第 11 章 决策树

KNN 算法是一种简便的非参数方法，但对于噪音变量并不稳健。

根本原因在于，KNN 在寻找邻居时，并未考虑响应变量 y 的信息。

决策树(decision tree)本质上也是一种近邻方法，可视为“自适应近邻法”(adaptive nearest neighbor)。

由于决策树在进行节点分裂时考虑了 y 的信息，故更有“智慧”，不受噪音变量的影响，且适用于高维数据。

决策树的思想与雏形形成于 1960 年代，而成熟于 1980 年代。在统计学领域，以 Breiman, Friedman, Stone and Olshen(1984) 的经典著作 *Classification and Regression Trees* 为代表，简称 CART 算法。

在计算机领域，则以 Quinlan (1979, 1986)为代表，提出 ID3 算法(Iterative Dichotomiser 3)，后演变为 C4.5 与 C5.0 算法 (C 表示 Classifier)。这两种算法比较类似，效果接近。

本书主要介绍 CART 算法，但也提及 C5.0 算法。

如果将决策树用于分类问题，则称为**分类树**(classification tree)。

如果将决策树用于回归问题，则称为**回归树**(regression tree)。

11.1 分类树的启发案例

Breiman et al. (1984)研究了 UCSD 医学中心的一个案例。

当心梗病人进入 UCSD 医学中心后 24 小时内，测量 19 个变量，包括血压、年龄以及 17 个排序或虚拟变量。数据集中包含 215 个病人，其中 37 人为高危病人。

研究目的是为了快速预测哪些心梗病人为“高危病人”(High risk，记为 H，无法活过 30 天)，而哪些是“低危病人” (Low risk，记为 L，可活过 30 天)，从而更好地配置医疗资源。

Breiman et al. (1984)建立了如下分类树，参见图 11.1。

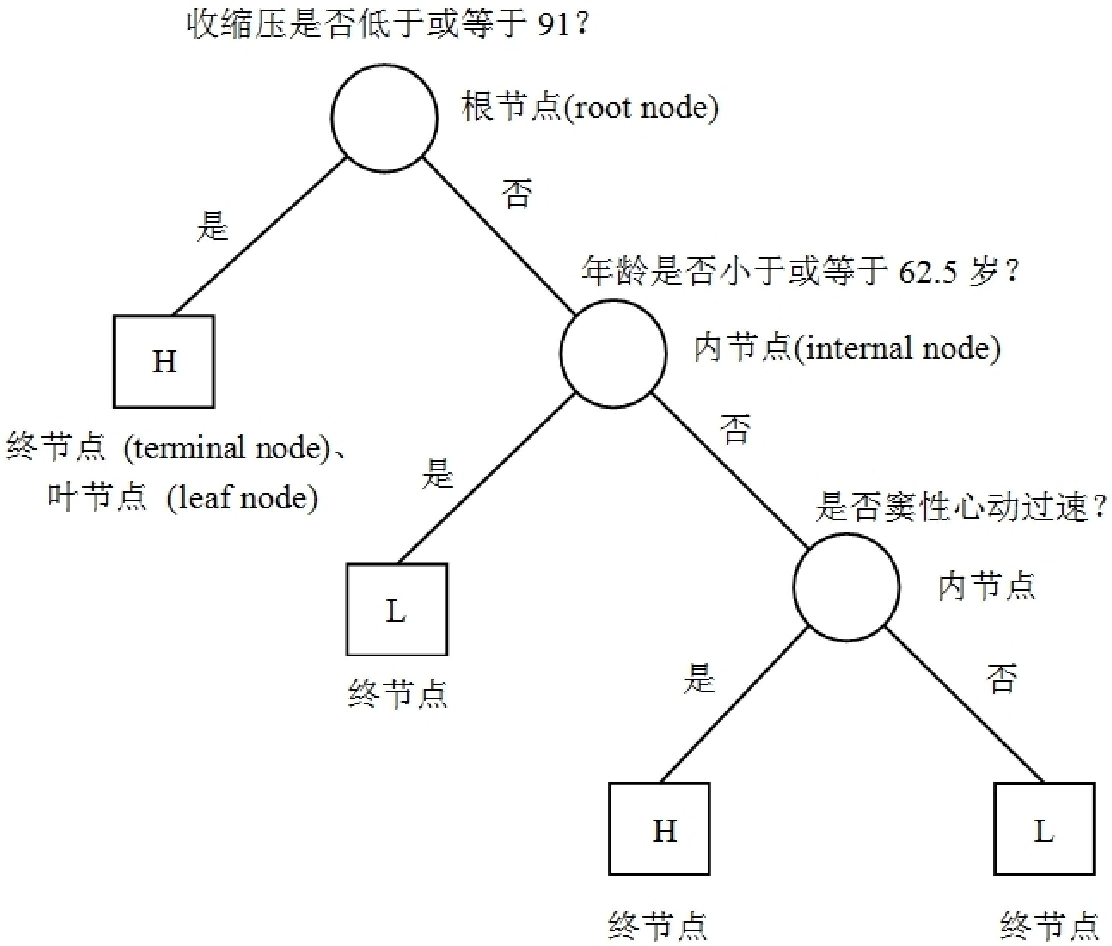

图 11.1 UCSD 医学中心判断高危病人的分类树

在图 11.1 中，首先从顶部的**根节点**(root node)出发，考察病人“收缩压是否低于或等于 91”。

如果答案为“是”，则向左，到达**终节点**(terminal node)或**叶节点**(leaf node)，归类为 H(高危)；反之，则向右，到下一个**内节点**(internal node)。

此时，需回答的问题为“年龄是否小于或等于 62.5 岁”。如何答案为“是”，则向左，到达终节点，归类为 L(低危)；反之，则继续向右，到再下一个内节点。

此时，需回答的问题为“是否窦性心动过速”。如何答案为“是”，则向左，到达终节点，归类为 H(高危)；反之，则继续向右，到达终节点，归类为 L(低危)。

在图 11.1 中，终节点(叶节点)为方形表示，为决策树的最底端，不再分裂。

在分裂之前，所有样本点都在最顶端的根节点，而根节点与终节点之间的节点均称为“内节点”。

建立分类树模型之后，要进行预测十分简单。只要将观测值从决策树放下(drop an observation down the tree)，回答一系列的是或否问题(是则向左，否即向右)，看它落入哪片叶节点。

然后使用“多数票规则”(majority vote rule)进行预测，即看落入该叶节点的训练数据最多为哪类。

Breiman et al. (1984)发现，由于决策树不对函数形式作任何假设，故比较稳健，其预测效果可能优于参数方法(比如判别分析、逻辑回归)。

只要决策树不过于枝繁叶茂(a bushy tree)，则其可解释性(interpretability)很强，甚至不用数学方程！

在此案例，虽然数据集共有 19 个特征变量，但所估计的分类树只用到 3 个变量。

通过图 11.1 的决策树模型，可清晰地知道高危病人与低危病人的类型。比如，模型所识别的高危病人可分为两种类型，即收缩压低于或等于 91 者(血压过低)；或收缩压虽高于 91，但年龄大于 62.5 岁，且窦性心动过速者。

11.2 二叉树的数学本质

CART 算法所用的决策树为**二叉树**(binary tree)，即每次总是将“母节点”(parent node)一分为二，分裂(split)为两个“子节点”(child node)，直至到达终节点。

本质上，二叉树将“特征空间”(feature space，也称 predictor space)进行**递归分割**(recursive partitioning)，每次总是沿着与某个特征变量x_j轴平行的方向进行切割(axis-parallel splits)，切成“矩形”(rectangle)或“超矩形”(hyper-rectangle)区域，即所谓“箱体”(boxes)。

分类树是一种通过分割特征空间进行分类的分类器(classifier as partition)。

假设只有两个特征变量(x_1, x_2)，则递归分裂的一种可能结果如图 11.2。

首先，根据是否$x_1 \leq t_1$进行分裂。

其次，根据是否$x_2 \leq t_2$进行分割，得到终节点R_1与R_2。

再次，根据是否$x_1 \leq t_3$进行分裂，得到终节点R_3。

最后，根据是否$x_2 \leq t_4$进行分割，得到终节点R_4与R_5。相应的决策树参见图 11.3。

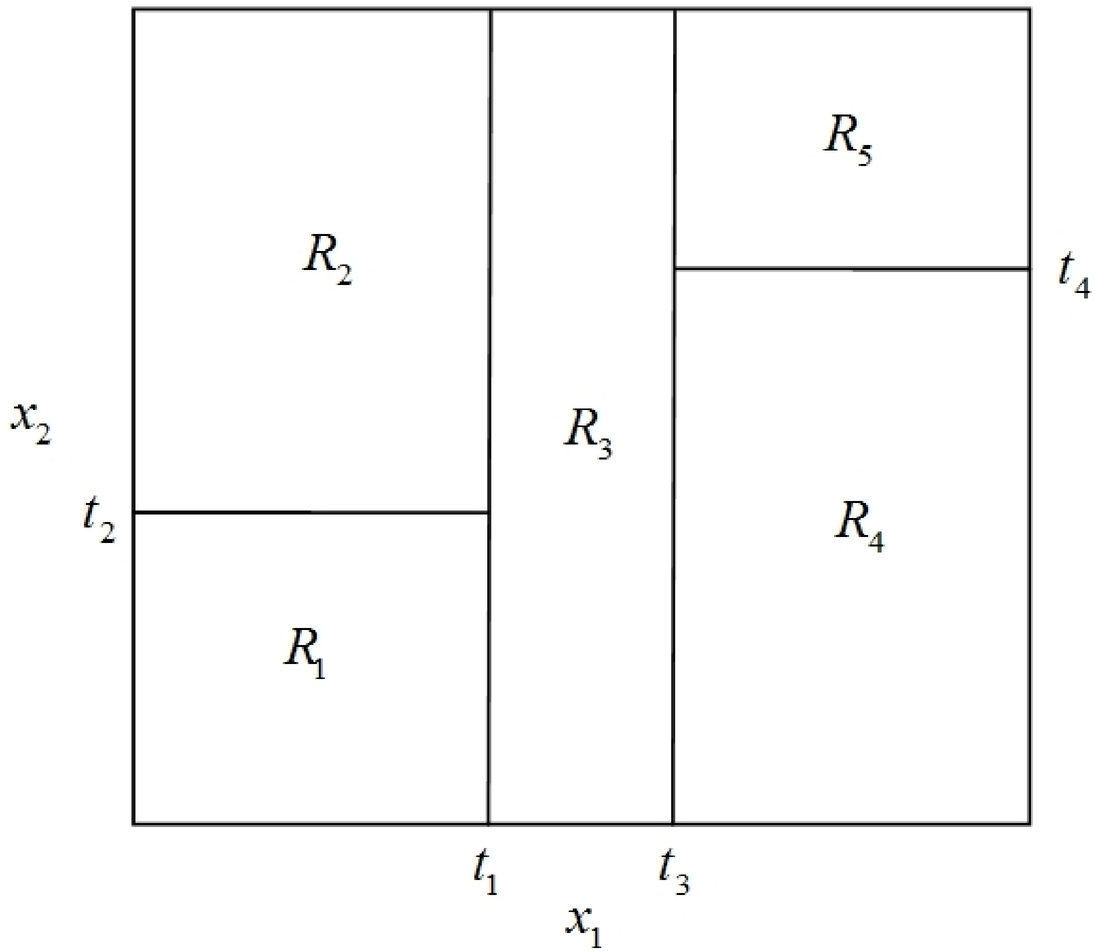

图 11.2 决策树对特征空间的递归分割

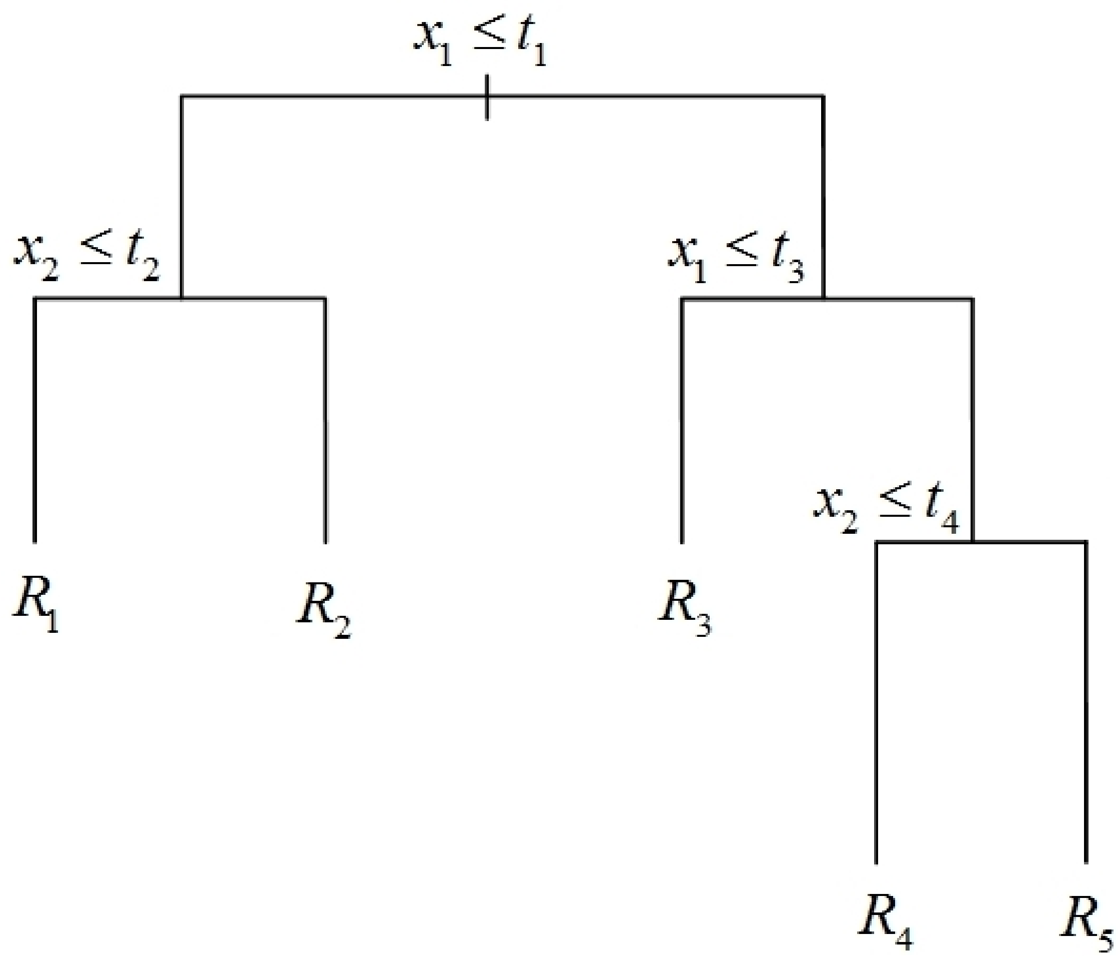

图 11.3 两个特征变量的决策树

对于三维以上的特征空间，无法使用类似于图 11.2 的方法来展示递归分割；但依然可用图 11.3 这样的树状结构来表示(这两种表示法等价)，因为决策树每次仅使用一个变量进行分裂(splitting)。

决策树模型将特征空间分割为若干(超)矩形的终节点。

在进行预测时，每个终节点只有一个共同的预测值。

对于分类问题，此预测值为该终节点所有训练样本的最常见类别(most commonly occurring class)。

对于回归问题，此预测值为该终节点所有训练样本的平均值。

在数学上，决策树为**分段常值函数**(piecewise constant function)，参见图 11.4。

这意味着，以决策树估计的函数 $\hat{f}(\mathbf{x})$ 竟然不是连续函数！

但这并不妨碍决策树成为一种灵活而有用的算法，特别是作为“基学习器”(base learner)，广泛用于随机森林与提升法(参见第 12-13 章)。

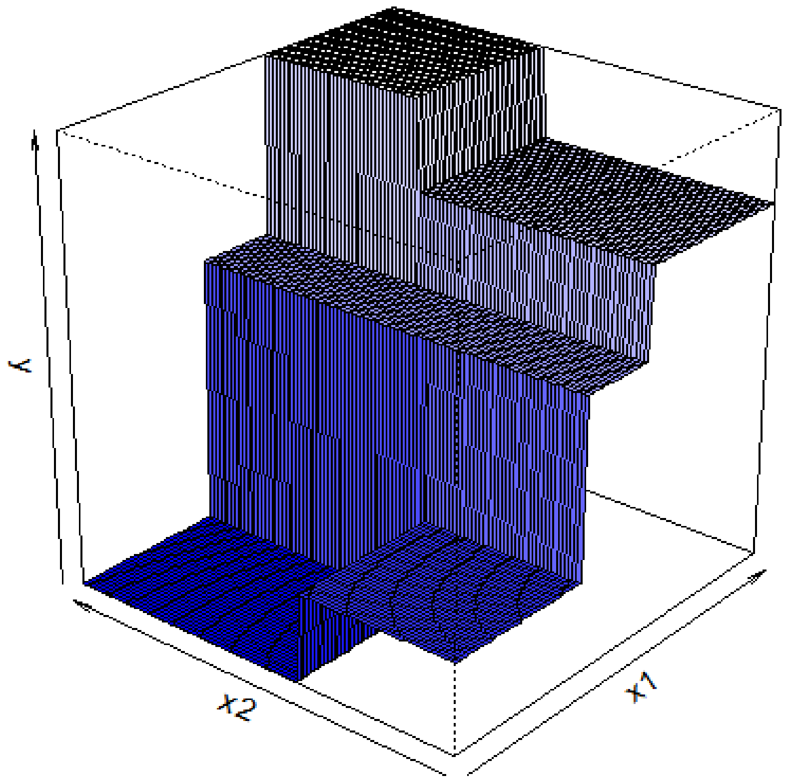

图 11.4 作为分段常值函数的决策树

在理论上，我们可以考虑任意形状的分区。比如，KNN 算法所定义的近邻区域一般是不规则的。

但决策树限制划分区域为超矩形，不仅可带来计算上的方便，而且也便于解释(每次分叉在形式上都是某变量$x_j \leq t$)。

在划分超矩形区域时，为何不一步到位，同时估计这些超矩形区域？因为这些超矩形的维度太高，同时估计在计算上并不可行。

决策树采取了一种“自上而下”(top-down)，每次仅分裂一个变量的方法。这其实是一种**贪心算法**(greedy algorithm)，因为它每次仅选择一个最优的分裂变量，而未通盘考虑全局的最优分区。

11.3 分类树的分裂准则

对于 CART 算法的二叉树，在每个节点进行分裂时，需要确定选择什么**分裂变量**(split variable)进行分裂，以及在该变量的什么临界值(cut)进行分裂。

对于分类树，我们希望在节点分裂之后，其两个子节点内部的“纯度”(purity)最高。

分裂之后，数据的“不纯度”(impurity)应下降最多。

假设响应变量y共分K类，取值为$y \in \{1, \cdots, K\}$。

在节点t(node t)，记y不同取值的相应概率(频率)为$p_1,\cdots,p_K$，其中$p_k \geq 0$，且$\sum_{k=1}^{K} p_k = 1$。

作为**分裂准则**(splitting criterion)，希望定义一个**节点不纯度函数**(node impurity function）$\varphi(p_1,\cdots,p_K) \geq 0$。该函数应具备以下性质：

(1) 当$p_1 = \cdots = p_K = \dfrac{1}{K}$时，不纯度最高，即$\varphi(p_1,\cdots,p_K)$达到最大值。

(2) 当且仅当$(p_1, p_2 \cdots, p_K) = (1,0,\cdots,0)$，$(0,1,\cdots,0)$，…，或$(0,0,\cdots,1)$时，不纯度为 0，即$\varphi(p_1,\cdots,p_K)$达到最小值 0。

(3) $\varphi(p_1,\cdots,p_K)$关于自变量$(p_1,\cdots,p_K)$是对称的。

满足这些性质的函数并不唯一。一个自然的选择是使用**错分率**(misclassification rate)作为不纯度函数：

$$\mathrm{Err}(p_1,\cdots,p_K) \equiv 1-\max\{p_1,\cdots,p_K\} \tag{11.1}$$

其中，$\max\{p_1,\cdots,p_K\}$为最多类别的发生频率，而$1-\max\{p_1,\cdots,p_K\}$则为以最多类别预测时的错分率。

对于二分类问题，错分率简化为

$$\mathrm{Err}(p_1, p_2) \equiv 1 - \max\left\{p_1, 1 - p_1\right\} = \begin{cases} p_1 & if \;\; 0 \leq p_1 < 0.5 \\ 1 - p_1 & if \;\; 1 \geq p_1 \geq 0.5 \end{cases} \tag{11.2}$$

将错分率视为 p_1 的函数，则错分率在 $p_1 = 0.5$ 处达到最大值 0.5，然后以 $p_1 = 0.5$ 为中心，向两边线性递减，呈三角形状，参见图 11.5。

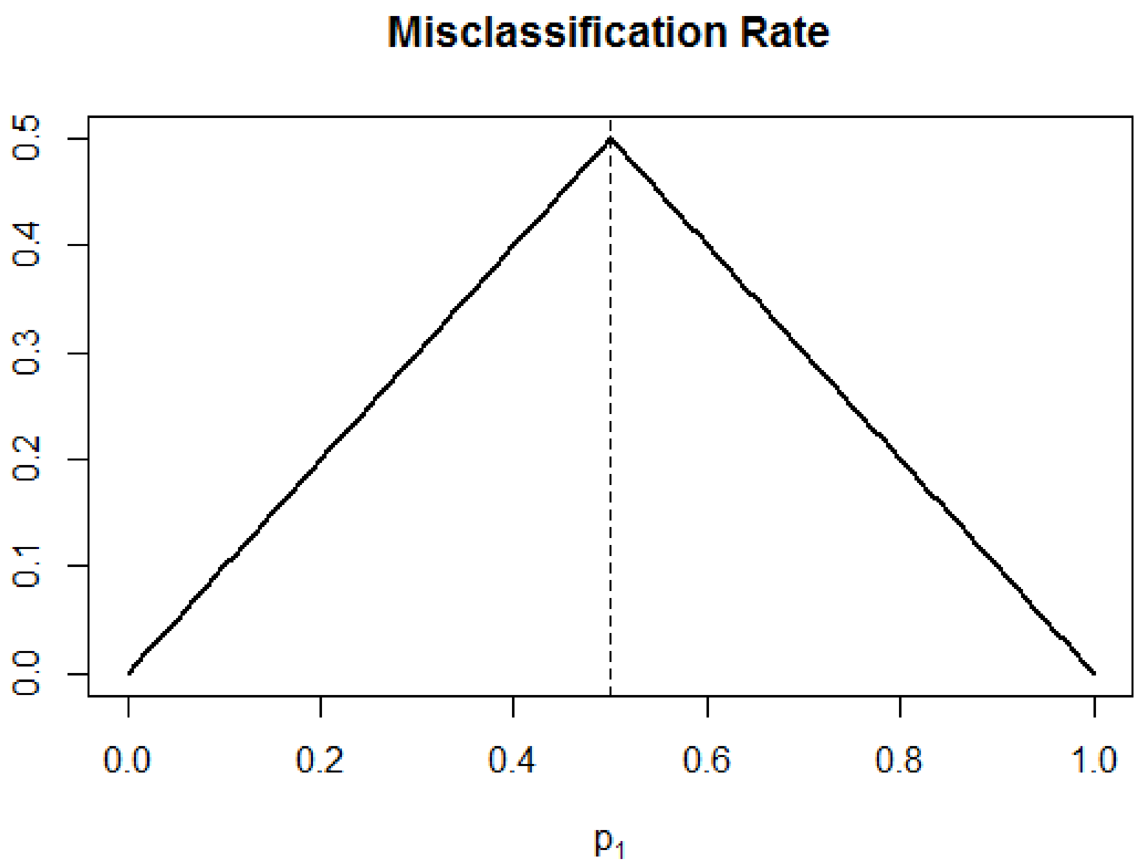

图 11.5 错分率函数

由图 11.5 可知，错分率为“分段线性函数”(piecewise linear function)，对于“不纯度”的度量并不敏感，故实际效果不太好。

实践中常用的两个不纯度函数分别为“基尼指数”与“信息熵”。

基尼指数(Gini index)度量的是，从概率分布$(p_1,\cdots,p_K)$中随机抽取两个观测值，则这两个观测值的类别不一致的概率为

$$\text{Gini}(p_1,\cdots,p_K)\equiv\sum_{k=1}^{K}p_k(1-p_k)=\underbrace{\sum_{k=1}^{K}p_k}_{=1}-\sum_{k=1}^{K}p_k^2=1-\sum_{k=1}^{K}p_k^2 \tag{11.3}$$

其中，$\sum_{k=1}^{K} p_k^2$ 可视为随机抽取的两个观测值之类别一致的概率。

对于二分类问题，基尼指数可写为

$$\text{Gini}(p_1, p_2) = 1 - p_1^2 - (1 - p_1)^2 = 2p_1(1 - p_1) \tag{11.4}$$

其中，$p_1(1 - p_1)$ 可视为两点分布(Bernoulli distribution)的方差。

在多分类的情况下，基尼指数也可解释为单独取出其中的某类，而将其他类别归并为一类，计算此两点分布的方差；然后重复此过程，将所有的方差加总。

基尼指数(11.3)为概率 p_1 的二次函数。

对于二分类问题，基尼指数(11.4)为抛物线，在 $p_1=0.5$ 处达到最大值 0.5，然后以二次曲线向两边下降，参见图 11.6。

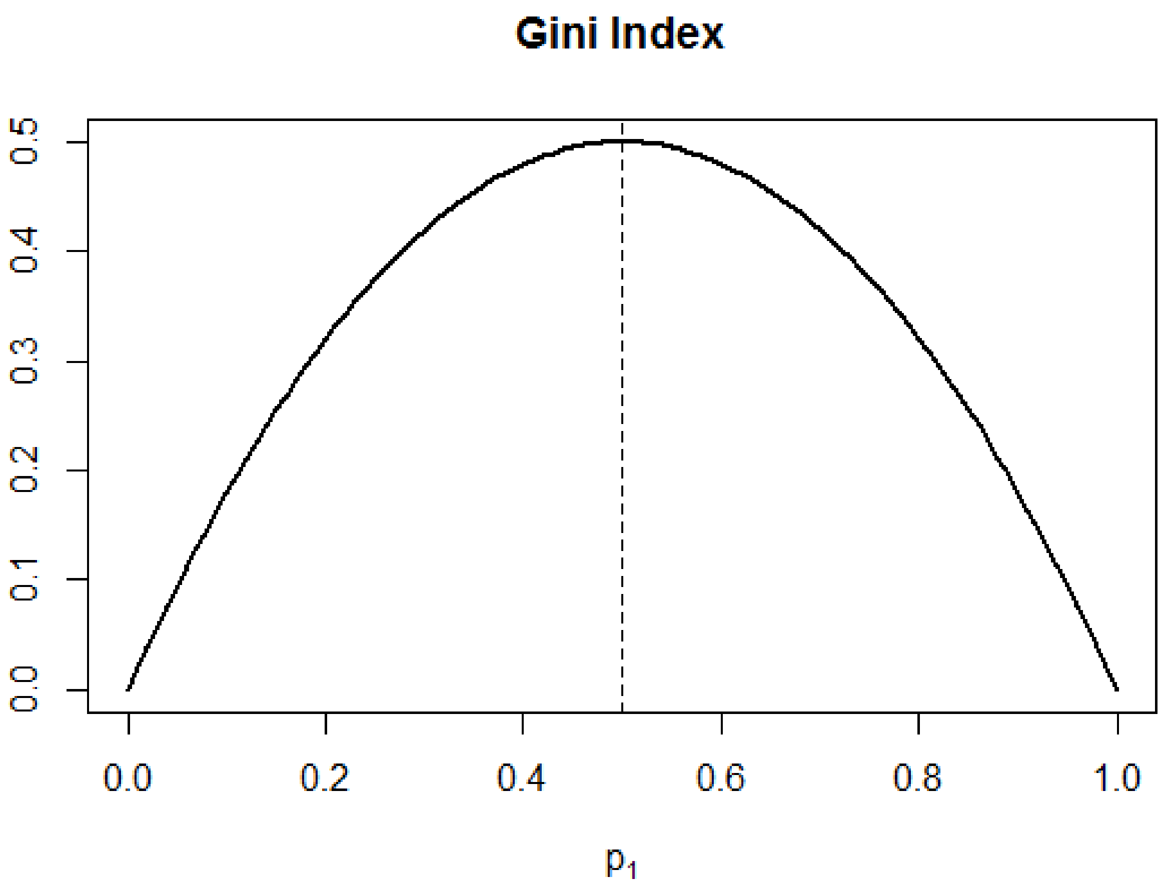

图 11.6 基尼指数

11.4 信息理论

另一常用的节点不纯度函数为**信息熵**(information entropy)。信息熵起源于信息理论(information theory) (Shannon, 1948)。

例 假如有人告诉你，“明天太阳会升起”，则此信息毫无价值，因为它是必然事件。如果有人告诉你，“明天能看见太阳”，这个信息的意义也不大，因为大多数日子都能看见太阳，你自己也很可能猜对。

假如有人告诉你，“明天会下雨”，这一消息的信息量就比较大，因为下雨的概率一般较小。而如果有人告诉你，“明天会有地震”，这条消息的信息量就很大，因为地震是稀有事件，发生概率很低。

一个随机事件的发生，其信息量似乎与它的发生概率成反比。直观上，信息度量的是“吃惊程度”(degree of surprise)。

记随机事件“$y=k$”的发生概率为p_k，初步猜想该事件发生的信息量为$1/p_k$。希望当$p_k=1$(必然事件)时，此事件的信息量为 0。一个自然的选择是取对数，即$\log_2(1/p_k)=-\log_2 p_k$。

将y的每个可能取值的信息量，以相应概率p_k为权重，加权求和即可求得期望信息量，即**信息熵**(information entropy)：

$$\text{Entropy}(p_1,\cdots,p_K) \equiv \mathrm{E}\left(-\log_2 p_k\right) = -\sum_{k=1}^{K} p_k \log_2 p_k \quad (p_k \geq 0) \tag{11.5}$$

其中，$\log_2(\cdot)$为以 2 为底的对数，其单位称为“比特”(bit，表示 binary digits)。

如果$p_k = 0$(发生概率为 0)，则定义$0 \cdot \log_2(0) \equiv 0$(因为根据洛必达法则，$\lim_{p\to 0} p \cdot \log_2 p = 0$)。

在定义式(11.5)中，也可使用以 e 为底的自然对数 $\ln(\cdot)$，其单位称为"奈特"(nat，表示 natural units)。无论使用什么底数，二者并无实质区别。

对于二分类问题，信息熵可写为

$$\text{Entropy}(p_1, p_2) = -p_1 \log_2 p_1 - (1-p_1)\log_2(1-p_1) \tag{11.6}$$

二分类问题的信息熵函数(11.6)，其几何图形参见图 11.7。

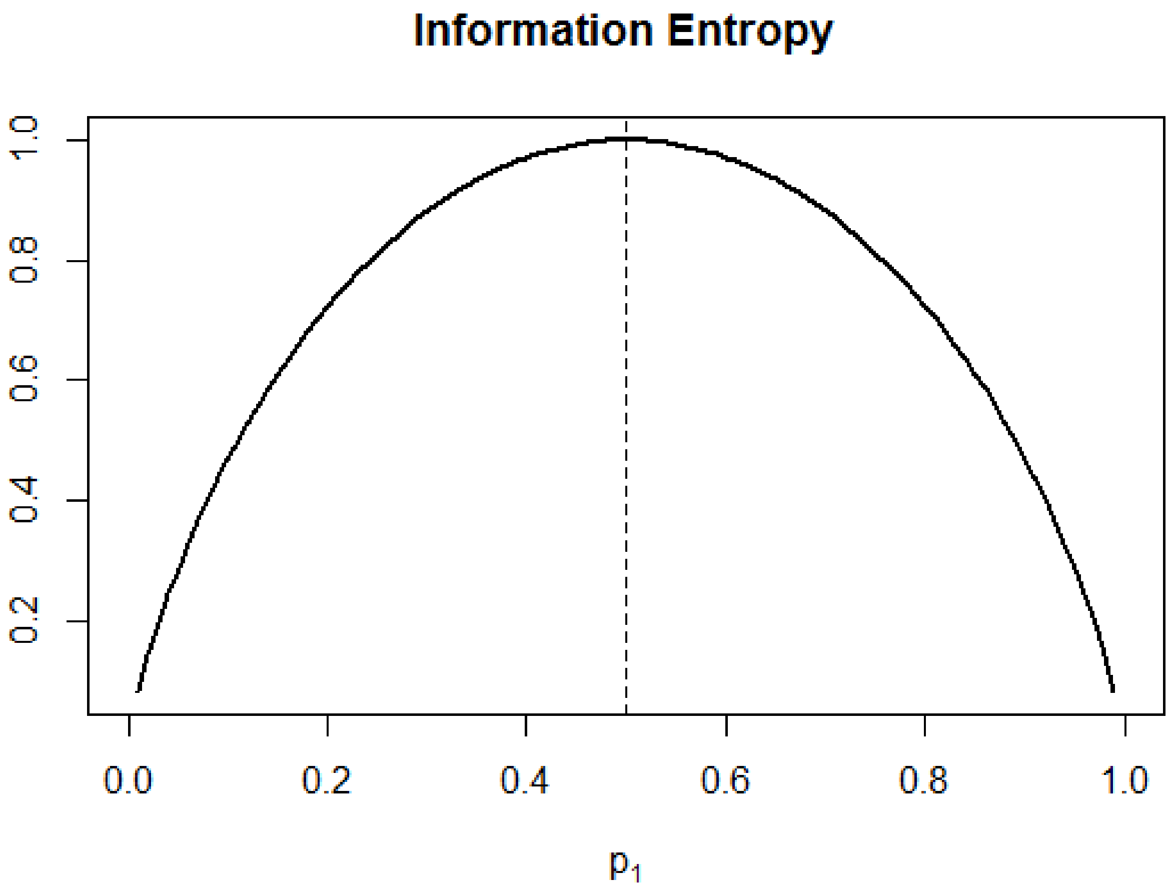

图 11.7 信息熵函数

信息熵与基尼指数在函数形状上很接近, 但信息熵的最大值为 1，而非 0.5。

图 11.8 将信息熵、基尼指数与错分率这三个节点不纯度函数画在一起。

其中，已将信息熵函数标准化，即除以 2，使得其最大值也为 0.5。

在实践中，使用基尼指数或信息熵的预测效果一般很接近。

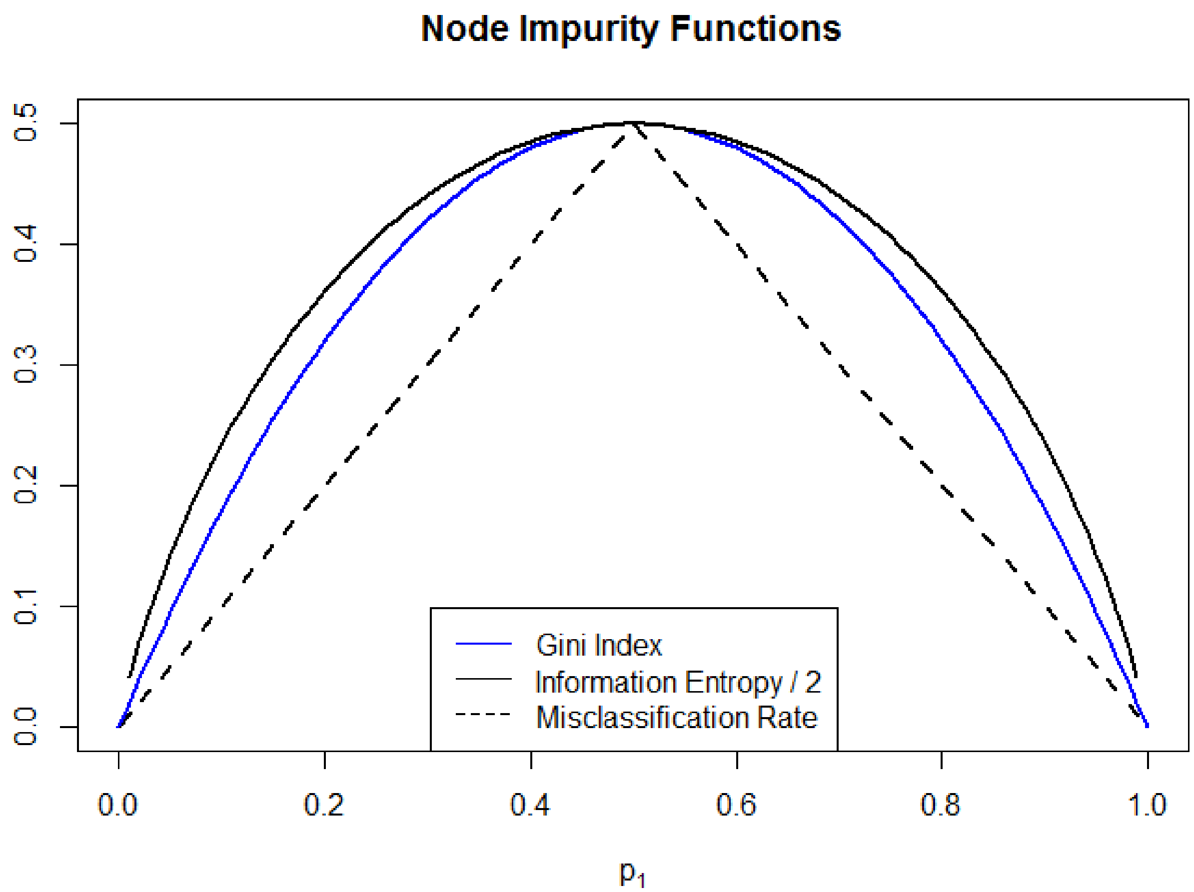

图 11.8 三种节点不纯度函数的比较

选定分裂准则(比如基尼指数)之后，在进行节点分裂时，针对每个特征变量，首先寻找其最优临界值(cut)，比如$x_j \le t$，并计算以该变量为分裂变量可带来的节点不纯度函数之下降幅度。

然后，选择可使节点不纯度下降最多的变量作为分裂变量。

11.5 成本复杂性修枝

在估计决策树模型时，面临一个选择，即何时停止节点分裂。

如果不停地进行分裂，将使得每个叶节点最终只有一个观测值(或多个相同的观测值)，此时训练误差为 0，导致过拟合，泛化预测能力下降。

需要选择一个合适的决策树规模，停止节点分裂。

但如果不让决策树长到最大，则很难知道应在何时停止分裂。

因为在某个节点进行分裂时，即使节点不纯度函数下降很少，但依然可能在以后的节点分裂中，不纯度函数大幅下降。

解决方法是，先让决策树尽情生长，记最大的树为$T_{\max}$，再进行“修枝”(pruning)，以得到一个“子树”(subtree)T。

对于任意子树$T \subseteq T_{\max}$，定义其“复杂性”(complexity) 为子树T的终节点数目(number of terminal nodes)，记为$\left|T\right|$。

为避免过拟合，不希望决策树过于复杂，故惩罚其规模$\left|T\right|$：

$$\min_{T} \underbrace{R(T)}_{cost} + \lambda \cdot \underbrace{\left|T\right|}_{complexity} \tag{11.7}$$

其中，$R(T)$为原来的损失函数，比如 0-1 损失函数(0-1 loss)，即在训练样本中，如果预测正确，则损失为 0，而若预测错误则损失为 1。

$\lambda \geq 0$为调节参数(tuning parameter)，也称为**成本复杂性参数**(cost-complexity parameter，简记 ccp)。

λ控制对决策树规模$|T|$的惩罚力度，可通过交叉验证确定。

这种修枝方法称为**成本复杂性修枝**(cost-complexity pruning)，即在成本(损失函数$R(T)$)与复杂性(决策树规模$|T|$)之间进行最优的权衡。

更具体地，将 0-1 损失函数代入表达式(11.7)可得：

$$\min_{T} \underbrace{\sum_{m=1}^{|T|} \sum_{\mathbf{x}_i \in R_m} I(y_i \neq \hat{y}_{R_m})}_{cost} + \lambda \cdot \underbrace{|T|}_{complexity} \tag{11.8}$$

其中，R_m为第m个终节点，$\hat{y}_{R_m}$为该终节点的预测值(该终节点观测值的众数)，而$I(y_i \neq \hat{y}_{R_m})$为示性函数(indicator function)。$\sum_{\mathbf{x}_i \in R_m} I(y_i \neq \hat{y}_{R_m})$表示在第 m 个终节点的损失，然后对所有终节点$m = 1, \cdots, |T|$进行加总。

如果$\lambda = 0$，则没有复杂性惩罚项，一定选择最大的树$T_{\max}$ (使训练误差为 0)，导致过拟合。

当λ增大时，则会得到一个相互嵌套的子树序列(a sequence of nested subtrees)，然后从中选择最优的子树。

11.6 回归树

将决策树应用于回归问题，则为回归树。

对于回归问题，其响应变量y为连续变量，故可使用“最小化残差平方和”作为节点的分裂准则。

在进行节点分裂时，希望分裂后，残差平方和下降最多，即两个子节点的残差平方和之总和最小。

为避免过拟合，对于回归树，也要使用惩罚项进行修枝，即最小化如下目标函数：

$$\min_{T} \underbrace{\sum_{m=1}^{|T|} \sum_{\mathbf{x}_i \in R_m} (y_i - \hat{y}_{R_m})^2}_{cost} + \lambda \cdot \underbrace{|T|}_{complexity} \quad (11.9)$$

其中，R_m为第m个终节点，而$\hat{y}_{R_m}$为该终节点的预测值(此终节点的样本均值)。$\sum_{\mathbf{x}_i \in R_m} (y_i - \hat{y}_{R_m})^2$为第 m 个终节点的残差平方和。

11.7 变量重要性

作为一种非参数方法，决策树并不包含回归系数。对于决策树模型，应如何度量不同变量的重要性？

由于在每次节点分裂，仅使用一个变量，故容易区分每个变量的贡献。比如，考察该分裂变量使得残差平方和(或基尼指数)下降多少。

对于每个变量，可计算在决策树的所有节点，由于此变量所导致的分裂准则函数(残差平方和或基尼指数)的总下降幅度，并以此作为**变量重要性**(Variable Importance)的度量。

更直观地，可将每个特征变量的重要性依次排序画图，即为“变量重要性图”(Variable Importance Plot)。

11.8 C5.0 算法

C5.0 算法起源于 Quinlan (1979)所提出的 ID3 算法(Iterative Dichotomiser 3)，后来演变为 C4.5 与 C5.0 算法 (C 表示 Classifier)。

与 CART 算法不同，C5.0 算法允许使用多叉树，而不局限于二叉树。

最初的 ID3 算法仅接受离散的特征变量。

因此，在某个节点进行分裂时，选中的分裂变量(split variable)分为几类，则决策树在此节点就开几叉。后来，此算法也推广到连续型的特征变量。

对于分类问题，C5.0 算法传统上使用信息熵作为分裂准则，而 CART 算法则一般使用基尼指数作为分裂准则。

在实践中，C5.0 算法的预测效果与 CART 类似。

由于 Breiman (2001)提出的随机森林算法，以及 Friedman(2001)提出的梯度提升法均基于 CART 算法，故本书主要介绍 CART 算法。

11.9 决策树的优缺点

决策树与 KNN 的共同点为，二者都采取“分而治之”(divide and conquer)的策略，将特征空间分割为若干区域，利用近邻进行预测。

KNN 在分区域时，不考虑响应变量y的信息，故在高维空间容易遇到维度灾难，且易受噪音变量的影响。

决策树在分区域时，考虑特征向量$\mathbf{x}$对y的影响，且每次仅使用一个分裂变量。这使得决策树很容易应用于高维空间，且不受噪音变量的影响。

如果特征向量$\mathbf{x}$包含噪音变量(对y无作用的变量)，也不会被选为分裂变量，故不影响决策树的建模。决策树的分区预测更具智慧，可视为**自适应近邻法**(adaptive nearest neighbor)。

决策树在进行递归分裂时，仅考虑“超矩形”(hyper-rectangle)区域。如果真实的决策边界与此相差较远或不规则，则可能导致较大误差。

幸运的是，基于决策树的集成学习(随机森林、提升法)，可得到比较光滑的决策边界，大幅提高预测准确率，参见第 12-13 章。

另外，如果真实模型或决策边界为线性，则线性模型的预测效果一般优于决策树。

反之，如果决策边界非线性，或接近于矩形，则决策树的预测效果可能更好。

11.10 回归树的 Python 案例

使用波士顿房价数据 `boston`，演示回归树的 Python 操作。

该数据集包含 1970 年波士顿 506 个社区有关房价的 14 个变量。

响应变量为社区房价中位数 `MEDV`(详见第 4 章)。

* 详见教材，以及配套 Python 程序（现场演示）。

11.11 分类树的 Python 案例

使用一个葡萄牙银行市场营销(bank marketing)的数据集(Moro et al., 2014)，来演示分类树的 Python 操作。

响应变量 y(取值为 yes 或 no)，表示在接到银行的直销电话后，客户是否会购买“银行定期存款”(bank term deposit)产品。

特征变量包括客户的以下特征：

(1)个人特征，比如年龄(age)、职业类型(job)、婚姻状况(marital)、教育程度(education)；

(2)经济状况，比如是否有信用违约(`default`)、是否有房贷(`housing`)、是否有个人贷款(`loan`)、工作单位人数(`nr.employed`)、消费者信心指数(`cons.conf.indx`)；

(3)营销状态，比如自上次联络以来的天数(`pdays`)、自上次去电后过了多少秒(`duration`)。

* 详见教材，以及配套 Python 程序（现场演示）。