数据挖掘与机器学习

1

潘斌

panbin@nankai.edu.cn

范孙楼227

上节回顾

- 贝叶斯分类器
- 典型分类器：朴素贝叶斯
- 贝叶斯网络
- 贝叶斯神经网络

本节提要

- 非线性分类器

5 非线性分类器

- 5.1 多类问题概述
- 5.2 最小距离分类器
- 5.3 分段线性分类器概述
- 5.5 近邻法分类器
- 5.4 决策树
- 5.6 人工神经网络
- 5.7 SVM
- 5.8 Adaboost

5.1 多类问题概述

- **5.1.1 多类问题**
- **5.1.2 解决方案**

5.1.1 多类问题概述

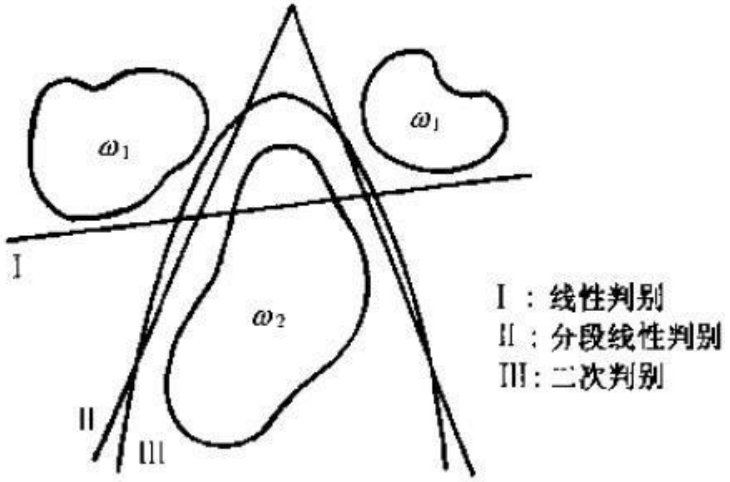
- **多类问题包括**
  - 多类情况（类别数C > 2）
    - 单峰分布
    - 多峰分布

5.1.1 多类问题

- **多类问题包括**
  - 两类情况（C = 2）
    - 样本集具有多峰分布

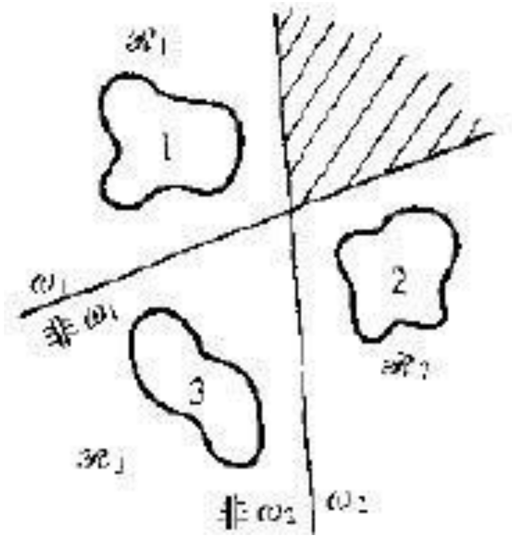
5.1.2 解决方案

- **如何解多类问题**
  - Bayes分类器
    - 二次型判别函数
  - 线性分类器
  - 其它分类器



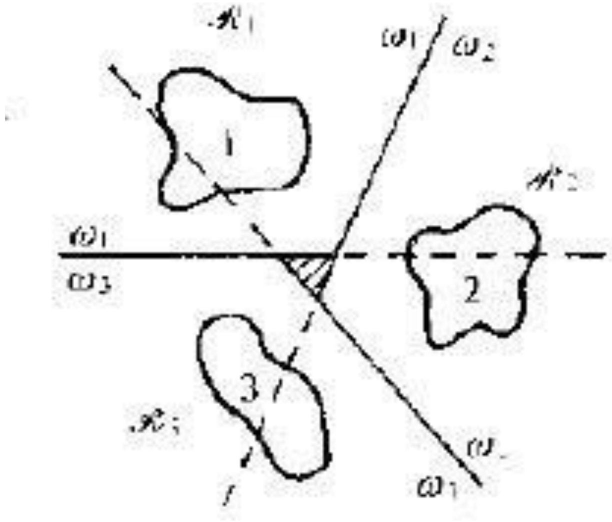
5.1.2 解决方案

- **多类问题是否可以用线性分类器解？**
  - 思路一
    - 对“类与类的非”进行线性分类
    - 只需要C – 1个线性分类器就可以



5.1.2 解决方案

- **多类问题是否可以用线性分类器解？**
  - 思路二
    - “两两分类”进行线性分类
    - 需要C(C – 1)/2个线性分类器就可以

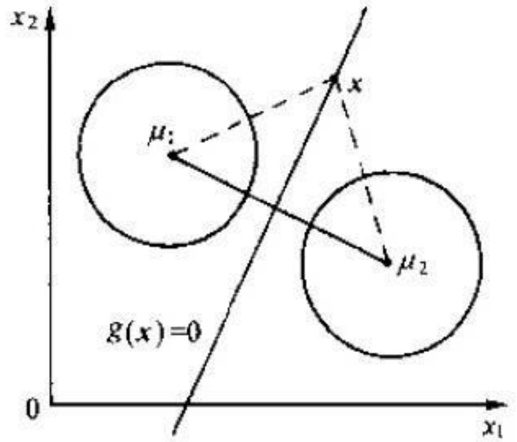


5.2 最小距离分类器

- 5.2.1 最小距离分类器原理
- 5.2.2 分段最小距离分类器
- 5.2.3 特点

5.2.1 最小距离分类器原理

- **回顾两类单峰线性分类器**
  - 垂直平分 / 最小距离分类器
  - 基于两类样本均值点作垂直平分线

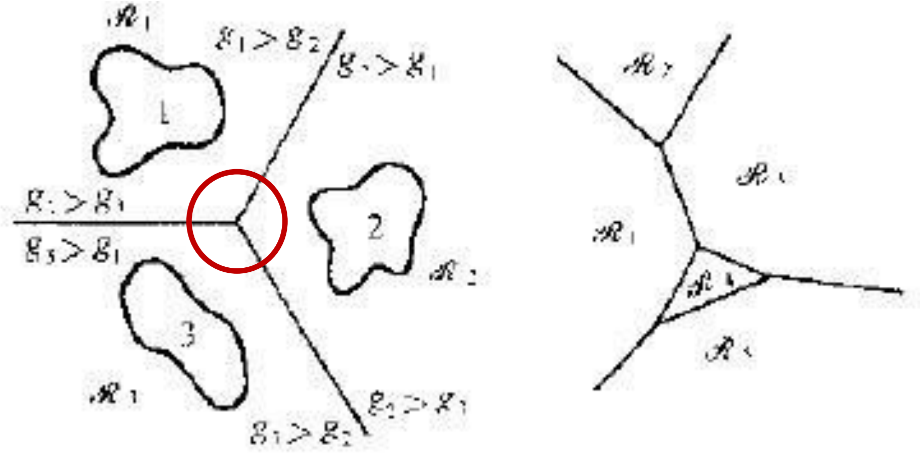


5.2.1 最小距离分类器原理

- **其最小距离形式**
  - 判别函数
    - $G_1(x) = d_1(x) = \|x - m_1\|$
    - $G_2(x) = d_2(x) = \|x - m_2\|$
  - 决策规则
    - 对于未知样本x，若$d_1(x) < d_2(x)$，则x决策为ω_1类
    - 若$d_1(x) > d_2(x)$，则x决策为ω_2类

5.2.1 最小距离分类器原理

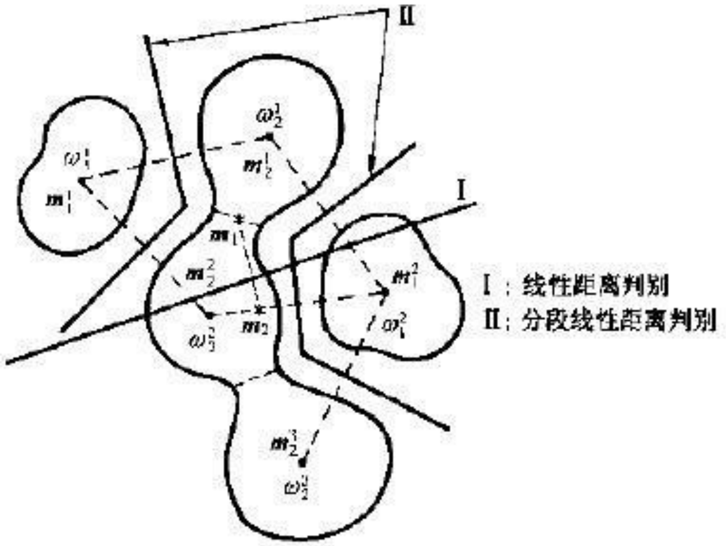
- **直接使用可以解决多类问题**
  - 解决C类单峰问题



(a) 三类　　(b) 五类

5.2.1 最小距离分类器原理

- 直接使用可以解决多类问题
  – 解决两类多峰问题



5.2.2 分段最小距离分类器

- **问题**
  - 已知各类及其子类
  - 求分段最小距离分类器

5.2.2 分段最小距离分类器

- **分类器设计**
  - 先求各子类均值
    - m_{ij} （ω_i类的第j子类）
  - 定义各类判别函数
    - $G_i(x) = \min\limits_{j} \|x - m_{ij}\|$
  - 决策规则
    - 对于未知样本x，若$G_k(x) = \min\limits_{i} G_i(x)$，则x决策为$\omega_k$类

5.2.3 特点

- **分类器特点**
  - 解决两类多峰或多类问题的分段线性分类器
  - 可以解决几乎所有分类问题但要已知各类子类
  - 概念直观简单，未经优化
  - 分类器设计简单容易
  - （无重叠区或空白区）

5.3 分段线性分类器概述

- 5.3.1 问题与思路
- 5.3.2 设计说明

5.3.1 问题与思路

- **思路**
  - 参考分段最小距离分类器
    - 定义判别函数
    - 定义决策规则

5.3.1 问题与思路

- **针对不同已知条件**
  - 1、已知各类子类个数及子类分布区域
  - 2、已知各类子类个数（分布区域不知）
  - 3、一般情况（子类个数和分布区域均不知）

5.3.2 设计说明

- **两种判别函数的区别**
  - 小写g函数（分界面）
    - 每段设计一个g函数，容易做
    - 多个分段，如何判断正负侧，需要特殊规则
  - 大写G函数（计算值）
    - 每个子类设计一个G函数，需要知道类别分布区域
    - 直接计算Max或Min，判别规则简单

5.3.2 设计说明

- **设计关键**
  - 如何确定各类的子类个数
  - 如何确定子类的分布区域
  - 如何求解各子类的权向量和阈值权
  - 若采用小写g函数，决策规则如何

5.5 近邻法分类器

- 5.5.1 近邻法原理
- 5.5.2 最近邻法
- 5.5.3 k-近邻法
- 5.5.4 近邻法分类器错误率

5.5.1 近邻法原理

- **近邻法分类是一种简单实用的分类方法**
  - 分段线性分类器

5.5.1 近邻法原理

- **最小距离分类器**
  - 用均值点作为代表点，按最近均值点进行决策
  - 有时候均值点不具有很好的代表性
- **近邻法分类器思路**
  - 全部训练样本都是代表点，按最近邻点进行决策

5.5.2 最近邻法

- **问题**
  - 设C类问题：$\omega_1, \omega_2, \ldots, \omega_C$
  - ω_i类样本集$Z_i = \{\ldots, x_{ik}, \ldots\}$

  - 求近邻法分类器

5.5.2 最近邻法

- **判别函数**
  - 定义$G_i(x) = \min \| x - x_{ik} \|$　　　　$i = 1, 2, \ldots, C$
- **决策规则**
  - 对于未知样本x，若$G_j(x) = \min G_i(x)$，则$x \in \omega_j$
- **决策面**

5.5.2 最近邻法

- **实例**
  - 甲类：$[0\ \ 3]^T$、$[2\ \ 4]^T$、$[1\ \ 3]^T$、$[2\ \ 3]^T$、$[0\ \ 2]^T$
  - 乙类：$[4\ \ 1]^T$、$[3\ \ 2]^T$、$[2\ \ 1]^T$、$[3\ \ 0]^T$、$[3\ \ 1]^T$
  - 待分类样本为$x = [5\ \ 0]^T$，问x应决策为哪一类？

5.5.2 最近邻法

- **近邻法特点**
  - 可以解决几乎所有分类问题
  - 概念直观简单未经优化，但错误率并不高
  - 分类器设计容易
  - 运算量大，需要设计快速算法

5.5.2 最近邻法

- **快速算法**
  - 剪辑法
    - 进行预分类
    - 剪辑掉错分样本
    - 剪辑法可以重复进行

5.5.2 最近邻法

- **快速算法**
  - 剪辑法例一

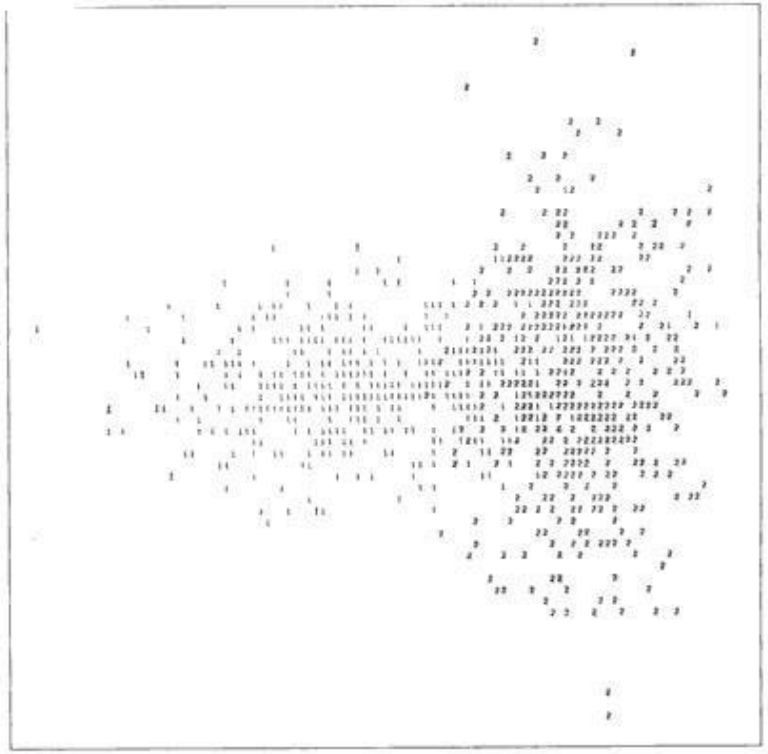


图 6.5 MULTIEDIT 算法实验:原始样本集

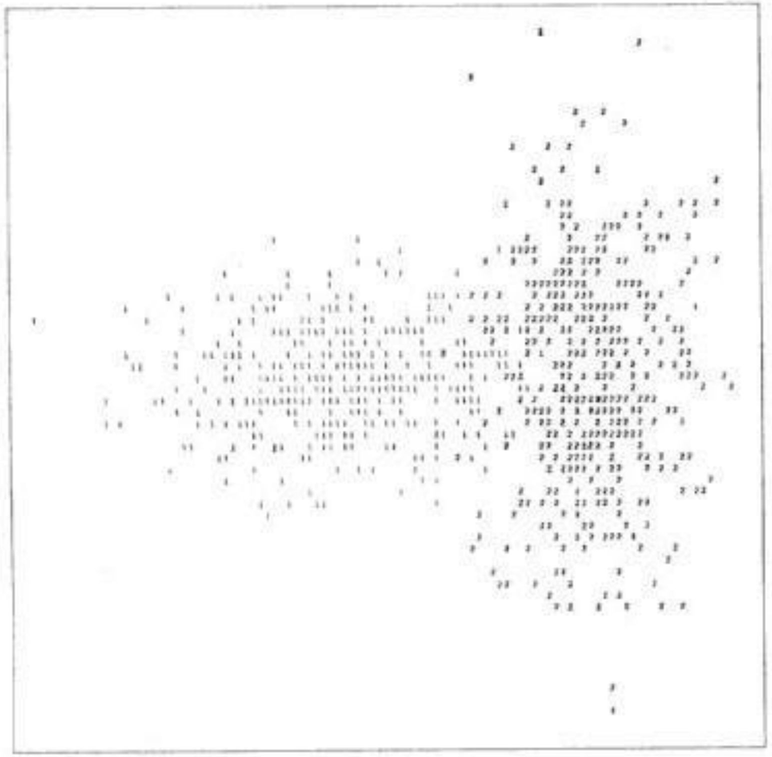


图 6.6 MULTIEDIT 算法实验:第一次迭代后留下的样本

5.5.2 最近邻法

- **快速算法**
  - 剪辑法例一

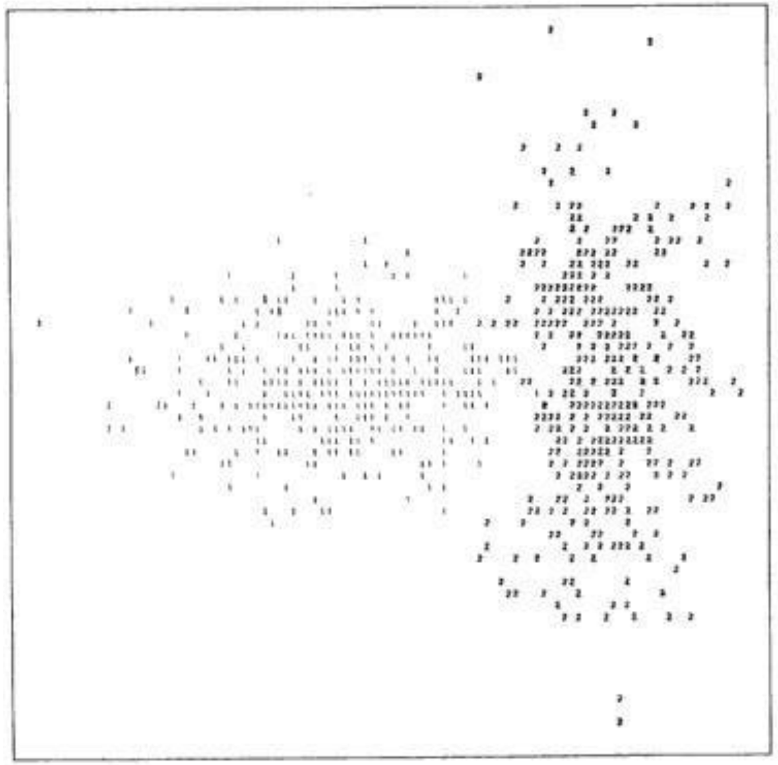


图 6.7　MULTIEDIT 算法实验：经三次迭代后留下的样本

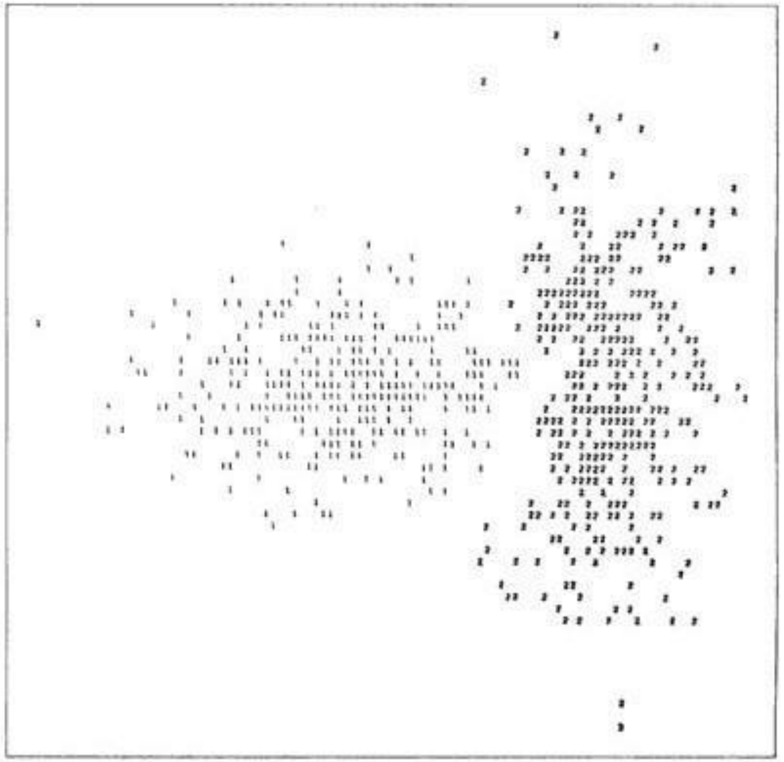


图 6.8　MULTIEDIT 算法实验：算法终止时留下的样本

5.5.2 最近邻法

- **快速算法**
  - 剪辑法例二

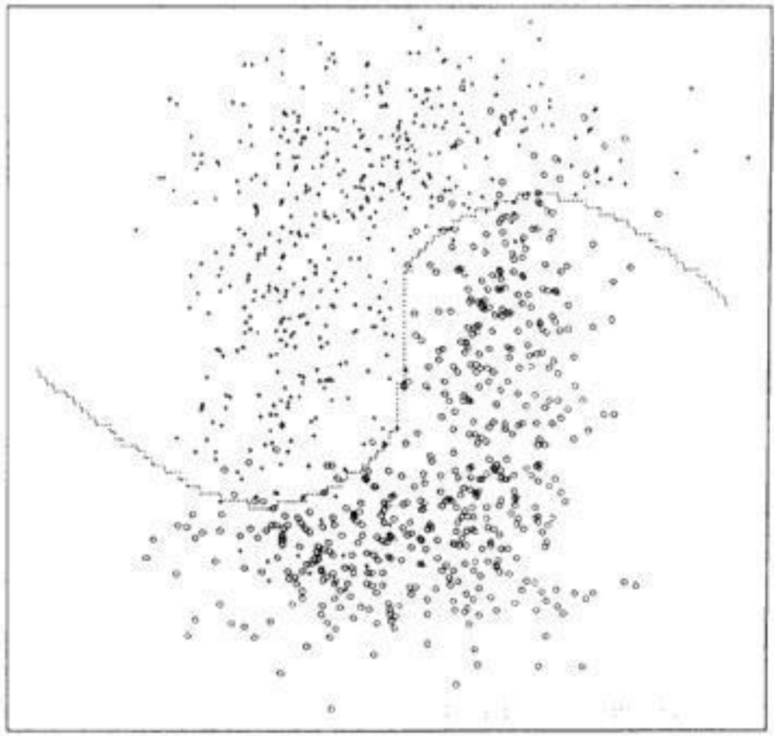


图 6.9 非正态分布下 MULTIEDIT 重复剪辑实验:初始样本集

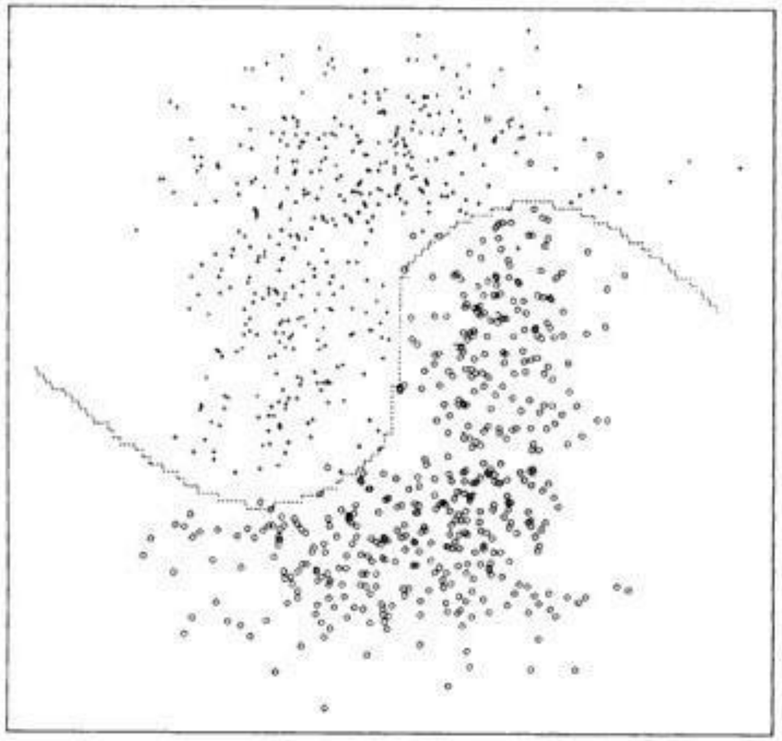


图 6.10 非正态分布下 MULTIEDIT 重复剪辑实验:第一次剪辑后的样本集

5.5.2 最近邻法

- **快速算法**
  - 剪辑法例二

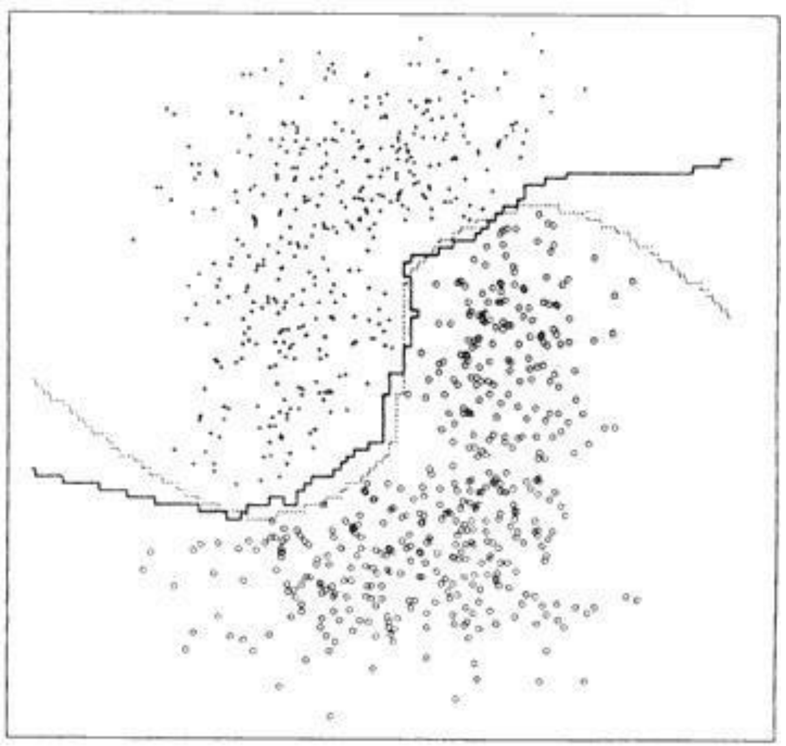


图 6.11 非正态分布下 MULTIEDIT 重复剪辑实验：最终结果

5.5.2 最近邻法

- **快速算法**
  - 压缩法
    - 大多数样本参与计算，但却不起决定作用，是多余样本。
  - 压缩法步骤
    - 每类各取一个代表样本（例如均值点近邻），组成压缩样本集Z_s 。
    - 以当前Z_s对样本集做最近邻分类，然后将错分样本放入Z_s 。
    - 重复上述步骤，直至无样本放入为止。

5.5.2 最近邻法

- **快速算法**
  - 压缩法示例

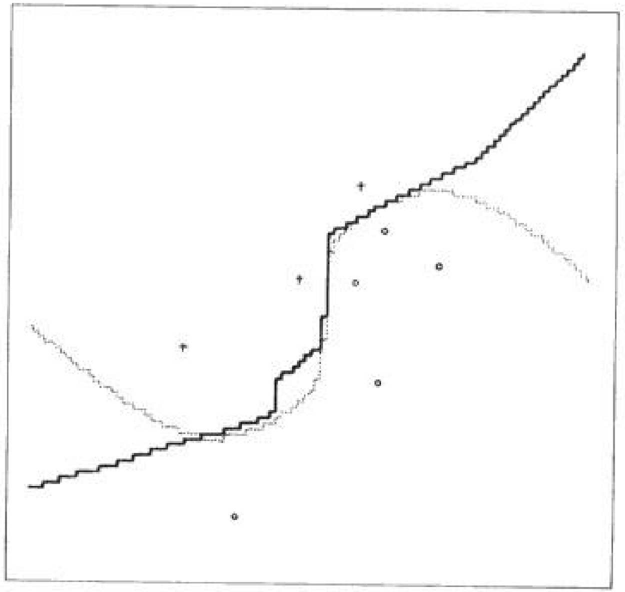


图 6.12 图 6.9 的数据经 MULTIEDIT 算法剪辑之后再使用 CONDENSING 压缩近邻算法的结果

5.5.3 k-近邻法

- **最近邻法的问题**
  - 噪点干扰

- **k-近邻法**

5.5.3 k-近邻法

- **判别函数**
  - 定义$G_i(x) = k_i$ $\quad\quad\quad\quad$ $i = 1, 2, \ldots, C$
- **决策规则**
  - 对于未知样本x，若$G_j(x) = \max G_i(x)$，则$x \in \omega_j$

5.5.4 近邻法分类错误率

- **近邻法分类错误率定义**
  - 最近邻法
    - 设有N个训练样本
    - 未知样本x的最近邻为x'
    - 再设最小错误率Bayes分类器的错误率为P*
  - 则平均错误率定义为

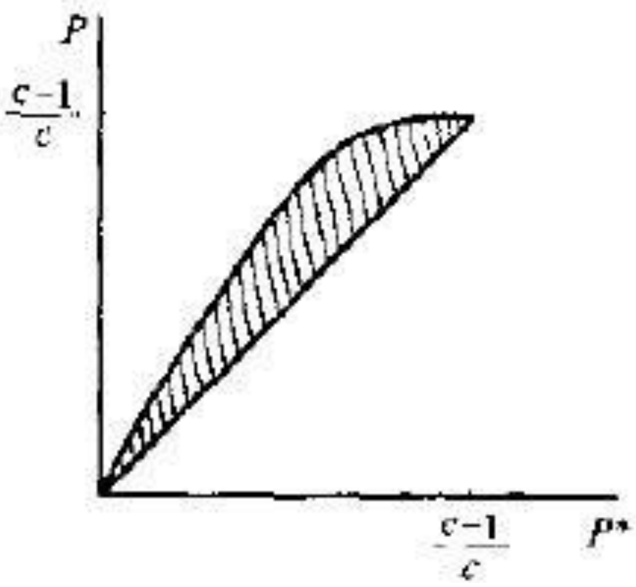
$$P_N(e) = \int\int P_N(e \mid x, x') p(x' \mid x) dx' p(x) dx$$

$$P(e) = \lim_{N \to \infty} P_N(e)$$

5.5.4 近邻法分类错误率

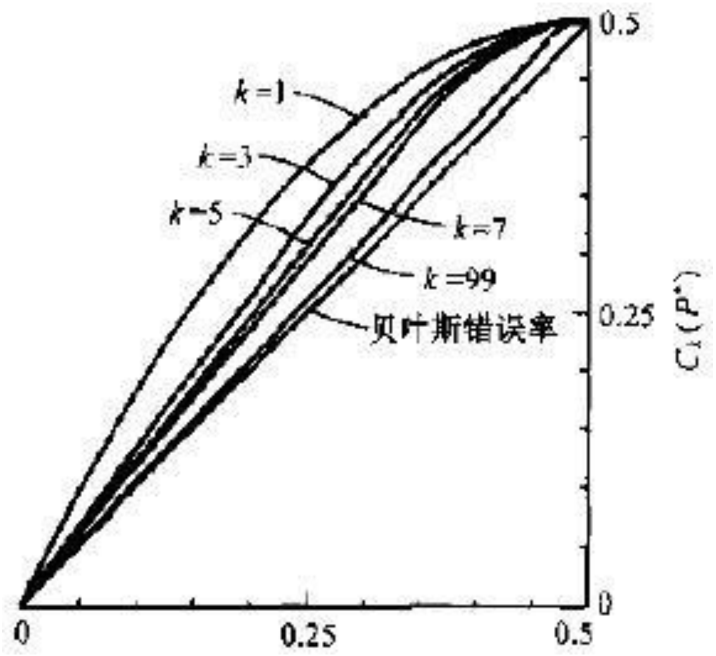
- **近邻法错误率上下界**
  - 最近邻法

$$P^* \le P \le P^*\left(2 - \frac{c}{c-1}P^*\right)$$



5.5.4 近邻法分类器错误率

- 近邻法错误率上下界
  - k-近邻法



练习

- **4、已知**
  - 甲类样本4个：$[2\ \ 2]^T$、 $[2\ \ 3]^T$、 $[1\ \ 2]^T$、 $[2\ \ 1]^T$
  - 乙类样本4个：$[-2\ \ -2]^T$、 $[-3\ \ -2]^T$、 $[-1\ \ -2]^T$、 $[-2\ \ -3]^T$

  - 试用3近邻分类器对未知样本$[-1\ \ -1]^T$和$[3\ \ 2]^T$进行分类。

决策树

Decision Tree

决策树的组织形式

- 通过一系列精心构思的关于测试记录属性的问题，可以解决分类问题。
- 每当一个问题得到答案，后继问题随即而来，直到得到记录的类标号。
- 决策树是一种非参数的监督学习方法，它主要用于分类和回归。目的是构造一种模型，使之能够从样本数据的特征属性中，通过学习简单的决策规则——IF THEN规则，从而预测目标变量的值。

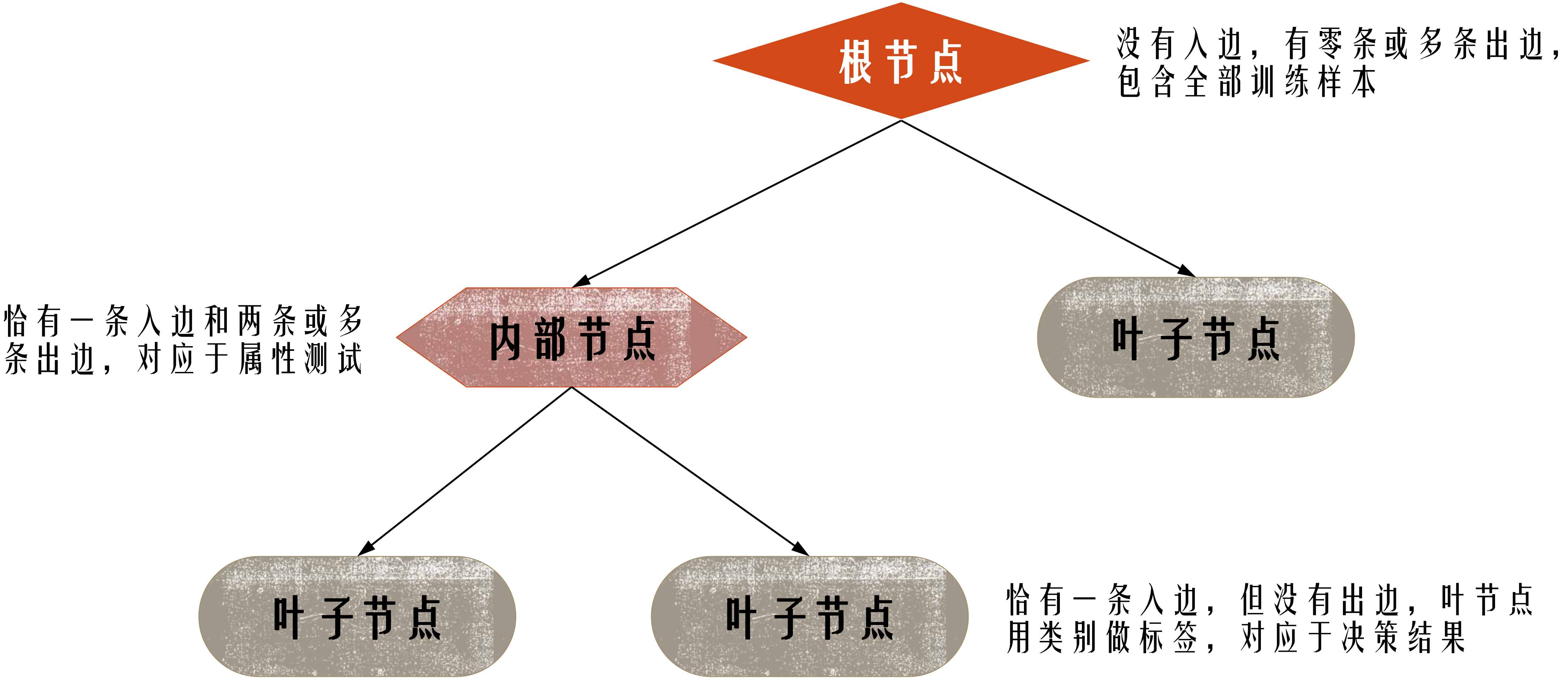
- 决策树是一种由节点和有向边组成的层次结构
  - **根节点（root node）**

    没有入边，有零条或多条出边
  - **内部节点（internal node）**

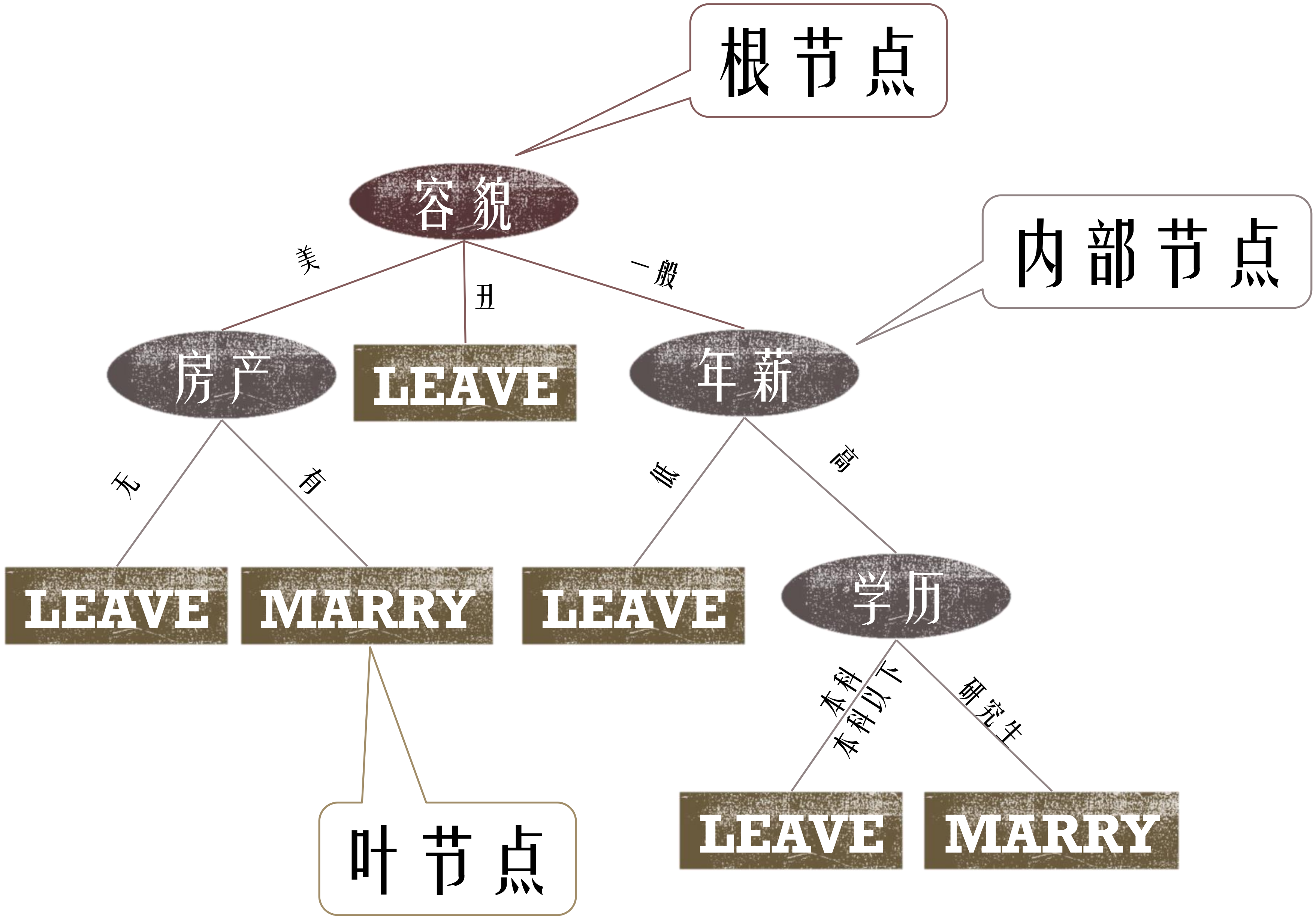
    恰有一条入边和两条或多条出边
  - **叶节点（leaf node）或终节点（terminal node）**

    恰有一条入边，但没有出边

    每个叶节点都赋予一个类标号



决策树内部的每一个节点代表的是对一个特征的测试，树的分支代表该特征的每一个测试结果，从根节点到每个叶子节点的路径对应一条决策规则



根节点
容貌
美
丑
一般
内部节点
房产
LEAVE
年薪
无
有
低
高
LEAVE
MARRY
LEAVE
学历
本科
本科以下
研究生
叶节点
LEAVE
MARRY

自顶向下，分而治之

决策树归纳算法

- 贪心策略
  - 采用自上而下的递归构造（top-down induction）贪婪搜索遍历可能的决策树空间。
- 可以构造的决策树的数目达到指数级。

HUNT算法

- Hunt等人于1966年提出，是许多决策树算法的基础。
- 算法基本过程
  - 记D_t为到达节点t的训练记录集
  - 如果D_t包含的记录属于同一个类y_t，那么t为叶子节点，标记为y_t。
  - 如果D_t包含属于多个类的记录，使用属性测试条件将数据划分为更小的子集。对于属性测试条件的每个输出，创建一个子女节点，并依测试结果将D_t中的记录分配到子女节点中去。
  - 递归地对每个子女节点应用该算法。

- 例： 预测贷款申请者会否按时归还贷款
  - 为此，考察以前贷款者的贷款记录

	二元的	分类的	连续的	类
Tid	有房者	婚姻状况	年收入	拖欠贷款者
1	是	单身	125K	否
2	否	已婚	100K	否
3	否	单身	70K	否
4	是	已婚	120K	否
5	否	离异	95K	是
6	否	已婚	60K	否
7	是	离异	220K	否
8	否	单身	85K	是
9	否	已婚	75K	否
10	否	单身	90K	是

训练数据集：预测拖欠银行贷款的贷款者

- 初始决策树只有一个节点，类标号为“拖欠贷款者=否”

	二元的	分类的	连续的	类
Tid	有房者	婚姻状况	年收入	拖欠贷款者
1	是	单身	125K	否
2	否	已婚	100K	否
3	否	单身	70K	否
4	是	已婚	120K	否
5	否	离异	95K	是
6	否	已婚	60K	否
7	是	离异	220K	否
8	否	单身	85K	是
9	否	已婚	75K	否
10	否	单身	90K	是

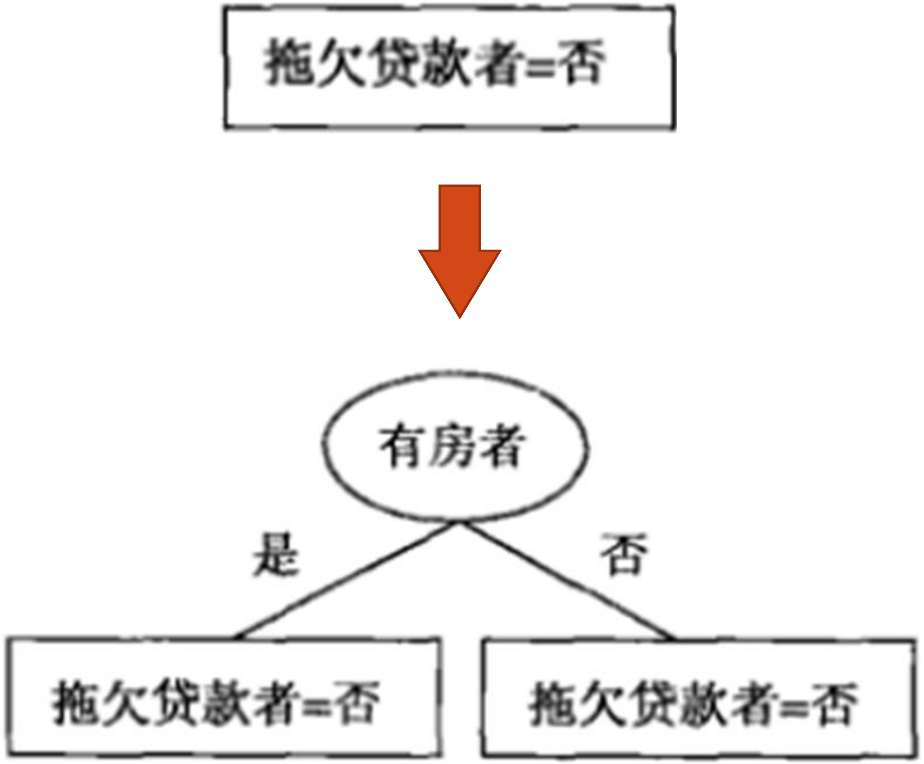
训练数据集：预测拖欠银行贷款的贷款者

拖欠贷款者=否

- 进一步细化该树，根据“有房者”测试条件，创建两个子女节点，将训练数据划分为两类较小的子集

	二元的	分类的	连续的	类
Tid	有房者	婚姻状况	年收入	拖欠贷款者
1	是	单身	125K	否
2	否	已婚	100K	否
3	否	单身	70K	否
4	是	已婚	120K	否
5	否	离异	95K	是
6	否	已婚	60K	否
7	是	离异	220K	否
8	否	单身	85K	是
9	否	已婚	75K	否
10	否	单身	90K	是

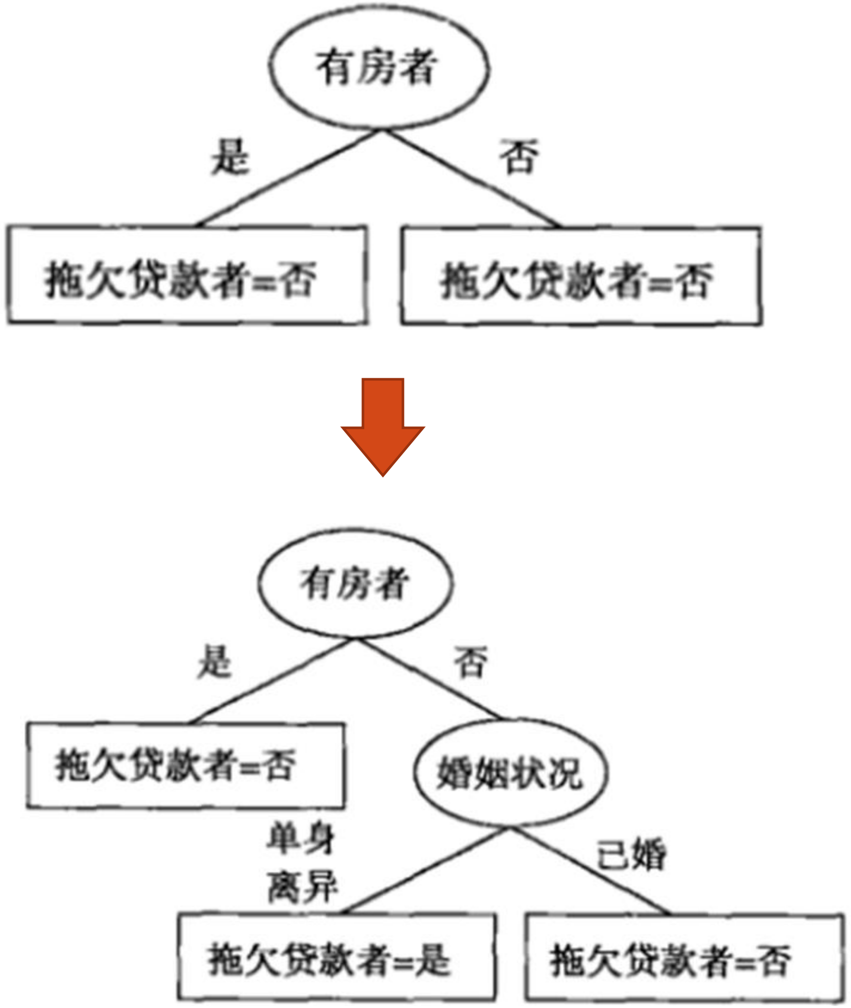
训练数据集：预测拖欠银行贷款的贷款者



▪ 对根节点的每个子女节点递归地调用Hunt算法

	二元的	分类的	连续的	类
Tid	有房者	婚姻状况	年收入	拖欠贷款者
1	是	单身	125K	否
2	否	已婚	100K	否
3	否	单身	70K	否
4	是	已婚	120K	否
5	否	离异	95K	是
6	否	已婚	60K	否
7	是	离异	220K	否
8	否	单身	85K	是
9	否	已婚	75K	否
10	否	单身	90K	是

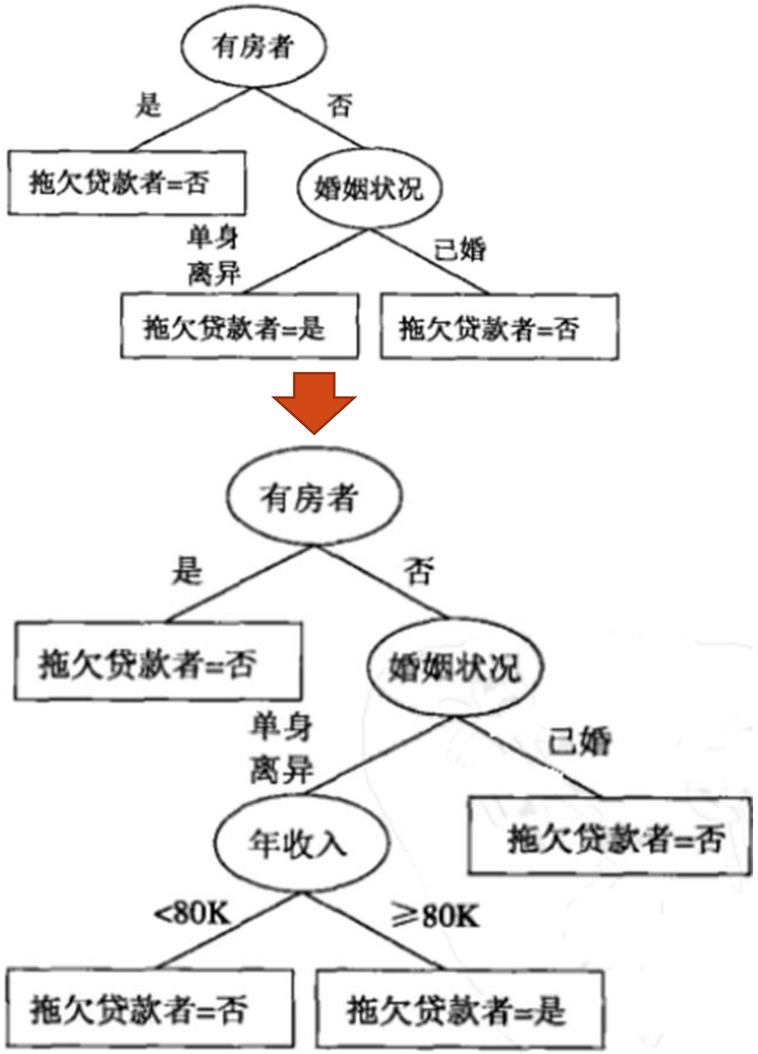
训练数据集：预测拖欠银行贷款的贷款者



- 继续算法直至所有记录都属于同一个类

	二元的	分类的	连续的	类
Tid	有房者	婚姻状况	年收入	拖欠贷款者
1	是	单身	125K	否
2	否	已婚	100K	否
3	否	单身	70K	否
4	是	已婚	120K	否
5	否	离异	95K	是
6	否	已婚	60K	否
7	是	离异	220K	否
8	否	单身	85K	是
9	否	已婚	75K	否
10	否	单身	90K	是

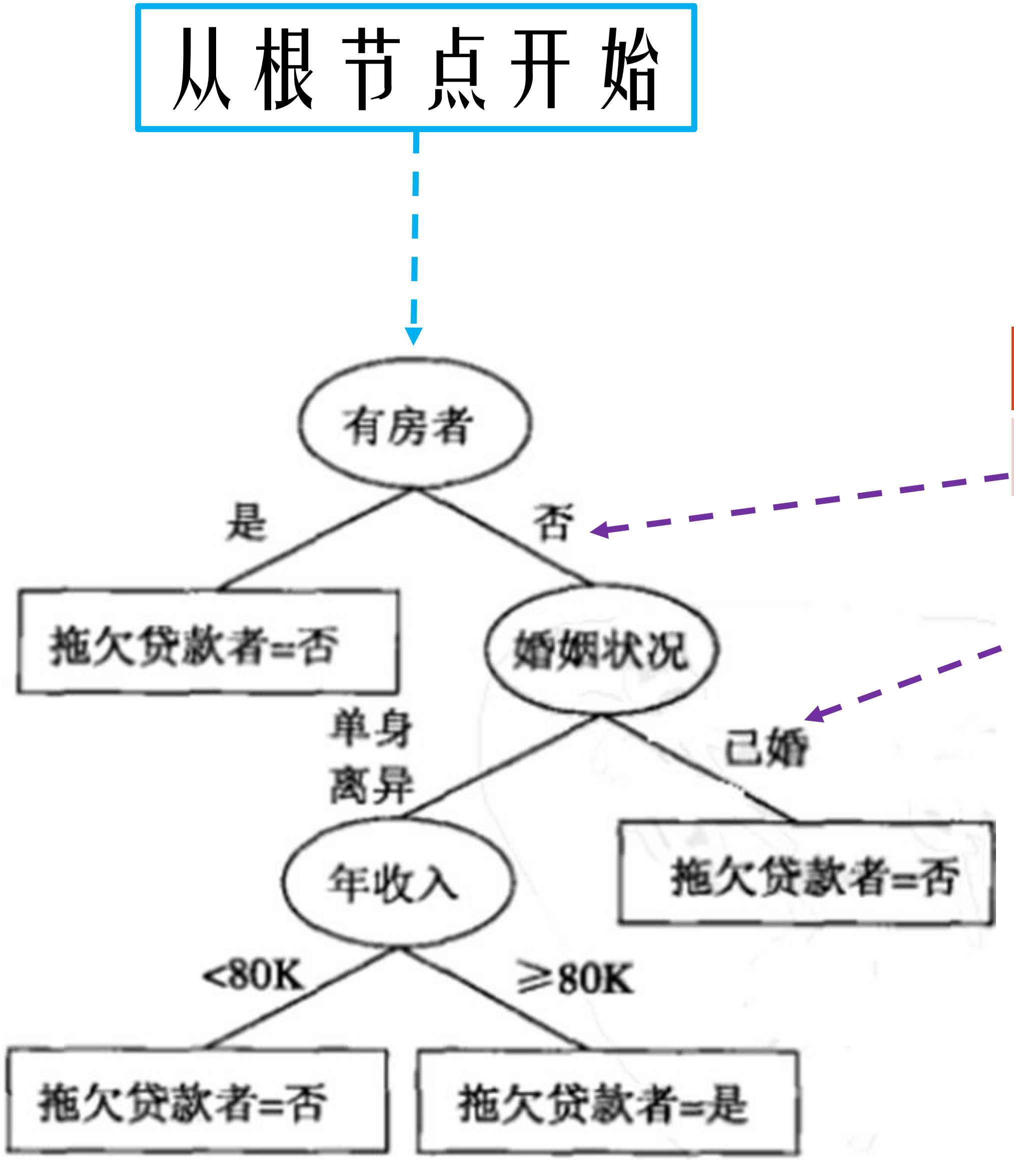
训练数据集：预测拖欠银行贷款的贷款者



利用决策树进行分类

- 一旦构造了决策树，则可对测试记录进行分类。
  - 从树的根节点开始，将测试条件用于测试记录，根据测试结果选择适当的分支
  - 沿着该分支
    - 到达另一个内部节点，使用新的测试条件
    - 到达一个叶节点
  - 叶节点的类标号被赋值给该测试记录

对未知记录应用模型



有房者	婚姻状况	年收入	拖欠贷款者
否	已婚	85k	?

拖欠贷款者=否

为什么第一次选择"有房者"来作为测试条件?我们是依据什么原则来选取属性测试条件的?

☞ 事实上，如果我们选择的属性测试条件不同，对于同一数据集来说所建立的决策树可能相差很大。

因此，在构建决策树时我们需要关心的问题包括：

1. 如何选择最优的属性测试条件?
2. 何时停止分裂?

何时停止分裂?

- 所有记录属于同一类
- 各属性值所占比例相同
- 没有相匹配的记录

如何选择最优的属性测试条件?

类分布

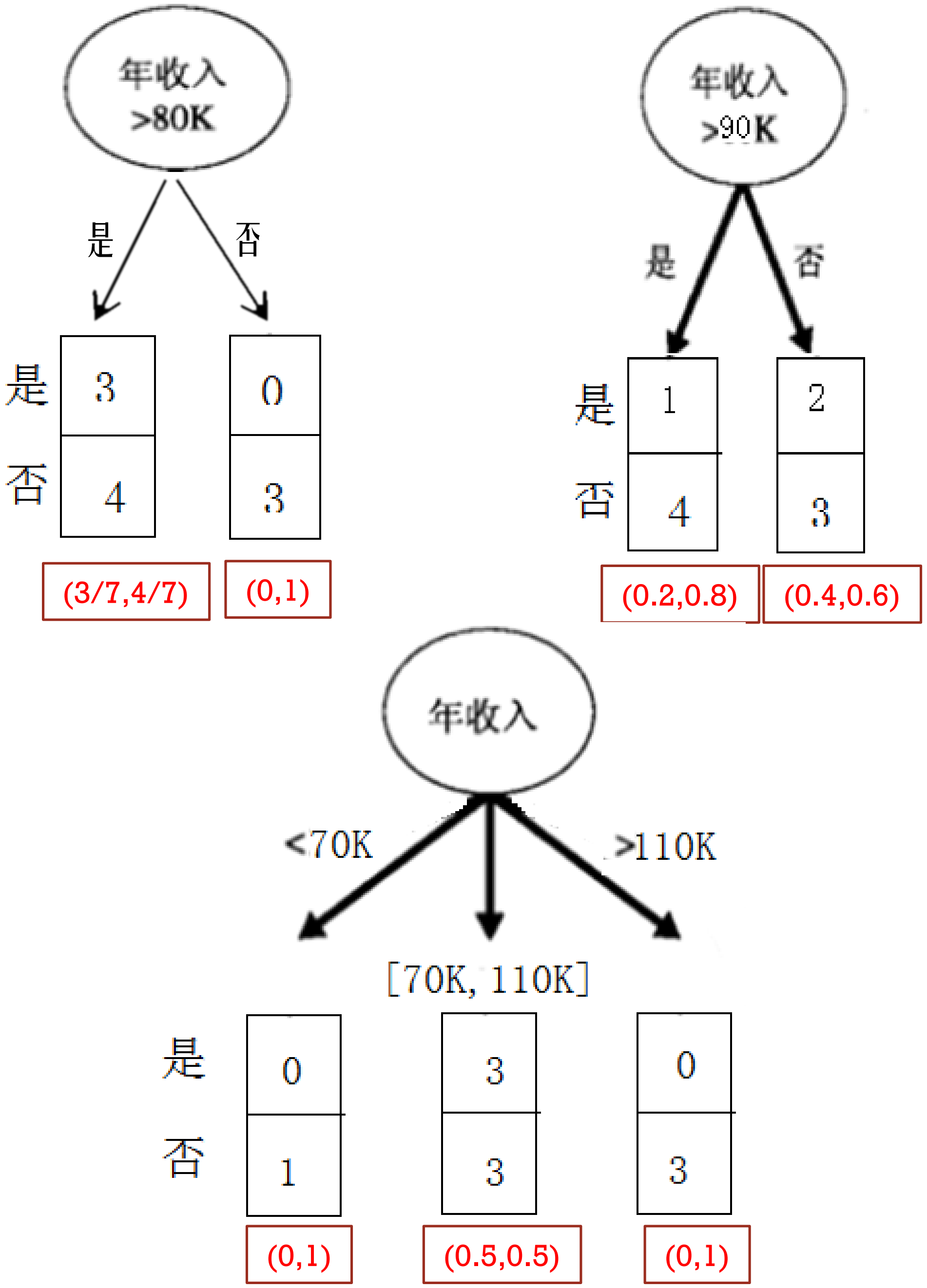
- p(i|t)是给定节点t中属于类i的记录所占的比例
  - 有时直接用pi表示该比例
  - 如，某结点样例集中正例和反例各有10个，则类分布为（0.5，0.5）

划分前：(0.3,0.7)

考察依属性“年收入”划分前后的类分布变化

如何划分该属性最优?

Tid	有房者	婚姻状况	年收入	拖欠贷款者
1	是	单身	125K	否
2	否	已婚	100K	否
3	否	单身	70K	否
4	是	已婚	120K	否
5	否	离异	95K	是
6	否	已婚	60K	否
7	是	离异	220K	否
8	否	单身	85K	是
9	否	已婚	75K	否
10	否	单身	90K	是

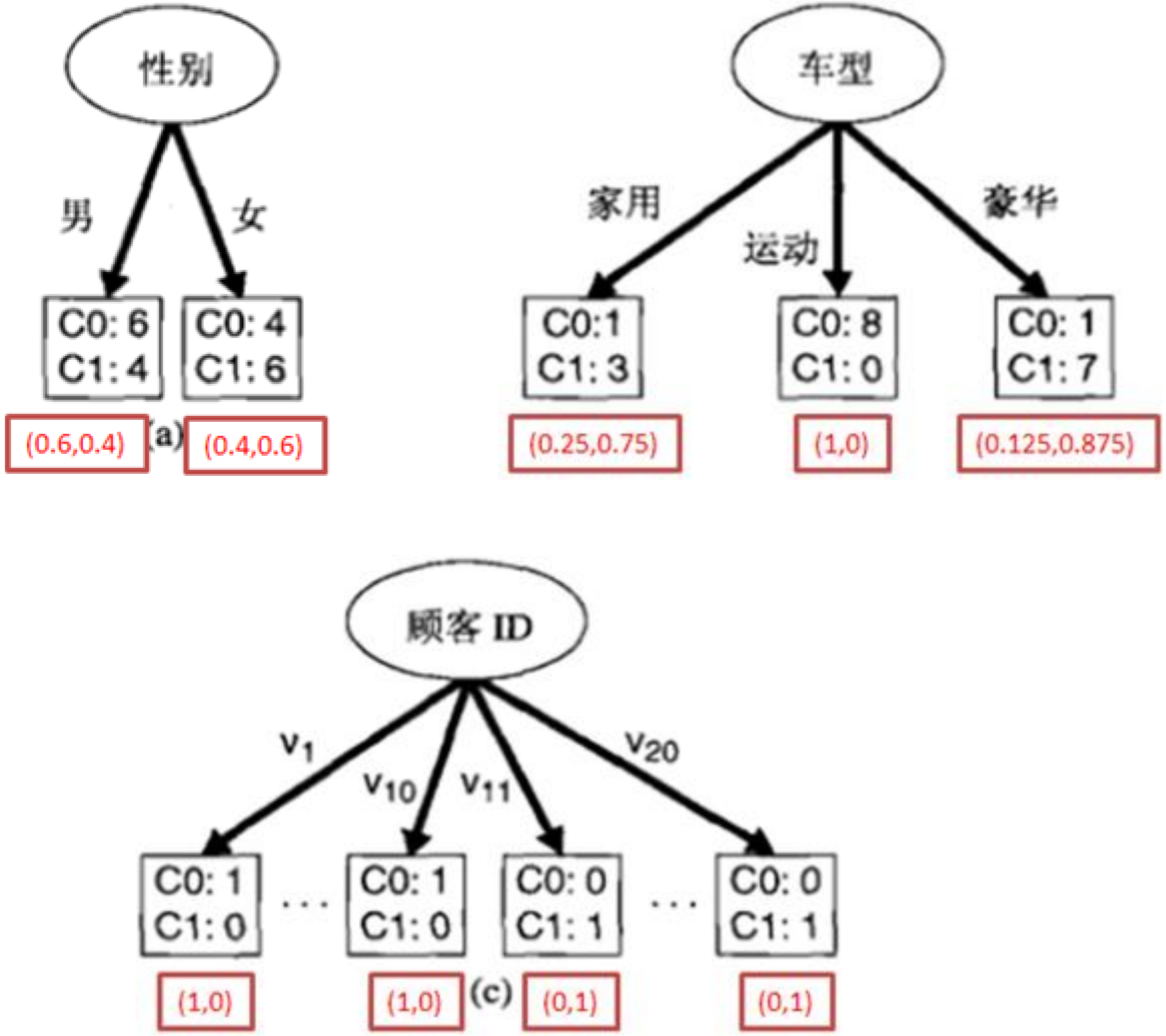


划分前：(0.5,0.5)

考察分别依属性“性别”、“车型”、“顾客ID”划分后的类分布。

数据集

顾客 ID	性别	车型	衬衣尺码	类
1	男	家用	小	C0
2	男	运动	中	C0
3	男	运动	中	C0
4	男	运动	大	C0
5	男	运动	加大	C0
6	男	运动	加大	C0
7	女	运动	小	C0
8	女	运动	小	C0
9	女	运动	中	C0
10	女	豪华	大	C0
11	男	家用	大	C1
12	男	家用	加大	C1
13	男	家用	中	C1
14	男	豪华	加大	C1
15	女	豪华	小	C1
16	女	豪华	小	C1
17	女	豪华	中	C1
18	女	豪华	中	C1
19	女	豪华	中	C1
20	女	豪华	大	C1



哪个属性用来划分数据最优？

- 选择最优分裂的度量通常是根据分裂后子女节点不纯性的程度，我们希望节点的不纯度逐渐降低
- 不纯的程度越低，类分布就越倾斜
- 度量不纯度的方法

  Entropy

  Gini Index

  Misclassification error

- 不纯度测量：熵

给定节点t:

$$\boxed{Entropy(t) = -\sum_{j} p(j|t)\log_2 p(j|t)}$$

p(j | t)是给定节点t中属于类j的记录所占的比例

某一节点t的熵值Entropy(t)越小，则该节点的纯度越高

- 不纯度测量：Gini Index

给定一个节点t的Gini Index为：

$$GINI(t) = 1 - \sum_j [p(j|t)]^2$$

p(j | t)是给定节点t中属于类j的记录所占的比例

反映了从节点t中随机抽取两个样本，其类别标记不一致的概率，Gini(t)越小，则节点t的纯度越高

- 不纯度测量：分类误差

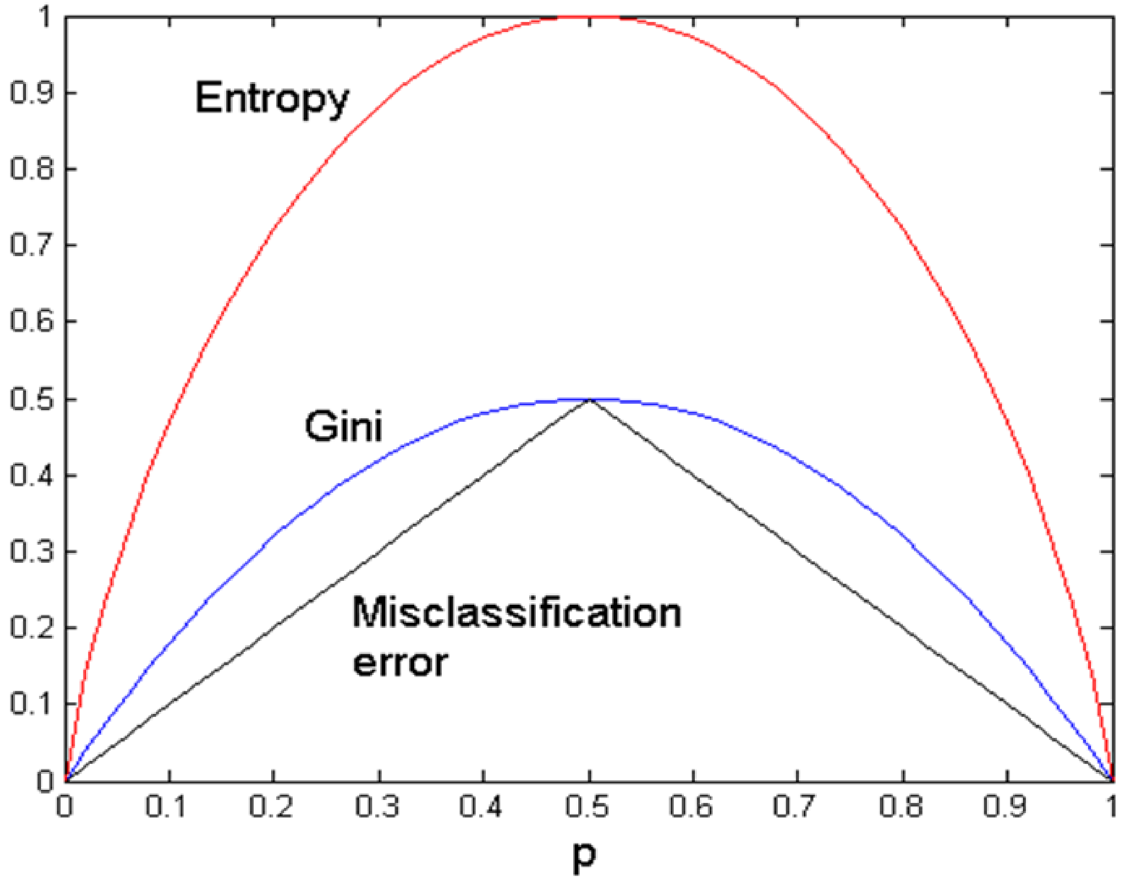
给定一个节点t的分类误差为：

$$\boxed{Error(t) = 1 - \max_i P(i|t)}$$

p(i|t)是给定节点t中属于类i的记录所占的比例

用来表示某一节点的分类误差，如果所有记录都属于同一类别，该值为0，所以分类误差越小越好

二元分类问题不纯性度量之间的比较



基尼系数和香农熵对于p的改变更为敏感（表现在图上就是曲线的斜率更大，更为陡峭），因此，这两种度量方法旨在寻找更为"pure“的nodes

Misclassification error略微有些敏感度不足

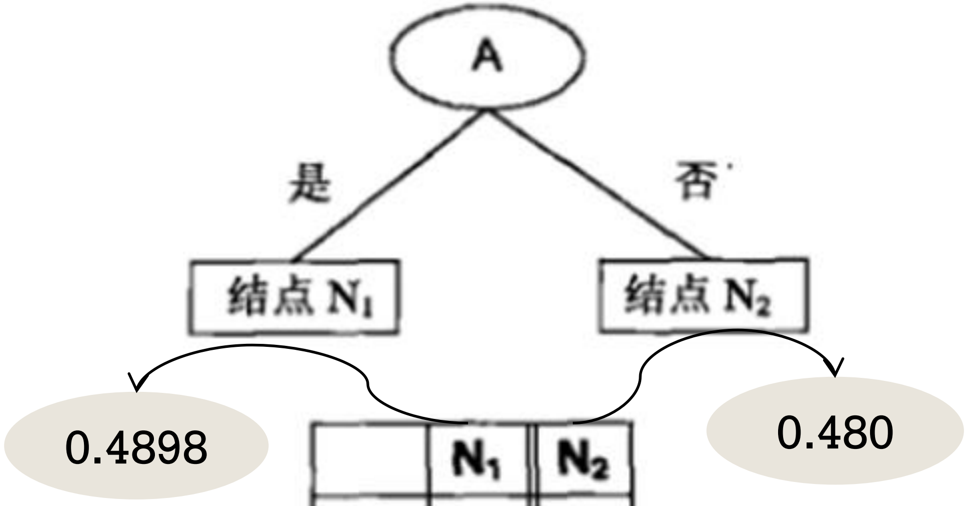
所有子女节点总不纯性的度量

- 用每个子女节点不纯度量的加权平均来表示

$$\sum_{j=1}^{k} \frac{N(v_j)}{N} I(v_j)$$

  - I(.)：给定节点的不纯性度量
  - N：父节点上的记录总数
  - k：属性值的个数
  - N(vj)：与子女节点vj相关联的记录个数

例（GINI INDEX）



	N_1	N_2
C0	4	2
C1	3	3
Gini = 0.486		

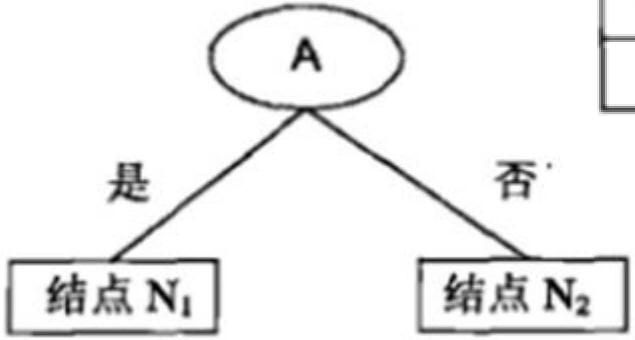
$$Gini(N_1) = 1-\left(\frac{4}{7}\right)^2-\left(\frac{3}{7}\right)^2 = \frac{24}{49}$$

$$Gini(N_2) = 1-\left(\frac{2}{5}\right)^2-\left(\frac{3}{5}\right)^2 = \frac{12}{25}$$

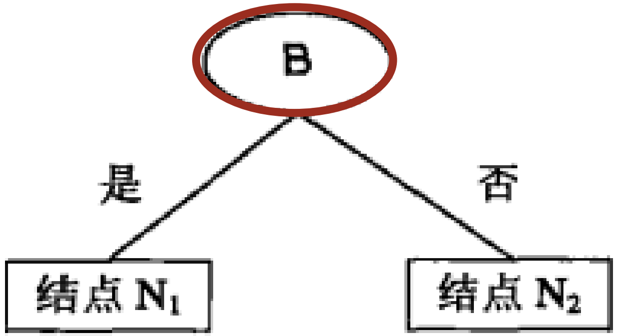
$$Gini = \frac{7}{12} \times \frac{24}{49} + \frac{5}{12} \times \frac{12}{25} = \frac{17}{35} \approx 0.486$$

节点的划分（GINI INDEX）

	父结点
C0	6
C1	6
Gini = 0.500	



	N_1	N_2
C0	4	2
C1	3	3
Gini = 0.486		



	N_1	N_2
C0	1	5
C1	4	2
Gini = 0.371		

THE END!