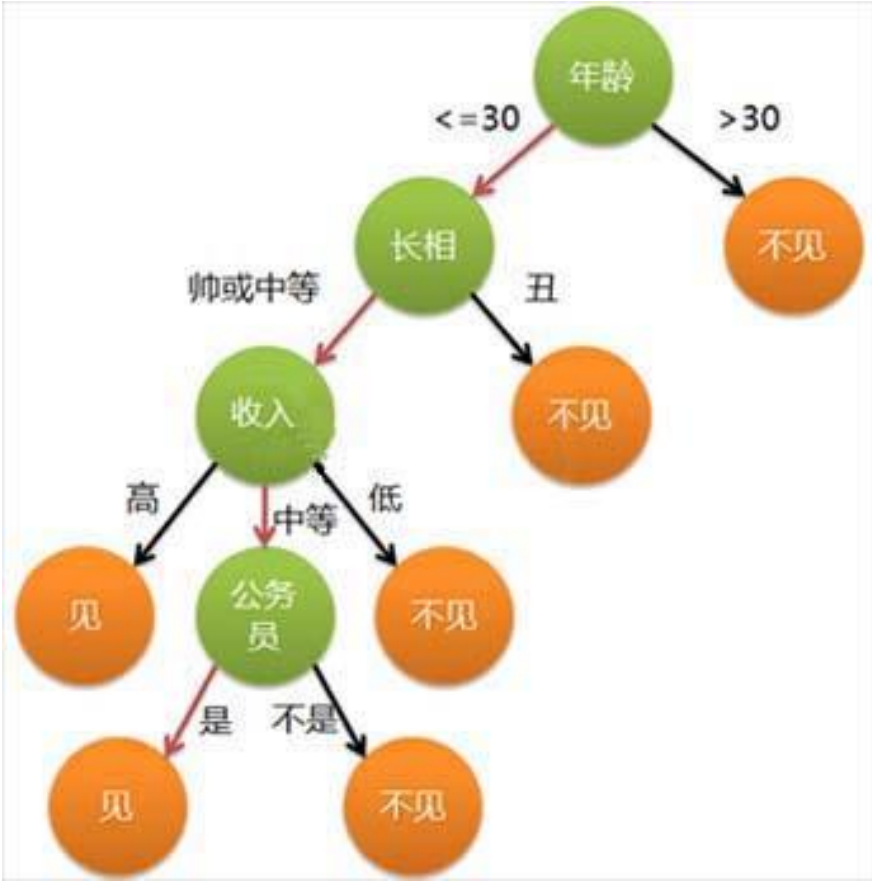
决策树 decision tree

- 什么是决策树
- 输入：学习集
- 输出：分类规则（决策树）

  输出没有公式或判别函数，只是树

决策树善于处理离散型数据，最好提前把连续性变量变为离散型变量



决策树算法概述

- 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

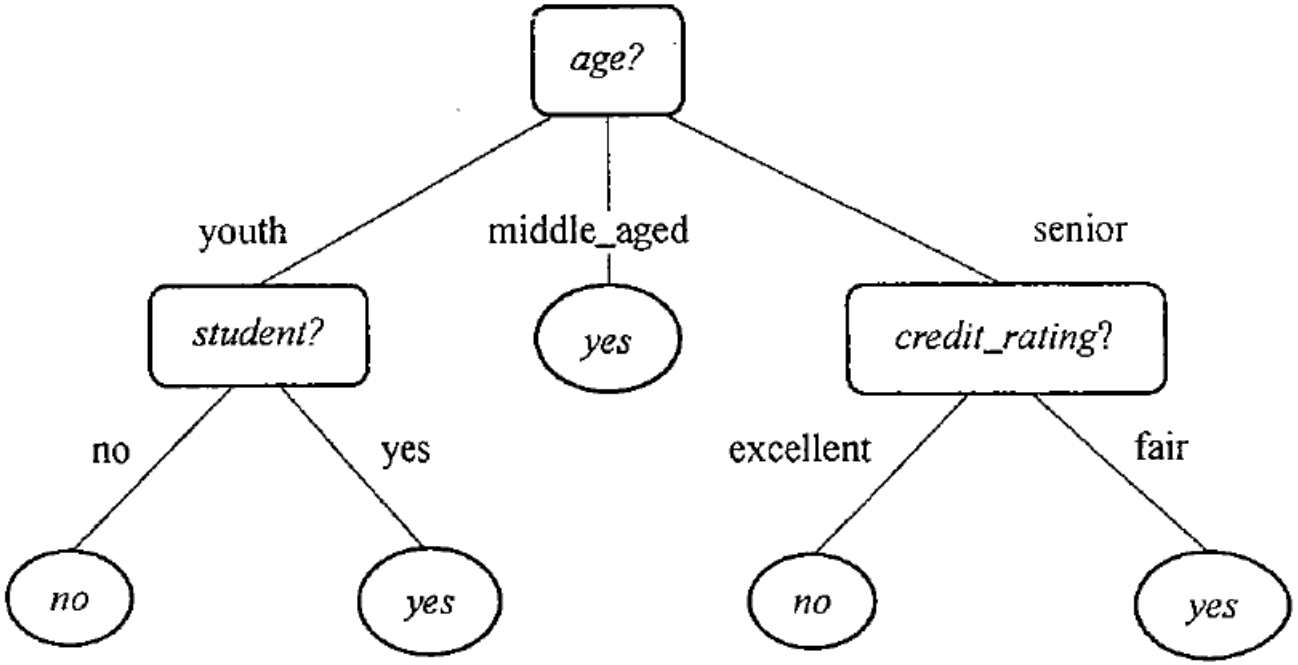
决策树涉及到剪枝问题，而随机森林没有这个问题

例子：训练数据

表 8.1　AllElectronics 顾客数据库标记类的训练元组

RID	*age*	*income*	*student*	*credit_rating*	*Class*：*buys_computer*
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


age?
youth
middle_aged
senior
student?
yes
credit_rating?
no
yes
excellent
fair
no
yes
no
yes

算法的核心问题

- 该按什么样的次序来选择变量（属性）？ 依据为:计算信息增益
- 最佳分离点（连续的情形）在哪儿？

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。

ID3算法

- 信息增益计算

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

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

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

某一变量的信息增益 = 总体样本的期望信息 - 按照某一变量分类下的期望信息

信息增益也可以理解为熵，即表示一类东西的有序程度。越有序，熵越高

例子：信息增益计算

总体样本的期望信息

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

按照age分类下的期望信息

$$Info_{age}(D) = \frac{5}{14}\times\left(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right) + \frac{4}{14}\times\left(-\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4}\right) + \frac{5}{14}\times\left(-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}\right)$$

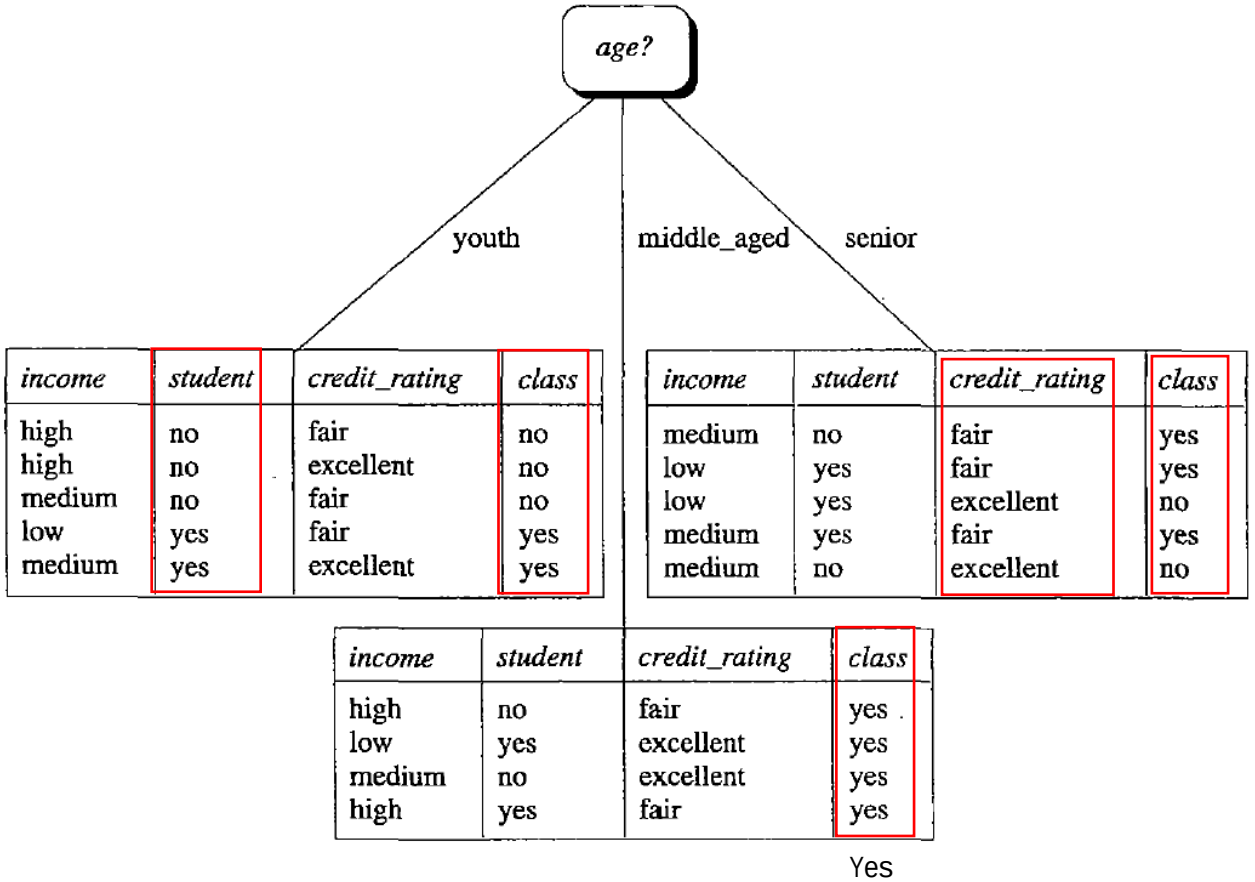
$$= 0.694 \text{ 位}$$

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

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

“但是，如何计算连续值属性的信息增益？”假设属性 A 是连续值的，而不是离散值的。（例如，假定有属性 age 的原始值，而不是该属性的离散化版本。）对于这种情况，必须确定 A 的“最佳”**分裂点**，其中分裂点是 A 上的阈值。

首先，将 A 的值按递增序排序。典型地，每对相邻值的中点被看做可能的分裂点。这样，给定 A 的 v 个值，则需要计算 $v-1$ 个可能的划分。例如，A 的值 a_i 和 a_{i+1} 之间的中点是

$$\frac{a_i + a_{i+1}}{2} \tag{8.4}$$

如果 A 的值已经预先排序，则确定 A 的最佳划分只需要扫描一遍这些值。对于 A 的每个可能分裂点，计算 $Info_A(D)$，其中分区的个数为 2，即（8.2）式中 $v=2$（或 $j=1, 2$）。A 具有最小期望信息需求的点选做 A 的分裂点。D_1 是满足 $A \leqslant split\_poin$ 的元组集合，而 D_2 是满足

也就是信息增益最大的点为分裂点

A>split_point的元组集合

例子

- 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

ASP.NET学习网
www.17aspx.com

信息增益

- 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。

$$info(D) = -0.7log_20.7 - 0.3log_20.3 = 0.7 * 0.51 + 0.3 * 1.74 = 0.879$$

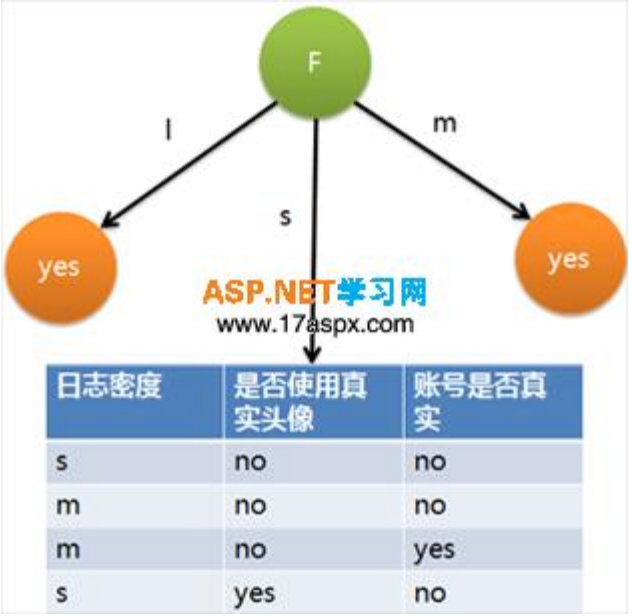
$$info_L(D) = 0.3 * (-\frac{0}{3}log_2\frac{0}{3} - \frac{3}{3}log_2\frac{3}{3}) + 0.4 * (-\frac{1}{4}log_2\frac{1}{4} - \frac{3}{4}log_2\frac{3}{4}) + 0.3 * (-\frac{1}{3}log_2\frac{1}{3} - \frac{2}{3}log_2\frac{2}{3}) = 0 + 0.326 + 0.277 = 0.603$$

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

ASP.NET学习网
www.17aspx.com

根据信息增益选择分裂属性

- 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：



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

递归+分而治之

- 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。

C4.5算法

ID3算法利用信息增益，其缺点为

- 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

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

这一项是用来平衡因子数量的

$$GrianRate(A) = \frac{Grain(A)}{SplitInfo_A(D)}$$

例子

- 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，$GainRatio(income) = 0.029/1.557 = 0.019$。 ■

CART算法

rpart包用的就是CART算法

- 使用基尼指数选择变量

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

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

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

例子

- 韩家炜书第221页

剪枝

C4.5和CART有剪枝，ID3没有剪枝过程

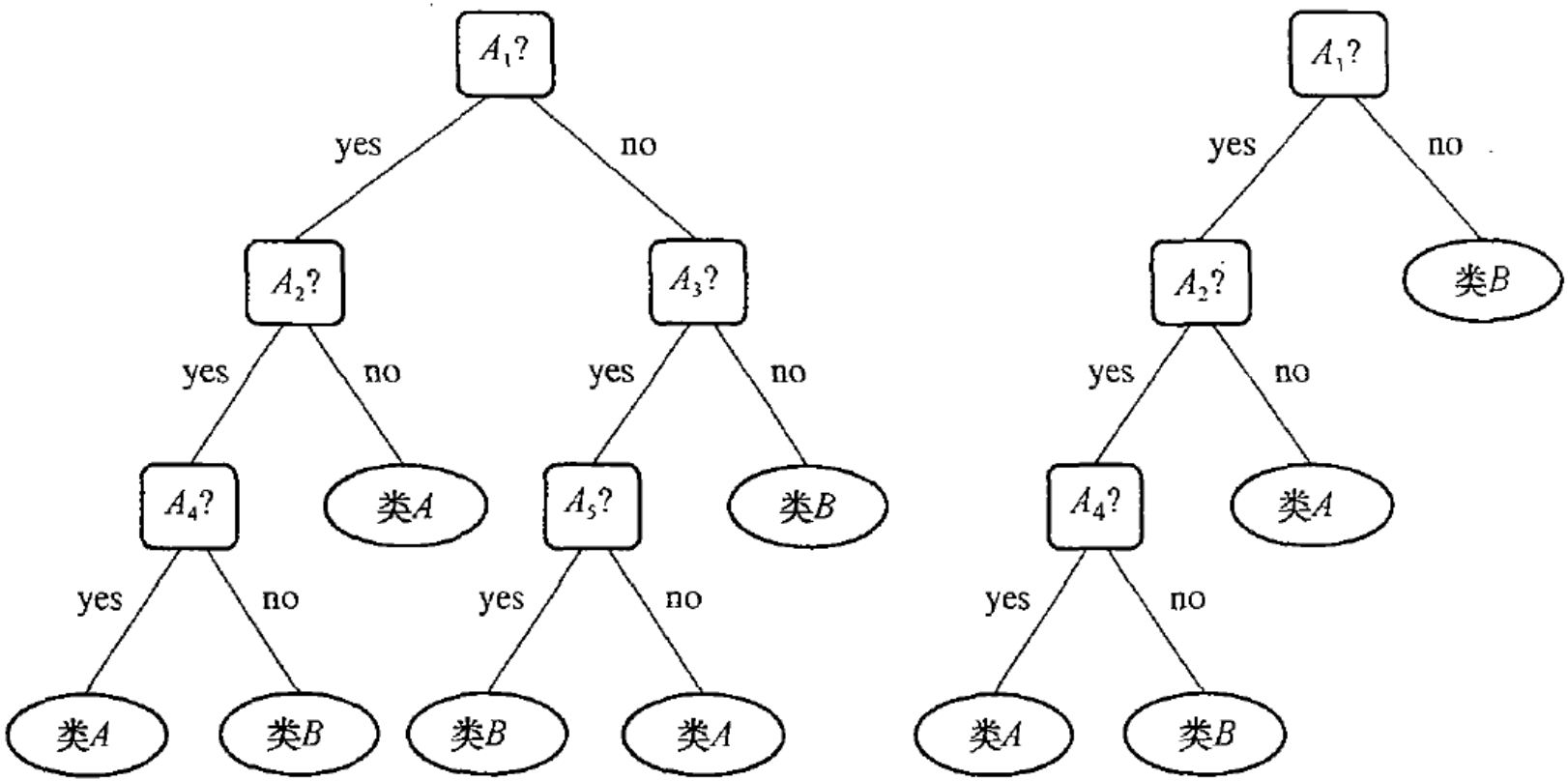


图 8.6　一棵未剪枝的决策树和它剪枝后的版本

CART的剪枝

前剪枝：先选择变量，再构建决策树模型

- 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

代价复杂度剪枝 C4.5剪枝

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

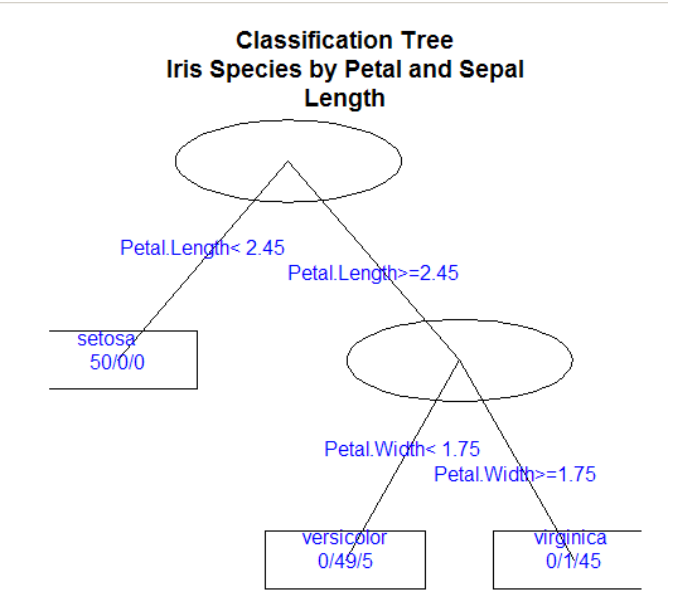
C4.5的悲观剪枝法

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

R语言实现决策树：rpart扩展包

- 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 Petal.Length>=2.45&Petal.Width<1.75, then it is versicolor(0/49/5)
Rule2: if Petal.Length>=2.45&Petal.Width>=1.75, then it is virginica (0/1/45)
Rule 3: if 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、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

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

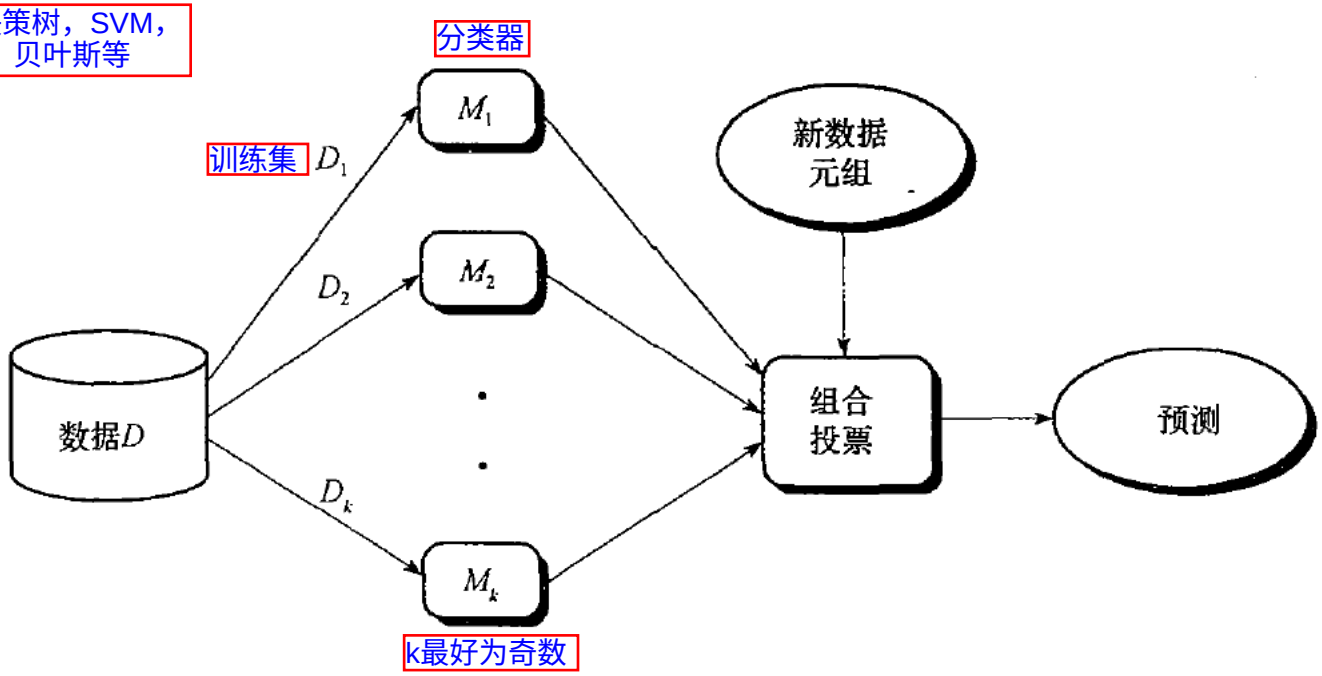
图 8.13　评估度量。注意：某些度量有多个名称。TP，TN，FP，FN，P，N 分别表示真正例、真负例、假正例、假负例、正和负样本数

提升分类器准确率的组合方法

- 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

两个核心问题：
如何抽样？选取什么样的抽样方法？
新学习集如何训练分类器？

分类器可为决策树，SVM，Logistic回归，贝叶斯等



k最好为奇数

为什么组合方法能提高分类准确率？

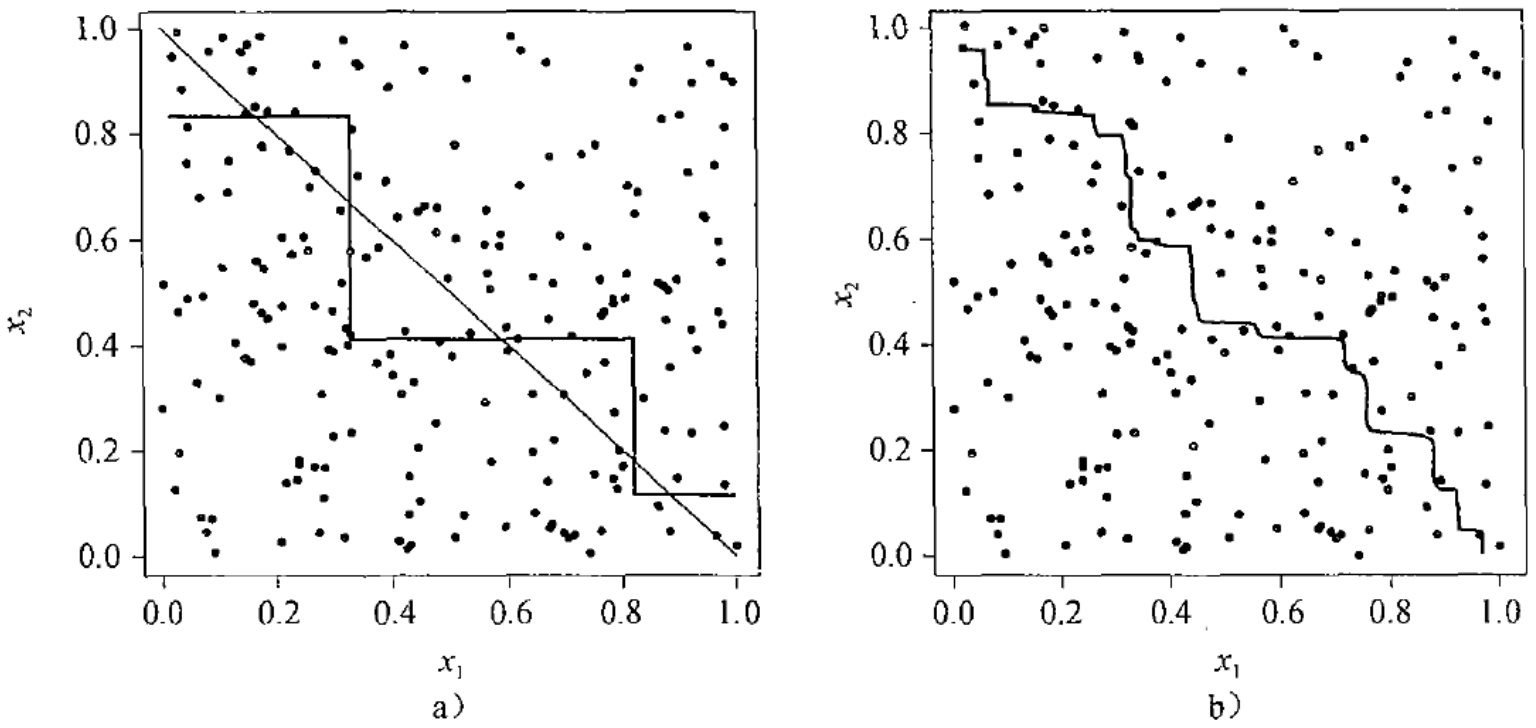


图 8.22　一个线性可分问题（即实际的决策边界是一条直线）的决策边界：a）单棵决策树；b）决策树的组合分类器。决策树努力近似线性边界。组合分类器更接近于真实的边界。取自 Seni 和 Elder[SE10]

组合算法的优势

- 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

装袋算法

bagging: bootstrap aggregating的缩写

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；

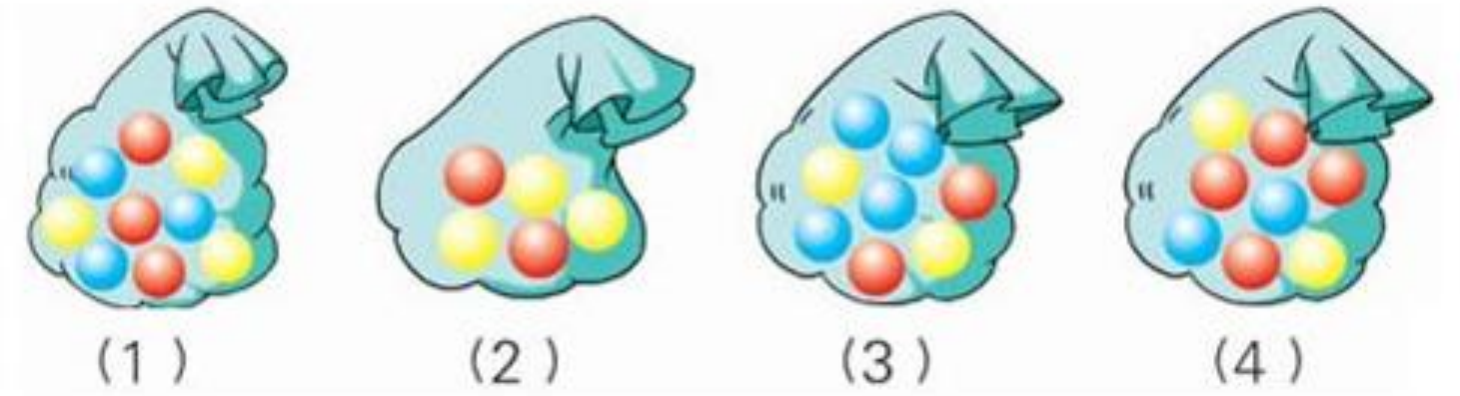
（4）**endfor**

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 有重复抽样



袋装算法的优势

- 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

提升（boosting）算法思想

其中主要的是AdaBoost（Adaptive Boosting）

- 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

Adaboost算法

Adaptive Boosting

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$error(M_i)$（8.34式）
(6) **if** $error(M_i) > 0.5$ **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$error(M_i)/(1-error(M_i))$; // 更新权重
(11) 规范化每个元组的权重;
(12) **endfor**

Adaboost算法

使用组合分类器对元组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) $w_i = \log \frac{1-error(M_i)}{error(M_i)}$； // 分类器的投票权重

(4) $c = M_i(x)$； // 从M_i得到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；

提升算法的优缺点

- 可以获得比bagging更高的准确率
- 容易过度拟合

Bagging与Boosting的区别：二者的主要区别是取样方式不同。
1.Bagging采用均匀取样，而Boosting根据错误率来取样，因此Boosting的分类精度要优于Bagging;
2.Bagging的训练集的选择是随机的，各轮训练集之间相互独立，而Boostlng的各轮训练集的选择与前面各轮的学习结果有关;
3.Bagging的各个预测函数没有权重，而Boosting是有权重的;
4.Bagging的各个预测函数可以并行生成，而Boosting的各个预测函数只能顺序生成。对于象神经网络这样极为耗时的学习方法。Bagging可通过并行训练节省大量时间开销;
5.bagging和boosting都可以有效地提高分类的准确性。在大多数数据集中，boosting的准确性比bagging高。在有些数据集中，boosting会引起退化--- Overfit。
6. Boosting思想的一种改进型AdaBoost方法在邮件过滤、文本分类方面都有很好的性能。

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”）也可以使用SVM, Logistic回归等其他分类器，不过仍然习惯性地称为随机森林
- 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

抵消随机误差

有放回的选取一定比例的样本是为了解决异常样本的问题，那些没选到异常样本的数据集训练结果会更好。

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

R的randomForest包

```
> 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
>
```