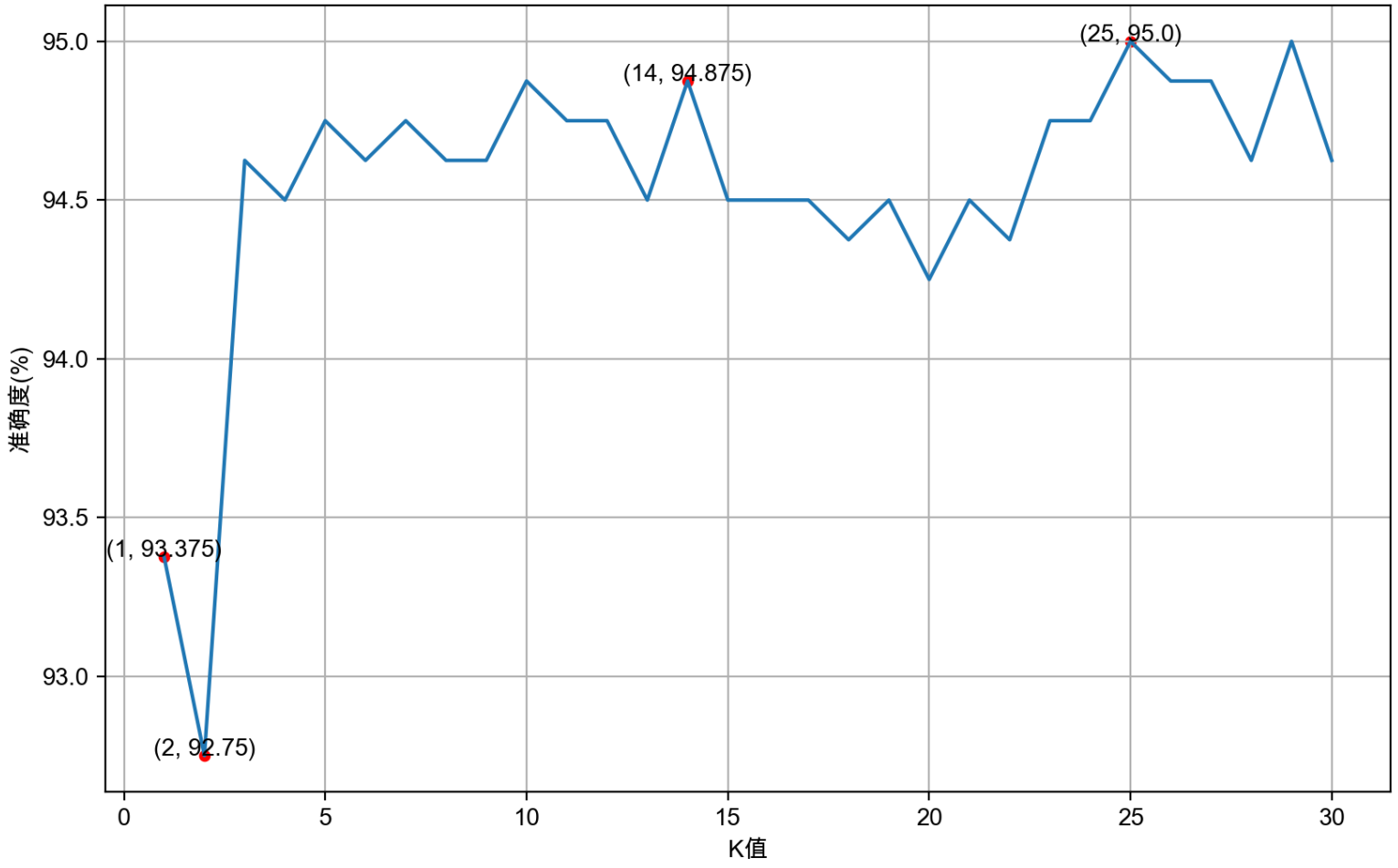
KNN算法K的正确取值

具体做法：K-Fold交叉验证（调参）

先令KNN中的K=1， K-Fold中的K=4，则将训练集分为四个相等大小的部分，分别为D1，D2，D3，D4。在第一次折叠中，使用D1，D2，D3作为训练数据，在D4上计算出模型精度a4，第二次折叠中，使用D1，D2，D4作为训练数据，在D3上计算出模型精度a3，以此类推，最后取a1，a2，a3，a4的平均值作为KNN中K=1的精度。再依次计算K=2，K=3，k=…时的精度，并取精度最高的K

Tips: 当数据量较少时，可以适当增加K-Fold中的K

K-Fold交叉验证示例



左图展示了不同的k值在10-Fold交叉验证中的准确度，则由此结果，我们可以将KNN中的K值确定为25

KNN的局限性

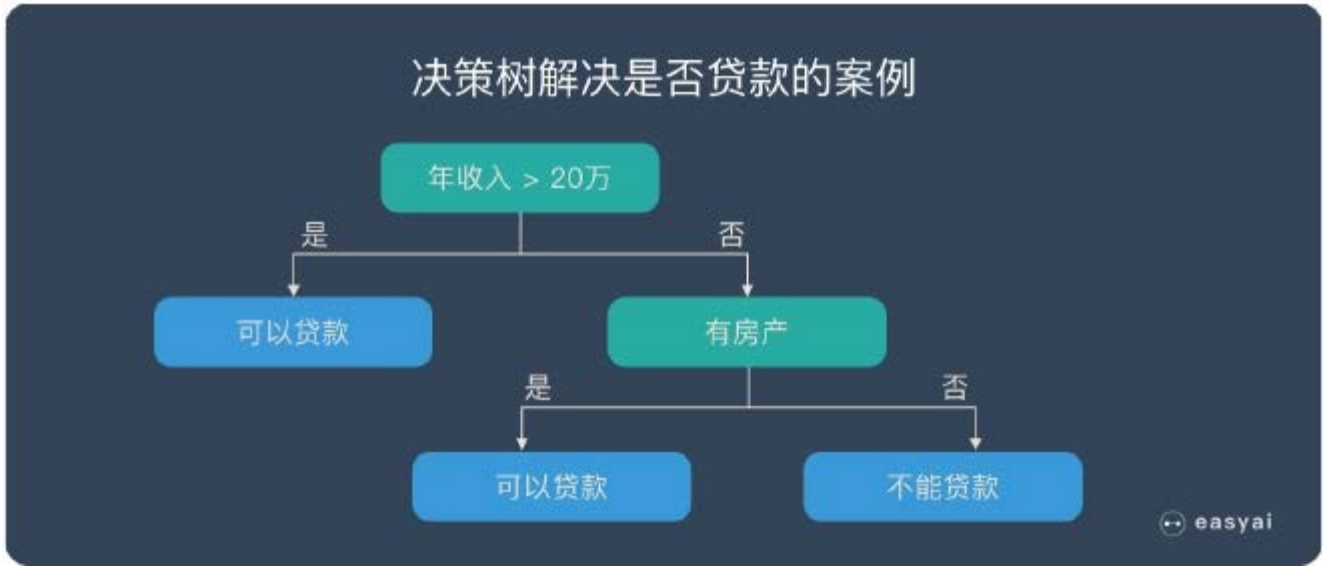
- 不适用于大型数据集
- 不适用于大量维度

KNN的最大问题是大的时间和空间复杂度，但是有两种数据结构，即Kd-tree和LSH，它们可以通过降低时间和空间复杂度来提高KNN的性能，具体实现及使用请参考sklearn库

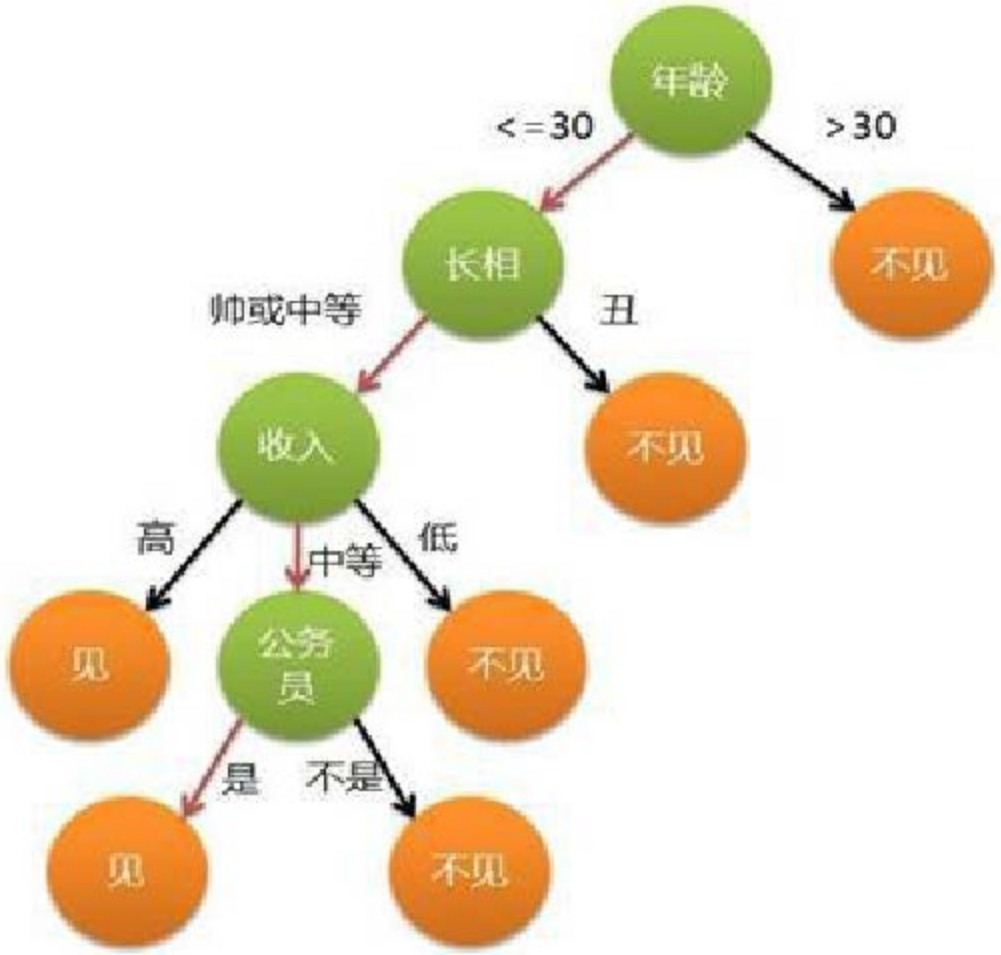
决策树-引例

银行贷款

银行要用机器学习算法来确定是否给客户发放贷款，为此需要考察客户的年收入，是否有房产这两个指标。

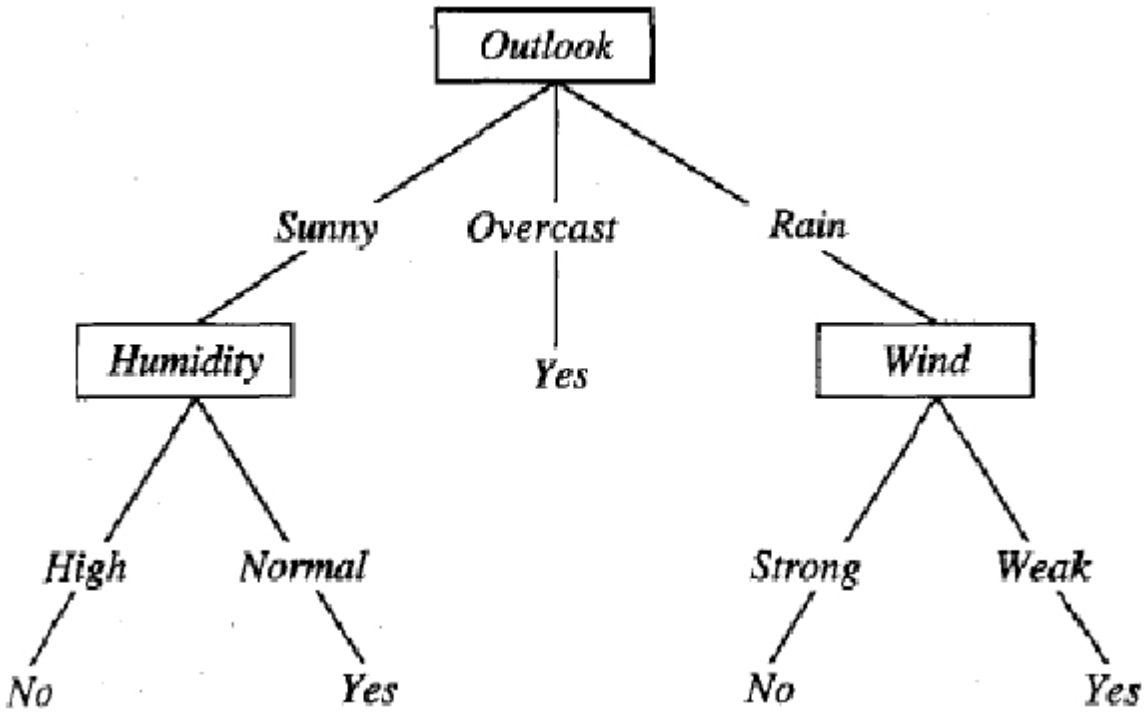


相亲

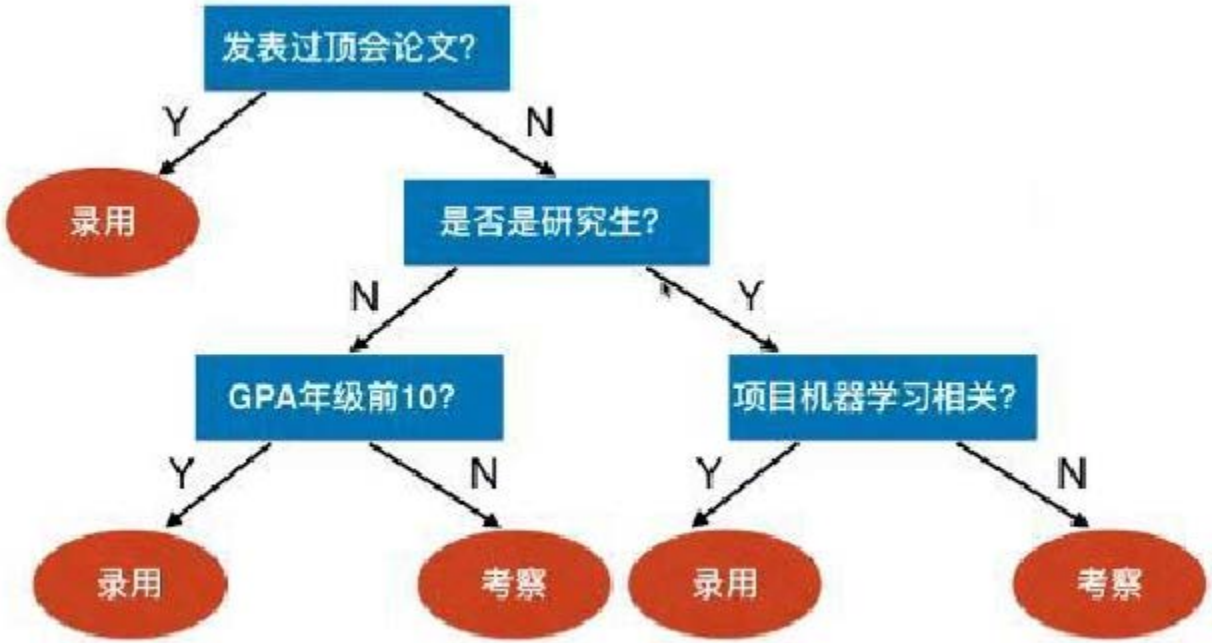


引例

天气是否适合打网球

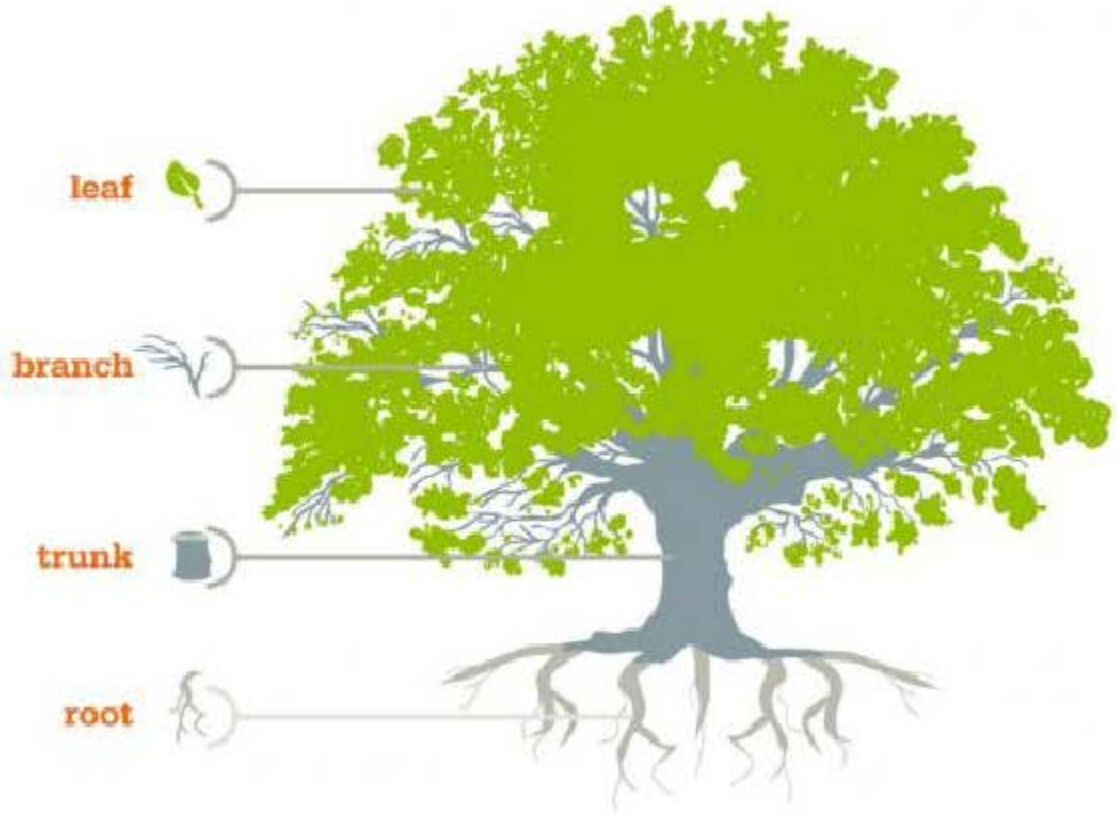


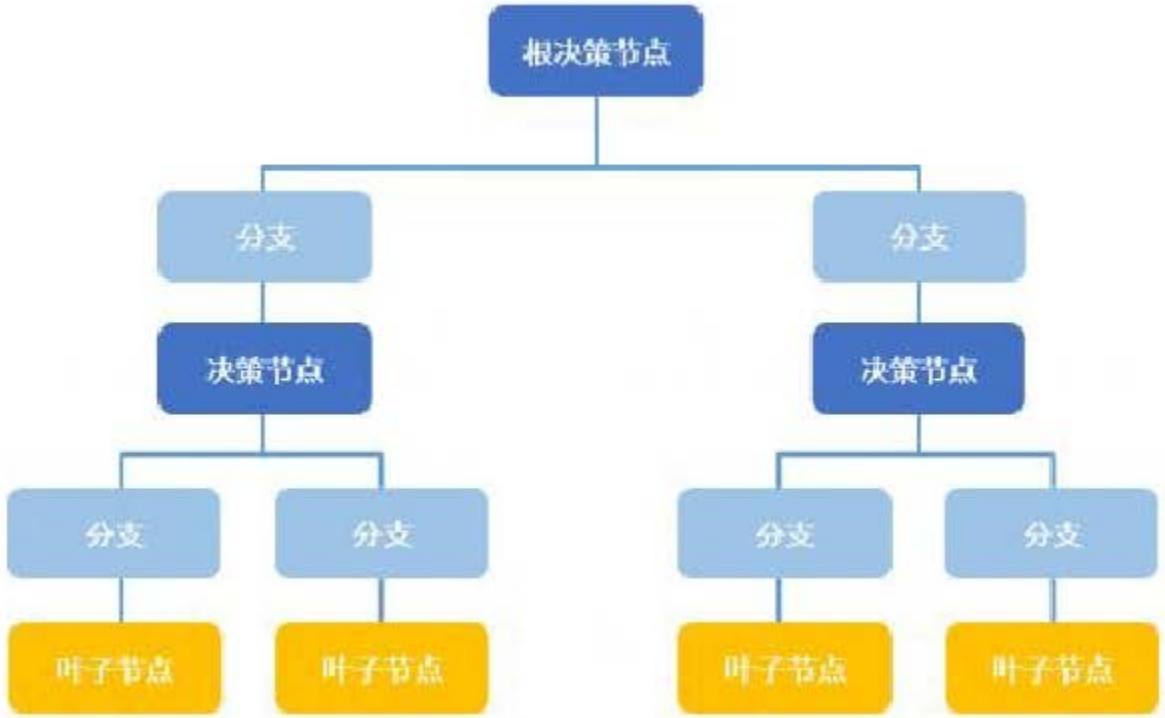
公司招聘



什么是决策树

决策树结构



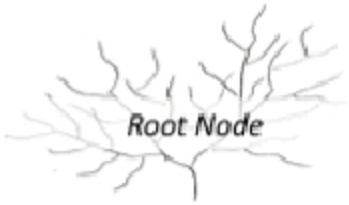


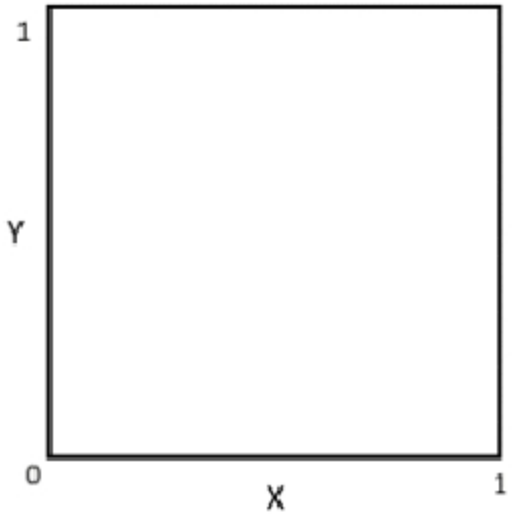
根节点：包含样本的全集；决策节点：对应特征属性测试；叶节点：代表决策的结果

什么是决策树

决策树模型呈树形结构，在分类问题中，表示基于特征对实例进行分类的过程。它可以认为是*if-then*规则的集合，使用层层推理来实现最终的分类。

决策树与条件概率分布





决策树将特征空间划分为互不相交的单元，在每个单元定义一个类的概率分布，这就构成了一个条件概率分布

决策树学习

决策树学习的三个步骤

特征选择：特征选择决定了使用哪些特征来做判断，在特征选择中通常使用的准则是：信息增益。

决策树生成：选择好特征后，就从根节点触发，对节点计算所有特征的信息增益，选择信息增益最大的特征作为节点特征，根据该特征的不同取值建立子节点。对每个子节点使用相同的方式生成新的子节点，直到信息增益很小或者没有特征可以选择为止。

决策树剪枝：剪枝的主要目的是对抗「过拟合」，通过主动去掉部分分支来降低过拟合的风险。

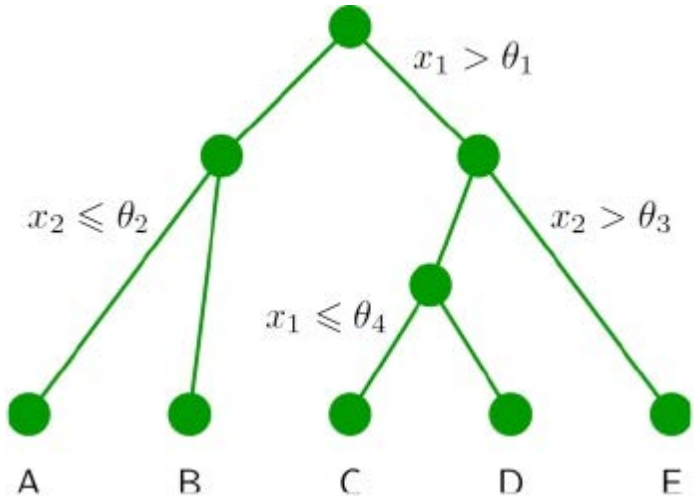
决策树学习

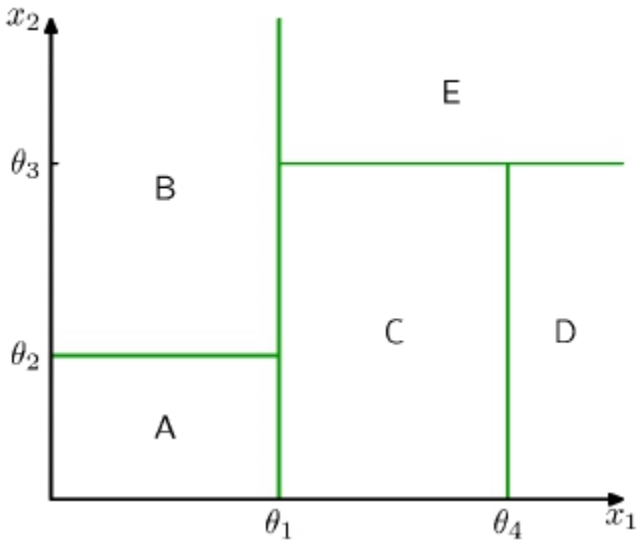
决策树学习算法思想

基本思想：采用自顶向下的递归方法，以信息熵为度量构造一棵熵值下降最快的树，到叶子结点处的熵值为零，此时每个叶结点中的实例都属于同一类。

决策树可以看成一个*if-then*的规则集合

一个决策树将特征空间划分为不相交的单元*(Cell)*或区域*(Region)*





决策树学习

决策树学习算法基本流程

Step1：选取一个属性作为决策树的根结点，然后就这个属性所有的取值创建树的分支

Step2：用这棵树来对训练数据集进行分类

- 如果一个叶结点的所有实例都属于同一类，则以该类为标记标识此叶结点。
- 如果所有的叶结点都有类标记，则算法终止

Step3：否则，选取一个从该结点到根路径中没有出现过的属性作为标记标识该结点，然后就这个属性的所有取值继续创建树的分支；重复算法步骤*2*

决策树学习

决策树学习算法分类

ID3算法

ID3算法的核心是在决策树各个结点上应用信息增益准则选择特征，递归地构建决策树

C4.5算法

C4.5算法与ID3算法相似，C4.5算法对ID3算法进行了改进。C4.5在生成的过程中，用信息增益比来选择特征

CART算法

这种算法即可以用于分类，也可以用于回归问题。CART 算法使用了基尼系数取代了信息熵模型。

ID3算法

*ID3(Iterative Dichotomiser 3)*迭代二分器算法，这个算法是建立在奥卡姆剃刀（用较少的东西，同样可以做好的事情）的基础上：越是小型的决策树越优于大的决策树。尽管如此，该算法也不是总是生成最小的树形结构，而是一个启发式算法

- 由*J. R. Quinlan*于*1979*年提出
- 一种最经典的决策树学习算法

基本思想：以信息熵为度量，用于决策树结点的属性选择，每次优先选取信息增益最大的属性，即能使熵值最小的属性，构造一棵熵值下降最快的决策树。到叶子结点的熵值为*0*，此时对应实例集中的实例属于同一类别

ID3算法

信息论与概率统计中，令离散随机变量X概率分布为$P(X = x_i) = p_i$，则随机变量X的熵定义为

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

假设当前样本集合D中第c $(c=1, 2, \ldots, C$，在二分类问题中$C = 2$）类样本所占比例为P_c，则D的信息熵定义为:

数据集D的信息熵

$$H(D) = -\sum_{c=1}^{c} p_c \log_2 p_c$$

(1/8,7/8) H(D)=0.375+0.16=0.5

$$= -\sum_{c=1}^{c} \frac{D_c}{D} \log_2 \frac{D_c}{D}$$

(4/8,4/8) H(D)=1

$H(D)$的值越小，则D的纯度越高

ID3算法

假设当前样本集合D共有C 类，每一类有D 个样本，属性 $a(a \in A$，A为属性集)不同的取值$\{a_1, a_2, \ldots, a_N\}$，$D$中属性为$a_i$的样本数为$D^n$，每一类中属性为$a_i$ 的样本数为D_c^n，则

D^n的信息熵定义为：

$$H(D^n) = -\sum_{c=1}^{C} \frac{D_c^n}{D^n} \log_2 \frac{D_c^n}{D^n}$$

属性 a 对集合D的信息增益（Information Gain）

$$G(D, a) = H(D) - \sum_{n=1}^{N} \frac{|D^n|}{|D|} H(D^n)$$

ID3算法即是以此信息增益为准则，对每次递归的结点属性进行选择的

ID3算法-示例(1)

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

ID3算法-示例(1)

计算信息熵-以属性色泽为例

$$H(D) = -\sum_{c=1}^{C} p_c \log_2 p_c = -\left(\frac{8}{17}\log_2\frac{8}{17} + \frac{9}{17}\log_2\frac{9}{17}\right) = 0.998$$

H(D青绿) = $-\left(\frac{3}{6}\log_2\frac{3}{6} + \frac{3}{6}\log_2\frac{3}{6}\right) = 1.000$

H(D乌黑) = $-\left(\frac{4}{6}\log_2\frac{4}{6} + \frac{2}{6}\log_2\frac{2}{6}\right) = 0.918$

H(D青绿) = $-\left(\frac{1}{5}\log_2\frac{1}{5} + \frac{4}{5}\log_2\frac{4}{5}\right) = 0.722$

ID3算法-示例(1)

计算信息增益-以属性色泽为例

$$G(D, 色泽) = H(D) - \sum_{n=1}^{3} \frac{|D^n|}{|D|} H(D^n)$$

$$= 0.998 - \left(\frac{6}{17} \times 1.000 + \frac{6}{17} \times 0.918 + \frac{5}{17} \times 0.722 \right)$$

$$= 0.109$$

同理，可计算出西瓜其它属性的信息增益

ID3算法-示例(1)

西瓜所有属性的信息增益

G（D，色泽）= 0.109　　G（D，根蒂）= 0.143

G（D，敲声）= 0.141　　G（D，纹理）= 0.381

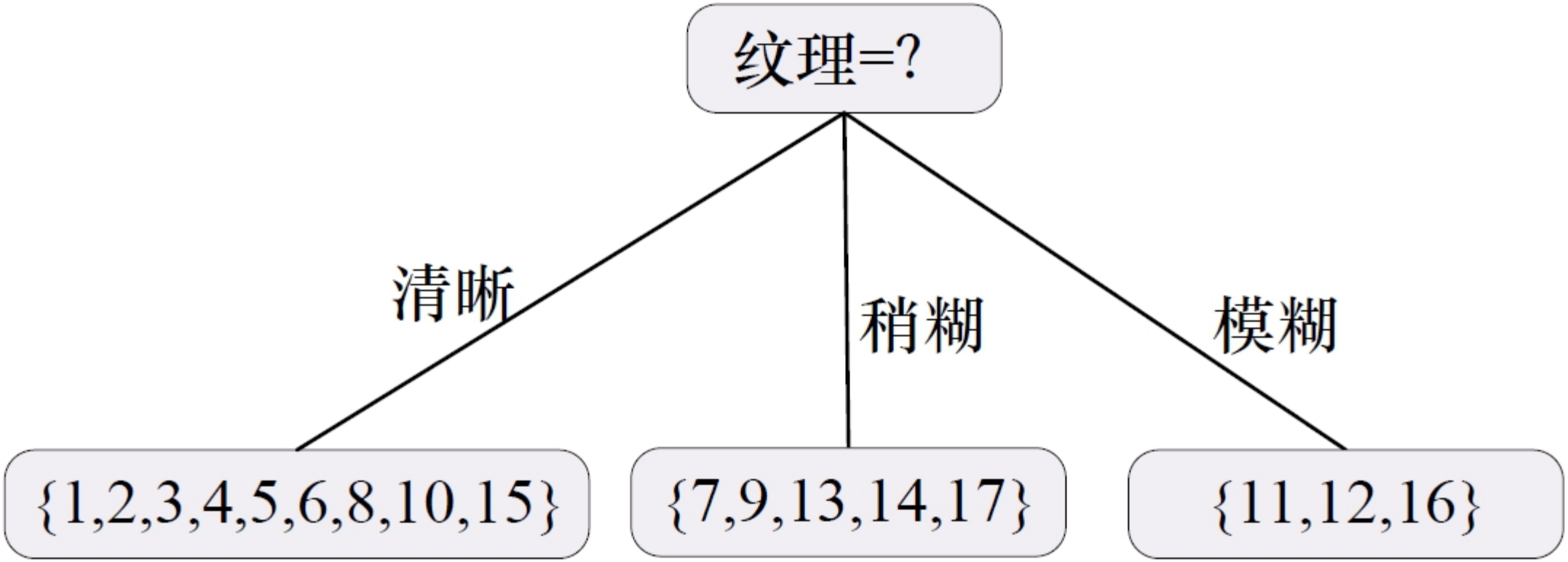
G（D，脐部）= 0.289　　G（D，触感）= 0.006

信息增益大的特征属性具有更强的分类能力。一般而言，信息增益越大，则意味着使用属性 a_i 来进行划分所获得的“纯度提升”越大

故而本次选择以纹理进行划分

ID3算法-示例(1)

基于属性“纹理”对根节点进行划分



ID3算法-示例(1)

继续进行划分-以“纹理=清晰”分支为例

“纹理=清晰”分支：

样本：$\{1,2,3,4,5,6,8,10,15\}$

计算信息增益

$$G\left(D_{\text{清晰}},\text{色泽}\right)=0.043$$

$$G\left(D_{\text{清晰}},\text{敲声}\right)=0.331$$

$$G\left(D_{\text{清晰}},\text{脐部}\right)=0.458$$

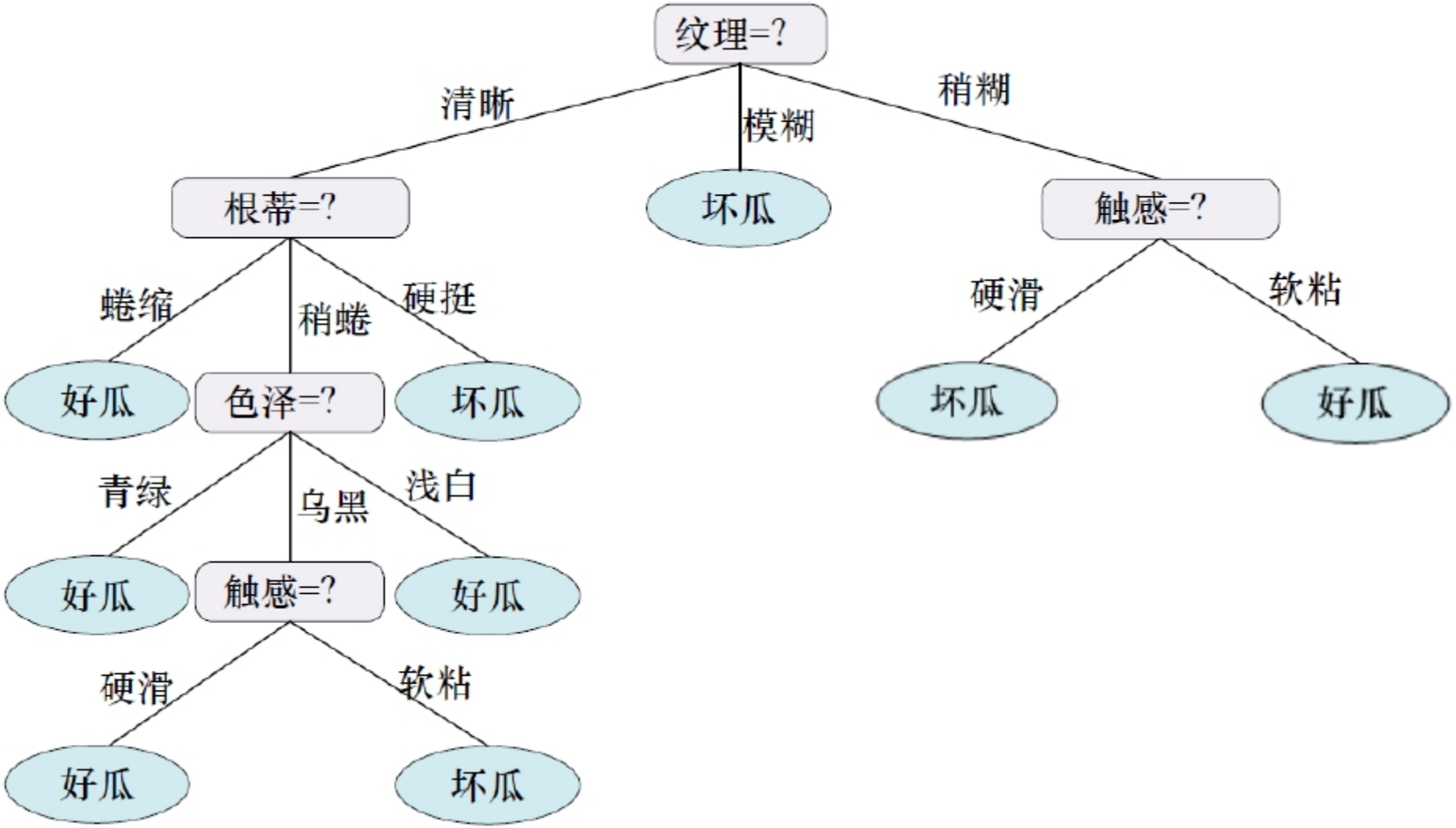
$$G\left(D_{\text{清晰}},\text{根蒂}\right)=0.458$$

$$G\left(D_{\text{清晰}},\text{触感}\right)=0.458$$

可选择根蒂继续进行划分

ID3算法-示例(1)

最终生成的决策树



ID3算法-示例(2)

谁在买计算机?

问题:

假定公司收集了右表数据，那么对于任意给定的客人（测试样例），预测这位客人是属于“买”计算机的一类，还是属于“不买”计算机的一类?

计数	年龄	收入	学生	信誉	买计算机?
64	青	高	否	良	不买
64	青	高	否	优	不买
128	中	高	否	良	买
60	老	中	否	良	买
64	老	低	是	良	买
64	老	低	是	优	不买
64	中	低	是	优	买
128	青	中	否	良	不买
64	青	低	是	良	买
132	老	中	是	良	买
64	青	中	是	优	买
32	中	中	否	优	买
32	中	高	是	良	买
63	老	中	否	优	不买
1	老	中	否	优	买

ID3算法-示例(2)

计数	年龄	收入	学生	信誉	买计算机?
64	青	高	否	良	不买
64	青	高	否	优	不买
128	中	高	否	良	买
60	老	中	否	良	买
64	老	低	是	良	买
64	老	低	是	优	不买
64	中	低	是	优	买
128	青	中	否	良	不买
64	青	低	是	良	买
132	老	中	是	良	买
64	青	中	是	优	买
32	中	中	否	优	买
32	中	高	是	良	买
63	老	中	否	优	不买
1	老	中	否	优	买

第1步：计算数据集的信息熵

决策属性“买计算机？”

该属性分两类：买/不买

$|C_1|$（买） = 641

$|C_2|$（不买） = 383

$|D| = |C_1| + |C_2| = 1024$

$$H(D) = -\sum_{c=1}^{C} \frac{D_c}{D} \log_2 \frac{D_c}{D}$$

$$= -\left(\frac{641}{1024} log_2 \frac{641}{1024} + \frac{383}{1024} log_2 \frac{383}{1024}\right)$$

$$= 0.9537$$

ID3算法-示例(2)

计数	年龄	收入	学生	信誉	归类：买计算机?
64	青	高	否	良	不买
64	青	高	否	优	不买
128	中	高	否	良	买
60	老	中	否	良	买
64	老	低	是	良	买
64	老	低	是	优	不买
64	中	低	是	优	买
128	青	中	否	良	不买
64	青	低	是	良	买
132	老	中	是	良	买
64	青	中	是	优	买
32	中	中	否	优	买
32	中	高	是	良	买
63	老	中	否	优	不买
1	老	中	否	优	买

第2步：计算特征的信息熵-以年龄为例

年龄共分三个组：

青年、中年、老年

青年买与不买比例为128/256

$$\left| D^{青年} \right| = \left| D_1^{青年} \right| + \left| D_2^{青年} \right| = 384$$

$$H(D^{青年}) = -\left(\frac{128}{384}\log_2\frac{128}{384} + \frac{256}{384}\log_2\frac{256}{384}\right)$$

$$= 0.9183$$

ID3算法-示例(2)

计数	年龄	收入	学生	信誉	归类：买计算机?
64	青	高	否	良	不买
64	青	高	否	优	不买
128	中	高	否	良	买
60	老	中	否	良	买
64	老	低	是	良	买
64	老	低	是	优	不买
64	中	低	是	优	买
128	青	中	否	良	不买
64	青	低	是	良	买
132	老	中	是	良	买
64	青	中	是	优	买
32	中	中	否	优	买
32	中	高	是	良	买
63	老	中	否	优	不买
1	老	中	否	优	买

第2步：计算特征的信息熵-以年龄为例

中年买与不买比例为256/0

$$\left|D_{\text{中年}}\right| = \left|D_1^{\text{中年}}\right| + \left|D_2^{\text{中年}}\right| = 256$$

$$H(D^{\text{中年}}) = -\left(\frac{256}{256}\log_2\frac{256}{256} + \frac{0}{256}\log_2\frac{0}{256}\right)$$

$$= 0$$

ID3算法-示例(2)

计数	年龄	收入	学生	信誉	归类：买计算机?
64	青	高	否	良	不买
64	青	高	否	优	不买
128	中	高	否	良	买
60	老	中	否	良	买
64	老	低	是	良	买
64	老	低	是	优	不买
64	中	低	是	优	买
128	青	中	否	良	不买
64	青	低	是	良	买
132	老	中	是	良	买
64	青	中	是	优	买
32	中	中	否	优	买
32	中	高	是	良	买
63	老	中	否	优	不买
1	老	中	否	优	买

第2步：计算特征的信息熵-以年龄为例

老年买与不买比例为257/127

$$\left|D_{老年}\right| = \left|D_1^{老年}\right| + \left|D_2^{老年}\right| = 384$$

$$H(D^{老年}) = -\left(\frac{257}{384}\log_2\frac{257}{384} + \frac{127}{384}\log_2\frac{127}{384}\right)$$

$$= 0.9157$$

ID3算法-示例(2)

计数	年龄	收入	学生	信誉	归类：买计算机?
64	青	高	否	良	不买
64	青	高	否	优	不买
128	中	高	否	良	买
60	老	中	否	良	买
64	老	低	是	良	买
64	老	低	是	优	不买
64	中	低	是	优	买
128	青	中	否	良	不买
64	青	低	是	良	买
132	老	中	是	良	买
64	青	中	是	优	买
32	中	中	否	优	买
32	中	高	是	良	买
63	老	中	否	优	不买
1	老	中	否	优	买

第3步：计算年龄的信息增益

中年买与不买比例为257/127

$$|D_{青年}| + |D_{中年}| + |D_{老年}| = 1024$$

$$G(D, 年龄) = H(D) - \sum_{n=1}^{3} \frac{|D^n|}{|D|} H(D^n)$$

$$= 0.9537 - \left(\frac{384}{1024} \times 0.9183 + \frac{256}{1024} \times 0 + \frac{384}{1024} \times 0.9157 \right)$$

$$= 0.2660$$

ID3算法-示例(2)

计数	年龄	收入	学生	信誉	归类：买计算机?
64	青	高	否	良	不买
64	青	高	否	优	不买
128	中	高	否	良	买
60	老	中	否	良	买
64	老	低	是	良	买
64	老	低	是	优	不买
64	中	低	是	优	买
128	青	中	否	良	不买
64	青	低	是	良	买
132	老	中	是	良	买
64	青	中	是	优	买
32	中	中	否	优	买
32	中	高	是	良	买
63	老	中	否	优	不买
1	老	中	否	优	买

第4步：计算所有特征的信息增益，并选择结点：年龄

$$G(D,\text{年龄}) = 0.2660$$

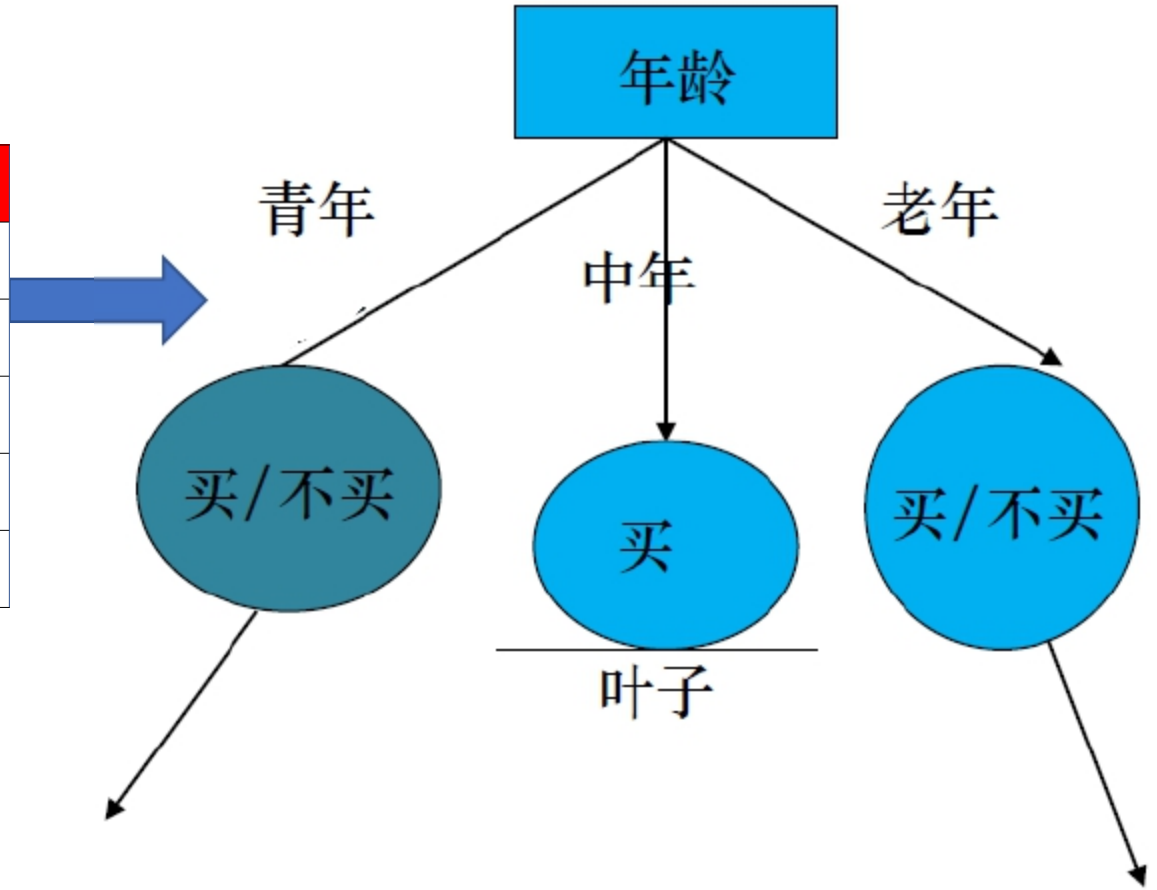
$$G(D,\text{收入}) = 0.0176$$

$$G(D,\text{学生}) = 0.1726$$

$$G(D,\text{信誉}) = 0.0453$$

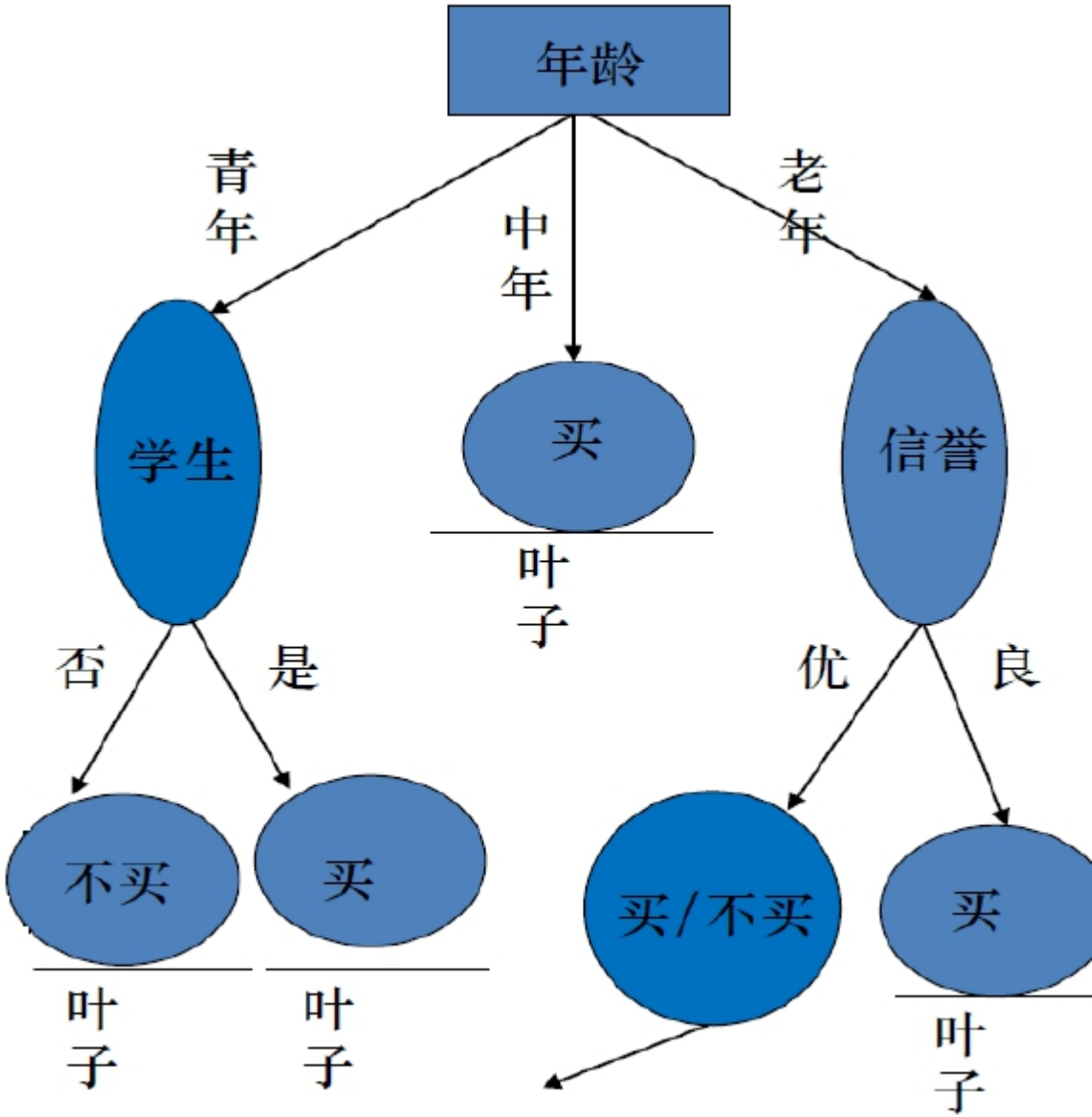
ID3算法-示例(2)

计数	年龄	收入	学生	信誉	归类：买计算机?
64	青	高	否	良	不买
64	青	高	否	优	不买
128	青	中	否	良	不买
64	青	低	是	良	买
64	青	中	是	优	买



ID3算法-示例(2)

计数	年龄	收入	学生	信誉	归类：买计算机?
64	青	高	否	良	不买
64	青	高	否	优	不买
128	中	高	否	良	买
60	老	中	否	良	买
64	老	低	是	良	买
64	老	低	是	优	不买
64	中	低	是	优	买
128	青	中	否	良	不买
64	青	低	是	良	买
132	老	中	是	良	买
64	青	中	是	优	买
32	中	中	否	优	买
32	中	高	是	良	买
63	老	中	否	优	不买
1	老	中	否	优	买



如此继续下去，直到最终划分

ID3算法

算法优点

- 只需对训练实例进行较好地标注，就能进行学习，从一类无序、无规则事物（概念）中推理出分类规则
- 分类模型是树状结构，简单直观，可将决策树中到达每个叶结点的路径转换为*IF—THEN*形式的分类规则，比较符合人类的理解方式

算法局限性

- 信息增益偏好取值多的属性（例如示例*1*中我们刻意没去考虑西瓜的编号属性，若把编号也作为一个候选划分属性，可得它的信息增益为*0.998*，远大于其他候选划分属性。.这很容易理解'编号'将产生*17*个分支，每个分支结点仅包含一个样本，这些分支结点的纯度已达最大.然而，这样的决策树显然不具有泛化能力，无法对新样本进行有效预测）
- 无法处理连续值的属性
- 无法处理属性值不完整的训练数据

属性筛选度量标准

信息增益的问题

$$G(D,a) = H(D) - \sum_{n=1}^{N} \frac{|D^n|}{|D|} H(D^n)$$

信息增益准则对可取值数目$\mathcal{N}$较多的属性有所偏好：

取值更多的属性容易使得数据更“纯”，其信息增益更大。决策树会首先挑选这个属性作为树的划分点；结果训练出来的形状是一棵庞大且深度很浅的树，这样的划分极不合理

属性筛选度量标准

信息增益率(Information Gain Ratio)

$$G_{ratio}(D,a)=\frac{G(D,a)}{H(a)}$$

其中

$$H(a)=-\sum_{n=1}^{N}\frac{|D^n|}{|D|}\log_2\frac{|D^n|}{|D|}$$

$\mathcal{N}$代表属性 a 取值的数目，该公式称为属性 a 的固有值

$\mathcal{N}$越大， $H(a)$通常也越大；因此采用信息增益率，可缓解信息增益准则对可取值数目较多的属性的偏好

*C4.5*算法就采用增益率替代了*ID3*算法的信息增益

属性筛选度量标准

基尼指数(Gini Index)

$$G_{ini}(D) = 1 - \sum_{c=1}^{C} \left(\frac{|D_c|}{|D|} \right)^2$$

数据集的纯度越高，基尼指数越小

属性 a 的基尼指数 $$G_{ini}(D,a) = \sum_{n=1}^{N} \frac{|D^n|}{|D|} Gini(D^n)$$

最优属性选择 $$a^* = argmin_{a \in A} Gini(D,a)$$

*CART*算法就采用基尼指数替代了*ID3*算法的信息增益

过拟合与欠拟合

过拟合

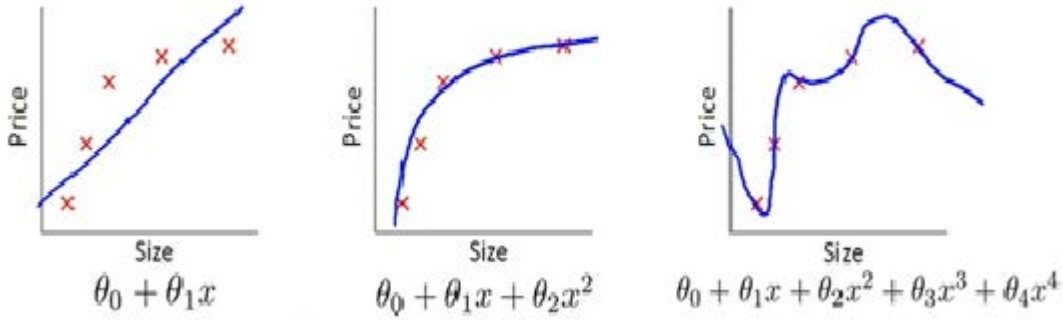
我们总是希望在机器学习训练时，机器学习模型能在新样本上很好的表现。过拟合时，通常是因为模型过于复杂，学习器把训练样本学得“太好了”，很可能把一些训练样本自身的特性当成了所有潜在样本的共性了，这样一来模型的泛化性能就下降了

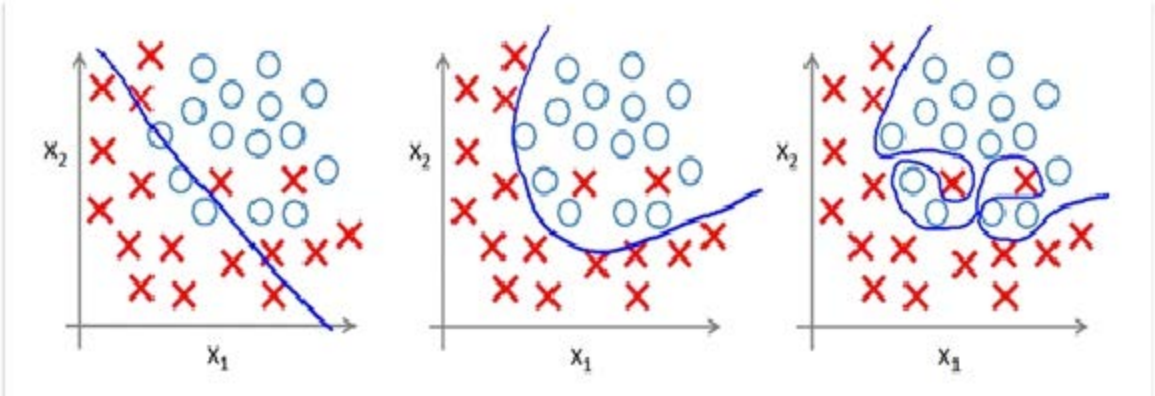
欠拟合

欠拟合时，模型又过于简单，学习器没有很好地学到训练样本的一般性质，所以不论在训练数据还是测试数据中表现都很差

过拟合与欠拟合

过拟合与欠拟合直观比较





欠拟合　　合适　　过拟合

过拟合与欠拟合

欠拟合可能原因以及解决措施

欠拟合则通常是由于拟合函数的能力不够、没有充分的利用数据而造成的，欠拟合比较容易克服，为此可以增加迭代次数继续训练、尝试换用其他算法、增加模型的参数数量和复杂程度，例如在决策树学习中扩展分支、在神经网络学习中增加训练轮数等

过拟合可能原因以及缓解措施

过拟合成因是给定的数据集相对过于简单，使得模型在拟合函数时过分地考虑了噪声等不必要的数据间的关联，或者说相对于给定数据集，模型过于复杂、拟合能力过强。过拟合是机器学习面临的关键障碍，各类学习算法都必然带有一些针对过拟合的措施，然而必须认识到，过拟合是无法彻底避免的，我们所能做的只是“缓解”或者说减小其风险。

P问题与NP问题

P问题

可以在多项式时间内解决的问题（如冒泡排序）

NP问题

无法在多项式时间内解决，能够在多项式时间内得到确认的一类问题（如*AlphaGo*下围棋，验证一局棋的结果显然是很简单的，但要保证每局都能赢的话目前的方法需要电脑穷举出所有的可能性，并根据每一步棋的变化搜索出最终达到胜利的下棋路径，目前的计算机性能显然是达不到的，围棋的变化多过宇宙的原子数量。*AlphaGo*所做的只不过是通过优化算法提高穷举效率，同时利用现有的大数据与云计算来提升计算性能而已）$\mathcal{P}$类问题是$\mathcal{NP}$问题的子集，因为存在多项式时间解法的问题，总能在多项式时间内验证。因为机器学习面临的问题通常是$\mathcal{NP}$难甚至更难，而有效的学习算法必然是在多项式时间内运行完成，若可彻底避免过拟合， 则通过经验误差最小化就能获最优解，这就意味着我们构造性地证明了“$\mathcal{P}=\mathcal{NP}$”;因此只要相信“$\mathcal{P}\neq\mathcal{NP}$”，过拟合就不可避免

剪枝处理(Pruning)

问题：过拟合

决策树对训练数据有很好的分类能力，但对未知的测试数据未必有好的分类能力，泛化能力弱，即可能发生过拟合现象

可能原因

- 训练数据有噪声，同时拟合了数据和噪音，影响分类效果
- 训练集样本太少，易出现耦合的规律性，使一些属性恰巧可很好地分类，但却与实际目标函数无关

剪枝处理(Pruning)

针对过拟合问题：剪枝是主要手段

基本策略

- 预剪枝策略*(Pre-pruning)*：决策树生成过程中，对每个结点在划分前进行估计，若划分不能带来决策树泛化性能提升，则停止划分并将该节点设为叶结点

- 后剪枝策略*(Post-pruning)*：先利用训练集生成决策树，自底向上对非叶结点进行考察，若将该结点对应子树替换该结点能带来泛化性能提升，则将该子树替换为叶结点

剪枝处理(Pruning)

训练样本

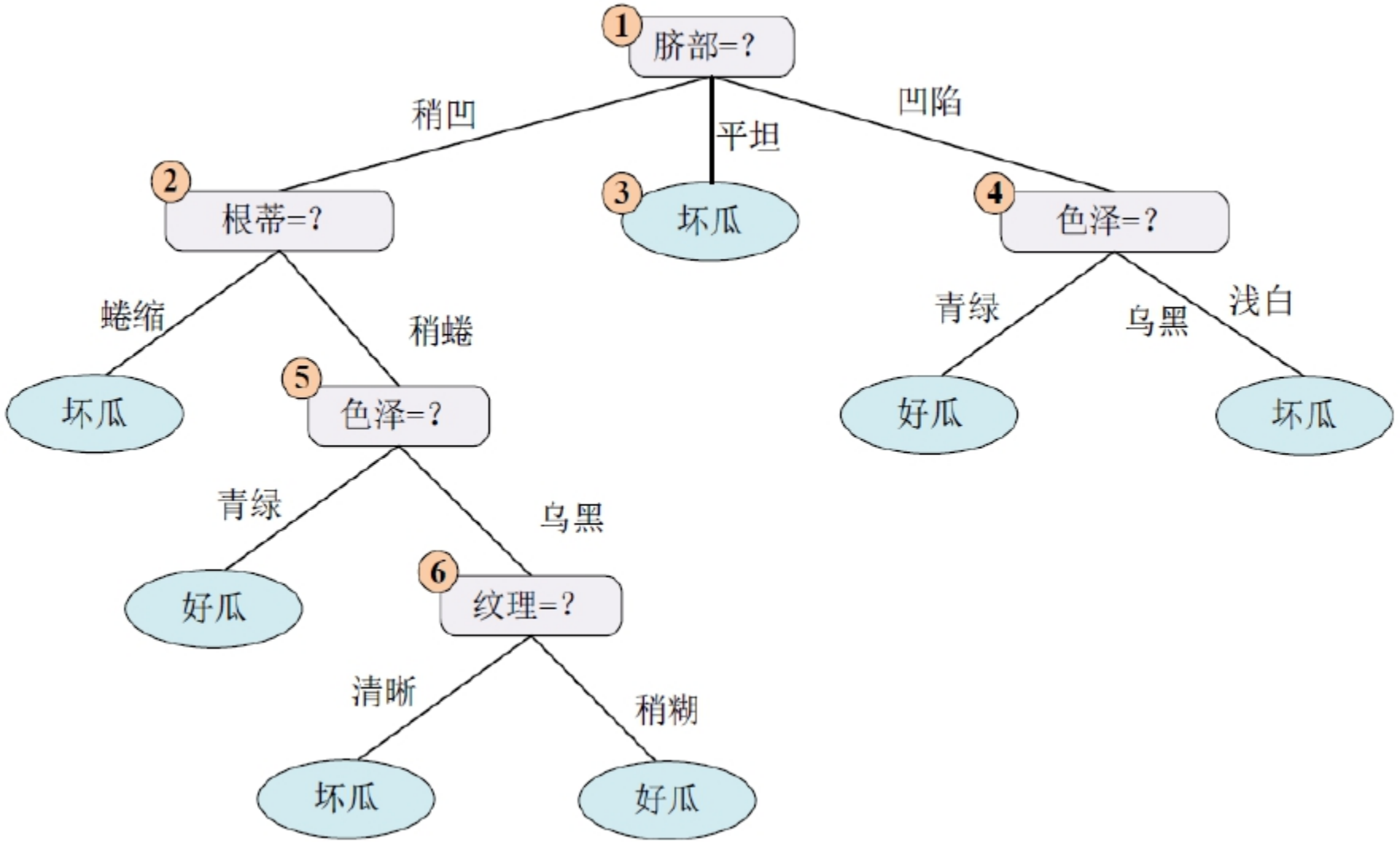
编号	色泽	根蒂	敲声	纹理	脐部	触感	好瓜
1	青绿	蜷缩	浊响	清晰	凹陷	硬滑	是
2	乌黑	蜷缩	沉闷	清晰	凹陷	硬滑	是
3	乌黑	蜷缩	浊响	清晰	凹陷	硬滑	是
6	青绿	稍蜷	浊响	清晰	稍凹	软粘	是
7	乌黑	稍蜷	浊响	稍糊	稍凹	软粘	是
10	青绿	硬挺	清脆	清晰	平坦	软粘	否
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	否
15	乌黑	稍蜷	浊响	清晰	稍凹	软粘	否
16	浅白	蜷缩	浊响	模糊	平坦	硬滑	否
17	青绿	蜷缩	沉闷	稍糊	稍凹	硬滑	否

测试样本

编号	色泽	根蒂	敲声	纹理	脐部	触感	好瓜
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
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
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

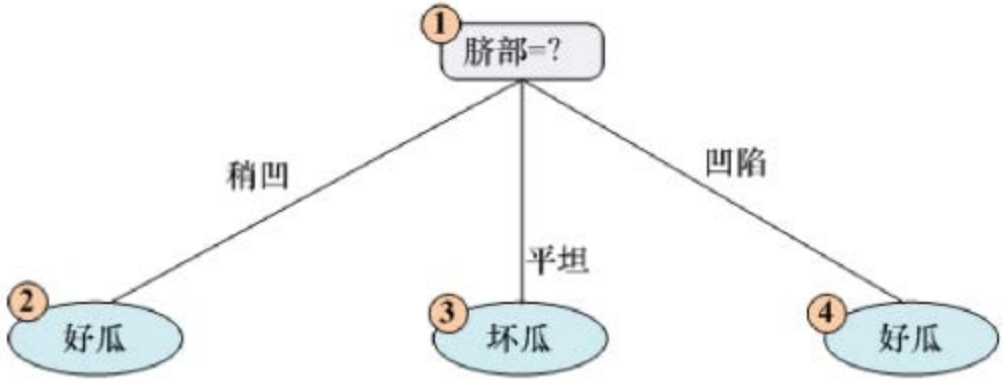
剪枝处理(Pruning)

ID3算法生成的决策树（训练集）



预剪枝算法

第一步：评估结点1



✓基于信息增益准则，选择属性“脐部”

不划分：

- 在划分之前，所有样例集中在根结点 *1* 。若不进行划分，该结点将被标记为叶结点，其类别标记为训练样例数最多的类别，“好瓜”（正负例一样多，任选其一）
- 泛化性能：{4，5，8} 被正确分类　　$3/7 = 42.9\%$

划分：

- 结点*4*：凹陷{1，2，3，14}　　“好瓜”
- 结点*2*：稍凹{6，7，15，17}　　“好瓜”
- 结点*3*：平坦{10，16}　　“坏瓜”
- 泛化性能：{4，5，8，11，12} 被正确分类　　$5/7 = 71.4\%$

预剪枝决策：划分

预剪枝算法

第二步：评估结点2　　训练样本{6，7，15，17}

✓基于信息增益准则，选择属性“根蒂”

不划分：　{4，5，8，11，12} 被正确分类　　$5/7 = 71.4\%$

划分：　{4，5，8，11，12} 被正确分类　　$5/7 = 71.4\%$

预剪枝决策：不划分

第三步：评估结点4　　训练样本{1，2，3，14}

✓基于信息增益准则，选择属性“色泽”

不划分：　{4，5，8，11，12} 被正确分类　　$5/7 = 71.4\%$

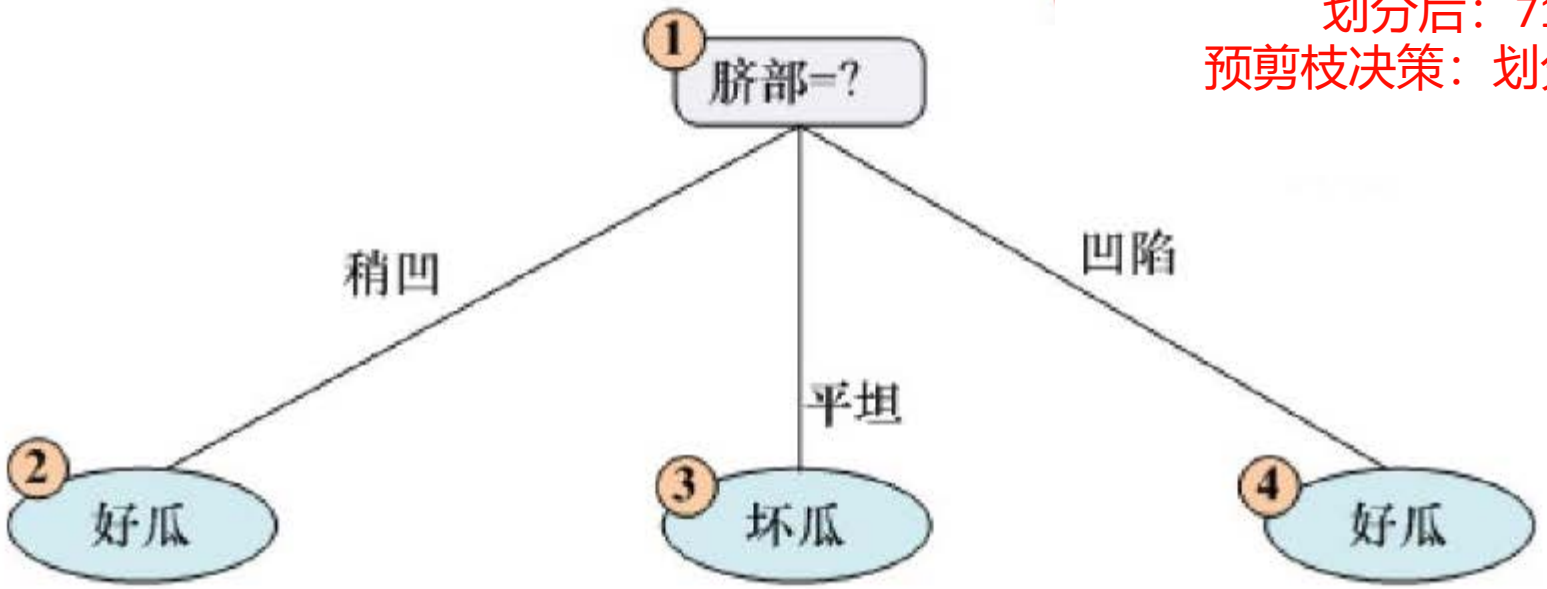
划分：　{4，~~5~~，8，11，12} 被正确分类　　$4/7 = 57.1\%$

预剪枝决策：不划分

预剪枝算法

最终生成的决策树

测试集精度
“脐部=？”。划分前：42.9%
划分后：71.4%
预剪枝决策：划分



测试集精度
“根蒂=？”。划分前：71.4%
划分后：71.4%
预剪枝决策：不划分

测试集精度
“色泽=？”。划分前：71.4%
划分后：57.1%
预剪枝决策：不划分

预剪枝算法

策略特点

优势：“剪掉”很多没必要展开的分支，降低了过拟合风险，并且显著减少了决策树的训练时间开销和测试时间开销

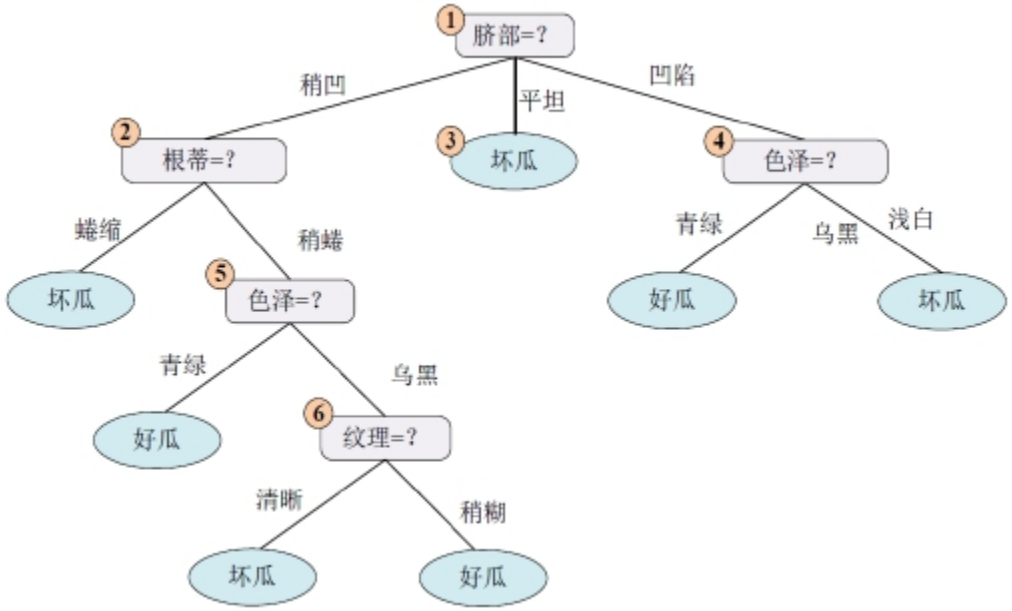
劣势：有些分支的当前划分有可能不能提高甚至降低泛化性能，但后续划分有可能提高泛化性能；预剪枝禁止这些后续分支的展开，可能会导致欠拟合

后剪枝算法

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
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
10	青绿	硬挺	清脆	清晰	平坦	软粘	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

编号	色泽	根蒂	敲声	纹理	脐部	触感	好瓜
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
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
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

后剪枝先从训练集生成一棵完整决策树



后剪枝算法

第一步：评估结点6

剪枝前：

- 剪枝前，属性为“纹理”，样本为{7，15}
- 泛化性能：{4，11，12} 被正确分类　　$3/7 = 42.9\%$

剪枝后：

- 把结点6替换为叶结点，“好/坏瓜”
- 泛化性能：{4，8/9，11，12} 被正确分类　　$4/7 = 57.1\%$

后剪枝决策：剪枝

后剪枝算法

第二步：评估结点5

剪枝前：

- 剪枝前，属性为“色泽”，样本为{6，7，15}
- 泛化性能：根据第一步结果，$4/7 = 57.1\%$

剪枝后：

- 把结点 *5* 替换为叶结点，“好瓜”
- 泛化性能：{4，8，11，12} 被正确分类 $4/7 = 57.1\%$

后剪枝决策：不剪枝

后剪枝算法

第三步：评估结点4

剪枝前：

- 剪枝前，属性为“色泽”，样本为{1，2，3，14}
- 泛化性能：根据第二步结果，$4/7 = 57.1\%$

剪枝后：

- 把结点 *4* 替换为叶结点，“好瓜”
- 泛化性能：{4，5，8，11，12} 被正确分类 $5/7 = 71.4\%$

后剪枝决策：剪枝

后剪枝算法

第四步：评估结点2

剪枝前：

- 剪枝前，属性为“根蒂”，样本为{6，7，15，17}
- 泛化性能：根据第三步结果，$5/7 = 71.4\%$

剪枝后：

- 把结点 *2* 替换为叶结点，“好/坏瓜”
- 泛化性能：{4，5，8/9，11，12} 被正确分类 $5/7 = 71.4\%$

后剪枝决策：不剪枝

后剪枝算法

第五步：评估结点1

剪枝前：

- 剪枝前，属性为“脐部”，样本为全部训练样本
- 泛化性能：根据第四步结果，$5/7 = 71.4\%$

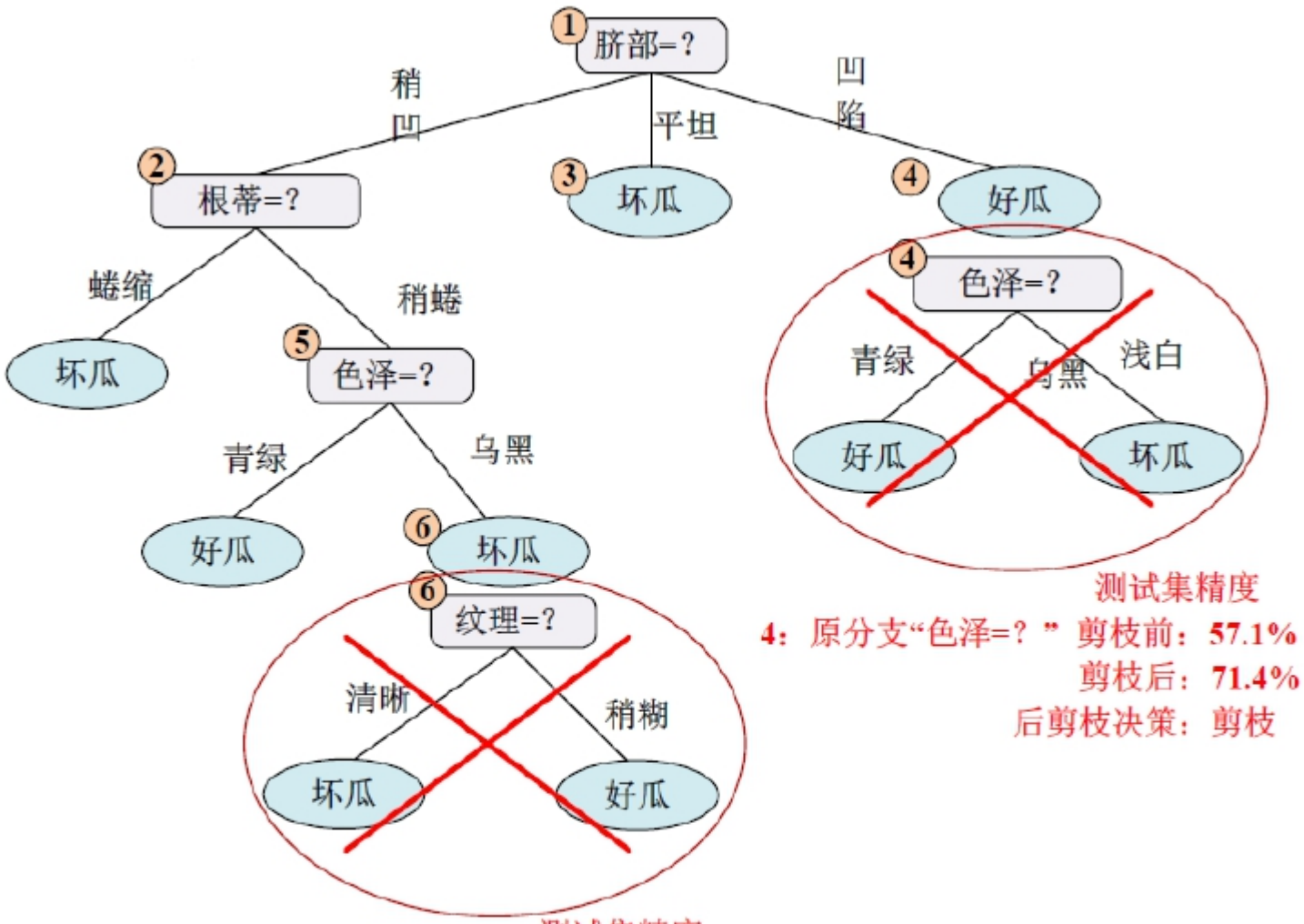
剪枝后：

- 把结点 *1* 替换为叶结点，正负例一样多，假设为 “坏瓜”
- 泛化性能：$\{9, 11, 12, 13\}$ 被正确分类 $4/7 = 57.1\%$

后剪枝决策：不剪枝

后剪枝算法

最终生成的决策树



测试集精度

4：原分支“色泽=？”剪枝前：57.1%

剪枝后：71.4%

后剪枝决策：剪枝

测试集精度

6：原分支“纹理=？”剪枝前：42.9%

剪枝后：57.1%

后剪枝决策：剪枝

后剪枝算法

策略特点

优势：测试了所有分支，比预剪枝决策树保留了更多分支，降低了欠拟合的风险，泛化性能一般优于预剪枝决策树

劣势：后剪枝过程在生成完全决策树后再进行，且要自底向上对所有非叶节点逐一评估；因此，决策树的训练时间开销要高于未剪枝决策树和预剪枝决策树