第十次作业

- 1、Apriori的算法步骤：
- 　　1-1 找所有的候选k项集（频繁k项集是由频繁k-1项集组成的）
- 　　1-2 得到所有的候选k项集的支持度或可信度
- 　　1-3 根据最小支持度或可信度阈值进行频繁k项集筛选
- 2、构造频繁项集的TDB数据库，设置一个支持度阈值，通过ppt上第21页的例子，自己手动完成频繁多项集的筛选和检索。
- 3、找一个关联规则的数据集，通过Apriori案例的搜索，找到一些关联规则。根据我们上课的知识把实现代码读懂。
- 4、分类算法的基本流程是什么。决策树ID3算的流程，朴素贝叶斯的流程。
- 5、分类算法的评价指标是什么？精度率，回归率。
- 6、找一个分类的数据集，通过某种分类算法的Java实现进行分类。

Chapter 7. 分类: 基本概念

Chapter 7. 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
- 模型评价与选择
- 提高分类准确率的技术:集成方法Ensemble Methods
- Summary

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连续的)
  - 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
  - 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

步骤(1): 模型构建

训练集Training Data

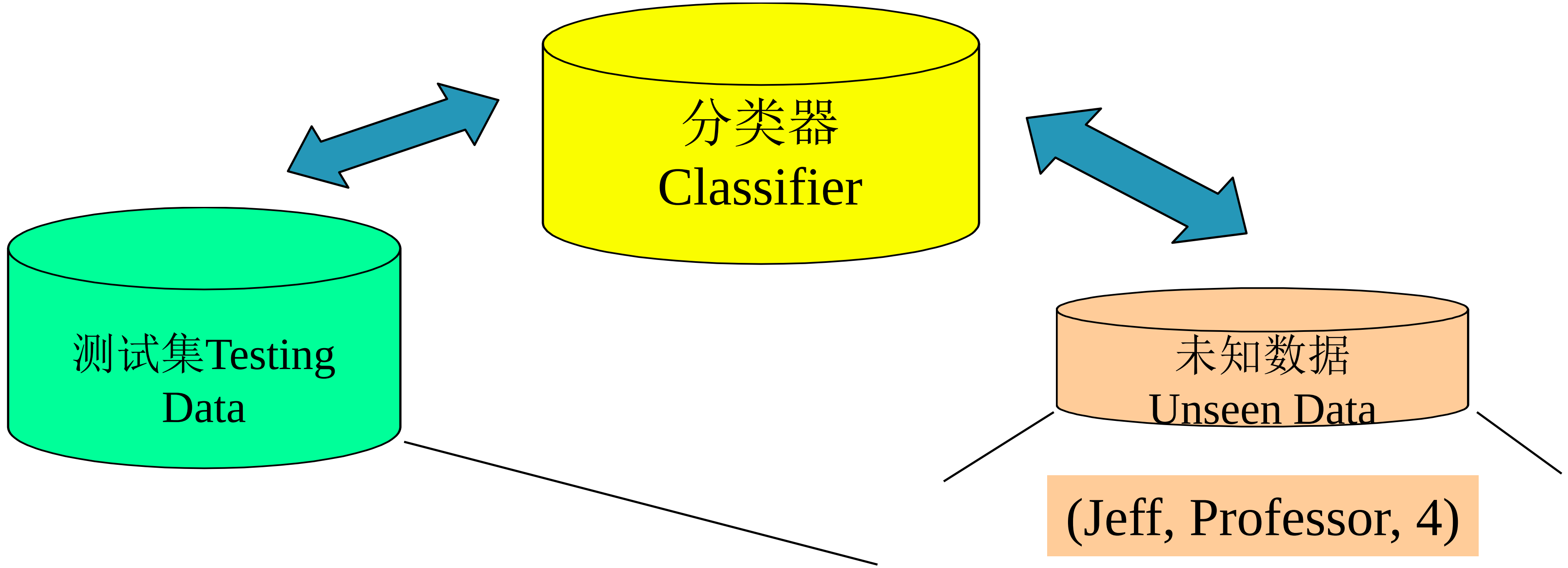
NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no
Wu	Assistant Prof	**3**	**?**

分类算法
Classification Algorithms

分类器
Classifier
(Model)

IF rank = 'professor'
OR years > 6
THEN tenured = 'yes'

步骤 (2): 使用模型进行预测



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

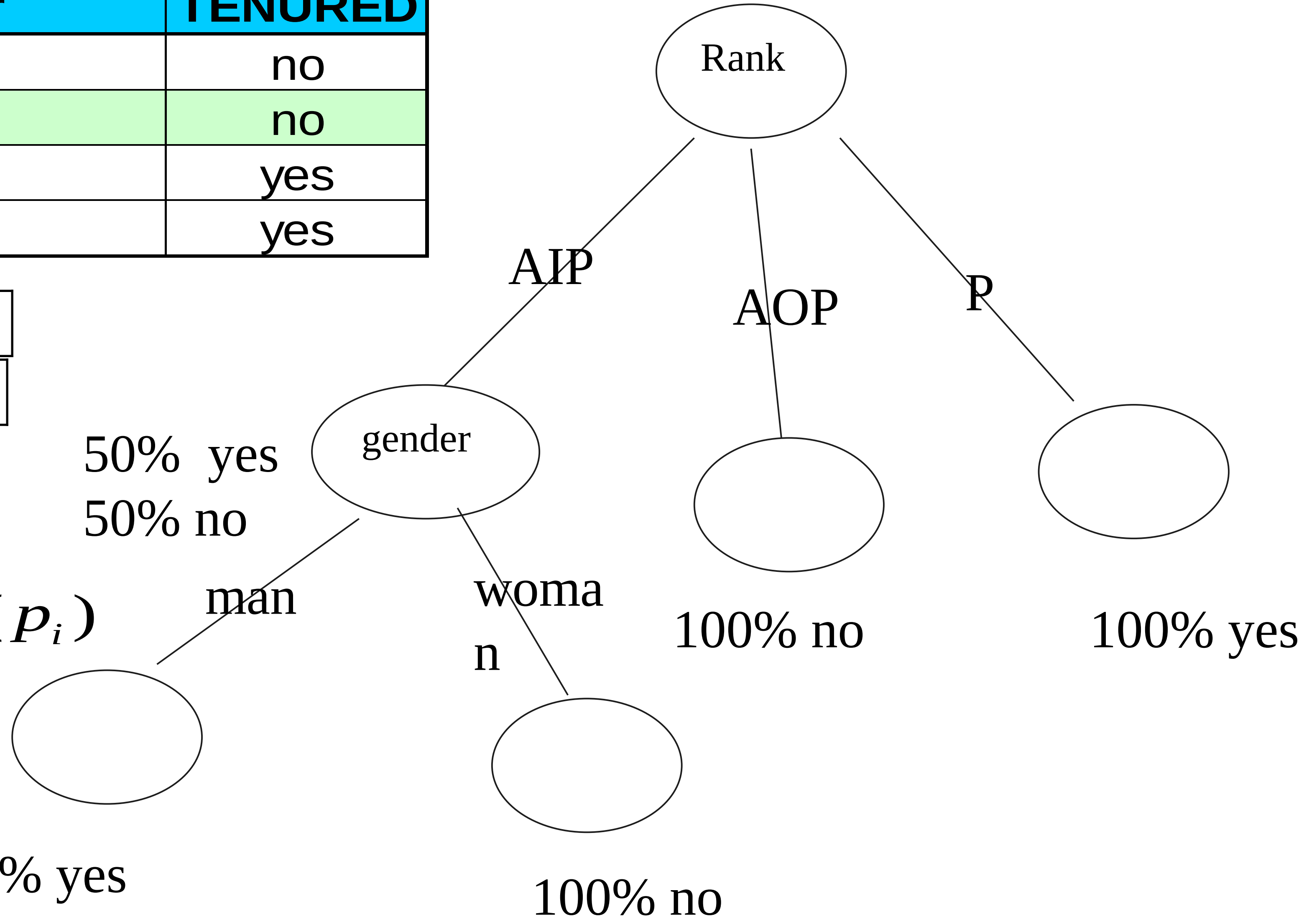
Tenured?

Yes

NAME	RANK	gender	TENURED
Tom	Assistant Prof	woman	no
Merlisa	Associate Prof	woman	no
George	Professor	man	yes
Joseph	Assistant Prof	man	yes

Wu	Professor	**man**	**?**
Li	Assistant Prof	**man**	**?**

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



Chapter 7. 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
- 模型评价与选择
- 提高分类准确率的技术:集成方法Ensemble Methods
- Summary

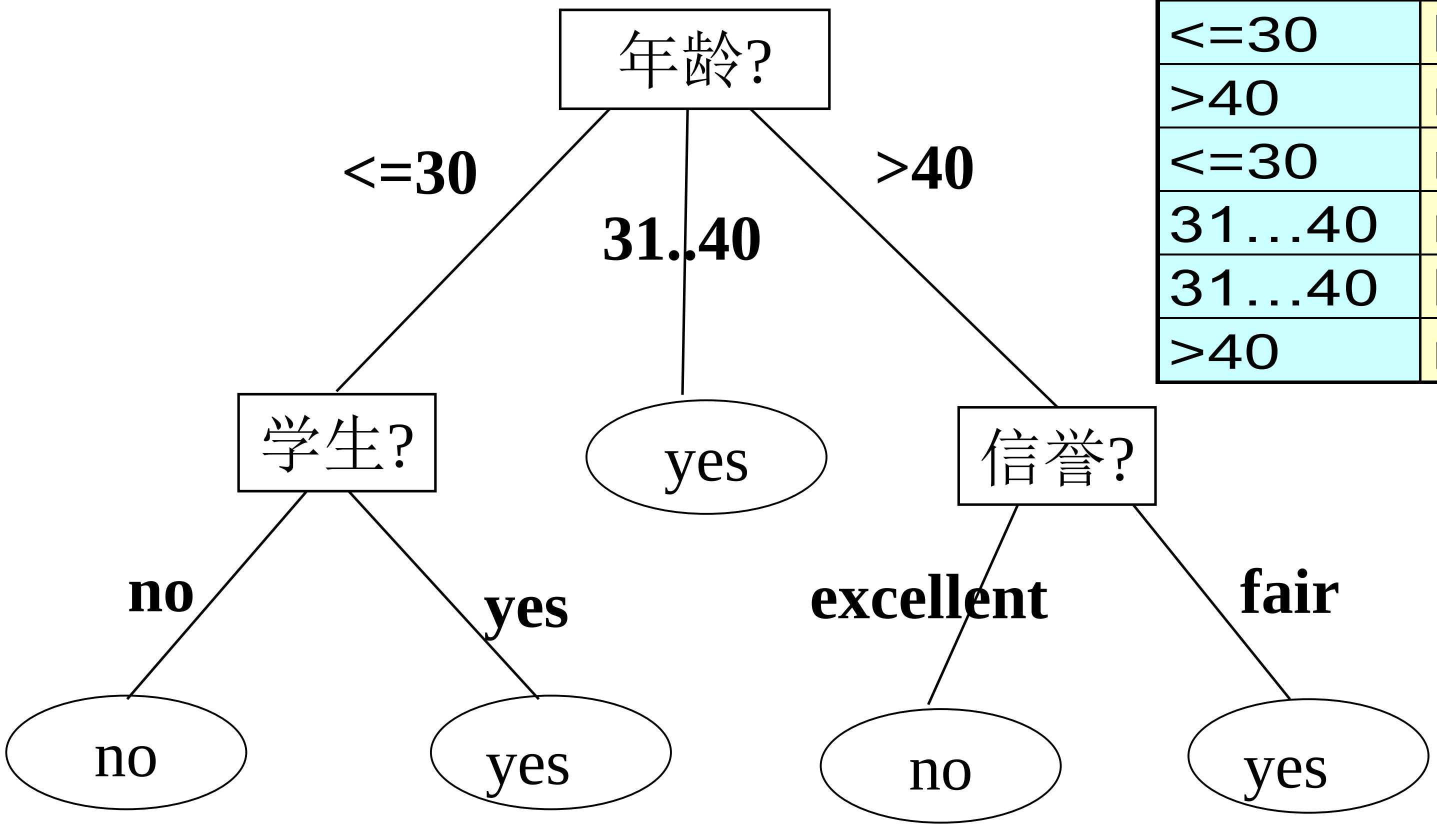
决策树归纳分类

- 什么是决策树？
  - 类似于流程图的树结构
  - 每个内部节点表示在一个属性上的划分
  - 每个分枝代表一个确定的属性值
  - 每个树叶节点代表类或类分布
- 决策树的生成由两个阶段组成
  - 决策树构建
    - 开始时，所有的训练样本都在根节点
    - 递归地通过选定的属性，来划分样本（必须是离散值）
  - 树剪枝
    - 许多分枝反映的是训练数据中的噪声和孤立点，树剪枝试图检测和减去这种分枝
- 决策树的使用：对未知样本进行分类
  - 通过将样本的属性值与决策树相比较

决策树归纳: 例子

- 训练集: 购买PC
- 结果:

年龄	收入	学生	信誉	购买PC
<=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：自顶向下递归地分治方式
  - 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/统计测量来选择测试属性 (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，用于进一步划分 ——运用多数投票来标记此节点
  - 没有剩余的样本

决策树算法

- 算法：Generate_decision_tree(samples, attribute)。由给定的训练数据产生一棵决策树。
- 输入：训练样本samples，由离散值属性表示；候选属性的集合attribute_list。
- 输出：一棵决策树。
- 方法：
- Generate_decision_tree(samples, attribute_list)
- 创建一个结点N;
- (1)if D中的元组都在同一类C中 then
- (2)　返回N作为叶结点，以类C标记;
- (3)if attribute_list为空 then
- (4)　返回N作为叶结点，标记为D中的多数类;

决策树算法

- (5)使用Attribute_selection_method(D, attribute_list)，找出“最好地”splitting_criterion; // 分裂准则
- (6)用splitting_criterion标记结点N;
- (7) if splitting_attribute是离散值的，并且允许多路划分 then
- (8)   attribute_list = attribute_list - splitting_attribute;
- (9) for splitting_criterion的每个输出j
- (10)   设D(j)是D中满足输出j的数据元组的集合;
- (11)   if D(j)为空 then
- (12)   加一个树叶到结点N，标记为D中的多数类;
- (13)   else
- (14)   加一个由Generate_decision_tree(D, attribute_list)返回的结点到N;
- (15) endfor
- (16)返回N;

决策树算法

- **(6) 选择attribute_list 中具有最高信息增益的属性best_attribute；**
  - //找出最好的划分属性
- **(7) 标记结点 N 为best_attribute；**
- **(8) for each best_attribute 中的未知值a i**

  //将样本samples按照best_attribute进行划分
- **(9) 由结点 N 长出一个条件为 best_attribute = a i 的分枝；**
- **(10) 设si 是samples 中best_attribute = a i 的样本的集合；//一个划分**
- **(11) if si 为空 then**
- **(12) 加上一个树叶，标记为 samples 中最普通的类；**

  //从样本中找出类标号数量最多的，作为此节点的标记
- **(13) else 加上一个由 Generate_decision_tree(si, attribute_list–best_attribute)返回的结点；**

  //对数据子集si,递归调用，此时候选属性已删除best_attribute

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
  - 具有最好度量得分的属性选定为分裂属性
- 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"

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

$$Info_{age}(D) = \frac{5}{14} \times (-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}) + \frac{4}{14} \times (-\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4}) + \frac{5}{14} \times (-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}) = 0.694 \text{ bits.}$$

age	income	student	redit_ratin	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

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

$$Gain(income) = 0.029$$

$$Gain(student) = 0.151$$

$$Gain(credit\_rating) = 0.048$$

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 ≤ split-point, D2 是元组满足 A > 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)

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

  - GainRatio(A) = Gain(A)/SplitInfo(A)
- Ex.

$$SplitInfo_{income}(D) = -\frac{4}{14} \times \log_2\left(\frac{4}{14}\right) - \frac{6}{14} \times \log_2\left(\frac{6}{14}\right) - \frac{4}{14} \times \log_2\left(\frac{4}{14}\right) = 1.557$$

  - gain_ratio(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指标 (CART)

- 数据 *D* 包含*n* 类别的样本, 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(or不纯度减少最大的) 用于分裂节点 (*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

过拟合与数剪枝

- 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

大型数据库中分类

- 分类—被统计学和机器学习研究人员广泛地研究一个经典问题
- 可伸缩性:以合理的速度分类由带有数百个属性的百万个样本组成的数据集
- 为什么决策树归纳受欢迎?
  - 相对快的训练速度 (与其他分类方法相比)
  - 转换为简单、易于理解的分类规则
  - 可用 SQL查询来访问数据库
  - 与其它方法可比的分类精度
- **RainForest (VLDB’98 — Gehrke, Ramakrishnan & Ganti)**
  - **Builds an AVC-list (attribute, value, class label)**

RainForest雨林的可扩展性框架

- 可扩展性和确定质量树的标准相分离
- 建并维持 AVC-list: AVC (属性-值, 类标号)
- AVC集 (of an attribute *X*)
  - 把训练集投影到属性*X*和类标签上，给出属性*X的每个值上的类标签计数*
- AVC组群 (在节点*n*)
  - 节点n上所有预测属性的AVC集合----组群

Rainforest：训练集和AVC集

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

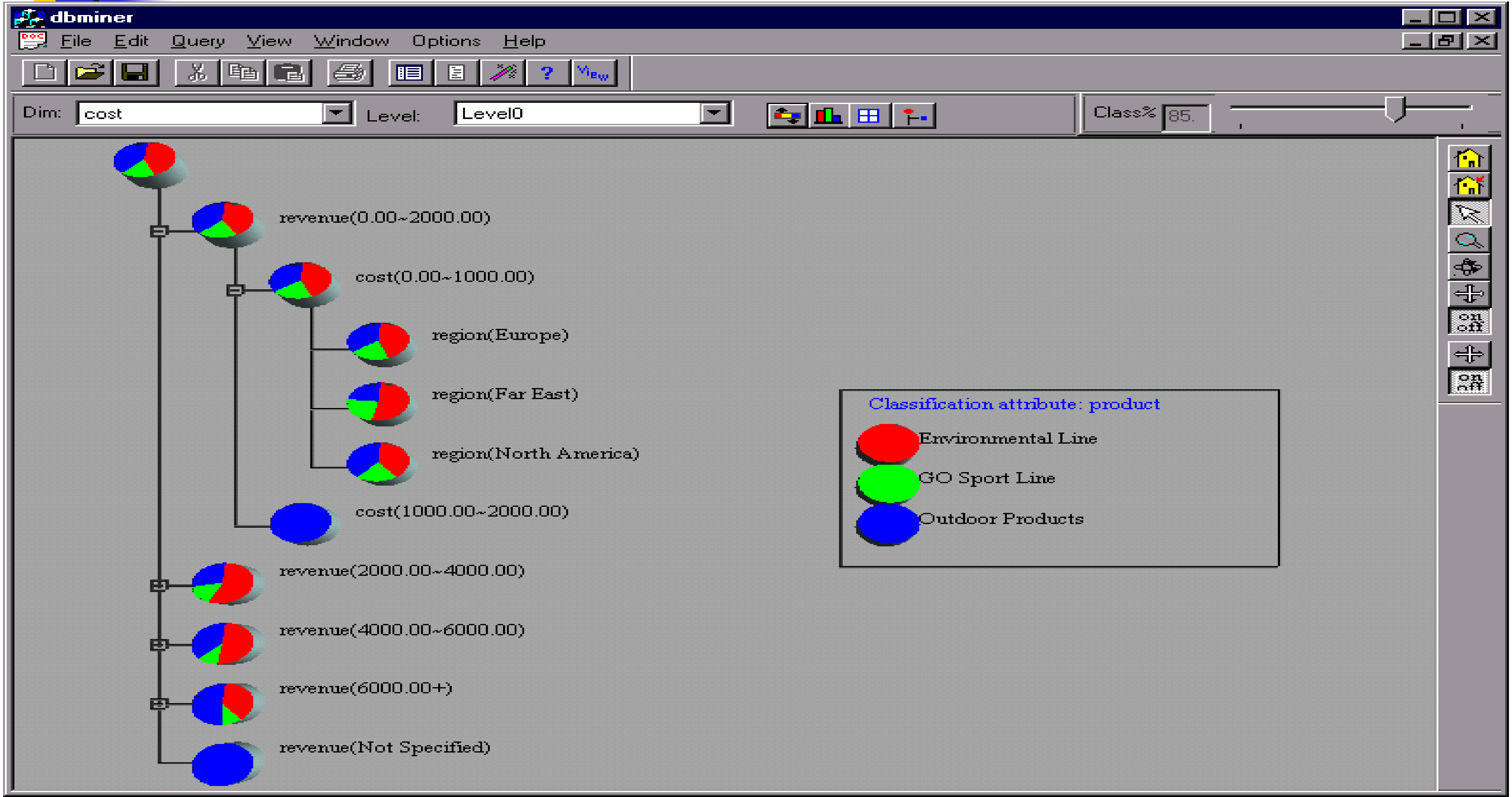
AVC-set on *credit_rating*

Credit rating	Buy_Computer	
	yes	no
fair	6	2
excellent	3	3

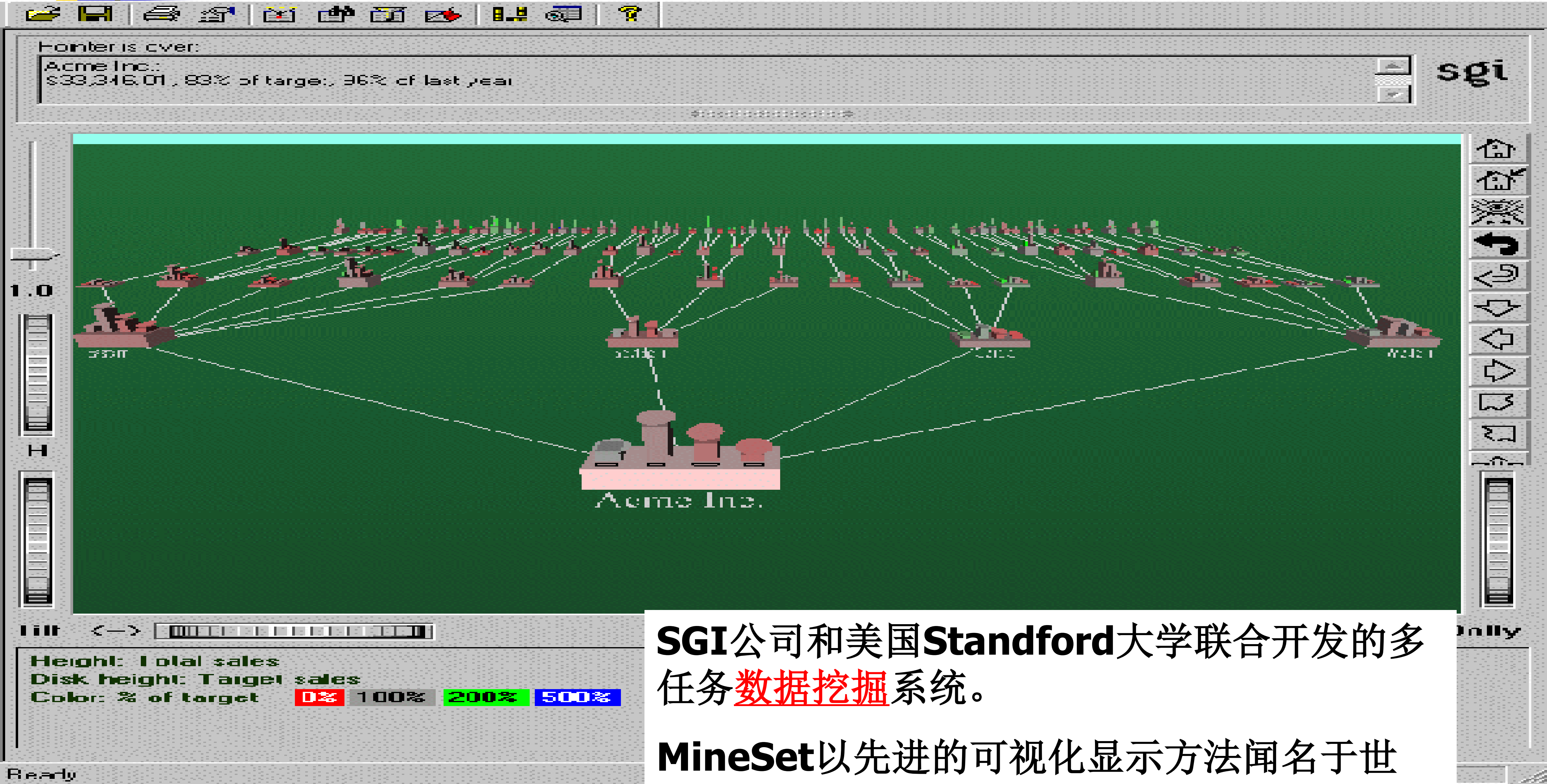
BOAT (Bootstrapped Optimistic Algorithm for Tree Construction)

- 使用一个叫做 ***bootstrapping*** 自助法的统计技术多个更小的样本集（子集）, 每一个可放入内存
- 每个子集产生一个树, 导致多个树
- 考察这些树并用他们构造一个新树 ***T’***
  - 事实证明， ***T’*** 非常接近于使用全部数据集构造的树
- **Adv:** 只要求扫描DB两遍,并且是一个增量算法.

分类结果的陈述/表示


dbminer
File Edit Query View Window Options Help
Dim: cost
Level: Level0
Class% 85.
revenue(0.00~2000.00)
cost(0.00~1000.00)
region(Europe)
region(Far East)
region(North America)
cost(1000.00~2000.00)
revenue(2000.00~4000.00)
revenue(4000.00~6000.00)
revenue(6000.00+)
revenue(Not Specified)
Classification attribute: product
Environmental Line
GO Sport Line
Outdoor Products

决策树可视化SGI/MineSet 3.0



SGI公司和美国**Standford**大学联合开发的多任务数据挖掘系统。

MineSet以先进的可视化显示方法闻名于世

第十一次作业

- 1、分类算法的基本流程是什么。什么叫做训练集（标签），测试集，验证集。
- 2、如何评价分类算法？混淆矩阵，TP，TN，FP，FN。精度，回归率。
- 3、决策树ID3的算法流程。
- 4、朴素贝叶斯的流程。
- 5、决策树ID3讲解过程中，两个案例，手工计算一下，尤其是信息增益的计算。
- 6、朴素贝叶斯的案例，手工计算一下，ppt上的那个例子。

Chapter 7. 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
- 模型评价与选择
- 提高分类准确率的技术:集成方法Ensemble Methods
- Summary

贝叶斯理论

- 令X 为数据样本: 类标签未知
- 令H为一个假设在： 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，假设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 X, *假设* H的后验概率 P(H|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 x prior/evidence

- 预测X属于类别C_2当且仅当概率$P(C_i|X)$是所有 $P(C_k|X)$ for all the *k* classes最大的
- 实际困难: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X = (x_1, x_2, …, x_n)**

- 假定有 ***m*** 个类别 **C_1, C_2, …, C_m.**

- 分类就是推导最大的后验概率, **i.e., the maximal $P(C_i|X)$**
- 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(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):

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

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

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

  - $P(x_k|C_i)$=

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

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	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

Naïve Bayesian Classifier: 例子

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

$P(X|C_i)$: P(X|buys_computer = "yes") = 0.222 x 0.444 x 0.667 x 0.667 = 0.044
P(X|buys_computer = "no") = 0.6 x 0.4 x 0.2 x 0.4 = 0.019

$P(X|C_i)*P(C_i)$: P(X|buys_computer = "yes") * P(buys_computer = "yes") = 0.028
P(X|buys_computer = "no") * P(buys_computer = "no") = 0.007

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

贝叶斯分类: Why?

- 一个统计学分类器: 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

- **Performance:** 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, 最优决策制定提供标准（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, e=low (0), income= medium (990), and income = high (10)
- Use Laplacian correction校准 (or Laplacian estimator估计法)
  - *Adding 1 to each case*

    Prob(income = low) = 1/1003

    Prob(income = medium) = 991/1003

    Prob(income = high) = 11/1003
  - 校准的 "corrected" 概率估计很接近未校准的

Naïve Bayesian Classifier:评论

- Advantages
  - Easy to implement
  - Good results obtained in most of the cases
- 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 7. 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
- 模型评价与选择
- 提高分类准确率的技术:集成方法Ensemble Methods
- Summary

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

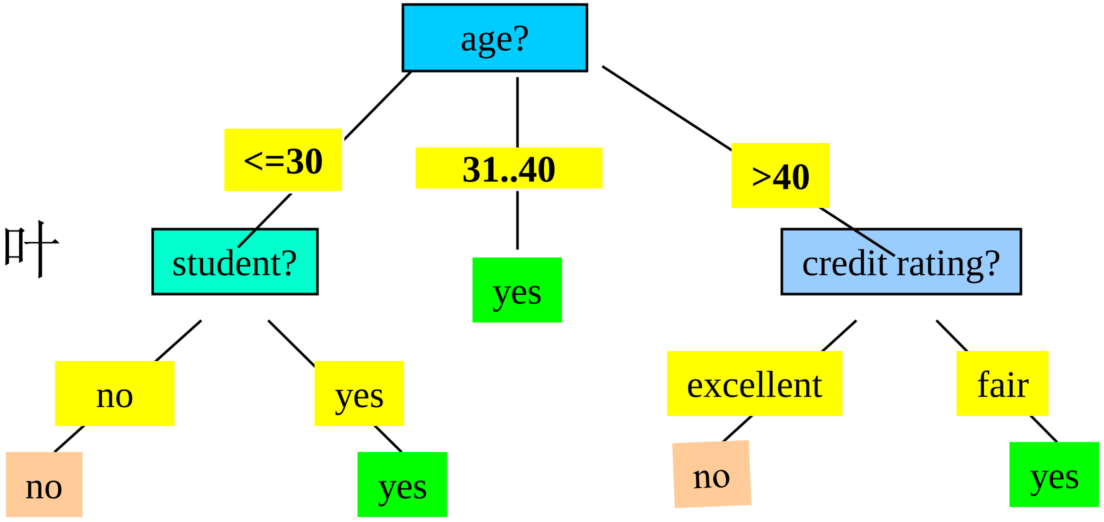
  - 规则前件/前提 vs. 规则结论
- 评估规则: 覆盖率*coverage* and 准确率*accuracy*
  - n_{covers} = # 规则R覆盖的元组数 % 给定元组，规则的前提满足—覆盖元组
  - $n_{correct}$ = # R正确分类的元组数

  $coverage(R) = n_{covers} /|D|$ /%D: 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*

顺序覆盖算法的规则归纳

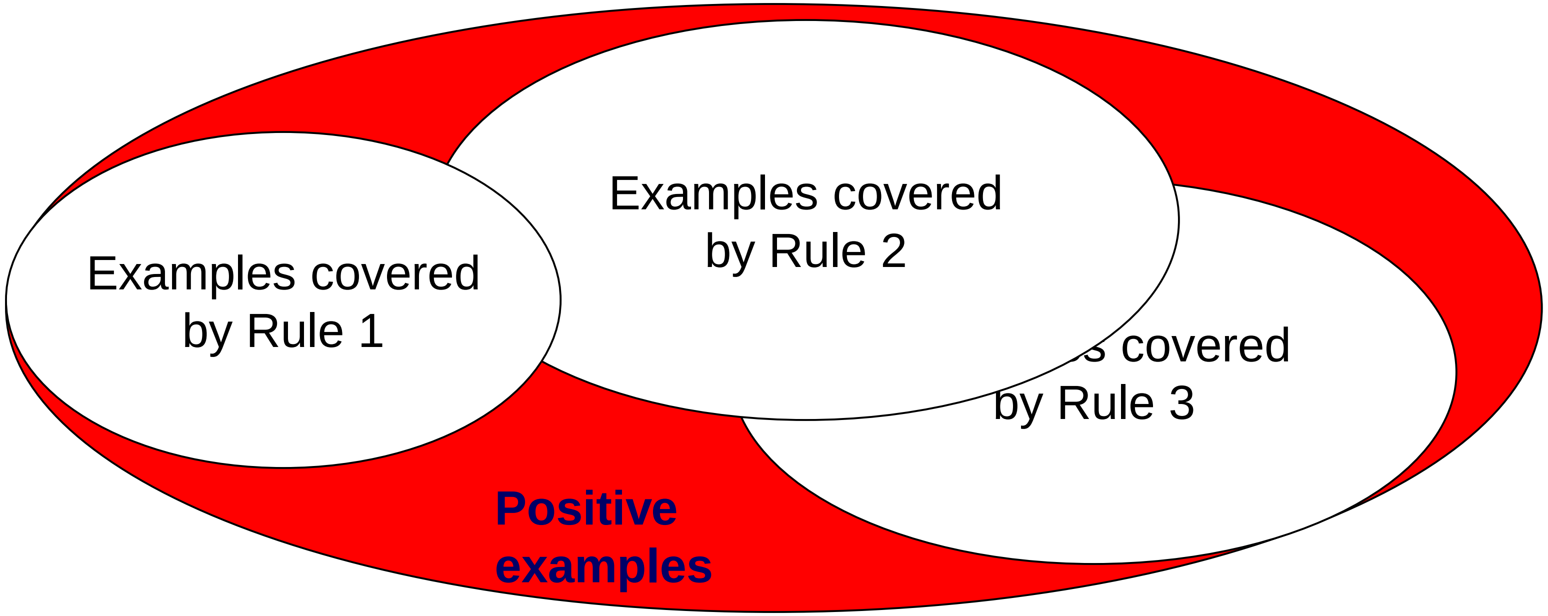
- 顺序覆盖算法：直接从训练数据抽取规则
- 典型的算法：FOIL, AQ, CN2, RIPPER
- 规则被顺序地学习，类C_i的规则将尽量覆盖C_i的元组，不或少覆盖其他类的元组
- Steps:
  - 一次学习一个规则
  - 每学习一个规则，删除此规则覆盖的元组
  - 对剩下的元组重复该过程直到终止条件，e.g.，没有训练样本/返回的规则的质量低于用户给定的阈值
- 与决策树对照： 同时学习一组规则

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

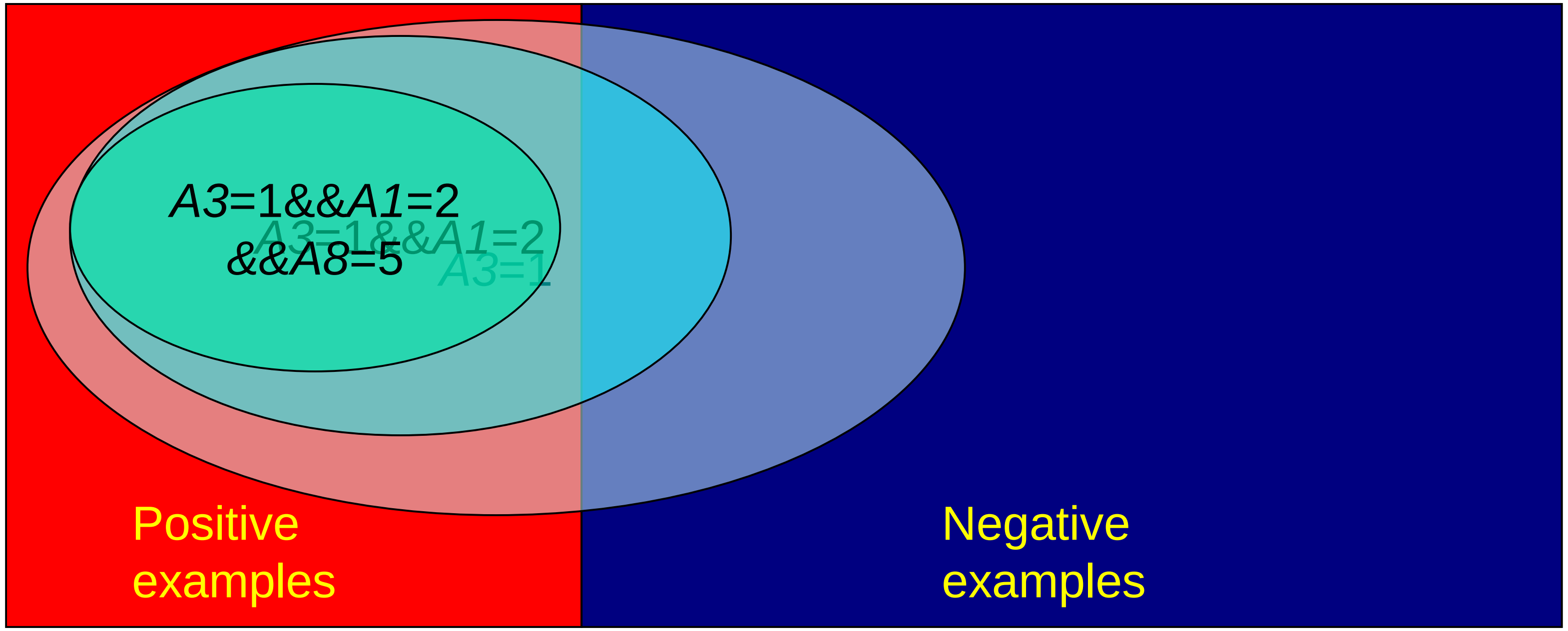
- To generate a rule

  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



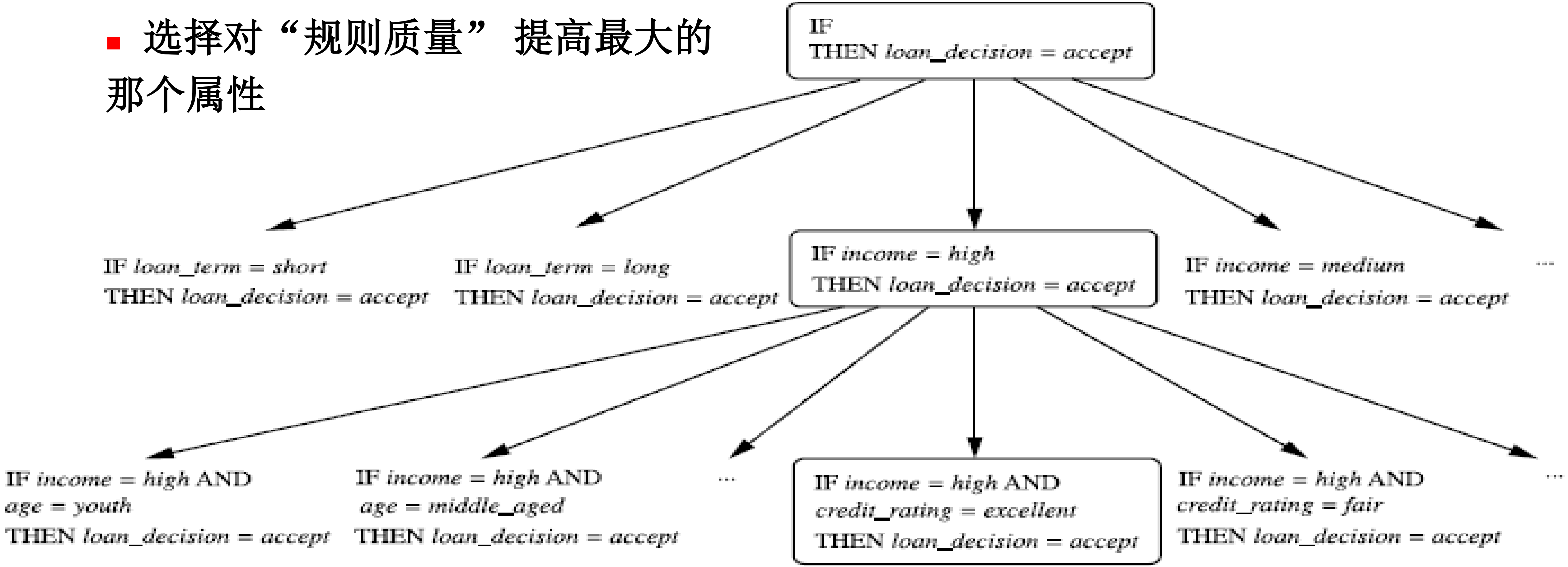
输出：IF-THEN规则的集合。

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 = 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（于规则中）*
  - 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 (in FOIL & RIPPER): 评价扩展条件的info_gain**

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

    - 偏向于具有高准确率并覆盖许多正元组的规则
    - 正用于学习规则的类的元组—正元组；其余为负元组
    - **Pos(neg)**:规则覆盖的正（负）元组数
- 基于一个独立的测试集进行规则剪枝（即删除一个属性测试）

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

Pos/neg are #被R覆盖的正/负元组.

If 规则R 剪枝后*FOIL_Prune* 较高, 那么剪枝R

Chapter 7. 分类:模型评价与选择

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

模型评价与选择

- 评价指标: 怎样度量准确率?考虑其他指标??
- 使用测试集（带标签）代替训练集评估准确度
- 估计分类器准确率的方法:
  - Holdout method, random subsampling
  - 交叉验证 Cross-validation
  - 自助法（解靴带）Bootstrap
- Comparing classifiers:
  - 置信区间Confidence intervals
  - 代价效益分析和ROC曲线
    - Cost-benefit analysis and ROC Curves

分类器评价指标: 混淆矩阵

混淆矩阵Confusion Matrix:

Actual class\Predicted class	C_1	¬ C_1
C_1	**True Positives (TP)**	**False Negatives (FN)**
¬ C_1	**False Positives (FP)**	**True Negatives (TN)**

- 感兴趣的类定为“正类”或“阳性类”，对应的为“负/阴性类”，正样本/负样本
- 给定 m 个类, $CM_{i,j}$ 表示 # 类i的样本被分类器分到类别 j 的个数
- 可以提供额外的行/列提供“合计”和“识别率”

例子:

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

分类器评价指标: 准确度, 误差率, 灵敏性Sensitivity, 特效性Specificity

A\P	C	¬C	
C	**TP**	**FN**	**P**
¬C	**FP**	**TN**	**N**
	P’	**N’**	**All**

- 分类器准确度, or 识别率: 测试元组被正确识别的比例

  Accuracy = (TP + TN)/All

- 误差率: *1 – accuracy*, or

  Error rate = (FP + FN)/All

- **Class Imbalance Problem类分布不平衡问题:**
  - One class may be *rare*, e.g. fraud, or HIV-positive
  - **Sensitivity**: True Positive recognition rate
    - **Sensitivity = TP/P**
  - **Specificity**: True Negative recognition rate
    - **Specificity = TN/N**

分类器评价指标:

- **Precision: 正确 – 被分类器标记为正类的样本中实际上属于“正类”的比例**

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

- **Recall: completeness完全– what % of positive tuples did the classifier label as positive?**
- **Perfect score is 1.0**

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

- **精度和召回率逆关系**
- ***F* measure (*F_1* or *F*-score):精度和召回的调和平均值,**

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

- ***$F_ß$*:精确度和召回率的加权量**
  - **assigns ß times as much weight to recall as to precision**

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

分类器评价指标: 例子

真实类\预测类	cancer = yes	cancer = no	Total	Recognition(%)
cancer = yes	**90**	**210**	300	30.00 (*sensitivity*
cancer = no	**140**	**9560**	9700	98.56 (*specificity*)
Total	230	9770	10000	96.40 (*accuracy*)

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

评测分类器的正确率

- **Holdout method**
  - 给定数据随机分成两个部分
    - 训练集 (e.g., 2/3) 用于模型构造
    - 测试集 (e.g., 1/3) 用于正确率估计
  - 随机抽样: **a variation of holdout**
    - 重复**holdout k**次, **accuracy =** 所有正确率的平均值
- **Cross-validation (*k*-fold, k = 10** 最常用**)**
  - 随机分割数据为 ***k*** 互不相交的子集, 每一个大小近似相等
  - 在***i*-th** 迭代中, 使用D_i为测试集其他的为训练集
  - 留一法: ***k*** **folds where *k* = # of tuples, for small sized data**
  - ***Stratified cross-validation***:每个部分分层使得每个子集中类分布近似于原始数据

评测分类器的正确率: Bootstrap

- **Bootstrap**
  - 对于小样本数据，效果很好
  - 从给定样本中又放回的均匀抽样 ***with replacement***
    - i.e., 每次一个样本被选中, 把它加入训练集并且等可能得被再次选中
- **多个自助法, 最常用的是 .632 boostrap**
  - 含 ***d*** 个样本的数据集有放回抽样***d*** 次, 产生d个样本的训练集. 没有被抽到的样本组成测试集. 大约63.2% 的样本被抽中，剩余的36.8% 形成测试集(因为$(1 - 1/d)^d \approx e^{-1} = 0.368$)
  - 重复抽样过程***k*** 次, 总体准确率为:

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

估计置信区间: 分类器 M_1 vs. M_2

- 假定有连个分类器 M_1 and M_2, 那一个更好?
- 用10-fold cross-validation获得了 $\overline{err}(M_1)$ $\overline{err}(M_2)$
- 这些平均误差率仅仅是未来数据总体误差的一种估计
- 2个错误率之间差别如果是否是偶然的?
  - 使用统计显著性检验
  - 获得估计误差的confidence limits置信界

估计置信区间: Null Hypothesis

- 执行 10-fold cross-validation
- 假定样本服从*k–1*个自由度的 t distribution (*k=10*) degrees of freedom
- Use t-test (or Student's t-test)
- 零假设 Null Hypothesis: M_1 & M_2 相同（即没有区别）
- 如果可以拒绝 null hypothesis, 那么
  - 可以断定M_1 & M_2 间的不同是统计上显著的
  - Chose model with lower error rate

估计置信区间: t-test

- 当只有一个测试集时: 成对比较 pairwise comparison
  - 对于10倍交叉验证中的 i^{th} round, 使用相同的样本分割 来计算 $err(M_1)_i$ and $err(M_2)_i$
  - 然后球平均over 10 $\overline{err}(M_1)$ and $\overline{err}(M_2)$
  - t-test computes t-statistic with *k-1* degrees of freedom:

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

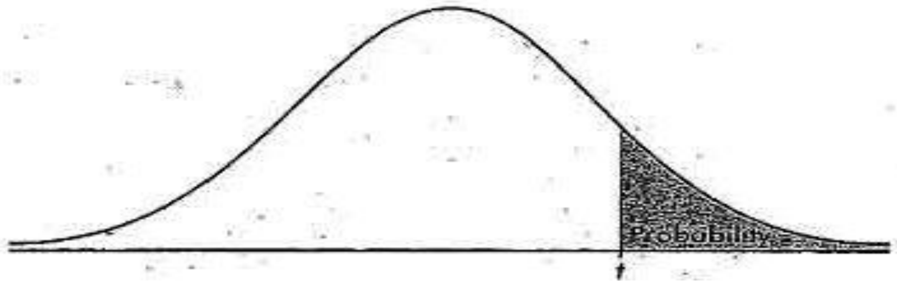
其中

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

where $$var(M_1 - M_2) = \sqrt{\frac{var(M_1)}{k_1} + \frac{var(M_2)}{k_2}},$$

where k_1 & k_2 are # of cross-validation samples used for M_1 & M_2, resp.

估计置信区间: Table for t-distribution



- **Symmetric**
- **Significance level, e.g., *sig = 0.05* or *5%* means M_1 & M_2 are *significantly different* for 95% of population**
- **Confidence limit, *z = sig/2***

TABLE B: t-DISTRIBUTION CRITICAL VALUES

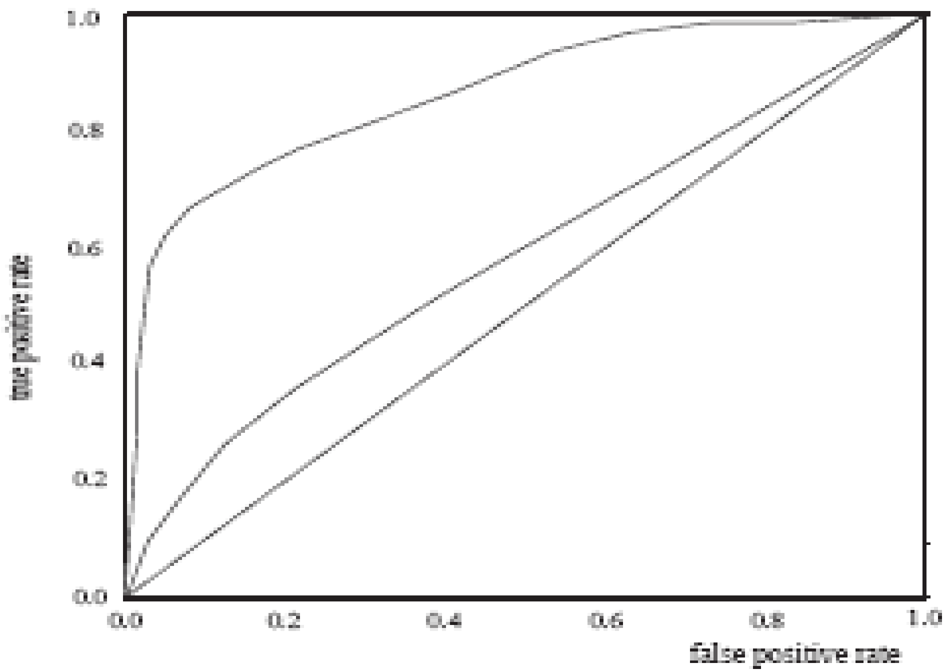
	Tail probability p											
df	.25	.20	.15	.10	.05	.025	.02	.01	.005	.0025	.001	.0005
1	1.000	1.376	1.963	3.078	6.314	12.71	15.89	31.82	63.66	127.3	318.3	636.6
2	.816	1.061	1.386	1.886	2.920	4.303	4.849	6.965	9.925	14.09	22.33	31.60
3	.765	.978	1.250	1.638	2.353	3.182	3.482	4.541	5.841	7.453	10.21	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											

估计置信区间: Statistical Significance

- M_1 & M_2 是否显著得不同?
  - Compute *t*. Select *significance level* (e.g. *sig = 5%)*
  - Consult table for t-distribution: Find *t value* corresponding to *k-1 degrees of freedom* (here, 9)
  - t-分布对称: 通常显示分布的上百分点 % → 查找值confidence limit *z=sig/2* (here, 0.025)
  - If t > z or t < -z, 那么t的值位于拒绝域:
    - Reject null hypothesis that mean error rates of M_1 & M_2 are same
    - Conclude: <u>statistically significant</u> difference between M_1 & M_2
  - Otherwise, conclude that any difference is chance

模型选择: ROC Curves

- ROC (Receiver Operating Characteristics) curves: 图形比较分类模型
- 源于信号检测理论
- true positive rate和false positive rate间的折衷
- ROC 曲线下的面积就是模型正确率的度量
- 测试元组递减序排列: 最可能属于正类的排在最顶端
- The closer to the diagonal line (i.e., the closer the area is to 0.5), the less accurate is the model

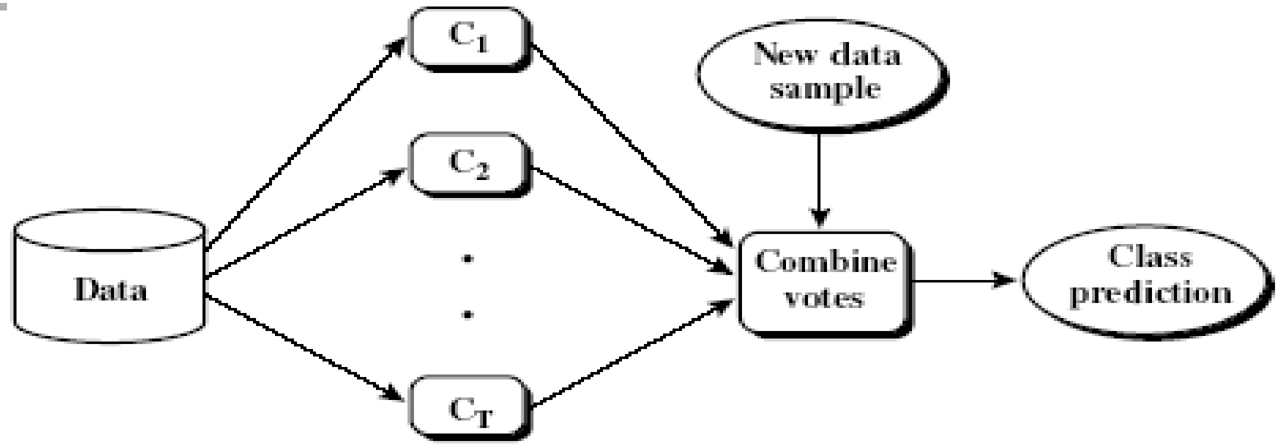


- 垂直坐标表示the true positive rate
- 水平坐标表示the false positive rate
- 同时显示对角线
- A model with perfect accuracy will have an area of 1.0

Chapter 7. 分类:集成方法

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

集成方法: Increasing the Accuracy



- 集成方法 Ensemble methods
  - 使用多个模型的组合来提高accuracy
  - 组合多个学习的模型, M_1, M_2, …, M_k,来获得一个提高的模型M*
- Popular ensemble methods
  - 装袋Bagging: 多个分类器的结果进行多数表决
  - 提升Boosting: 多个分类器的结果权重投票
  - 集成Ensemble: combining a set of heterogeneous classifiers

装袋Bagging

- 训练
  - 给定包含d个元组的数据D, 在第 *i*次迭代，从D中有放回抽取d个样本组成训练集D_i (i.e., bootstrap)， 从D_i 学习一个分类器M_i
- 分类: 分类一个未知样本 X
  - 每个分类器 M_i 给出预测结果
  - 装袋分类器M* 计算投票，把得票最多的类分配给X
- 预测: 每个分类器预测的值的平均值
- 正确性Accuracy
  - 常常优于D 上单个分类器的正确率
  - 对噪音数据: 不会很差，更健壮
  - Proved improved accuracy in prediction

提升 Boosting

- 类比:咨询几个医生，在原来的诊断准确性的基础上分配权重，加权诊断的组合为结果
- Boosting如何工作?
  - **Weights 分配给每个训练样本**
  - **迭代学习一系列分类器**
  - **学习M_i 后, 权重更新使得，后续得分类器M_{i+1}更关注于M_i错误分类的训练样本**
  - **最后的分类器M* 组合了每个独立分类器的投票,其中每个分类器的权重势其正确率的函数**
- 可以扩充Boosting 算法用于数值预测
- 与bagging比较: Boosting倾向于得到更高的准确率, 但有过拟合错误分类数据的风险

Adaboost (Freund and Schapire, 1997)

- 数据集含 *d* class-labeled 元组, $(X_1, y_1), \ldots, (X_d, y_d)$
- 最初, 每个元组的权重为1/d
- 在k轮中产生 k classifiers. 在第 i轮,
  - 从D 有放回抽取训练集 D_i （大小相等
  - 每个元组被选中的概率基于其权重
  - 分类模型M_i 学习自D_i
  - 使用D_i为测试集计算误差率
  - 如果一个元组被错分,权重增加, o.w. 否则下降
- 误差率: $err(X_j)$ 为错误分类元组X_j误差，分类器M_i 误差率是元组错误分类的权重和:

$$error(M_i) = \sum_j^d w_j \times err(\mathbf{X_j})$$

- 分类器M_i投票权重为

$$\log \frac{1 - error(M_i)}{error(M_i)}$$

随机森林 (Breiman 2001)

- **Random Forest:**
  - 每个分类器为***decision tree***，在每个结点上使用随机选出的属性来分裂产生判定树
  - 分类时, 每棵树投票得票最多的类返回结果
- 两种构造方法:
  - **Forest-RI (*random input selection*):** 每个结点随机选**F** 个属性为分裂的候选.用**CART**方法产生最大尺寸的树
  - **Forest-RC (*random linear combinations*):** 以现有属性的线性组合来产生新属性(降低了单个分类器间的相关性)
- 准确率比得上**Adaboost**, 对误差和孤立点更稳健
- 每次分裂时对选出的候选属性数目不敏感，**faster than bagging or boosting**

分类类别不平衡数据集

- 类别不平衡问题.
- 传统的方法假定平衡的类别分布和相等的错误代价: 不适合
- 二元分类中典型的方法处理不平衡数据:
  - 过采样Oversampling:对正类数据过/多采样
  - Under-sampling: 随机减少负类的样本
  - 阈值-移动Threshold-moving: 移动判定阈值t, 使得少数类元组更容易识别, 减少（昂贵的）假阴性错误的机会
  - 集成技术:
- Still difficult for class imbalance problem on multiclass tasks

预测误差的度量

- 度量预测准确率: 度量预测值与真实值的距离
- 损失函数: 度量 y_i 和预测值y_i' 间的误差
  - 绝对误差Absolute error: $| y_i – y_i'|$
  - 平方误差Squared error: $(y_i – y_i')^2$
- 检验误差 (泛化误差generalization error):
  - 平均绝对误差: 均方误差Mean squared error: $$\frac{\sum_{i=1}^{d}(y_i - y_i')^2}{d}$$

$$\frac{\sum_{i=1}^{d}| y_i - y_i'|}{d}$$

  - Relative absolute error: $$\frac{\sum_{i=1}^{d}| y_i - y_i'|}{\sum_{i=1}^{d}| y_i - \bar{y} |}$$ $$\frac{\sum_{i=1}^{d}(y_i - y_i')^2}{\sum_{i=1}^{d}(y_i - \bar{y})^2}$$
  - Relative squared error:
  - 均方误差夸大了离群点

Popularly use (square) root mean-square error, similarly, root relative squared error

What Is Prediction?

- (Numerical) 预测类似于分类
  - 构建一个模型
  - 利用模型来估计给定输入的连续或排序的值
- 与分类的不同
  - 分类是预测类别标签
  - 预测是模型连续值函数
- **Major method for prediction: regression**
  - 模型一个或多个预测变量和相应变量间的关系
- **Regression analysis**
  - 线性和多元回归
  - 非线性回归
  - 其他方法: **generalized linear model, Poisson regression, log-linear models, regression trees**

线性回归

- <u>Linear regression</u>: 包含一个响应变量y 和一个预测变量x

  $y = w_0 + w_1 x$

- <u>Method of least squares</u>: estimates the best-fitting straight line

$$w_1 = \frac{\sum_{i=1}^{|D|}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{|D|}(x_i - \bar{x})^2} \qquad w_0 = \bar{y} - w_1 \bar{x}$$

- <u>多元线性回归</u>: 包含多个预测变量
  - Training data is of the form $(X_1, y_1), (X_2, y_2), \ldots, (X_{|D|}, y_{|D|})$
  - 对2-D数据,有$y = w_0 + w_1 x_1 + w_2 x_2$
  - 通常用统计软件包求解 SAS, S-Plus
  - 多个非线性函数可以表示成上面这种形式

非线性回归

- 某些非线性模型可以用多项式函数
- 多项式回归模型可以变换为线性回归模型. 例如

  $$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3$$

  借助新变量: $x_2 = x^2$, $x_3 = x^3$

  $$y = w_0 + w_1 x + w_2 x_2 + w_3 x_3$$

- 其他函数,如幂函数, 也可以转化为线性函数
- **Some models are intractable nonlinear (e.g., 指数相求和)**
  - 可能通过更复杂的公式综合计算，得到最小二乘估计

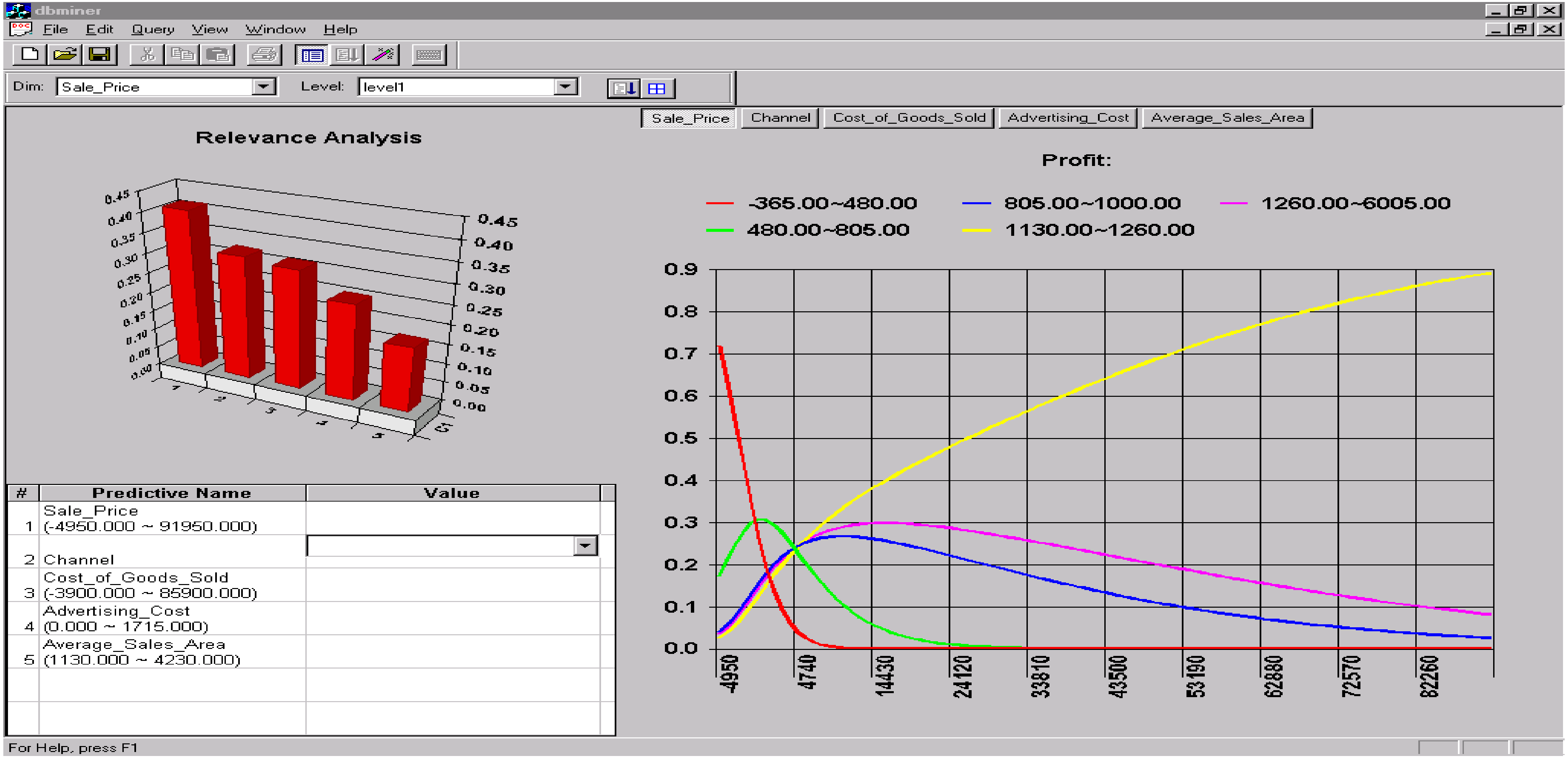
Other Regression-Based Models

- Generalized linear model:
  - Foundation on which linear regression can be applied to modeling categorical response variables
  - Variance of y is a function of the mean value of y, not a constant
  - Logistic regression: models the prob. of some event occurring as a linear function of a set of predictor variables
  - Poisson regression: models the data that exhibit a Poisson distribution
- Log-linear models: (for categorical data)
  - Approximate discrete multidimensional prob. distributions
  - Also useful for data compression and smoothing
- Regression trees and model trees
  - Trees to predict continuous values rather than class labels

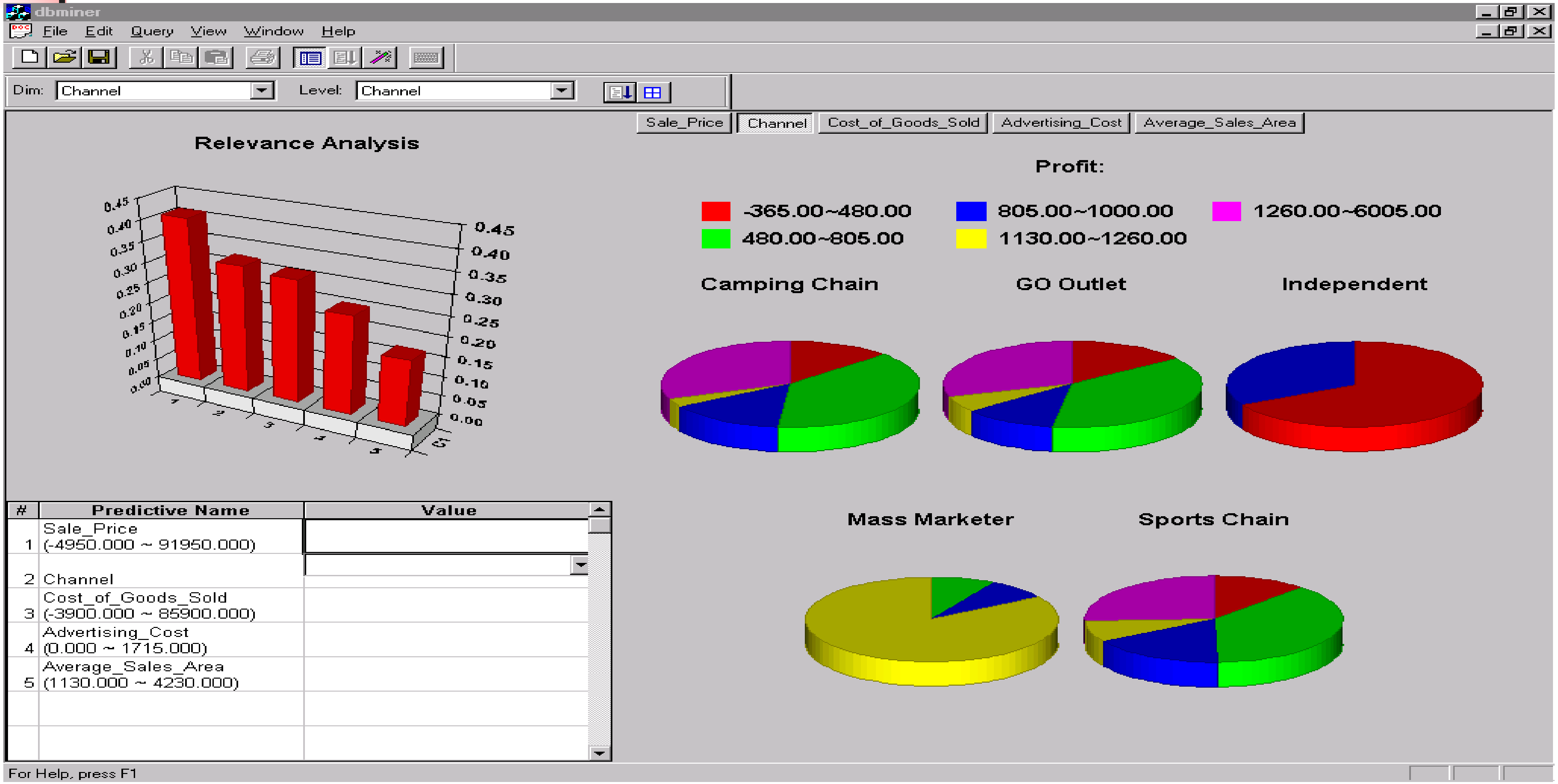
Regression Trees and Model Trees

- **Regression tree: proposed in CART system (Breiman et al. 1984)**
  - **CART: Classification And Regression Trees**
  - **Each leaf stores a *continuous-valued prediction***
  - **It is the *average value of the predicted attribute* for the training tuples that reach the leaf**
- **Model tree: proposed by Quinlan (1992)**
  - **Each leaf holds a regression model—a multivariate linear equation for the predicted attribute**
  - **A more general case than regression tree**
- **Regression and model trees tend to be more accurate than linear regression when the data are not represented well by a simple linear model**

Prediction: Numerical Data



Prediction: Categorical Data



Chapter 7. 分类: Summary

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

Summary (I)

- Classification is a form of data analysis that extracts models describing important data classes.
- Effective and scalable methods have been developed for decision tree induction, Naive Bayesian classification, rule-based classification, and many other classification methods.
- Evaluation metrics include: accuracy, sensitivity, specificity, precision, recall, F measure, and $F_{ß}$ measure.
- Stratified k-fold cross-validation is recommended for accuracy estimation. Bagging and boosting can be used to increase overall accuracy by learning and combining a series of individual models.

Summary (II)

- Significance tests and ROC curves are useful for model selection.
- There have been numerous comparisons of the different classification methods; the matter remains a research topic
- No single method has been found to be superior over all others for all data sets
- Issues such as accuracy, training time, robustness, scalability, and interpretability must be considered and can involve trade-offs, further complicating the quest for an overall superior method

References (1)

- C. Apte and S. Weiss. Data mining with decision trees and decision rules. Future Generation Computer Systems, 13, 1997
- C. M. Bishop, Neural Networks for Pattern Recognition. Oxford University Press, 1995
- L. Breiman, J. Friedman, R. Olshen, and C. Stone. Classification and Regression Trees. Wadsworth International Group, 1984
- C. J. C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2): 121-168, 1998
- P. K. Chan and S. J. Stolfo. Learning arbiter and combiner trees from partitioned data for scaling machine learning. KDD'95
- H. Cheng, X. Yan, J. Han, and C.-W. Hsu, Discriminative Frequent Pattern Analysis for Effective Classification, ICDE'07
- H. Cheng, X. Yan, J. Han, and P. S. Yu, Direct Discriminative Pattern Mining for Effective Classification, ICDE'08
- W. Cohen. Fast effective rule induction. ICML'95
- G. Cong, K.-L. Tan, A. K. H. Tung, and X. Xu. Mining top-k covering rule groups for gene expression data. SIGMOD'05

References (2)

- A. J. Dobson. An Introduction to Generalized Linear Models. Chapman & Hall, 1990.
- G. Dong and J. Li. Efficient mining of emerging patterns: Discovering trends and differences. KDD'99.
- R. O. Duda, P. E. Hart, and D. G. Stork. Pattern Classification, 2ed. John Wiley, 2001
- U. M. Fayyad. Branching on attribute values in decision tree generation. AAAI'94.
- Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. J. Computer and System Sciences, 1997.
- J. Gehrke, R. Ramakrishnan, and V. Ganti. Rainforest: A framework for fast decision tree construction of large datasets. VLDB'98.
- J. Gehrke, V. Gant, R. Ramakrishnan, and W.-Y. Loh, BOAT -- Optimistic Decision Tree Construction. SIGMOD'99.
- T. Hastie, R. Tibshirani, and J. Friedman. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer-Verlag, 2001.
- D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. Machine Learning, 1995.
- W. Li, J. Han, and J. Pei, CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules, ICDM'01.

References (3)

- T.-S. Lim, W.-Y. Loh, and Y.-S. Shih. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. Machine Learning, 2000.
- J. Magidson. The Chaid approach to segmentation modeling: Chi-squared automatic interaction detection. In R. P. Bagozzi, editor, Advanced Methods of Marketing Research, Blackwell Business, 1994.
- M. Mehta, R. Agrawal, and J. Rissanen. SLIQ : A fast scalable classifier for data mining. EDBT'96.
- T. M. Mitchell. Machine Learning. McGraw Hill, 1997.
- S. K. Murthy, Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey, Data Mining and Knowledge Discovery 2(4): 345-389, 1998
- J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81-106, 1986.
- J. R. Quinlan and R. M. Cameron-Jones. FOIL: A midterm report. ECML'93.
- J. R. Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann, 1993.
- J. R. Quinlan. Bagging, boosting, and c4.5. AAAI'96.

References (4)

- R. Rastogi and K. Shim. Public: A decision tree classifier that integrates building and pruning. VLDB’98.
- J. Shafer, R. Agrawal, and M. Mehta. SPRINT : A scalable parallel classifier for data mining. VLDB’96.
- J. W. Shavlik and T. G. Dietterich. Readings in Machine Learning. Morgan Kaufmann, 1990.
- P. Tan, M. Steinbach, and V. Kumar. Introduction to Data Mining. Addison Wesley, 2005.
- S. M. Weiss and C. A. Kulikowski. Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems. Morgan Kaufman, 1991.
- S. M. Weiss and N. Indurkhya. Predictive Data Mining. Morgan Kaufmann, 1997.
- I. H. Witten and E. Frank. Data Mining: Practical Machine Learning Tools and Techniques, 2ed. Morgan Kaufmann, 2005.
- X. Yin and J. Han. CPAR: Classification based on predictive association rules. SDM'03
- H. Yu, J. Yang, and J. Han. Classifying large data sets using SVM with hierarchical clusters. KDD'03.

Scalable Decision Tree Induction Methods

- SLIQ (EDBT'96 — Mehta et al.)
  - Builds an index for each attribute and only class list and the current attribute list reside in memory
- SPRINT (VLDB'96 — J. Shafer et al.)
  - Constructs an attribute list data structure
- PUBLIC (VLDB'98 — Rastogi & Shim)
  - Integrates tree splitting and tree pruning: stop growing the tree earlier
- RainForest (VLDB'98 — Gehrke, Ramakrishnan & Ganti)
  - Builds an AVC-list (attribute, value, class label)
- BOAT (PODS'99 — Gehrke, Ganti, Ramakrishnan & Loh)
  - Uses bootstrapping to create several small samples

Data Cube-Based Decision-Tree Induction

- Integration of generalization with decision-tree induction (Kamber et al.'97)
- Classification at primitive concept levels
  - E.g., precise temperature, humidity, outlook, etc.
  - Low-level concepts, scattered classes, bushy classification-trees
  - Semantic interpretation problems
- Cube-based multi-level classification
  - Relevance analysis at multi-levels
  - Information-gain analysis with dimension + level

第十二次作业

- 1、决策树ID3算法的Java实现。
- 2、朴素贝叶斯算法的Java实现。
- 3、基本概念学习：kNN算法流程；SVM；NN(ANN)；
- 4、聚类算法的基本流程
- 5、k-means算法的基本流程