# 决策树分类算法总结

虹 *

2023 年 5 月 6 日

**本文概要**

决策树是一种十分常见的分类和回归算法，决策树学习是从训练集中归纳出一种分类规则，并以树状图形的方式表现出来，可以看成为一个 if-then 规则的集合．决策树分类通常包含 ID3、C4.5、CART 三种算法，三者选择最优特征的方法有所不同：其中 ID3 算法使用的是信息增益；C4.5 算法使用的是信息增益比；CART 算法使用的是基尼指数．三种算法适用的场景也各不相同：

- ID3 算法：适用于离散型数据．ID3 算法使用信息增益来选择最佳的分裂变量，因此需要将数据集离散化为有限的分类．因此，ID3 算法通常用于文本分类、垃圾邮件分类等离散型数据的分类问题．
- C4.5 算法：适用于离散型和连续型数据．与 ID3 算法不同，C4.5 算法使用信息增益比来选择最佳的分裂变量，能够处理连续型和离散型的数据，因此适用范围更广．另一方面：使用信息增益比来选择最佳划分特征，可以防止过度关注具有大量值的特征．C4.5 算法常用于数据挖掘、信用评分等领域．
- CART 算法：适用于离散型和连续型数据，且 CART 算法可以用于分类与回归．该算法通过构建二叉树，来快速地进行分类和预测，因此该算法具有较高的准确性和鲁棒性，对于噪声数据和缺失值具有一定的容错性．

决策树很容易出现过拟合现象,即决策树很容易学习到训练集中所有的信息，这样一来，模型的泛化能力便会降低，因此为了减少决策树的过拟合现象，通常的手段有：**预剪枝与后剪枝**．其中预剪枝是在决策树的生成过程中便开始了，而后剪枝是要事先不加限制地生成一棵完整的决策树，然后自下而上地利用验证集进行剪枝．预剪枝在算法运行时间上上要少于后剪枝，但是针对不同的数据集，树允许的**最大深度，叶节点所需的最小样本量**需要自己进行调参；而后剪枝操作虽然训练成本增加，但是会提高模型的泛化能力．因此，选择预剪枝还是后剪枝操作取决于具体的数据集情况．

关键词：ID3　C4.5　CART　决策树减枝　递归算法

*微信公众号：数学与智能科学

# 目录

# 1 三种算法的优缺点

## 1.1 三种算法的优点

- ID3 算法：算法简单易懂，计算效率高．对于数据缺失的情况有很好的处理能力．
- C4.5 算法：
  - 支持离散型和连续型数据的处理．
  - 采用信息增益比来选择最佳分裂变量，能够避免 ID3 算法中选择取值较多的属性作为分裂变量的问题．
  - 采用剪枝操作，能够有效地防止过拟合．
- CART 算法：
  - 能够处理离散型和连续型数据的分类和回归问题．
  - 采用基尼指数来选择最佳分裂变量，能够更好地处理连续型数据
  - 采用剪枝操作，能够有效地防止过拟合．

## 1.2 三种算法的缺点

- ID3 算法：
  - 对于连续型数据的处理能力不足．
  - 容易产生过拟合，不能很好地应对噪声数据．
- C4.5 算法：
  - 对于噪声数据的处理能力不足．
  - 计算效率较低，需要对数据进行多次扫描．
- CART 算法：
  - CART 算法生成的是二叉树，对于多分类问题需要进行二次划分，增加了计算复杂度．
  - 对于缺失数据的处理能力有限．

# 2 算法数学原理

## 2.1 ID3 算法

ID3 算法使用信息增益来选择数据的特征：样本 D 对特征 A 的信息增益：

$$
\begin{aligned}
g(D,A) &= H(D) - H(D|A) \\
H(D) &= -\sum_{k=1}^{K} \frac{|c_k|}{|D|} \\
H(D|A) &= \sum_{i=1}^{n} \frac{|D_i|}{|D|} H(D_i)
\end{aligned}
\tag{1}
$$

利用 ID3 算法生成决策树的框架如下：

**Algorithm 1** Generate ID3 DecisionTree T

**Input:** X_train,y_train,feature_names,depth=0.

**Initialize:**

确定预剪枝参数：max_depth,min_Samples_split,min_Samples_leaf

1: **while** $X\_train \neq \emptyset$ *or* $len(Counter(y\_train) \neq 1)$ **do**

2: 1. 计算 X_train 的信息熵

2. 如果 y_train 中所有标签属于同一类 $C_k$，则 T 为单结点树，将 $C_k$ 作为类标记，返回 T

3. 如果 X_train = ∅, 则 T 为单结点树，将 y_train 中数量最大的类 $C_k$ 作为该结点的类标记，返回 T

4. 按照（1）计算每个特征的信息增益，选择最优的特征 $A_g$

5. 特征 $A_g$ 的每个取值 $a_i$ 将 D 划分为多个非空子集 $D_i$

6. 遍历 5 中的 $a_i$，令 $X\_train=D_i$ 递归调用 1-5，构建子树.

3: **end while**

4: **Output:** ID3 DecisionTree.

## 2.2 C4.5 算法

C4.5 算法使用的是信息增益比来选择数据的特征，样本 D 对特征 A 的信息增益比为：

$$g_R(D,A) = \frac{g(D,A)}{H_A(D)}$$
$$H_A(D) = -\sum_{i=1}^{n} \frac{|D_i|}{|D|} \log_2 \frac{|D_i|}{|D|} \tag{2}$$

利用 C4.5 算法生成决策树的框架如下：

**Algorithm 2** Generate C4.5 DecisionTree T

**Input:** X_train,y_train,feature_names,depth=0.

**Initialize:**

确定预剪枝参数：max_depth,min_Samples_split,min_Samples_leaf

1: **while** $X\_train \neq \emptyset$ *or* $len(Counter(y\_train) \neq 1)$ **do**

2: 1. 计算 X_train 的信息熵

2. 如果 y_train 中所有标签属于同一类 $C_k$，则 T 为单结点树，将 $C_k$ 作为类标记，返回 T

3. 如果 X_train = ∅, 则 T 为单结点树，将 y_train 中数量最大的类 $C_k$ 作为该结点的类标记，返回 T

4. 按照（2）计算每个特征的信息增益，选择最优的特征 $A_g$

5. 特征 $A_g$ 的每个取值 $a_i$ 将 D 划分为多个非空子集 $D_i$

6. 遍历 5 中的 $a_i$，令 $X\_train=D_i$ 递归调用 1-5，构建子树.

3: **end while**

4: **Output:** C4.5 DecisionTree.

# 3 CART 分类算法

定义 **1.** 基尼指数：在分类问题中，假设有 $K$ 个类，样本点属于第 $k$ 类的概率为 $p_k$, 则概率分布的基尼指数为：

$$Gini(p) = \sum_{k=1}^{K} p_k(1-p_k)$$

特征 A 条件下集合 D 的基尼指数：

$$Gini(D,A)=\frac{|D_1|}{|D|}Gini(D_1)+\frac{|D_2|}{|D|}Gini(D_2)$$

CART 算法构建的是二叉树模型，不仅要选择最优的特征进行分支，同时还要选择最优特征的最优切分点

## 4 决策树的剪枝

决策树容易过拟合，因此我们需要对决策树进行剪枝操作，常见的减枝操作有预剪枝、后减枝.

下面给一个对决策树进行预剪枝的示例：我们对 make_moons 数据集分别建立两颗决策树，一颗不加限制，另一颗决策树要求叶节点具有最小的样本数为 4，否则不再继续向下分枝，对比两棵决策树的分类效果如下：下面再给出一个决策树进行后剪枝的示例：同样的对 make_moons 的数据集分别建立两颗决

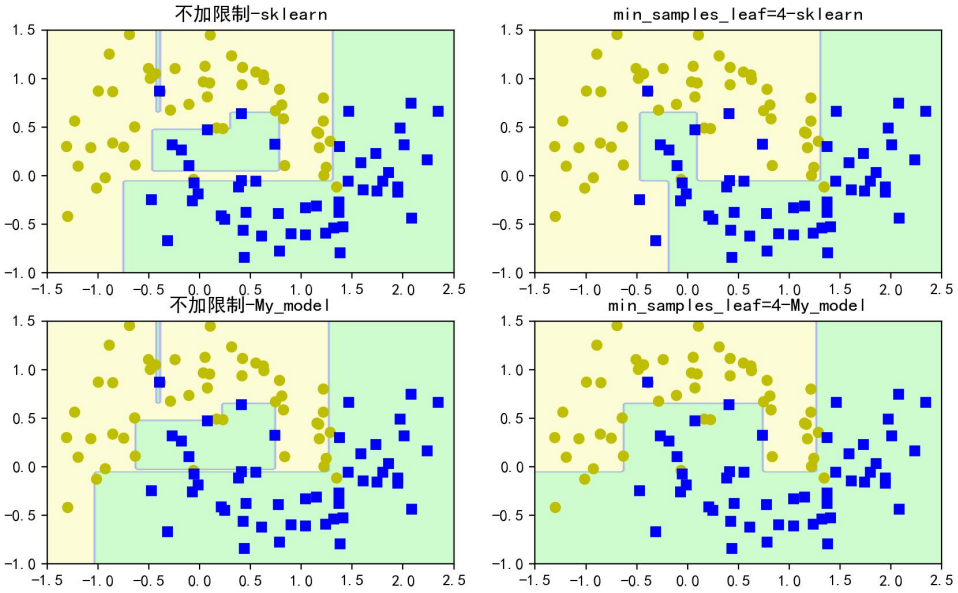


图 1 有无预剪枝的决策树边界对比

策树，其中一颗不加限制，第二棵进行后剪枝操作，对比两棵决策树的分类效果如下：从图1中可以看出：第一棵无剪枝决策树的边界十分复杂，很明显出现了过拟合的现象. 上面的两个子图是基于 sklearn 中的 DecisionTreeClassifier 函数的结果，下面的两个子图是自己根据三种算法的数学原理自己编写的，可以看出，算法的复现是正确的.

### 4.1 预剪枝

一般来说，预剪枝对于何时停止决策树的生长有下面这几种方法：1. 当决策树达到一定的深度的时候，停止生长；2. 当到达当前节点的样本数量小于某个阈值的时候，停止树的生长；3. 计算决策树每一次分裂对验证集的准确度是否提升，当没有提升或者提升程度小于某个阈值的时候，则停止决策树的生长；预剪枝具有思想直接、算法简单、效率高等特点，适合解决大规模的问题，但是如何准确地估计何时停止决策树的生长，针对不同的问题应该分别考虑，需要一定的经验判断.

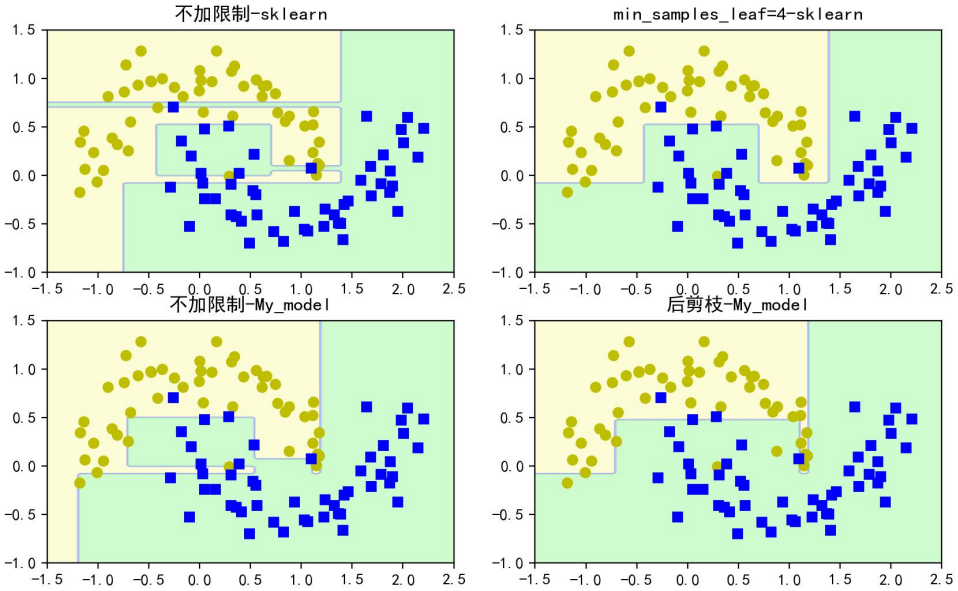


图 2 有无后剪枝的决策树边界对比

## 4.2 后剪枝

决策树的预剪枝操作是在决策树生成的过程中就已经开始了，然而后剪枝操作是按照某种特征筛选准则（ID3、C4.5、CART）先无限制地生成一颗决策树，然后在这棵决策树的基础上进行自下而上的剪枝.

决策树的后剪枝（post-pruning）是一种用于减小决策树过拟合的技术，它通过修剪已经生成的决策树来提高模型的泛化能力. 后剪枝的过程是在决策树构建完成后进行的，具体步骤如下：

- 将训练集分为训练集和验证集两部分，通常使用交叉验证的方法来得到验证集.
- 对于决策树的每个非叶节点，尝试将其替换成一个叶节点，并计算在验证集上的性能.
- 如果替换后的决策树在验证集上的性能没有降低，则接受替换操作，否则不接受替换操作，维持原来的决策树
- 重复步骤 2 和 3 直到所有非叶节点都被尝试过

需要注意的是，后剪枝过程可能会导致决策树的性能下降，因为我们可能会削减掉一些有用的分支. 因此，我们需要在训练集和验证集之间找到一个平衡点，以尽可能地减少过拟合的风险，同时保持模型的预测能力.