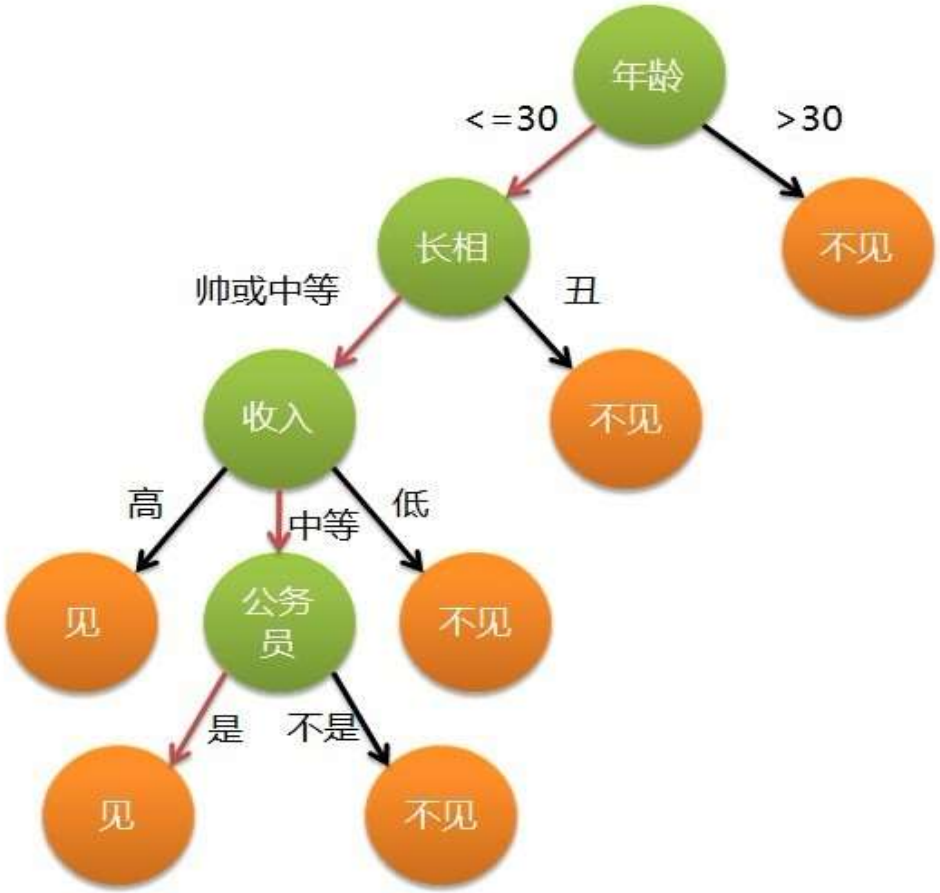
清華大學
Tsinghua University

第五章
决策树

例子

- 套用俗语，决策树分类的思想类似于找对象。现想象一个女孩的母亲要给这个女孩介绍男朋友，于是有了下面的对话：
- 女儿：多大年纪了？
  母亲：26。
  女儿：长的帅不帅？
  母亲：挺帅的。
  女儿：收入高不？
  母亲：不算很高，中等情况。
  女儿：是公务员不？
  母亲：是，在税务局上班呢。
  女儿：那好，我去见见。

例子



决策树

关于分类问题

名称	体温	表皮覆盖	胎生	水生动物	飞行动物	有腿	冬眠	类标号
人类	恒温	毛发	是	否	否	是	否	哺乳动物
海龟	冷血	鳞片	否	半	否	是	否	爬行类
鸽子	恒温	羽毛	否	否	是	是	否	鸟类
鲸	恒温	毛发	是	是	否	否	否	哺乳类

x

y

分类与回归

分类目标属性y是离散的，回归目标属性y是连续的

决策树-解决分类问题的一般方法

通过以上对分类问题一般方法的描述，可以看出分类问题一般包括两个步骤：

1、模型构建（归纳）
通过对训练集合的归纳，建立分类模型。

2、预测应用（推论）
根据建立的分类模型，对测试集合进行测试。

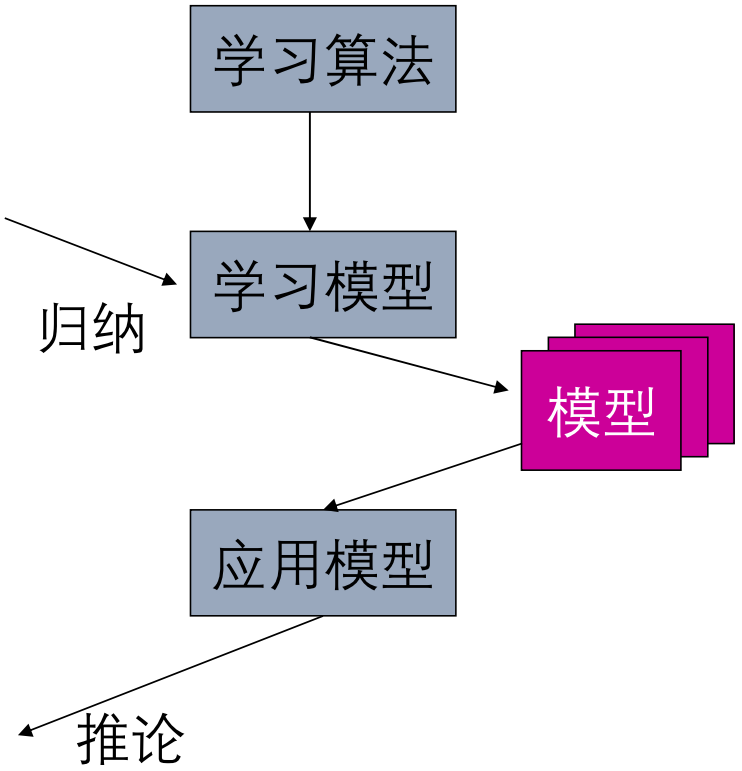
决策树-解决分类问题的一般方法

训练集（类标号已知）

TID	A1	A2	A3	类
1	Y	100	L	N
2	N	125	S	N
3	Y	400	L	Y
4	N	415	M	N

检验集（类标号未知）

TID	A1	A2	A3	类
1	Y	100	L	?
2	N	125	S	?
3	Y	400	L	?
4	N	415	M	?



决策树

- 决策树是一种典型的分类方法
  - 首先对数据进行处理，利用归纳算法生成可读的规则和决策树，
  - 然后使用决策对新数据进行分析。
- 本质上决策树是通过一系列规则对数据进行分类的过程。

决策树

- 决策树的优点
  - 1、推理过程容易理解，决策推理过程可以表示成If Then形式；
  - 2、推理过程完全依赖于属性变量的取值特点；
  - 3、可自动忽略目标变量没有贡献的属性变量，也为判断属性变量的重要性，减少变量的数目提供参考。

决策树和归纳算法

- 决策树技术发现数据模式和规则的核心是归纳算法。
- 归纳是从特殊到一般的过程。
- 归纳推理从若干个事实中表征出的特征、特性和属性中，通过比较、总结、概括而得出一个规律性的结论。
- 归纳推理试图从对象的一部分或整体的特定的观察中获得一个完备且正确的描述。即从特殊事实到普遍性规律的结论。
- 归纳对于认识的发展和完善具有重要的意义。人类知识的增长主要来源于归纳学习。

决策树和归纳算法

- 归纳学习由于依赖于检验数据，因此又称为检验学习。
- 归纳学习存在一个基本的假设：
  - 任一假设如果能够在足够大的训练样本集中很好的逼近目标函数，则它也能在未见样本中很好地逼近目标函数。该假定是归纳学习的有效性的前提条件。

决策树和归纳算法

- 归纳过程就是在描述空间中进行搜索的过程。
- 归纳可分为自顶向下，自底向上和双向搜索三种方式。
  - 自底向上法一次处理一个输入对象。将描述逐步一般化。直到最终的一般化描述。
  - 自顶向下法对可能的一般性描述集进行搜索，试图找到一些满足一定要求的最优的描述。

决策树-从机器学习看分类及归纳推理等问题

- 从特殊的训练样例中归纳出一般函数是机器学习的中心问题;
- 从训练样例中进行学习通常被视为归纳推理。
  - 每个例子都是一个对偶（序偶）（x, f(x)），对每个输入的x，都有确定的输出f(x)。
  - 学习过程将产生对目标函数f的不同逼近。F的每一个逼近都叫做一个假设。
  - 假设需要以某种形式表示。例如，y=ax+b。通过调整假设的表示，学习过程将产生出假设的不同变形。
  - 在表示中通常需要修改参数（如a, b）。

决策树-从机器学习看分类及归纳推理等问题

- 从这些不同的变形中选择最佳的假设（或者说权值集合）。
- 方法的定义：
  - 使训练值与假设值 预测出的值之间的误差平方和E最小为最佳。

$$E = \sum_{<b,Vtrain(b)>\in trainingexamples} (Vtrain(b) - \hat{V}(b))^2$$

决策树算法

- 与决策树相关的重要算法包括:
  - CLS, ID3，C4.5，CART
- 算法的发展过程
  - Hunt,Marin和Stone 于1966年研制的CLS学习系统，用于学习单个概　念。
  - 1979年, J.R. Quinlan 给出ID3算法，并在1983年和1986年对ID3 进行了总结和简化，使其成为决策树学习算法的典型。
  - Schlimmer 和Fisher 于1986年对ID3进行改造，在每个可能的决策树节点创建缓冲区，使决策树可以递增式生成，得到ID4算法。
  - 1988年，Utgoff 在ID4基础上提出了ID5学习算法，进一步提高了效率。
  - 1993年，Quinlan 进一步发展了ID3算法，改进成C4.5算法。
  - 另一类决策树算法为CART，与C4.5不同的是，CART的决策树由二元逻辑问题生成，每个树节点只有两个分枝，分别包括学习实例的正例与反例。

决策树算法

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），你能帮助公司将这位客人归类吗？

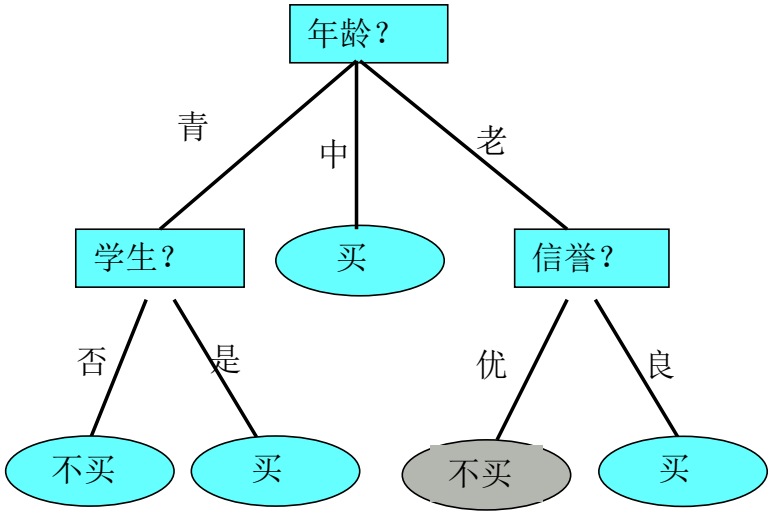
即：你能预测这位客人是属于“买”计算机的那一类，还是属于“不买”计算机的那一类？

又：你需要多少有关这位客人的信息才能回答这个问题？

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

决策树算法

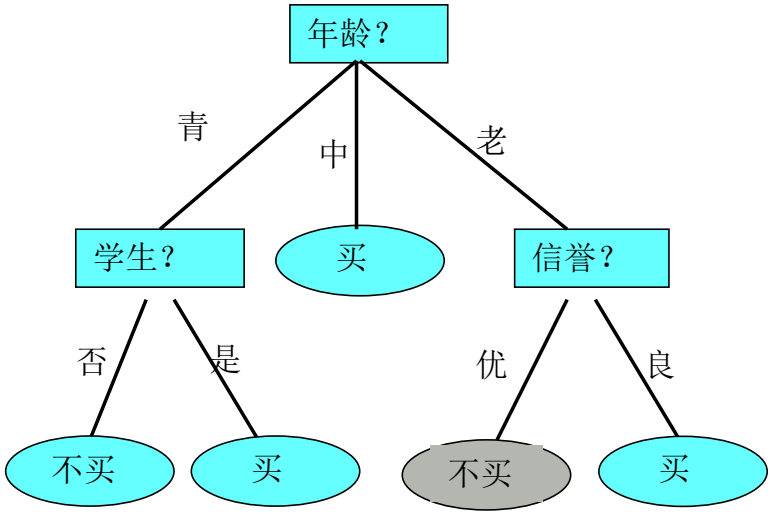
谁在买计算机？



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

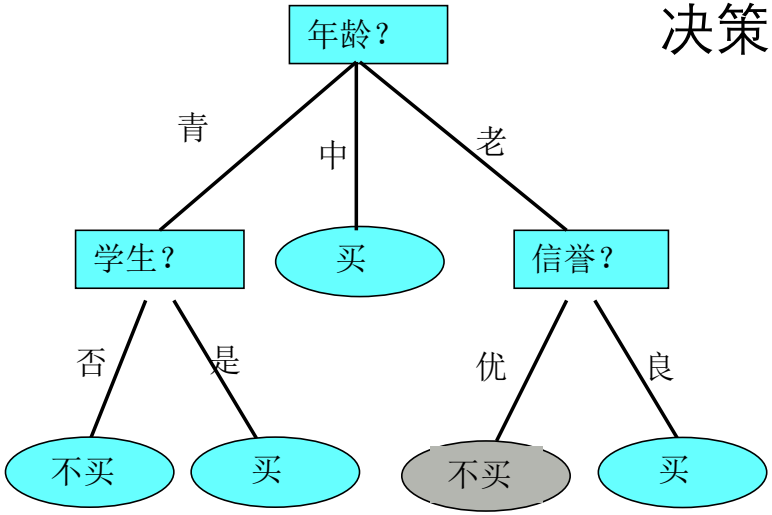
决策树算法

谁在买计算机？



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

决策树的表示



决策树的基本组成部分：决策结点、分支和叶子。

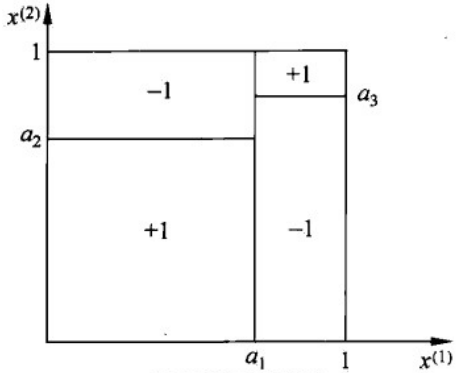
决策树中最上面的结点称为根结点。是整个决策树的开始。每个分支是一个新的决策结点，或者是树的叶子。每个决策结点代表一个问题或者决策.通常对应待分类对象的属性。每个叶结点代表一种可能的分类结果

在沿着决策树从上到下的遍历过程中，在每个结点都有一个测试。对每个结点上问题的不同测试输出导致不同的分枝，最后会达到一个叶子结点。这一过程就是利用决策树进行分类的过程，利用若干个变量来判断属性的类别

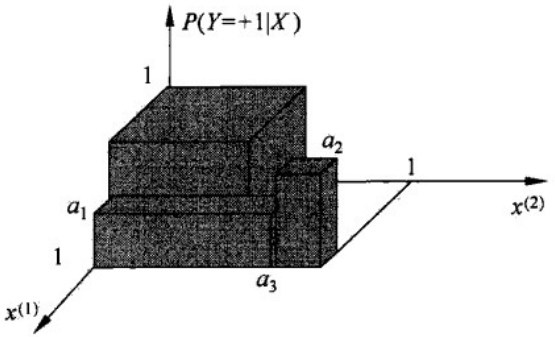
决策树与条件概率分布

- 决策树表示给定特征条件下类的条件概率分布。
- 条件概率分布定义在特征空间的一个划分(partition)上.将特征空间划分为互不相交的单元(cell)或区域(region),并在每个单元定义一个类的概率分布就构成了一个条件概率分布。
- 决策树的一条路径对应于划分中的一个单元。
- 决策树所表示的条件概率分布由各个单元给定条件下类的条件概率分布组成

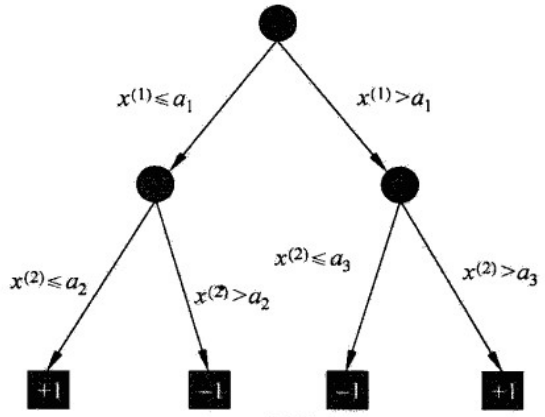
决策树与条件概率分布



(a) 特征空间划分



(b) 条件概率分布



(c) 决策树

决策树与条件概率分布

- 决策树学习本质上是从训练数据集中归纳出一组分类规则，与训练数据集不相矛盾的决策树。
- 能对训练数据进行正确分类的决策树可能有多个，也可能 一个也没有.我们需要的是一个与训练数据矛盾较小的决策树，同时具有很好的 泛化能力.
- 决策树学习是由训练数据集估计条件概率模型.基于特征空间划分的类的条件概率模型有无穷多个.
- 我们选择的条件概率模型应该不仅对训练数据有很好的拟合，而且对未知数据有很好的预测.

决策树CLS算法

- CLS（Concept Learning System）算法
  - CLS算法是早期的决策树学习算法。它是许多决策树学习算法的基础
- CLS基本思想
  - 从一棵空决策树开始，选择某一属性（分类属性）作为测试属性。该测试属性对应决策树中的决策结点。根据该属性的值的不同，可将训练样本分成相应的子集：
    - 如果该子集为空，或该子集中的样本属于同一个类，则该子集为叶结点，
    - 否则该子集对应于决策树的内部结点，即测试结点，需要选择一个新的分类属性对该子集进行划分，直到所有的子集都为空或者属于同一类。

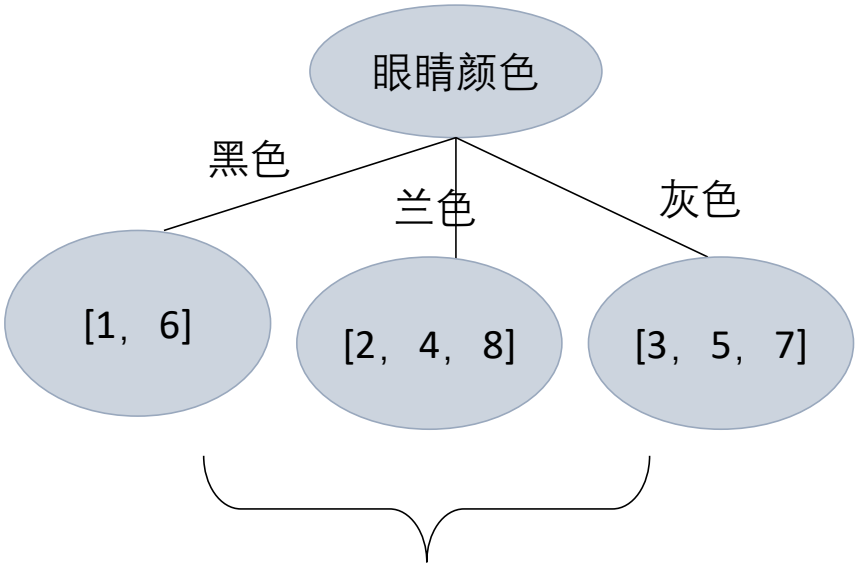
决策树CLS算法

人员	眼睛颜色	头发颜色	所属人种
1	黑色	黑色	黄种人
2	蓝色	金色	白种人
3	灰色	金色	白种人
4	蓝色	红色	白种人
5	灰色	红色	白种人
6	黑色	金色	混血
7	灰色	黑色	混血
8	蓝色	黑色	混血

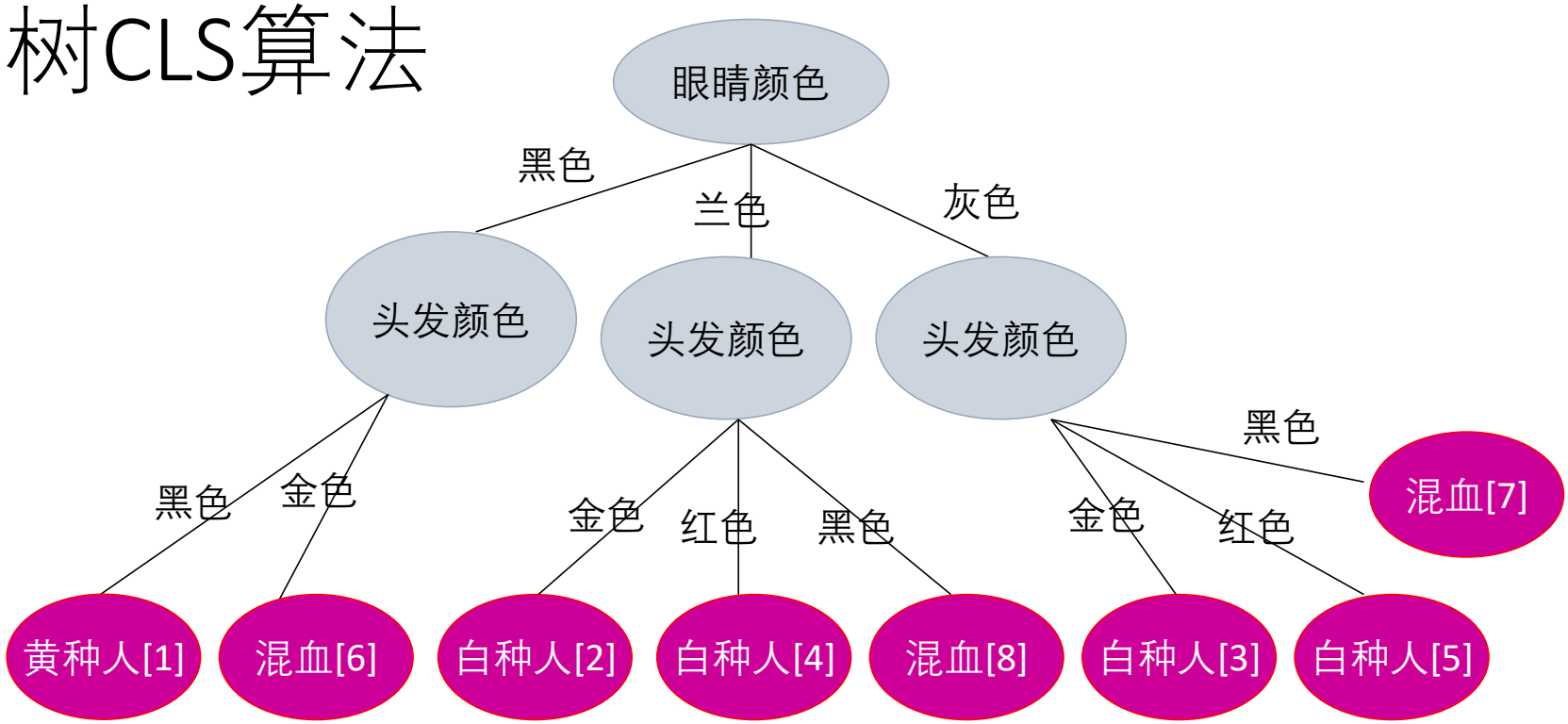
决策树CLS算法

决策树的构建

人员	眼睛颜色	头发颜色	所属人种
1	黑色	黑色	黄种人
2	蓝色	金色	白种人
3	灰色	金色	白种人
4	蓝色	红色	白种人
5	灰色	红色	白种人
6	黑色	金色	混血
7	灰色	黑色	混血
8	蓝色	黑色	混血



决策树CLS算法


眼睛颜色
黑色
兰色
灰色
头发颜色
头发颜色
头发颜色
黑色
金色
金色
红色
黑色
黑色
金色
红色
黄种人[1]
混血[6]
白种人[2]
白种人[4]
混血[8]
白种人[3]
白种人[5]
混血[7]

决策树CLS算法

- 步骤:
  - 生成一颗空决策树和一张训练样本属性集;
  - 若训练样本集T 中所有的样本都属于同一类,则生成结点T , 并终止学习算法;否则
  - 根据某种策略从训练样本属性表中选择属性A 作为测试属性, 生成测试结点A
  - 若A的取值为v1,v2,…,vm, 则根据A 的取值的不同,将T 划分成 m个子集 T1,T2,…,Tm;
  - 从训练样本属性表中删除属性A;
  - 转步骤2, 对每个子集递归调用CLS;

决策树CLS算法

- CLS算法问题：
  - 在步骤3中，根据某种策略从训练样本属性表中选择属性A作为测试属性。没有规定采用何种测试属性。实践表明，测试属性集的组成以及测试属性的先后对决策树的学习具有举足轻重的影响。

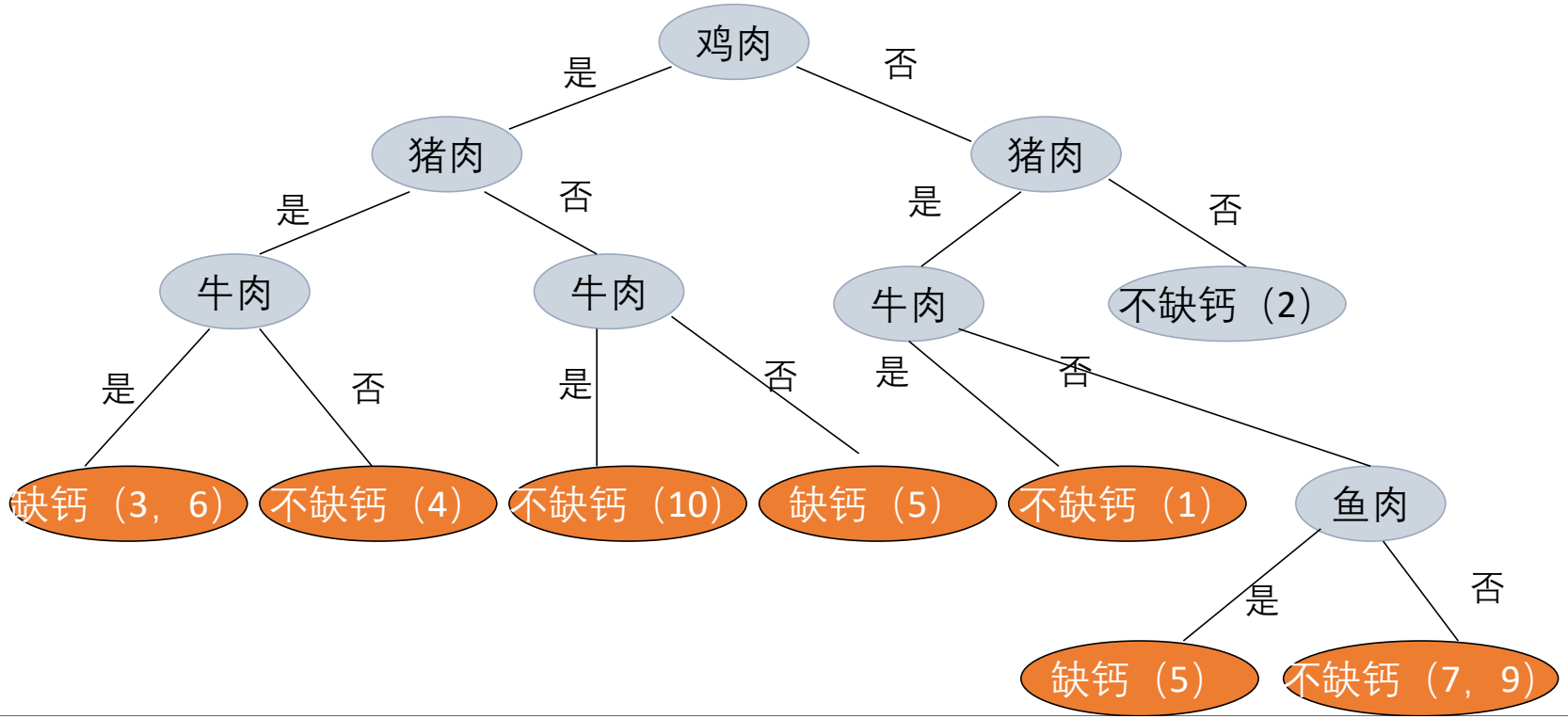
决策树CLS算法

学生膳食结构和缺钙调查表

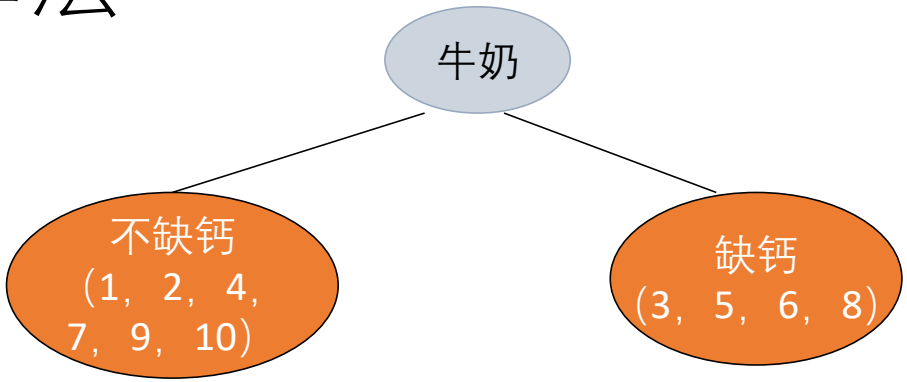
学生	鸡肉	猪肉	牛肉	羊肉	鱼肉	鸡蛋	青菜	番茄	牛奶	健康情况
1	0	1	1	0	1	0	1	0	1	不缺钙
2	0	0	0	0	1	1	1	1	1	不缺钙
3	1	1	1	1	1	0	1	0	0	缺钙
4	1	1	0	0	1	1	0	0	1	不缺钙
5	1	0	0	1	1	1	0	0	0	缺钙
6	1	1	1	0	0	1	0	1	0	缺钙
7	0	1	0	0	0	1	1	1	1	不缺钙
8	0	1	0	0	0	1	1	1	1	缺钙
9	0	1	0	0	0	1	1	1	1	不缺钙
10	1	0	1	1	1	1	0	1	1	不缺钙

决策树CLS算法

采用不同的测试属性及其先后顺序将会生成不同的决策树



决策树CLS算法



决策树ID3算法

- ID3算法是一种经典的决策树学习算法，由Quinlan于1979年提出。
- ID3算法主要针对属性选择问题。是决策树学习方法中最具影响和最为典型的算法。
- 该方法使用信息增益度选择测试属性。
- 当获取信息时，将不确定的内容转为确定的内容，因此信息伴着不确定性。
- 从直觉上讲，小概率事件比大概率事件包含的信息量大。如果某件事情是“百年一见”则肯定比“习以为常”的事件包含的信息量大。

- **如何度量信息量的大小?**

信息增益

- Shannon 1948年提出的信息论理论：
- 熵(entropy)：信息量大小的度量，即表示随机变量不确定性的度量。
- 熵的通俗解释：事件a_i的信息量$I(a_i)$可如下度量：

$$I(a_i) = p(a_i)\log_2 \frac{1}{p(a_i)}$$

- 其中$p(a_i)$表示事件a_i发生的概率。
- 假设有n个互不相容的事件a1,a2,a3,….,an,它们中有且仅有一个发生，则其平均的信息量(熵)可如下度量：

$$I(a_1, a_2, ..., a_n) = \sum_{i=1}^{n} I(a_i) = \sum_{i=1}^{n} p(a_i)\log_2 \frac{1}{p(a_i)}$$

信息增益

- 熵的理论解释:
- 设X是一个取有限个值的离散随机变量，其概率分布为:

$$P(X=x_i)=p_i\,,\quad i=1,2,\cdots,n$$

- 则随机变量X的熵定义为:

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

- 对数以2为底或以e为底(自然对数)，这时熵的单位分别称作比特(bit)或纳特(nat),熵只依赖于X的分布，与X的取值无关。

$$H(p)=-\sum_{i=1}^{n}p_i\log p_i$$

信息增益

- 熵的理论解释:
- 熵越大，随机变量的不确定性越大:

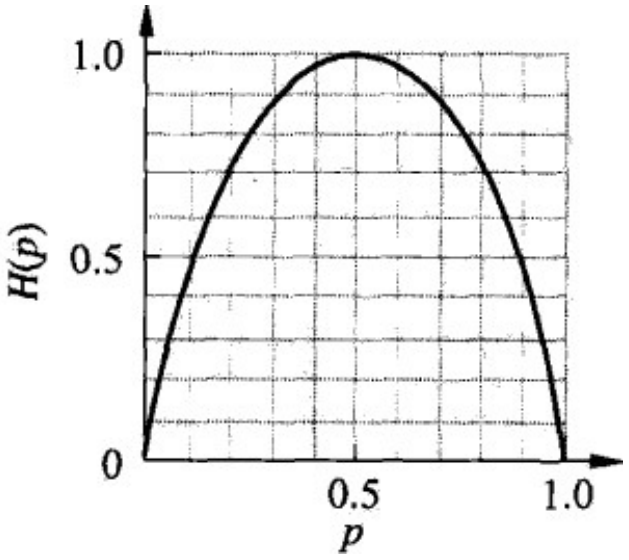
$$0 \leqslant H(p) \leqslant \log n$$

- 当X为1,0分布时:

$$P(X=1)=p\,, \quad P(X=0)=1-p\,, \quad 0 \leqslant p \leqslant 1$$

- 熵:

$$H(p)=-p\log_2 p-(1-p)\log_2(1-p)$$



信息增益

- 设有随机变量(X,Y),其联合概率分布为:

$$P(X=x_i, Y=y_j)=p_{ij}, \quad i=1,2,\cdots,n; \quad j=1,2,\cdots,m$$

- 条件熵H(Y|X)：表示在已知随机变量X的条件下随机变量Y的不确定性,定义为X给定条件下Y的条件概率分布的熵对X的数学期望:

$$H(Y|X)=\sum_{i=1}^{n} p_i H(Y|X=x_i)$$

- 当熵和条件熵中的概率由数据估计（特别是极大似然估计）得到时,所对应的熵与条件熵分别称为经验熵（empirical entropy)和经验条件熵（empirical conditional entropy)

信息增益

- 定义5.2 (信息增益):特征A对训练数据集D的信息增益，g(D,A), 定义为集合D的经验熵H(D)与特征A给定条件下D的经验条件熵H(D|A)之差，即

$$g(D,A)=H(D)-H(D|A)$$

- (Information gain)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.
- 一般地，熵H(Y)与条件熵H(Y|X)之差称为互信息（mutual information)
- 决策树学习中的信息增益等价于训练数据集中类与特征的互信息.

信息增益的算法

- 设训练数据集为D
- |D|表示其样本容量，即样本个数
- 设有K个类C_k, k = 1,2，…K,
- $|C_k|$为属于类C_k的样本个数
- 特征A有n个不同的 取值{a_1,a_2…a_n}根据特征A的取值将D划分为n个子集D_1.。。D_n
- $|D_i|$为 D_i的样本个数
- 记子集D_i中属于类C_k的样本集合为D_{ik}
- $|D_{ik}|$为D_{ik}的样本个数

信息增益的算法

- 输入：训练数据集D和特征A;
- 输出：特征A对训练数据集D的信息增益g(D,A)
- 1、计算数据集D的经验熵H(D)

$$H(D)=-\sum_{k=1}^{K}\frac{|C_k|}{|D|}\log_2\frac{|C_k|}{|D|}$$

- 2、计算特征A对数据集D的经验条件熵H(D|A)

$$H(D|A)=\sum_{i=1}^{n}\frac{|D_i|}{|D|}H(D_i)=-\sum_{i=1}^{n}\frac{|D_i|}{|D|}\sum_{k=1}^{K}\frac{|D_{ik}|}{|D_i|}\log_2\frac{|D_{ik}|}{|D_i|}$$

- 3、计算信息增益

$$g(D,A)=H(D)-H(D|A)$$

信息增益比

- 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

- 定义5.3（信息增益比） 特征A对训练数据集D的信息增益比定义为信息增益与训练数据集D关于特征A的值的熵之比

$$g_R(D, A) = \frac{g(D, A)}{H_A(D)}$$

$$H_A(D) = -\sum_{i=1}^{n} \frac{|D_i|}{|D|} \log_2 \frac{|D_i|}{|D|}, \ n \text{ 是特征 } A \text{ 取值的个数。}$$

决策树ID3算法

- 在决策树分类中，假设S是训练样本集合，|S|是训练样本数，样本划分为n个不同的类C1,C2,….Cn，这些类的大小分别标记为|C1|，|C2|，…..,|Cn|。则任意样本S属于类Ci的概率为：

$$p(S_i) = \frac{|C_i|}{|S|}$$

Entropy（S|A）=∑（|Sv|/|S|）* Entropy（Sv）

- ∑是属性A的所有可能的值v, Sv是属性A有v值的S子集
- |Sv|是Sv 中元素的个数；|S|是S中元素的个数。

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

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1步计算决策属性的熵

决策属性“买计算机？”。
该属性分两类：买/不买

|C1|(买)=641
|C2|（不买）= 383
|D|=|C1|+|C2|=1024

P1=641/1024=0.6260
P2=383/1024=0.3740

$H(D)=-P1\text{Log}_2P1-P2\text{Log}_2P2$
$=-(P1\text{Log}_2P1+P2\text{Log}_2P2)$
$=0.9537$

$$H(D)=-\sum_{k=1}^{K}\frac{|C_k|}{|D|}\log_2\frac{|C_k|}{|D|}$$

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步计算条件属性的熵

条件属性共有4个:
年龄、收入、学生、信誉。
分别计算不同属性的信息增益。

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2-1步计算年龄的熵

年龄共分三个组：
青年、中年、老年
青年买与不买比例为128/256

|D11|(买)=128
|D12|(不买)= 256
|D1|=384

P1=128/384
P2=256/384

$H(D1)=-P1\log_2 P1-P2\log_2 P2$
$=-(P1\log_2 P1+P2\log_2 P2)$
$=0.9183$

$$H(D\mid A)=\sum_{i=1}^{n}\frac{|D_i|}{|D|}H(D_i)=-\sum_{i=1}^{n}\frac{|D_i|}{|D|}\sum_{k=1}^{K}\frac{|D_{ik}|}{|D_i|}\log_2\frac{|D_{ik}|}{|D_i|}$$

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2-2步计算年龄的熵

年龄共分三个组：
青年、中年、老年
中年买与不买比例为256/0

|D21|(买)=256
|D22|(不买)= 0
|D2|=256

P1=256/256
P2=0/256

$H(D2)=-P1Log_2P1-P2Log_2P2$
$=-(P1Log_2P1+P2Log_2P2)$
$=0$

Tsinghua University

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2-3步计算年龄的熵

年龄共分三个组：
青年、中年、老年
老年买与不买比例为125/127

|D31|(买)=125
|D32|（不买）=127
|D3|=S1+S2=252

P1=125/252
P2=127/252

$H(D3)=-P1Log_2P1-P2Log_2P2$
$=-(P1Log_2P1+P2Log_2P2)$
$=0.9157$

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2-4步计算年龄的熵

年龄共分三个组：
青年、中年、老年
所占比例
青年组 384/1025=0.375
中年组 256/1024=0.25
老年组 384/1024=0.375

计算年龄的平均信息期望
E（年龄）=0.375*0.9183+
0.25*0+
0.375*0.9157
=0.6877
G（年龄信息增益）
=0.9537-0.6877
=0.2660 （1）

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 计算收入的熵

收入共分三个组：
高、中、低

E（收入）=0.9361

收入信息增益=0.9537-0.9361
=0.0176 (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步计算学生的熵

学生共分二个组：

学生、非学生

E（学生）=0.7811

年龄信息增益=0.9537-0.7811

=0.1726 （3）

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5步计算信誉的熵

信誉分二个组：

良好，优秀

E（信誉）= 0.9048

信誉信息增益=0.9537-0.9048

=0.0453 (4)

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6步计算选择节点

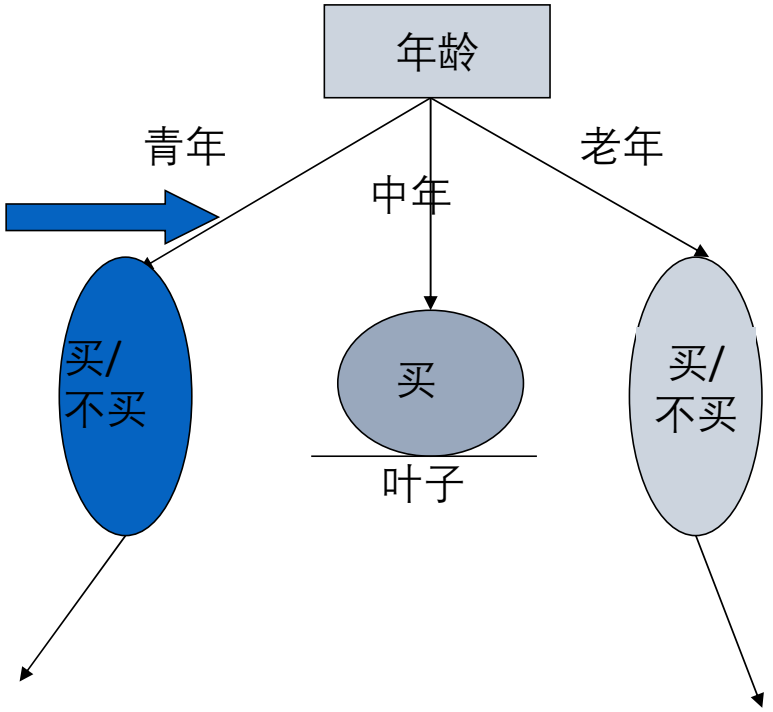
年龄信息增益=0.9537-0.6877
=0.2660 （1）

收入信息增益=0.9537-0.9361
=0.0176 （2）

年龄信息增益=0.9537-0.7811
=0.1726 （3）

信誉信息增益=0.9537-0.9048
=0.0453 （4）

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



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

青年买与不买比例为
128/256

|C1|(买)=128
|C2|（不买）= 256
|D|=384

P1=128/384
P2=256/384

H(D)=-P1Log2P1-P2Log2P2
=-(P1Log2P1+P2Log2P2)
=0.9183

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

如果选择收入作为节点
分高、中、低

H(D1)=0
比例:
128/384=0.3333

H(D2)=0.9183
比例: 192/384=0.5

H(D3)=0
比例: 64/384=0.1667

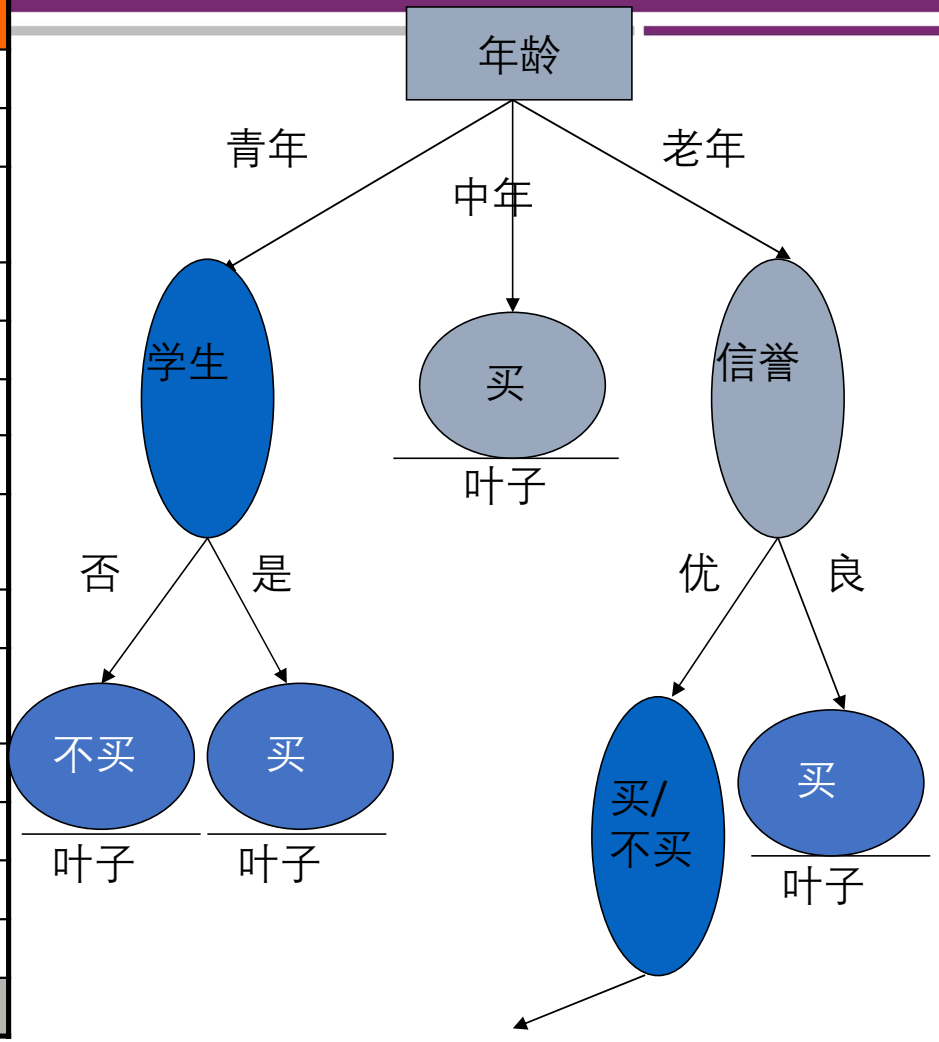
平均信息期望（加权总和）：

E(收入) = 0.3333 * 0 + 0.5 * 0.9183 + 0.1667 * 0 = 0.4592

Gain(收入) = I(128, 256) - E(收入)=0.9183 – 0.4592 = 0.4591

注意

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



决策树ID3算法-流程

- 1 决定分类属性;
- 2 对目前的数据表，建立一个节点N
- 3 如果数据库中的数据都属于同一个类，N就是树叶，在树叶上标出所属的类
- 4 如果数据表中没有其他属性可以考虑，则N也是树叶，按照少数服从多数的原则在树叶上标出所属类别
- 5 否则，根据平均信息期望值E或GAIN值选出一个最佳属性作为节点N的测试属性
- 6 节点属性选定后，对于该属性中的每个值:
  - 从N生成一个分支，并将数据表中与该分支有关的数据收集形成分支节点的数据表，在表中删除节点属性那一栏如果分支数据表非空，则运用以上算法从该节点建立子树。

决策树ID3算法-实际使用

原始表

姓名	年龄	收入	学生	信誉	电话	地址	邮编	买计算机
张三	23	4000	是	良	281-322-0328	2714 Ave. M	77388	买
李四	34	2800	否	优	713-239-7830	5606 Holly Cr	78766	买
王二	70	1900	否	优	281-242-3222	2000 Bell Blvd.	70244	不买
赵五	18	900	是	良	281-550-0544	100 Main Street	70244	买
刘兰	34	2500	否	优	713-239-7430	606 Holly Ct	78566	买
杨俊	27	8900	否	优	281-355-7990	233 Rice Blvd.	70388	不买
张毅	38	9500	否	优	281-556-0544	399 Sugar Rd.	78244	买
。。。	。。							
。。。								

决策树ID3算法-实际使用

决策树的数据准备

整理后的数据表

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
。。。					

- Data cleaning
  删除/减少noise，
  补填missing values
- Data transformation
  数据标准化（data normalization）
  数据归纳（generalize data to higher-level concepts using concept hierarchies）
  例如：年龄归纳为老、中、青三类
  控制每个属性的可能值不超过七种（最好不超过五种）
- Relevance analysis
  对于与问题无关的属性：删
  对于属性的可能值大于七种又不能归纳的属性：删

决策树ID3算法-小结

- ID3算法的基本思想是，以信息熵为度量，用于决策树节点的属性选择，每次优先选取信息量最多的属性，亦即能使熵值变为最小的属性，以构造一颗熵值下降最快的决策树，到叶子节点处的熵值为0。此时，每个叶子节点对应的实例集中的实例属于同一类。

决策树面临的问题

- 理想的决策树有三种:
  - (1)叶子结点数最少;
  - (2)叶子结点深度最小;
  - (3)叶子结点数最少且叶子结点深度最小。
- 然而，洪家荣等人已经证明了要找到这种最优的决策树是NP难题。因此，决策树优化的目的就是要找到尽可能趋向于最优的决策树。

决策树面临的问题

- 过度拟合
- 决策树算法增长树的每一个分支的深度，直到恰好能对训练样例比较完美地分类。实际应用中，当数据中有噪声或训练样例的数量太少以至于不能产生目标函数的有代表性的采样时，该策略可能会遇到困难。
- 在以上情况发生时，这个简单的算法产生的树会过渡拟合训练样例（过渡拟合：Over Fitting）.

决策树面临的问题

- 过度拟合
- 对学习算法是否成功的真正测试是看它对于训练中未见到的数据的执行性能。训练过程应该包含训练样本和验证样本。验证样本用于测试训练后的性能。如果验证结果差，则需要考虑采用不同的结构重新进行训练，例如使用更大的样本集，或者改变从连续值到离散值得数据转换等。

- 通常应该建立一个验证过程，在训练最终完成后用来检测训练结果的泛化能力。

决策树面临的问题

- 一般可以将分类模型的误差分为：
  - 1、训练误差（Training Error）；
  - 2、泛化误差（Generalization Error）
- 训练误差是在训练记录上误分类样本比例；
- 泛化误差是模型在未知记录上的期望误差；

- 一个好的模型不仅要能够很好地拟合训练数据，而且对未知样本也要能够准确地分类。
- 一个好的分类模型必须具有低的训练误差和泛化误差。因为一个具有低训练误差的模型，其泛化误差可能比具有较高训练误差的模型高。（训练误差低，泛化误差高，称为过渡拟合）

决策树面临的问题

- 决策树算法比较适合处理离散数值的属性。实际应用中属性是连续的或者离散的情况都比较常见。

- 在应用连续属性值时，在一个树结点可以将属性Ai的值划分为几个区间。然后信息增益的计算就可以采用和离散值处理一样的方法。原则上可以将Ai的属性划分为任意数目的空间。C4.5中采用的是二元分割（Binary Split）。需要找出一个合适的分割阈值。
- **本书第九章将介绍**

  参考C4.5算法 Top 10 algorithms in data mining

  Knowledge Information System 2008 14:1–37

决策树的剪枝

- 通过极小化决策树整体的损失函数或代价函数来实现。
- 设树T的叶结点个数为|T|,t是树T的叶结点，该叶结点有N_t个样本点，其中k类的样本点有N_{tk}个，k=1,2..K,
- $H_t(T)$为叶结点t上的经验熵，α≥0为参数，损失函数：

$$C_\alpha(T)=\sum_{t=1}^{|T|}N_tH_t(T)+\alpha|T|$$

- 经验熵：

$$H_t(T)=-\sum_k\frac{N_{tk}}{N_t}\log\frac{N_{tk}}{N_t}$$

- 原式第一项：

$$C(T)=\sum_{t=1}^{|T|}N_tH_t(T)=-\sum_{t=1}^{|T|}\sum_{k=1}^{K}N_{tk}\log\frac{N_{tk}}{N_t}$$

- 则：

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

决策树的剪枝

- 树的剪枝算法:

输入：生成算法产生的整个树 T，参数 α；

输出：修剪后的子树 T_α.

（1）计算每个结点的经验熵.

（2）递归地从树的叶结点向上回缩.

- 设一组叶结点回缩到其父结点之前与之和的损失函数分别为： $C_\alpha(T_B)$ 与 $C_\alpha(T_A)$
- 如果： $C_\alpha(T_A) \leqslant C_\alpha(T_B)$ 则进行剪枝

（3）返回 (2)，直至不能继续为止，得到损失函数最小的子树 T_α.

CART树

- 分类回归树CART(Classification and Regression Trees)
  - 1984 << Classification and Regression Trees >>
- L.Breiman，J.Friedman，R.Olshen和C.Stone
  - http://www.stat.berkeley.edu/~breiman/
  - http://www-stat.stanford.edu/~jhf/
  - http://www-stat.stanford.edu/~olshen/

- 目标变量是类别的 --- 分类树
- 目标变量是连续的 --- 回归树

CART与ID3的不同

- 二元划分：
  - 二叉树不易产生数据碎片，精确度往往也会高于多叉树
- CART中选择变量的不纯性度量：
  - 分类目标：Gini指标、Towing、order Towing
  - 连续目标：最小平方残差、最小绝对残差
- 剪枝：
  - 用预剪枝或后剪枝对训练集生长的树进行剪枝
- 树的建立：
  - 如果目标变量是标称的，并且是具有两个以上的类别，则CART可能考虑将目标类别合并成两个超类别（双化）；
  - 如果目标变量是连续的，则CART算法找出一组基于树的回归方程来预测目标变量。

CART树

- CART算法由两部分组成:
  - 决策树生成
  - 决策树剪枝
- 回归树: 平方误差最小化
- 分类树: Gini Index

CART的生成

- 回归树的生成
- 设Y是连续变量，给定训练数据集： $D=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}$
- 假设已将输入空间划分为M各单元R1,R2..Rm,并且每个单元Rm上有一个固定的输出Cm，回归树表示为：

$$f(x)=\sum_{m=1}^{M}c_m I(x\in R_m)$$

- 平方误差来表示预测误差，用平方误差最小准则求解每个单元上的最优输出值

$$\sum_{x_i\in R_m}(y_i-f(x_i))^2$$

- Rm上的Cm的最优值： $\hat{c}_m=\text{ave}(y_i\mid x_i\in R_m)$

CART的生成

- 问题：如何对输入空间进行划分？
- 启发式：选择第j个变量$x^{(j)}$和它取的值s，作为切分变量和切分点，定义两个区域：

$$R_1(j,s)=\{x \mid x^{(j)} \leqslant s\} \quad 和 \quad R_2(j,s)=\{x \mid x^{(j)} > s\}$$

- 然后寻找最优切分变量和切分点：

$$\min_{j,s}\left[\min_{c_1}\sum_{x_i\in R_1(j,s)}(y_i-c_1)^2+\min_{c_2}\sum_{x_i\in R_2(j,s)}(y_i-c_2)^2\right]$$

- 且：$\hat{c}_1=\text{ave}(y_i \mid x_i\in R_1(j,s))$ 和 $\hat{c}_2=\text{ave}(y_i \mid x_i\in R_2(j,s))$

- 再对两个区域重复上述划分，直到满足停止条件。

CART的生成

- 最小二乘回归树生成算法

输入：训练数据集 D；

输出：回归树 $f(x)$.

在训练数据集所在的输入空间中，递归地将每个区域划分为两个子区域并决定每个子区域上的输出值，构建二叉决策树：

（1）选择最优切分变量 j 与切分点 s，求解

$$\min_{j,s}\left[\min_{c_1}\sum_{x_i\in R_1(j,s)}(y_i-c_1)^2+\min_{c_2}\sum_{x_i\in R_2(j,s)}(y_i-c_2)^2\right]$$

遍历变量 j，对固定的切分变量 j 扫描切分点 s，选择使式(
达到最小值的对 (j,s)

CART的生成

- 最小二乘回归树生成算法

（2）用选定的对 (j,s) 划分区域并决定相应的输出值：

$$R_1(j,s)=\{x \mid x^{(j)} \leqslant s\}, \quad R_2(j,s)=\{x \mid x^{(j)} > s\}$$

$$\hat{c}_m=\frac{1}{N_m}\sum_{x_i \in R_m(j,s)} y_i, \quad x \in R_m, \quad m=1,2$$

（3）继续对两个子区域调用步骤 (1)，(2)，直至满足停止条件.

（4）将输入空间划分为 M 个区域 $R_1, R_2, \cdots, R_M$，生成决策树：

$$f(x)=\sum_{m=1}^{M} \hat{c}_m I(x \in R_m)$$

CART的生成

- 分类树的生成:
- 基尼指数
- 分类问题中，假设有k个类，样本点属于k的概率Pk，则概率分布的基尼指数:

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

- 二分类问题: $\text{Gini}(p)=2p(1-p)$
- 对给定的样本集合D，基尼指数 $\text{Gini}(D)=1-\sum_{k=1}^{K}\left(\frac{|C_k|}{|D|}\right)^2$

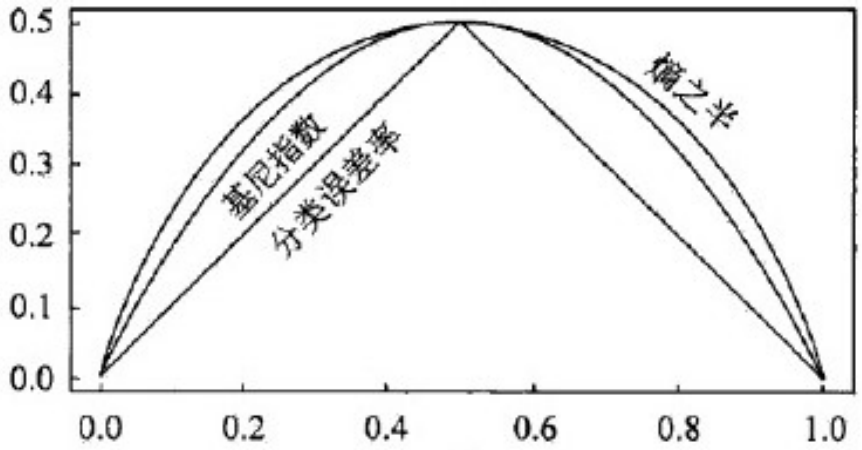
CART的生成

- 如果样本集合D根据特征A是否为a被分割成D1和D2，即

$$D_1 = \{(x, y) \in D \mid A(x) = a\}, \quad D_2 = D - D_1$$

- 则在特征A的条件下，集合D的基尼指数：

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



CART的生成

- CART生成算法
- 输入：训练数据集D，停止计算条件
- 输出：CART决策树
- 从根节点开始，递归对美国结点操作
- 1、设结点数据集为D，对每个特征A，对其每个值a，根据样本点对A=a的测试为是或否，将D分为D1，D2，计算A=a的基尼指数
- 2、在所有的特征A以及所有可能的切分点a中，选择基尼指数最小的特征和切分点，将数据集分配到两个子结点中。
- 3、对两个子结点递归调用1，2步骤
- 4、生成CART树

CART的生成

- CART剪枝
- 两步
- 1、从生成算法产生的决策树T0底端开始不断剪枝，直到T0的根结点，形成子树序列{T0,T1..Tn},
- 2、通过交叉验证法在独立的验证数据集上对子树序列进行测试，从中选择最优子树

CART的生成

- CART剪枝
- 1、剪枝，形成子树序列
- 剪枝过程中，计算子树的损失函数:

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

- 对固定的a一定存在损失函数最小的子树，表示为Ta，
- 当a变大时，最优子树Ta偏小，
- a=0时，整体树最优，a趋近无穷大，单结点最优
- 将a从小增大，0= $\alpha_0 < \alpha_1 < \cdots < \alpha_n < +\infty$

最优子树序列 $\{T_0, T_1, \cdots, T_n\}$

CART的生成

- CART剪枝
- 1、剪枝，形成子树序列
- 具体：从T0开始剪枝，以t为单结点树的损失函数：

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

- 以t为根结点的子树T_t的损失函数：

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

- 当a=0及a很小时，$C_\alpha(T_t) < C_\alpha(t)$
- 不断增大a，当 $C_\alpha(T_t) = C_\alpha(t)$ $\alpha = \dfrac{C(t) - C(T_t)}{|T_t| - 1}$

- Tt与t有相同损失函数值，但t结点更少，所以剪枝Tt。

CART的生成

- CART剪枝
- 1、剪枝，形成子树序列
- 对T0中每个内部结点t，计算：

$$g(t)=\frac{C(t)-C(T_t)}{|T_t|-1}$$

- 在T0中剪去g(t)最小的Tt，将得到的子树作为T1，同时将最小的g(t)设为a1，T1为区间[a1,a2) 的最优子树
- 如此剪枝下去，直到根节点，不断增加a的值，产生新的区间。

CART的生成

- CART剪枝
- 2、在剪枝得到的子树序列{T0,T1…Tn}中通过交叉验证选取最优子树Ta
- 利用独立的验证数据集，测试子树序列中各子树的平方误差或基尼指数，最小的决策树就是最优决策树。

决策树的Python实现

- 计算香农熵
  - def calcShannonEnt(dataSet):
  - numEntries = len(dataSet)
  - labelCounts = {}
  - for featVec in dataSet: #the the number of unique elements and their occurance
  - currentLabel = featVec[-1]
  - if currentLabel not in labelCounts.keys():
  - labelCounts[currentLabel] = 0
  - labelCounts[currentLabel] += 1
  - shannonEnt = 0.0
  - for key in labelCounts:
  - prob = float(labelCounts[key])/numEntries
  - shannonEnt -= prob * log(prob,2) #log base 2
  - return shannonEnt

决策树的Python实现

- 划分数据集：参数包括数据集、特征、值
  - def splitDataSet(dataSet, axis, value):
  - retDataSet = []
  - for featVec in dataSet:
  - if featVec[axis] == value:
  - reducedFeatVec = featVec[:axis] #chop out axis used for splitting
  - reducedFeatVec.extend(featVec[axis+1:])
  - retDataSet.append(reducedFeatVec)
  - return retDataSet

决策树的Python实现

- 选择最好的数据集划分方式
  - def chooseBestFeatureToSplit(dataSet):
  - numFeatures = len(dataSet[0]) - 1 #the last column is used for the labels
  - baseEntropy = calcShannonEnt(dataSet)
  - bestInfoGain = 0.0; bestFeature = -1
  - for i in range(numFeatures): #iterate over all the features
  - featList = [example[i] for example in dataSet]#create a list of all the examples of this feature
  - uniqueVals = set(featList) #get a set of unique values
  - newEntropy = 0.0
  - for value in uniqueVals:

决策树的Python实现

- 选择最好的数据集划分方式
  - for value in uniqueVals:
  - subDataSet = splitDataSet(dataSet, i, value)
  - prob = len(subDataSet)/float(len(dataSet))
  - newEntropy += prob * calcShannonEnt(subDataSet)
  - infoGain = baseEntropy - newEntropy #calculate the info gain; ie reduction in entropy
  - if (infoGain > bestInfoGain): #compare this to the best gain so far
  - bestInfoGain = infoGain #if better than current best, set to best
  - bestFeature = i
  - return bestFeature

决策树的Python实现

- 多数表决的方法决定叶子节点的分类
- def majorityCnt(classList):
- classCount={}
- for vote in classList:
- if vote not in classCount.keys(): classCount[vote] = 0
- classCount[vote] += 1
- sortedClassCount = sorted(classCount.iteritems(), key=operator.itemgetter(1), reverse=True)
- return sortedClassCount[0][0]

决策树的Python实现

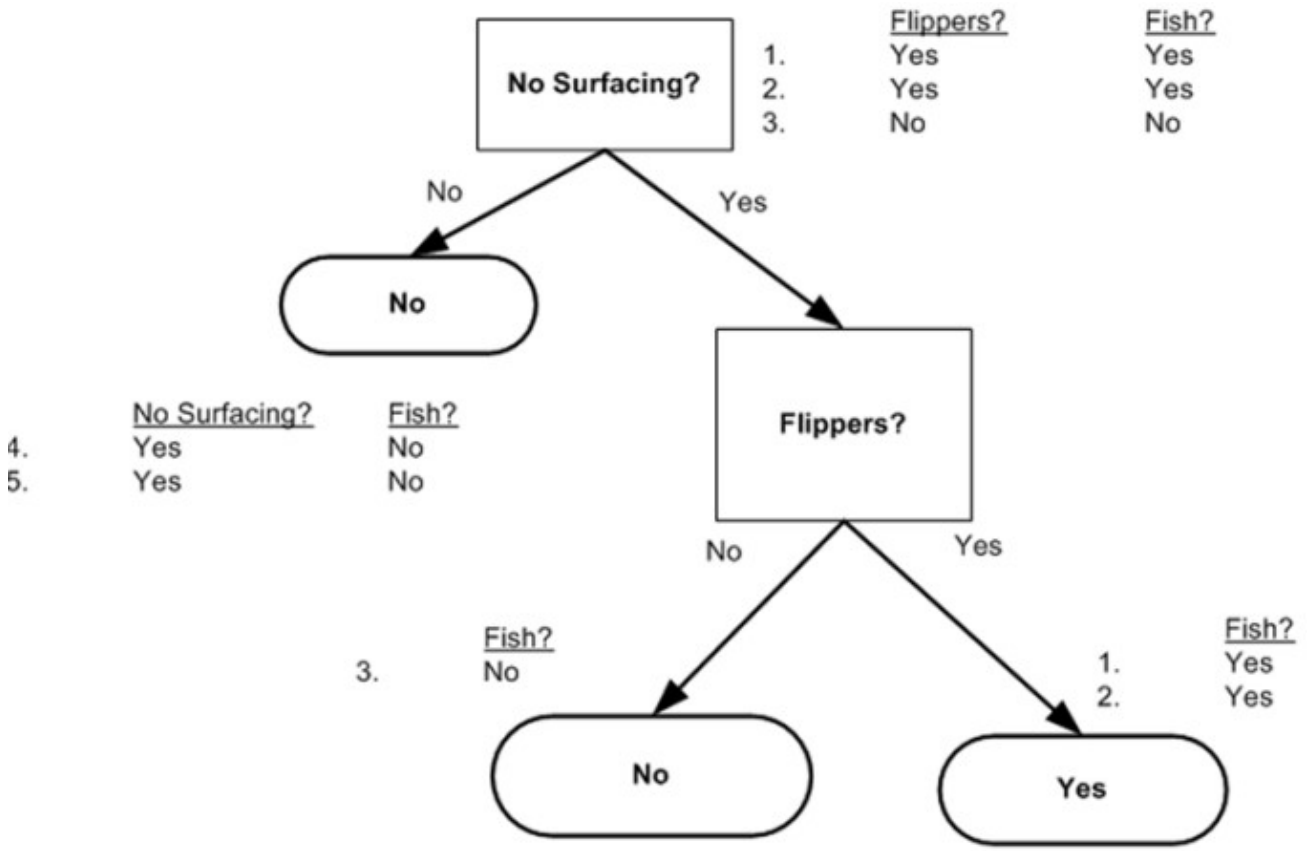
- 创建树
- def createTree(dataSet,labels):
- classList = [example[-1] for example in dataSet]
- if classList.count(classList[0]) == len(classList):
- return classList[0]#stop splitting when all of the classes are equal
- if len(dataSet[0]) == 1: #stop splitting when there are no more features in dataSet
- return majorityCnt(classList)
- bestFeat = chooseBestFeatureToSplit(dataSet)
- bestFeatLabel = labels[bestFeat]

决策树的Python实现

- 创建树
- bestFeatLabel = labels[bestFeat]
- myTree = {bestFeatLabel:{}}
- del(labels[bestFeat])
- featValues = [example[bestFeat] for example in dataSet]
- uniqueVals = set(featValues)
- for value in uniqueVals:
- subLabels = labels[:] #copy all of labels, so trees don't mess up existing labels
- myTree[bestFeatLabel][value] = createTree(splitDataSet(dataSet, bestFeat, value),subLabels)
- return myTree

	No surfacing	Flippers?	Fish?
1.	Yes	Yes	Yes
2.	Yes	Yes	Yes
3.	Yes	No	No
4.	No	Yes	No
5.	No	Yes	No

- 创建树



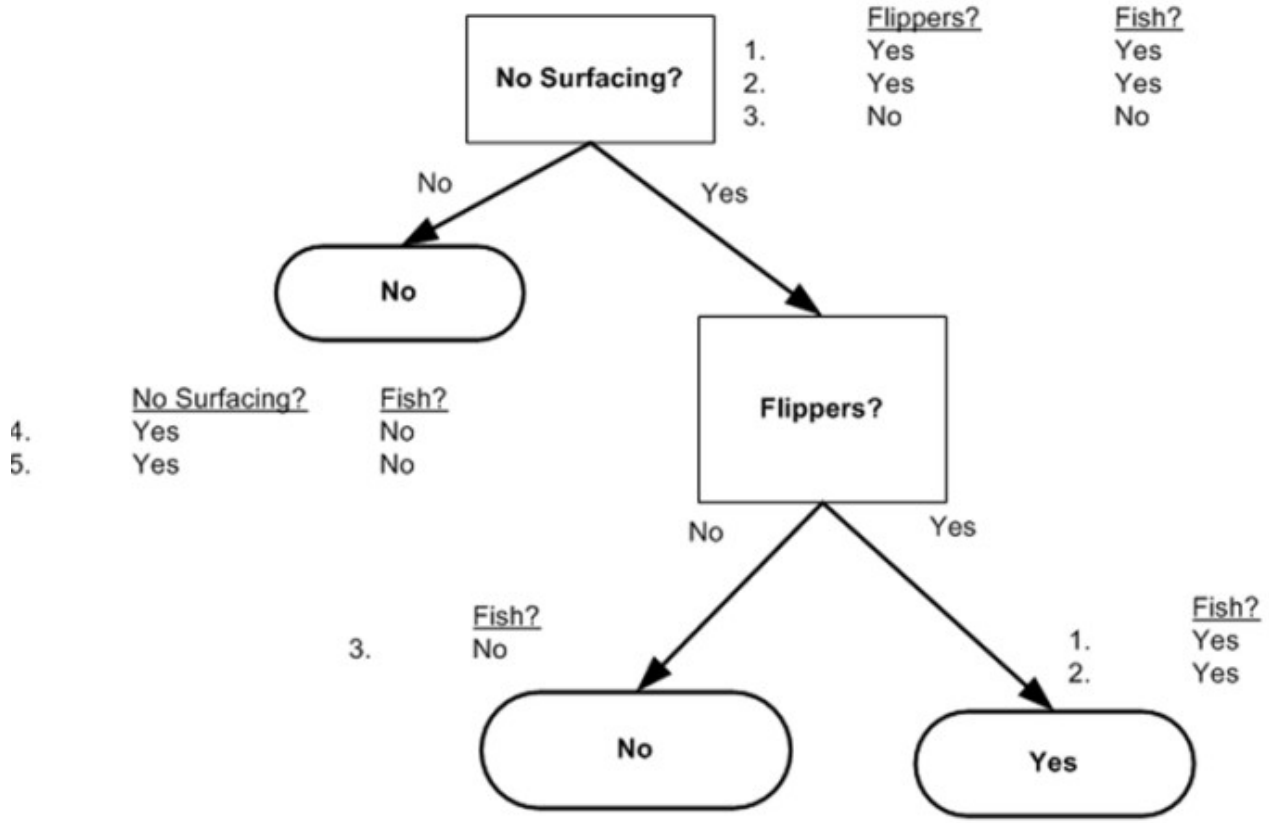
决策树的Python实现

- 创建树

```
>>> reload(trees)
<module 'trees' from 'trees.pyc'>
>>> myDat,labels=trees.createDataSet()
>>> myTree = trees.createTree(myDat,labels)
>>> myTree
{'no surfacing': {0: 'no', 1: {'flippers': {0: 'no', 1: 'yes'}}}}
```

例决策树的Python实现子

- 创建树



决策树的Python实现

- 测试算法：使用决策树执行分类

```
def classify(inputTree,featLabels,testVec):
    firstStr = inputTree.keys()[0]
    secondDict = inputTree[firstStr]
    featIndex = featLabels.index(firstStr)
    key = testVec[featIndex]
    valueOfFeat = secondDict[key]
    if isinstance(valueOfFeat, dict):
        classLabel = classify(valueOfFeat, featLabels, testVec)
    else: classLabel = valueOfFeat
    return classLabel
```

决策树的Python实现

- 测试算法：使用决策树执行分类

```
>>> myDat,labels=trees.createDataSet()
>>> labels
['no surfacing', 'flippers']
>>> myTree=treePlotter.retrieveTree (0)
>>> myTree
{'no surfacing': {0: 'no', 1: {'flippers': {0: 'no', 1: 'yes'}}}}
>>> trees.classify(myTree,labels,[1,0])
'no'
>>> trees.classify(myTree,labels,[1,1])
'yes'
```

决策树的Python实现

- 使用Pickle模块存储决策树

- *pickle*是为了序列化/反序列化一个对象的，可以把一个对象持久化存储。比如你有一个对象，想下次运行程序的时候直接用，可以直接用*pickle*打包存到硬盘上。或者你想把一个对象传给网络上的其他程序，可以用*pickle*打包，然后传过去

决策树的Python实现

- 使用Pickle模块存储决策树

```
def storeTree(inputTree,filename):
    import pickle
    fw = open(filename,'w')
    pickle.dump(inputTree,fw)
    fw.close()

def grabTree(filename):
    import pickle
    fr = open(filename)
    return pickle.load(fr)
```

决策树的Python实现

- 使用决策树预测隐形眼镜类型

- >>>import os
- >>> os.chdir('D:\MLiA_SourceCode\machinelearninginaction')
- >>> os.chdir('D:\MLiA_SourceCode\machinelearninginaction\Ch03')
- >>> fr=open('lenses.txt')
- >>> fr
- <open file 'lenses.txt', mode 'r' at 0x0000000002B41420>
- >>> lenses=[inst.strip().split('\t') for inst in fr.readlines()]
- >>> lenses

决策树的Python实现

- 使用决策树预测隐形眼镜类型

- >>>lensesLabels=['a','b','c','d']
- >>> lensesTree=trees.createTree(lenses,lensesLabels)
- >>> lensesTree

- >>>import treePlotter
- >>> treePlotter.createPlot(lensesTree)

- END