机器学习方法在车险定价中的应用

程建华

2021 年 11-12 月

目录

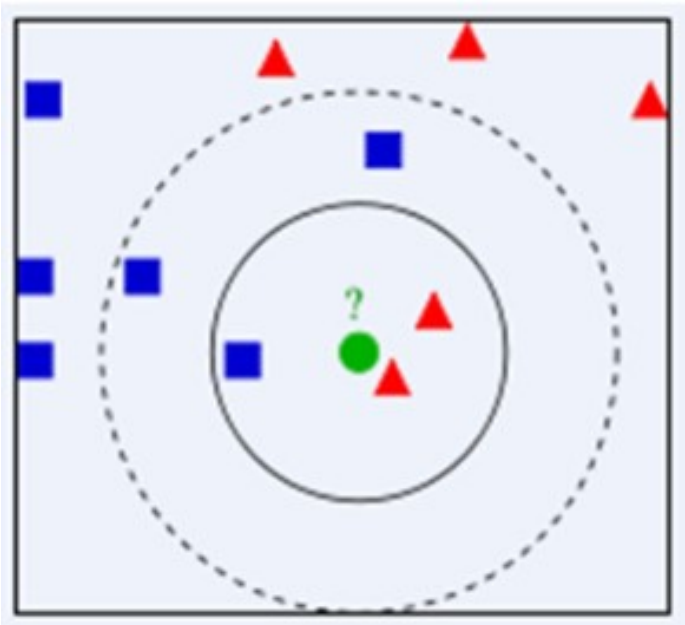
1. K 近邻法
2. 决策树

目录

1. K 近邻法
2. 决策树

3.1 K 近邻法简介

K 近邻法 (K Nearest Neighbors，简记 KNN) 由 Fix and Hodges (1951) 提出，是一个理论上比较成熟的非参数监督学习方法，也是最简单的机器学习算法之一. 如果一个样本在特征空间中的 K 个最相似（即特征空间中最近邻）的样本中的大多数属于某一个类别，则该样本也属于这个类别，即以最近的 K 个邻居进行预测.



3.2 K 近邻法三要素

K 近邻法使用的模型实际上对应于特征空间的划分.

- K 值的选择;
- 距离度量;
- 分类决策规则.

3.3 K 值的选择

一个极端，如果 $K = 1$(最近邻法)，则偏差较小 (仅使用最近邻的信息)，但方差较大 (未将近邻的 y 观测值进行平均)。所估计的回归函数很不规则，导致“过拟合”(overfit)，使得模型的泛化能力 (generalization) 较差.

另一极端，如果 K 很大，则偏差较大 (用到更远邻居的信息)，而方差较小 (用更多观测值进行平均). 在极端情况下，如果 $K = n$，则使用样本均值进行所有预测，导致偏差很大，出现“欠拟合”(underfit). 如果 K 太大，则决策边界将过于光滑，无法充分捕捉数据中的信号，使得算法的泛化能力下降，导致欠拟合.

在应用中，K 值一般取一个比较小的数值，通常采用交叉验证法来选取最优的 K 值.

3.4 距离度量

特征空间中两个实例点的距离是它们相似程度的反映. 设

$$\boldsymbol{X}_1 = (X_{11}, X_{12}, \cdots, X_{1q})^{\mathrm{T}}, \quad \boldsymbol{X}_2 = (X_{21}, X_{22}, \cdots, X_{2q})^{\mathrm{T}},$$

是两个实例的特征向量，则其 L_p（闵可夫斯基，Minkowski）距离定义为

$$L_p(\boldsymbol{X}_1, \boldsymbol{X}_2) = \left(\sum_{l=1}^{q} |X_{1l} - X_{2l}|^p \right)^{\frac{1}{p}},$$

这里 $p \geq 1$. 当 $p = 2$ 时，称为欧氏距离（Euclidean），即

$$L_2(\boldsymbol{X}_1, \boldsymbol{X}_2) = \left(\sum_{l=1}^{q} |X_{1l} - X_{2l}|^2 \right)^{\frac{1}{2}}.$$

3.4 距离度量

当 $p=1$ 时，称为曼哈顿距离（Manhattan），即

$$L_1(\boldsymbol{X}_1,\boldsymbol{X}_2)=\sum_{l=1}^{q}|X_{1l}-X_{2l}|\,.$$

当 $p=\infty$ 时，称为切比雪夫距离（Chebyshev），即

$$L_1(\boldsymbol{X}_1,\boldsymbol{X}_2)=\max_{1\le l\le q}|X_{1l}-X_{2l}|\,.$$

注 1：其他距离：皮尔逊相关系数（Pearson Correlation），余弦相似度（Cosine Similarity），相异度（binary）等.

注 2：使用 KNN 法的一个前提是，所有特征变量均为数值型；否则，将无法计算欧氏距离.

注 3：为避免某些变量对于距离函数的影响太大，一般先将所有变量标准化，即减去该变量的均值，再除以其标准差，使得所有变量的均值为 0 而标准差为 1.

3.5 分类决策规则

KNN 方法既可以做分类，也可以做回归.

- KNN 做分类预测时，一般是选择多数表决法，即训练集里和预测的样本特征最近的 K 个样本，预测为里面有最多类别数的类别；
- KNN 做回归时，一般是选择平均法，即最近的 K 个样本的样本输出的平均值作为回归预测值.

3.6 K 近邻法的优缺点

KNN 的优点之一是算法简单易懂.

作为非参数估计，不依赖于具体的函数形式，故较为稳健，适合多分类问题.

但 KNN 所估计的回归函数或决策边界一般较不规则，因为当 $\boldsymbol{X}$ 在特征空间变动时，其 K 个邻居之集合可能发生不连续的变化，即加入新邻居而去掉旧邻居. 如果真实的回归函数或决策边界也较不规则，则 KNN 效果较好.

KNN 是一种“懒惰学习”(lazy learning). KNN 算法平时不学习，属于“死记硬背型”(memory-based)，并没有估计真正意义上的模型. 直到预测时才去找邻居，导致预测较慢，不适用于“在线学习”(online learning)，即需要实时预测的场景.

在高维空间中可能很难找到邻居，会遇到所谓“维度灾难”(curse of dimensionality)，此时 KNN 算法的效果可能较差. KNN 算法一般要求 $n \gg q$，即样本容量 n 须远大于特征向量的维度 q.

KNN 对于“噪音变量”(noise variable) 也不稳健. 如果特征向量包含对响应变量毫无作用的噪音变量，KNN 在计算观测值之间的欧氏距离时，也依然不加区别地对待这些变量，导致估计效率下降.

目录

1 K 近邻法

2 决策树

4.1 决策树简介

- 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；
- 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).

4.2 分类树的启发案例

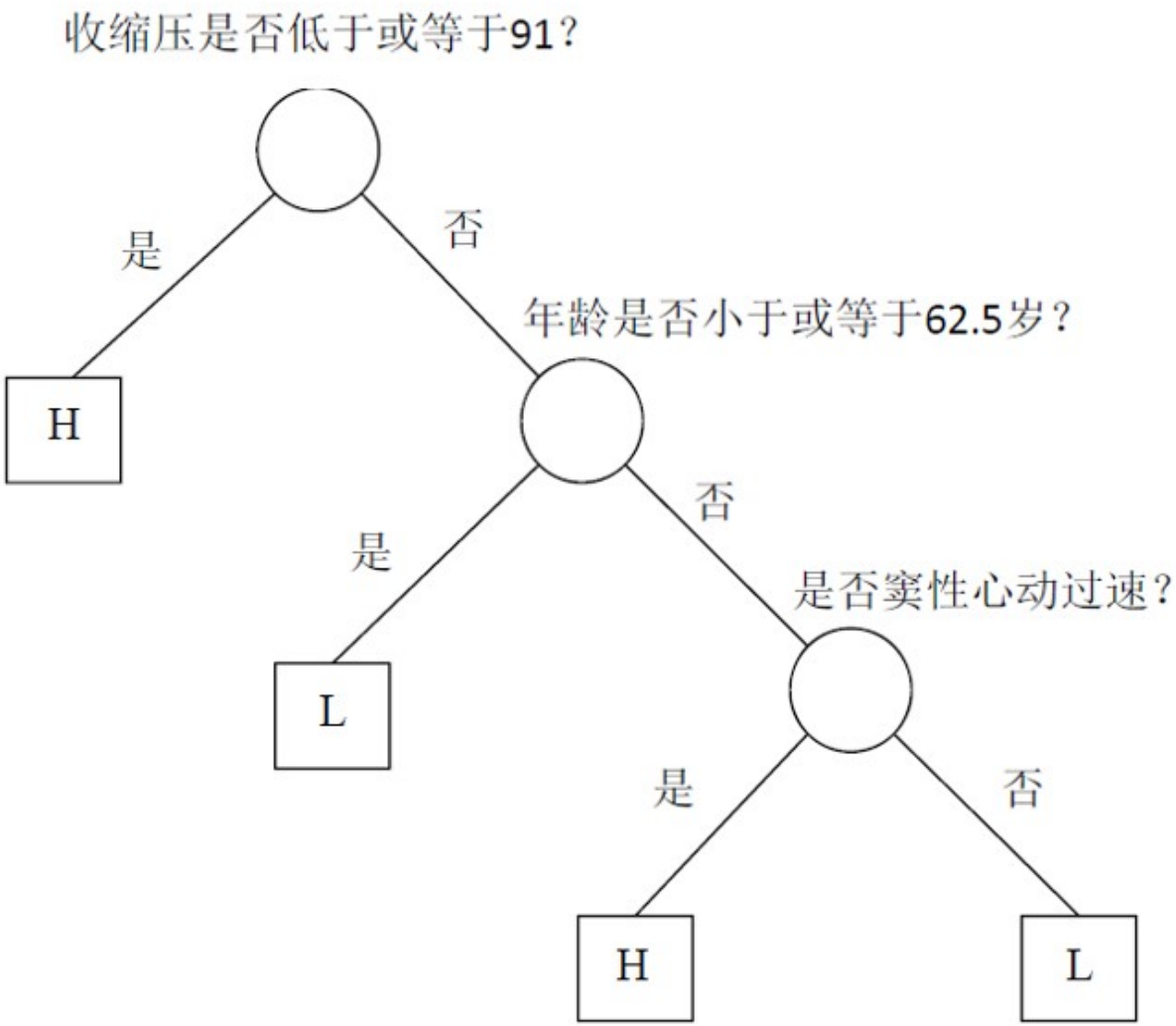
Breiman et al. (1984) 研究了 UCSD 医学中心的一个案例.

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：

4.2 分类树的启发案例



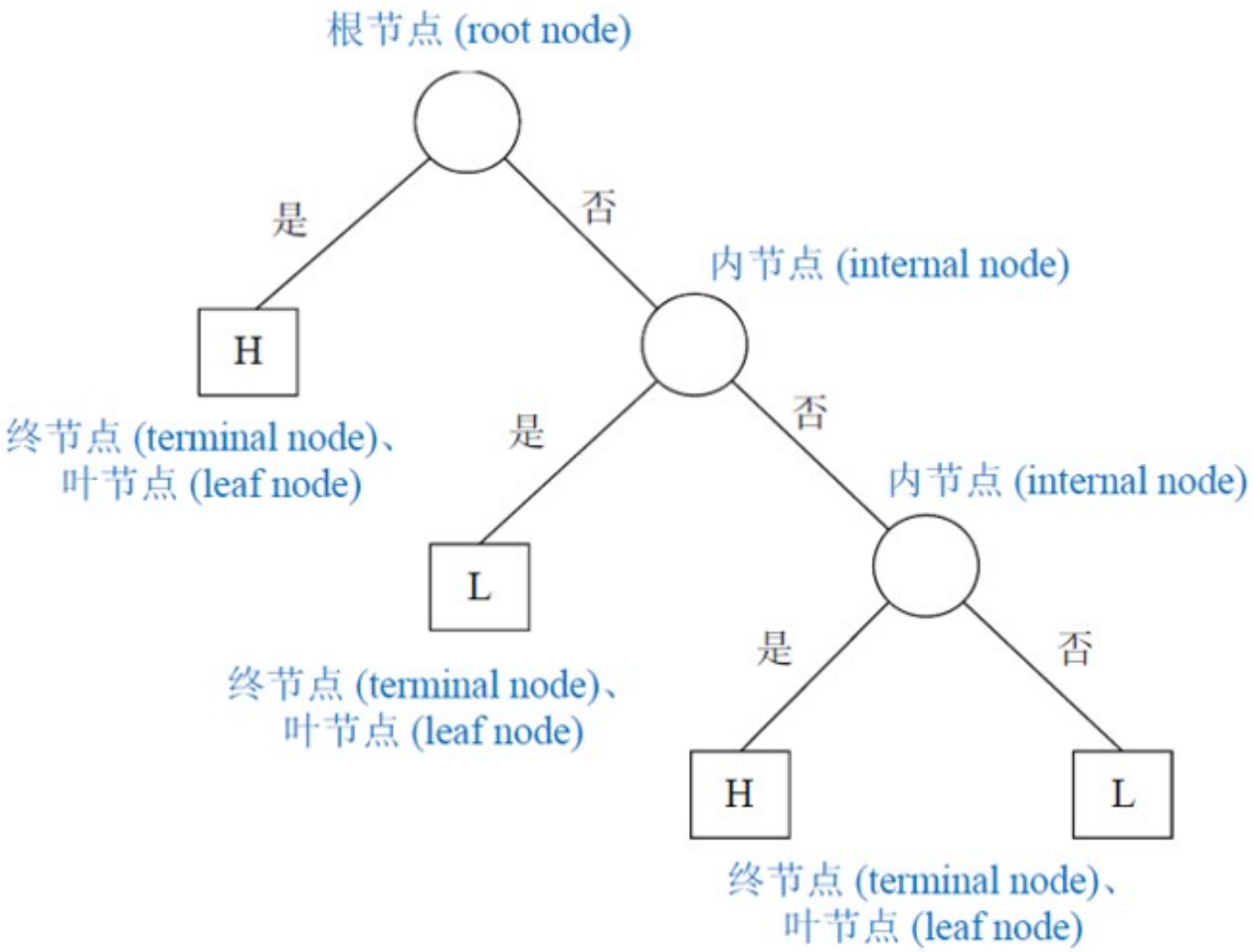
4.3 决策树的优点

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！在以上案例中，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.

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) 或者平均值进行预测，即看落入该叶节点的训练数据最多为哪类.

4.4 根节点、内节点和叶节点



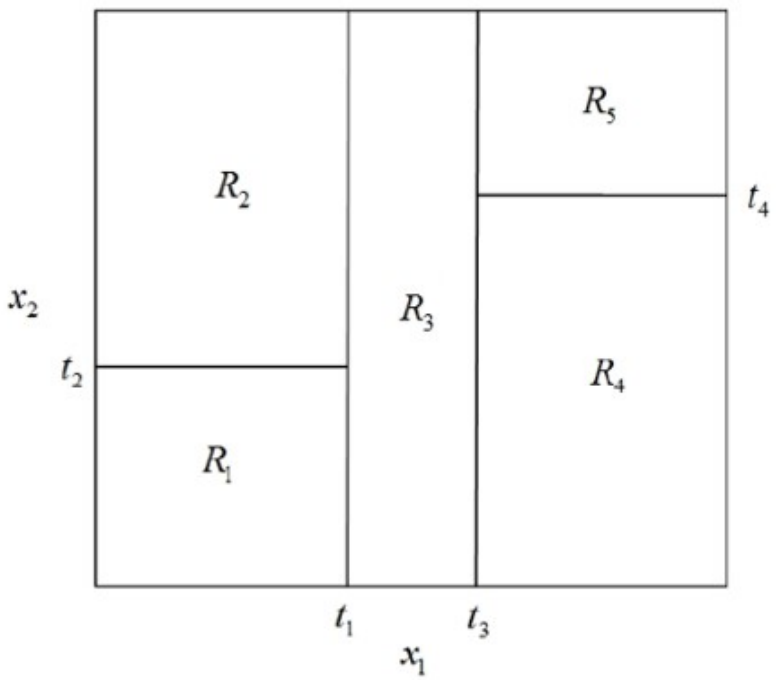
4.5 递归分割

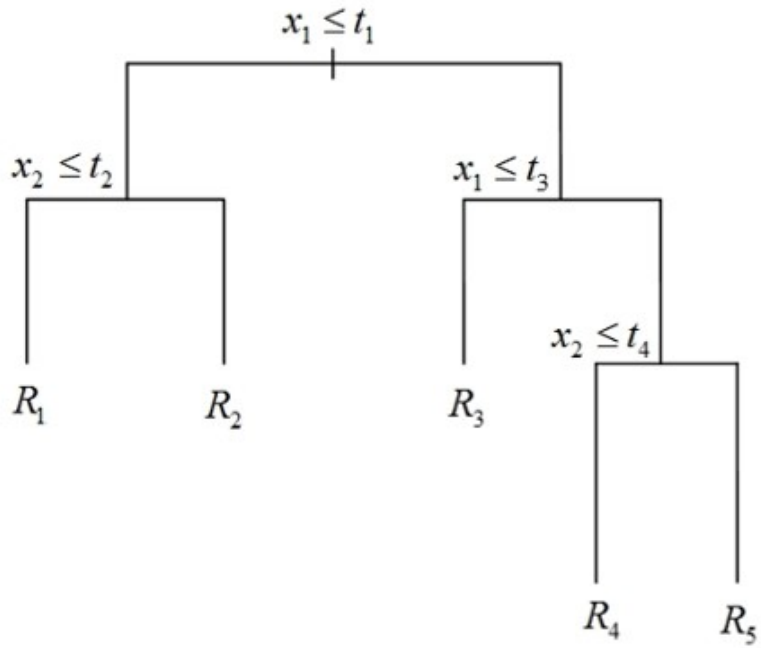
CART 算法所用的决策树为二叉树 (binary tree)，即每次总是将“母节点”(parent node) 一分为二，分裂 (split) 为两个“子节点”(child node)，直至到达终节点.

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).

分类树是通过分割特征空间进行分类的分类器 (classifier as partition). 从数学角度看，决策树为“分段常值函数”(piecewise constant function).

4.5 递归分割


R_5
R_2
t_4
x_2
R_3
R_4
t_2
R_1
t_1
t_3
x_1


$x_1 \le t_1$
$x_2 \le t_2$
$x_1 \le t_3$
$x_2 \le t_4$
R_1
R_2
R_3
R_4
R_5

4.6 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.

4.7 节点不纯度函数（Node Impurity Function）

假设响应变量 Y 共分 K 类，取值为 $Y \in \{1, 2, \cdots, K\}$.

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)，首先定义一个“节点不纯度函数”

$$\varphi(p_1, \cdots, p_K) \geq 0.$$

. 该函数应具备以下性质：

- 当 $p_1 = \cdots = p_K = \dfrac{1}{K}$，不纯度最高，即 $\varphi(p_1, \cdots, p_K)$ 达到最大值；
- 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；
- 关于 $\varphi(p_1, \cdots, p_K)$ 关于自变量 $(p_1, \cdots, p_K)$ 是对称的.

注：满足这些性质的函数并不唯一.

4.7 节点不纯度函数：错误率

错误率也称 (error rate) 错分率 (misclassification rate)，其定义为：

$$\mathrm{Err}(p_1, \cdots, p_K) = 1 - \max\{p_1, \cdots, p_K\},$$

不难验证，错分率满足以上三条性质. 在二分类问题中，错分率简化为：

$$\mathrm{Err}(p_1, p_2) = 1 - \max\{p_1, p_2\} = \begin{cases} p_1, & \text{如果 } 0 \le p_1 < 0.5, \\ p_2, & \text{如果 } 0.5 \le p_1 \le 1. \end{cases}$$

4.7 节点不纯度函数：基尼指数

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，其定义为：

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

其中 $\sum_{k=1}^{K} p_k^2$ 可视为随机抽样的两个观测值之类别一致的概率. 在二分类问题中，基尼指数简化为：

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

其中 $p_1(1 - p_1)$ 可视为两点分布 (Bernoulli distribution) 的方差. 基尼指数为抛物线，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.

4.7 节点不纯度函数：信息熵

信息熵 (information entropy) 起源于信息理论 (information theory). 一个随机事件的发生，其信息量被认为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”：

$$\text{Entropy}(p_1, \cdots, p_K) = -\sum_{k=1}^{K} p_k \log_2 p_k,$$

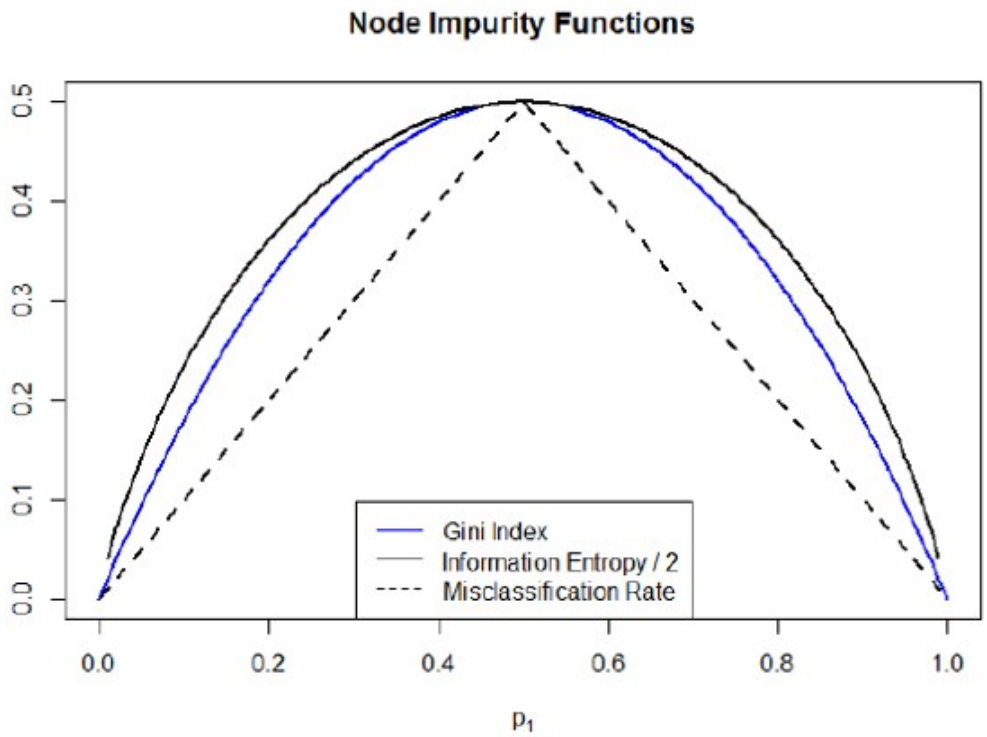
其中定义 $0 \cdot \log_2(0) = 0$. 对于二分类问题，信息熵可写为

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

4.7 节点不纯度函数：信息熵

注 1：定义中也可以选取以 e 为底的自然对数.

注 2：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（将信息熵函数标准化，即除以 2）. 在实践中，使用基尼指数或信息熵的预测效果一般很接近.

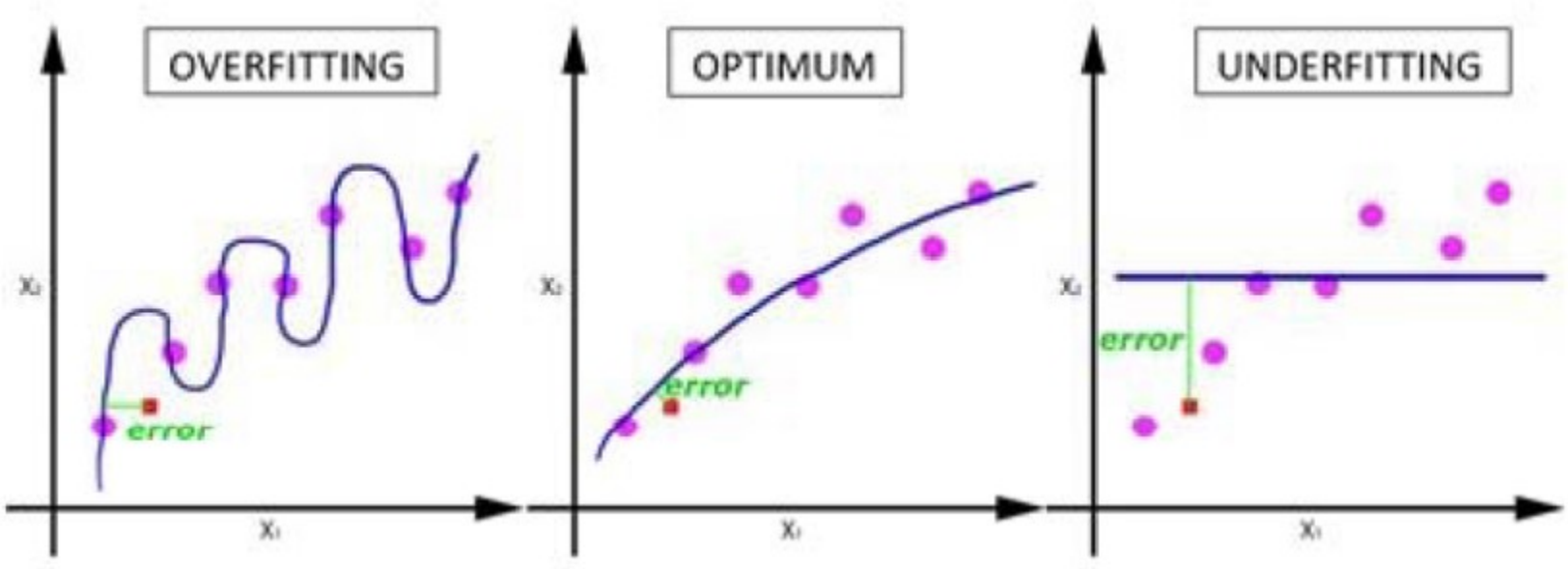


4.7 节点不纯度函数

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 $X_i \leq t$，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.

4.8 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.



4.8 成本复杂性修枝

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 |T|.

为避免过拟合，不希望决策树过于复杂，故惩罚其规模 |T|：

$$\min_T R(T) + \lambda \cdot |T|,$$

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“复杂性参数”(complexity parameter，简记 CP)，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.

4.8 成本复杂性修枝

对于分类问题，将 0-1 损失函数代入上述目标函数可得

$$\min_{T} \sum_{m=1}^{|T|} \sum_{\boldsymbol{X}_i \in R_m} I(Y_i \neq \hat{Y}_{R_m}) + \lambda \cdot |T|,$$

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)，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，此时一定选择最大的树 $T_{\max}$(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.

4.9 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. 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.

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.

4.10 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 $\boldsymbol{X}$ 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 $\boldsymbol{X}$ 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.