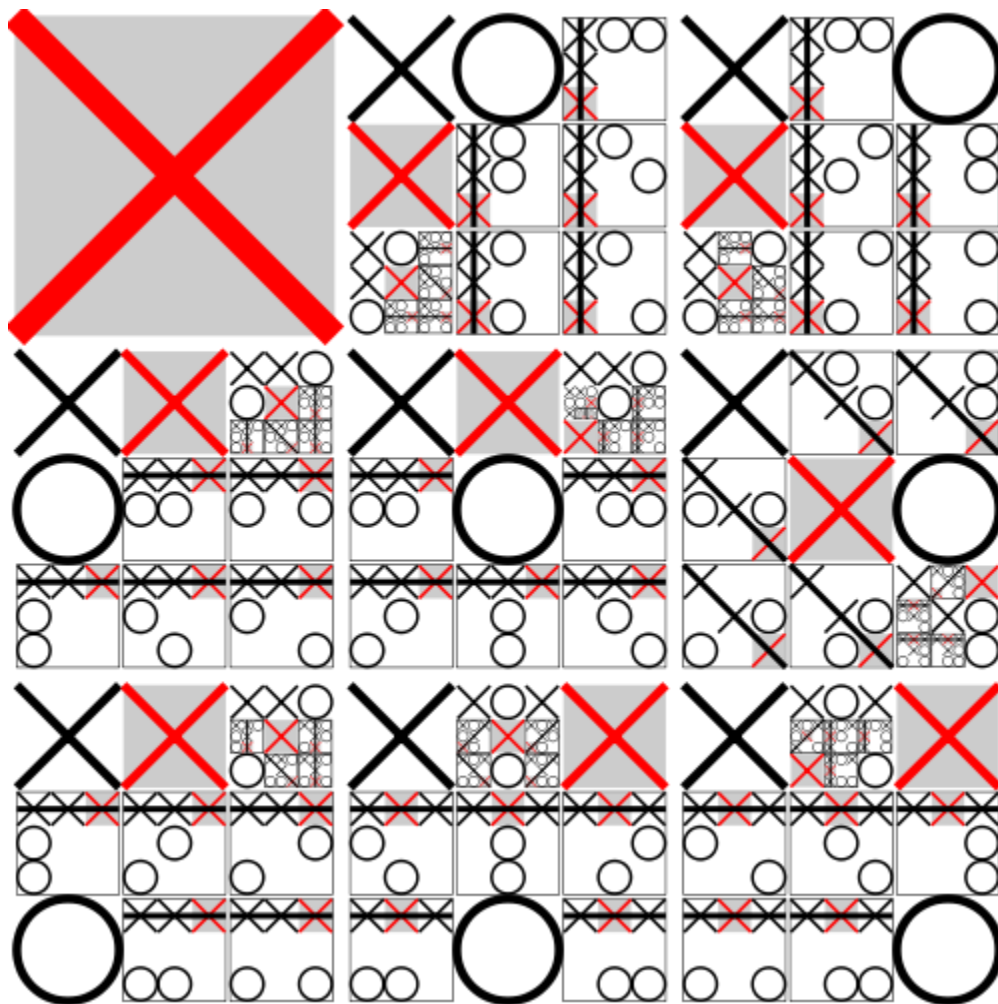


机器学习

第6周



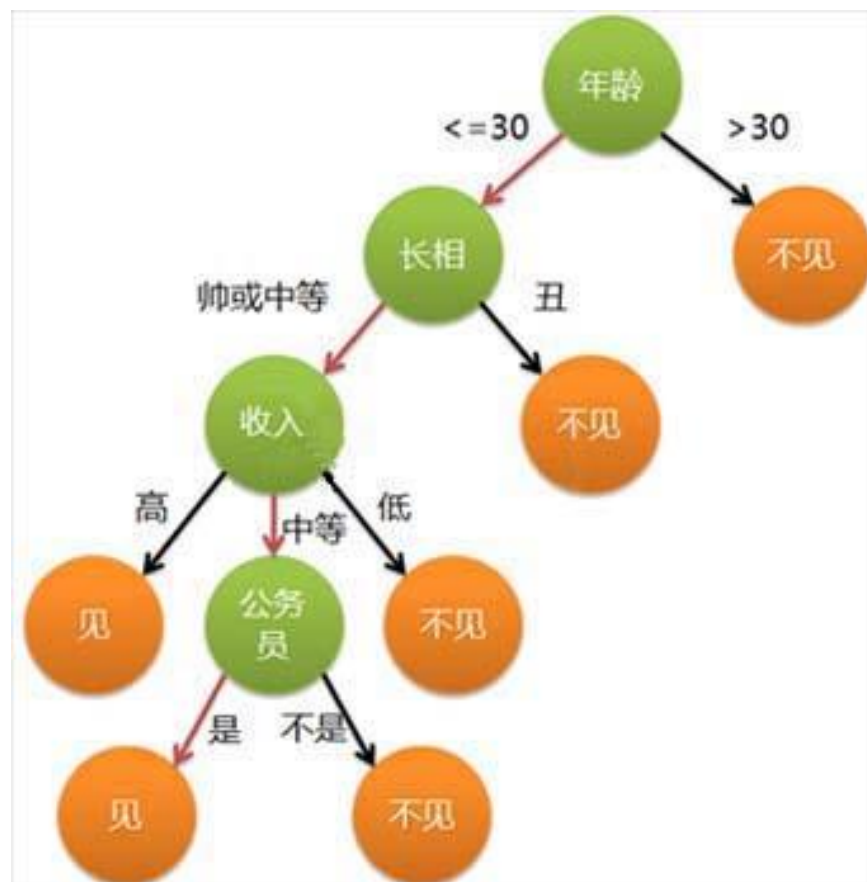
【声明】 本视频和幻灯片为炼数成金网络课程的教学资料，所有资料只能在课程内使用，不得在课程以外范围散播，违者将可能被追究法律和经济责任。

课程详情访问炼数成金培训网站

<http://edu.dataguru.cn>

决策树 decision tree

- 什么是决策树
- 输入：学习集
- 输出：分类规则（决策树）

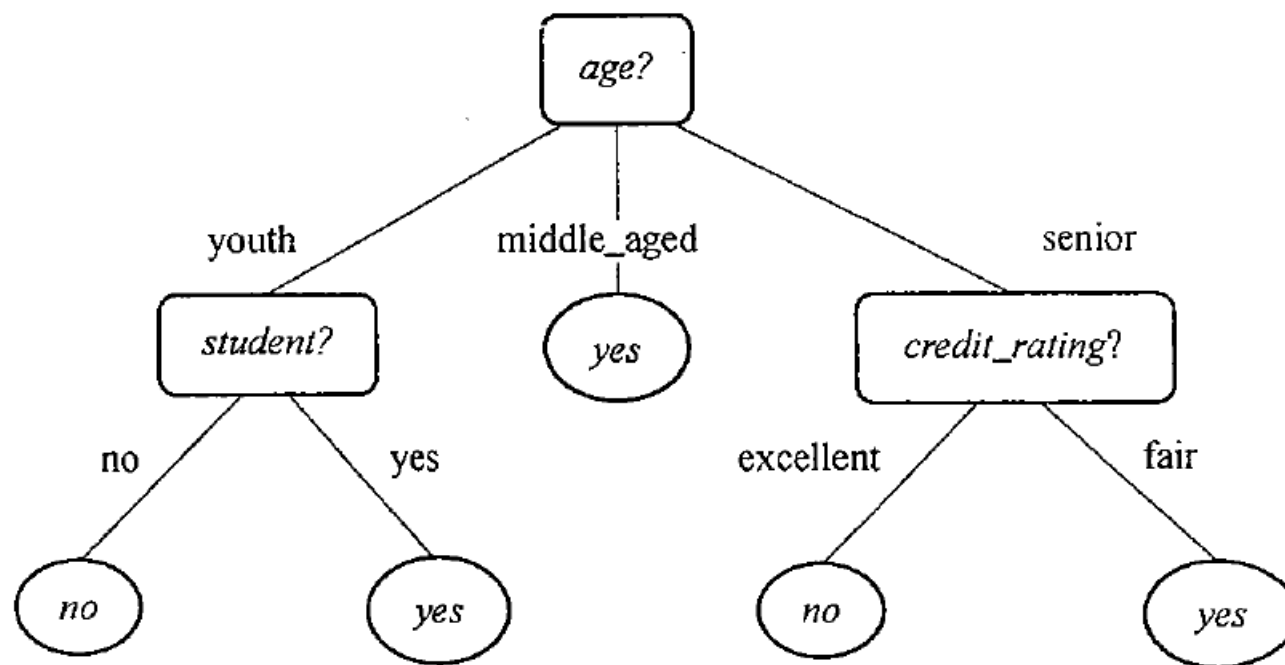


- 70年代后期至80年代初期，Quinlan开发了ID3算法（迭代的二分器）
- Quinlan改进了ID3算法，称为C4.5算法
- 1984年，多位统计学家在著名的《Classification and regression tree》书里提出了CART算法
- ID3和CART几乎同期出现，引起了研究决策树算法的旋风，至今已经有多种算法被提出

表 8.1 AllElectronics 顾客数据库标记类的训练元组

<i>RID</i>	<i>age</i>	<i>income</i>	<i>student</i>	<i>credit_rating</i>	<i>Class: buys_computer</i>
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

例子：期待输出的结果



- 该按什么样的次序来选择变量（属性）？
- 最佳分离点（连续的情形）在哪儿？

■ 信息增益计算

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

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

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

例子：信息增益计算

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

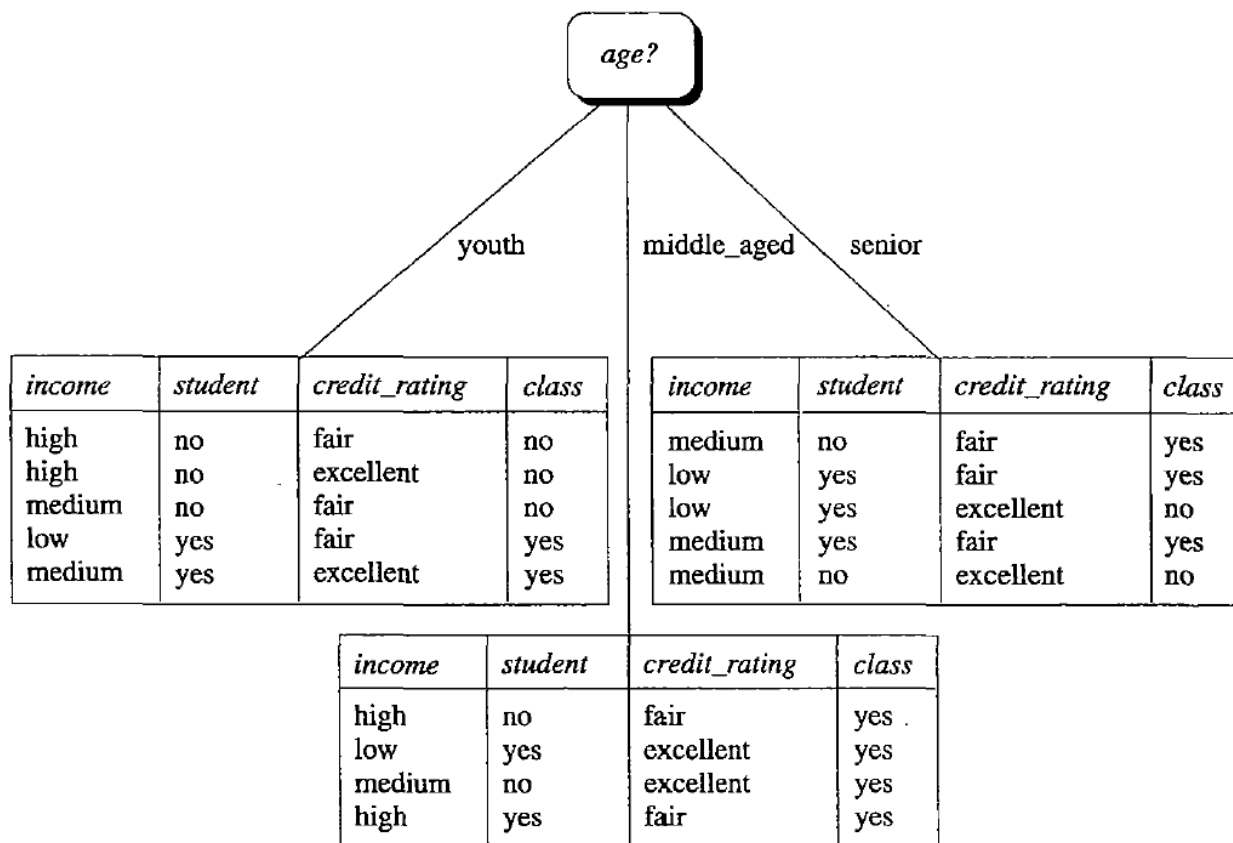
$$\begin{aligned} Info_{age}(D) &= \frac{5}{14} \times \left(-\frac{2}{5}\log_2 \frac{2}{5} - \frac{3}{5}\log_2 \frac{3}{5} \right) + \frac{4}{14} \times \left(-\frac{4}{4}\log_2 \frac{4}{4} - \frac{0}{4}\log_2 \frac{0}{4} \right) \\ &\quad + \frac{5}{14} \times \left(-\frac{3}{5}\log_2 \frac{3}{5} -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

$$Gain(age) = Info(D) - Info_{age}(D) = 0.940 - 0.694 = 0.246 \text{ 位}$$

Age属性的信息增益最高，故首先选择这个变量

例子：继续在子树重复挑选变量的步骤

- 已经可以肉眼观察：左侧student，右侧credit_rating，下方直接输出叶子yes



连续变量怎样选择分离点？

- 韩家炜书第219页

- 用SNS社区中不真实账号检测的例子说明如何使用ID3算法构造决策树。为了简单起见，我们假设训练集合包含10个元素。其中s、m和l分别表示小、中和大。

日志密度	好友密度	是否使用真实头像	账号是否真实
s	s	no	no
s	l	yes	yes
l	m	yes	yes
m	m	yes	yes
l	m	yes	yes
m	l	no	yes
m	s	no	no
l	m	no	yes
m	s	no	yes
s	s	yes	no

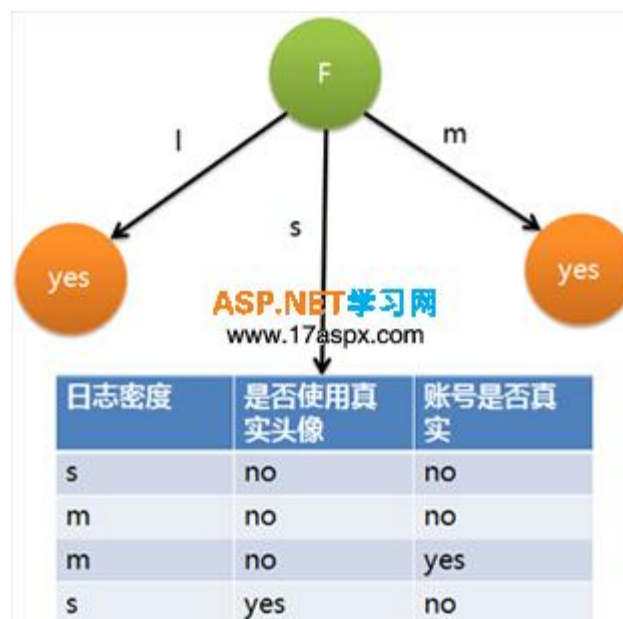
- 设L、F、H和R表示日志密度、好友密度、是否使用真实头像和账号是否真实，下面计算各属性的信息增益。

$$info(D) = -0.7\log_2 0.7 - 0.3\log_2 0.3 = 0.7 * 0.51 + 0.3 * 1.74 = 0.879$$

$$info_L(D) = 0.3 * \left(-\frac{0}{3}\log_2 \frac{0}{3} - \frac{3}{3}\log_2 \frac{3}{3}\right) + 0.4 * \left(-\frac{1}{4}\log_2 \frac{1}{4} - \frac{3}{4}\log_2 \frac{3}{4}\right) + 0.3 * \left(-\frac{1}{3}\log_2 \frac{1}{3} - \frac{2}{3}\log_2 \frac{2}{3}\right) = 0 + 0.326 + 0.277 = 0.603$$

$$gain(L) = 0.879 - 0.603 = 0.276$$

- 因此日志密度的信息增益是0.276。用同样方法得到H和F的信息增益分别为0.033和0.553。因为F具有最大的信息增益，所以第一次分裂选择F为分裂属性，分裂后的结果如下图表示：



- 在上图的基础上，再递归使用这个方法计算子节点的分裂属性，最终就可以得到整个决策树。
- 这个方法称为ID3算法，还有其它的算法也可以产生决策树
- 对于特征属性为**连续值**，可以如此使用ID3算法：先将D中元素按照特征属性排序，则每两个相邻元素的中间点可以看做潜在分裂点，从第一个潜在分裂点开始，分裂D并计算两个集合的期望信息，具有最小期望信息的点称为这个属性的最佳分裂点，其信息期望作为此属性的信息期望。

- 信息增益的方法倾向于首先选择因子数较多的变量
- 信息增益的改进：增益率

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

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

■ 重新计算前例

例 8.2 属性 *income* 的增益率的计算。属性 *income* 的测试将表 8.1 中的数据划分成 3 个分区，即 *low*、*medium* 和 *high*，分别包含 4、6 和 4 个元组。为了计算 *income* 的增益率，首先使用 (8.5) 式得到

$$SplitInfo_A(D) = -\frac{4}{14} \times \log_2 \frac{4}{14} - \frac{6}{14} \times \log_2 \frac{6}{14} - \frac{4}{14} \times \log_2 \frac{4}{14} = 1.557$$

由例 8.1， $Gain(income) = 0.029$ 。因此， $GainRatio(income) = 0.029/1.557 = 0.019$ 。 ■

- 使用基尼指数选择变量

$$Gini(D) = 1 - \sum_{i=1}^m p_i^2$$

$$Gini_A(D) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2)$$

$$\Delta Gini(A) = Gini(D) - Gini_A(D)$$

- 韩家炜书第221页

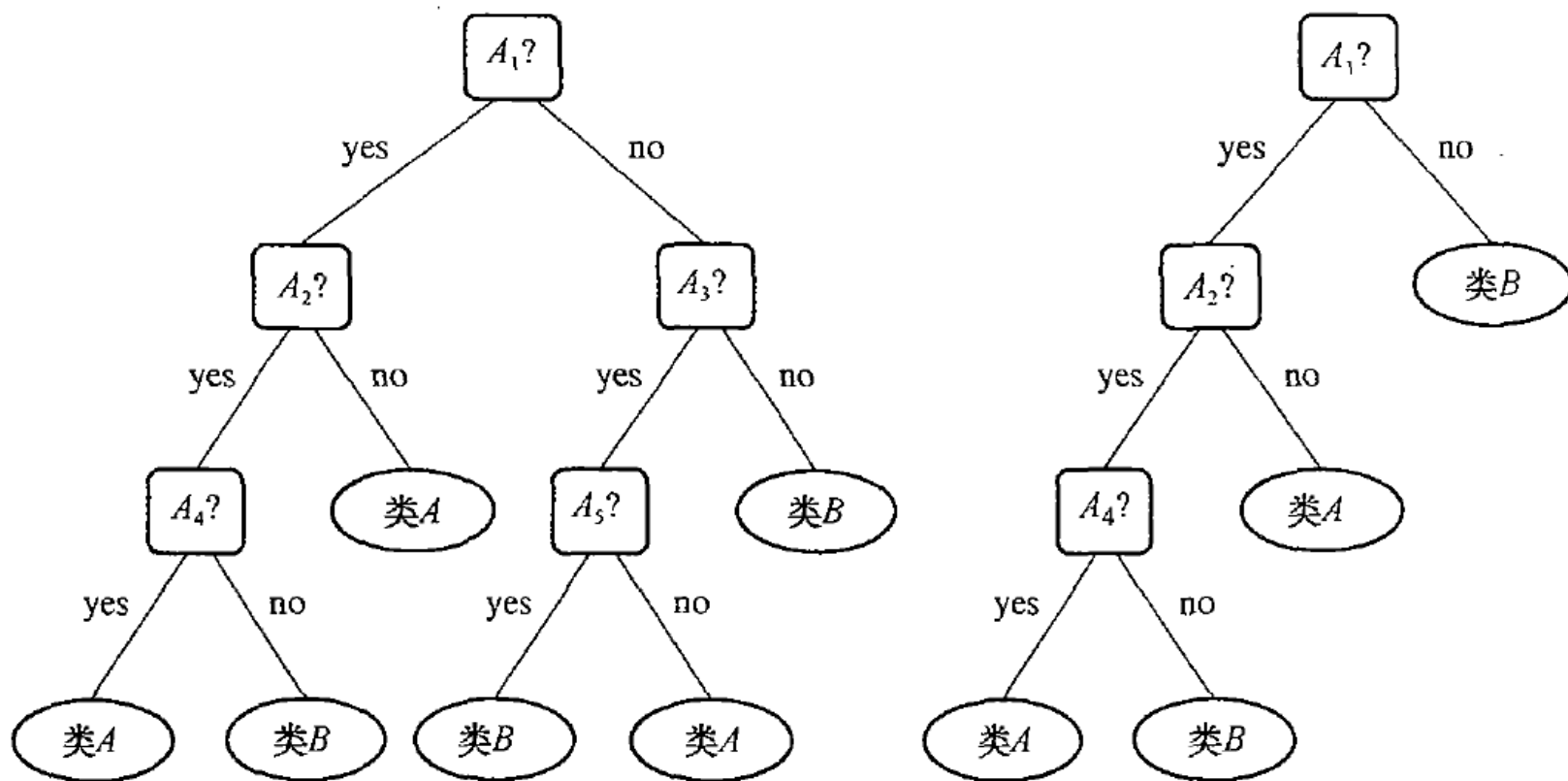


图 8.6 一棵未剪枝的决策树和它剪枝后的版本

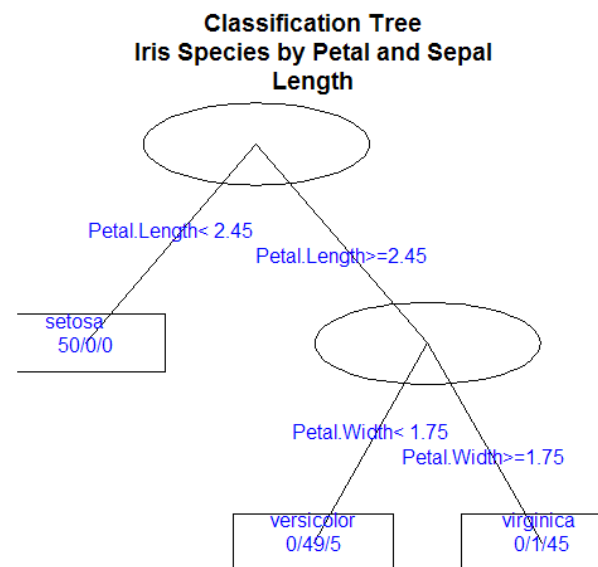
- 后剪枝：先产生完全的决策树，再进行裁剪。与之相对的做法是前剪枝
- 代价复杂度：叶节点个数（裁减对象）和树的错误率的函数
- 如果剪枝能使代价复杂度下降，则实施之
- 剪枝集

- <http://blog.csdn.net/tianguokaka/article/details/9018933>

- <http://blog.csdn.net/zjd950131/article/details/8027081>

■ 以鸢尾花数据集作为算例说明

```
iris.rp = rpart(Species~., data=iris,  
  method="class")  
  
plot(iris.rp, uniform=T, branch=0,  
  margin=0.1, main= " Classification  
Tree\nIris Species by Petal and Sepal  
Length")  
  
text(iris.rp, use.n=T, fancy=T, col="blue")
```



Rule 1: if $\text{Petal.Length} \geq 2.45 \& \text{Petal.Width} < 1.75$, then it is versicolor(0/49/5)
Rule2: if $\text{Petal.Length} \geq 2.45 \& \text{Petal.Width} \geq 1.75$, then it is virginica (0/1/45)
Rule 3: if $\text{Petal.Length} < 2.45$, then it is setosa (50/0/0)

怎样评估分类器效能？

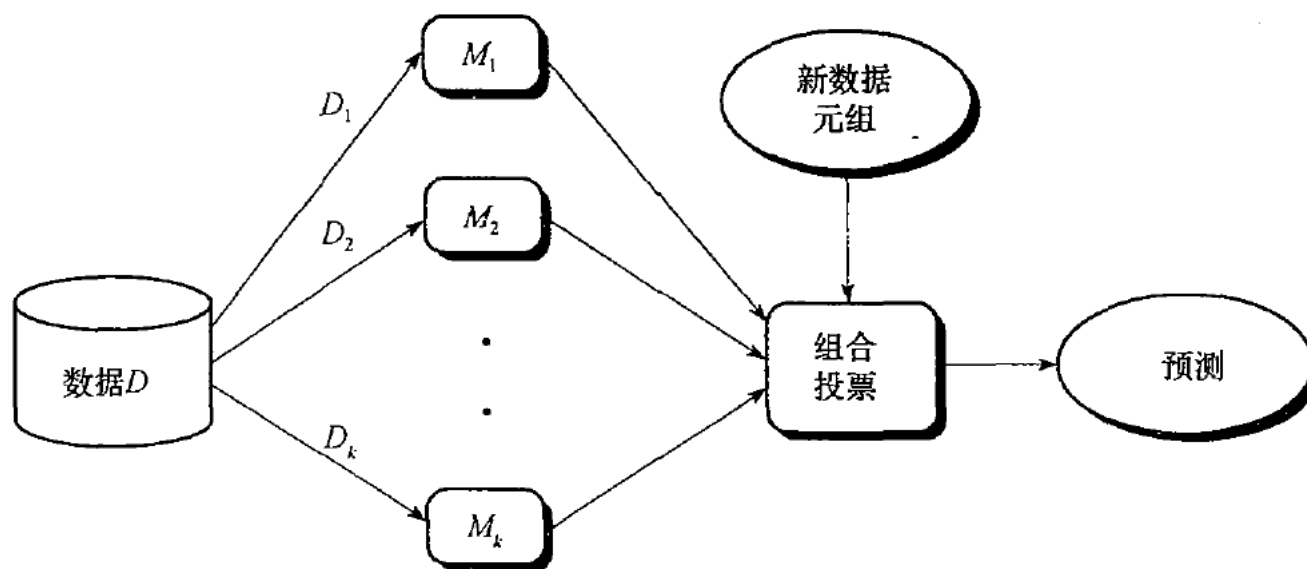
■ 韩家炜书第237页

度量	公式
准确率、识别率	$\frac{TP+TN}{P+N}$
错误率、误分类率	$\frac{FP+FN}{P+N}$
敏感度、真正例率、召回率	$\frac{TP}{P}$
特效性、真负例率	$\frac{TN}{N}$
精度	$\frac{TP}{TP+FP}$
F_1 、 F 分数 精度和召回率的调和均值	$\frac{2 \times precision \times recall}{precision+recall}$
F_β , 其中 β 是非负实数	$\frac{(1+\beta^2) \times precision \times recall}{\beta^2 \times precision+recall}$

		预测的类		
		yes	no	
实际的类	yes	TP	FN	P
	no	FP	TN	N
合计		P'	N'	P+N

图 8.13 评估度量。注意：某些度量有多个名称。 TP , TN , FP , FN , P , N 分别表示真正例、真负例、假正例、假负例、正和负样本数

- 组合方法包括：装袋（bagging），提升（boosting）和随机森林
- 基于学习数据集抽样产生若干训练集
- 使用训练集产生若干分类器
- 每个分类器分别进行预测，通过简单选举多数，判定最终所属分类



为什么组合方法能提高分类准确率？

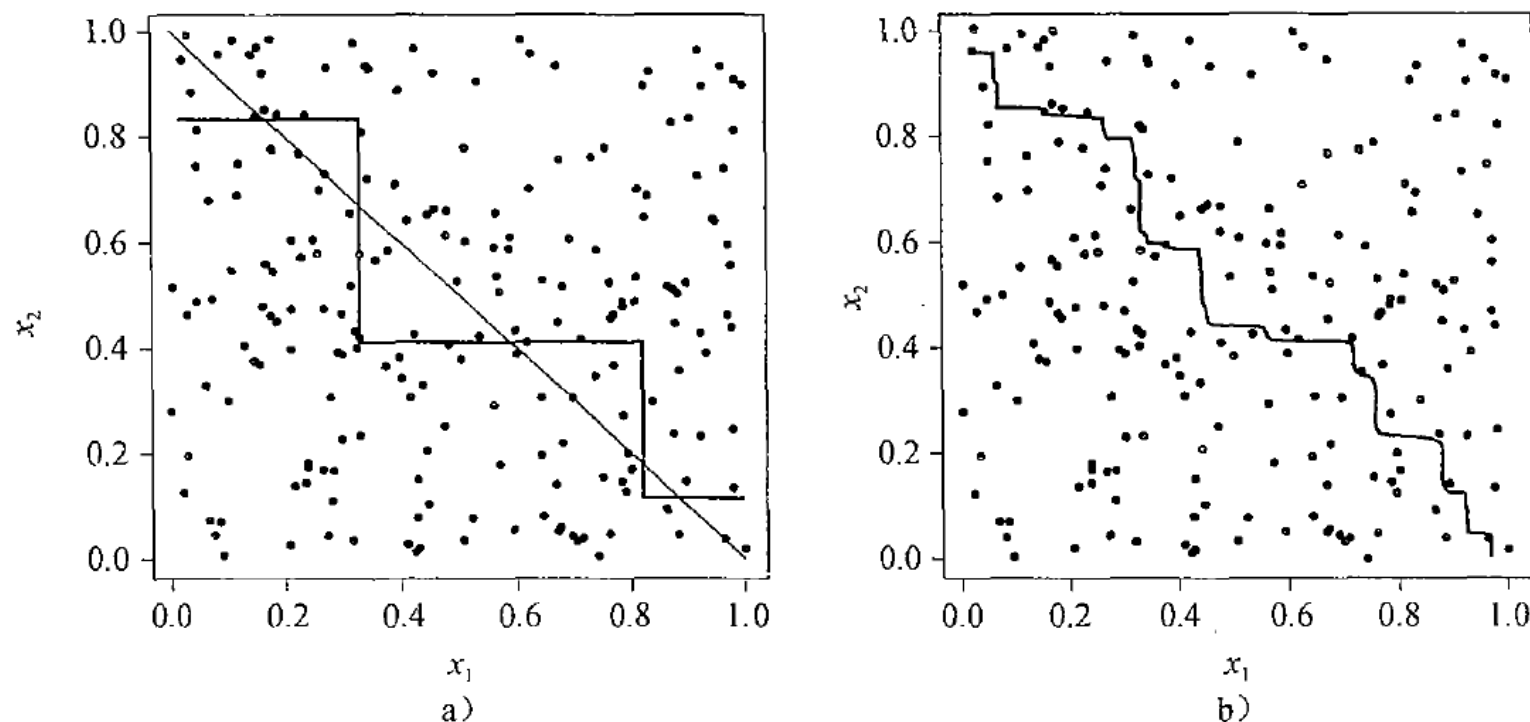


图 8.22 一个线性可分问题（即实际的决策边界是一条直线）的决策边界：a) 单棵决策树；b) 决策树的组合分类器。决策树努力近似线性边界。组合分类器更接近于真实的边界。取自 Seni 和 Elder[SE10]

- 能明显提升判别准确率
- 对误差和噪音更加鲁棒性
- 一定程度抵消过度拟合
- 适合并行化计算

算法：装袋。装袋算法——为学习方案创建组合分类模型,其中每个模型给出等权重预测。

输入：

- D : d 个训练元组的集合；
- k : 组合分类器中的模型数；
- 一种学习方案（例如,决策树算法、后向传播等）

输出：组合分类器—复合模型 M^* 。

方法：

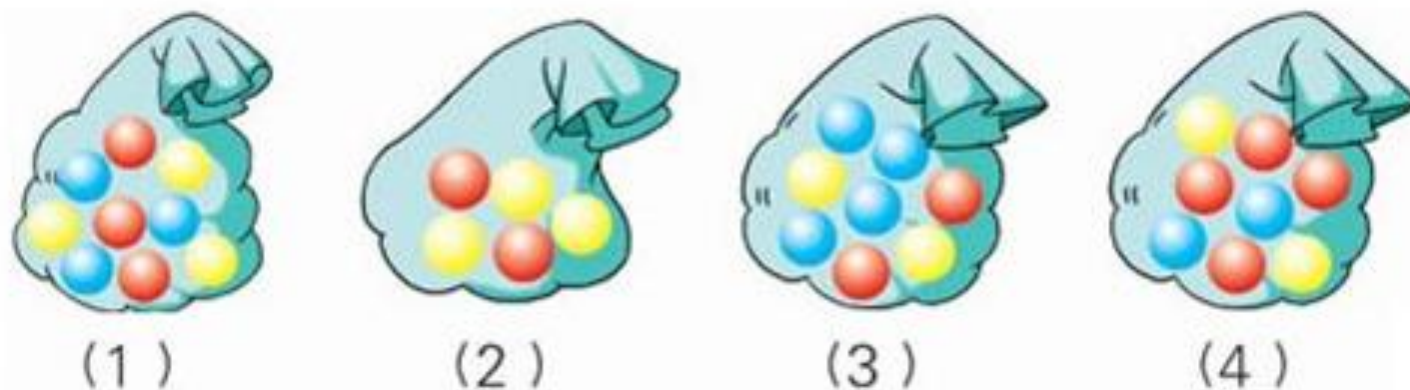
- (1) **for** $i = 1$ to k **do** // 创建 k 个模型
- (2) 通过对 D 有放回抽样, 创建自助样本 D_i ;
- (3) 使用 D_i 和学习方法导出模型 M_i ;
- (4) **endfor**

使用组合分类器对元组 x 分类：

让 k 个模型都对 x 分类并返回多数表决；

解释：有放回抽样与自助样本

- 有放回抽样
- 自助样本（bootstrap），韩家炜书第241页



- 准确率明显高于组合中任何单个的分类器
- 对于较大的噪音，表现不至于很差，并且具有鲁棒性
- 不容易过度拟合



- 训练集中的元组被分配权重
- 权重影响抽样，权重越大，越可能被抽取
- 迭代训练若干个分类器，在前一个分类器中被错误分类的元组，会被提高权重，使到它在后面建立的分类器里被更加“关注”
- 最后分类也是由所有分类器一起投票，投票权重取决于分类器的准确率

算法: Adaboost.一种提升算法——创建分类器的组合。每个给出一个加权投票。

输入:

- D : 类标记的训练元组集。
- k : 轮数(每轮产生一个分类器)。
- 一种分类学习方案。

输出: 一个复合模型。

方法:

- (1) 将 D 中每个元组的权重初始化为 $1/d$;
- (2) **for** $i = 1$ **to** k **do** // 对于每一轮
- (3) 根据元组的权重从 D 中有放回抽样,得到 D_i ;
- (4) 使用训练集 D_i 导出模型 M_i ;
- (5) 计算 M_i 的错误率 $\text{error}(M_i)$ (8.34式)
- (6) **if** $\text{error}(M_i) > 0.5$ **then**
- (7) 转步骤(3)重试;
- (8) **endif**
- (9) **for** D_i 的每个被正确分类的元组 **do**
- (10) 元组的权重乘以 $\text{error}(M_i) / (1 - \text{error}(M_i))$; // 更新权重
- (11) 规范化每个元组的权重;
- (12) **endfor**

使用组合分类器对元组 \mathbf{x} 分类:

- (1) 将每个类的权重初始化为0;
- (2) **for** $i = 1$ to k **do** // 对于每个分类器
- (3) $w_i = \log \frac{1 - \text{error}(M_i)}{\text{error}(M_i)}$; // 分类器的投票权重
- (4) $c = M_i(\mathbf{x})$; // 从 M_i 得到 \mathbf{x} 的类预测
- (5) 将 w_i 加到类 c 的权重;
- (6) **endfor**
- (7) 返回具有最大权重的类;

- 可以获得比bagging更高的准确率
- 容易过度拟合

随机森林 (Random Forest) 算法

- 由很多决策树分类器组合而成（因而称为“森林”）
- 单个的决策树分类器用随机方法构成。首先，学习集是从原训练集中通过有放回抽样得到的自助样本。其次，参与构建该决策树的变量也是随机抽出，参与变量数通常大大小于可用变量数。
- 单个决策树在产生学习集和确定参与变量后，使用CART算法计算，不剪枝
- 最后分类结果取决于各个决策树分类器简单多数选举

随机森林算法优点

- 准确率可以和Adaboost媲美
- 对错误和离群点更加鲁棒性
- 决策树容易过度拟合的问题会随着森林规模而削弱
- 在大数据情况下速度快，性能好

```
> library(randomForest)
randomForest 4.6-7
Type rfNews() to see new features/changes/bug fixes.
警告信息:
程辑包 'randomForest' 是用R版本3.0.3 来建造的
> model.forest <- randomForest(Species ~ ., data = iris)
> pre.forest = predict(model.forest, iris)
> table(pre.forest, iris$Species)

pre.forest   setosa versicolor virginica
  setosa      50          0           0
versicolor    0          50          0
 virginica    0          0          50
> library(rpart)
> model.tree = rpart(Species ~ ., data = iris, method = 'class')
> pre.tree = predict(model.tree, data = iris, type = 'class')
> table(pre.tree, iris$Species)

pre.tree      setosa versicolor virginica
  setosa       50          0           0
versicolor     0          49           5
 virginica     0          1          45
> |
```

- Dataguru（炼数成金）是专业数据分析网站，提供教育，媒体，内容，社区，出版，数据分析业务等服务。我们的课程采用新兴的互联网教育形式，独创地发展了逆向收费式网络培训课程模式。既继承传统教育重学习氛围，重竞争压力的特点，同时又发挥互联网的威力打破时空限制，把天南地北志同道合的朋友组织在一起交流学习，使到原先孤立的学习个体组合成有组织的探索力量。并且把原先动辄成千上万的学习成本，直线下降至百元范围，造福大众。我们的目标是：低成本传播高价值知识，构架中国第一的网上知识流转阵地。
- 关于逆向收费式网络的详情，请看我们的培训网站 <http://edu.dataguru.cn>

Thanks

FAQ时间