MACHINE LEARNING

机器学习

Decision Trees

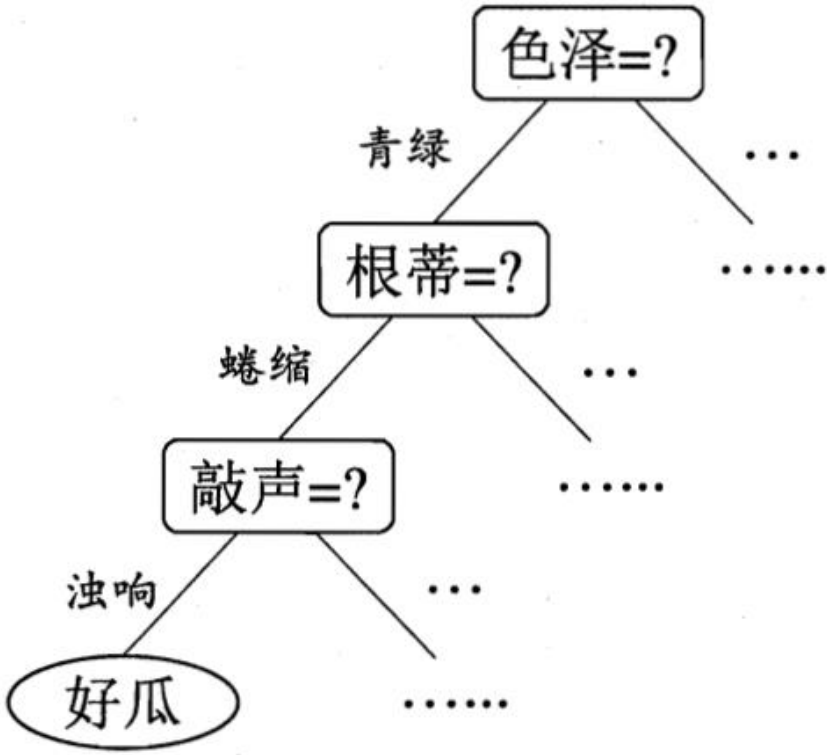
决策树

参考：
《机器学习》

Machine Learning Course

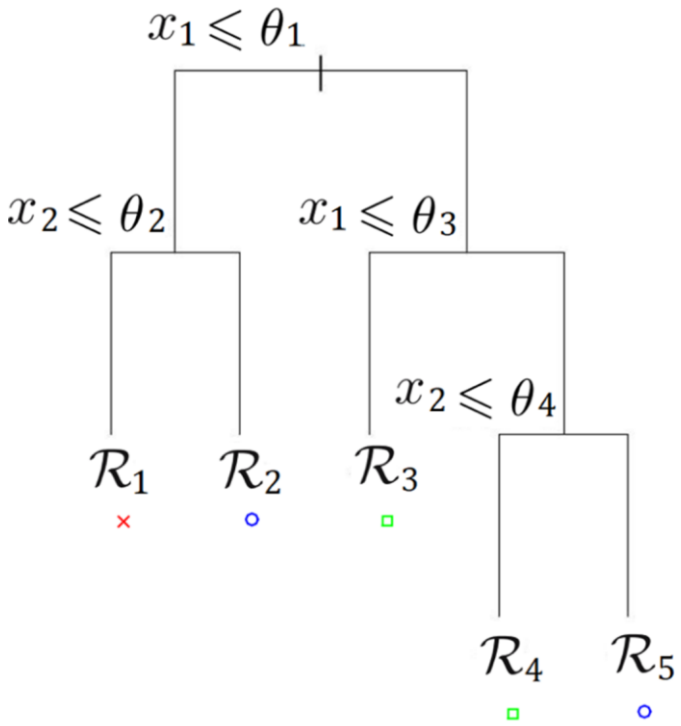
定义

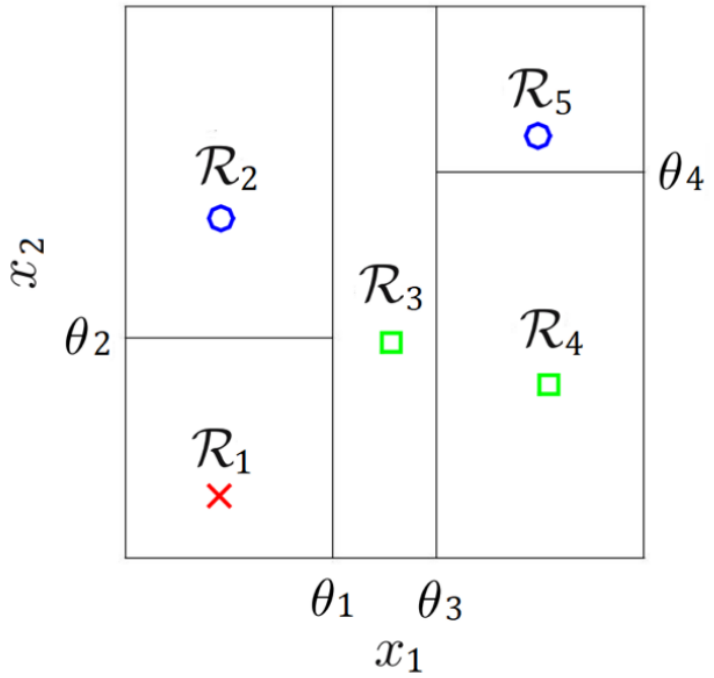
- 树结构的决策函数



定义

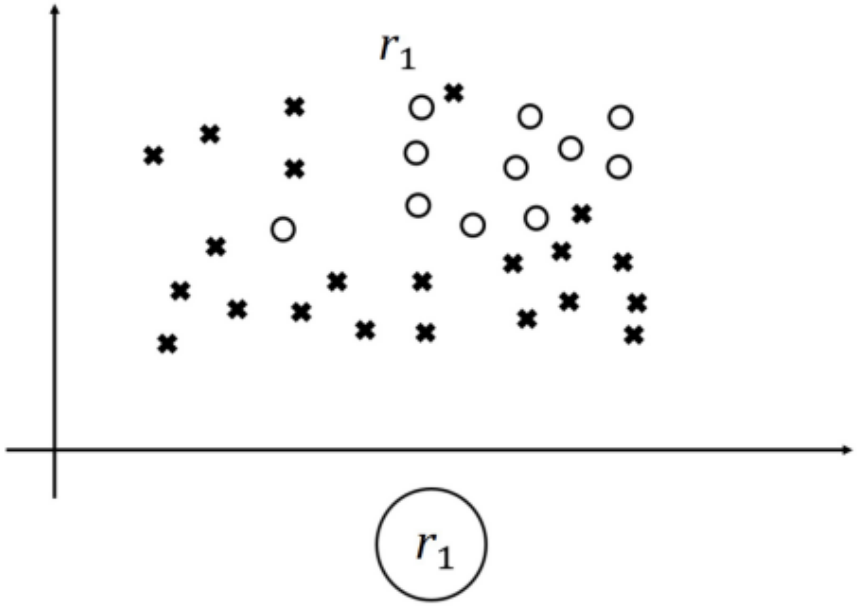
- 决策树的决策空间


$x_1 \leqslant \theta_1$
$x_2 \leqslant \theta_2$
$x_1 \leqslant \theta_3$
$x_2 \leqslant \theta_4$
$\mathcal{R}_1$
$\mathcal{R}_2$
$\mathcal{R}_3$
$\mathcal{R}_4$
$\mathcal{R}_5$


$\mathcal{R}_1$
$\mathcal{R}_2$
$\mathcal{R}_3$
$\mathcal{R}_4$
$\mathcal{R}_5$
θ_1
θ_2
θ_3
θ_4
x_1
x_2

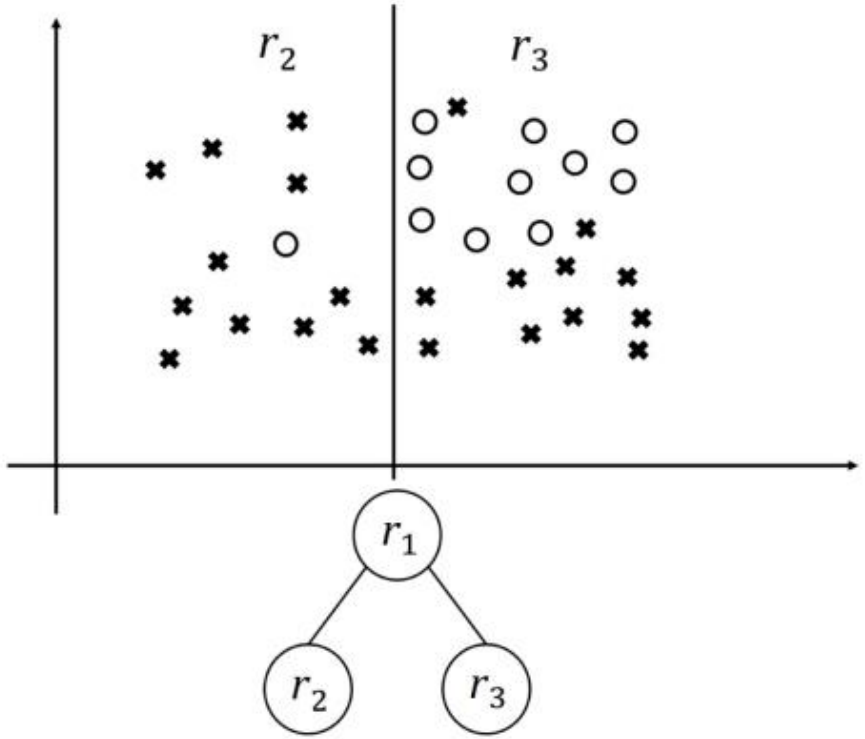
定义

- “分而治之”(divide - and - conquer)的构建策略



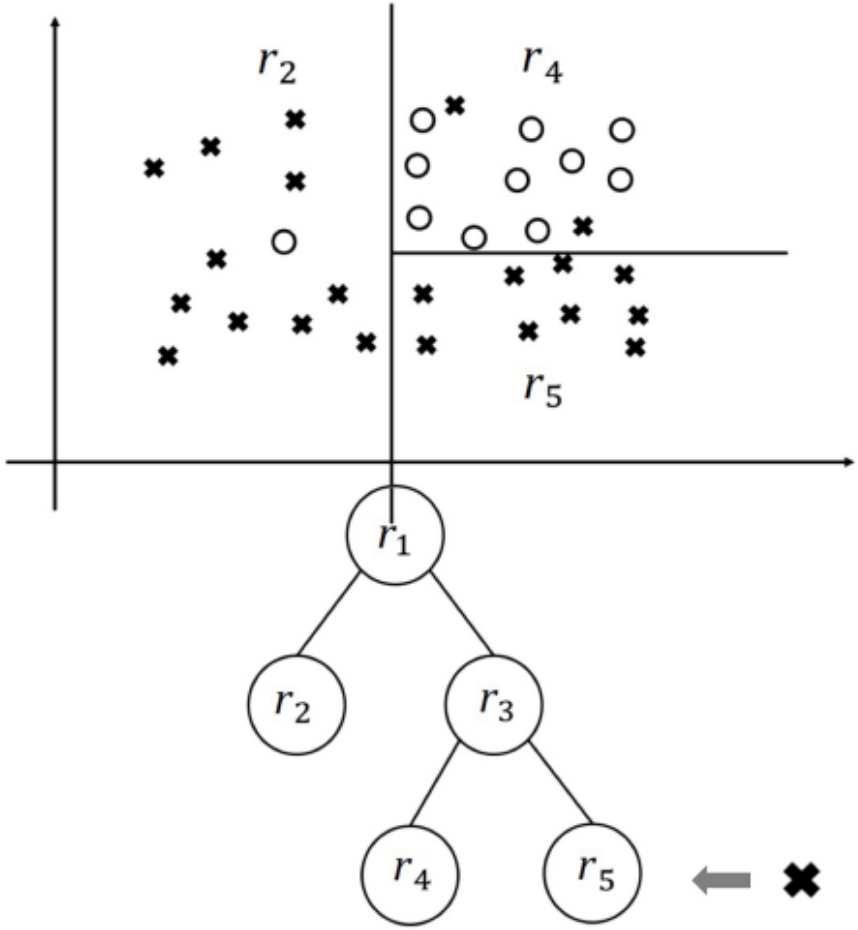
定义

- “分而治之”(divide - and - conquer)的构建策略



定义

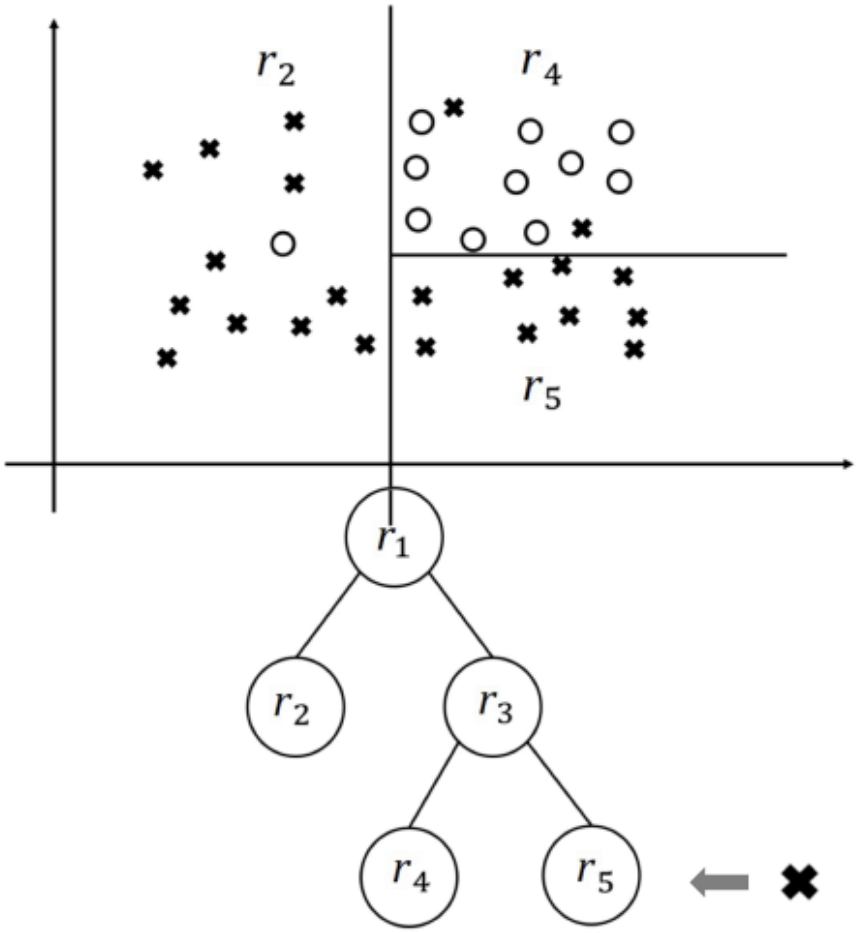
- “分而治之”(divide - and - conquer)的构建策略



学习算法

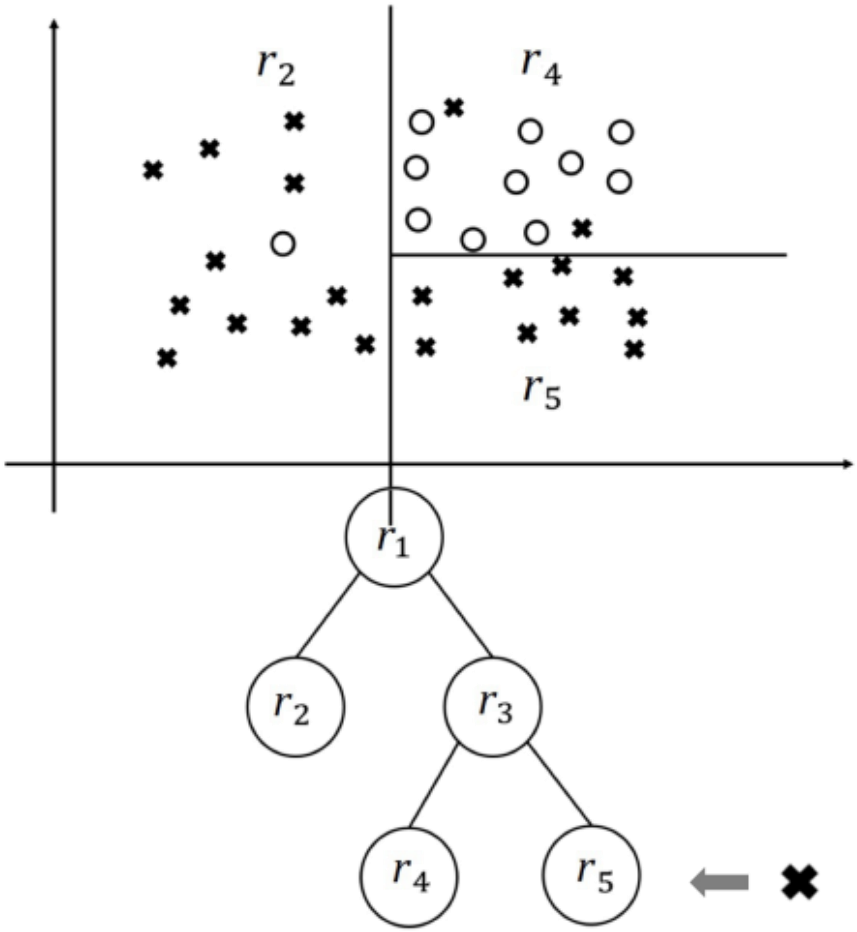
- “分而治之”(divide - and - conquer)的构建策略

- 递归地进行以下步骤：
  - 选择一个划分数据的特征作为当前节点
  - 将划分的分支刻画出来
  - 对划分的分支进行停止条件的检查：
    - 如果停止条件满足，则生成叶子节点
    - 否则继续选择特征继续划分



学习算法

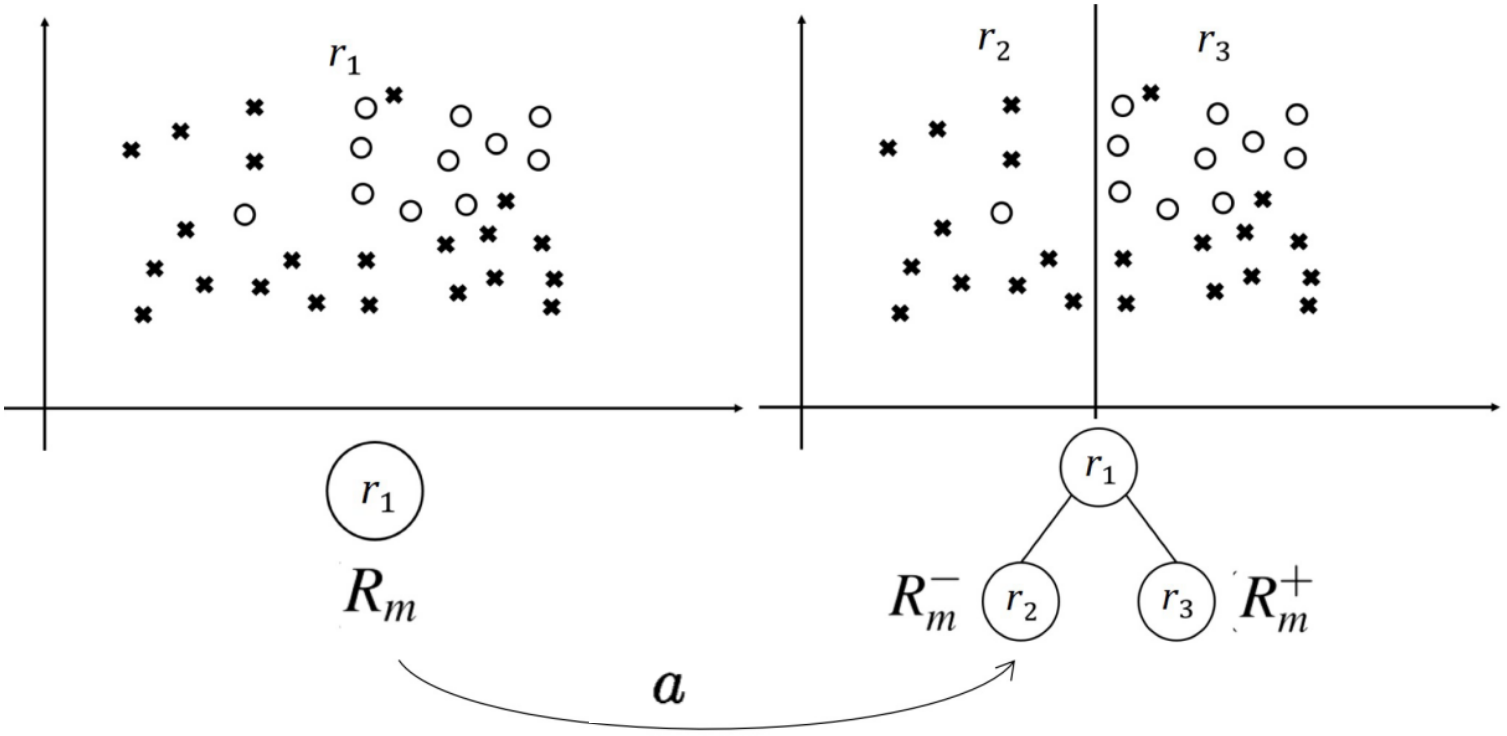
- 停止划分条件
  - 样本全属于同一类别
  - 样本的特征向量完全一样



学习算法

- “最优的划分”

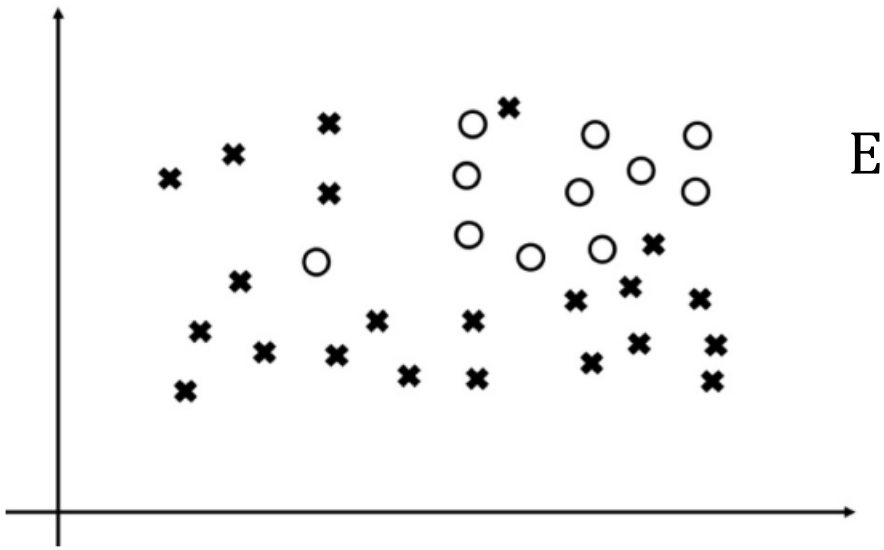
- 随着划分过程不断进行，我们希望决策树的分支结点所包含的样本尽可能属于同一类别，即结点的“纯度” (purity)越来越高.



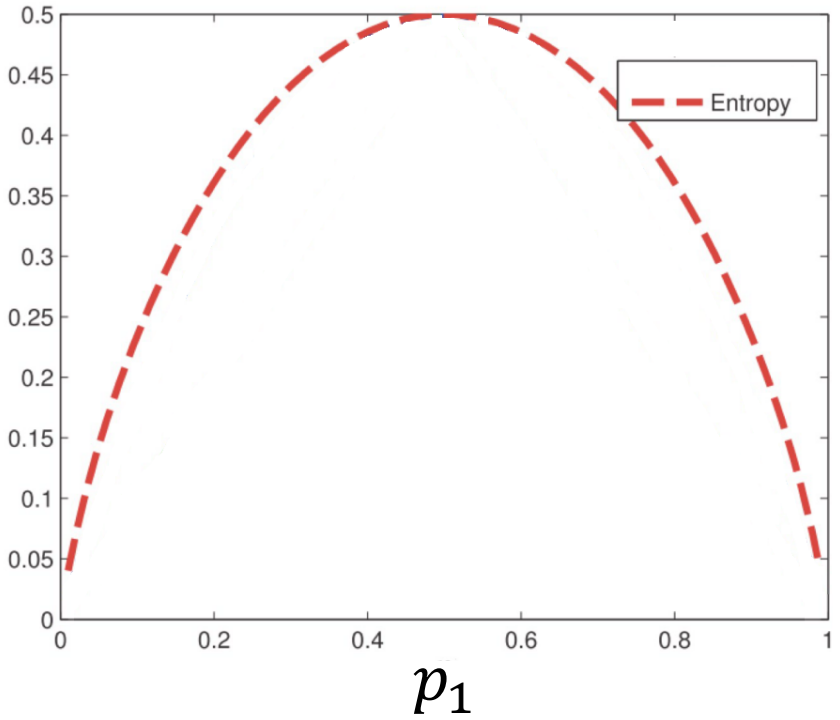
学习算法

- 信息熵

$$\mathrm{Ent}(D) = -\sum_{k=1}^{|\mathcal{Y}|} p_k \log_2 p_k$$



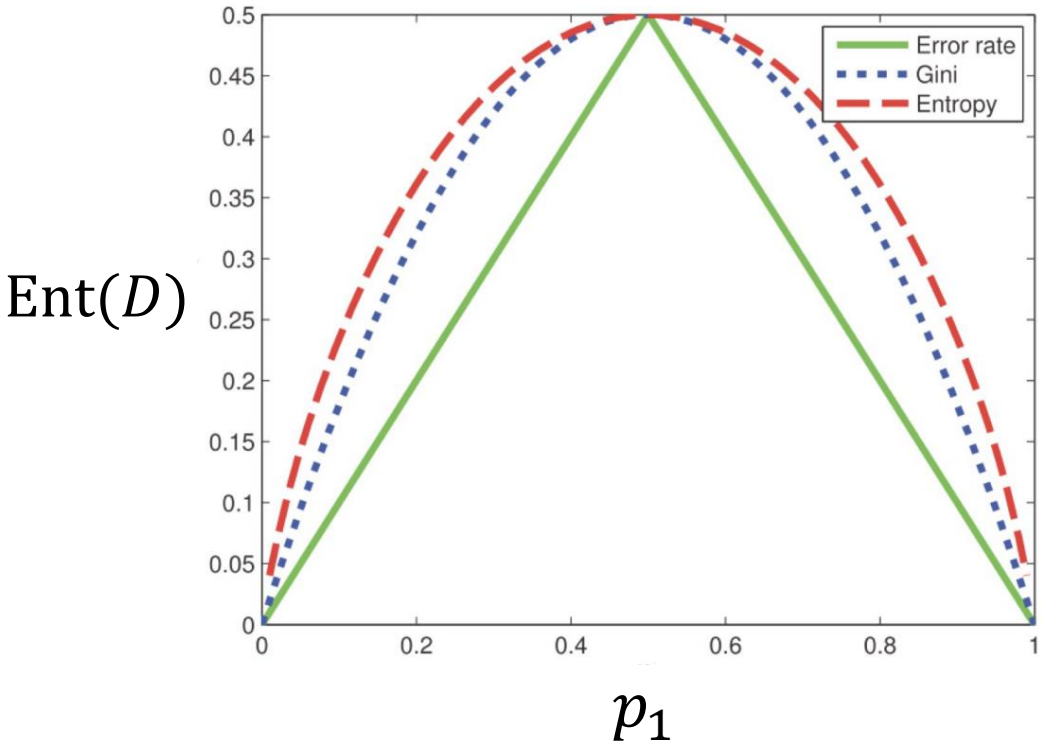
$\mathrm{Ent}(D)$



学习算法

- 基尼指数

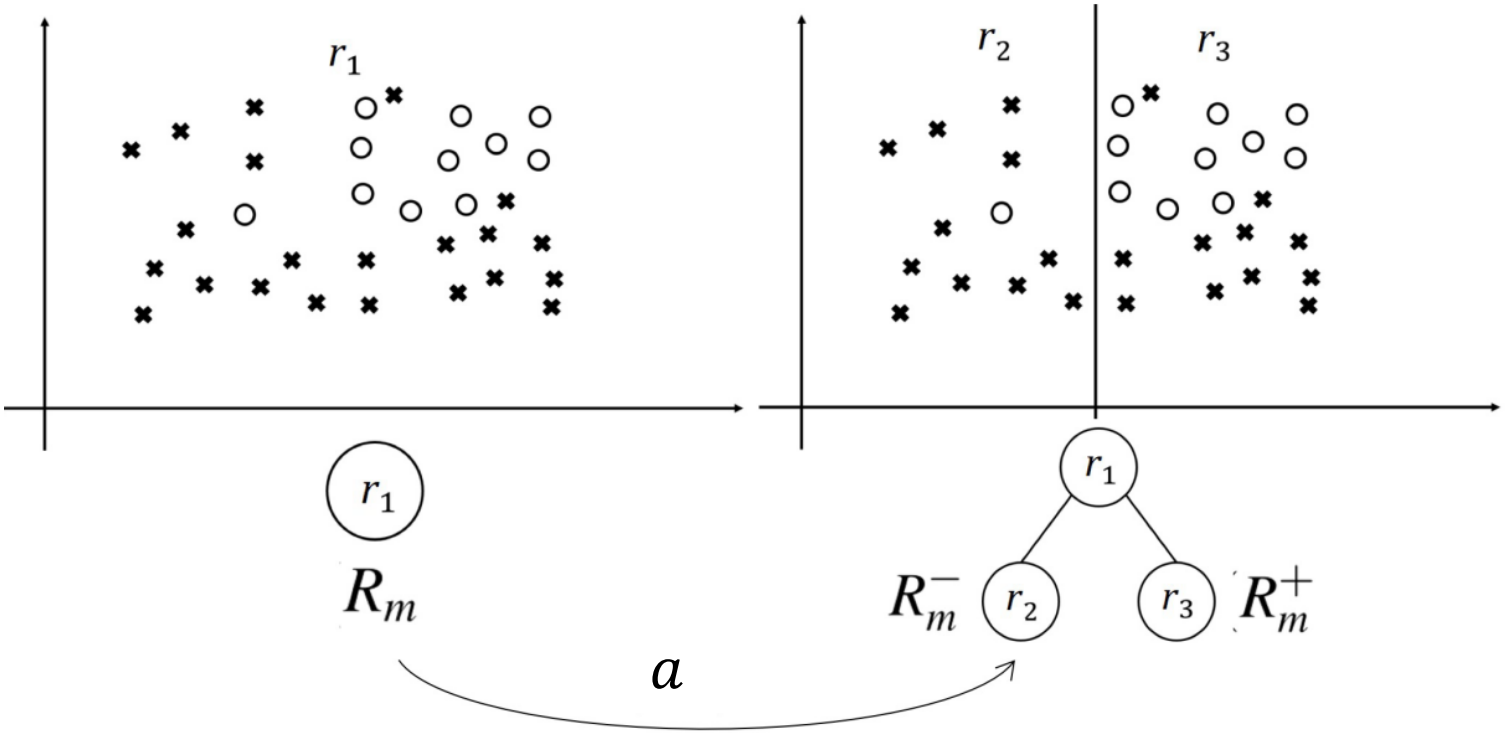
$$\text{Gini}(D) = \sum_{k=1}^{|\mathcal{Y}|} \sum_{k' \neq k} p_k p_{k'} = 1 - \sum_{k=1}^{|\mathcal{Y}|} p_k^2$$



学习算法

- 信息增益

$$\text{Gain}(D,a) = \text{Ent}(D) - \sum_{v=1}^{V} \frac{|D^v|}{|D|} \text{Ent}(D^v)$$



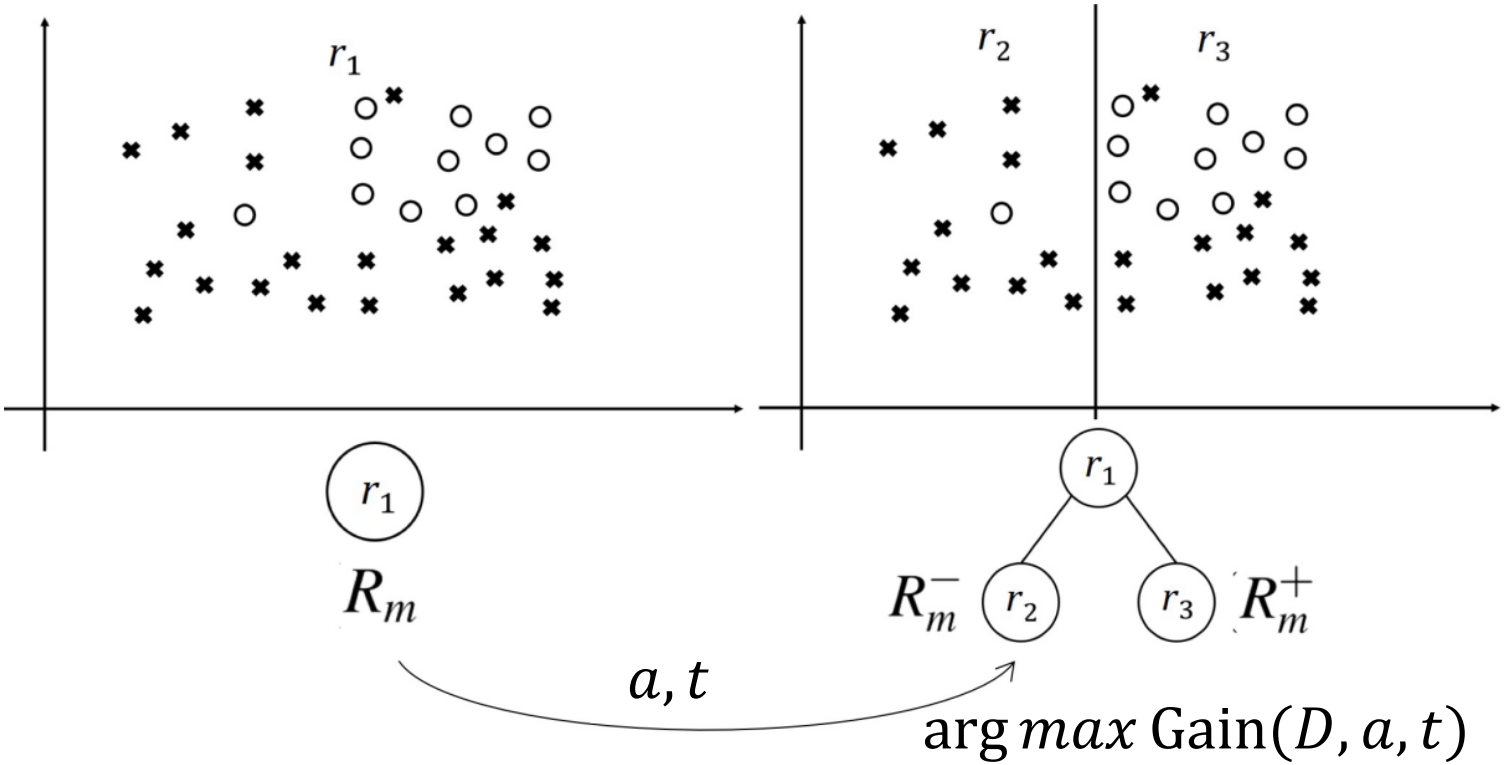
$$a_* = \arg\max_{a \in A} \text{Gain}(D,a)$$

学习算法

- 离散特征与连续特征

$$T_a = \{\frac{a^i + a^{i+1}}{2} \mid 1 \leqslant i \leqslant n-1\}$$

$$\text{Gain}(D,a) = \max_{t \in T_a} \text{Gain}(D,a,t) = \max_{t \in T_a} \text{Ent}(D) - \sum_{\lambda \in \{-,+\}} \frac{|D_t^\lambda|}{|D|} \text{Ent}(D_t^\lambda)$$



学习算法

- 含缺失值特征
- 让同一个样本以不同的概率划入到不同的子结点中去
- 其他

MACHINE LEARNING
机器学习
Decision Trees

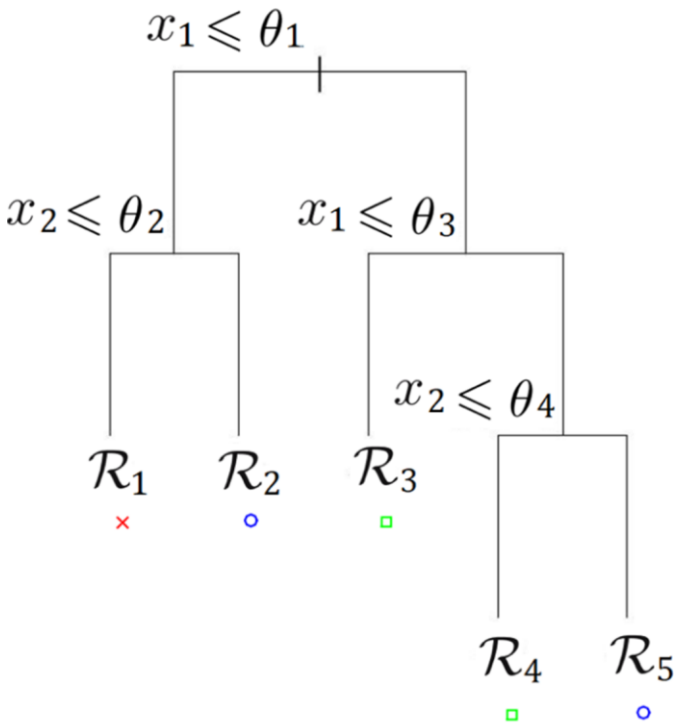
决策树扩展

参考：
《统计学习方法》

Machine Learning Course

分类与回归树 Classification and Regression Trees

- CART算法
  - 二叉树结构



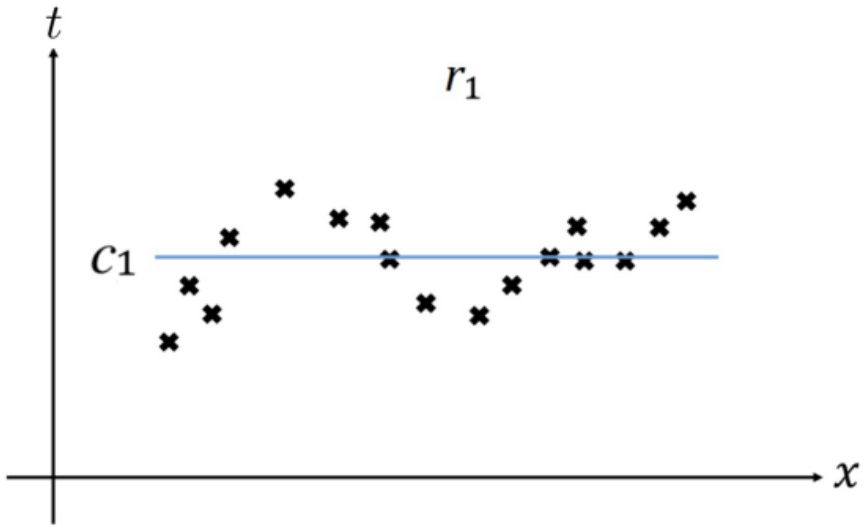
决策树的生成就是递归地构建二叉决策树的过程。对分类树用基尼指数 (Gini index) 最小化准则，进行特征选择，生成二叉树。

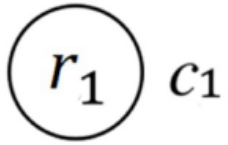
CART也包含了回归树的构建，对回归树其采用平方误差最小化准则。

分类与回归树 Classification and Regression Trees

- CART算法
- 回归树

$$y(x_n) = \sum_{m=1}^{M} c_m \cdot \mathbb{I}\{x_{n \in R_m}\}$$





c_1

分类与回归树 Classification and Regression Trees

- CART算法
  - 决策树的决策函数（分类树）

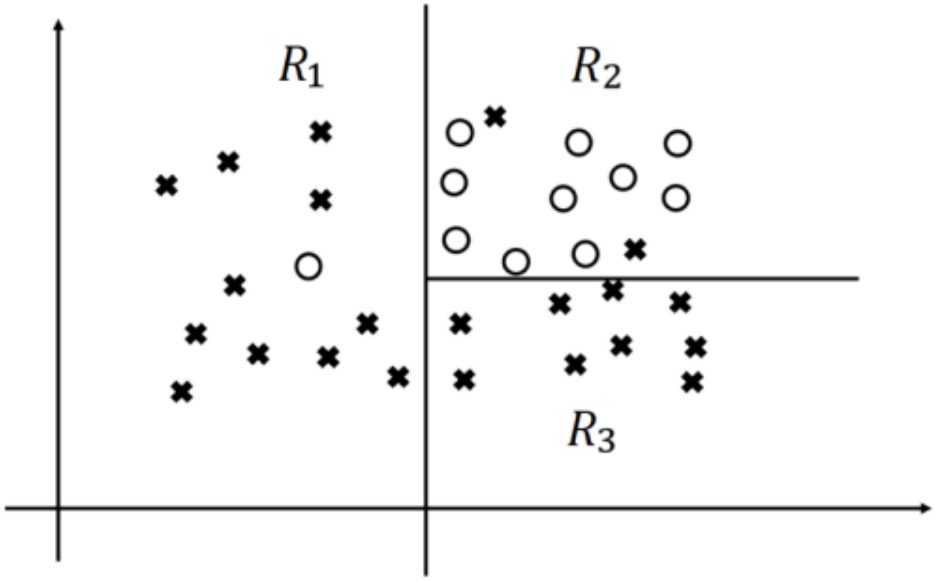
$$y(x_n) = \sum_{m=1}^{M} c_m \cdot \mathbb{I}\{x_{n \in R_m}\}$$

$$\widehat{p}(\mathcal{C}_k | x_n) = \sum_{m=1}^{M} p_k^m \cdot \mathbb{I}\{x_{n \in R_m}\}$$

$$\{R_1, \ldots, R_M\} \longrightarrow c_m \text{ or } p_k^m ?$$

$$c_m = \max_k \sum_{x_n \in R_m} \mathbb{I}\{t_n = k\}$$

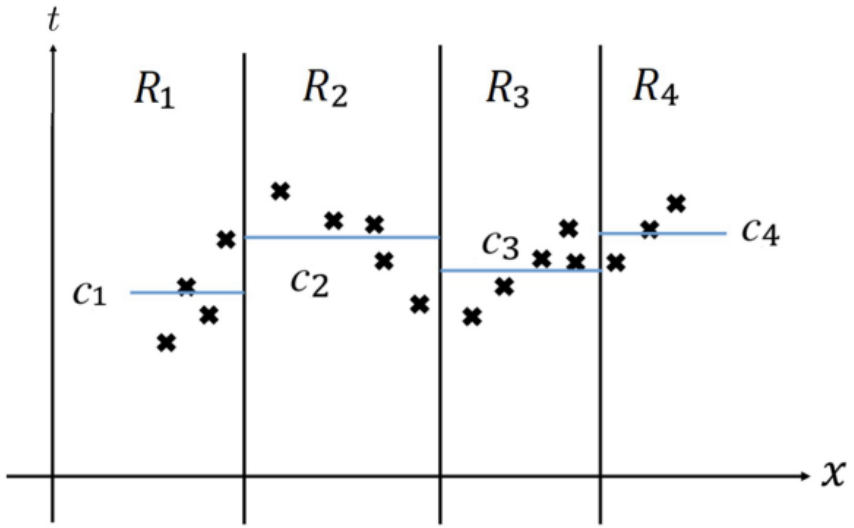
$$p_k^m = \frac{1}{|R_m|} \sum_{x_n \in R_m} \mathbb{I}\{t_n = k\}$$

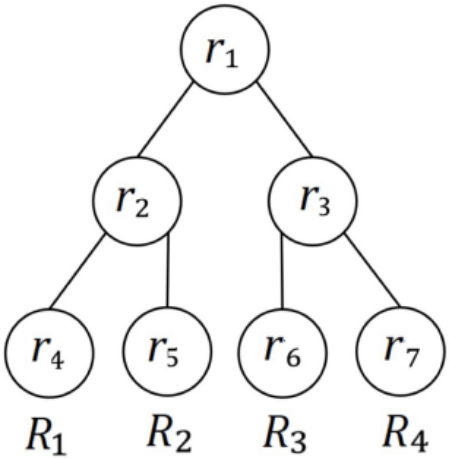


分类与回归树 Classification and Regression Trees

- CART算法
- 决策树的决策函数（回归树）

$$y(x_n) = \sum_{m=1}^{M} c_m \cdot \mathbb{I}\{x_{n \in R_m}\}$$





分类与回归树 Classification and Regression Trees

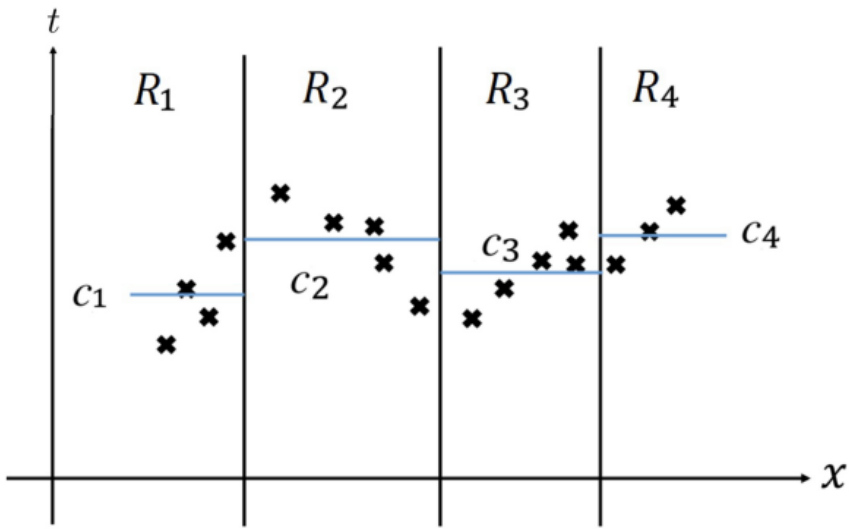
- CART算法
- 决策树的决策函数（回归树）

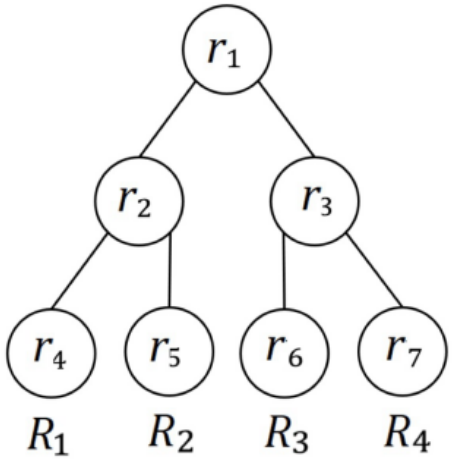
$$y(x_n) = \sum_{m=1}^{M} c_m \cdot \mathbb{I}\{x_{n \in R_m}\}$$

$$\{R_1, \ldots, R_M\} \longrightarrow c_m$$

$$c_m = \arg\min_{c_m} \sum_{x_n \in R_m} (t_n - c_m)^2$$

$$= \text{ave}(t_n | x_{n \in R_m})$$





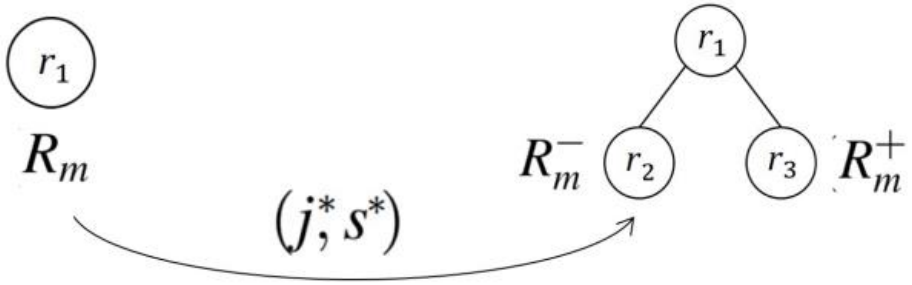
分类与回归树 Classification and Regression Trees

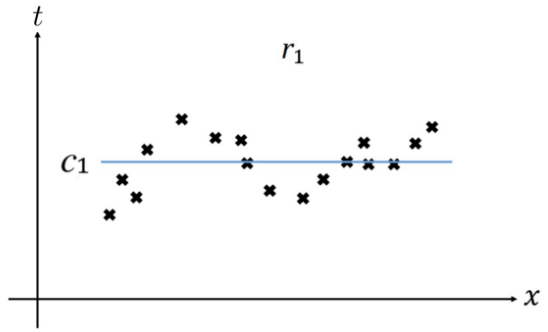
- CART算法
- 决策树的决策函数（回归树）

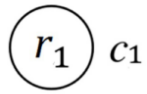
$$E(R_m) = \sum_{x_n \in R_m} (t_n - c_m)^2 = \sum_{x_n \in R_m} (t_n - \text{ave}(t_n \mid x_{n \in R_m}))^2$$

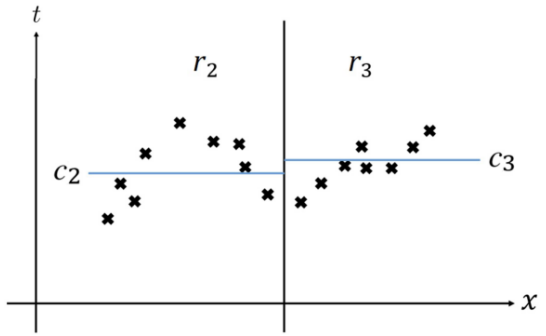
$$L(j,s) = \left(p_{R_m^-} \cdot E(R_m^-) + p_{R_m^+} \cdot E(R_m^+) \right) - E(R_m)$$

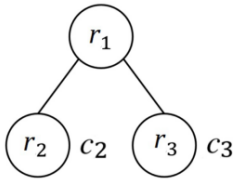
$$(j^*, s^*) = \arg\min_{(j,s)} L(j,s)$$







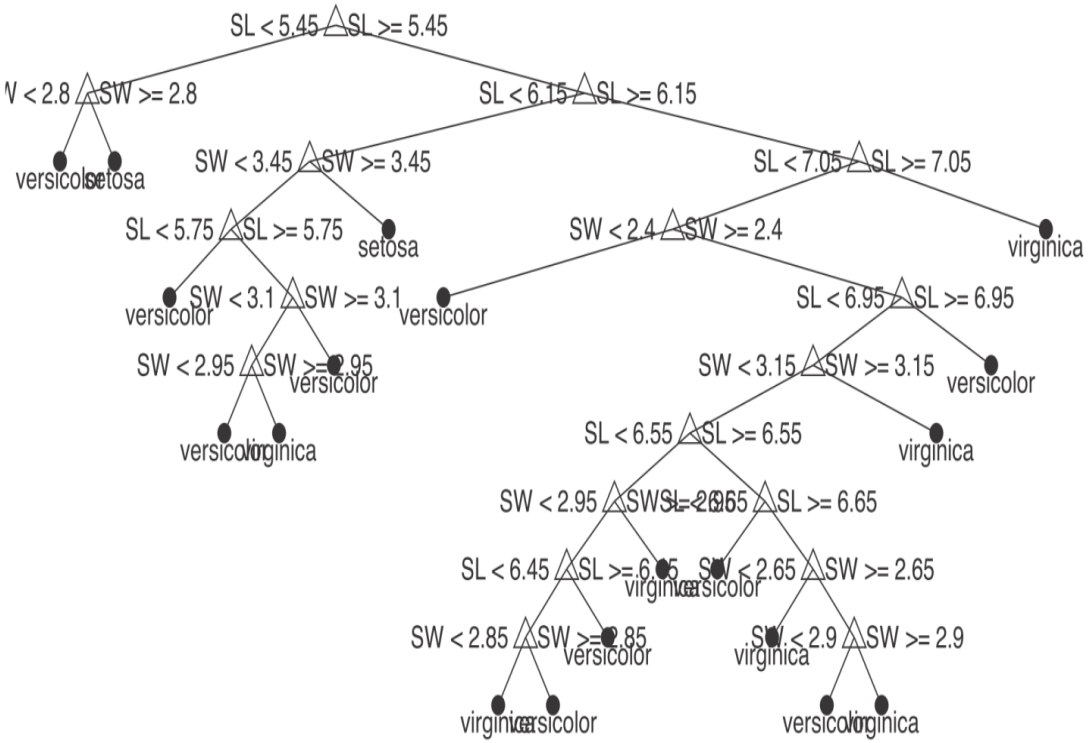


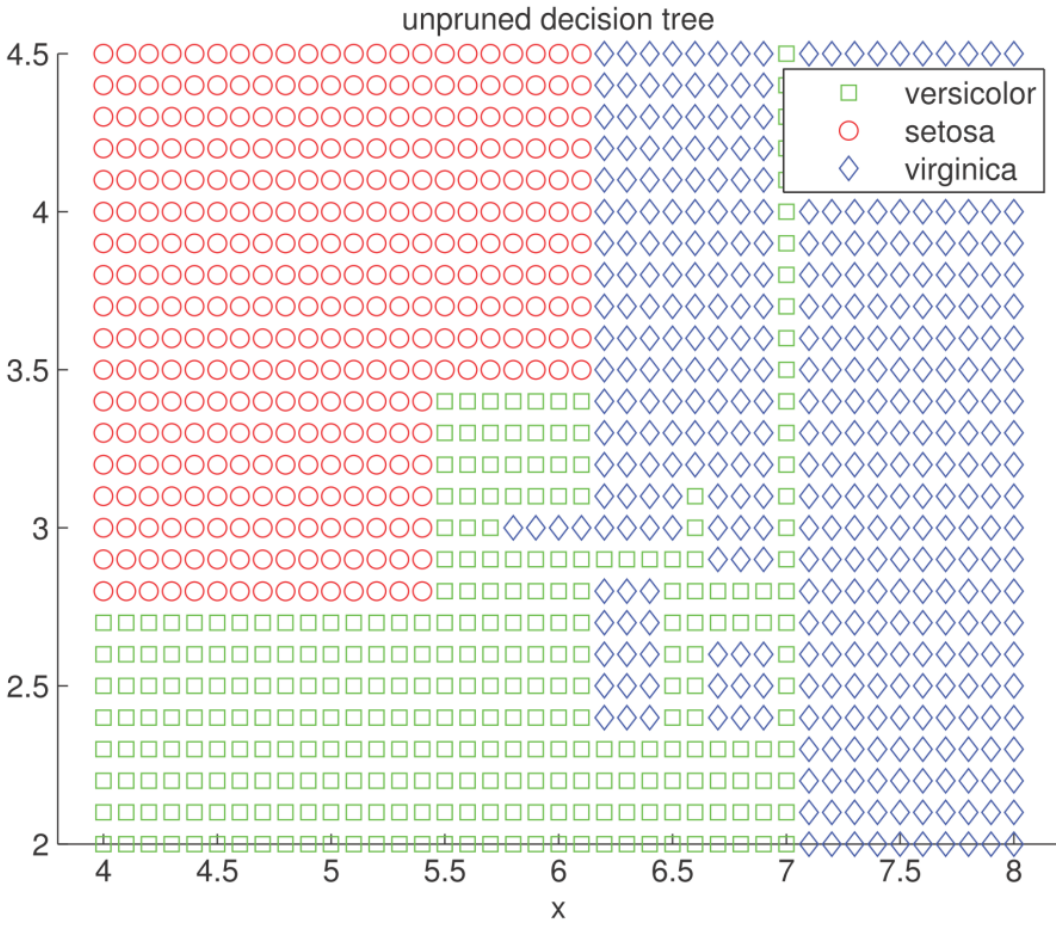


剪枝

- 决策树的生长与过拟合风险

过度生长可能会导致过拟合

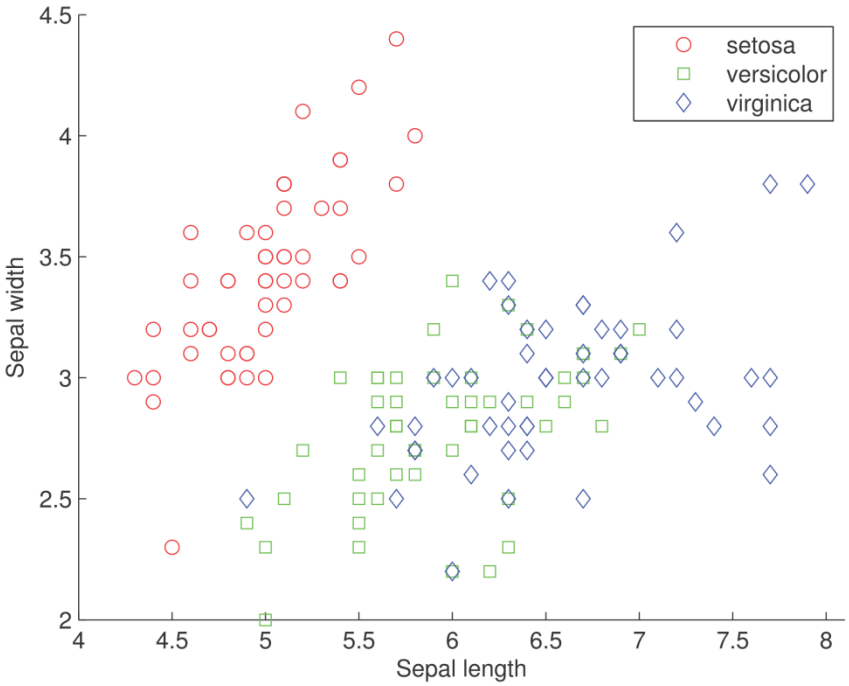


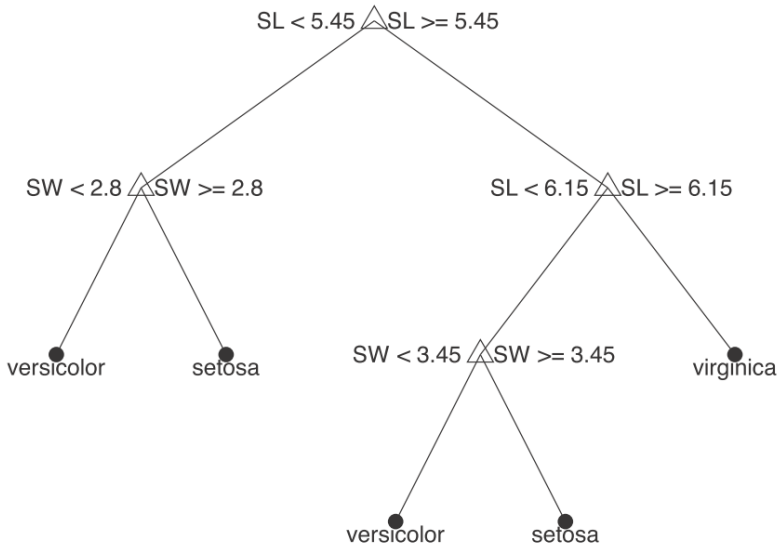


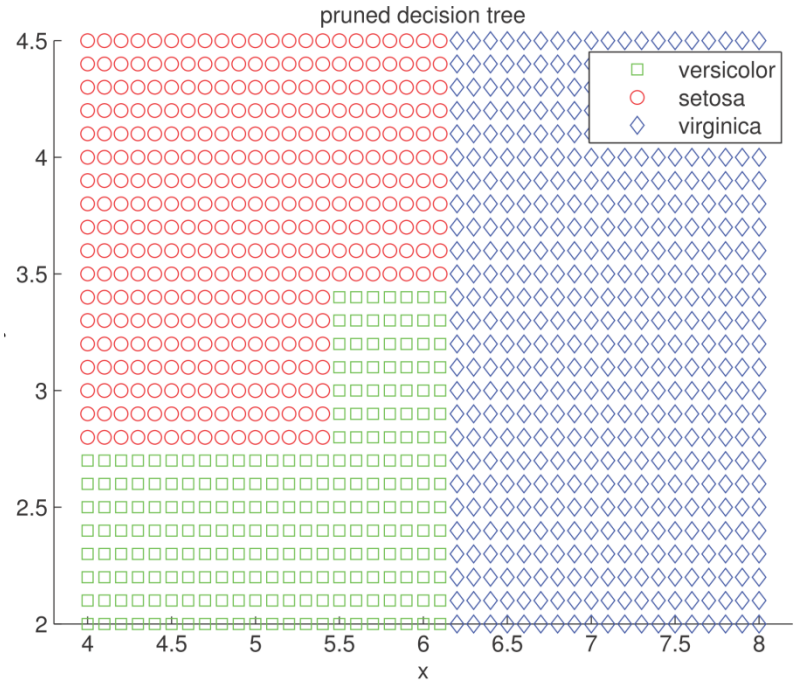
剪枝

- 决策树的生长与过拟合风险

利用剪枝操作避免过拟合







剪枝

在决策树学习中将已生成的树进行简化的过程称为剪枝 *pruning*。具体地，剪枝从已生成的树上裁掉一些子树或叶结点， 并将其根结点或父结点作为新的叶结点，从而简化分类树模型。

- 预剪枝

在决策树生成过程中，对每个节点在划分前先进行估计若当前节点的划分不能带来决策树泛化性能的提升，则停止划分并将当前节点标记为叶节点。

- 后剪枝

先从训练集生成一颗完整的决策树，然后自底向上地对非叶节点进行考察。若将该节点对应的子树替换成叶节点能够带来决策树**泛化性能的提升**，则将该子树替换为叶节点

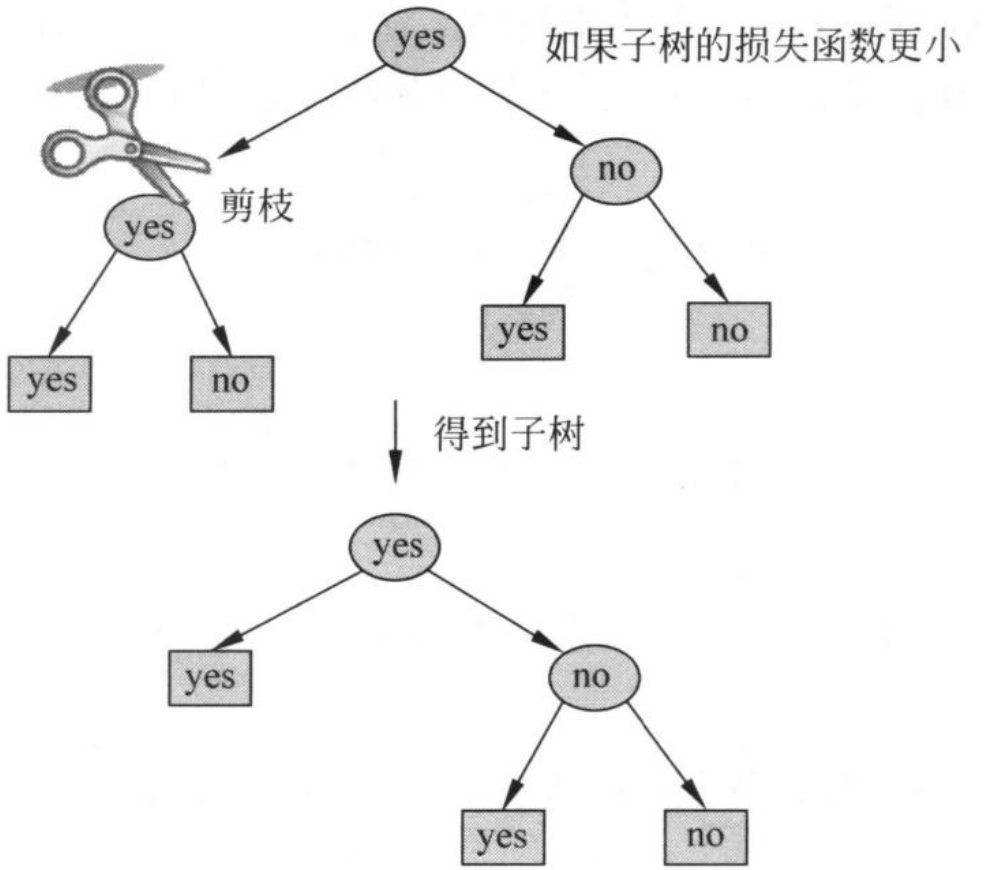
剪枝

- 示例

$$C_\alpha(T) = C(T) + \alpha|T|$$

$$C_\alpha(\mathrm{T}) = \sum_{t=1}^{|\mathrm{T}|} \mathrm{N_t H_t(T)} + \alpha|\mathrm{T}|$$

$$H_t(T) = -\sum_k \frac{N_{tk}}{N_t} \log \frac{N_{tk}}{N_t}$$



输入: 生成算法产生的整个树 T ， 参数 α;
输出: 修剪后的子树 T_α
(1) 计算每个结点的经验熵。
(2) 递归地从树的叶结点向上回缩。

设一组叶结点回缩到其父结点之前与之后的整体树分别为 T_B 与 T_A, 其对应的损失函数值分别是 $C_\alpha(T_B)$与 $C_\alpha(T_A)$, 如果$C_\alpha(T_A) \leq C_\alpha(T_B)$,则进行剪枝，即将父结点变为新的叶结点。

(3) 返回 (2)，直至不能继续为止, 得到损失函数最小的子树