

Chapter 6. 分类: 基本概念



- 分类: 基本概念
- 决策树归纳
- 贝叶斯分类
- 基于规则的分类

有监督 vs. 无监督学习

■ 有监督学习 (分类)

- 监督：训练数据（观察，测量等）都带有标签，指示观察的类别
- 根据训练集分类新数据

■ 无监督学习 (聚类)

- 训练集的类别（标签）未知
- 给定一个观察，测量等的集合，目标是建立数据中存在的数据的类或簇

预测问题： 分类vs.数值预测

■ 分类

- 预测分类的类标签(离散 or 名义)
- 基于训练数据和类标签 构造一个模型，并分类新数据

■ 数值预测

- 建连续值函数/模型, 预测未知/缺失值

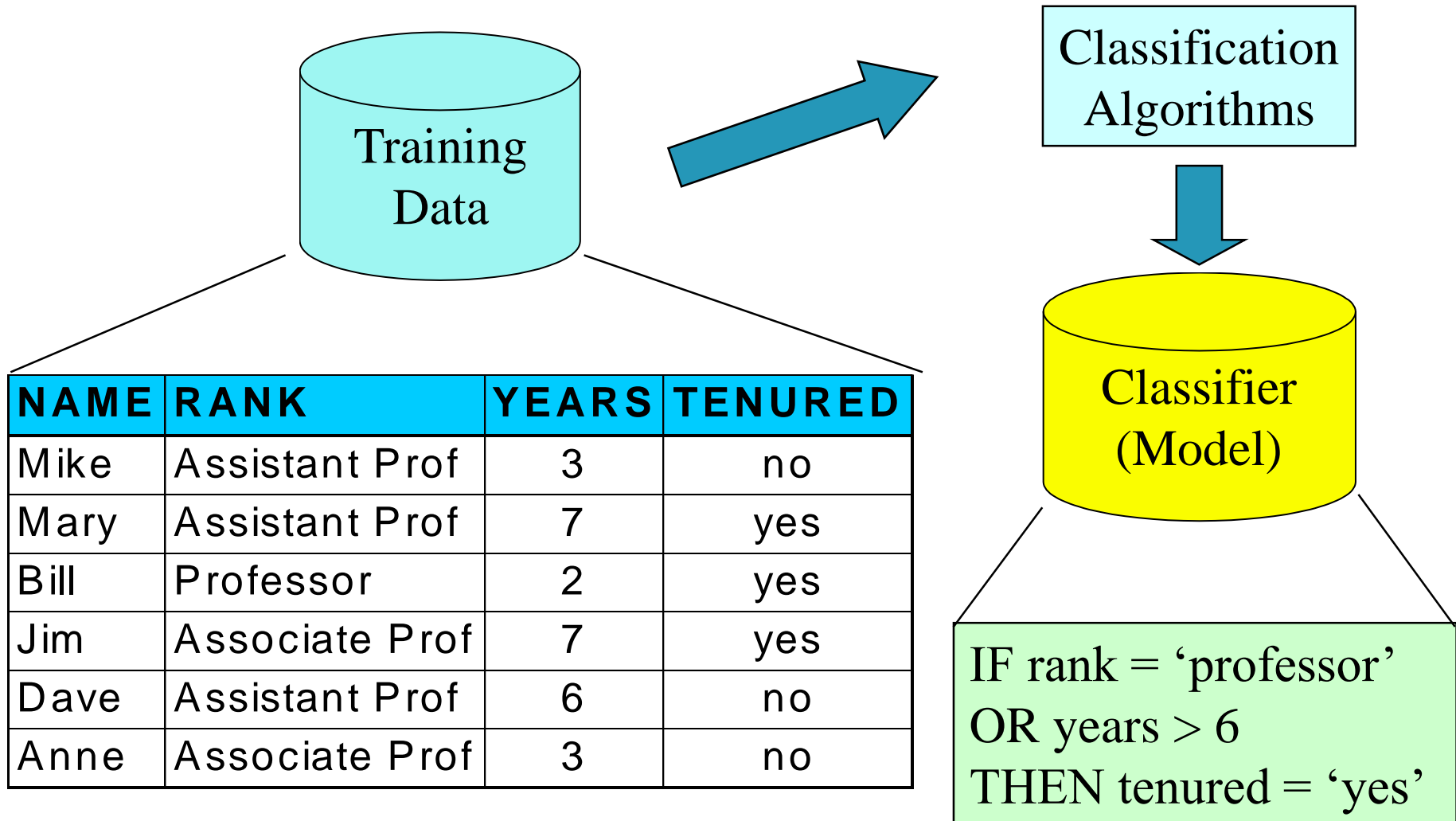
■ 典型应用

- 信用卡/贷款审批:
- 医疗诊断: 肿瘤是癌或良性?
- 欺诈检测: 交易欺诈?
- 网页分类: 这是哪一类?

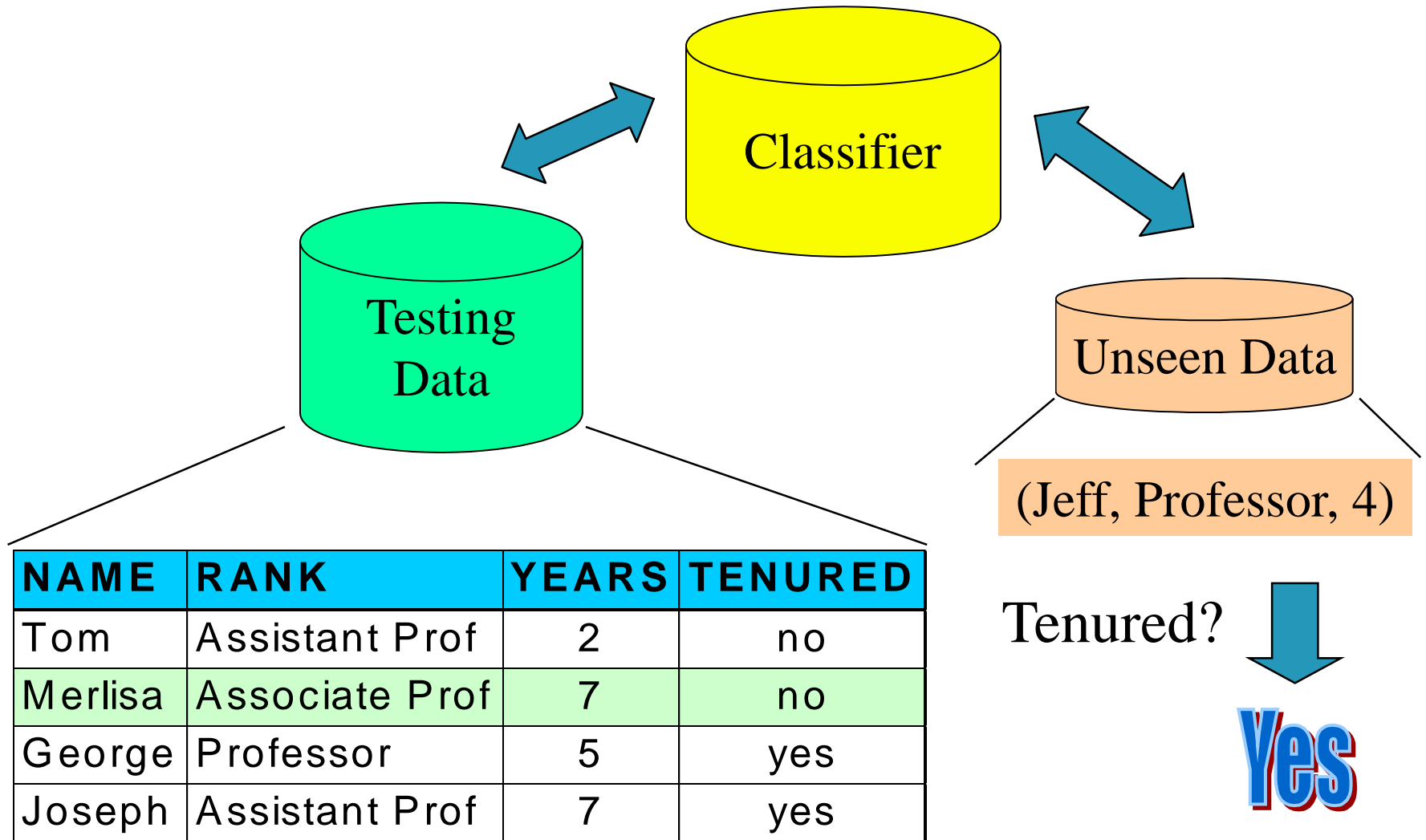
分类: 一个两步的过程

- **模型构建**: 描述一组预先定义的种类
 - 假定每个元组/样本 属于一个类, 由类标签属性设定
 - 用于构建模型的元组集合称为训练集 **training set**
 - 模型可以表示为分类规则, 决策树, 数学公式
- **模型使用**: 分类将来/未知对象
 - **估计模型的准确率**
 - **测试集**: 独立于训练集的样本 (避免过分拟合 **overfitting**)
 - 比较测试样本的已知标签/由模型预测 (得到) 标签
 - **准确率**: 测试样本集中模型正确预测/分类的样本的比率
 - 如果准确率合适, 使用模型来分类标签为未知的样本

Process (1): 模型构建




Process (2): Using the Model in Prediction



评价分类方法 **Evaluating Classification Methods**

- Accuracy
 - classifier accuracy: predicting class label
 - predictor accuracy: guessing value of predicted attributes
- Speed
 - time to construct the model (training time)
 - time to use the model (classification/prediction time)
- Robustness: handling noise and missing values
- Scalability: efficiency in disk-resident databases
- Interpretability
 - understanding and insight provided by the model
- Other measures, e.g., goodness of rules, such as decision tree size or compactness of classification rules

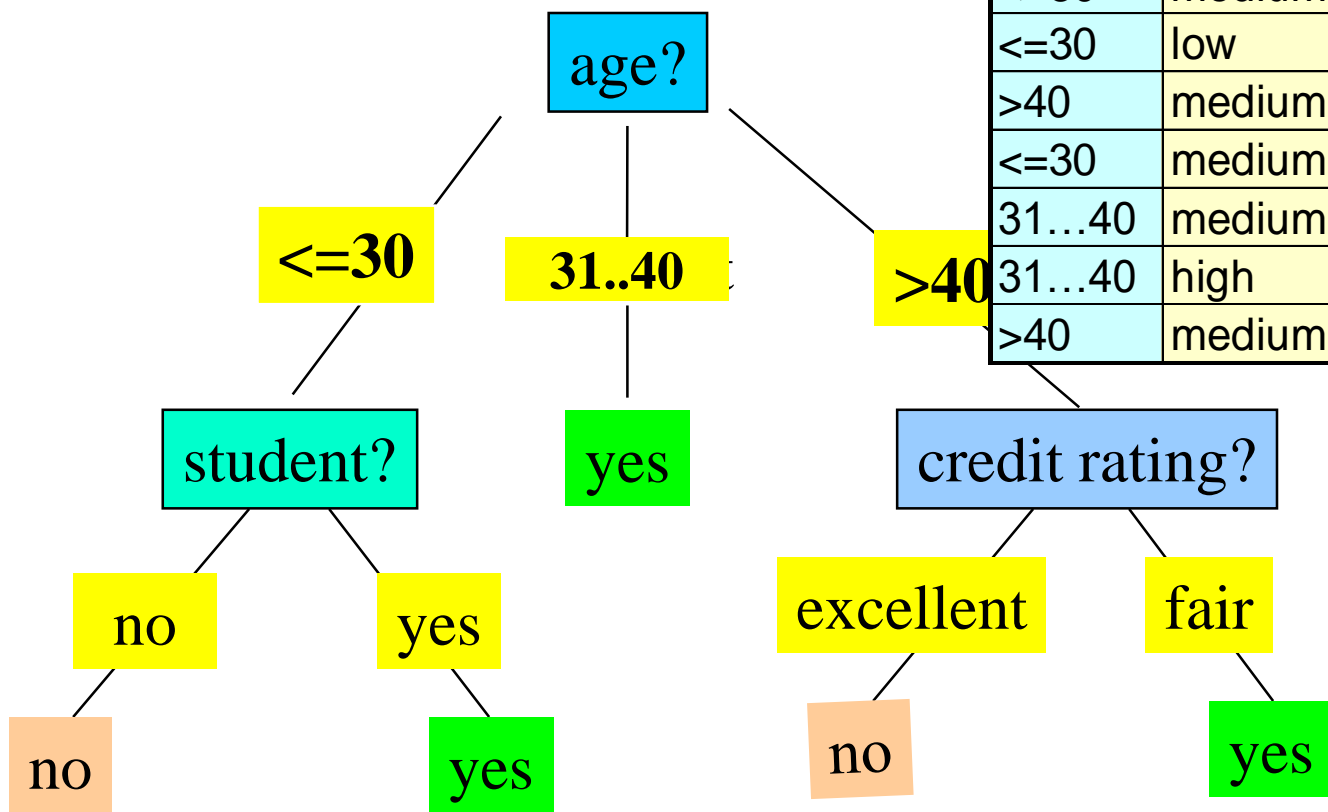
Chapter 6. 分类:决策树归纳

- 分类: 基本概念
- 决策树归纳 
- 贝叶斯分类
- 基于规则的分类
- 提高分类准确率的技术:集成方法Ensemble Methods
- Summary

决策树归纳: 例子

- 训练集: 购买计算机
- 结果:

age	income	student	信誉	购买计算机
<=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



决策树归纳的算法

- 基本算法 (贪心算法)
 - 树构建：自顶向下递归的分治方式
 - 开始，所有的训练样本位于根节点
 - 属性是分类属性(若是连续值,事先离散化)
 - 基于选择的属性，样本被递归地分割
 - 基于启发式/统计量来选择测试属性 (例如 信息增益)
- 终止划分的条件
 - 一个给定节点的所有样本属于一个类别
 - 没有属性剩下可用于进一步划分（运用多数投票标记此节点的类别）
 - 没有样本剩下

输入: 训练集 $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$;

属性集 $A = \{a_1, a_2, \dots, a_d\}$.

过程: 函数 TreeGenerate(D, A)

- 1: 生成结点 node;
- 2: **if** D 中样本全属于同一类别 C **then**
- 3: 将 node 标记为 C 类叶结点; **return**
- 4: **end if**
- 5: **if** $A = \emptyset$ **OR** D 中样本在 A 上取值相同 **then**
- 6: 将 node 标记为叶结点, 其类别标记为 D 中样本数最多的类; **return**
- 7: **end if**
- 8: 从 A 中选择最优划分属性 a_* ;
- 9: **for** a_* 的每一个值 a_*^v **do**
- 10: 为 node 生成一个分支; 令 D_v 表示 D 中在 a_* 上取值为 a_*^v 的样本子集;
- 11: **if** D_v 为空 **then**
- 12: 将分支结点标记为叶结点, 其类别标记为 D 中样本最多的类; **return**
- 13: **else**
- 14: 以 TreeGenerate($D_v, A \setminus \{a_*\}$) 为分支结点
- 15: **end if**
- 16: **end for**

输出: 以 node 为根结点的一棵决策树

属性选择度量

- 属性选择度量
 - 分裂规则-决定给定节点上的元组如何分裂
 - 有最好度量得分的属性为分裂属性
- 三种度量
 - 信息增益、增益率、**Gini**指标
- 数学符号
 - D 为元组的训练集，元组属于 m 个不同的类 $C_i(i=1,...,m)$
 - $C_{i,D}$ 是 D 中的 C_i 类的元组集合
 - $|C_{i,D}|$ 和 $|D|$ 分别表示各自的元组个数

属性选择度量: 信息增益(ID3/C4.5)

- 选择具有最高信息增益的属性
- 令 p_i 为 D 中的任一元组属于类 c_i 概率, 估计为 $|C_{i,D}|/|D|$
- 分类 D 中元组需要的期望信息(entropy):

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

- (利用 A 分裂 D 为 v 个部分后)分类 D 需要的信息为:

$$Info_A(D) = \sum_{j=1}^v \frac{|D_j|}{|D|} \times Info(D_j)$$

- 以属性 A 分枝得到的信息增益

$$Gain(A) = Info(D) - Info_A(D)$$

属性选择: 信息增益

■ Class P: 买电脑 = “yes”

■ Class N: 买电脑 = “no”

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

age	income	student	credit_rating	s_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

$$\begin{aligned} Info_{age}(D) &= \frac{5}{14} \times \left(-\frac{2}{5} \log_2 \frac{2}{5} - \frac{3}{5} \log_2 \frac{3}{5} \right) \\ &\quad + \frac{4}{14} \times \left(-\frac{4}{4} \log_2 \frac{4}{4} - \frac{0}{4} \log_2 \frac{0}{4} \right) \\ &\quad + \frac{5}{14} \times \left(-\frac{3}{5} \log_2 \frac{3}{5} - \frac{2}{5} \log_2 \frac{2}{5} \right) \\ &= 0.694 \text{ bits.} \end{aligned}$$

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

$$Gain(income) = 0.029$$

$$Gain(student) = 0.151$$

$$Gain(credit_rating) = 0.048$$

计算信息增益-连续值属性

- 令 A 为连续属性
- 必须为A确定一个最佳分裂点 *best split point*
 - 上升序排序 A
 - 典型地, 每对相邻值的中点是一个可能的分裂点
 - $(a_i + a_{i+1})/2$ is the midpoint between the values of a_i and a_{i+1}
 - 具有最小期望信息需求的点选为A的分裂点
- Split:
 - D1 为D中元组满足 $A \leq \text{split-point}$, D2 是元组满足 $A > \text{split-point}$

增益率 (C4.5)

- 信息增益倾向于有大量不同取值的属性（划分更细更纯）
 - 极端：每个划分子集只有一个样本，即一个类
 - 此时 $\text{Info}(d)=0$
- C4.5 (ID3 后继) 使用增益率来克服这一问题(规范化信息增益)

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

- $\text{GainRatio}(A) = \text{Gain}(A) / \text{SplitInfo}(A)$
- Ex.
$$\text{SplitInfo}_{\text{income}}(D) = -\frac{4}{14} \times \log_2 \left(\frac{4}{14} \right) - \frac{6}{14} \times \log_2 \left(\frac{6}{14} \right) - \frac{4}{14} \times \log_2 \left(\frac{4}{14} \right) = 1.557$$
 - $\text{gain_ratio}(\text{income}) = 0.029 / 1.557 = 0.019$
- 具有最大增益率的属性选为分裂属性

Gini Index基尼指数

- 数据 D 包含 n 个类的样本, gini指标, $gini(D)$ 定义为

p_j 类别 j 在 D 中的频率

$$gini(D) = 1 - \sum_{j=1}^n p_j^2$$

- 数据集 D 基于属性 A 分裂为子集 D_1 和 D_2 , gini 指标定义为

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

- 不纯度减少: $\Delta gini(A) = gini(D) - gini_A(D)$
- 具有最小 $gini_{split}(D)$ 的属性(不纯度减少最大的)用于分裂节点
(如果属性 A 为名词性, 需要枚举所有可能的分裂情况)

计算 Gini Index 指标

- D 有 9 个元组买电脑 = “yes” / 5 个买电脑 = “no”

$$gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$$

- 设属性 income 分裂 D 为包含 10 个元组的 D_1 : {low, medium} / 4 个元组的 D_2

$$\begin{aligned} Gini_{income \in \{low, medium\}}(D) &= \frac{10}{14} Gini(D_1) + \frac{4}{14} Gini(D_2) \\ &= \frac{10}{14} \left(1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2 \right) + \frac{4}{14} \left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2 \right) \\ &= 0.443 \\ &= Gini_{income \in \{high\}}(D). \end{aligned}$$

$Gini_{\{low, high\}} = 0.458$; $Gini_{\{medium, high\}} = 0.450$. 因此 {low, medium} / {high} 分裂, 由于其有最小的 Gini index

- 假设所有属性都是连续值, 需要其他技术, e.g., 聚类, 来获得可能的分裂点

比较属性选择度量

- 通常三种度量获得较好的结果
 - 信息增益**Information gain**:
 - 偏向于多值属性
 - 增益率**Gain ratio**:
 - 倾向于不平衡的分裂，其中一个子集比其他小得多
 - **Gini index**:
 - 偏向于多值属性
 - 当类数目较大时，计算困难
 - 倾向于导致大小相等的分区和纯度

其他属性选择度量

- CHAID: 一种流行的决策树算法, 基于独立 χ^2 检验的选择度量
- C-SEP: 某些情况下比信息增益gini指标更好
- G-statistic: 非常近似于 χ^2 分布
- MDL (最小描述长度) (i.e., 首选最简单的解):
 - 最佳树为需要最小二进位的树 (1) 编码树, (2) 编码树的异常
- 多元划分 (基于多变量组合来划分)
 - CART: 基于属性的线性组合来发现多元划分
- 哪一个是最好的?
 - 大部分可以获得较好结果, 没有一个显著地优于其他


过拟合与树剪枝

- 过拟合Overfitting: 一棵归纳的树 可能过分拟合训练数据
 - 分枝太多,某些反映训练数据中的异常, 噪音/孤立点
 - 对未参与训练的样本的低精度预测
- 两种处理方法
 - 先剪枝: 提前终止树构造
 - 如果对一个节点的分裂会产生低于给定的阈值的度量, 划分停止
 - 选择一个合适的阈值很难
 - 后剪枝: 从完全生长的树中剪去树枝—得到一个逐步修剪树
 - 例如, 最小化代价复杂度 (树节点个数和错误率的函数)
 - 使用不同于训练集的数据来确定哪一个是 “**best pruned tree**”

决策树归纳的增强

- 允许连续值属性
 - 动态地定义新的离散值属性，其把连续值属性分成离散的区间
- 处理缺失属性值
 - 分配属性的最常见值
 - 为每一个可能的值分配概率
- 属性构造
 - 基于现有的稀少出现的属性创建新的属性，
 - 这减少了分散，重复和复制

Chapter 6. 分类: 贝叶斯分类

- 分类: 基本概念
- 决策树归纳
- 贝叶斯分类 
- 基于规则的分类

贝叶斯理论

- X 为数据样本: 类标签未知
- H 为一个假设: 样本 X 属于类别 C
- 分类就是确定 $P(H|X)$ (后验概率),
 - 给定观察数据 X 后, H 成立的概率
- $P(H)$ (先验概率)——最初的概率
 - 例, 不管年龄和收入等条件 X 将会购买计算机
- $P(X)$: 样本数据 x 被观察到的概率
- $P(X|H)$ (可能性),
 - 假设 H 成立, 那么观测到样本 X 的概率
 - E.g., 已知 X 购买计算机, X 为 31..40 且中等收入的概率

贝叶斯理论 **Bayesian Theorem**

- 给定训练数据 \mathbf{x} , 假设 H 的后验概率 $P(H|\mathbf{X})$ 满足贝叶斯理论

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

- 通俗地说, 这可以写成

posteriori = likelihood * prior/evidence

- 预测 \mathbf{x} 属于类别 C_2 当且仅当概率 $P(C_2|\mathbf{X})$ 是所有 $P(C_k|\mathbf{X})$ ($1 \leq k \leq L$) 最大
- 实际计算困难: 需要许多可能性的初步知识, 计算成本显著

Naïve Bayesian Classifier

- **D**为训练数据集（包含类别标签），并且每个元组表示为一个n-维的属性向量 $\mathbf{X} = (x_1, x_2, \dots, x_n)$
- 假定有 m 个类别 C_1, C_2, \dots, C_m .
- 分类就是推导最大的后验概率, i.e., the maximal $P(C_i | \mathbf{X})$
- 可以由贝叶斯理论计算
$$P(C_i | \mathbf{X}) = \frac{P(\mathbf{X} | C_i)P(C_i)}{P(\mathbf{X})}$$
- 由于对所有类 $P(\mathbf{X})$ 是常量，只需要最大化

$$P(C_i | \mathbf{X}) = P(\mathbf{X} | C_i)P(C_i)$$

朴素贝叶斯分类器的推导

- 一个简单假定: 属性是条件独立的 (i.e., 属性间没有依赖关系):

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

- 这样极大地减少了计算代价: 只需要统计类的分布
- 若 \mathbf{A}_k 是分类属性
 - $P(x_k | C_i) = C_i$ 类中 \mathbf{A}_k 取值为 x_k 的元组数 / $|C_i|$ (类 C_i 的大小)
- 若 \mathbf{A}_k 是连续值, $P(x_k | C_i)$ 通常基于样本的均值 μ 和标准差 σ 的高斯分布计算

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- $P(x_k | C_i)$

$$P(X_k | C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i})$$

朴素贝叶斯分类: 训练数据集

两个类别:

C1:buys_computer = 'yes'

C2:buys_computer = 'no'

数据样本

X = (age <=30,

Income = medium,

Student = yes

Credit_rating = Fair)

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

Naïve Bayesian Classifier: 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
 - $P(\text{age} = \text{"<=30"} \mid \text{buys_computer} = \text{"yes"}) = 2/9 = 0.222$
 - $P(\text{age} = \text{"<= 30"} \mid \text{buys_computer} = \text{"no"}) = 3/5 = 0.6$
 - $P(\text{income} = \text{"medium"} \mid \text{buys_computer} = \text{"yes"}) = 4/9 = 0.444$
 - $P(\text{income} = \text{"medium"} \mid \text{buys_computer} = \text{"no"}) = 2/5 = 0.4$
 - $P(\text{student} = \text{"yes"} \mid \text{buys_computer} = \text{"yes"}) = 6/9 = 0.667$
 - $P(\text{student} = \text{"yes"} \mid \text{buys_computer} = \text{"no"}) = 1/5 = 0.2$
 - $P(\text{credit_rating} = \text{"fair"} \mid \text{buys_computer} = \text{"yes"}) = 6/9 = 0.667$
 - $P(\text{credit_rating} = \text{"fair"} \mid \text{buys_computer} = \text{"no"}) = 2/5 = 0.4$
 - **$X = (\text{age} \leq 30, \text{income} = \text{medium}, \text{student} = \text{yes}, \text{credit_rating} = \text{fair})$**
 - $P(X|C_i)$** : $P(X \mid \text{buys_computer} = \text{"yes"}) = 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044$
 $P(X \mid \text{buys_computer} = \text{"no"}) = 0.6 \times 0.4 \times 0.2 \times 0.4 = 0.019$
 - $P(X|C_i) \cdot P(C_i)$** : $P(X \mid \text{buys_computer} = \text{"yes"}) \cdot P(\text{buys_computer} = \text{"yes"}) = 0.028$
 $P(X \mid \text{buys_computer} = \text{"no"}) \cdot P(\text{buys_computer} = \text{"no"}) = 0.007$
- Therefore, X belongs to class ("buys_computer = yes")**

贝叶斯分类: Why?

- 基于统计学分类器: 执行概率预测, *i.e.*, 预测类成员的概率
- 基础: 基于贝叶斯理论
- 性能表现: 一个简单的贝叶斯分类器（朴素贝叶斯分类器）可以与决策树和经过挑选的神经网络分类器相媲美
- 增量: 每次训练的样本可以逐步增加/减少一个, 假设是正确的可能性——先验知识可与观测数据相结合
- Standard: 即使贝叶斯方法是难以计算的, 为制定最优决策提供标准（其他方法可以衡量）

避免零概率问题

- 朴素贝叶斯要求每个条件概率非零. 然而, 预测的概率可能为零


$$P(X | C_i) = \prod_{k=1}^n P(x_k | C_i)$$

- Ex. 假定有1000 元组, income=low (0), income= medium (990), and income = high (10)
- 使用**Laplacian correction**校准 (or Laplacian estimator估计法)
 - 为每个类别增加 1 例样本
$$\text{Prob}(\text{income} = \text{low}) = 1/1003$$
$$\text{Prob}(\text{income} = \text{medium}) = 991/1003$$
$$\text{Prob}(\text{income} = \text{high}) = 11/1003$$
 - 校准的 “corrected” 概率估计很接近未校准的

Naïve Bayesian Classifier:评论

- 优势Advantages
 - Easy to implement
 - Good results obtained in most of the cases
- 缺陷Disadvantages
 - Assumption: 类条件独立性, 损失精度
 - 实际中, 变量间存在依赖
 - E.g.,医院: 患者: 简介: 年龄, 家族病史等
症状: 发烧, 咳嗽等疾病: 肺癌, 糖尿病等
 - Dependencies among these cannot be modeled by Naïve Bayesian Classifier
- How to deal with these dependencies? Bayesian Belief Networks

Chapter 6. 分类:基于规则的分类

- 分类: 基本概念
- 决策树归纳
- 贝叶斯分类
- 基于规则的分类 

使用IF-THEN 规则分类

■ 使用 IF-THEN 规则表示知识

R: IF *age* = youth AND *student* = yes THEN *buys_computer* = yes

- 规则前件(前提) vs. 规则结论

■ 评估规则: 覆盖率 $coverage$ 和 准确率 $accuracy$

- n_{covers} 表示规则R覆盖的元组数 (给定元组, 规则的前提满足的元组)
- $n_{correct}$ 表示 R正确分类的元组数

$$coverage(R) = n_{covers} / |D| \quad (D: \text{是训练数据集})$$

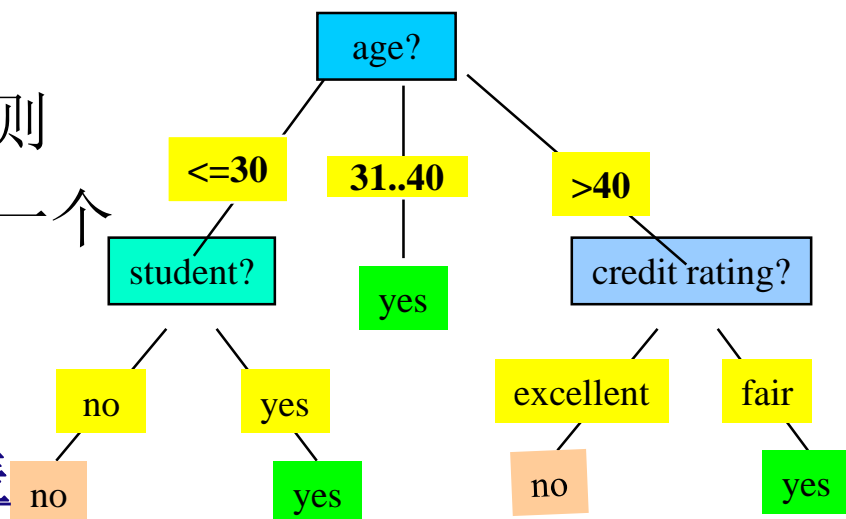
$$accuracy(R) = n_{correct} / n_{covers}$$

■ 如果超过1条规则被触发,需要解决冲突

- 规模序Size ordering: 最高优先权赋予 “最苛刻” 的规则(即, 最多属性测试)
- 基于类的序: 每个类的错误分类代价的下降序
- 基于规则的序(决策表): 根据一些规则的质量度量或由专家建议, 规则被组织成一个长的优先级列表

从决策树提取规则

- 规则比一棵大的决策树更容易理解
- 从根到每个叶子的路径产生一个规则
- 沿路径的每个属性值对一起形成了一个联合: 叶节点形成规则后件
- 规则是互斥的和穷举的
 - 没有冲突规则, 每个元组被覆盖



- Example: Rule extraction from our *buys_computer* decision-tree

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

规则归纳-顺序覆盖算法

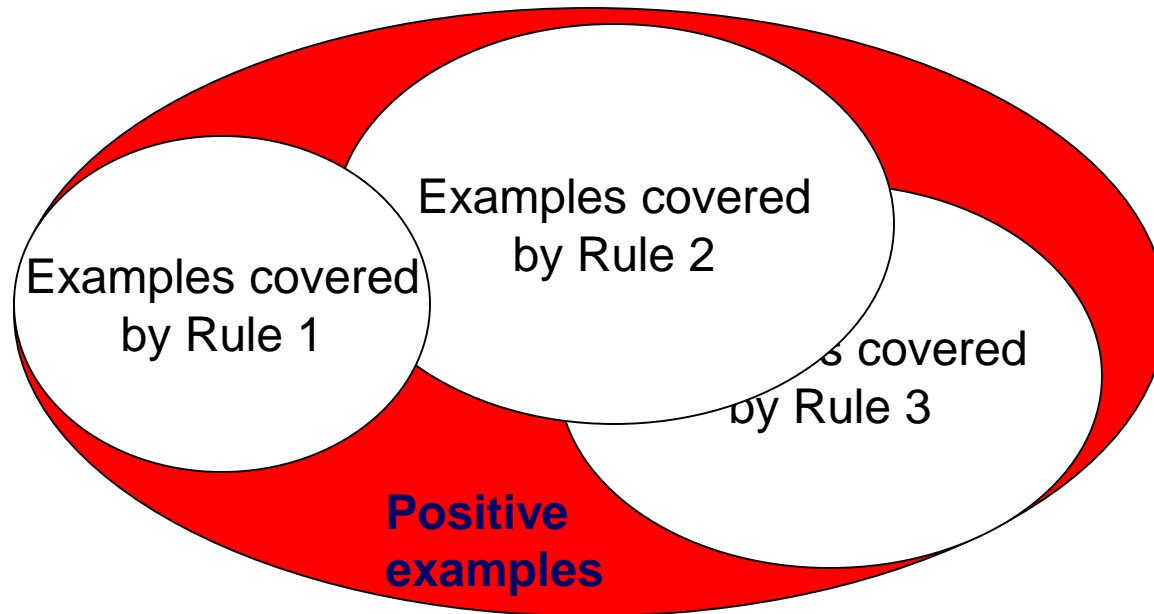
- 顺序覆盖算法：直接从训练数据抽取规则
- 典型的算法：FOIL, AQ, CN2, RIPPER
- 规则被顺序地学习，类 C_i 的规则将尽量覆盖 C_i 的样本，少覆盖或不覆盖其他类别的样本
- Steps:
 - 一次学习一个规则
 - 每学习一个规则，删除此规则覆盖的样本
 - 对剩下的样本重复该过程直到终止条件，e. g., 没有训练样本/返回的规则的质量低于用户给定的阈值
- 与决策树对照：同时学习一组规则

顺序覆盖算法

while (enough target tuples left)

产生一个规则

删除这个规则覆盖的元组



Rule Generation

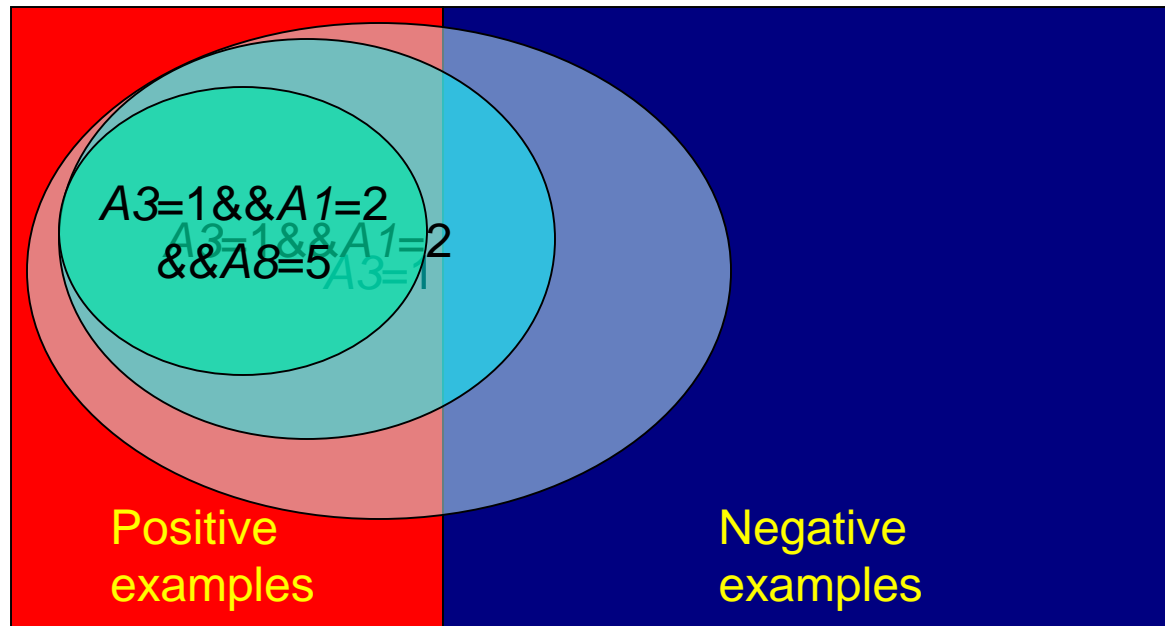
- To generate a rule

while(true)

找到最好的谓词 p

if 规则质量度量(p) > threshold **then** add p to current rule

else break



形式化的算法

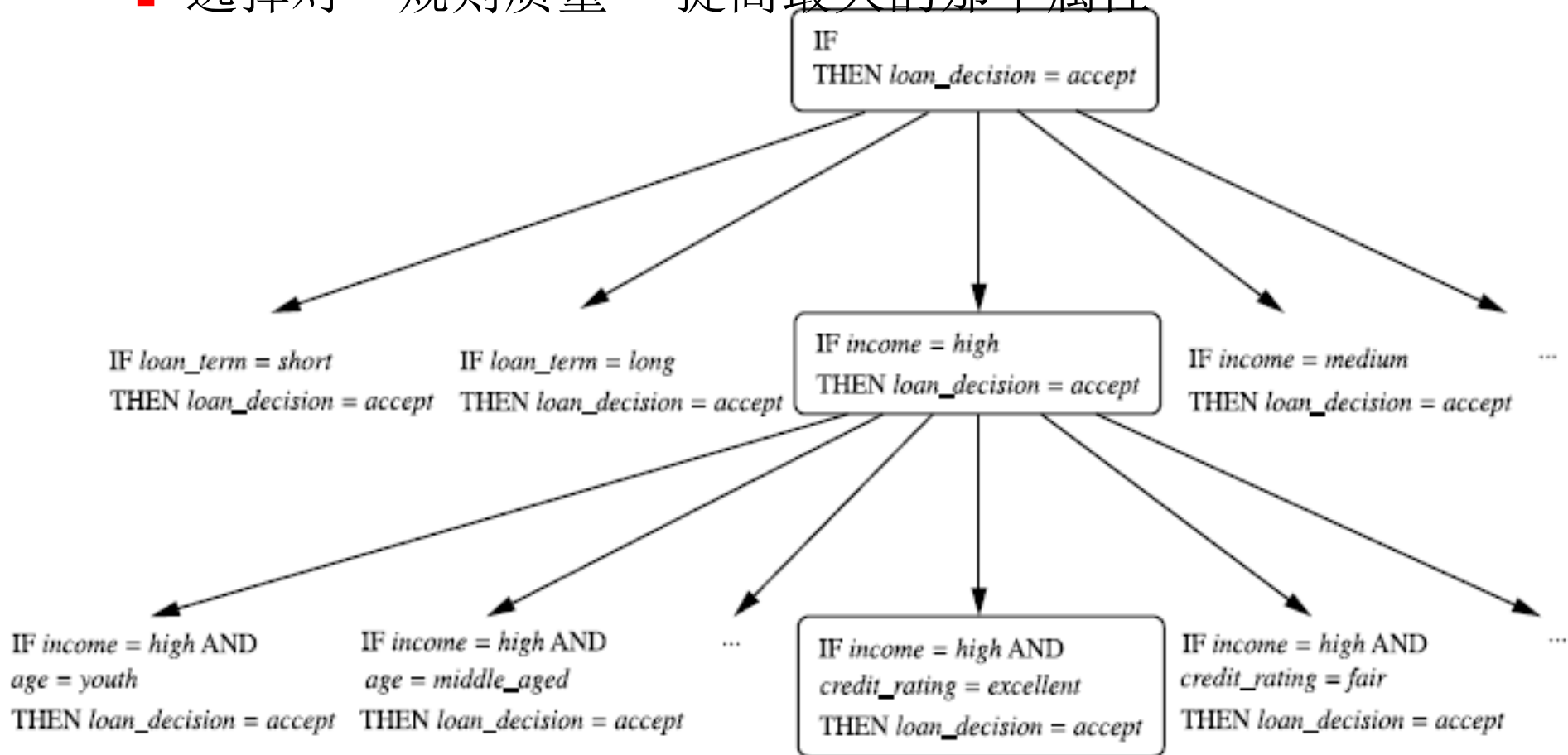
输出：IF-THEN规则的集合。

方法：

- (1) $Rule_set = \{\}$; // 学习的规则的初始集为空
- (2) **for** 每个类 c **do**
- (3) **repeat**
- (4) $Rule = \mathbf{Learn_One_Rule} (D, Att_vals, c) ;$
- (5) 从 D 中删除 $Rule$ 覆盖的元组;
- (6) **until** 终止条件满足;
- (7) $Rule_set = Rule_set + Rule;$ // 将新规则添加到规则集
- (8) **endfor**
- (9) 返回 $Rule_Set$;

如何学习一个规则?

- 从可能的最一般的规则开始: `condition = empty`
- 采用贪心的深度优先策略添加新属性到规则中
 - 选择对“规则质量”提高最大的那个属性



A general-to-specific search through rule space.

规则质量度量与剪枝

- 规则质量度量: 同时考虑 覆盖率和准确率
 - Foil-gain (FOIL & RIPPER): 评价扩展 “前提” 获得的信息(变大更好)

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

- 用于学习规则的类的元组—**正元组**; 其余为**负元组**
- 偏向于具有高准确率并覆盖许多正元组的规则
- Pos/neg: 规则R覆盖的正元组数/负元组数
- Pos'(neg'): 规则R'覆盖的正元组数/负元组数;
- 基于一个独立的测试集进行规则剪枝（即删除一个属性），这个值将随着规则R在测试集上的准确率的增加而增加。
 - Pos/neg 是被规则R覆盖的正/负元组。

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

如果规则R 剪枝后的 $FOIL_Prune$ 较高, 那么剪枝R