

Data Mining 数据挖掘

-- Chapter8: 分类 (基本概念) --

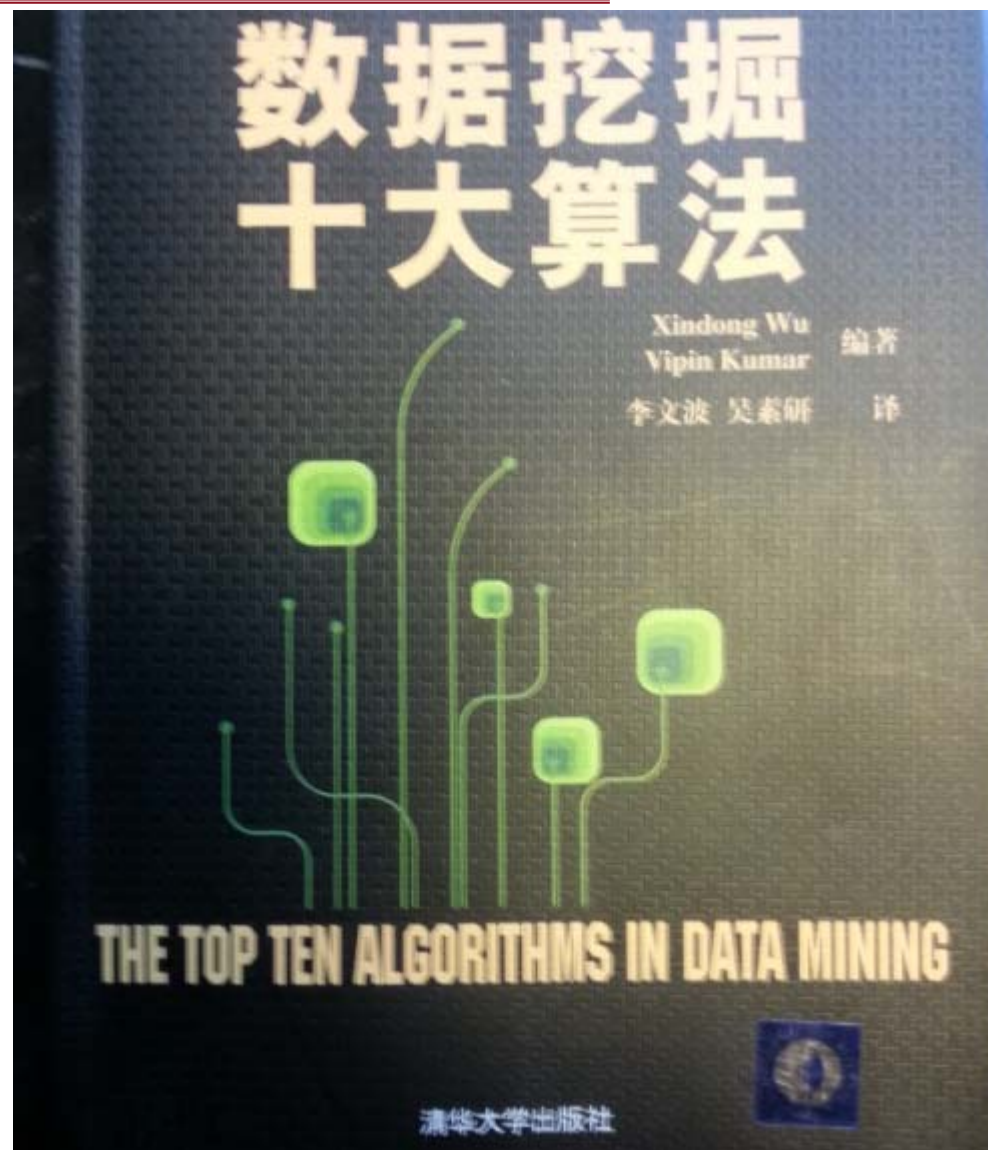
Wang Dong 王东
Shanghai Jiaotong University

2014.4



数据挖掘的“十大”最具影响力的算法

- **C4.5**
- **k-means**
- **SVM: 支持向量机**
- **Apriori**
- **EM: 期望最大化**
- **PageRank**
- **AdaBoost**
- **KNN: k-最近邻**
- **Naïve Bayes**
- **CART: 分类和回归数**





Chapter 8. Class 分类

- **Classification: Basic Concepts** 基本概念
- **Decision Tree Induction** 决策树归纳
- **Bayes Classification Methods** 贝叶斯分类
- **Rule-Based Classification** 基于规则的分类
- **Model Evaluation and Selection** 模型评估与选择
- **Techniques to Improve Classification Accuracy**
提高分类准确率的技术



Supervised vs. Unsupervised Learning

监督学习 v.s. 非监督学习

- **Supervised Learning** 监督学习（用于**分类**）**类标号已知**
 - 模型的学习在被告知每个训练样本属于 **哪个类的“指导”**下进行
 - 新数据使用 训练数据集 中得到的规则进行分类
- **Unsupervised Learning** 非监督学习（用于聚类）
 - 每个训练样本的类编号是未知的，要学习的类集合或数量也可能是事先未知的
 - 通过一系列的度量、观察来 **建立数据中的类编号** 或进行 **聚类**



Classification 分类的两种形式

● Classification 分类

- 需要一个模型或分类器，来 **预测类标号** (分类的、无序的)
- E.g.
 - AllElectronic 的销售经理需要数据分析，以便猜测具有某些特征的顾客是否会购买新计算机
 - 医学研究人员希望分析乳腺癌数据，以便预测病人应当接受三种治疗方案中的哪一种

● Numeric Prediction 数值预测

- 构造模型（预测器）预测一个 **连续值函数** 或 **有序值**
- E.g.
 - AllElectronic 的销售经理想要预测给定的客户，在一次购物期间将花费多少钱



Classification 分类的一般方法

● A Two-Step Process 二阶段过程

— 学习阶段

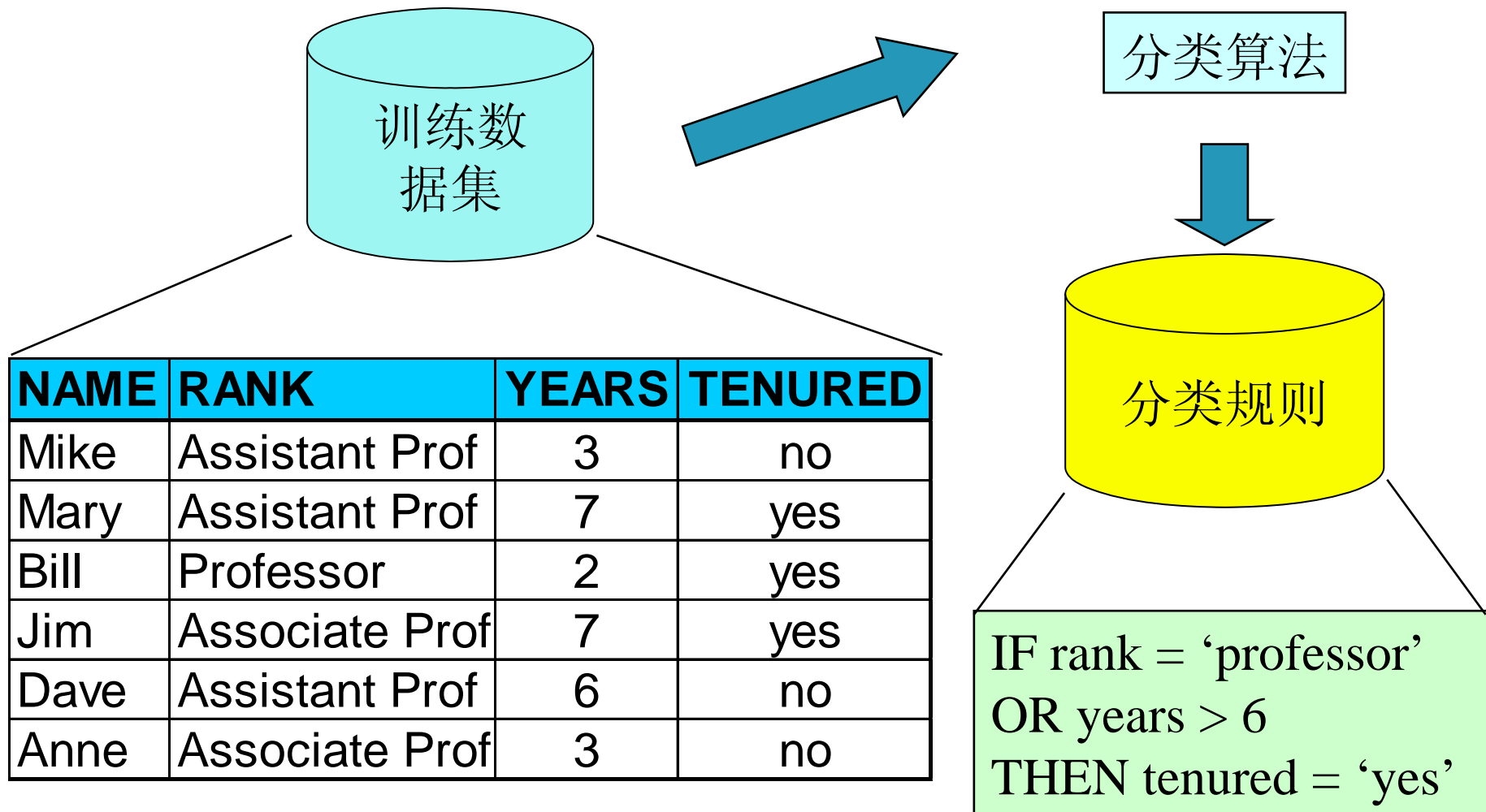
- 基于分析 **训练集** 来构建一个 **分类器模型**
 - 训练集 = 数据库元组(即 **训练元组** X) + 相关联的类标号(标称属性), 元组 $X=(x_1, x_2, \dots, x_n)$ n 维属性向量, 分别描述在属性 A_1, A_2, \dots, A_n 上的 n 个度量
 - 学习一个映射或函数 $y=f(X)$, 预测给定元组 X 的类标号 y , 一般通过 **分类规则、决策树或数学公式** 的形式提供

— 分类阶段

- 评估该模型的 **准确率** 是否可以接受, 如果可以, 使用该模型对新的数据进行分类
 - 准确率: 比较 **校验元组** 的类标号与学习模型对该元组的类预测进行比较, 分类器正确分类的校验元组所占的百分比
 - 校验元组 独立于训练元组, 否则会出现“过分适应数据”(过分拟合, **overfit**) 的情况

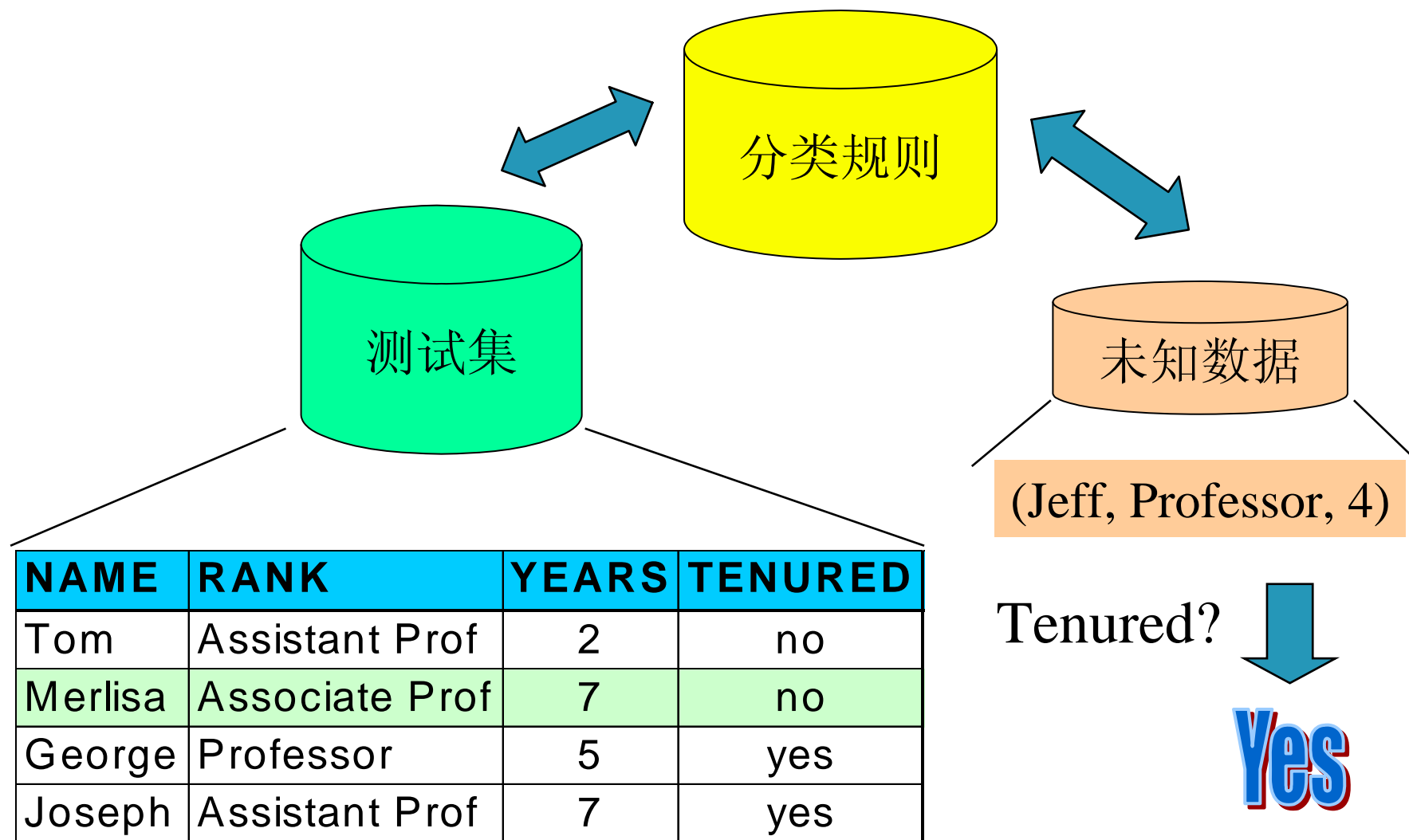


Classification 分类的一般方法 —— Step1 学习、建立分类模型





Classification 分类的一般方法 —— Step2 检验正确度、分类





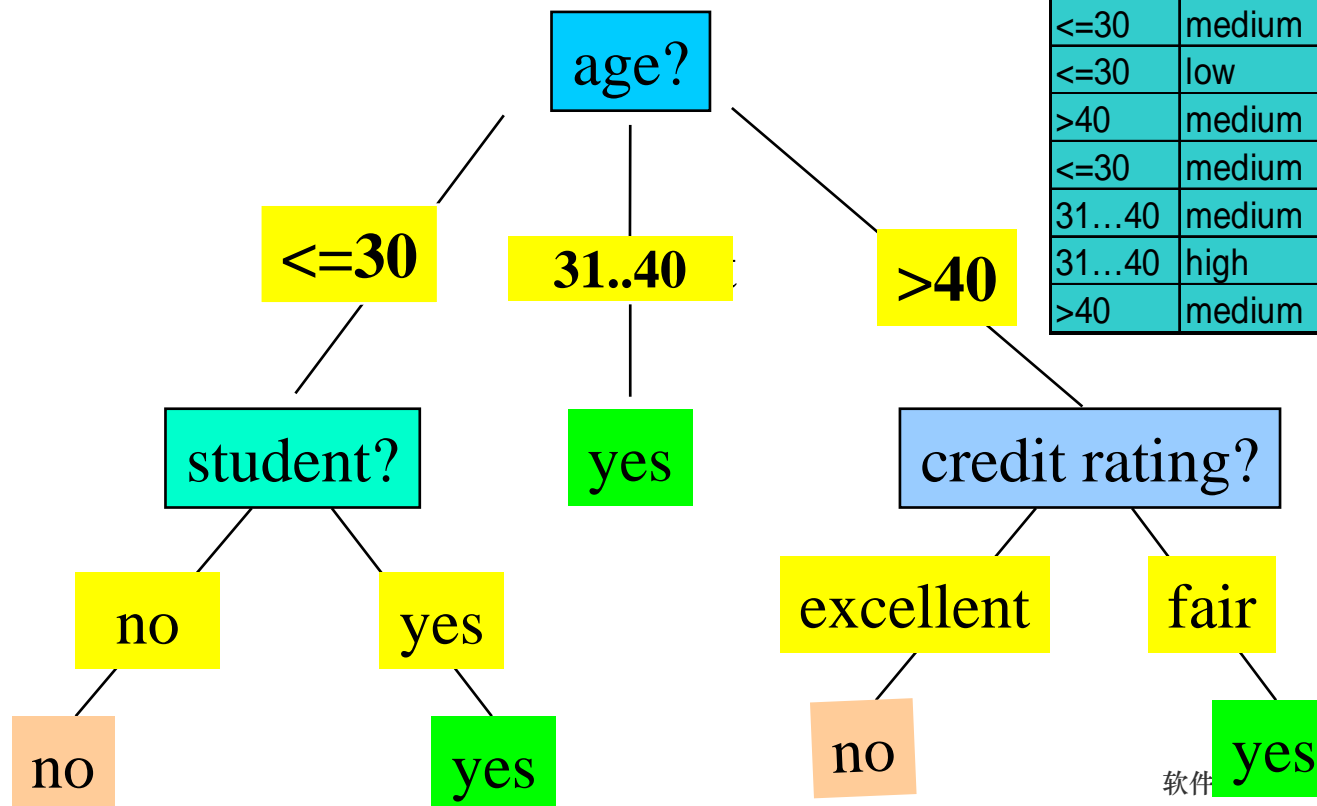
Chapter 8. Class 分类

- Classification: Basic Concepts 基本概念
- **Decision Tree Induction** 决策树归纳
- Bayes Classification Methods 贝叶斯分类
- Rule-Based Classification 基于规则的分类
- Model Evaluation and Selection 模型评估与选择
- Techniques to Improve Classification Accuracy
提高分类准确率的技术



Decision Tree 决策树

- 目的：对未知样本进行分类
— 将样本的属性值与决策树相比较
- E.g., Buys_computer的决策树:

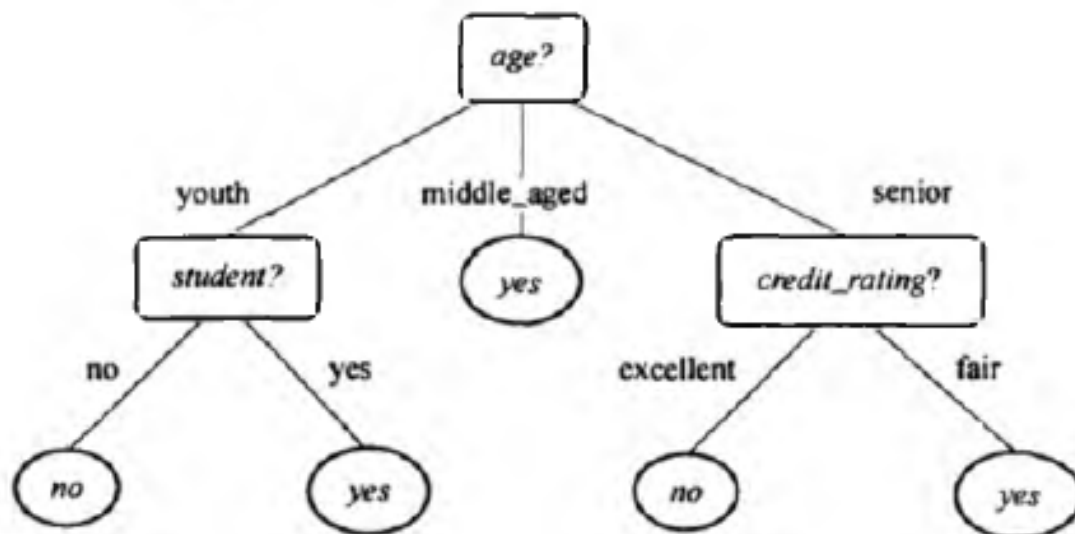


age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no



决策树归纳的概念

- 决策树归纳是从有类标号的训练元组中学习决策树
- 决策树是一种类似于流程图的树结构
 - 每个**内部结点**（非树叶结点）表示在一个属性上的测试
 - 每个**分枝**代表该测试的一个输出
 - 每个**树叶结点**（或终端结点）存放一个类标号
 - 树的最顶层结点是**根结点**
- 内部结点用矩形表示，而叶结点用椭圆表示。
- 有些决策树算法只产生二叉树（其中，每个内部结点正好分叉出两个其他结点），另一些决策树算法可能产生非二叉的树。





Algorithm for Decision Tree 决策树的算法

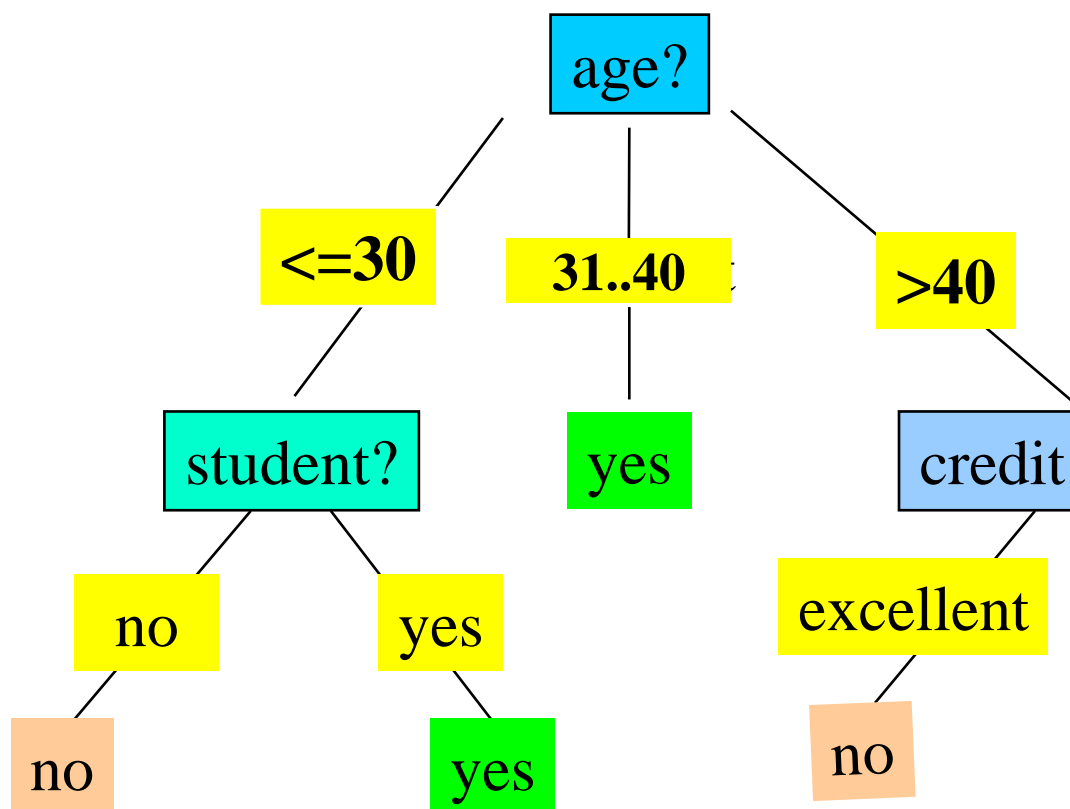
- 决策树归纳算法（一个贪心算法）
 - 自顶向下的分治方式 构造决策树
 - 树从代表训练样本的单个 根节点 开始
 - 递归地通过选择相应的 分裂属性，来划分样本，一旦一个属性出现在一个节点上，就不在该节点的任何后代上出现



决策树归纳（决策树构建）的关键问题

决策树构建过程中需要解决两个关键问题

- 如何选择合适的属性（分裂属性）作为决策树的节点去划分训练样本
- 如何在适当位置停止划分过程，从而得到大小合适的决策树



age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no

软件

yes



决策树归纳（决策树构建）的关键问题

（1）分裂属性的选择

- 虽然可以采用任何一个属性作为分裂属性对数据集进行划分，但**最后形成的决策树会差异很大**，有的是非常简化的，有的是很臃肿的
 - 对于分类算法，简洁的表示往往意味着性能要好，即对未知的实例分类效果好，需要找到合适的**分裂属性选择方法**
- 分裂属性选择度量原则
 - 根据分裂准则，把数据分区 D 划分为较小分区（最好是“**纯**”的，一个分区是纯的，如果它的所有元组都属于同一类）
 - 选择具有 **最好度量得分** 的属性作为给定元组的分裂属性
- 分裂属性选择度量方法
 - 一般是根据某种启发信息或者是统计信息来进行选择，常用属性选择度量方法
 - **信息增益**
 - **增益率**
 - **基尼指数**



决策树归纳（决策树构建）的关键问题

（2）获得合适的树

- 理论上，决策树划分可以进行到数据样本集中所有样本都属于同个类别为止，但这样得到的决策树可能层次太深，甚至每个叶节点上只有一个实例，这样的决策树叶节点由于支持度不够，即规律不具有普遍性，预测能力较弱
- 决策树归纳的目的是希望生成能够揭示数据集结构并且预测能力很强的树，在树完全生长的时候有可能预测能力反而下降，为此需要获得大小合适的树
- 一般来说，获取方法有两种
 - 定义树的停止生长条作
 - 最小划分实例数：当处理节点对应的数据集子集的大小小于指定的最小划分实例数时，即使它们不属于同一类，也不再进一步划分
 - 划分阈值：当使用的划分方法所得的值与父节点的值的差小于指定阈值时，不再进一步划分
 - 最大树深度：当进一步划分将超过最大树深度时，停止划分
 - 对完全生长的决策树进行剪枝
 - 对决策树的子树进行评估，若去掉该子树后整个决策树表现更好，则该子树将被剪枝



决策树归纳算法

'ite,reitiv dai,kote'ko

- 在20世纪70年代后期和20世纪80年代初期，机器学习研究人员J.Ross Quinlan开发了决策树算法，称为**迭代的二分器（Iterative Dichotomiser, ID3）**。这项工作扩展了E.B. Hunt, J.Marín和P.T.Stone的概念学习系统。
- Quinlan后来提出了**C4.5（ID3的后继）**，成为了新的监督学习算法的性能比较基准
- 1984年，多位统计学家（L.Breiman、J.Friedman、R.Olshen、C.Stone）出版著作《Classification and Regression Trees》（**CART**），介绍二叉决策树的产生
- **ID3和CART**大约同时独立地发明，但是从训练元组学习决策树都采用了类似的方法，**这两个基础算法引发了决策树归纳研究的旋风**



由训练元组归纳决策树的基本算法

- **ID3、C4.5、CART**都采用贪心方法，其中决策树以自顶向下递归的分治方式构造。大多数决策树归纳算法都沿用这种自顶向下方法，从训练元组集和它们相关联的类标号开始构造决策树。随着树的构建，训练集递归地划分成较小的子集。
- 基本决策树归纳算法概括在图8.3



由训练元组归纳决策树的基本算法

Algorithm: Generate_decision_tree. Generate a decision tree from the training tuples of data partition, D .

Input:

- Data partition, D , which is a set of training tuples and their associated class labels;
- *attribute_list*, the set of candidate attributes;
- *Attribute_selection_method*, a procedure to determine the splitting criterion that “best” partitions the data tuples into individual classes. This criterion consists of a *splitting_attribute* and, possibly, either a *split-point* or *splitting subset*.

Output: A decision tree.

Method:

- (1) create a node N ;
- (2) **if** tuples in D are all of the same class, C , **then**
- (3) return N as a leaf node labeled with the class C ;
- (4) **if** *attribute_list* is empty **then**
- (5) return N as a leaf node labeled with the majority class in D ; // majority voting
- (6) apply **Attribute_selection_method**(D , *attribute_list*) to **find** the “best” *splitting_criterion*;
- (7) label node N with *splitting_criterion*;
- (8) **if** *splitting_attribute* is discrete-valued **and**
 multiway splits allowed **then** // not restricted to binary trees
- (9) *attribute_list* \leftarrow *attribute_list* $-$ *splitting_attribute*; // remove *splitting_attribute*
- (10) **for each** outcome j of *splitting_criterion*
 // partition the tuples and grow subtrees for each partition
- (11) let D_j be the set of data tuples in D satisfying outcome j ; // a partition
- (12) **if** D_j is empty **then**
- (13) attach a leaf labeled with the majority class in D to node N ;
- (14) **else** attach the node returned by **Generate_decision_tree**(D_j , *attribute_list*) to node N ;
- endfor**
- (15) return N ;



由训练元组归纳决策树的基本算法

算法: Generate decision tree. 由数据分区 D 中的训练元组产生决策树。

輸入:

- 数据分区 D , 训练元组和它们对应类标号的集合。
- `attribute_list`, 候选属性的集合。
- `Attribute_selection_method`, 一个确定“最好地”划分数据元组为个体类的分裂准则的过程。这个准则由分裂属性 (`splitting attribute`) 和分裂点或划分子集组成。

- *attribute list*, 候选属性的集合。

- Attribute_selection_method, 一个确定“最好地”划分数据元组为个体类的分裂准则的过程。这个准则由分裂属性 (splitting attribute) 和分裂点或划分子集组成。

输出：一棵决策树。

方法：

- ```

(1) 创建一个结点N;
(2) if D中的元组都在同一类C中then
(3) 返回N作为叶结点,以类C标记;
(4) if attribute_list为空then
(5) 返回N作为叶结点,标记为D中的多数类; //多数表决
(6) 使用Attribute_selection_method(D, attribute_list),找出“最好的”splitting_criterion;
(7) 用splitting_criterion标记结点N;
(8) if splitting_attribute是离散值的,并且允许多路划分then //不限于二叉树
(9) attribute_list=attribute_list-splitting_attribute; // 删除分裂属性
(10) for splitting_criterion的每个输出j
 //划分元组并对每个分区产生子树
(11) 设 D_j 是D中满足输出j的数据元组的集合; // 一个分区
(12) if D_j 为空then
(13) 加一个树叶到结点N,标记为D中的多数类;
(14) else加一个由Generate_decision_tree(D_j ,attribute_list)返回的结点到N;
 endfor
(15) 返回N;

```

(2) 如果  $D$  中的元组都在同一类  $C$  中 then

(3) 返回N作为叶结点,以类C标记:

(4) **if** attribut list为空**then**

(5) 返回N作为叶结点, 标记为D中的多数类; //多数表决

(6) 使用Attribute selection method(D, attribute list),找出“最好的”splitting criterion;

(7) 用splitting criterion标记结点N:

(8) *if* splitting attribute是离散值的,并且允许多路划分*then* //不限于二叉树

(9) `attribute list-attribute list-splitting attribute; // 刪除分裂屬性`

(10) **for** *splitting criterion* 的每个输出  $j$

```
//划分元组并对每个分区产生子树
```

(11) 设  $D_j$  是  $D$  中满足输出  $j$  的数据元组的集合; // 一个分区

(12) if  $D_i$  is empty then

(13) 加一个树叶到结点  $N$ , 标记为  $D$  中的多数类;

(14) **else** 加一个由 `Generate decision tree( $D_i$ , attribute list)` 返回的结点到  $N$ ;

endfor

(15) 返回N;



# 由训练元组归纳决策树的基本算法 说明（1）

---

- 用三个参数D、attribute\_list和Attribute\_selection\_method调用算法**Generate\_decision\_tree**
  - **D**称为数据分区，初始它是训练元组和它们相应类标号的完全集
  - **attribute\_list**是描述元组属性的列表
  - **Attribute\_selection\_method**指定选择属性的启发式过程，用来选择可以按类“最好地”区分给定元组的属性（分裂属性）
    - 该过程使用一种属性选择度量，如信息增益或基尼指数
    - 树是否是严格的二叉树由属性选择度量确定
      - 某些属性选择度量，如基尼指数强制结果树是二叉树
      - 其他度量，如信息增益并非如此，它允许多路划分（即从一个结点生长两个或多个分枝）



- ```

(1) 创建一个结点N;
(2) if D中的元组都在同一类C中then
(3)     返回N作为叶结点,以类C标记;
(4) if attribut_list为空then
(5)     返回N作为叶结点,标记为D中的多数类;           //多数表决
(6) 使用Attribute selection method(D, attribute_list),找出“最好的”splitting criterion;

```



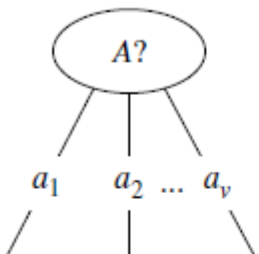

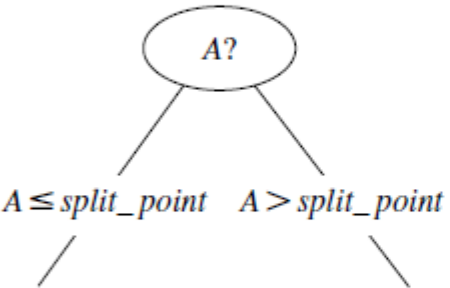
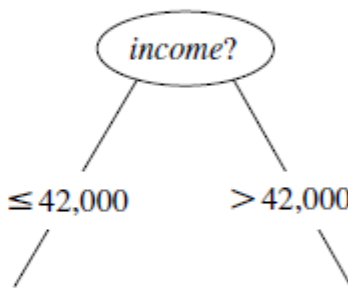
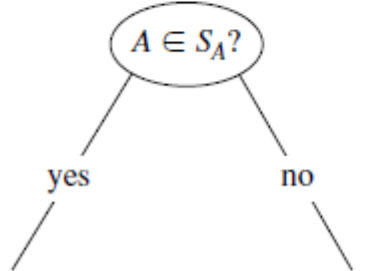
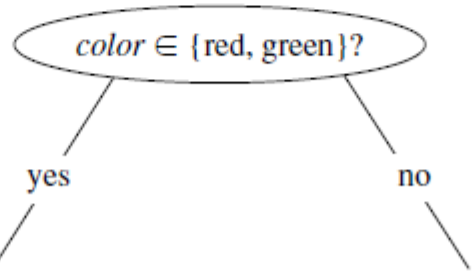
由训练元组归纳决策树的基本算法 说明（3）

- 结点N用分裂准则标记作为结点上的测试（**步骤7**）。对分裂准则的每个输出，由结点N生长一个分枝。D中的元组据此进行划分（**步骤10 ~ 11**）。有三种可能的情况，如图8.4所示。设A是分裂属性。根据训练数据，A具有v个不同值 $\{a_1, a_2, \dots, a_v\}$ 。
- A是离散值的
 - 结点N的输出直接对应于A的已知值
 - 对每个已知值 a_j 创建一个分支
 - 分区 D_j 是D中A上取值为 a_j 的类标记元组的子集
- A是连续值的
 - 结点N有两个可能的输出
 - split_point为分裂点，对应两个条件 $A \leq \text{split_point}$ $A > \text{split_point}$
- A是离散值且必须产生二叉树（由属性选择度量或所使用的算法指出）
 - 测试形如 A属于SA? 其中SA是A的分裂子集，也就是A已知值的子集



由训练元组归纳决策树的基本算法 说明（3）

根据分裂准则划分元组的三种可能性

Partitioning scenarios	Examples
(a) 	
(b) 	
(c) 	



由训练元组归纳决策树的基本算法 说明（4）

- 对于D中的每个结果分区 D_j 上的元组，算法使用同样的过程递归地形成决策树（步骤14）
- 递归划分步骤仅当下列终止条件之一成立时停止：
 - 1) 分区D（在结点N提供）的所有元组都属同一个类（步骤2 和步骤3）
 - 2) 没有剩余属性可以用来进一步划分元组（步骤4）。在此情况下，使用多数表决（步骤5）。这涉及将N转换成树叶，并用D中的多数类标记它。另外，也可以存放结点元组的类分布。
 - 3) 给定的分枝没有元组，即分区 D_j 为空（步骤12）。在这种情况下，用D中的多数类创建一个树叶（步骤13）。
- 返回结果决策树（步骤15）

```
(1) 创建一个结点N;  
(2) if D中的元组都在同一类C中 then  
(3)   返回N作为叶结点,以类C标记;  
(4) if attribute_list为空 then  
(5)   返回N作为叶结点,标记为D中的多数类; //多数表决  
(6) 使用Attribute_selection_method(D, attribute_list),找出“最好的”splitting_criterion;  
(7) 用splitting_criterion标记结点N;  
(8) if splitting_attribute是离散值的,并且允许多路划分 then //不限于二叉树  
(9)   attribute_list=attribute_list-splitting_attribute; // 删除分裂属性  
(10) for splitting_criterion的每个输出j  
      //划分元组并对每个分区产生子树  
(11)   设 $D_j$ 是D中满足输出j的数据元组的集合; // 一个分区  
(12)   if  $D_j$ 为空 then  
(13)     加一个树叶到结点N,标记为D中的多数类;  
(14)   else 加一个由Generate_decision_tree( $D_j$ , attribute_list)返回的结点到N;  
      andfor  
(15) 返回N;
```




各类决策树算法之间的差别

- 基本算法对于树的每一层，需要扫描一遍 D 中的元组。在处理大型数据库时，这可能导致很长的训练时间和内存不足

基本算法的计算复杂度

给定训练集 D ，算法的计算复杂度为 $O(n \times |D| \times \log(|D|))$ ，其中 n 是描述 D 中元组的属性个数， $|D|$ 是 D 中的训练元组数。这意味以 $|D|$ 个元组产生一棵树的计算开销最多为 $n \times |D| \times \log(|D|)$ 。

- 关于决策树归纳的改进算法很多，主要差别体现在
 - 在创建树时如何选择属性
 - 用于剪枝的机制



决策树归纳经典算法

(一) ID3



决策树归纳经典算法

ID3基本原理

- ④ 由Quinlan在1986年提出
- ④ 使用**信息增益**作为属性选择标准
- ④ 算法原理
 - 首先检测所有属性，选择**信息增益值最大属性**产生决策树节点
 - 由该属性的不同取值建立分支
 - 再对各分支的子集递归调用该方法建立决策树节点的分支
 - **直到所有子集仅包含同一个类别的数据为止**，最后得到一棵决策树，对所有的新样本进行分类



Information Gain 属性选择度量 之 信息增益

- 信息增益
 - 在树的每个节点上使用信息增益度量选择测试属性
 - 根据当前节点对应的训练样本，计算各属性的信息增益，然后选择具有**最高信息增益**的属性作为**分裂属性**来做样本划分
- 信息增益计算方式
 - 信息熵（entropy）：用于度量一个属性的信息量
 - ✓ 假定S为训练集，S的目标属性C具有m个可能的类标号值， $C=\{C_1, C_2, \dots, C_m\}$ ，假定训练集S中 C_i 在所有样本中出现的概率为 $p_i (i=1, 2, 3, \dots, m)$ ，则该训练集S所包含的信息熵定义为

$$\text{Entropy}(S) = \text{Entropy}(p_1, p_2, \dots, p_m) = - \sum_{i=1}^m p_i \log_2(p_i)$$

- ✓ **熵越小**，表示样本对目标属性的分**布越纯**。熵越大，表示样本对目标属性的分布越乱
 - 熵为0则意味着所有样本的目标属性取值相同
 - 当S中不同类别的记录数相当时，取得最大值 $\log_2 m$, m为类别的个数，对于两个类别来说，最大值为1



Information Gain 属性选择度量 之 信息增益

- 信息增益计算方式

- 信息增益

- ✓ 用属性A划分样本数据集S，划分前样本数据集的不纯程度（熵）和划分后样本数据集的不纯程度（熵）的差值

$$\text{Gain}(S, A) = \text{Entropy}(S) - \text{Entropy}_A(S)$$

- ✓ 按属性A划分S后的样本子集的熵定义：假定属性A具有K个不同的取值，将S划分为k个样本子集 $\{S_1, S_2, \dots, S_k\}$ ，则按属性A划分S后的样本子集的信息熵如下，其中 $|S_i|$ ($i=1,2,\dots,k$) 为样本子集 S_i 中包含的样本数， $|S|$ 为样本集S中包含的样本数

$$\text{Entropy}_A(S) = \sum_{i=1}^k \frac{|S_i|}{|S|} \text{Entropy}(S_i)$$

- ✓ 信息增益越大，说明使用属性A划分后的样本子集越纯，越有利于分类



Weather数据集

表 3-2 Weather

outlook	temperature	humidity	wind	play ball
sunny	hot	high	weak	no
sunny	hot	high	strong	no
overcast	hot	high	weak	yes
rain	mild	high	weak	yes
rain	cool	normal	weak	yes
rain	cool	normal	strong	no
overcast	cool	normal	strong	yes
sunny	mild	high	weak	no
sunny	cool	normal	weak	yes
rain	mild	normal	weak	yes
sunny	mild	normal	strong	yes
overcast	mild	high	strong	yes
overcast	hot	normal	weak	yes
rain	mild	high	strong	no



信息熵 (entropy) 的计算

$$\text{Entropy}(S) = \text{Entropy}(p_1, p_2, \dots, p_m) = - \sum_{i=1}^m p_i \log_2(p_i)$$

$$\text{Entropy}(S) = \text{Entropy}\left(\frac{9}{14}, \frac{5}{14}\right) = -\frac{9}{14} \log_2 \frac{9}{14} - \frac{5}{14} \log_2 \frac{5}{14} = 0.94$$



信息增益的计算

$$\text{Gain}(S, A) = \text{Entropy}(S) - \text{Entropy}_A(S) \quad \text{Entropy}_A(S) = \sum_{i=1}^k \frac{|S_i|}{|S|} \text{Entropy}(S_i)$$

- 计算以属性wind来划分数数据集S，求S对属性wind的信息增益
- 属性wind有2个可能的取值{weak,strong}，将S划分为2个子集{S₁,S₂}
 - S₁是wind属性取值为weak的样本子集，共有8个样本
 - S₂是wind属性取值为strong的样本子集，共有6个样本
- 样本子集S₁的熵计算
 - Play ball=yes的有6个样本， Play ball=no的有2个样本

$$\text{Entropy}(S_1) = -\frac{6}{8} \log_2 \frac{6}{8} - \frac{2}{8} \log_2 \frac{2}{8} = 0.811$$

- 样本子集S₂的熵计算
 - Play ball=yes的有3个样本， Play ball=no的有3个样本

$$\text{Entropy}(S_2) = -\frac{3}{6} \log_2 \frac{3}{6} - \frac{3}{6} \log_2 \frac{3}{6} = 1$$



信息增益的计算

$$\text{Gain}(S, A) = \text{Entropy}(S) - \text{Entropy}_A(S) \quad \text{Entropy}_A(S) = \sum_{i=1}^k \frac{|S_i|}{|S|} \text{Entropy}(S_i)$$

- 样本子集 S_1 的熵计算

- Play ball=yes的有6个样本, Play ball=no的有2个样本

$$\text{Entropy}(S_1) = -\frac{6}{8} \log_2 \frac{6}{8} - \frac{2}{8} \log_2 \frac{2}{8} = 0.811$$

- 样本子集 S_2 的熵计算

- Play ball=yes的有3个样本, Play ball=no的有3个样本

$$\text{Entropy}(S_2) = -\frac{3}{6} \log_2 \frac{3}{6} - \frac{3}{6} \log_2 \frac{3}{6} = 1$$

- 利用属性wind划分S后的熵和信息增益

$$\begin{aligned} \text{Entropy}_{\text{wind}}(S) &= \sum_{i=1}^k \frac{|S_i|}{|S|} \text{Entropy}(S_i) = \frac{|S_1|}{|S|} \text{Entropy}(S_1) + \frac{|S_2|}{|S|} \text{Entropy}(S_2) \\ &= \frac{8}{14} \text{Entropy}(S_1) + \frac{6}{14} \text{Entropy}(S_2) = 0.571 \times 0.811 + 0.428 \times 1 = 0.891 \end{aligned}$$

$$\text{Gain}(S, \text{wind}) = \text{Entropy}(S) - \text{Entropy}_{\text{wind}}(S) = 0.94 - 0.891 = 0.049$$



ID3构建决策树的算法过程

- 数据集weather具有属性{outlook,temperature,humidity,wind}

— 属性outlook={sunny,overcast,rain}

— 属性temperature={hot,mid,cool}

— 属性humidity={high,normal}

— 属性wind={weak,strong}

第一步：寻找信息增益最大的属性

$$\begin{aligned}\text{Entropy}_{\text{outlook}}(S) &= \sum_{i=1}^k \frac{|S_i|}{|S|} \text{Entropy}(S_i) = \frac{|S_1|}{|S|} \text{Entropy}(S_1) + \frac{|S_2|}{|S|} \text{Entropy}(S_2) + \frac{|S_3|}{|S|} \text{Entropy}(S_3) \\ &= \frac{5}{14} \left(-\frac{2}{5} \log_2 \frac{2}{5} - \frac{3}{5} \log_2 \frac{3}{5} \right) + \frac{4}{14} \left(-\frac{4}{4} \log_2 \frac{4}{4} - \frac{0}{4} \log_2 \frac{0}{4} \right) + \frac{5}{14} \left(-\frac{3}{5} \log_2 \frac{3}{5} - \frac{2}{5} \log_2 \frac{2}{5} \right) \\ &= 0.694\end{aligned}$$

$$\text{Gain}(S, \text{outlook}) = \text{Entropy}(S) - \text{Entropy}_{\text{outlook}}(S) = 0.94 - 0.694 = 0.246$$

$$\begin{aligned}\text{Entropy}_{\text{temperature}}(S) &= \sum_{i=1}^k \frac{|S_i|}{|S|} \text{Entropy}(S_i) = \frac{|S_1|}{|S|} \text{Entropy}(S_1) + \frac{|S_2|}{|S|} \text{Entropy}(S_2) + \frac{|S_3|}{|S|} \text{Entropy}(S_3) \\ &= \frac{4}{14} \left(-\frac{2}{4} \log_2 \frac{2}{4} - 2 \log_2 \frac{2}{4} \right) + \frac{6}{14} \left(-\frac{4}{6} \log_2 \frac{4}{6} - \frac{2}{6} \log_2 \frac{2}{6} \right) + \frac{4}{14} \left(-\frac{3}{4} \log_2 \frac{3}{4} - \frac{1}{4} \log_2 \frac{1}{4} \right) \\ &= 0.911\end{aligned}$$

$$\text{Gain}(S, \text{temperature}) = \text{Entropy}(S) - \text{Entropy}_{\text{temperature}}(S) = 0.94 - 0.911 = 0.029$$

$$\begin{aligned}\text{Entropy}_{\text{humidity}}(S) &= \sum_{i=1}^k \frac{|S_i|}{|S|} \text{Entropy}(S_i) = \frac{|S_1|}{|S|} \text{Entropy}(S_1) + \frac{|S_2|}{|S|} \text{Entropy}(S_2) \\ &= \frac{7}{14} \left(-\frac{3}{7} \log_2 \frac{3}{7} - \frac{4}{7} \log_2 \frac{4}{7} \right) + \frac{7}{14} \left(-\frac{6}{7} \log_2 \frac{6}{7} - \frac{1}{7} \log_2 \frac{1}{7} \right) \\ &= 0.788\end{aligned}$$

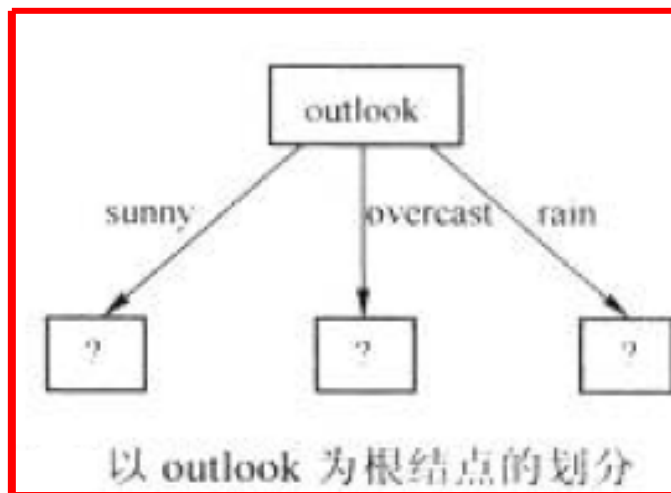
$$\text{Gain}(S, \text{humidity}) = \text{Entropy}(S) - \text{Entropy}_{\text{humidity}}(S) = 0.94 - 0.788 = 0.152$$

$$\text{Gain}(S, \text{wind}) = 0.049$$



ID3构建决策树的算法过程

第二步：以信息增益最大的属性**outlook**为分裂属性，建立根节点的**3**个分支





ID3构建决策树的算法过程

第三步：计算确定最佳划分根节点3个分支的决策属性，即哪些属性分别用来最佳划分根节点的sunny、overcast、rain分支

- 首先对outlook的sunny分支建立子树
 - 找出样本数据集weather中outlook取值sunny的样本子集 S_{sunny}
 - 依次计算剩下三个属性对该样本子集 S_{sunny} 划分后的信息增益
 - 建立sunny分支子树

weather 数据集中 outlook 取值为 sunny 的数据子集 S_{sunny}

outlook	temperature	humidity	wind	play ball
sunny	hot	high	weak	no
sunny	hot	high	strong	no
sunny	mild	high	weak	no
sunny	cool	normal	weak	yes
sunny	mild	normal	strong	yes



ID3构建决策树的算法过程

- 首先对outlook的sunny分支建立子树
 - 找出样本数据集weather中outlook取值sunny的样本子集 S_{sunny}
 - 依次计算剩下三个属性对该样本子集 S_{sunny} 划分后的信息增益
 - 建立sunny分支子树

子集 S_{sunny} 的熵为

$$\text{Entropy}(S_{\text{sunny}}) = \text{Entropy}\left(\frac{3}{5}, \frac{2}{5}\right) = -\frac{3}{5} \log \frac{3}{5} - \frac{2}{5} \log \frac{2}{5} = 0.971$$

① 属性 humidity 在数据子集 S_{sunny} 中有 2 个取值 {high, normal}, 且不同取值下类标号取值相同, 即

$$\begin{aligned} \text{Entropy}_{\text{humidity}}(S_{\text{sunny}}) &= \sum_{i=1}^k \frac{|S_i|}{|S_{\text{sunny}}|} \text{Entropy}(S_i) = \frac{|S_1|}{|S_{\text{sunny}}|} \text{Entropy}(S_1) + \frac{|S_2|}{|S_{\text{sunny}}|} \text{Entropy}(S_2) \\ &= \frac{3}{5} * 0 + \frac{2}{5} * 0 = 0 \end{aligned}$$

因此, 属性 humidity 划分子集 S_{sunny} 的信息增益值为

$$\begin{aligned} \text{Gain}(S_{\text{sunny}}, \text{humidity}) &= \text{Entropy}(S_{\text{sunny}}) - \text{Entropy}_{\text{humidity}}(S_{\text{sunny}}) \\ &= 0.971 - 0 = 0.971 \end{aligned}$$



ID3构建决策树的算法过程

- 首先对outlook的sunny分支建立子树

- 找出样本数据集weather中outlook取值sunny的样本子集 S_{sunny}
- 依次计算剩下三个属性对该样本子集 S_{sunny} 划分后的信息增益
- 建立sunny分支子树

② 属性 temperature 在数据子集 S_{sunny} 中有 3 个取值 {hot, mild, cool}, 其熵值为

$$\begin{aligned}\text{Entropy}_{\text{temperature}}(S_{\text{sunny}}) &= \sum_{i=1}^k \frac{|S_i|}{|S_{\text{sunny}}|} \text{Entropy}(S_i) \\ &= \frac{|S_1|}{|S_{\text{sunny}}|} \text{Entropy}(S_1) + \frac{|S_2|}{|S_{\text{sunny}}|} \text{Entropy}(S_2) + \frac{|S_3|}{|S_{\text{sunny}}|} \text{Entropy}(S_3) \\ &= \frac{2}{5} * 0 + \frac{2}{5} * \left(-\frac{1}{2} * \log \frac{1}{2} - \frac{1}{2} * \log \frac{1}{2} \right) + \frac{1}{5} * 0 = 0.4\end{aligned}$$

因此, 属性 temperature 划分子集 S_{sunny} 的信息增益值为

$$\begin{aligned}\text{Gain}(S_{\text{sunny}}, \text{temperature}) &= \text{Entropy}(S_{\text{sunny}}) - \text{Entropy}_{\text{temperature}}(S_{\text{sunny}}) \\ &= 0.971 - 0.4 = 0.571\end{aligned}$$

③ 属性 wind 在数据子集 S_{sunny} 中有 2 个取值 {weak, strong}, 其熵值为

$$\begin{aligned}\text{Entropy}_{\text{wind}}(S_{\text{sunny}}) &= \sum_{i=1}^k \frac{|S_i|}{|S_{\text{sunny}}|} \text{Entropy}(S_i) = \frac{|S_1|}{|S_{\text{sunny}}|} \text{Entropy}(S_1) + \frac{|S_2|}{|S_{\text{sunny}}|} \text{Entropy}(S_2) \\ &= \frac{3}{5} * \left(-\frac{1}{3} * \log \frac{1}{3} - \frac{2}{3} * \log \frac{2}{3} \right) + \frac{2}{5} * \left(-\frac{1}{2} * \log \frac{1}{2} - \frac{1}{2} * \log \frac{1}{2} \right) = 0.6\end{aligned}$$

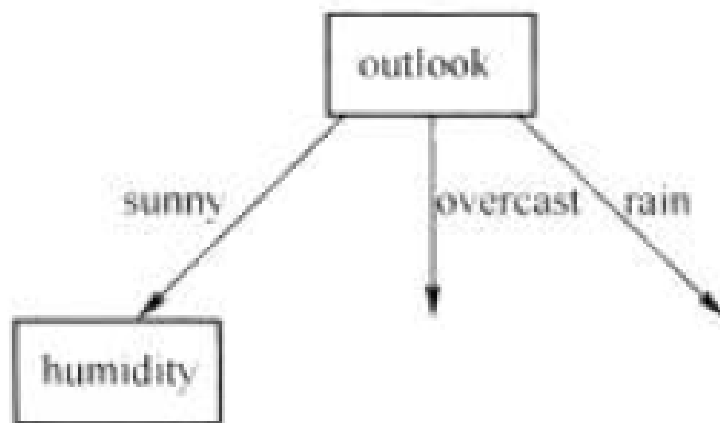
因此, 属性 wind 划分子集 S_{sunny} 的信息增益值为

$$\begin{aligned}\text{Gain}(S_{\text{sunny}}, \text{wind}) &= \text{Entropy}(S_{\text{sunny}}) - \text{Entropy}_{\text{wind}}(S_{\text{sunny}}) \\ &= 0.971 - 0.6 = 0.371\end{aligned}$$



ID3构建决策树的算法过程

- 首先对outlook的sunny分支建立子树
 - 找出样本数据集weather中outlook取值sunny的样本子集 S_{sunny}
 - 依次计算剩下三个属性对该样本子集 S_{sunny} 划分后的信息增益
 - 建立sunny分支子树



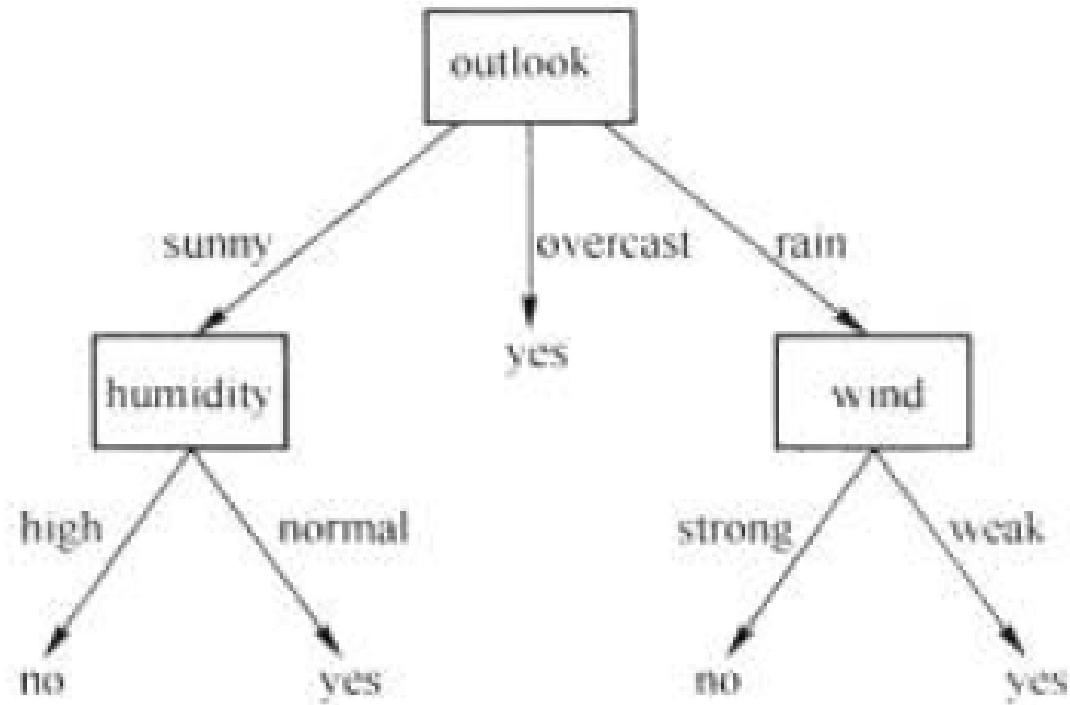
sunny 分支的划分

- 然后以同样的方法依次对outlook的overcast分支和rain分支建立子树



ID3构建决策树的算法过程

第四步：形成决策树



第五步：利用决策树预测未知样本的类标号**play ball**上所属分类

$X=\{\text{rain, hot, normal, weak, ?}\}$



ID3算法分析

- 优点

- 理论清晰，方法简单，学习能力较强

- 缺点

- 只能处理**分类属性数据**，无法处理**连续型数据** discrete data

- 对测试属性的每个取值相应产生一个分支，且划分相应的数据样本集，这样的划分会导致产生很多小的子集。随着子集被划分的越来越小，**划分过程将会由于子集规模过小造成的统计特征不充分而停止**

- 使用信息增益作为决策树节点属性选择的标准。由于**信息增益在类别值多的属性上计算结果大于类别值少的属性上计算结果**，这将导致决策树算法**偏向选择具有较多分支的属性，因而会造成过度拟合**

- 考虑充当唯一标识符的属性，如`product_ID`，在该属性上划分将导致大量分区（与值一样多），每个只包含一个元组。由于每个分区都是纯的，所以基于该划分对数据集S分类的信息熵为0，通过对该属性的划分得到的信息增益最大。但显然，这种划分对分类没有用。



决策树归纳经典算法

(一) ID3 (教材中的描述)



Information Gain 属性选择度量 之 信息增益(Cont.)

● 信息增益 E.g., (类标号属性buys_computer)

— Class P: buys_computer = “yes”; Class N: buys_computer = “no”

— 步骤一：计算对D中元组分类所需要的期望信息：

$$Info(D) = I(9,5) = -\frac{9}{14} \log_2\left(\frac{9}{14}\right) - \frac{5}{14} \log_2\left(\frac{5}{14}\right) = 0.940$$

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no



Information Gain 属性选择度量 之 信息增益(Cont.)

● 信息增益 E.g., (类标号属性buys_computer)

— 步骤二，计算每个属性的期望信息需求（以Age为例）

age	p_i	n_i	$I(p_i, n_i)$
≤ 30	2	3	0.971
31...40	4	0	0
> 40	3	2	0.971

$$\frac{5}{14} I(2,3)$$

即 14 个 sample 中，共 5 个“age ≤ 30 ”
其中有 2 个 yes 元组和 3 个 no 元组

$$Info_{age}(D) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2) = 0.694$$

— 因此，

$$Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

— 类似的有， $Gain(income) = 0.029$

$$Gain(student) = 0.151$$

$$Gain(credit_rating) = 0.048$$

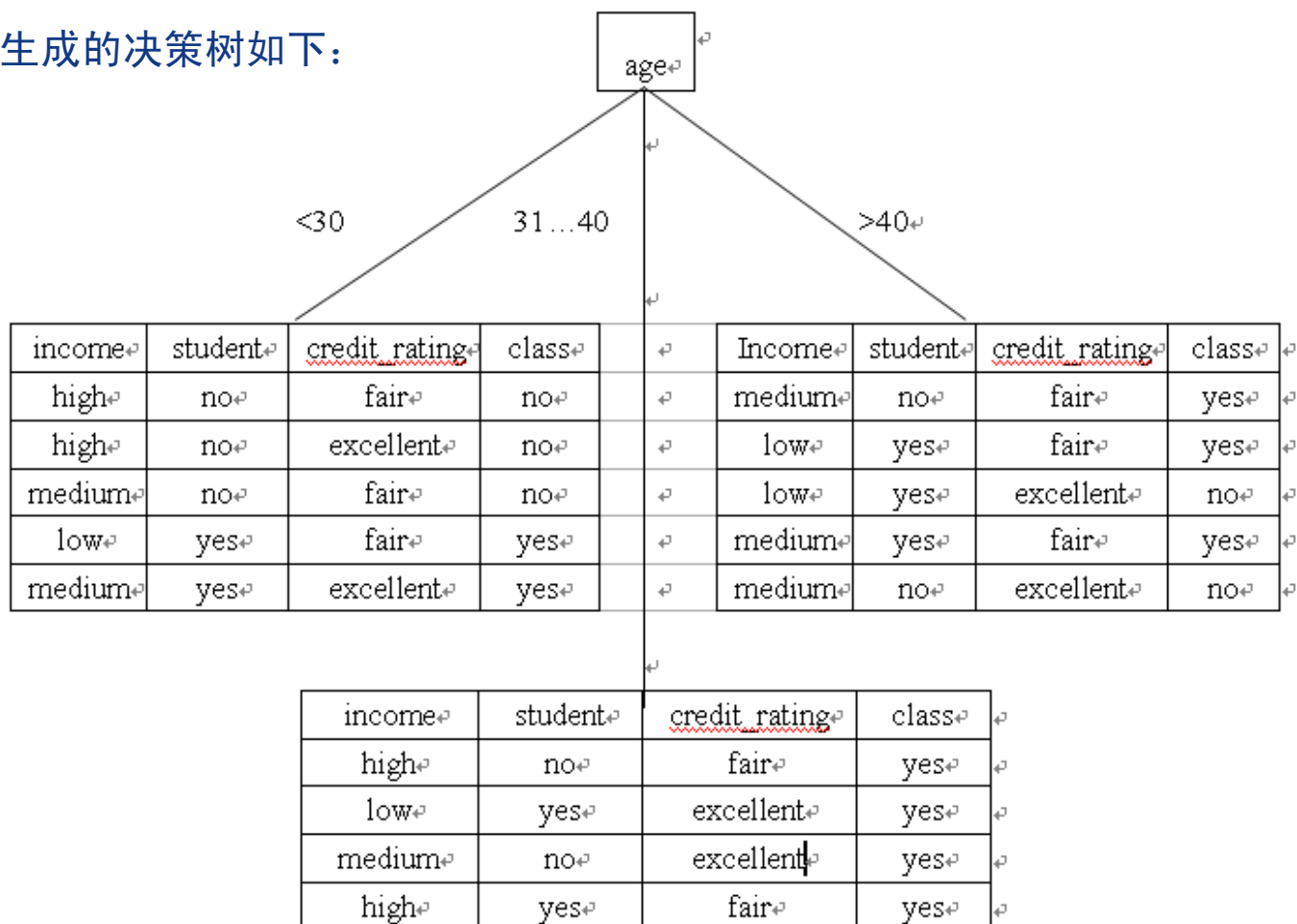
* 注: $I(a,b) = -\frac{a}{a+b} \log_2(\frac{a}{a+b}) - \frac{b}{a+b} \log_2(\frac{b}{a+b})$



Information Gain 属性选择度量 之 信息增益(Cont.)

● 信息增益 E.g., (类标号属性buys_computer)

- 步骤三：判断分裂属性并生成决策树
- 由于 Age 在属性中具有最高的信息增益，所以选它为分裂属性
- 最终生成的决策树如下：





如何计算连续属性值的信息增益？

- ④ 连续值属性A
- ④ 首先，将A的值按递增序排序
 - 每对相邻值的中点被看做可能的分裂点
 - 给定A的v个值，需要计算v-1个可能的划分
 - A的值 a_i 和 a_{i+1} 之间的中点为 $(a_i + a_{i+1})/2$
- ④ 对每个可能的分裂点计算 $\text{InfoA}(D)$

算出阈值，分裂为两个，



决策树归纳经典算法

(二) C4.5



C4.5分类算法概述

- 基于ID3算法的不足，Quinlan在1993年对其做出改进，提出了改进的决策树分类算法C4.5
 - 能够处理连续型属性数据和离散型属性数据
 - 能够处理具有缺失值的数据
 - 使用信息增益率作为决策树的属性选择标准
 - 对生成的树进行剪枝处理，以获取简略的决策树
 - 从决策树到规则的自动产生
- C4.5后续发展为商用的C5.0，在C4.5的基础上做了进一步改进
 - 使用提升技术，生成一系列决策树，然后集体投票决定分类结果
 - 支持新的数据类型，入日期等
 - 规则集没有顺序，而是所有匹配规则进行投票，更易解释



C4.5分类算法概念描述

— 信息熵 (entropy)：用于度量一个属性的信息量

- ✓ 假定 S 为训练集， S 的目标属性 C 具有 m 个可能的类标号值， $C=\{C_1, C_2, \dots, C_m\}$ ，假定训练集 S 中 C_i 在所有样本中出现的概率为 $p_i (i=1, 2, 3, \dots, m)$ ，则该训练集 S 所包含的信息熵定义为

$$\text{Entropy}(S) = \text{Entropy}(p_1, p_2, \dots, p_m) = - \sum_{i=1}^m p_i \log_2(p_i)$$

— 按属性 A 划分 S 后的样本子集的熵定义：

- ✓ 假定属性 A 为离散型数据，具有 K 个不同的取值，将 S 划分为 k 个样本子集 $\{S_1, S_2, \dots, S_k\}$ ，则按属性 A 划分 S 后的样本子集的信息熵如下，其中 $|S_i|$ ($i=1, 2, \dots, k$) 为样本子集 S_i 中包含的样本数， $|S|$ 为样本集 S 中包含的样本数

$$\text{Entropy}_A(S) = \sum_{i=1}^k \frac{|S_i|}{|S|} \text{Entropy}(S_i)$$

- ✓ 假定属性 A 为连续型数据，则按属性 A 的取值递增排序，将每对相邻值的中点看做可能的分裂点，对每个可能的分裂点，计算信息熵。 S_L 和 S_R 对应于分裂点划分的左右两部分子集，选择信息熵最小点作为分裂点

$$\text{Entropy}_A(S) = \frac{|S_L|}{|S|} \text{Entropy}(S_L) + \frac{|S_R|}{|S|} \text{Entropy}(S_R)$$



C4.5分类算法概念描述

— 信息增益率

- ✓ 不仅考虑信息增益的大小程度，还兼顾考虑获得信息增益所付出的“代价”
- ✓ 引入属性的分裂信息来调节信息增益，如果某个属性有较多的分类取值，则它的信息熵会偏大，但信息增益率由于考虑了分裂信息而降低，进而消除了属性取值数目所带来的影响

$$\text{SplitE}(A) = - \sum_{i=1}^k \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

$$\text{GainRatio}(A) = \frac{\text{Gain}(A)}{\text{SplitE}(A)}$$

$$\text{Gain}(S, A) = \text{Entropy}(S) - \text{Entropy}_A(S)$$



C4.5分类算法 对缺失数据的处理

- ID3算法不允许训练集和测试集存在缺失数据
 - ✓ 决策树中节点测试输出决定于单个属性的不同的取值，当训练集和测试集中某个样本数据的测试属性值未知，就无法得到当前节点的测试输出
- 一般处理方法
 - ✓ 抛弃数据集中具有缺失值的数据
 - 当只有少量数据具有缺失值，可以抛弃，不适合有大量数据具有缺失值的情况
 - ✓ 以某种方式填充缺失的数据
 - 以该属性中最常见值或平均值替代该缺失值
 - 以与缺失值样本所对应的类标号属性值相同的样本中该缺失值属性的最常见值或平均值替代
- C4.5处理方法
 - 采用概率的方法，为缺失值的每个可能值赋予一个概率

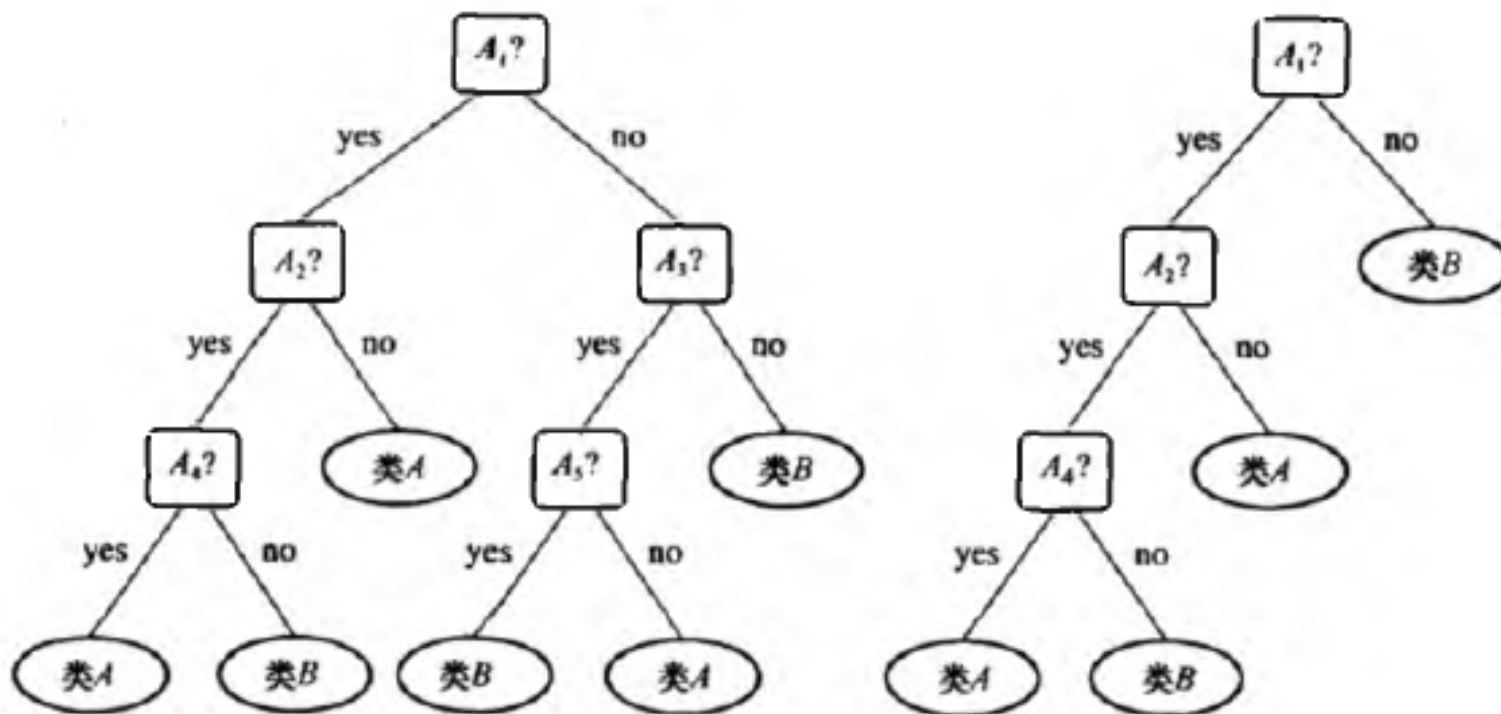
$\text{Gain}(A) = \text{属性 } A \text{ 在样本集中不空值的比率} \times (\text{Entropy}(S) - \text{Entropy}_A(S))$

$$\text{SplitE}(A) = - \sum_{i=1}^k \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|} - \frac{|S_{\text{unknown}}|}{|S|} \log_2 \frac{|S_{\text{unknown}}|}{|S|}$$



C4.5分类算法 对决策树的剪枝处理

- 在决策树创建时，由于数据中的噪声和离群点，许多分枝反映的是训练数据中的异常。利用剪枝方法处理这种过分拟合数据问题，通常，使用统计度量剪掉最不可靠的分枝。
- 一棵未剪枝树和它剪枝后的版本显示在图8.6中。剪枝后的树更小、更简单，因此更容易理解。通常，它们在正确地对独立的检验集分类时比未剪枝的树更快、更好。





C4.5分类算法

对决策树的剪枝处理

— 先剪枝 (prepruning)

- ✓ 通过提前停止树的构建（例如，通过决定在给定的结点不再分裂或划分训练元组的子集）而对树“剪枝”。一旦停止，结点就成为树叶。该树叶可以持有子集元组中最频繁的类，或这些元组的概率分布。
- ✓ 在构造树时，可以使用诸如统计显著性、信息增益、基尼指数等度量来评估划分的优劣。如果划分一个结点的元组导致低于预定义阈值的划分，则给定子集的进一步划分将停止。然而，选取一个适当的阈值是困难的。高阈值可能导致过分简化的树，而低阈值可能使得树的简化太少。

— 后剪枝

- 由“完全生长”的树剪去子树。通过删除结点的分枝并用树叶替换它而剪掉给定结点上的子树。该树叶的类标号用子树中最频繁的类标记。

— C4.5采用后剪枝

- ✓ **CART**使用的代价复杂度剪枝算法是后剪枝方法的一个实例。该方法把树的复杂度看做树中树叶结点的个数和树的错误率的函数（其中，错误率是树误分类的元组所占的百分比）。它从树的底部开始，对于每个内部结点 N ，计算 N 的子树的代价复杂度和该子树剪枝后 N 的子树（即用一个树叶结点替换）的代价复杂度。比较这两个值。如果剪去结点 N 的子树导致较小的代价复杂度，则剪掉该子树；否则，保留该子树。
- ✓ 使用一个标记类元组的剪枝集来评估代价复杂度。该集合独立于用于建立未剪枝树的训练集和用于准确率评估的检验集。算法产生一个渐进的剪枝树的集合。一般而言，最小化代价复杂度的最小决策树是首选。
- ✓ **C4.5**使用一种称为悲观剪枝的方法，它类似于代价复杂度方法，因为它也使用错误率评估，对子树剪枝做出决定。然而，悲观剪枝不需要使用剪枝集，而是使用训练集估计错误率。注意，基于训练集评估准确率或错误率过于乐观，因此具有较大的偏倚。因此，悲观剪枝方法通过加上一个惩罚来调节从训练集得到的错误率，以抵消所出现的偏倚。



C4.5分类算法基本描述

- <1> 对数据集进行预处理，对连续型属性按照基于信息熵的方法找到数据的最佳分裂点。
- <2> 计算每个属性的信息增益率，选取信息增益率最大的属性作为决策节点的划分属性。
- <3> 对决策节点属性的每个可能取值所对应的样本子集递归地执行步骤<2>，直到划分的每个子集中的观测数据都属于同一个类标号，最终生成决策树。
- <4> 对完全生长的决策树进行剪枝，得到优化后的决策树。
- <5> 从剪枝后的决策树中提取分类规则，对新的数据集进行分类。



C4.5分类算法

决策树生长阶段算法伪代码

函数名: CDT(S, F)

输入: 训练集数据 S , 训练集数据属性集合 F

输出: 一棵未剪枝的 C4.5 决策树

(1) if 样本 S 全部属于同一个类别 C then

(2) 创建一个叶节点, 并标记类标号为 C ;

(3) return;

(4) else

(5) 计算属性集 F 中每个属性的信息增益率, 假定增益率值最大的属性为 A ;

(6) 创建节点, 取属性 A 为该节点的决策属性;

(7) for 节点属性 A 的每个可能的取值 V do

(8) 为该节点添加一个新的分支, 假设 S_V 为属性 A 取值为 V 的样本子集;

(9) if 样本 S_V 全部属于同一个类别 C then

(10) 为该分支添加一个叶节点, 并标记为类标号为 C ;

(11) else

(12) 递归调用 CDT($S_V, F - \{A\}$), 为该分支创建子树;

(13) end if

(14) end for

(15) end if



C4.5分类算法

决策树剪枝阶段算法伪代码

函数名: Prune(node)

输入: 待剪枝子树 node

输出: 剪枝后的子树

(1) 计算待剪子树 node 中叶节点的加权估计误差 leafError;

(2) if 待剪子树 node 是一个叶节点 then

(3) return 叶节点误差;

(4) else

(5) 计算 node 的子树误差 subtreeError;

(6) 计算 node 的分支误差 branchError 为该节点中频率最大一个分支误差

(7) if leafError 小于 branchError 和 subtreeError then

(8) 剪枝, 设置该节点为叶节点;

(9) error=leafError;

(10) else if branchError 小于 leafError 和 subtreeError then *

(11) 剪枝, 以该节点中频率最大那个分支替换该节点;

(12) error=branchError;

(13) else

(14) 不剪枝

(15) error=subtreeError;

(16) return error;

(17) end if

(18) end if



C4.5分类算法 算例

- 数据集weather具有属性
 $\{\text{outlook}, \text{temperature}, \text{humidity}, \text{wind}\}$
 - 属性outlook={sunny, overcast, rain}
 - 属性temperature={hot, mid, cool}
 - 属性humidity={high, normal}
 - 属性wind={weak, strong}

第一步：计算所有属性划分数据集**S**所得的信息增益

$$\text{Gain}(S, \text{outlook}) = 0.246$$

$$\text{Gain}(S, \text{temperature}) = 0.029$$

$$\text{Gain}(S, \text{humidity}) = 0.152$$

$$\text{Gain}(S, \text{wind}) = 0.049$$



C4.5分类算法 算例

第二步：计算所有属性的分裂信息和信息增益率

对 outlook 属性，取值为 overcast 的样本有 4 条，取值为 rain 的样本有 5 条，取值为 sunny 的样本有 5 条，则

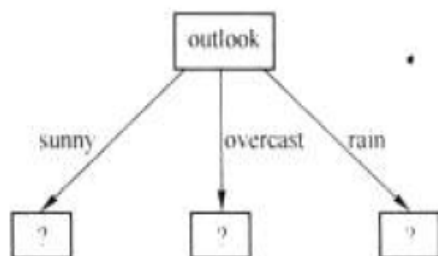
$$\text{SplitE}_{\text{outlook}} = -\frac{5}{14} \log_2 \frac{5}{14} - \frac{5}{14} \log_2 \frac{5}{14} - \frac{4}{14} \log_2 \frac{4}{14} = 1.576$$

$$\text{GainRatio}_{\text{outlook}} = \frac{\text{Gain}_{\text{outlook}}}{\text{SplitE}_{\text{outlook}}} = 0.44$$

对 temperature 属性，取值为 cool 的样本有 4 条，取值为 hot 的样本有 4 条，取值为 mild 的有 6 条，则

$$\text{SplitE}_{\text{temperature}} = -\frac{4}{14} \log_2 \frac{4}{14} - \frac{4}{14} \log_2 \frac{4}{14} - \frac{6}{14} \log_2 \frac{6}{14} = 1.556$$

$$\text{GainRatio}_{\text{temperature}} = \frac{\text{Gain}_{\text{temperature}}}{\text{SplitE}_{\text{temperature}}} = \frac{0.029}{1.556} = 0.019$$



对 humidity 属性，取值为 high 的样本有 7 条，取值为 normal 的样本有 7 条，则

$$\text{SplitE}_{\text{humidity}} = -\frac{7}{14} \log_2 \frac{7}{14} - \frac{7}{14} \log_2 \frac{7}{14} = 1$$

$$\text{GainRatio}_{\text{humidity}} = \frac{\text{Gain}_{\text{humidity}}}{\text{SplitE}_{\text{humidity}}} = \frac{0.152}{1} = 0.152$$

对 wind 属性，取值为 weak 的样本有 8 条，取值为 strong 的样本有 6 条，则

$$\text{SplitE}_{\text{wind}} = -\frac{8}{14} \log_2 \frac{8}{14} - \frac{6}{14} \log_2 \frac{6}{14} = 0.985$$

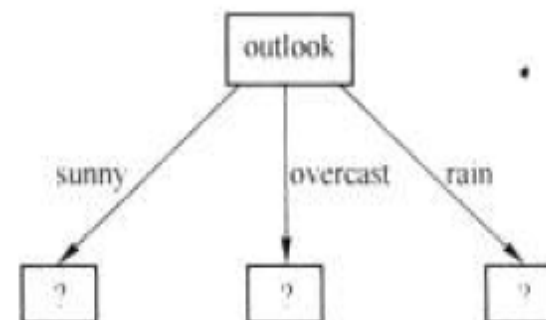
$$\text{GainRatio}_{\text{wind}} = \frac{\text{Gain}_{\text{wind}}}{\text{SplitE}_{\text{wind}}} = \frac{0.049}{0.985} = 0.0497$$



C4.5分类算法 算例

第三步：计算确定最佳划分根节点3个分支的决策属性，即哪些属性分别用来最佳划分根节点的**sunny**、**overcast**、**rain**分支

- 首先对outlook的sunny分支建立子树
 - 找出样本数据集weather中outlook取值sunny的样本子集S_{sunny}
 - 依次计算剩下三个属性对该样本子集S_{sunny}划分后的信息增益率
 - 建立sunny分支子树

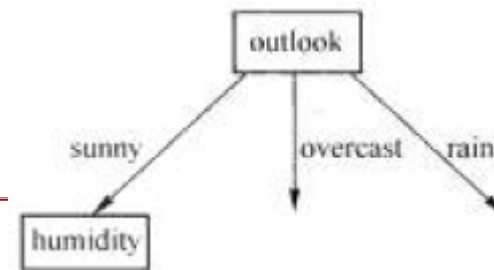


weather 数据集中 outlook 取值为 sunny 的数据子集 S_{sunny}

outlook	temperature	humidity	wind	play ball
sunny	hot	high	weak	no
sunny	hot	high	strong	no
sunny	mild	high	weak	no
sunny	cool	normal	weak	yes
sunny	mild	normal	strong	yes



C4.5分类算法 算例



● 首先对outlook的sunny分支建立子树

— 找出样本数据集weather中outlook取值sunny的样本子集S_{sunny}

— 依次计算剩下三个属性对该样本子集S_{sunny}划分后的信息增益率

— 建立sunny分支子树

① 属性 humidity 划分子集 S_{sunny} 的信息增益、分裂信息及信息增益率为：

$$\text{Gain}(S_{\text{sunny}}, \text{humidity}) = 0.971$$

$$\text{SplitE}(S_{\text{sunny}}, \text{humidity}) = -\frac{3}{5} \log_2 \frac{3}{5} - \frac{2}{5} \log_2 \frac{2}{5} = 0.971$$

$$\text{GainRatio}(S_{\text{sunny}}, \text{humidity}) = \frac{\text{Gain}(S_{\text{sunny}}, \text{humidity})}{\text{SplitE}(S_{\text{sunny}}, \text{humidity})} = \frac{0.971}{0.971} = 1$$

② 属性 temperature 划分子集 S_{sunny} 的信息增益、分裂信息及信息增益率为：

$$\text{Gain}(S_{\text{sunny}}, \text{temperature}) = 0.571$$

$$\text{SplitE}(S_{\text{sunny}}, \text{temperature}) = -\frac{2}{5} \log_2 \frac{2}{5} - \frac{2}{5} \log_2 \frac{2}{5} - \frac{1}{5} \log_2 \frac{1}{5} = 1.521$$

$$\text{GainRatio}(S_{\text{sunny}}, \text{temperature}) = \frac{\text{Gain}(S_{\text{sunny}}, \text{temperature})}{\text{SplitE}(S_{\text{sunny}}, \text{temperature})} = \frac{0.571}{1.521} = 0.375$$

③ 属性 wind 划分子集 S_{sunny} 的信息增益、分裂信息及信息增益率为：

$$\text{Gain}(S_{\text{sunny}}, \text{wind}) = 0.371$$

$$\text{SplitE}(S_{\text{sunny}}, \text{wind}) = -\frac{3}{5} \log_2 \frac{3}{5} - \frac{2}{5} \log_2 \frac{2}{5} = 0.971$$

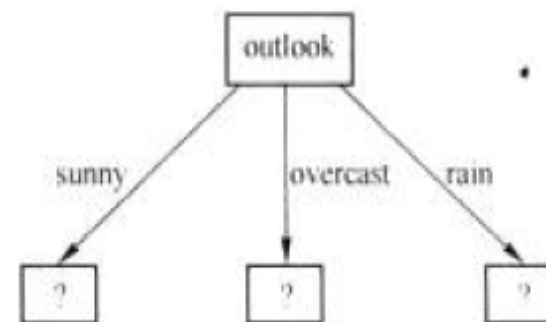
$$\text{GainRatio}(S_{\text{sunny}}, \text{wind}) = \frac{\text{Gain}(S_{\text{sunny}}, \text{wind})}{\text{SplitE}(S_{\text{sunny}}, \text{wind})} = \frac{0.371}{0.971} = 0.382$$



C4.5分类算法 算例

第三步：计算确定最佳划分根节点3个分支的决策属性，即哪些属性分别用来最佳划分根节点的sunny、overcast、rain分支

- 其次对outlook的overcast分支建立子树
 - 找出样本数据集weather中outlook取值overcast的样本子集S_{overcast}
 - 依次计算剩下两个属性对该样本子集S_{overcast}划分后的信息增益率
 - 建立overcast分支子树



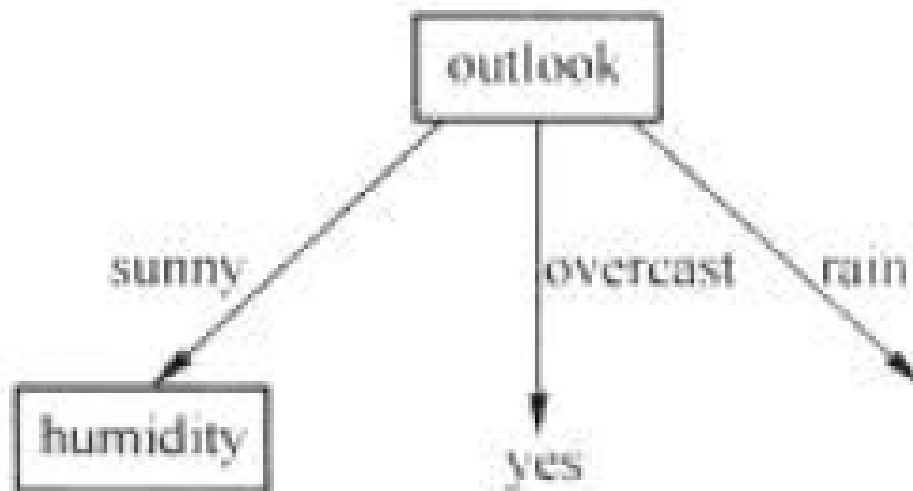
weather 数据集中 outlook 取值为 overcast 的数据子集 S_{overcast}

outlook	temperature	humidity	wind	play ball
overcast	not	high	weak	yes
overcast	cool	normal	strong	yes
overcast	mild	high	strong	yes
overcast	hot	normal	weak	yes



C4.5分类算法 算例

- 其次对outlook的overcast分支建立子树
 - 找出样本数据集weather中outlook取值overcast的样本子集Sovercast
 - 依次计算剩下两个属性对该样本子集Sovercast划分后的信息增益率
 - 建立overcast分支子树



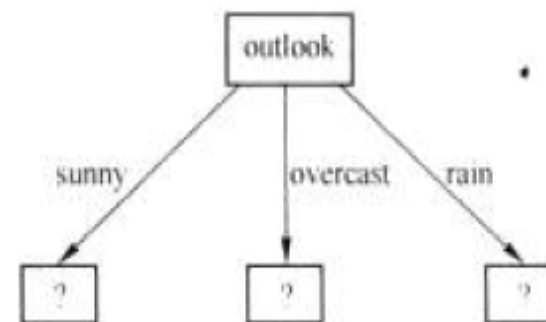


C4.5分类算法 算例

第三步：计算确定最佳划分根节点3个分支的决策属性，即哪些属性分别用来最佳划分根节点的sunny、overcast、rain分支

- 最后对outlook的rain分支建立子树

- 找出样本数据集weather中outlook取值rain的样本子集 S_{rain}
- 依次计算剩下两个属性对该样本子集 S_{rain} 划分后的信息增益率
- 建立rain分支子树



weather 数据集中 outlook 取值为 rain 的数据子集 S_{rain}

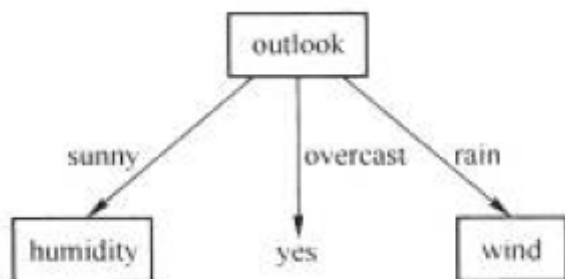
outlook	temperature	humidity	wind	play ball
rain	mild	high	weak	yes
rain	cool	normal	weak	yes
rain	cool	normal	strong	no
rain	mild	normal	weak	yes
rain	mild	high	strong	no



C4.5分类算法 算例

最后对outlook的rain分支建立子树

- 找出样本数据集weather中outlook取值rain的样本子集S_{rain}
- 依次计算剩下两个属性对该样本子集S_{rain}划分后的信息增益率
- 建立rain分支子树



① $\text{Entropy}(S_{\text{rain}}) = \text{Entropy}\left(\frac{3}{5}, \frac{2}{5}\right) = -\frac{3}{5} \log \frac{3}{5} - \frac{2}{5} \log \frac{2}{5} = 0.971$

② 属性 temperature 划分子集 S_{rain} 的信息增益、分裂信息及信息增益率为

$$\begin{aligned}\text{Entropy}_{\text{temperature}}(S_{\text{rain}}) &= \sum_{i=1}^k \frac{|S_i|}{|S|} \text{Entropy}(S_i) = \frac{|S_1|}{|S|} \text{Entropy}(S_1) + \frac{|S_2|}{|S|} \text{Entropy}(S_2) \\ &= \frac{3}{5} \left(-\frac{2}{3} \log_2 \frac{2}{3} - \frac{1}{3} \log_2 \frac{1}{3} \right) + \frac{2}{5} \left(-\frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{2} \log_2 \frac{1}{2} \right) \\ &= 0.9509\end{aligned}$$

$$\text{Gain}(S_{\text{rain}}, \text{temperature}) = \text{Entropy}(S_{\text{rain}}) - \text{Entropy}_{\text{temperature}}(S_{\text{rain}}) = 0.971 - 0.9509 = 0.02$$

$$\text{SplitE}(S_{\text{rain}}, \text{temperature}) = -\frac{3}{5} \log \frac{3}{5} - \frac{2}{5} \log \frac{2}{5} = 0.971$$

$$\text{GainRatio}(S_{\text{rain}}, \text{temperature}) = \frac{\text{Gain}(S_{\text{rain}}, \text{temperature})}{\text{SplitE}(S_{\text{rain}}, \text{temperature})} = \frac{0.02}{0.971} = 0.0205$$

③ 属性 wind 划分子集 S_{rain} 的信息增益、分裂信息及信息增益率为

$$\begin{aligned}\text{Entropy}_{\text{wind}}(S_{\text{rain}}) &= \sum_{i=1}^k \frac{|S_i|}{|S|} \text{Entropy}(S_i) = \frac{|S_1|}{|S|} \text{Entropy}(S_1) + \frac{|S_2|}{|S|} \text{Entropy}(S_2) \\ &= \frac{3}{5} \times 0 + \frac{2}{5} \times 0 \\ &= 0\end{aligned}$$

$$\text{Gain}(S_{\text{rain}}, \text{wind}) = \text{Entropy}(S_{\text{rain}}) - \text{Entropy}_{\text{wind}}(S_{\text{rain}}) = 0.971 - 0 = 0.971$$

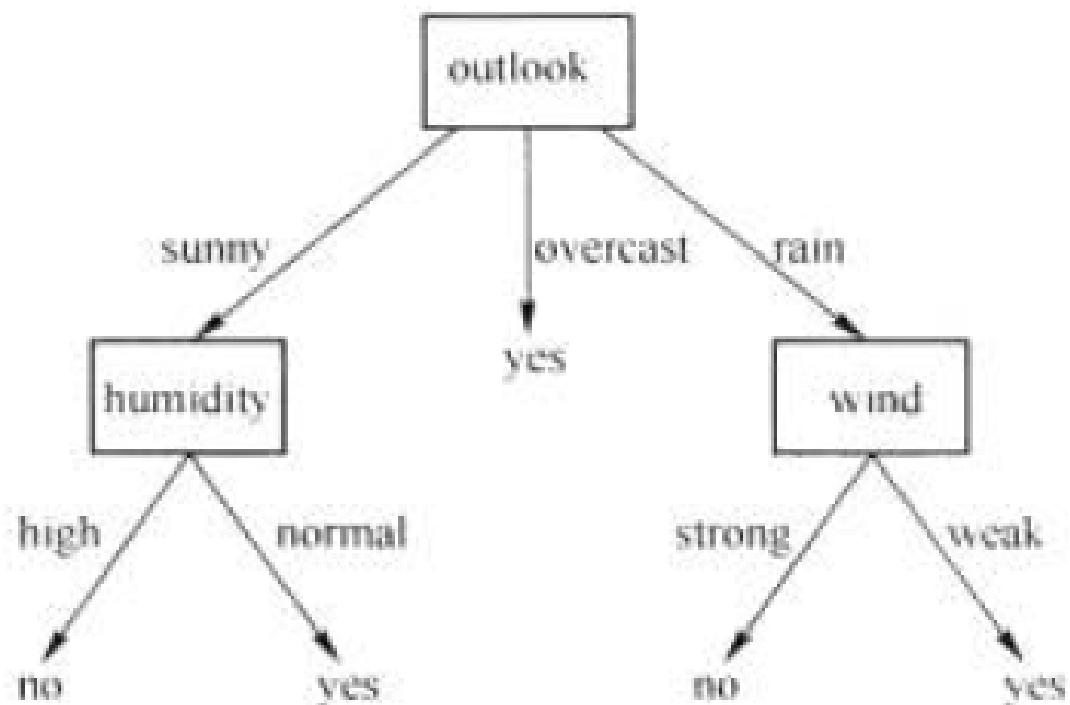
$$\text{SplitE}(S_{\text{rain}}, \text{wind}) = -\frac{3}{5} \log \frac{3}{5} - \frac{2}{5} \log \frac{2}{5} = 0.971$$

$$\text{GainRatio}(S_{\text{rain}}, \text{wind}) = \frac{\text{Gain}(S_{\text{rain}}, \text{wind})}{\text{SplitE}(S_{\text{rain}}, \text{wind})} = \frac{0.971}{0.971} = 1$$



C4.5分类算法 算例

第四步：分别对**humidity**和**wind**两个节点递归调用本方法，最后形成决策树





C4.5算法分析

④ 优点

- 产生的决策树简单直观
- 产生的分类规则易于解释和应用
- 分类准确率高

④ 缺点

- 在构造决策树过程中，需要对数据集进行**多次顺序扫描和排序**，导致算法低效



决策树归纳经典算法

(二) C4.5 (教材中描述)



Gain Ratio 属性选择度量 之 增益率

- 信息增益度量偏向具有许多输出的测试（大量值的属性）
- 不希望产生大量输出的测试条件
 - 限制测试条件，只能二元划分 e.g. CART
 - 修改评估划分标准，把属性测试条件产生输出考虑进去
- 增益率 用 分裂信息 值将信息增益规范化

- 计算方式

— 分裂信息

$$SplitInfo_A(D) = - \sum_{j=1}^v \frac{|D_j|}{|D|} \times \log_2 \left(\frac{|D_j|}{|D|} \right)$$

— 增益率

$$GainRate(A) = \frac{Gain(A)}{SplitInfo_A(D)}$$



决策树归纳经典算法

(三) CART



CART算法简介

- **CART (Classification and Regression Tree, 分类与回归树)** 算法由斯坦福大学和加州大学伯克利分校的Breiman等人与1984年提出
- **CART决策树能够处理连续属性数据和标称属性作为预测变量和目标变量下的分类**
 - 当输出变量是标称属性数据时，所建立的决策树被称为**分类树 (classification tree)**，用于分类的预测
 - 当输出变量是数值型的数据时，所建立的决策树被称为**回归树 (regression tree)**，用于数值的预测
- 分类回归树算法同样包括**决策树增长**和**决策树剪枝**两个过程，算法采用**二元递归划分方法**，将当前的样本集分为两个样本子集，使得生成的决策树的每个非叶节点都有两个分支，**CART算法生成的决策树是结构简洁的二叉树**



CART与C4.5算法比较

- **决策树形式不同**
 - CART为二叉树
 - C4.5是多叉树
- **不纯度量不同**
 - CART使用Gini系数
 - C4.5使用信息增益率
- **输入标量和输出变量类型不同**
 - CART中输入标量和输出变量可以是分类型或数值型的，而C4.5的输出变量只能是分类型
 - 如果目标变量是标称的，并且具有两个以上的类别，CART算法可能考虑将目标类别合并成两个超类别
 - 如果目标变量是连续的，CART算法找出一组基于树的回归方程来预测目标变量
- **对于缺失值的处理方法不同**
 - CART采用代理测试来估计测试的输出值
 - C4.5直接将其分配到该分支中概率最大的分类
- **对决策树的剪枝方法不同**
 - CART采用的是代价复杂度模型，通过交叉验证来估计对测试样本集的误分类损失，产生最小交叉验证误分类估计的树
 - C4.5启发式地调整在训练样本上估计出的误差率，使用调整的误差率，以找到使评分函数最大化的树



CART算法的基本概念

- ④ 设训练样本集 $S=\{X_1, X_2, \dots, X_m\}$ ，其中 X_m 称为属性向量， Y 称为类标号向量
 - 当 Y 为连续型数据时，称为回归树
 - 当 Y 为离散型数据时，称为分类树
- ④ 属性选择标准
 - 使用Gini系数度量对某个属性变量测试输出的两组取值的差异性
 - 理想的分组应该尽量使两组中样本输出变量取值的差异性总和达到最小，即“纯度”最大，也就是使两组变量取值的差异性下降最快，“纯度”增加最快



Gini Index 属性选择度量之基尼系数

- 设 t 是分类回归树中的某个节点， k 为当前属性下测试输出的类别数， $p(j|t)$ 为节点 t 中样本测试输出取类别 j 的概率

$$\text{Gini}(t) = 1 - \sum_{j=1}^k p^2(j|t)$$

- 对节点 t ， $\text{G}(t)$ 越小，意味着该节点中所包含的样本越集中在某一类上，即该节点越纯，否则越不纯，差异性就越大
 - 当节点样本的测试输出均取同一类取值时，输出变量取值的差异性最小，Gini系数为0
 - 当各类别取概率值相等时，测试输出取值的差异性最大，Gini系数也最大，为 $1-1/k$ ，其中 k 为目标变量的类别数



Gini Index 属性选择度量之基尼系数

- 设 t 是一个节点， ξ 为该节点的一个属性分支条件，该分支条件将节点 t 中样本分别分到左分支 S_L 和右分支 S_R 右，则称下式为在分支条件 ξ 下节点 t 的差异性损失，其中， $gini(t)$ 为划分前测试输出的 Gini 系数， $|S_R|$ 和 $|S_L|$ 分别表示划分后左右分支的样本个数

$$\Delta gini(\xi, t) = gini(t) - \frac{|S_R|}{|S_L| + |S_R|} gini(t_R) - \frac{|S_L|}{|S_L| + |S_R|} gini(t_L)$$

- 为了使节点 t 尽可能纯，需要选择某个属性分支条件 ξ ，使该节点的差异性损失尽可能大，用 $\xi(t)$ 表示所考虑的分支条件 ξ 的全体，则选择分支条件因为

$$\xi_{\max} = \arg \max_{\xi \in \xi(t)} \nabla gini(\xi, t)$$



CART算法的属性选择

- 分类树的属性选择

- 分类型

- CART只能建立二叉树，对于取多个值的属性变量，需要将多类别合并成两个类别，形成“超类”，然后计算两“超类”下样本测试输出取值的差异性

- 数值型

- 将数据按升序排序，然后从小到大依次以相邻数值的中间值作为分隔，将样本分为两组，并计算所得组中样本测试输出取值的差异性

- 回归树的属性选择

- 回归树确定当前最佳分组变量的策略与分类树相同，主要不同在于度量节点测试输出值差异性的指标有所不同

- t 为节点， N 为节点 t 所含的样本个数， $y_i(t)$ 为节点 t 中测试输出变量值， $\overline{y(t)}$ 为节点 t 中测试输出变量的平均值， $R(t)$ 和 N 分别为分组前输出变量的方差和样本个数

- $R(t_R)$ 、 N_R 和 $R(t_L)$ 、 N_L 分别为分组后右子树的方差和样本量以及左子树的方差和样本量

- 差异性损失的度量指标为方差的减少量，使其达到最大的属性变量为当前最佳划分属性变量

$$R(t) = \frac{1}{N-1} \sum_{i=1}^N (y_i(t) - \overline{y(t)})^2 \quad \Delta R(t) = R(t) - \frac{N_R}{N} R(t_R) - \frac{N_L}{N} R(t_L)$$



CART算法的描述

- 函数名: **CART(S,F)**
- 输入: 样本数据集**S**, 训练集数据属性集合**F**
- 输出: **CART**树
- (1)if样本**S**全部属于同一个类别**C**, then
- (2) 创建一个叶节点, 并标记类标号为**C**
- (3) return
- (4)else
- (5) 计算属性集**F**中每一个属性划分的差异性损失, 假定差异性损失最大的属性为**A**
- (6) 创建节点, 取属性**A**为该节点的决策属性
- (7) 以属性**A**划分**S**得到**S1**和**S2**两个子集 /*当属性**A**有多于2个取值时, 可能会有多个两组划分, 此时取差异性损失最大的那个划分**S1**和**S2** */
- (8) 递归调用**CART(S1, F)**
- (9) 递归调用**CART(S2, F)**
- (10)end



CART算法构造决策树：一个例子

序号	是否有房	婚姻状况	年收入	拖欠贷款
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



CART算法构造决策树：一个例子

- (1) 首先对数据集非类标号属性{是否有房，婚姻状况，年收入}分别计算他们的差异性损失，取差异性损失最大的属性作为决策树的根节点属性

开始创建的节点为根节点，假定为 r ，该根节点的Gini系数为

$$\text{Gini}(r) = 1 - \left(\frac{3}{10}\right)^2 - \left(\frac{7}{10}\right)^2 = 0.42$$



CART算法构造决策树：一个例子

①对是否有房属性

$$\begin{aligned}\Delta gini(\text{是否有房}|r) &= gini(r) - \frac{|S_{\text{是否有房=no}}|}{|S_{\text{是否有房=yes}}| + |S_{\text{是否有房=no}}|} gini(r_{\text{是否有房=no}}) - \frac{|S_{\text{是否有房=yes}}|}{|S_{\text{是否有房=yes}}| + |S_{\text{是否有房=no}}|} gini(r_{\text{是否有房=yes}}) \\ &= 0.42 - \frac{7}{10} (1 - (\frac{3}{7})^2 - (\frac{4}{7})^2) - \frac{3}{10} \times (1 - (\frac{3}{3})^2 - (\frac{0}{3})^2) = 0.077\end{aligned}$$



CART算法构造决策树：一个例子

②对婚姻状况属性

属性婚姻状况有3个可能的取值{married,single,divorced}，分别划分后的超类{married}/{single,divorced}、{single}/{married,divorced}、{divorced}/{single, married}的差异性损失

1) 当分组为{married}/{single,divorced}时， S_L 表示婚姻状况取值为 married 的分组， S_R 表示婚姻状况取值为{single,divorced}的分组

$$\Delta gini(\text{婚姻状况}, r) = gini(r) - \frac{|S_R|}{|S_L| + |S_R|} gini(r_R) - \frac{|S_L|}{|S_L| + |S_R|} gini(r_L)$$

$$= 0.42 - \frac{4}{10} \left(1 - \left(\frac{4}{4} \right)^2 - \left(\frac{0}{4} \right)^2 \right) - \frac{6}{10} \times \left(1 - \left(\frac{3}{6} \right)^2 - \left(\frac{3}{6} \right)^2 \right) = 0.12$$



CART算法构造决策树：一个例子

②对婚姻状况属性

属性婚姻状况有3个可能的取值{married,single,divorced}，分别划分后的超类{married}/{single,divorced}、{single}/{married,divorced}、{divorced}/{single, married}的差异性损失

2) 当分组为{single}/{married,divorced}时， S_L 表示婚姻状况取值为single的分组， S_R 表示婚姻状况取值为{married,divorced}的分组

$$\Delta gini(\text{婚姻状况}, r) = gini(r) - \frac{|S_R|}{|S_L| + |S_R|} gini(r_R) - \frac{|S_L|}{|S_L| + |S_R|} gini(r_L)$$

$$= 0.42 - \frac{4}{10} \left(1 - \left(\frac{2}{4} \right)^2 - \left(\frac{2}{4} \right)^2 \right) - \frac{6}{10} \times \left(1 - \left(\frac{1}{6} \right)^2 - \left(\frac{5}{6} \right)^2 \right) = 0.053$$



CART算法构造决策树：一个例子

②对婚姻状况属性

属性婚姻状况有3个可能的取值{married,single,divorced}，分别划分后的超类{married}/{single,divorced}、{single}/{married,divorced}、{divorced}/{single, married}的差异性损失

3) 当分组为{divorced}/{single, married}时， S_L 表示婚姻状况取值为divorced的分组， S_R 表示婚姻状况取值为{single, married}的分组

$$\begin{aligned}\Delta gini(\text{婚姻状况}, r) &= gini(r) - \frac{|S_R|}{|S_L| + |S_R|} gini(r_R) - \frac{|S_L|}{|S_L| + |S_R|} gini(r_L) \\ &= 0.42 - \frac{2}{10} \left(1 - \left(\frac{1}{2} \right)^2 - \left(\frac{1}{2} \right)^2 \right) - \frac{8}{10} \times \left(1 - \left(\frac{6}{8} \right)^2 - \left(\frac{2}{8} \right)^2 \right) = 0.02\end{aligned}$$

根据1)、2)和3)计算，属性婚姻状况划分根节点时取差异性损失最大的分组作为划分结果，即{married}/{single,divorced}



CART算法构造决策树：一个例子

③对年收入属性

年收入为数值型属性，首先需要对数据按升序排列，然后从小到大一次以相邻值的中间值作为分隔将样本分为两组，取差异性损失最大的分隔作为该属性的分组

以第一个相邻值得中间值**65**为例，将其作为分隔点， S_L 表示年收入小于65的样本， S_R 表示年收入大于65的样本

$$\begin{aligned}\Delta gini(\text{年收入}, r) &= gini(r) - \frac{|S_R|}{|S_L| + |S_R|} gini(r_R) - \frac{|S_L|}{|S_L| + |S_R|} gini(r_L) \\ &= 0.42 - \frac{1}{10} (1 - (\frac{1}{1})^2 - (\frac{0}{1})^2) - \frac{9}{10} \times (1 - (\frac{6}{9})^2 - (\frac{3}{9})^2) = 0.02\end{aligned}$$

拖欠贷款	no	no	no	yes	yes	yes	no	no	no	no
年收入	60	70	75	85	90	95	100	120	125	220
相邻值中点	65	72.5	80	87.5	92.5	97.5	110	122.5	172.5	
差异性损失	0.02	0.045	0.077	0.003	0.02	0.12	0.077	0.045	0.02	



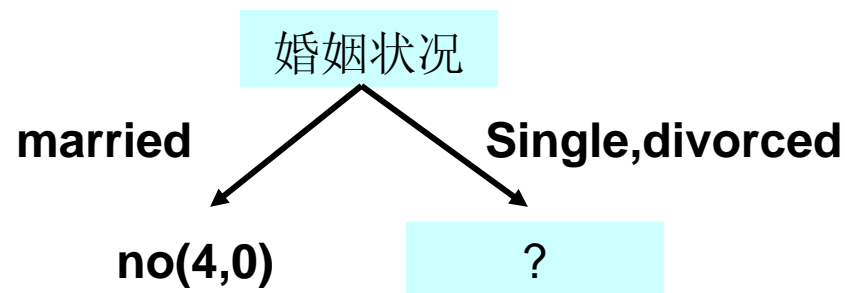
CART算法构造决策树：一个例子

(1) 首先对数据集非类标号属性{是否有房，婚姻状况，年收入}分别计算他们的差异性损失，取差异性损失最大的属性作为决策树的根节点属性

根据①、②、③的计算，{是否有房，婚姻状况，年收入}3个属性中划分根节点差异性损失最大的有2个：婚姻状况、年收入的差异性损失值都为0.12

按CART采取的方法，根据属性出现的先后顺序来选择其中一个作为当前节点划分的决策属性。本例中，婚姻状况属性在顺序上先于年收入属性，因此取婚姻状况作为根节点的决策属性

此时得到第一次划分结果



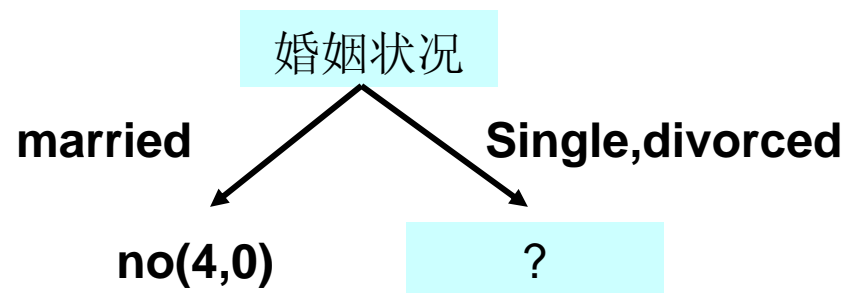


CART算法构造决策树：一个例子

- (2) 采用同样方法，分别计算3个属性对婚姻状况取{single,divorced}的数据子集进行划分的差异性损失，取最大的那个属性作为当前节点的决策属性

假设当前节点为t，它的Gini系数为

$$\text{Gini}(t) = 1 - \left(\frac{3}{6}\right)^2 - \left(\frac{3}{10}\right)^2 = 0.5$$





CART算法构造决策树：一个例子

①对是否有房属性

$$\begin{aligned}\Delta gini(\text{是否有房}, t) &= gini(t) - \frac{|S_{\text{是否有房=no}}|}{|S_{\text{是否有房=yes}}| + |S_{\text{是否有房=no}}|} gini(t_{\text{是否有房=no}}) - \frac{|S_{\text{是否有房=yes}}|}{|S_{\text{是否有房=yes}}| + |S_{\text{是否有房=no}}|} gini(t_{\text{是否有房=yes}}) \\ &= 0.5 - \frac{4}{6} (1 - (\frac{3}{4})^2 - (\frac{1}{4})^2) - \frac{2}{6} \times (1 - (\frac{2}{2})^2 - (\frac{0}{2})^2) = 0.25\end{aligned}$$



CART算法构造决策树：一个例子

②对婚姻状况属性

属性婚姻状况有3个可能的取值{married,single,divorced}，分别划分后的超类{married}/{single,divorced}、{single}/{married,divorced}、{divorced}/{single, married}的差异性损失

1) 当分组为{married}/{single,divorced}时， S_L 表示婚姻状况取值为married的分组， S_R 表示婚姻状况取值为{single,divorced}的分组

$$\begin{aligned}\Delta gini(\text{婚姻状况}, t) &= gini(t) - \frac{|S_R|}{|S_L| + |S_R|} gini(t_R) - \frac{|S_L|}{|S_L| + |S_R|} gini(t_L) \\ &= 0.5 - \frac{4}{6} \left(1 - \left(\frac{2}{4} \right)^2 - \left(\frac{2}{4} \right)^2 \right) - \frac{2}{6} \times \left(1 - \left(\frac{1}{2} \right)^2 - \left(\frac{1}{2} \right)^2 \right) = 0\end{aligned}$$

2) 当分组{single}/{married,divorced}时，结果0.056

3) 当分组 {divorced}/{single, married}时，结果0.1

根据1)、2)、3)的计算，取差异性损失最大的分组{married}/{single,divorced}作为该属性的分组



CART算法构造决策树：一个例子

③对年收入属性

年收入为数值型属性，首先需要对数据按升序排列，然后从小到大一次以相邻值的中间值作为分隔将样本分为两组，取差异性损失最大的分隔作为该属性的分组

以第一个相邻值得中间值**77.5**为例，将其作为分隔点， S_L 表示年收入小于77.5的样本， S_R 表示年收入大于77.5的样本

$$\begin{aligned}\Delta gini(\text{年收入}, r) &= gini(t) - \frac{|S_R|}{|S_L| + |S_R|} gini(t_R) - \frac{|S_L|}{|S_L| + |S_R|} gini(t_L) \\ &= 0.5 - \frac{1}{6} \left(1 - \left(\frac{1}{1} \right)^2 - \left(\frac{0}{1} \right)^2 \right) - \frac{5}{6} \times \left(1 - \left(\frac{2}{5} \right)^2 - \left(\frac{3}{5} \right)^2 \right) = 0.1\end{aligned}$$

拖欠贷款	no	yes	yes	yes	no	no
年收入	70	85	90	95	125	220
相邻值中点	77.5	87.5	92.5	110	172.5	
差异性损失	0.1	0.25	0.05	0.25	0.1	



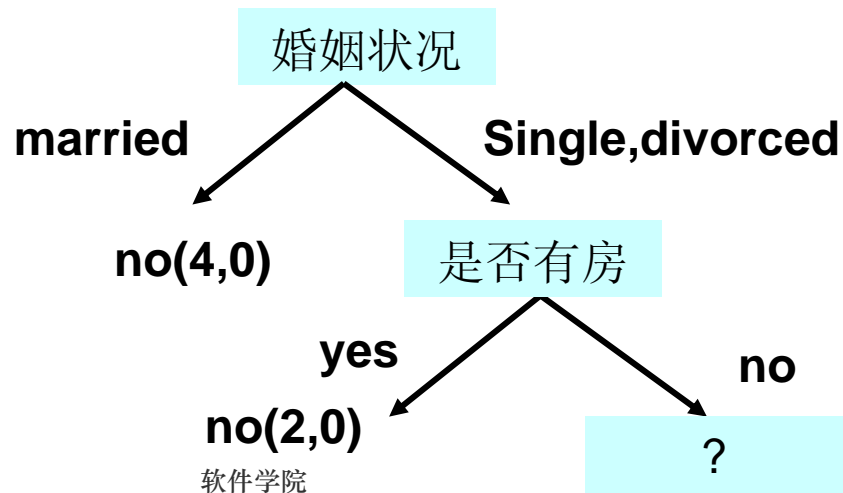
CART算法构造决策树：一个例子

(2) 采用同样方法，分别计算3个属性对婚姻状况取{single,divorced}的数据子集进行划分的差异性损失，取最大的那个属性作为当前节点的决策属性

根据①、②、③的计算，{是否有房，婚姻状况，年收入}3个属性划分根节点差异性损失最大值为0.25：是否有房、年收入的差异性损失值都为0.25

按CART采取的方法，根据属性出现的先后顺序来选择其中一个作为当前节点划分的决策属性。本例中，是否有房属性在顺序上先于年收入属性，因此取是否有房作为当前节点的决策属性

此时得到第二次划分结果

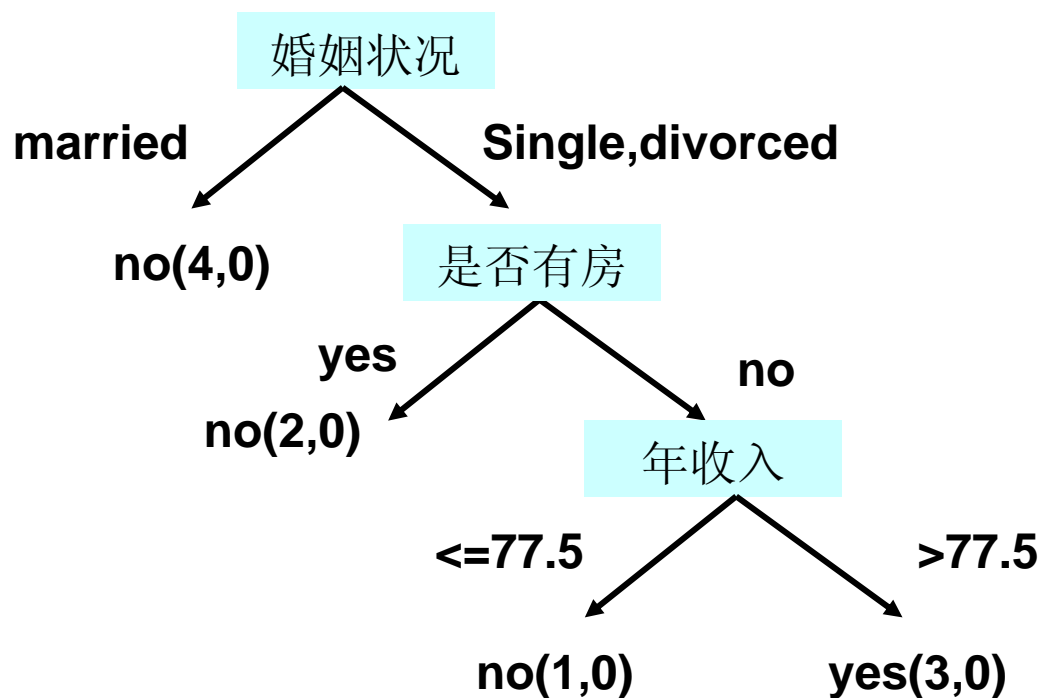




CART算法构造决策树：一个例子

(3) 采用同样方法，分别计算3个属性对剩下数据子集的划分，取最大的那个属性作为当前节点的决策属性

得到第三次划分结果





CART算法的剪枝

CART采用预剪枝和后剪枝结合的方式进行剪枝

(1) CART的预剪枝

预剪枝的目标是控制决策树的充分生长，通过事先指定一些控制参数完成

- ① **决策树最大深度**：如果决策树的层数已经达到指定深度，则停止生长
- ② **树中父节点和子节点所包含的最少样本量或比例**：对父节点，是指如果节点所包含的样本量已经低于最少样本量或比例，则不再划分；对子节点，是指如果划分后生成的子节点所包含的样本量已经低于最少样本量或比例，则不再划分
- ③ **树节点中测试输出结果的最小差异减少量**：如果划分后所产生的测试输出结果差异性变化量小于一个指定值，则不必进行划分



CART算法的剪枝

CART采用预剪枝和后剪枝结合的方式进行剪枝

(2) CART的后剪枝

后剪枝允许决策树的充分生长，然后在此基础上根据一定规则，剪去决策树中的那些不具有代表性的叶节点或子树，是一个边剪枝边检验的过程。在剪枝过程中，不断计算当前决策子树对检验样本集中测试输出结果的预测精度或误差，并判断是继续剪枝还是停止剪枝。



不同决策树算法的比较

- 属性选择度量方法不同
 - 信息增益
 - 增益率
 - 基尼指数
- 防止过分适应数据 的方法不同
 - 先剪枝
 - 后剪枝
- 适应的预测变量和目标变量的属性类型不同
 - 标称属性
 - 数值型数据
 - 缺失数据...



大型数据库的分类挖掘 —— 可伸缩性

- 分类挖掘是一个在统计学和机器学习的领域也被广为研究的问题，并提出了很多算法，但是这些算法都是 **内存驻留** 的
- **可伸缩性问题：**

由于训练元组在主存和高速缓存之间换进换出，决策树的构造可能变得效率低下。

因此，要求以合理的速度对数以百万计的样本和数以百计的属性的进行分类挖掘。
- **解决方法：由大型数据库构造判定树**
 - 首先将样本划分为**子集**，每个子集可以放在内存中
 - 然后由每个子集构造一颗决策树
 - 输出的分类法将每个子集的分类法组合在一起
 - （其他方法包括SLIQ, SPRINT,RainForest等等）



RainForest (雨林)

- ④ 在每个节点，对每个属性维护一个AVC-集（属性-值，类标号）
- ④ 节点N上属性A的AVC-集
 - N上元组中，对应A的每个值，不同的类标号对应的计数
- ④ AVC-组群
 - 结点N上所有AVC-集的集合
- ④ 节点N上属性A的AVC-集的大小依赖于
 - A的不同值的个数
 - N上元组集合中类的个数



RainForest (雨林) Cont.

• E.g., RainForest(雨林)

Training Examples

age	income	student	credit_rating	comp
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no

AVC-set on Age

Age	Buy_Computer	
	yes	no
<=30	2	3
31..40	4	0
>40	3	2

AVC-set on *income*

income	Buy_Computer	
	yes	no
high	2	2
medium	4	2
low	3	1

AVC-set on *Student*

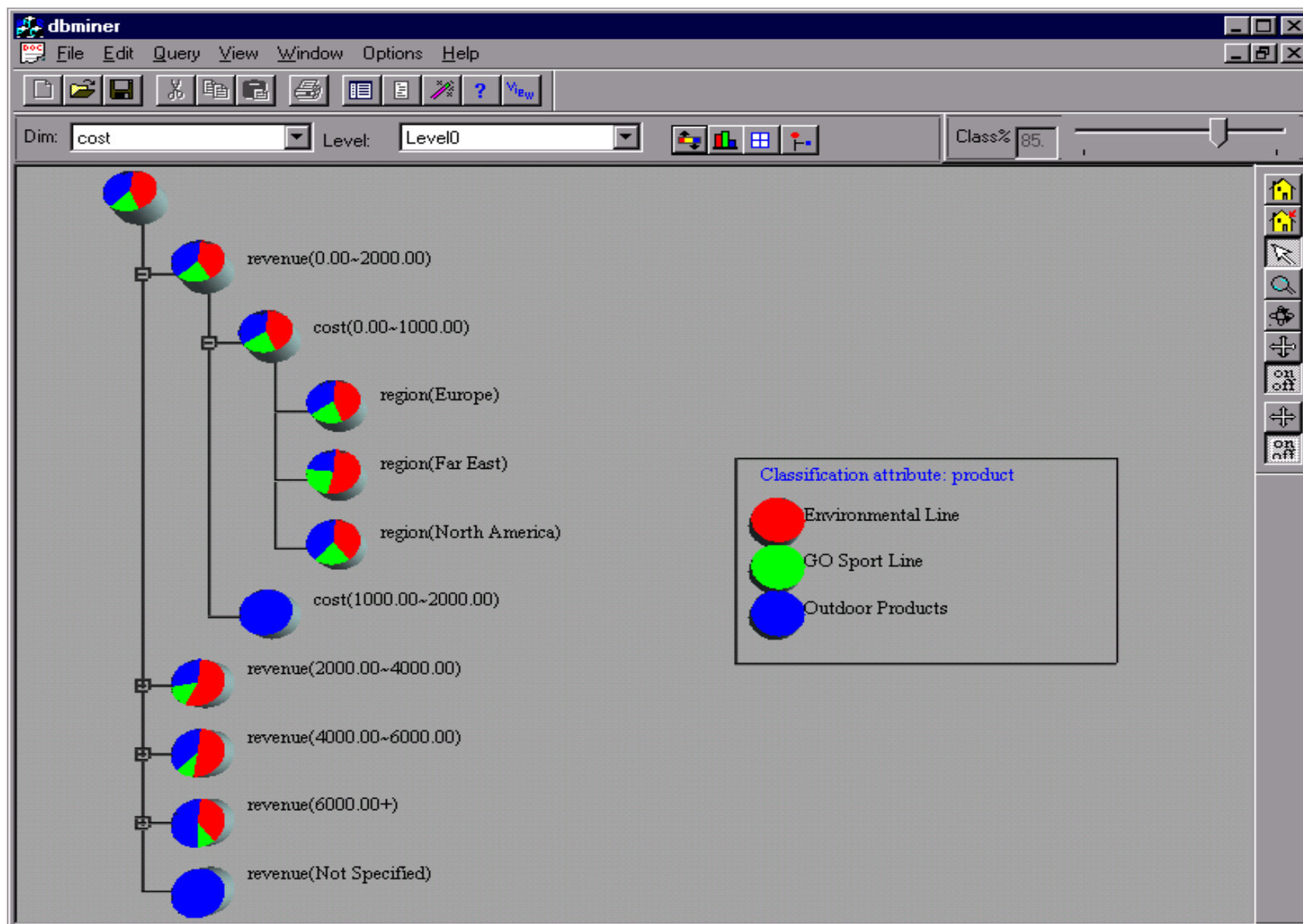
student	Buy_Computer	
	yes	no
yes	6	1
no	3	4

AVC-set on *credit_rating*

Credit rating	Buy_Computer	
	yes	no
fair	6	2
excellent	3	3

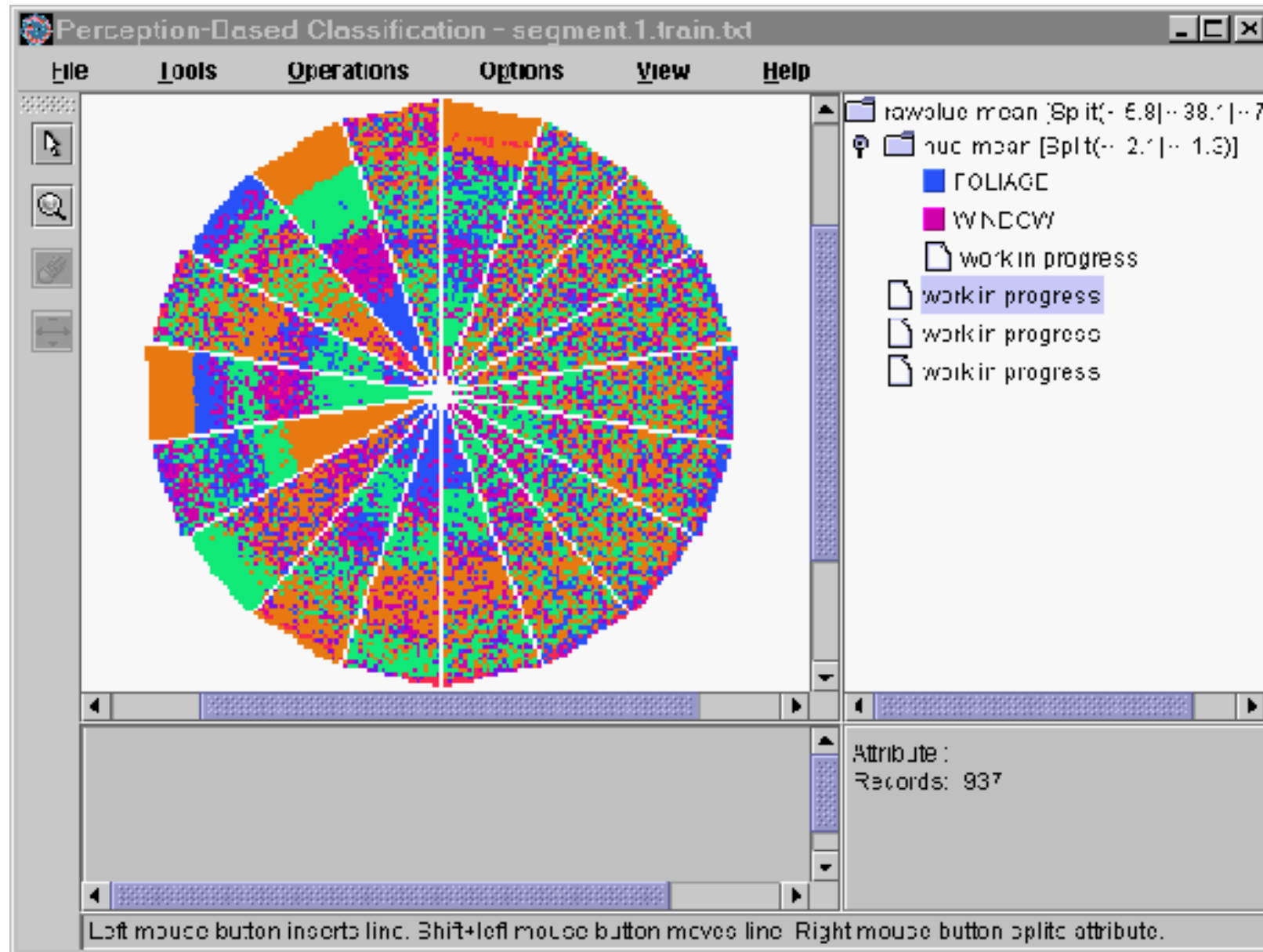


决策树 —— 可视化挖掘





决策树 —— 可视化挖掘(Cont.)





Chapter 8. Class 分类

- Classification: Basic Concepts 基本概念
- Decision Tree Induction 决策树归纳
- **Bayes Classification Methods** 贝叶斯分类
- Rule-Based Classification 基于规则的分类
- Model Evaluation and Selection 模型评估与选择
- Techniques to Improve Classification Accuracy
提高分类准确率的技术



贝叶斯分类方法：为什么？

- 一个统计学的分类方法
 - 预测类隶属关系的概率
 - 一个给定的元组属于一个特定类的概率
- 基础： **贝叶斯定理**
- 有效性： 与决策树和神经网络等多种分类法的对比实验表明，在某些领域，贝叶斯分类法足以与它们相媲美
- 标准： 贝叶斯分类可以用来为不直接使用贝叶斯定理的其他分类方法提供理论判定



贝叶斯分类方法的主要特点

- 充分利用领域知识和其它先验信息，显示地计算假设概率，分类结果是领域知识和数据样本信息的综合体现
- 利用有向图表示方式，用弧表示变量之间的依赖关系，用概率分布表示依赖关系的强弱，表示方法非常直观，有利于对领域知识的理解
- 能进行增量学习，数据样本可以增量地提高或降低某种假设的估计，并且能方便地处理不完整的数据



贝叶斯定理：基础

- 朴素贝叶斯分类算法和贝叶斯信念网络分类算法都是建立在贝叶斯定理基础上

- 基本概念

- 设 X 是类标签未知的数据样本
 - X : 一位35岁、收入4万美元的顾客
- 设 H 为某种假设，如数据元组 X 属于某个特定类 C
 - H : 顾客将购买计算机
- $P(H)$ （先验概率）：假设 H 的先验概率 任意给定样本：先验
 - 任意给定顾客将购买计算机的概率，不管年龄、收入或其他任何信息
- $P(H|X)$ （后验概率）：给定一个观察数据样本 X ，假设成立的概率 给定特定样本
 - 反映知道顾客的年龄和收入时，顾客 X 将买计算机的概率
- $P(X)$ （先验概率）：样本数据 X 的先验概率
 - 顾客集合中35岁、收入4万美元的概率
- $P(X|H)$ （后验概率）：在给定假设 H 的前提条件下，样本 X 的后验概率
 - 已知顾客购买计算机，那么该顾客35岁且收入4万美元的概率



贝叶斯定理：基础

● 贝叶斯定理

- 假设 X 和 Y 是一对随机变量
- **联合概率** $P(X=x, Y=y)$ 是指 X 取 x 且 Y 取值 y 的概率
- **条件概率**是指一随机变量在另一随机变量取值已知的情况下取某一特定值得概率
 - $P(Y=y|X=x)$ 是指在变量 X 取值 x 的情况下，变量 Y 取值 y 的概率
- **X 和 Y 的联合概率和条件概率满足**

$$P(X, Y) = P(Y | X)P(X) = P(X | Y)P(Y) \Rightarrow P(Y | X) = \frac{P(X | Y)}{P(X)} P(Y)$$

- 对应样本 \mathbf{X} 和假设 H ，根据贝叶斯定理，满足如下关系

$$P(H | \mathbf{X}) = \frac{P(\mathbf{X} | H)P(H)}{P(\mathbf{X})} = P(\mathbf{X} | H) \times P(H) / P(\mathbf{X})$$

- 贝叶斯定理是在 H 和 \mathbf{X} 的先验概率、 \mathbf{X} 的后验概率已知的情况下，计算 H 后验概率的方法，分类取决于 **$P(H|\mathbf{X})$** （ H 的后验概率）



贝叶斯定理：例

- 一堆水果，其中60%是苹果，剩下的是梨；苹果为黄色的概率为20%，梨为黄色的概率为80%。随机从这堆水果中拿出一个黄色的水果，最有可能是拿到梨还是苹果？
 - 假定随机变量X代表水果，X取值范围为{苹果,梨}，随机变量Y代表黄色
 - 已知苹果在水果堆中的概率为 $P(X=\text{苹果})=0.6$ ，梨在水果堆中的概率为 $P(X=\text{梨})=0.4$ ，苹果为黄色的概率为 $P(Y=\text{黄色}|X=\text{苹果})=0.2$ ，梨为黄色的概率为 $P(Y=\text{黄色}|X=\text{梨})=0.8$
 - 需要计算 $P(X=\text{苹果}|Y=\text{黄色})$ 和 $P(X=\text{梨}|Y=\text{黄色})$
- $P(X=\text{苹果}|Y=\text{黄色})=(P(Y=\text{黄色}|X=\text{苹果}) * P(X=\text{苹果}))/ P(Y=\text{黄色})$
 $=0.2*0.6/ P(Y=\text{黄色})=0.12 / P(Y=\text{黄色})$
- $P(X=\text{梨}|Y=\text{黄色})=(P(Y=\text{黄色}|X=\text{梨}) * P(X=\text{梨}))/ P(Y=\text{黄色})$
 $=0.8*0.4/ P(Y=\text{黄色})=0.32/P(Y=\text{黄色})$
- X=梨的后验概率更大，最有可能拿到梨



基于贝叶斯定理的分类器模型

- **朴素贝叶斯 (Naïve Bayes, NB) 分类器**
 - 贝叶斯分类器中最简单有效的，并且在实际使用中较为成功的一种分类器。其性能可以与神经网络、决策树分类器相比，在某些场合优于其他分类器。朴素贝叶斯分类器的特征是假定每个属性的取值对给定类的影响独立于其他属性的取值，即给定类变量的条件下各属性变量之间条件独立
- **树扩展的朴素贝叶斯 (Tree-Augmented Naïve Bayes, TANB)**
 - 在朴素贝叶斯分类器的基础上，在属性之间添加连接弧，在一定程度上消除朴素贝叶斯分类器的条件独立性假设，这样的弧称为扩展弧，说明树形约束
- **BAN (Bayesian network-Augmented Naïve Bayes)**
 - 一种增强的朴素贝叶斯分类器，改进了朴素贝叶斯分类器的条件独立假设，并且取消了TANB分类器中属性变量之间必须符合树状结构的要求，假定属性之间存在贝叶斯网络关系而不是树状关系，从而能够表达属性变量的各种依赖关系
- **贝叶斯多网 (Bayesian Muti-Net, BMN) 分类器**
 - TANB和BAN的一个扩展，TANB和BAN分类器认为对不同的类别，各属性变量之间的关系是不变的，即对于不同的类别具有相同的网络结构。而BMN认为对类变量的不同取值，各属性变量之间的关系可能是不一样的
- **GBN分类器 (General Bayesian Network)**
 - 如果抛弃条件独立假设，就可以用一般贝叶斯网络作为分类器。GBN是一种无约束的贝叶斯网络分类器，与前面4类贝叶斯分类器不同的，前面4类分类器都将类变量作为一个特殊节点，类节点在网络结构中是各个属性的父节点，而GBN将类节点作为一个普通节点



朴素贝叶斯分类的概念

- 设 D 是训练元组和他们相关联的类标号的集合，每个元组用一个 n 维属性向量 $X = (x_1, x_2, \dots, x_n)$ ，假设有 m 个类 C_1, C_2, \dots, C_m
- 分类法将预测 X 属于具有 **最高后验概率** 的类，即 $P(C_i|X)$ 的最大值
 $P(C_i|X) > P(C_j|X) (1 \leq j \leq m, j \neq i)$

- 根据贝叶斯定理，由于 $P(X)$ 对于所有类为常数，所以只要 **$P(X|C_i)P(C_i)$** 最大即可

$$P(C_i | X) = \frac{P(X | C_i)P(C_i)}{P(X)} = P(X | C_i) \times P(C_i) / P(X)$$

- 类的先验概率可以用 **$P(C_i) = |C_i, D| / |D|$** 估计， $|C_i, D|$ 是 D 中 C_i 类的训练元组数
- 如果 $P(C_i)$ 未知，那么假设这些类等概率，软件学院只要 **$P(X|C_i)$** 最大即可



朴素贝叶斯分类的概念Cont.

- 计算 $P(\mathbf{X}|\mathbf{C}_i)$ 开销可能非常大
- 简单假设 —— 属性有条件的相互独立（属性之间不存在依赖关系），则：

$$P(\mathbf{X}|\mathbf{C}_i) = \prod_{k=1}^n P(x_k | \mathbf{C}_i) = P(x_1 | \mathbf{C}_i) \times P(x_2 | \mathbf{C}_i) \times \dots \times P(x_n | \mathbf{C}_i)$$

- 其中 x_k 表示元组 \mathbf{X} 在属性 A_k 的值
- 如果 A_k 是分类属性， $P(x_k|\mathbf{C}_i)$ 是 D 中属性 A_k 的值为 x_k 的 \mathbf{C}_i 类的元组数除以 \mathbf{C}_i 类的元组数 $|\mathbf{C}_i, D|$



朴素贝叶斯分类的概念Cont.

- 如果 A_k 是连续值属性，可以用两种方法估计连续属性的类条件概率
- 方法一
 - 可以将每个连续的属性离散化，然后用相应的离散区间替换连续属性值。把连续属性转换为序数属性，通过计算类 C_i 的训练样本中落入 x_k 对应区间的比例来估计条件概率 $P(x_k|C_i)$
 - 估计误差由离散策略和离散区间的数目决定：如果离散区间的数目太大，则会因为每个区间中训练样本太少而不能对 $P(x_k|C_i)$ 做出可靠的估计；如果区间数目太小，有些区间可能会包含来自不同类的样本，因此会失去正确的决策边界
- 方法二
 - 可以假设连续变量服从某种概率分布，然后使用训练样本估计分布的参数
 - 正态分布通常被用来表示连续属性的类条件概率分布，分布有均值和方差（数据集中属于 C_i 类的样本属性 A_k 的平均值和方差），类别 C_i 下属性 x_k 的类条件概率近似为

$$P(X|C_i) \approx g(x_k, \mu_{C_i}, \sigma_{C_i}) \quad g(x, \mu_{C_i}, \sigma_{C_i}) = \frac{1}{\sqrt{2\pi}\sigma_{C_i}} e^{-\frac{(x-\mu_{C_i})^2}{2\sigma_{C_i}^2}}$$



朴素贝叶斯分类的算法基本描述

- 函数名: NaïveBayes
- 输入: 类标号未知的样本 $X=\{x_1, x_2, \dots, x_n\}$
- 输出: 位置样本 X 所属的类标号
- (1) for $j=1$ to m
- (2) 计算 X 属于每个类别 C_j 的概率
$$P(X|C_j) = \prod_{k=1}^n P(x_k | C_j) = P(x_1 | C_j) \times P(x_2 | C_j) \times \dots \times P(x_n | C_j)$$
- (3) 计算训练集中每个类别的概率 $P(C_j)$
- (4) 计算概率值 $\mu = P(X|C_j) * P(C_j)$
- End for
- 选择计算概率值 μ 最大的 C_i ($1 \leq i \leq m$) 最为类别的输出



朴素贝叶斯分类：一个例子

Class:

C1:buys_computer = 'yes'

C2:buys_computer = 'no'

Data to be classified:

X = (age <=30,

Income = medium,

Student = yes

Credit_rating = Fair)

age	income	student	credit_rating	_com
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no



朴素贝叶斯分类：一个例子

- $P(C_i)$: $P(\text{buys_computer} = \text{"yes"}) = 9/14 = 0.643$
 $P(\text{buys_computer} = \text{"no"}) = 5/14 = 0.357$
- Compute $P(X|C_i)$ for each class
 - $P(\text{age} = \text{"<=30"} | \text{buys_computer} = \text{"yes"}) = 2/9 = 0.222$
 - $P(\text{age} = \text{"<= 30"} | \text{buys_computer} = \text{"no"}) = 3/5 = 0.6$
 - $P(\text{income} = \text{"medium"} | \text{buys_computer} = \text{"yes"}) = 4/9 = 0.444$
 - $P(\text{income} = \text{"medium"} | \text{buys_computer} = \text{"no"}) = 2/5 = 0.4$
 - $P(\text{student} = \text{"yes"} | \text{buys_computer} = \text{"yes"}) = 6/9 = 0.667$
 - $P(\text{student} = \text{"yes"} | \text{buys_computer} = \text{"no"}) = 1/5 = 0.2$
 - $P(\text{credit_rating} = \text{"fair"} | \text{buys_computer} = \text{"yes"}) = 6/9 = 0.667$
 - $P(\text{credit_rating} = \text{"fair"} | \text{buys_computer} = \text{"no"}) = 2/5 = 0.4$
- $X = (\text{age} \leq 30, \text{income} = \text{medium}, \text{student} = \text{yes}, \text{credit_rating} = \text{fair})$
 - $P(X|C_i) : P(X|\text{buys_computer} = \text{"yes"}) = 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044$
 - $P(X|\text{buys_computer} = \text{"no"}) = 0.6 \times 0.4 \times 0.2 \times 0.4 = 0.019$
 - $P(X|C_i) \cdot P(C_i) : P(X|\text{buys_computer} = \text{"yes"}) \cdot P(\text{buys_computer} = \text{"yes"}) = 0.028$
 - $P(X|\text{buys_computer} = \text{"no"}) \cdot P(\text{buys_computer} = \text{"no"}) = 0.007$

Therefore, X belongs to class ("buys_computer = yes")

软件学院



朴素贝叶斯分类：小作业

表 3-2 weather

outlook	temperature	humidity	wind	play ball
sunny	hot	high	weak	no
sunny	hot	high	strong	no
overcast	hot	high	weak	yes
rain	mild	high	weak	yes
rain	cool	normal	weak	yes
rain	cool	normal	strong	no
overcast	cool	normal	strong	yes
sunny	mild	high	weak	no
sunny	cool	normal	weak	yes
rain	mild	normal	weak	yes
sunny	mild	normal	strong	yes
overcast	mild	high	strong	yes
overcast	hot	normal	weak	yes
rain	mild	high	strong	no



朴素贝叶斯分类：避免零概率值问题

- 如果某个 $P(x_k|C_i)$ 为零概率值，那么 $P(X|C_i)$ 的值将返回零
 - 一个零概率将消除乘积中涉及的（ C_i 上）所有其他（后验）概率的影响
- 解决方法
 - 有一个简单的技巧来避免该问题。可以假定训练数据库D很大，以至于对每个计数加1造成的估计概率的变化可以忽略不计，但可以方便地避免概率值为零。这种概率估计技术称为拉普拉斯校准或拉普拉斯估计法，以法国数学家皮埃尔·拉普拉斯（Pierre Laplace, 1749—1827年）的名字命名
 - 如果对q个计数加上1，则必须记住在用于计算概率的对应分母上加上q
- 例：假设一个数据集有1000个元组
 - $income=low(0), income=medium(990), income=high(10)$
 - 使用拉布拉斯校准
 - 假定训练数据库D很大，以至于对每个计数加1造成的估计概率的变化可以忽略不计
 - $Prob(income = low) = 1/1003$
 - $Prob(income = medium) = 991/1003$
 - $Prob(income = high) = 11/1003$
 - 校准的概率估计对应未校准的估计很接近



Chapter 8. Class 分类

- Classification: Basic Concepts 基本概念
- Decision Tree Induction 决策树归纳
- Bayes Classification Methods 贝叶斯分类
- **Rule-Based Classification** 基于规则的分类
- Model Evaluation and Selection 模型评估与选择
- Techniques to Improve Classification Accuracy
提高分类准确率的技术



Using IF-THEN Rules for Classification

使用 IF-THEN 规则分类 (Cont)

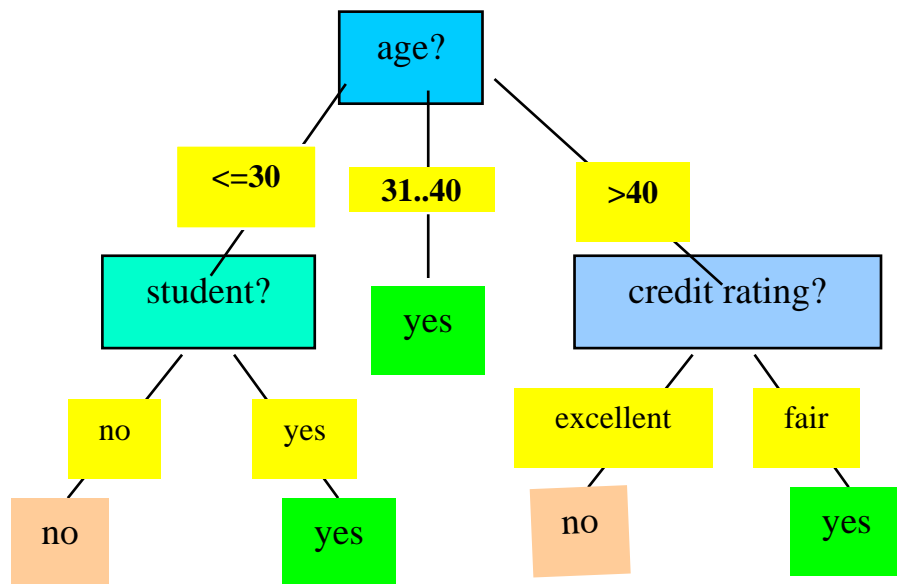
- 如果规则被 X 满足，则称该规则被 **触发**。
- 如果规则 $R1$ 是 **唯一** 满足的规则，则该规则被 **激活**，返回 X 的类预测。
- 如果有 **多个规则** 被触发，则需要解决冲突
 - 如果她们指向了不同的类怎么办？
 - 规模序：把最高优先权赋予具有**最多**属性测试的规则 **按属性个数排序**
 - 基于类的规则序：优先级按照普遍性降序排序。每个类中规则是无序的，不存在类冲突。**按照类的频繁度降序排列**
 - 基于规则的规则序：**根据准确度、覆盖率**等等吧规则组织成一个优先权列表 (**决策表**)
 - 如果不存在 X 满足规则的情况？
 - 根据训练集指定一个默认类（默认类的选择视算法而定）



Rule Extraction from a Decision Tree

由决策树提取规则

- IF-THEN规则更容易理解
- 对每条从根到树叶节点的路径创建一个规则
- E.g., *buys_computer* 决策树



IF *age* = young AND *student* = no
THEN *buys_computer* = no

IF *age* = young AND *student* = yes
THEN *buys_computer* = yes

IF *age* = mid-age
THEN *buys_computer* = yes

IF *age* = old AND *credit_rating* =
excellent THEN *buys_computer* = no

IF *age* = old AND *credit_rating* = *fair*
THEN *buys_computer* = yes



Rule Induction: Sequential Covering Method

使用顺序覆盖算法的规则归纳

- ④ 顺序覆盖算法：直接从训练数据提取IF-THEN规则
 - 不必产生决策树
 - 规则被顺序地学习，一次一个
- ④ 典型的顺序覆盖算法：FOIL,AQ,CN2,RIPPER
- ④ 为C类学习规则时，希望它能够覆盖C类所有（或许多）的训练元组，并且没有（或很少）覆盖其他类的元组
- ④ 步骤：
 - 一次学习一条规则
 - 每学习一个规则，就把该规则覆盖的元组删除
 - 在剩下的元组上重复该过程直到满足终止条件 e.g. 没有更多的训练元组或者返回的规则质量低于用户指定的阈值
- ④ 与决策树归纳做比较：同时学习一组规则



顺序覆盖算法

- **Input:**

- *D, a data set of class-labeled tuples;*
- *Att vals, the set of all attributes and their possible values.*

- **Output: A set of IF-THEN rules.**

- **Method:**

- (1) *Rule set D fg; // initial set of rules learned is empty*
- (2) *for each class c do*
- (3) *repeat*
- (4) *Rule D Learn One Rule(D, Att vals, c);*
- (5) *remove tuples covered by Rule from D;*
- (6) *Rule set D Rule set CRule; // add new rule to rule set*
- (7) *until terminating condition;*
- (8) *endfor*
- (9) *return Rule Set ;*



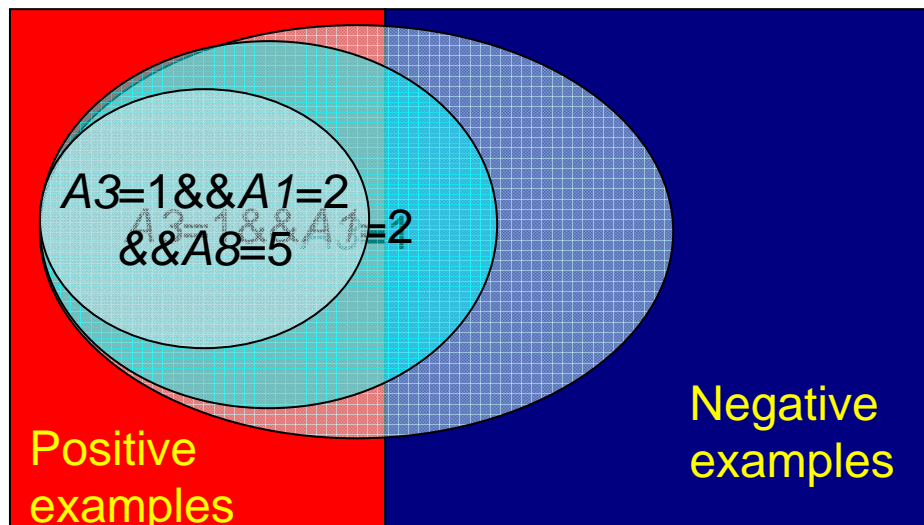
规则生成

- ④ 从空规则开始，然后逐渐向它添加属性测试
 - 添加的属性测试作为规则前件条件的逻辑合取
- ④ 一个例子：训练集D由贷款申请数据组成：
 - 属性：年龄，收入，文化程度，住处，信誉等级和贷款期限
 - 分类属性：loan_decision（是否接受贷款申请）
- ④ 刚开始的规则为：IF THEN load_decision=accept
- ④ 考虑每个可能的属性测试
 - 对属性-值对(attr, val)考虑attr=val, attr<=val, attr>val
- ④ 缺点：计算昂贵



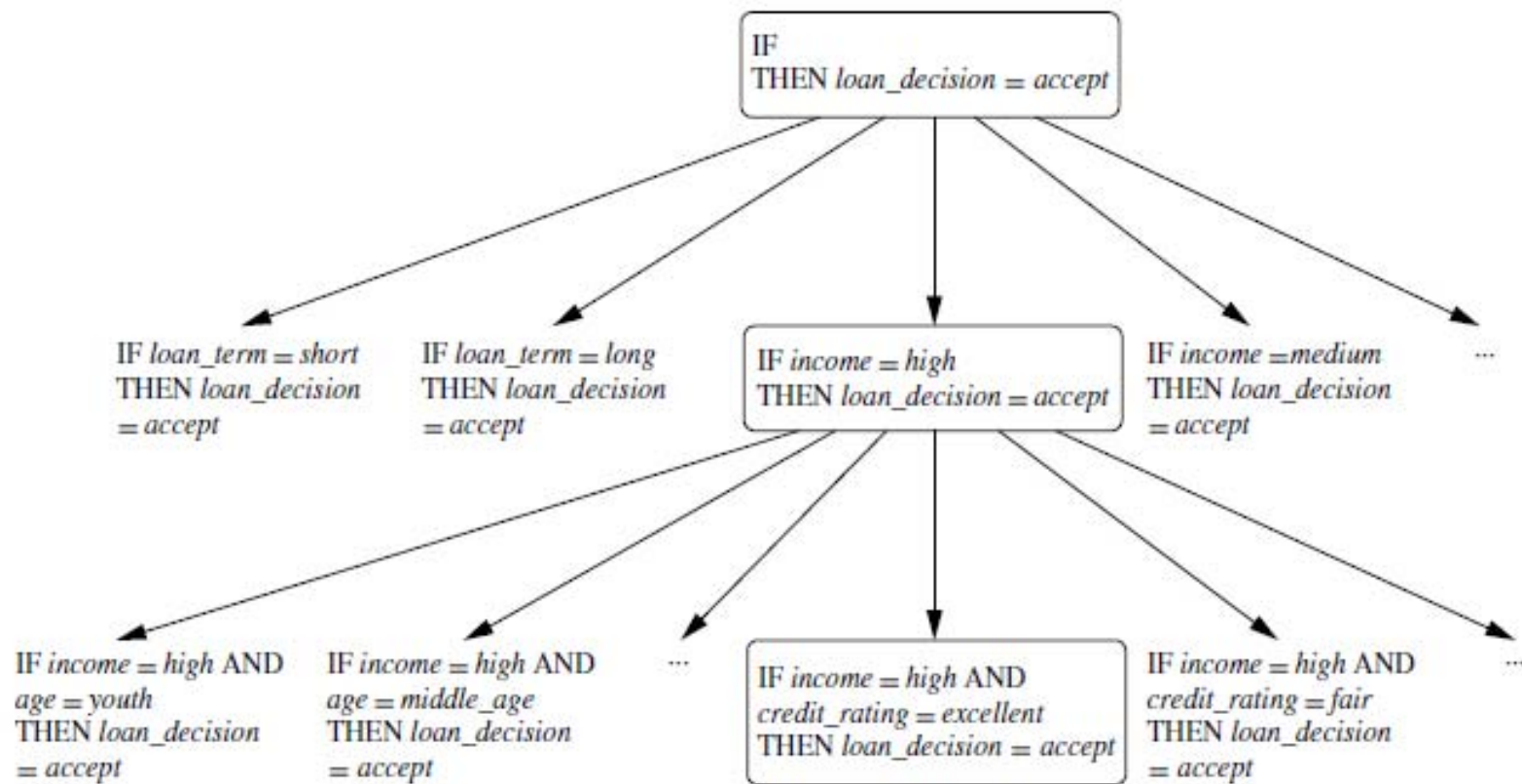
规则生成

- 可以采用一种贪心的深度优先策略
 - 选择最能提高规则质量属性的测试
 - 质量度量方法：准确率，熵，FOIL中信息增益的度量
 - 每一步继续贪心地增长规则，直到结果规则达到可接受的质量水平
- 为了生成一条规则：
while(true)
 find the best predicate p
 if $\text{foil-gain}(p) > \text{threshold}$ then add p to current rule
 else break





规则生成





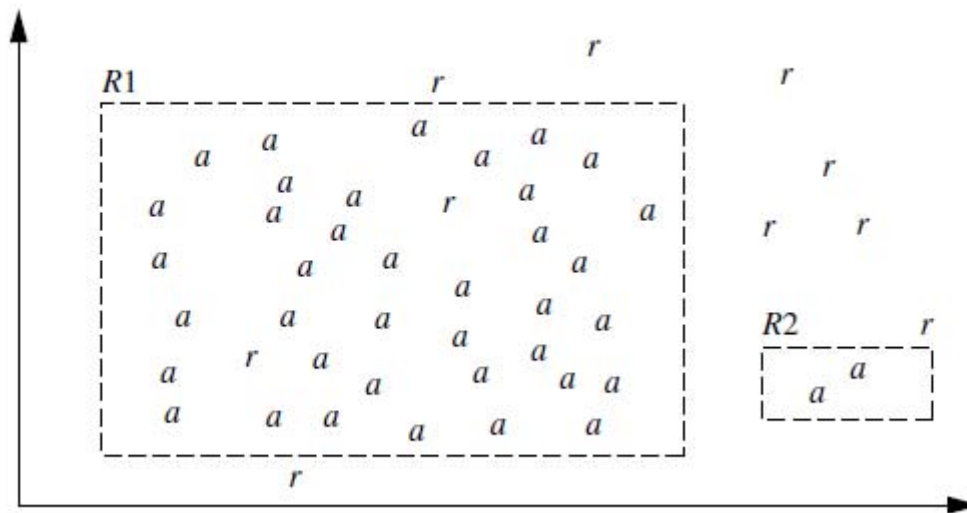
规则质量度量

④ 准确率

—该规则正确分类的元组所占的百分比

④ 有例外情况

④ 如下图所示，两个规则都是loan_decision=accept类的规则





规则质量度量

- ④ 当前的规则是R:IF condition THEN class = c
- ④ 新的条件为condition', R':IF condition' THEN class = c
- ④ 想知道R'是否比R更好
- ④ 熵

——数据集D的元组分类所需要的期望信息

$$Info(D) = - \sum_{i=1}^m p_i \log_2(p_i),$$

- ④ D是condition'覆盖元组的集合, p_i 是D中类 C_i 的概率, 熵
越小, condition'越好
- ④ 偏向于覆盖单个类大量元组和少量其他元组的条件



规则质量度量 Cont.

- ④ FOIL的基于信息增益的度量
- ⑤ 在机器学习中，用于学习规则的类的元组称为正元组，其余元组为负元组
- ⑥ 设 $pos(neg)$ 为R覆盖的正（负）元组数， $pos'(neg')$ 为R'覆盖的正（负）元组数
- ⑦ 用下式估计扩展condition'而获得的信息

$$FOIL_Gain = pos' \times \left(\log_2 \frac{pos'}{pos' + neg'} - \log_2 \frac{pos}{pos + neg} \right).$$

- ⑧ 偏向于具有高准确率并且覆盖许多正元组的规则



规则剪枝

- ④ 上面介绍的规则评估规则使用原训练数据集：
 - 乐观的，过分拟合
 - 训练数据上性能好，以后的数据上就不那么好
- ④ 为了补偿，通过删除一个属性测试对规则剪枝
- ④ 如果在独立的元组集上评估，R剪枝后的版本有着更高的质量，那么选择对规则R剪枝
 - 悲观剪枝
- ④ FOIL中使用的方法

$$FOIL_Prune(R) = \frac{pos - neg}{pos + neg},$$

- ④ 剪枝后FOIL_Prune值较高，就对R剪枝
- ④ RIPPER从最近添加的合取项开始



Chapter 8. Class 分类

- Classification: Basic Concepts 基本概念
- Decision Tree Induction 决策树归纳
- Bayes Classification Methods 贝叶斯分类
- Rule-Based Classification 基于规则的分类
- **Model Evaluation and Selection** 模型评估与选择
- Techniques to Improve Classification Accuracy
提高分类准确率的技术



Model Evaluation and Selection

模型评估与选择

- ④ 评估度量：我们如何衡量准确率？其他的度量方法呢？
- ④ 当评估准确率时使用**验证测试集**而不是训练集中带有类标签的元组
- ④ 用于评估一个分类器的准确性的方法：
 - 保持和随机子抽样
 - 交叉验证
 - 自助方法
- ④ 比较分类器：
 - 置信区间
 - ROC曲线和成本效益分析



模型评估与选择

- ④ 正元组：感兴趣的主要类的元组
- ④ 负元组：其他元组
- ④ 真阳性TP
 - 被分类器正确分类的正元组
- ④ 真阴性TN
 - 被分类器正确分类的负元组
- ④ 假阳性FP
 - 被错误地标记为正元组的负元组
- ④ 假阴性FN
 - 被错误地标记为负元组的正元组
- ④ TP和TN告诉我们分类器何时分类正确
- ④ FP和FN告诉我们分类器何时错误



混淆矩阵

- 混淆矩阵：一个给定 $m(m \geq 2)$ 个类，至少为 $m \times m$ 的表

Actual class \ Predicted class	C_1	$\neg C_1$
C_1	True Positives (TP)	False Negatives (FN)
$\neg C_1$	False Positives (FP)	True Negatives (TN)

- 一个混淆矩阵样例：

Actual class \ Predicted class	buy_computer = yes	buy_computer = no	Total
buy_computer = yes	6954	46	7000
buy_computer = no	412	2588	3000
Total	7366	2634	10000

- $CM_{i,j}$ 指出类 i 的元组被分类器标记为类 j 的个数
- 可能需要多余的列和行来提供合计



准确率，错误率，灵敏性和特效性

A\P	C	-C	
C	TP	FN	P
-C	FP	TN	N
	P'	N'	All

- 分类器**准确率**: 正确分类元组所占的百分比

$$\text{Accuracy} = (TP + TN)/All$$

- Error rate**: $1 - \text{accuracy}$, or
 $\text{Error rate} = (FP + FN)/All$

- 类不平衡问题:

- 感兴趣的主类是稀少的, e.g. 欺诈
- 负类显著地占多数, 正类占少数
- **灵敏性**: 正确识别正元组的百分比
 - $\text{Sensitivity} = TP/P$
- **特效性**: 正确识别负元组的百分比
 - $\text{Specificity} = TN/N$



精度和召回率

- 精度： 标记为正类的元组实际为正类的百分比

$$precision = \frac{TP}{TP + FP}$$

- 召回率： 正元组标记为正的百分比
- 满分为1.0

$$recall = \frac{TP}{TP + FN}$$

- 将精度和召回率组合到一个度量中：
 - **F度量 (F_1 or F -分数):**精度和召回率的调和平均数

$$F = \frac{2 \times precision \times recall}{precision + recall}$$

- **F_β :**精度和召回率的加权度量

$$F_\beta = \frac{(1 + \beta^2) \times precision \times recall}{\beta^2 \times precision + recall},$$



样例

Actual Class\Predicted class	cancer = yes	cancer = no	Total	Recognition(%)
cancer = yes	90	210	300	30.00 (<i>sensitivity</i>)
cancer = no	140	9560	9700	98.56 (<i>specificity</i>)
Total	230	9770	10000	96.40 (<i>accuracy</i>)

- ***Precision* = 90/230 = 39.13%**
- ***Recall* = 90/300 = 30.00%**



Holdout Methods

模型评估与选择 —— 1) 保持 & 随机二次抽样

● Holdout Methods 保持方法

- 给定数据随机地划分为两个独立的集合：训练集(E.g., 2/3) 和 检验集(E.g., 1/3)
- 使用 训练集 导出模型，使用 检验集 估计其准确率（估计是悲观的）

● Random sampling 随机二次抽样

- 保持方法的一种变形
- 将保持方法重复 k 次，总准确率估计取每次迭代准确率的平均值 (avg)



Cross-Validation Methods

模型评估与选择 —— 2) 交叉验证

- K-折交叉验证（ $K=10$ 最为普遍）
- 初始数据随机的划分成 k 个互不相交的子集或者“折”
 - 每个折的大小大致相等
- 在第 i 次迭代，使用 D_i 作为测试集，其余的作为训练集
- 对于分类，准确率估计是 k 次迭代正确分类的元组总数除以初始数据中的元组总数
- 留一： k 设置为初始样本数，每次只给检验集留出一个样本



Bootstrap Methods

模型评估与选择 —— 3) 自助法

● Bootstrap 自助法

— 从给定的训练元组中 **有放回** 的均匀抽样

● 常见自助法: **632 自助法**

— 假定数据集包含 d 个元组, 该数据集有放回地抽样 d 次, 产生 d 个样本的训练集。平均情况下, 63.2% 的原数据元组将会出现在训练集中, 36.8% 的原数据将会形成检验集。

$$(1 - 1/d)^d \approx e^{-1} = 0.368$$

— 模型的总体准确率为:

$$Acc(M) = \frac{1}{k} \sum_{i=1}^k (0.632 \times Acc(M_i)_{test_set} + 0.368 \times Acc(M_i)_{train_set})$$



使用统计显著性检验选择模型

- ❶ 假设我们有两个分类器，M1和M2，哪一个更好？
- ❷ 直观的，可以选择最低错误率的模型
- ❸ 然而
 - 10-折交叉验证实验的错误率之间存在很大的方差
 - 平均错误率可能不同，但差别不是统计显著的



使用统计显著性检验选择模型 Cont.

- ④ 10次10-折交叉验证，每次使用数据不同的10折划分
- ④ 每个划分都独立的抽取
- ④ 每个错误率都可以看做来自一种概率分布不同的独立样本
- ④ 假设这些错误率服从具有k-1个自由度的t分布，其中k等于10

$$t = \frac{\overline{err}(M_1) - \overline{err}(M_2)}{\sqrt{var(M_1 - M_2)/k}}$$

- ④ 使用t-检验
- ④ 单检验集

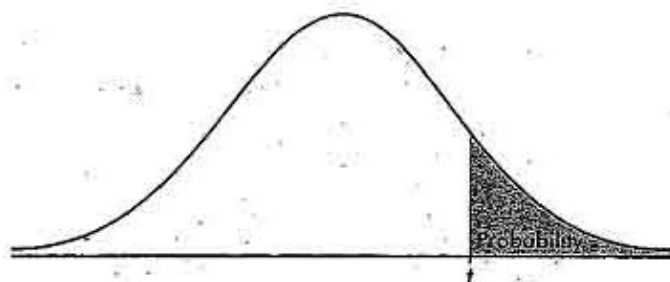
$$var(M_1 - M_2) = \frac{1}{k} \sum_{i=1}^k \left[err(M_1)_i - err(M_2)_i - (\overline{err}(M_1) - \overline{err}(M_2)) \right]^2$$

- ④ 假设两者模型相同（平均错误率之差为0）
- ④ 若拒绝假设，那么差是统计显著的

——选择具有较低错误率的模型



使用统计显著性检验选择模型 Cont.



- 对称的
- 显著水平, e.g., $\text{sig} = 0.05$ or 5%
意味着 M_1 & M_2
的差对总体的
95%显著不同
- 置信界, $z = \text{sig}/2$
- $t > z$ 或 $t < -z$, 可以
拒绝原假设, 有
着统计显著的差
别

TABLE B: t-DISTRIBUTION CRITICAL VALUES

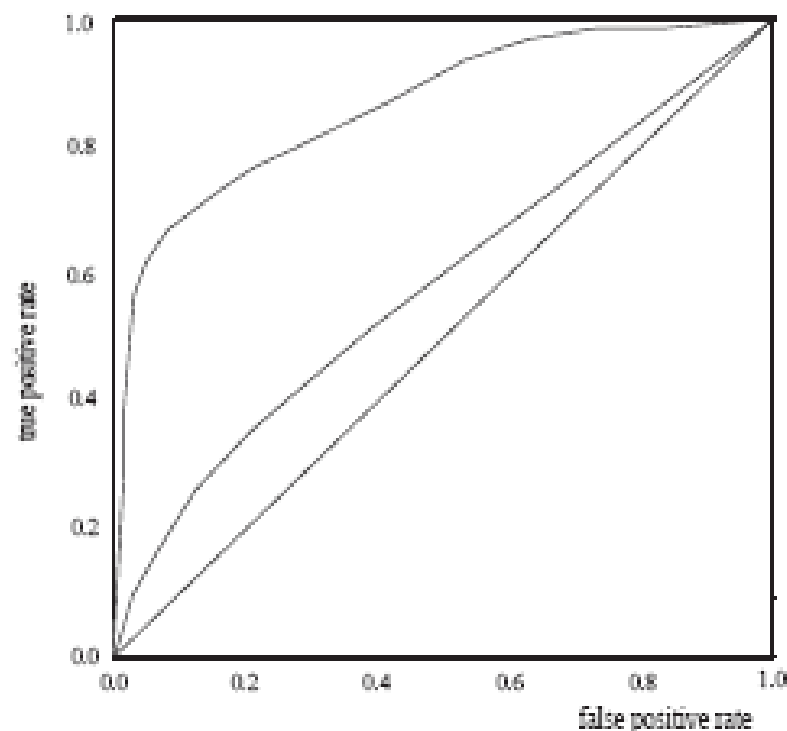
df	Tail probability p											
	.25	.20	.15	.10	.05	.025	.02	.01	.005	.0025	.001	.0005
1	1.000	1.376	1.963	3.078	6.314	12.71	15.89	31.82	63.66	127.3	318.3	636.6
2	.816	1.061	1.386	1.886	2.920	4.303	4.849	6.965	9.925	14.09	22.33	31.60
3	.765	.978	1.250	1.638	2.353	3.182	3.482	4.541	5.841	7.453	10.21	12.92
4	.741	.941	1.190	1.533	2.132	2.776	2.999	3.747	4.604	5.598	7.173	8.610
5	.727	.920	1.156	1.476	2.015	2.571	2.757	3.365	4.032	4.773	5.893	6.869
6	.718	.906	1.134	1.440	1.943	2.447	2.612	3.143	3.707	4.317	5.208	5.959
7	.711	.896	1.119	1.415	1.895	2.365	2.517	2.998	3.499	4.029	4.785	5.408
8	.706	.889	1.108	1.397	1.860	2.306	2.449	2.896	3.355	3.833	4.501	5.041
9	.703	.883	1.100	1.383	1.833	2.262	2.398	2.821	3.250	3.690	4.297	4.781
10	.700	.879	1.093	1.372	1.812	2.228	2.359	2.764	3.169	3.581	4.144	4.587
11	.697	.876	1.088	1.363	1.796	2.201	2.328	2.718	3.106	3.497	4.025	4.437
12	.695	.873	1.083	1.356	1.782	2.179	2.303	2.681	3.055	3.428	3.930	4.318
13	.694	.870	1.079	1.350	1.771	2.160	2.282	2.650	3.012	3.372	3.852	4.221
14	.692	.868	1.076	1.345	1.761	2.145	2.264	2.624	2.977	3.326	3.787	4.140
15	.691	.866	1.074	1.341	1.753	2.131	2.249	2.602	2.947	3.286	3.733	4.073
16	.690	.865	1.071	1.337	1.746	2.120	2.235	2.583	2.921	3.252	3.686	4.015
17	.689	.863	1.069	1.333	1.740	2.110	2.224	2.567	2.898	3.222	3.646	3.965
18	.688	.862	1.067	1.330	1.734	2.101	2.214	2.552	2.878	3.197	3.611	3.922
19	.688	.861	1.066	1.328	1.729	2.093	2.205	2.539	2.861	3.174	3.579	3.883
20	.687	.860	1.064	1.325	1.725	2.086	2.197	2.528	2.845	3.153	3.552	3.850
21	.686	.859	1.063	1.323	1.721	2.080	2.189	2.518	2.831	3.135	3.527	3.819
22	.686	.858	1.061	1.321	1.717	2.074	2.183	2.508	2.819	3.119	3.505	3.792
23	.685	.858	1.060	1.319	1.714	2.069	2.177	2.500	2.807	3.104	3.485	3.768
24	.685	.857	1.059	1.318	1.711	2.064	2.172	2.492	2.797	3.091	3.467	3.745
25	.684	.856	1.058	1.316	1.708	2.060	2.167	2.485	2.787	3.078	3.450	3.725
26	.684	.856	1.058	1.315	1.706	2.056	2.162	2.479	2.779	3.067	3.435	3.707
27	.684	.855	1.057	1.314	1.703	2.052	2.158	2.473	2.771	3.057	3.421	3.690
28	.683	.855	1.056	1.313	1.701	2.048	2.154	2.467	2.763	3.047	3.408	3.674
29	.683	.854	1.055	1.311	1.699	2.045	2.150	2.462	2.756	3.038	3.396	3.659
30	.683	.854	1.055	1.310	1.697	2.042	2.147	2.457	2.750	3.030	3.385	3.646
40	.681	.851	1.050	1.303	1.684	2.021	2.123	2.423	2.704	2.971	3.307	3.551
50	.679	.849	1.047	1.299	1.676	2.009	2.109	2.403	2.678	2.937	3.261	3.496
60	.679	.848	1.045	1.296	1.671	2.000	2.099	2.390	2.660	2.915	3.232	3.460
80	.678	.846	1.043	1.292	1.664	1.990	2.088	2.374	2.639	2.887	3.195	3.416
100	.677	.845	1.042	1.290	1.660	1.984	2.081	2.364	2.626	2.871	3.174	3.390
1000	.675	.842	1.037	1.282	1.646	1.962	2.056	2.330	2.581	2.813	3.098	3.300
∞	.674	.841	1.036	1.282	1.645	1.960	2.054	2.326	2.576	2.807	3.091	3.291
	50%	60%	70%	80%	90%	95%	96%	98%	99%	99.5%	99.8%	99.9%
	Confidence level C											



模型评估与选择 —— ROC 曲线

● ROC (Receiver Operating Characteristic) 曲线

- 显示了给定模型的真正利率(TPR)和假正利率(FPR)之间的权衡
- ROC 曲线离对角线越近
模型的准确率越低
 - 好的模型的 ROC 曲线
- 曲线下方的面积 -> 模型准确率
 - 面积越接近0.5, 准确率越低
- 横坐标: **FPR** 假正利率
- 纵坐标: **TPR** 真正利率





Chapter 8. Class 分类

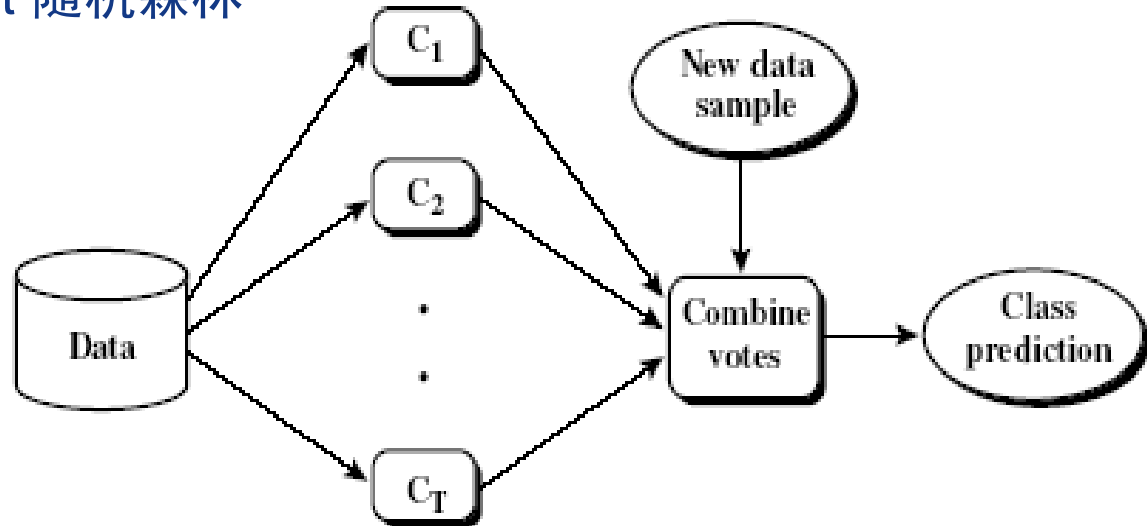
- Classification: Basic Concepts 基本概念
- Decision Tree Induction 决策树归纳
- Bayes Classification Methods 贝叶斯分类
- Rule-Based Classification 基于规则的分类
- Model Evaluation and Selection 模型评估与选择
- **Techniques to Improve Classification Accuracy**
提高分类准确率的技术



Techniques to Improve Classification Accuracy

提高分类准确率的技术

- 通过组合分类器，基于个体分类器投票，返回最终的类标号预测
- 常见的组合方法：
 - Bagging 装袋
 - Boosting 提升
 - Random Forest 随机森林





Bagging: Bootstrap Aggregation

提高分类准确率的技术 —— 1) 装袋

- ④ 多个分类器，每个分类器有着相同的投票权重
 - 多数表决
- ④ d 个元组的集合 D
- ④ 对于迭代 $i(i=1,2,\dots,k)$, d 个元组的训练集 D_i 采用有放回抽样
 - 有些元组不出现
 - 有些元组重复多次
- ④ 每个训练集 D_i 学习得到分类模型 M_i
- ④ 未知元组 X 分类
 - 每个分类器 M_i 返回它的类预测
 - 得票最高的类赋予 X
- ④ 可以用于连续值的预测



Bagging: Bootstrap Aggregation

提高分类准确率的技术 —— 1) 装袋

Algorithm: Bagging. The bagging algorithm—create an ensemble of classification models for a learning scheme where each model gives an equally weighted prediction.

Input:

- D , a set of d training tuples;
- k , the number of models in the ensemble;
- a classification learning scheme (decision tree algorithm, naïve Bayesian, etc.).

Output: The ensemble—a composite model, M^* .

Method:

- (1) **for** $i = 1$ to k **do** // create k models:
- (2) create bootstrap sample, D_i , by sampling D with replacement;
- (3) use D_i and the learning scheme to derive a model, M_i ;
- (4) **endfor**

To use the ensemble to classify a tuple, X :

let each of the k models classify X and return the majority vote;



Boosting 提高分类准确率的技术 —— 2) 提升

- 基本思想：
 - 赋予每个训练元组以权重，迭代地学习 k 个分类器 M_i
 - 更新每个训练元组的权重，再次生成新的分类器
 - 最终根据权重衡量各个子分类器的投票结果，得到最终的类
- Adaboost: 一种流行的提升算法
 - 1) 给予所有训练元组相同的权重 $1/d$
 - 2) 根据权重，有放回的抽样，导出 k 个分类器 M_i
 - 3) 如果元组不正确地分类，则增加它的权重
 - 希望能更关注上一轮“误分类”的元组
 - 模型的错误率（如果元组被误分类，则 $err(X_j)$ 被置为1）
$$error(M_i) = \sum_j^d w_j \times err(X_j)$$
 - 4) 各个子分类器的投票权重为
$$\log \frac{1 - error(M_i)}{error(M_i)}$$
 - 5) 具有最大权重和的类为“赢家”，并返回作为对元组 X 的类预测



Random Forest

提高分类准确率的技术 —— 3) 随机森林

- 想象组合分类器中的每个分类器都是一颗决策树
因此分类器的集合就是一个“森林”
- 分类时，每棵树都投票并返回得票最多的类
- 随机森林可以使用 **装袋** 与 **随机属性选择** 结合来构建
- 与 Adaboost 相比：
 - 对错误和离群点鲁棒性更好



Classification of Class-Imbalanced Data Sets

提高类不平衡数据的分类准确率

- 当感兴趣的主类只有少量元组代表时，会出现 **类不平衡问题**
- 常见策略包括：
 - 过抽样：对正元组重复抽样
 - 欠抽样：随机的从多数（负）类中删去元组
 - 阈值移动：移动阈值 t ，使稀有类的元组容易分类（提高出现几率）
 - 组合方法：以上方法的综合
- 仍需更好的解决方案.....