第 7 章　非度量方法

7．1 引言

在前面的章节中，我们研究了特征为实数的模式识别问题。但是在实际中，我们往往会遇见特征不是实数的情形。例如，考虑用下面四种属性来描述一种水果：颜色、纹理、味道和大小。这样，一种水果可以使用一个 4 元组来表达，如：（红色，有光泽，甜，小），即该水果的“颜色是红的”，“纹理是有光泽的”，“味道是甜的”，“尺寸是小的”。

在以实数为特征的问题中，往往需要计算距离度量，并且距离度量也是比较容易得到的。例如，在最近邻分类器中，两个样本之间的距离就可以定义为两个向量之间的距离。一个待测试样本的标签就是距离最近的训练样本的标签。然而在上面的水果的例子中，没有一个自然的方法来定义水果的颜色的相似性，味道的相似性，因而也就没有一个自然的方法定义水果的相似性。这样的数据叫做名义数据（nominal data），或非实数数据。这样的特征叫做名义特征，或名义变量，或非实数变量，或非度量（nonmetric）属性。非实数特征取值是离散的，没有自然的相似性的概念，甚至没有次序关系。如水果的颜色就是这样的一个特征。

在本章中，我们的要考虑的是，当我们遇见的问题存在非实数特征时，如何有效地从非实数数据中学习和发现类别信息？如何用这样的数据设计分类器？

7．2 决策树

可以用决策树（decision tree）对非实数数据分类。决策树对应一个树结构，树的根节点在最上面，叶子节点在最下面。就像“数据结构”这样的课程中使用的树结构一样。（如图 7－1 所示）。我们先来看如何用决策树分类。分类过程的第一步要从根节点开始，首先对待测样本的某一特征的取值提问。与根节点相连的不同分支对应其不同取值，因而不同的回答对应于不同的分支，这样到达了不同的后续子节点。然后，在已经到达的节点处作同样的分支判断，即把它作为一棵子树的根节点。继续这一过程，直到到达叶节点。每一个叶节点上都有一个类别标号，测试样本就被标记为它所到达的叶节点的类别。

和其他的分类器方法相比，决策树方法非常易于解释和理解。这是一个很重要的优点。实际上，我们可以很容易从图 7－1 上“读”出一个数据为什么被分为“苹果”而不是“香蕉”。不仅如此，我们还可以直接用逻辑表达式表示出树对应的分类规则。可以知道，从根节点到任何一个叶子节点的路径可以用一个“合取式”（conjunction）表达[1]。如图 7－1 中最左边的路径可以表示为（绿色 and 大）→西瓜。这是一条分类规则。这样，一棵决策树就对应于一组规则。有时候也把决策树的构建过程看作是一组规则的学习过程。

另外，由于不同的叶子节点可能对应于同一类别，因此，要准确描述一个类别，需要利用合取式和析取式构造一个逻辑表达式。如，苹果=（绿色 and 中等大小）or（红色 and 中等大小）。

树分类器的另一个优点是分类速度快，因为对于一个待测试样本只需一系列简单的询问就可以得到结果。还有，我们可以很自然的把专家的先验知识嵌入到树分类器中。特别是在实际应用中，当问题比较简单并且训练样本很少时，这类专家知识对分类非常有效。

[1] 概念“合取式”可以在人工智能方面的教材中找到。

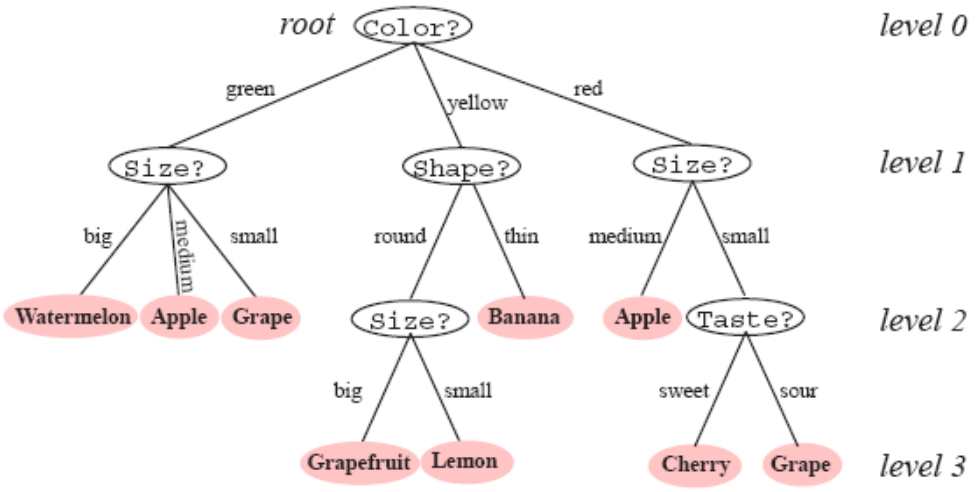


图 7－1　决策树的一次基本的自上而下的分类操作。每个节点处的问题是关于样本的一个属性的，而向下的分支对应可能的回答。连续访问节点，直到到达某个叶子节点，就可以从叶节点处得到类别标号。请注意，问题“大小？”出现在多个节点处，并且节点的分支数目存在不同，许多叶节点具有同样的类别标号（例如“苹果”）

7．3 CART

要使用树分类器，遇到的一个问题就是如何用一个有类别标号的训练样本集 D，构造一棵决策树。我们也把这个过程叫做树的“生成”过程。可以知道，随着决策树从树根到叶子节点，训练集被逐步划分成越来越小的子集。当一个子集中所有样本的类别标号相同时，树的分支操作就停止了，这时，该子集是“纯”’的子集。该子集对应的节点就是叶子节点。而当一个子集中的类别标号仍有混杂时，我们要么停止分支，这时叶子节点对应的样本集合是不“纯”的；要么另选一个属性进一步生长该树。很明显，这是一种递归的树生长过程。

分类和回归树（ classification and regression tree, CART）算法是这样一种通用的树生长算法。要实现 CART，需要解决下面 6 个问题。

1．特征必须是二值的还是允许取多个数值？这时节点处的分支数应该是多少？
2．在一个节点处应该测试哪个特征？
3．什么时候一个分支不再继续生长，而成为了一个叶节点？
4．如果一棵树生长得“过大”，怎样使其变小而简单，即：如何“剪枝”？
5．如果一个叶节点不“纯”，那么怎样给它分配类别标号？
6．如何使用缺失数据？

下面我们来依次讨论这些问题。

7．3．1 分支数目

决策树的根节点对应于全部训练样本。一个节点处的分支把训练样本划分成若干子集。分支的数目与前面的第二个问题紧密相关，因为第二个问题决定了在该节点处要根据哪个属性分叉。一般来说，一个节点处的分支数目由树的设计者确定，不同的节点处的分支数目也可能不同（如图 7－1）。虽然不同的树在不同节点处的分支数不同，但任何一棵数都等价于一棵二叉树，即一棵树的任何一个节点的分支数等于二（习题 2）。又由于二叉树的训练算法比较简便，所以下面的讨论主要是针对二叉树的（图 7－2）。

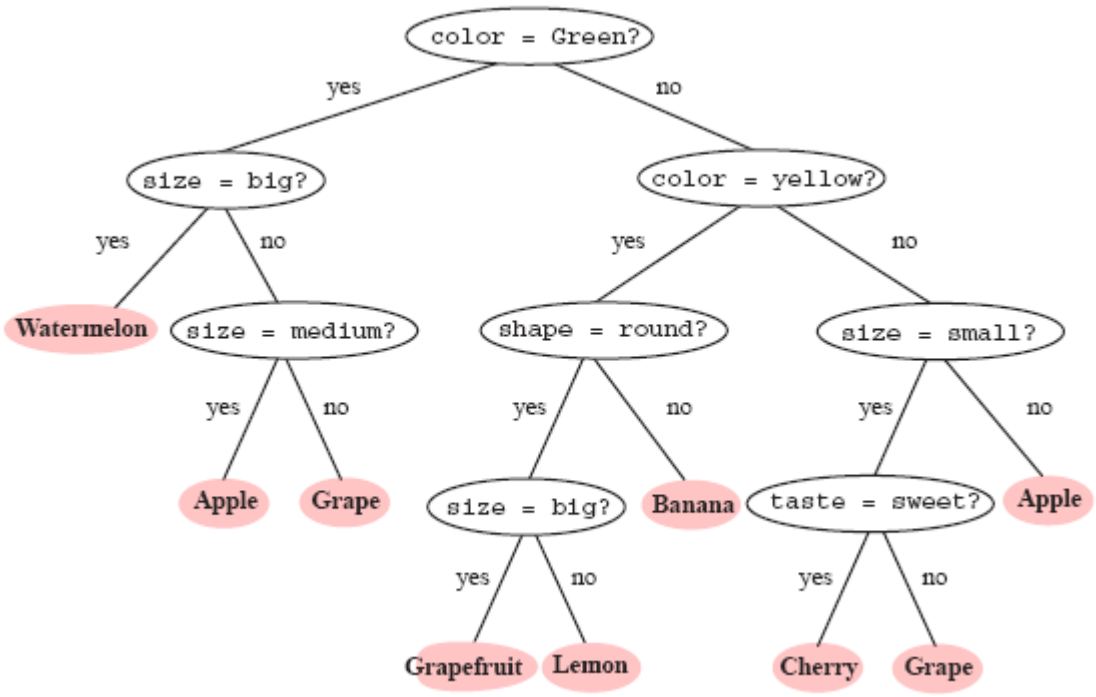


图 7—2 具有任意分支数目的一棵树都可以用一棵二叉树等价表示。这里的二叉树，只有是和否两个分支。它与图 7—1 的树具有同样的分类功能。

7．3．2 特征的选取与节点不纯度

在构建决策树时，一个重要问题是在每个节点处应该测试或查询哪一个特征。对于实数特征来说，用决策树方法得到的分类边界有较为直观的几何解释。例如，当在一个节点处问这样一个问题 s：“$x_1 < 5?$”时，会产生垂直于坐标轴 x_1 的一个超平面（$x_1 = 5$）的决策边界以及相应决策区域。决策树的各个节点处的所有查询问题产生的所有超平面合在一起，构成了最后的决策边界和决策区域。如图 7—3 所示。我们知道，空间任何一个决策边界可以用一系列的超平面来逼近（习题）。因此，可以使用决策树的方法以任何精度逼近一个确定的分类器。然而，对非数值数据来说，在节点处作查询并划分数据的过程并没有直观的几何解释。

上面讨论的是在一个节点处测试和查询一个特征的情形。实际上，也可以同时查询多个特征。例如，查询问题可以是几个特征的逻辑组合，如用“（大小=中）and（not（颜色=黄））？”作为问题。可以知道，在一个节点处的查询如果是几个特征的逻辑组合，这样的树也可以等价于一棵在一个节点处只查询一个特征的树(习题)。由于一个节点处只测试一个特征比较简单。因此，下面主要讨论这种比较简单的情况。

构造决策树的一个基本原则是“简单性”：我们期望获得的决策树简单、紧凑。这是 Occam 剃刀原则的一个形式，即能够解释数据的最简单的模型就是最好的模型（参见第 13 章）。根据这一原则，应该寻找这样一个查询 T，它能使后继节点对应的数据子集尽可能“纯”。为了把这个想法形式化，定义一个指标：“不纯度”（impurity）。实际上，不纯度和纯度是对应的。由于已经有了几种数学公式可以度量不纯度，因此考虑不纯度更方便。

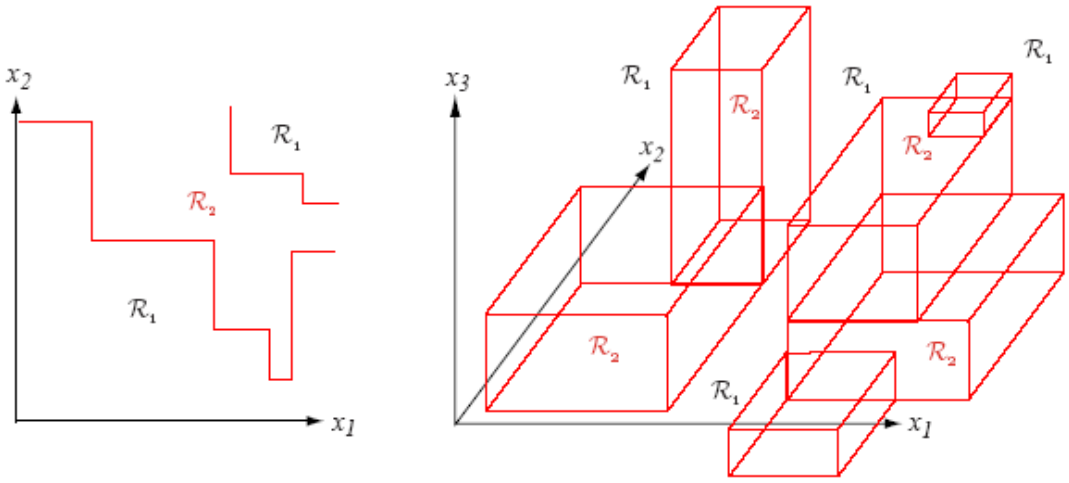


图 7－3 在一个节点处只查询一个特征产生的分界面垂直于所查询的特征轴。在两类问题中，决策区域用 R_i 表示。上面左右两个图分别给出了二维和三维的情况。

我们用 $i(N)$ 表示节点 N 的不纯度。不纯度应该具有下面的性质：当节点对应的子集中的数据标号相同时，要求 $i(N)=0$；而子集中数据类别标号均匀分布时，$i(N)$ 应当很大。

经常使用的一种不纯度度量是“熵不纯度”[2]（entropy impurity）：

$$i(N)=-\sum_j P(w_j)\log_2 P(w_j)\,, \qquad (7-1)$$

其中 $P(w_j)$ 是节点 N 处属于 w_j 类样本占总样本数的比例[3]。根据熵的性质，如果所有样本都来自同一类别，则熵不纯度为零，否则是大于零的正数；当所有类别以等概率出现时，熵取最大值。

另一种不纯度的定义在两类分类问题中非常有用。它可以用如下多项式形式定义：

$$i(N)=P(w_1)P(w_2) \qquad (7-2)$$

它也被称作“方差不纯度”，因为在某种合理的假设下，该度量与两类分布的总体分布方差有关（习题 10）。把（7－2）推广到多类情况时，就是“Gini 不纯度”：

$$i(N)=\sum_{i\neq j} P(w_i)P(w_j)=1-\sum_j P^2(w_j) \qquad (7-3)$$

还有一种不纯度度量是“错分不纯度”（misclassification impurity），它定义为：

$$i(N)=1-\max_j P(w_j) \qquad (7-4)$$

它可以度量节点 N 处训练样本被错分的最小概率。

---

[2] 概念“熵”可以在信息论的教材中找到。

[3] 准确的说，这里的比例和概率论中的概率并不相同，可以参看概率论的教材。但是在实际中，我们常常把这样的比例值看作概率的一个近似，这样可以解决计算问题。实际应用结果表明，这样的近似

这三种指标各自具有不同的特性。当各个类别概率相等时，Gini 不纯度比熵不纯度的峰更尖锐，错分不纯度在三个指标中峰是最尖锐的，但它此时的导数不连续。当在连续参数空间搜索最大值时不连续的导数会影响计算过程。图 7—4 给出了两类情况下，不纯度指标作为其中一个类别概率的函数的图形。

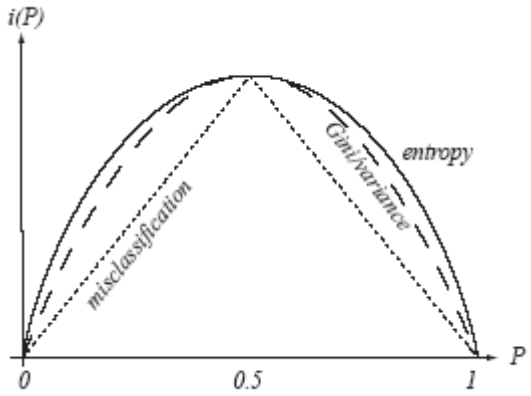


图 7—4 两类问题中，三个不纯度指标都在等类别概率时达到峰值点。为便于比较，图中三个指标的幅度已经被调整过。

有了不纯度的度量，现在开始考虑一个关键问题：对于节点 N，选择哪一个特征做查询？一个很明显的思路是选择能够使不纯度下降最快的那个查询。不纯度的下降程度可以记作：

$$\Delta i(N) = i(N) - P_L i(N_L) - (1-P_L) i(N_R) \qquad (7-5)$$

其中 N_L 和 N_R 分别是左、右子节点，$i(N_L)$，$i(N_R)$ 是相应的不纯度。P_L 是当查询一个特征时被分配到 N_L 中样本的数量占节点 N 处的样本数量的比例。根据上式，当采取不同的查询时，得到的 $\Delta i(N)$ 是不同的。我们需要寻找的就是使得 $\Delta i(N)$ 最大的查询。

在算法实现时，特征的选择是基于不纯度函数的极值的。因此，对该函数作一些适当修改，比如加上一个常数或乘上一个尺度因子，并不影响最终的结果。为了简单起见，我们经常选择容易计算的函数，比如在每个节点处只考虑单个特征。当我们面对的是非实数特征时，就必须充分搜索训练样本的全部可能的子集，以找到使 $\Delta i(N)$ 最大的查询。而当特征取值为实数时，则可以采用梯度下降算法寻找分支超平面（见 7．3．8 节）。实际中，经常构建的是二叉树，其原因是计算较为简单。如果分支数大于 2，就必须采用更复杂和更困难的优化技术。

有时，可能会有多个不同的查询，其产生的不纯度下降量是相同的，这时我们该选择哪一个查询？比如，如果一个特征取值为实数，变量 x 在开区间 (x_l, x_r) 中任何一点的取值都会产生相同的最大不纯度下降量。这时经常采用的方法是取其中间一个数值

可以得到很满意的结果。

$(x_l + x_r)/2$，或者取加权平均$(1-P)x_l + x_r P$，其中P是样本被分到左边分支的概率。

请注意式（7－5）的优化是局部的，只考虑了单一特征带来的不纯度下降。这是一种局部贪婪算法。一般来说，这样的算法无法保证优化过程会达到全局最优。如果要达到全局最优，在选择特征时还应该考虑特征之间的关系，即在该节点处选择一个特征会对于后续的节点选择什么特征有什么影响。但这样一来，其算法复杂性就会大大增加。这与第八章内容有关（见第八章）。

确定合理的不纯度函数和学习算法之后，总能够继续子树的分支直到在叶节点处得到最小的不纯度。不过，可能仍然无法保证叶节点的不纯度一定为零。例如：如果有两个样本，虽然来自不同类别，但有相同的特征取值，这时对应的叶节点的不纯度也会比零大。

在树的生长过程中，有时会出现错分不纯度不再下降，而 Gini 不纯度却仍在下降的现象。因此，我们更倾向用 Gini 指标。考虑下面的情况，如果节点N处有 90 个w_1类样本，和 10 个w_2类样本，这时的错分不纯度是 0.1。若假定后续的分支将不可能出现w_2占优势的情形，那么不管怎样分支，错分不纯度会一直为 0.1（习题）。所以我们可以假定有 70 个w_1类样本和 0 个w_2类样本被分到右边，20 个w_1类样本和 10 个w_2类样本被分到左边。这样一个结果是令人满意的，但从错分不纯度上（这时为 0.1）看不出来。而 Gini 不纯度却明显显示出了不纯度的下降。简单说，Gini 不纯度指标指出这是一次好的分支，而错分不纯度却无能为力。

一些实验结果表明，不纯度度量函数的选择对最终的分类效果影响不大。熵不纯度因其计算简单，并且来源于信息论而被普遍采用。Gini 不纯度也同样受到重视。

多重分支

上面集中研究了二叉树的情况。我们也可以调整算法中的部分细节从而允许树的节点分支数目在训练过程中发生变动。在后面讲述 ID3 算法时（7．4．l 节）还要讨论这个问题。这种情况下，可以把式（7－5）推广到如下多重分支的情况：

$$\Delta i(N) = i(N) - \sum_{k=1}^{B} P_k i(N_k) \qquad (7-6)$$

其中B是分支数目，P_k是节点N_k处样本占N处样本的比例，满足$\sum_{k=1}^{B} P_k = 1$。但是，式（7－6）存在这样的缺点，当B取值比较大时，$\Delta i(N)$会比较大，但是其对应的分支不一定有意义。读者可以给出一个这样的例子。为避免这一缺点，不纯度变化量应该在式（7－6）的基础上做下面的调整（习题 17）：

$$\Delta i_B(N) = \frac{\Delta i(N)}{-\sum_{k=1}^{B} P_k \log_2 P_k} \qquad (7-7)$$

同以前一样，使$\Delta i_B(N)$最大的分支就是最优的分支。

7．3．3 分支停止准则

现在我们讨论二叉树的训练什么时候应该停止。如果我们让树一直生长到所有的叶节点都到达最小的不纯度为止，那么数据一般会“过拟合”（overfit）（参见第 9 章）。例如，一个极端的情况就是每一个叶节点只对应一个训练样本。这时，决策树就退化成为一个查找表（lookup table）。而对于两类混迭在一起的分类问题，这样的决策树的推广性能就不可能很好。可是，如果树过早停止生长，那么对训练样本的误差就比较大，导致分类性能很差。

究竟什么时候应该停止分支？常规的做法之一，是验证和交叉验证技术（validation and cross-validation），（见第 13 章）。从技术上看，验证过程就是先用部分的训练样本（如 90%）来训练一棵树，然后用剩余的（10%）部分测试树的准确性。这样，当一个节点处的准确性已经达到最小，该节点就不再继续分支。为了对该节点处的测试更具有统计意义，我们对训练集合多次独立随机划分，用划分的子集多次的训练和测试。

另一种作法是预先设定一个不纯度下降量的阈值。当所有的候选分支的不纯度下降量都小于这个阈值，则停止分支。这种作法有两个优点。其一，全部样本都可用来训练。相比之下，交叉验证方法只使用了部分数据。其二，树的各个深度上都可能存在叶节点。当数据中存在各种复杂情况时，这一点很重要。这是一棵非平衡树，对不同的待测试样本的测试次数可能是不同的。但该方法的缺点也很明显：即很难预先设定一个合适的阈值。因为树的分类准确性与阈值大小通常不是简单的函数关系，因此，很难从树的性能出发推算出阈值（上机练习 2）。使用阈值的一种简单方法是检查每个节点代表的样本数目是否小于某值，比如 10 个，或者小于某个固定的比例，如 5%。如果是，则停止节点的生长。

还有一种做法是用高的复杂度换取高的准确率，它通过最小化如下这个准则：

$$\alpha \cdot size + \sum_{\text{叶子节点}} i(N) \qquad (7-8)$$

这里的 $size$ 表示节点的数目，α 是一个正常数。公式（7－8）的第一项对应于树的复杂程度，第二项对应于树的分类准确性。如果树比较简单（即第一项比较小），并且树的准确性较高（即第二项比较小），则这个准则值比较小。当树很复杂时，会出现过拟合的现象，其泛化能力可能比较差。如果 $i(N)$ 采用熵不纯度指标，那么式（7－8）可以从第 13 章讨论的“最小描述长度”（Minimum Description Length, MDL）得到支持。所有叶节点的不纯度之和代表了使用该分类树对训练样本分类时的不确定性（单位为 bit）。而 $size$ 用于衡量该分类器的复杂度（单位也为 bit）。α 是在两者之间的一个折中。不过 α 的设定也不容易，它同样与分类器的性能不是简单的函数关系。

还有一种方法是基于不纯度下降量的统计显著性（statistical significance）分析。在构造树的过程中，估计目前全部已有节点的不纯度下降量 Δi 的概率分布。我们假定它就是 Δi 的总体分布。对某一候选的节点分支而言，我们检验它与上述分布是否存在统计差异，比如用 χ^2 检验（参见附录 A.6.1）。如果某个候选分支的不纯度下降量统计不显著，则停止该节点处的分支（习题 15）。

该技术的一种变形，即“假设检验”（hypothesis test）技术也可以被采用。当对 Δi 的分布的知识很少的时候，可以使用该方法。要确定候选分支是否有统计上的“意义”，就是要判断该分支是否明显有别于一次随机分支。假定节点 N 处有 n 个样本（w_1 类有 n_1 个，w_2 类有 n_2 个）。我们需要检验一个候选分支 s 是否与随机分支有明显的区别。假

定某个候选分支 s 把 Pn 个样本分到左边子节点，而把 $(1-P)n$ 个样本分到右边子节点。如果此分支是随机划分的，则应该有 Pn_1 个 w_1 类样本和 Pn_2 个 w_2 类样本分到了左边子节点，而其他的都到了右边子节点。我们用 χ^2 统计量来定量估计这次分支 s 与随机分支的偏差。在两类情况下，该统计量为

$$\chi^2 = \sum_{i=1}^{2} \frac{(n_{iL} - n_{ie})^2}{n_{ie}} \qquad (7-9)$$

其中 n_{iL} 是在决策 s 下 w_i 类样本分到左边的数目，而 $n_{ie} = Pn_i$ 是随机分支情况下对应的值。当两者相同时，χ^2 统计量为零。两者差异越大，χ^2 统计量也越大。当 χ^2 大于某临界值时（多用查表的方法得到（参见附录 A.6.1）），就可拒绝零假设（null hypothesis），因为 s 与随机分支已经以某概率或在某置信水平（confidence level），例如 0.01 或 0.05, 上有了显著性的差异。置信水平的临界值与问题的自由度有关。在上面的问题中，自由度是 1，因为对某给定概率 P，只要 n_{1L} 值已知，那么其他所有的值 n_{2L}，n_{1e}，n_{2e} 也就都确定了。当某个节点“最显著”的分支产生的 χ^2 统计量比给定的置信水平还要低，则应停止该节点的生长。

7．3．4 剪枝

有时分支会过早停止。在节点 N 处进行的分支决策没有考虑对其下面一层节点的决策的影响。一旦停止分支，使得节点 N 成为叶节点，就断绝了其后继节点进行“好”的分支操作的任何可能性。因此说分支过早的停止了。

我们也可以采取剪枝（pruning）而不是停止分支的策略。在剪枝过程中，树首先要充分生长，直到叶节点都有最小的不纯度值为止。然后，对所有相邻的成对叶节点（它们连到同一个公共父节点上），考虑是否应该消去它们。如果消去它们能引起不纯度增长，那么就消去这一对叶节点，并令它们的公共父节点成为新的叶节点（该父节点当然也可能在以后的处理中被成对消去）。很显然，这种“合并”两个叶节点的做法与节点分支的过程刚好相反。经过上述剪枝后，叶节点常常会分布在树的不同的深度上，因而树也变得不平衡了。

从叶节点开始剪枝是一种通常的作法，但并非是唯一的方法。其他方法包括：基于代价复杂度的剪枝技术可以直接用叶节点一次替换一棵复杂的子树。7．4．2 节的 C4.5 算法能消去树上的任意一个节点。

使用剪枝技术，可以充分利用全部训练集中样本生成树，而无需保留部分样本用于交叉验证。当然，分支停止方法相比，算法的计算量会增加。不过对于小样本集的情况，由于总的计算量低，剪枝方法就优于分支停止方法。有时，“停止分支”技术和“剪枝”技术，也分别被称为“预剪枝”技术和“后剪枝”技术。

还有一种剪枝方法的思路和上面的方法非常不同。它是通过对规则做调整而达到剪枝的目的。这一部分内容请看 7.4.2.

7．3．5 叶节点的标号

给每个叶节点确定类别标号是比较容易的。如果每个叶节点都只包括一个样本，那么该类别的标号就是叶节点的标号。多数情况下，叶节点都有正的不纯度。这样就应该用其中占优势的样本类别来标号。

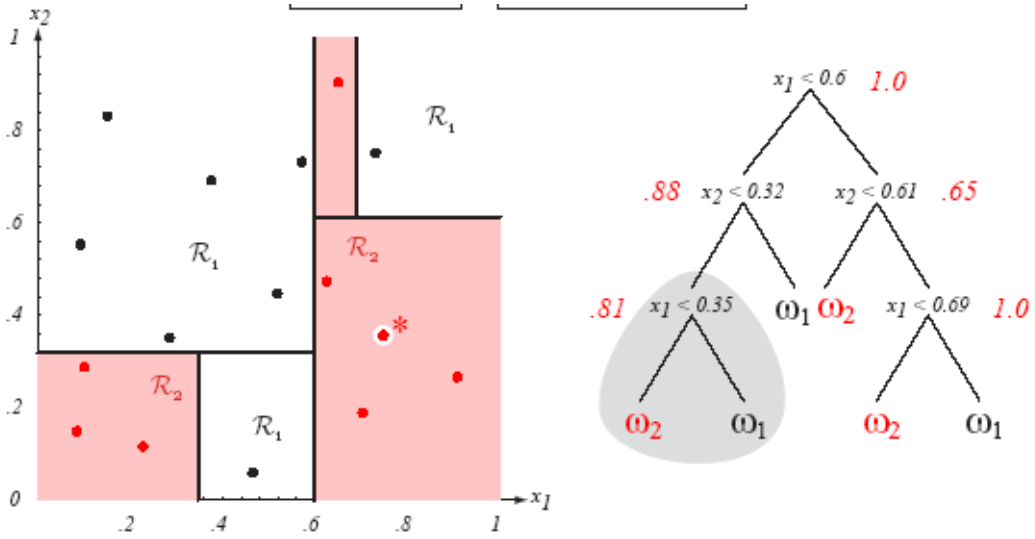
下面用例 7.1 说明上述步骤。

例 7.1 一棵简单的决策树

采用熵不纯度度量和如下二维空间的 16 个样本（$n=16$）训练一棵二叉树。

ω_1 (black)	
x_1	x_2
.15	.83
.09	.55
.29	.35
.38	.70
.52	.48
.57	.73
.73	.75
.47	.06

ω_2 (red)	
x_1	x_2
.10	.29
.08	.15
.23	.16
.70	.19
.62	.47
.91	.27
.65	.90
.75	.36* (.32†)



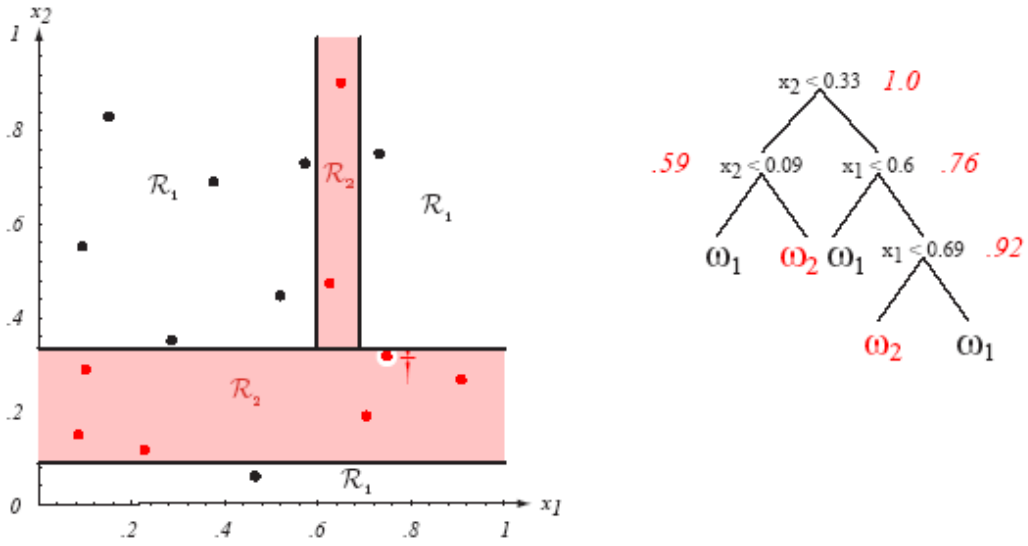


图 7—5 训练样本在表 7.1 中，图 7—5（a）是得到的未剪枝树。其中非叶节点的不纯度已在图上标出。叶节点不纯度均为 0。如果图（a）中标有“*”的样本往下稍微偏一点儿，则产生的树和决策区域（图 7—5（b））与（a）会有很大不同。

根节点的不纯度等于：

$$i(N_{root}) = -\sum_{i=1}^{2} P(w_i)\log_2 P(w_i) = -[0.5\log_2 0.5 + 0.5\log_2 0.5] = 1.0$$

为简单起见，考虑平行于特征轴的分支——即“ $x_i < x_{is}$？”的形式。利用方程（7—5）通过对 x_1 在 $n-1$ 个位置和对 x_2 进行穷举搜索，寻找最大的 $\Delta i(N)$。发现不纯度下降最大的位置是 $x_{1s} = 0.6$。因此也就确定了根节点。继续对其子树进行上述过程，直到每个叶节点对应的训练样本具有相同的类别标号（即不纯度=0），如图所示。

如果要剪枝，图 7—5 第一行左下方的叶子对会最先被消去，因为消去它们后不纯度增长得最小。在本例中，采用具有适当的阈值的停止分支技术也会得到同样的结果。但通常情况下，特别是决策树很大时，或者经过多次剪枝，二者的最终结果是不同的。例子中给定的训练样本集，表明树的生长对训练样本的微小位置变动很敏感。例如图中标有*的 w_2 类样本的稍微移动（标有 t）。树的这种不稳定性，很大程度上是由离散性和节点选择时的“贪婪”性所导致的。

从例 7．1 中看到决策树对于训练样本的不稳定性。当然，用稍微有所不同的训练集去训练任何普通的分类器，其最终结果都会有所不同。但是，对于 CART，即使是一个样本点的改变也会导致非常不同的最终判决结果。不稳定性也告诉我们对树进行增量式的训练(incremental learning)和离线式的训练会得到很不一样的分类器，即使在训练样本集相同的情况下也不例外。

7．3．6 计算复杂度

假如给定 n 个 d 维训练样本，希望使用熵不纯度构造一棵二叉树。在树的每一个节

点处采用平行于特征轴的方向划分，来解决一个两类分类问题。我们讨论一下这种计算的时间复杂度和空间复杂度是怎样的。

在根节点（第 0 层），首先要将全部训练数据排序。于是对任何一维特征，需要 $O(n\log n)$ 次计算。熵值的计算量是 $O(n)+(n-1)O(d)$，因为要检查 $n-1$ 个可能的分支点。于是根节点总的计算量是 $O(dn\log n)$。考虑平均的情况，大致各有一半的训练样本点平均分配到分支两边。上述分析表明每一个第一层的节点分支的计算复杂度为 $O(d\,n/2\log n/2)$。又因为第一层共有两个节点，所以总计算量为 $O(dn\log n/2)$。同样，第二层的计算量为：$O(dn\log n/4)$。依此类推。由于树的总层数为 $O(\log n)$，对全部层求和得到总的平均时间复杂度为 $O(dn(\log n)^2)$。

在识别阶段，总的时间复杂度与树的深度相同一即 $O(\log n)$。在某些极端情况下，（比如，假设每个叶节点上只有一个样本点），树的深度就是节点的数目，这时的时间复杂度就是 $O(n)$。

值得说明的是，上面的假设条件（如节点的平均划分）很少严格成立，而且，训练中启发式（heuristics）技术可用于分支搜索的加速。但是，当维数 d 固定时，训练时的复杂度和识别时的复杂度可以让我们对于树的复杂度有大致的认识。它告诉我们训练要远比识别更复杂，当数据量很大时，二者的差异更大。

对实值数据而言，有几种方法可用于降低树训练时的复杂度。其中最简单的一种是从训练集定义域的中间位置选定初始分支点 x_{is}，然后通过二分法递归寻找最优分支点。由于在最优分支点处相邻点分属于不同的类别，所以可以通过只测试一个点附近样本是否同类来选择排除一些分支点。这样的一些技术可以适当降低训练复杂度。

7．3．7 特征选择

与大多数模式识别技术一样，选择适当的特征时 CART 和其他树分类方法的性能会非常好（图 7－6）。对实值特征向量来说，可在建造树之前应用标准的数据预处理技术。主分量分析是一种有效的数据预处理手段，因为它能找到数据的“重要性”轴（即主轴）。然而，如果不同区域的主轴有不同的方向，那么，简单的使用主分量方法是不够的。这时，可以利用其他的技术。

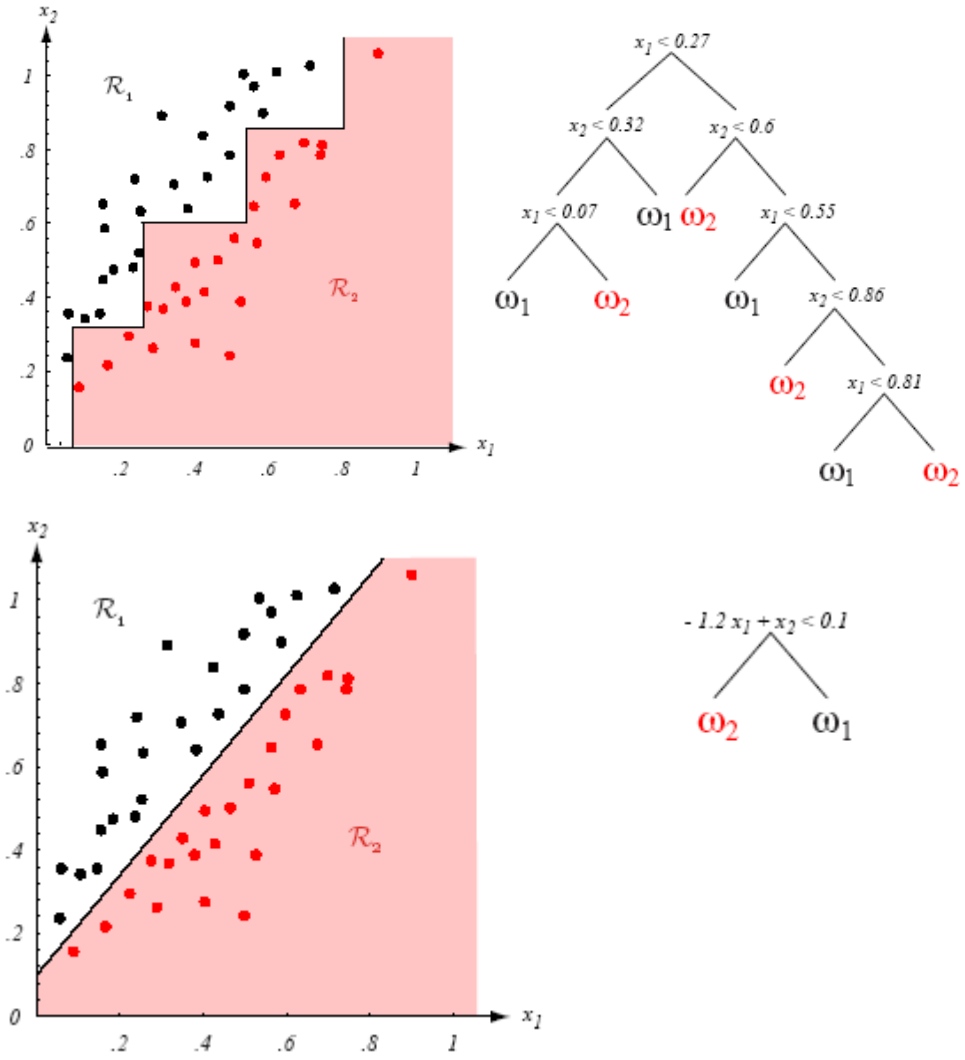


图 7-6　在一个节点处选择的特征不好将导致十分复杂的决策面。比如第一行采用的是平行于特征轴的方式划分数据，因而在一个节点处只选择一个特征。然而，如果特征选择的合适，树和决策面都会很简单，如图的第二行。这里采用的是特征的线性组合。

7．3．8 多元决策树

对于实值数据，当样本的分布比较复杂时，平行于特征轴的特征选择方法的效率和推广性都将很差（如图 7-7）。即使剪枝也无法得到好的分类器。最简单的解决方案是允许分支可以不平行于特征轴，也就是说可以采用一般的线性分类器。该分类器可以用基于分类或误差平方和准则的梯度下降算法来训练（参见第 4 章）。如果采用这种方法，当训练样本较大时，在靠近根的节点处训练过程会很慢，因为其训练涉及很大的样本集合。但远离根的节点处训练会很快。

在识别过程中，由于每个节点处的线性函数可以被很快地计算出来，所以识别的过程仍会很快。

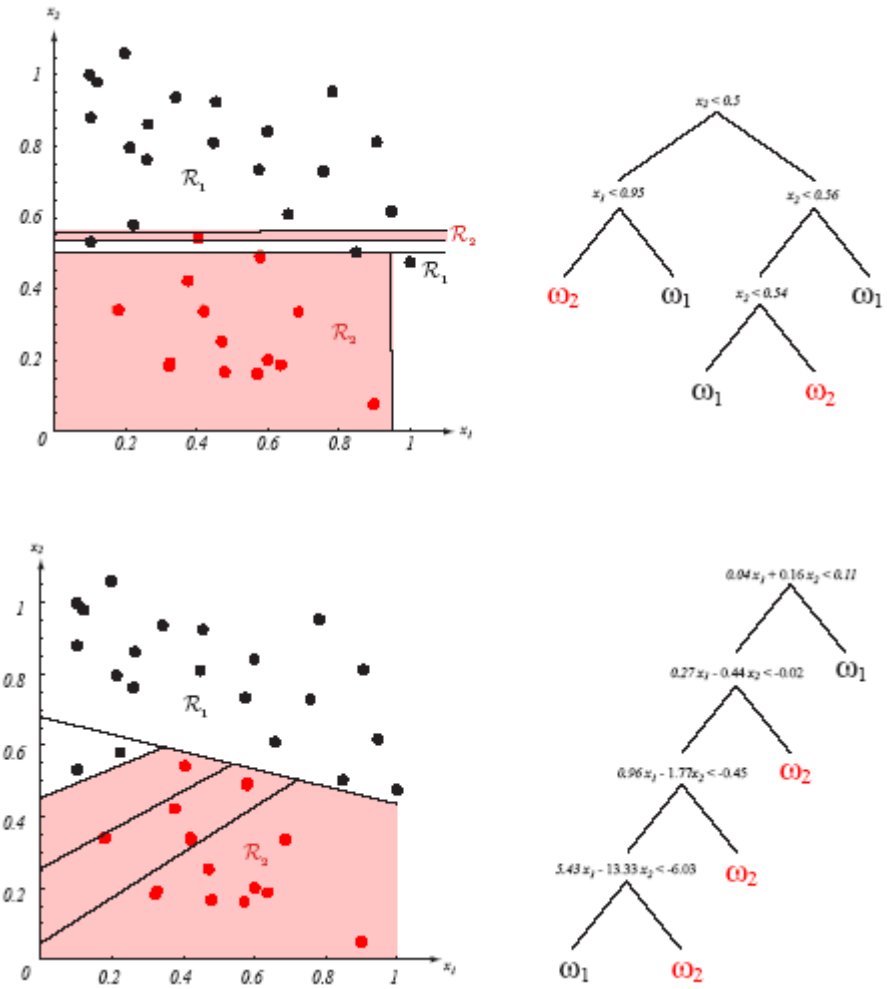


图 7—7　一种多元决策树。该树在每一个节点处采用了线性决策方法。

7．3．9 先验概率和代价函数

到目前为止，我们都假定任何一类样本在训练样本集和测试样本集中出现的概率相同。如果情况并非如此，就需要修改树的生长方法。最直接的做法是对每个样本依据先验频率加权（习题 16）。另外，我们可以不局限于误分类率或 0—1 代价而是使用一个更一般的代价函数。

例如，可以使用在第 2 章中讨论的代价函数 λ_{ij}，即将 w_i 的样本误分类为 w_j 所引起的代价。代价函数可以容易地放到 Gini 不纯度的公式中，成为加权 Gini 不纯度：

$$i(N) = \sum_{i \neq j} \lambda_{ij} P(w_i) P(w_j) \qquad (7-10)$$

以上讨论用于训练过程。代价函数也可以直接放到其他的不纯度函数中（习题 11）。

7．3．10 属性丢失问题

获取的数据往往存在属性缺失的问题。我们首先考虑训练集中部分样本特征丢失的情况。一种简单的做法是不考虑那些有特征缺失的样本，只利用完整的数据训练决策树。但这会导致很大的浪费，只有当特征完整的样本足够多时才适用。一种更好的方法是按前面所描述的方法进行训练，在计算节点 N 的不纯度时，只利用存在的属性信息。假

定在 N 处有 n 个训练样本，其中一个样本的属性 x_3 缺失了，其他每个样本的属性都很齐全，有 3 个属性。为找到 N 处的最好分支，用属性 x_1 对全部 n 个点计算，再用属性 x_2 对 n 个点计算，最后用属性 x_3 只对没有属性缺失的 $n-1$ 个样本计算。每个可能的分支都会导致不纯度的下降，只是其样本数目可能不同。经过比较，不纯度下降最大的就是最优分支。这种过程还可以直接推广到多重分支、多个缺损模式、甚至丢失多个属性的情况（习题 14）。刚才讨论的是有缺失数据时的训练过程。但是这样生成的决策树对于有属性缺失的数据不能分类。下面考虑如何训练和构造一棵决策树，它能够分类有属性缺失的数据。其基本思想就是在每一个节点处首先按照前面描述的方法给出一个分支，称为主分支（Primary split）。此外，还要考虑如果一个样本刚好缺失了该节点处要使用的属性的话应该如何决策。其策略是，给出一个有序的“替代分支”（surrogate split）集合。替代分支的顺序是由该分支与主分支之间“相似”度决定的。对于两个分支 s_1 和 s_2 的相似度可以这样来计算：对于一个样本集合，被两个分支都分到左边的样本数加上被两个分支都分到右边的样本数。这样两个样本数之和作为两个分支的相似度。这样在每一个节点处，除了主分支以外，还需要对其他的每一个属性做替代分支，最后把所有的替代分支排序。在对于属性缺失的数据分类时，首先测试主分支。如果缺失主分支要求的属性，则测试第一替代分支。如果第一替代分支不能用，则测试第二替代分支。这样下去直到找到一个可用的分支为止。

与替代分支方法不同的另一种方法是：虚拟值方法。对于一个数据，其缺失的属性值用它的最大似然值替代。

例 7．2 替代分支和属性缺失

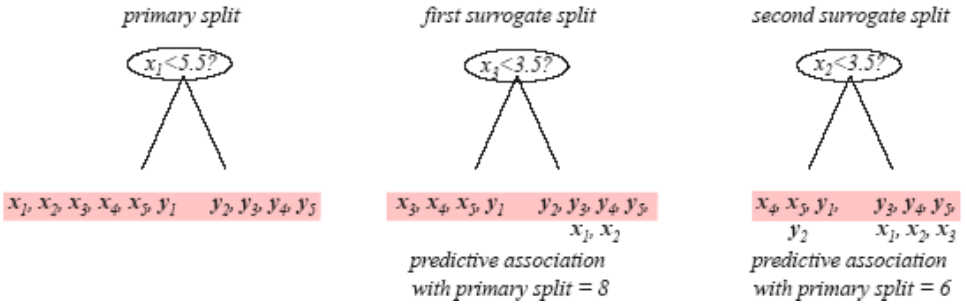
考虑用熵不纯度和下面 10 个训练样本构造一棵决策树。由于该树可能被用于识别有属性缺失的样本，因此，需要给每个节点提供替代分支。

$$\omega_1:\quad \mathbf{x}_1=\begin{pmatrix}0\\7\\8\end{pmatrix},\ \mathbf{x}_2=\begin{pmatrix}1\\8\\9\end{pmatrix},\ \mathbf{x}_3=\begin{pmatrix}2\\9\\0\end{pmatrix},\ \mathbf{x}_4=\begin{pmatrix}4\\1\\1\end{pmatrix},\ \mathbf{x}_5=\begin{pmatrix}5\\2\\2\end{pmatrix}$$

$$\omega_2:\quad \mathbf{y}_1=\begin{pmatrix}3\\3\\3\end{pmatrix},\ \mathbf{y}_2=\begin{pmatrix}6\\0\\4\end{pmatrix},\ \mathbf{y}_3=\begin{pmatrix}7\\4\\5\end{pmatrix},\ \mathbf{y}_4=\begin{pmatrix}8\\5\\6\end{pmatrix},\ \mathbf{y}_5=\begin{pmatrix}9\\6\\7\end{pmatrix}.$$

在所有基于单一特征的分支中，主分支“ $x_1<5.5?$ ”对整个训练集能最小化“熵不纯度”。替代分支必须采用不同于 x_1 的特征，只能是 x_2 或 x_3。它的阈值需要适当选择，使得与主分支的相似度最大。通过穷举搜索，我们发现“ $x_3<3.5?$ ”与主分支最为相似，相似度是 8，因为有 8 个样本同时被分到左边或右边。另一个分支“ $x_2<3.5?$ ”

的相似度是 6（实际上这个分支并不对应最优的不纯度下降。选择它是因为它与主分支最相似）。因此，“$x_3 < 3.5?$”和“$x_2 < 3.5?$”分别是第一和第二替代分支。



识别时，被测样本中若包含 x_1，则首先用主分支。如果被测样本中 x_1 的特征缺失，就需用第一替代分支来判决。如果被测样本中 x_3 的特征也缺失，就只能使用第二替代分支了。

以上只描述了根节点的分支的选择和使用。其他节点的选择和使用的做法是类似的。

样本属性的缺失未必都是坏事，有时反而能提供某些信息。比如在医疗诊断中，一个属性（如血糖水平）缺失了，也许意味着医师因为什么特殊原因而没有测量它。这样，属性缺失可作为一个新特征用于分类中。

7．4 其他决策树方法

上面讨论的基本技术可以用到几乎所有的树分类器中去。实际上，我们上面的讨论已经超出了最原始的 CART 所涉及的核心技术了。大多数的树生长算法都选用了熵不纯度的公式。但对于停止规则、剪枝方法和属性缺失数据的处理方法，有多种不同的选择方案。下面讨论另外两种主要的树算法。

7．4．1　ID3

ID3 的名称由来是因为它是“交互式二分法”程序的第 3 版（interactive dichotomizer-3）。其设计意图是为了处理名义数据。如果要解决的问题涉及实值变量，则首先把实值变量离散化，然后把它当作无序的名义属性来处理。每个节点的分支数等于其属性的取值个数。在 ID3 的应用中，由于很少只用二叉树，所以常需利用公式（7－7）计算不纯度。决策树一直生长，直到所有叶节点都为纯，或者没有其他可以使用的变量为止。在 ID3 的标准版本中没有剪枝操作，实际上，也可以运用前面提到的剪枝技术。

7. 4.2　C4. 5

C4.5 算法是最受重视的分类树方法。该算法对于实值变量的处理和 CART 一样，对名义属性采用多重分支，不纯度的计算采用的是公式（7－7）。

C4.5 与 CART 的一个显著差别是对属性缺失数据的处理上。在训练阶段，C4.5 并没有计算替代分支。因此在识别阶段，一个待测试样本如果在分支率为 B 的节点 N 处不具有该节点要求的属性时，C4.5 就把该样本分到所有 B 个分支中去进行下一次的判别，直到最后的叶节点。因此，一个属性缺失的数据可能最后到达 M 个叶节点。最终的分类结果是这 M 个叶节点的加权决策的结果，其中的权值是训练集样本在节点 N 处

的判决概率。 N 的每一个后继节点都可以看作是一棵子树的根。这种处理缺失属性的方案，就是用训练样本在 N 处的判决概率 $P(N)$，对每一个子树进行加权。因为 C4.5 中没有使用替代分支，因而也不必存储它们，所以该算法要比 CART 存储量小。

C4.5 算法能够实现基于规则的剪枝。在 7．2 节已经讨论过，一棵决策树的每一个叶子节点对应一条规则。这条规则就是从根节点到这个叶节点的路径上的所有决策的逻辑合取式。这样，一棵树对应一个规则集合。其中可能存在一些冗余判决，也可能存在一些“过细”的规则。因此，如果消去不“必要”的规则前提，会简化规则而不影响分类能力。

规则的修剪算法如下：对于一条待修剪的规则，尝试删除其任何一个前件而得到不同的几个规则。从这几个规则中选择一个使得性能提高最大的规则作为这一步的修剪后的结果。重复该修剪过程，直到这条规则不能继续被修剪为止。这个规则修剪过程对于规则集合中所有规则修剪一遍，然后算法结束。

我们用下面的例子来说明规则剪枝的过程。考虑图 7—7 下方树的最左边的叶子，它对应下述规则：

$$\begin{aligned} IF[\quad & (0.40x_1 + 0.16x_2 \quad < \ 0.11) \\ AND \quad & (0.27x_1 - 0.44x_2 \quad < -0.02) \\ AND \quad & (0.96x_1 - 1.77x_2 \quad < -0.45) \\ AND \quad & (5.43x_1 - 13.33x_2 \quad < -6.03)] \\ THEN \quad & \mathbf{x} \in \omega_1. \end{aligned}$$

这条规则有 4 个前件。如果删除任何一个前件，就可以得到 4 条不同的规则。现在测试一下这 4 条规则的性能。发现，如果删除第一个前件 $0.40x_1 + 0.16x_2 < 0.11$，和删除最后一个前件 $5.43x_1 - 13.33x_2 < -6.03$ 得到的两条规则性能下降了，而剩下的两条规则性能没有发生变化。因此从剩下的两条规则中任意选择一条规则作为修剪后的规则：

$$\begin{aligned} IF[\quad & 0.40x_1 + 0.16x_2 < 0.11 \\ AND \quad & 0.27x_1 - 0.44x_2 < -0.02 \\ AND \quad & 5.43x_1 - 13.33x_2 < -6.03] \\ THEN \quad & x \in w_1 \end{aligned}$$

继续上面的过程，最后这条规则可以简化为:

$$\begin{aligned} IF[\quad & (0.40x_1 + \ 0.16x_2 \quad < 0.11) \\ AND \ & (5.43x_1 - 13.33x_2 \quad < -6.03)] \\ & THEN \quad \mathbf{x} \in \omega_1, \end{aligned}$$

剪枝以后我们需要对修剪后的所有规则根据其性能排序，性能最好的排在最前面。对于一个测试样本，依次尝试这些规则。一旦找到符合条件的规则，就对测试样本分类。

如果采用传统的剪枝技术，节点 N 要么被消去要么被保留。而在规则剪枝中，则可以根据具体的输入模式来决定该节点是否需要保留。因此该技术的好处之一在于它允许

在特定节点 N 处能够考虑上下文信息。

值得注意的是即使靠近根节点处的节点也有可能被剪枝。这比利用叶子合并的剪枝技术更加通用。

另外，简化了的规则可用于更好的类别表达。尽管规则剪枝并非原始的 CART 方法的组成部分，但它很容易嵌入到 CART 之中。

7．5 小结

在本小节，我们对树分类器中几种不同的实现步骤做对比研究。在使用树分类器时设计人员可以选用任何适当的特征预处理技术、不纯度度量方法、树停止生长准则及剪枝技术来构建一棵分类树，而不必局限于本章描述的方法。如果设计人员对特征预处理有深入的理解，那么就应该充分利用其拥有的知识。在 ID3 的方法中，对实值变量离散化后并没有用到数据次序的信息，如果不会导致计算代价太大的话，就应该尝试利用次序信息。熵不纯度在很多情况下非常有效。实际中，树的生长停止准则和剪枝算法，要比不纯度函数的选择对最终分类准确率影响更大。不仅如此，剪枝技术要比分支停止技术或交叉验证技术用得更多，因为它充分利用了训练集的样本。当然，使用很大的样本集时采用剪枝技术计算代价很高。对于含较多噪声的问题，规则剪枝用的相对较少。但是，如果模式的确是利用规则产生的，它确实可以得到简单的分类器。

对很多应用问题来说，树分类器要比某些分类器，如神经网络分类器，能取得更准确的分类结果。特别是当我们不知道应该选择什么分类器时，使用树分类器会得到不错的结果。对于非度量数据，树分类器特别有用。也正因为这个理由，它成为模式识别研究中一个重要工具。

树分类器的许多内容都来自对概念学习系统（Concept Learning System，CLS）的研究（文献［42］），有关 CART 的一本重要的书［10］为它提供了坚实的统计基础。Quinlan 对树分类器研究得比较早，他提出了 ID3［66］、C4.5［69］，并利用最小描述长度（minimum description length）准则来剪枝[56，71]。文献［61］是一个好的综述，文献［11］给出了多元决策树方法的比较。基于概率的分支和剪枝方面的讨论可参考［53］。Gini 指标最早出现在类别数据的方差分析中［47］。文献［85］给出了决策树的增量学习或在线学习算法。有关决策树中属性缺失的问题请参考［10］和［67］，其中提出的方法比本教材的方法更具有普遍性。

由于决策树分类器不够稳定，因此在此基础上发展了决策森林方法。读者可以在第 12 章找到相关的内容。

参考文献：

[l] Alfred V Aho, John E. Hopcroft, and Jeffrey D. Ull-man. The Design and Analysis of Computer Algorithms. Addison-Wesley, Reading, MA, l974.

[2] Alfred V Aho, John E. Hopcroft, and Jeffrey D. Ullman. Data Structures and Algorithms. Addison-Wesley, Reading, MA, 1987.

[3] Alfred V Aho and Jeffrey D. Ullman. The Theory of Parsing Translation, and Compiling, volume 1: Parsing. Prentice-Hall, Englewood Cliffs, NJ, l972.

[4] Alberto Apostolico and Raffaele Giancarlo. The Boyer-Moore-Galil string searching strategies revisited. SIAM Journal of Computing, 5l(l)t98--l05, l986.

[5] Ricardo Baeza-Yates. Algorithms for string searching: A survey. SIGIR Forum, 23(3,4):34--58, l989.

[6] Alan A. Bertossi, Fabrizio Luccio, Elena Lodi, and Linda Pagli. String matching with weighted errors. Theoretical Computer Science, 73(3)t3l9--328, l990.

[7] Anselm Blumer, Janet A. Blumer, Andrzej Ehrenfeucht, David Haussler, Mu-Tian Chen, and Joel I. Seif eras. The smallest automaton recognizing the subwords of a text. Theoretical Computer Science, 40(1).3 l--55, l985.

[8] Taylor L. Booth and Richard A. Thompson. Applying probability measures to abstract languages. IEEE Transactions on Computers, C-22(5).442--450, l973.

[9] Robert S. Boyer and J. Strother Moore. A fast string-searching algorithm. Communications of the ACM, 20(l0)t762--772, l977.

[l0] Leo Breiman, Jerome H. Friedman, Richard A. O1-shen, and Charles J. Stone. Classification and Regression Trees. Chapman & Hall, New York, l993.

[11] Carla E. Brodley and Paul E. Utgoff. Multivariate decision trees. Machine Learning, l9(l)t45--77, l995.

[l2] Duncan A. Buell, Jeffrey M. Arnold, and Walter J. Klein-felder, editors. Splash 2: FPGAs in a Custom Computing Machine. IEEE Computer Society, Los Alamitos, CA, l996.

[l3] Horst Bunke, editor. Advances in Structural and Syntactic Pattern Recognition. World Scientific, River Edge, NJ, l992.

[l4] Horst Bunke and Alberto Sanfeliu, editors. Syntactic and Structural Pattern Recognition: Theory and Applications. World Scientific, River Edge, NJ, l990.

[15] Wray L. Buntine. Decision tree induction systems: A Bayesian analysis. In Laveen N. Kanal, Tod S. Levitt, and John F Lemmer, editors, Uncertainty in Artificial Intelligence 3, pages l09--l 27. North-Holland, Amsterdam, l989.

Il6] Noam Chomsky. Syntactic Structures. Mouton, The Hague, Netherlands, l957.

[l7] Peter Clark and Tim Niblett. The CN2 induction algorithm. Machine Learning, 3(4)t26l--284, l989.

[l8] Richard Cole. Tight bounds on the complexity of the Boyer-Moore string matching algorithm. In Proceedings of the Second Annual ACMSIAM Symposium on Discrete Algorithms, pages 224--233, San Francisco, CA, l99l.

[19] Livio Colussi, Ziv Galil, and Raffaele Giancarlo. On the exact complexity of string matching. In Proceedings of the 31st Annual Symposium on Foundations of Computer Science, pages l 35--l44, l990.

[20] Craig M. Cook and Azriel Rosenfeld. Some experiments in grammatical inference. In Jean-Claude Simon, editor, Computer Oriented Learning Processes, pages l57--l7 l. Springer-Verlag, NATO ASI, New York, l974.

[2l] Thomas H. Cormen, Charles E. Leiserson, and Ronald L.Rivest. Introduction to Algorithms. MIT Press, Cambridge, MA, l990.

[22] Stefano Crespi-Reghizzi. An effective model for grammatical inference. In C. V Freiman, editor, Proceedings of the International Federation for Information Processing (IFIP) Congress, Ljubljana Yugoslavia, pages 524-529. North-Holland, Amsterdam, l972.

[23] Maxime Crochemore and Wojciech Rytter Text Algorithms. Oxford University Press, Oxford, UK, l994.

[24] Min-Wen Du and Shih-Chio Chang. Approach to designing very fast approximate

string matching algorithms. M Transactions on Knowledge and Data Engineering, KDE--6(4)f62M33, l994.

[25] Richard O. Duda and Peter E. Hart. Experiments in scene analysis. Proceedings of the First National Symposium on Industrial Robots, pages l l9--l30, l970.

[26] Richard M. Durbin, Sean R. Eddy, Anders Krogh, and Graeme J. Mitchison. Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids. Cambridge University Press, Cambridge, UK, l998.

[27] John Durkin. Induction via ID3. AI Expert, 7(4)t48--53, l 992.

[28] Jay Earley. An efficient context-free parsing algorithm. Communications of the ACM, l3t94--l02, l970.

[29] Bruce G. Buchanan, Edward A. Feigenbaum and Joshua Lederberg. On generality and problem solving: A case study using the DENDRAL program. In Bernard Meltzer and Donald Michie, editors, Machine Intelligence 6, pages l65--l90. Edinburgh University Press, Edinburgh, Scotland, l97l.

[30] Usama M. Fayyad and Keki B. Irani. On the handling of continuous-valued attributes in decision tree generation. Machine Learning, 8:87--l02, l992.

[3l] Edward A. Feigenbaum and Bruce G. Buchanan. DEN-DRAL and Meta-DENDRAL: Roots of knowledge systems and expert system applications. A1pcial Intelligence, 59(l--2)t233--240, l993.

[32] King-Sun Fu. Syntactic Pattern Recognition and Applications. Prentice-Hall, Englewood Cliffs, NJ, l982.

[33] King-Sun Fu and Taylor L. Booth. Grammatical inference: Introduction and Survey-Part I. IEEE Transactions on Pattern Analysis and Machine Intelligence, PAML 8(3)t343--359, l986.

[34]King-Sun Fu and Taylor L. Booth. Grammatical inference: Introduction and Survey-part II. IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-8(3):36th375, l986.

[35] Zvi Galil and Joel I. Seiferas. Time-space-optimal string matching. Joumal of Computer and System Sciences, 26(3):280--294, l983.

[36] E. Mark Gold. Language identification in the limit. Information and Control, l0:447--474, 1967.

[37] Dan Gusfield. Algorithms on Strings, Trees and Sequences. Cambridge University Press, New York, l997.

[38] Peter E. Hart, Richard O. Duda, and Marco T. Einaudi. PROSPECTOR--a computer-based consultation system for mineral exploration. Mathematical Geology, l0(5):589--6l0, l978.

[39] John E. Hopcroft and Jeffrey D. Ullman. Introduction to Automata Theory Languages, and Computation. Addison-Wesley, Reading, MA, l979.

[40] R. Nigel Horspool. Practical fast searching in strings. Software-Practice and Experience, l0(6):501--506, l980.

[41] Xiaoqiu Huang. A lower bound for the edit-distance problem under an arbitrary cost function. Information Processing Letters, 27(6):3l9--321, l988.

[42] Earl B. Hunt, Janet Marin, and Philip J. Stone. Experiments in Induction. Academic Press, New York, 1966.

[43] Zvi Kohavi. Switching and Finite Automata Theory. McGraw-Hill, New York, second edition, 1978.

[44] Pat Langley. Elements of Machine Learning. Morgan Kaufmann, San Francisco, CA, l996.

[45] Karim Lari and Stephen J. Young. Estimation of stochastic context-free grammars using the inside-out algorithm. Computer Speech and Language, 4(l):35--56, l990.

[46] Nada Lavra6 and Saso D eroski. Inductive Logic Programming: Techniques and Applications. Ellis Horwood (Simon & Schuster), New York, l994.

[47] Richard J. Light and Barry H. Margolin. An analysis of variance for categorical data. Journal Of the American Statistical Association, 66(335).534--544, l97l.

[48] Dan Lopresti and Andrew Tomkins. Block edit models for approximate string matching. Theoretical Computer Science, l81(1); l59--179, 1997.

[49] M. Lothaire, editor. Combinatorics on Words. Cambridge University Press, New York, second edition, l997. M. Lothaire is a joint pseudonym for the following: Robert Cor, Dominque Perrin, Jean Berstel, Christian Choffrut, Dominque Foata, Jean Eric Pin, Guiseppe Pirillo, Christophe Reutenauer, Marcel R Schutzen-berger, Jadcques Sakaroovitch, and Imre Simon.

[50] Shin-Yee Lu and Xing-Sun Fu. Stochastic error-correcting syntax analysis and recognition of noisy patterns. IEEE Transactions on Computers, F-26(12)t l268--l276, 1977.

[5l] Maurice Maes. Polygonal shape recognition using string matching techniques. Pattern Recognition, 24(5):433-440, l99l.

[52] Ramon L pez De M ntaras. A distance-based attribute selection measure for decision tree induction. Machine Learning, 6t8l--92, l99l.

[53] J. Kent Martin. An exact probability metric for decision tree splitting and stopping. Machine Learning, 28(2-3):257--29l, l997.

[54] Andres Martin and Enrique Vidal. Computation of nor malized edit distance and applications. IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-l5(9).92fu932, l993.

[55] Gerhard Mehlsam, Hermann Kaindl, and Wilhelm Barth. Feature construction during tree learning. In Klaus P Jantke and Steffen Lange, editors, Algorithmic Learning for Knowledge-Based Systems, volume 96l of Lecture Notes in Artificial Intelligence, pages 39l--403. Springer-Verlag, Berlin, l995.

[56] Manish Mehta, Jorma Rissanen, and Rakesh Agrawal. MDL-based decision tree pruning. In Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD '95), pages 2l 6--22l, l995.

[57] Ryszard S. Michalski. AQVAL/lf Computer implementation of a variable valued logic system VLl and examples of its application to pattern recognition. In Proceedings of the First International Conference on Pattern Recognition, pages 3--17, l973.

[58] Ryszard S. Michalski. Pattern recognition as rule-guided inductive inference. IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-2(4):349--36l, l980.

[59] Laurent Miclet. Grammatical inference. In Horst Bunke and Alberto Sanfeliu, editors, Syntactic and Structural Pattern Recognition: Theory and Applications, chapter 9, pages 237--290. World Scientific, River Edge, NJ, l990.

[60] William F Miller and Alan C. Shaw. Linguistic methods in picture processing--A

survey. Proceedings of the Fall Joint Computer Conference FJCC, pages 279--290, l 968.

[6l] Tom M. Mitchell. Machine Learning. McGraw-Hill, New York, l997.

[62] Roger Mohr, Theo Pavlidis, and Alberto Sanfeliu, editors. Structural Pattern Recognition. World Scientific, River Edge, NJ, l990.

[63] Stephen Muggleton, editor. Inductive Logic Programming. Academic Press, London, UK, l992.

[64] Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino-acid sequence of two proteins. Journal of Molecular Biology, 48(3):443-53, l970.

[65] Partha Niyogi. The Informational Complexity of Learning: Perspectives on Neural Networks and Generative Grammar. Kluwer, Boston, MA, l998.

[66] J. Ross Quinlan. Learning efficient classification procedures and their application to chess end games. In Ryszard S. Michalski, Jaime G. Carbonel1, and Tom M. Mitchel1, editors, Machine Learning: An Artificial Intelligence Approach, pages 463--482. Morgan Kaufmann, San Francisco, CA, l983.

[67] J. Ross Quinlan. Unknown attribute values in induction. In Alberto Maria Segre, editor Proceedings of the Sixth. Internatinal Workshop on Machine Learning, pages l64--l68. Morgan Kaufmann, San Mateo, CA, l989.

[68] J. Ross Quinlan. Learning logical definitions from relations. Machine Learning, 5(3):239--266, l990.

[69] J. Ross Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann, San Francisco, CA, l993.

[70] J. Ross Quinlan. Improved use of continuous attributes in C4.5. Journal of Artificial Intelligence, 4:77--90, l996.

[7l] J. Ross Quinlan and Ronald L. Rivest. Inferring decision tress using the minimum description length principle. In-formation and Computation, 80(3):227--248, l989.

[72] Stephen V Rice, Horst Bunke, and Thomas A. Nartker. Classes of cost functions for string edit distance. Algo-rithmica, l8(2):27l--l80, l997.

[73] Eric Sven Rivest and Peter N. Yianilos. Learning string-edit distance. IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-20(5).522--532, l998.

[74] Ron L. Rivest. Learning decision lists. Machine Leaning, 2(3)'229--246, l987.

[75] Stuart Russell and Peter Norvig. Artificial Intelligence: A Modern Approach. Prentice-Hall Series in Artificial intelligence. Prentice--Ha1l, Englewood Cliffs, NJ, l995.

[76] Wojciech Rytter. On the comp1exity of parallel parsing of general context-f fee languages. Theoretical Computer Science, 47(3):3 I 5--32l, 1986.

[77] David Sank off and Joseph B. Kruskal. Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison. Addison-Wesley, New York, 1983.

[78] Janet Saylor and David G. Stork. Parallel analog neural networks for tree searching. In John S. Denker, editor, Neural Networks for Computing, pages 392--397. American Institute of Physics, New York, l986.

[79] Edward H. Shortliffe, editor. Computer-Based Medical Consultations: MYCIN. Elsevier/North-Holland, New York, l976.

[80] Temple F Smith and Michael S. Waterman. Identification of common molecular

sequences. Journal of Molecular Biology, l47. l95--l97, l98l.

[81J Ray J. Solomonoff A new method for discovering the grammars of phrase structure languages. In Proceedings of the International Conference on Information Processing, pages 285--290, l959.

[82] Graham A. Stephen. String Searching Algorithms, volume 3. World Scientific, River Edge, NJ, Lecture Notes on Computing, l994.

[83] Daniel Sunday. A very fast substring search algorithm. Communications of the ACM, 33(8): l32--l42, l990.

[84] Eiichi T8naka and Xing-Sun Fu. Error-correcting parsers for formal languages. IEEE Transactions on Computers, c--27(7)f605All6, l978.

[85] Paul E. Utgoff. Incremental induction of decision trees. Machine Learning, 4(2): l6l--l86, l989.

[86] Paul E. Utgoff. Perceptron trees f A case study in hybrid concept representations. Connection Science, l(4):377-391, l989.

[87] Uzi Vishkin. Optimal parallel pattern matching in strings. Information and Control, 67:91--l l3, l985.

[88] Robert A. Wagner and Michael J. Fischer The string-to-string correction problem. Journal of the ACM 2l(1): l68--l78, l974.

[89] Sholom M. Weiss and Nitin Indurkhya. Decision tree pruning f Biased or optimal? In Proceedings of the l2th National Conference on Artificial Intelligence, volume l, pages 626--632. AAAI Press, Menlo Park, CA,USA l994.

[90] Allan P White and Wei Zhong Liu. Bias in information-based measures in decision tree induction. Machine Learning, l5(3):32l--329, l994.

[9l] Patrick Henry Winston. Learning structural descriptions from examples. In Patrick Henry Winston, editor, The Psychology of Computer Vision, pages l57--209. McGraw-Hill, New York, l975.