

中山大学数据科学与计算机学院 计算机科学与技术专业-人工智能 本科生实验报告

(2018-2019 学年秋季学期)

学 号： 16337113
姓 名： 劳马东
教学班级： 教务 2 班
专 业： 超算

1 实验题目

实现 ID3,C4.5,CART 三种决策树。不要求实现连续型数据的处理，不要求实现剪枝。

2 实验内容

2.1 算法原理

2.1.1 决策树

决策树是一棵多叉树，它通过执行一系列测试达到决策。树中内部节点代表对输入属性 A_i 的值的测试（如年龄），从结点射出的分支代表属性 A_i 的可能值（如 ≥ 18 岁），叶节点代表一种决策。

分类决策树模型是一种描述对实例进行分类的决策树，叶节点表示一个类。用决策树分类，从根节点开始，对实例的某一特征进行测试，根据测试结果，将实例分配到其子节点；这时，每一个子节点对应着该特征的一个取值。如此递归地对实例进行测试并分配，直到达到叶节点。最后将实例分到叶节点的类中。

2.1.2 决策树的建树过程

1、初始化

创建根结点，它拥有全部数据集和特征。

2、选择特征

遍历当前结点的数据集和特征，根据某种原则选择一个特征。

3、划分数据

根据这个特征的取值，将当前数据集划分为若干子集。

4、创建结点

为每个子数据集创建一个子结点，并删去选中的特征。

5、递归建树

对每个子结点，回到第 2 步，直到达到边界条件，则回溯。

递归边界条件：

- 剩余样例属于同一类 C ，该节点标记为 C 类叶节点，返回 C ；
- 无样例，此时返回一个缺省值，该值是构造其父节点用到的所有样例中得票最多的类 C_p ，该节点标记为 C_p 类叶节点；
- 无属性可选择，根据多数投票原则选出剩余样例的类别 C_v ，该节点标记为 C_v 类叶节点，返回 C_v 。

6、完成建树

叶子结点采用多数投票的方式判定自身类别。

2.1.3 特征选择指标

假设数据集 (X, Y) 的子数据集个数为 n ，每个子数据集为 $(X_i, Y_i) (i \in [0, n))$ ，属性集 F 的每个属性为 F_j ，则属性 F_j 在指标 M 上所得的分数为：

$$S(Y|F_j) = \sum_{i=0}^{n-1} \left(\frac{|Y_i|}{Y} \times M(Y_i) \right) \quad (1)$$

1、 错误率

对于 Y_i 根据多数投票原则得到其类别 C_i ，错误率计算 Y_i 中出现错误的频率。公式如下：

$$M(Y_i) = \frac{|\{y_k \in Y_i : y_k \neq C_i\}|}{|Y_i|} \quad (2)$$

我们希望最优属性使得错误率最小，即：

$$\begin{aligned} best(F) &= \arg \min_{F_j \in F} S(Y|F_j) \\ &= \arg \max_{F_j \in F} [S(Y) - S(Y|F_j)] \end{aligned} \quad (3)$$

2、 信息增益

信息增益采用熵计算，熵是随机变量不确定性的度量，其计算公式为：

$$M(Y_i) = H(Y_i) = - \sum_k p(y_k) \log_2 p(y_k) \quad (4)$$

$p(v_k)$ 表示 v_k 在 V 上的频率。随机变量 A 的信息增益表示 A 对 Y 的不确定性的减小程度，公式为：

$$I(A) = H(Y) - H(Y|A) \quad (5)$$

我们希望最优属性使得 Y 的不确定性最小，即：

$$\begin{aligned} best(F) &= \arg \max_{F_j \in F} [H(Y) - H(Y|F_j)] \\ &= \arg \max_{F_j \in F} [S(Y) - S(Y|F_j)] \end{aligned} \quad (6)$$

3、 信息增益率

信息增益的方法偏向于找出分支多的属性作为最优属性，因为分支越多代表不确定性越小，信息增益就越大。然而选择分支多的属性容易造成过拟合，因此该方法对其进行改进，在信息增益的基础之上乘上一个惩罚参数。特征取值较多时，惩罚参数较小；特征取值较少时，惩罚参数较大。该参数为特征的熵的倒数。公式如下：

$$\begin{aligned} best(F) &= \arg \max_{F_j \in F} \frac{H(Y) - H(Y|F_j)}{H(F_j)} \\ &= \arg \max_{F_j \in F} \frac{S(Y) - S(Y|F_j)}{S(F_j)} \end{aligned} \quad (7)$$

4、基尼系数

基尼系数原本是用以衡量一个国家或地区居民收入差距的指标，介于 0-1 之间。基尼系数越大，表示不平等程度越高。对于一个随机变量，基尼系数代表其取值的差异程度，计算公式如下：

$$M(Y_i) = G(Y_i) = \sum_k p(y_k) \times (1 - p(y_k)) = 1 - \sum_k p(y_k)^2 \quad (8)$$

我们希望选择的最优属性使 Y 取值的差异程度最小，即：

$$\begin{aligned} best(F) &= \arg \min_{F_j \in F} G(Y|F_j) \\ &= \arg \min_{F_j \in F} S(Y|F_j) \\ &= \arg \max_{F_j \in F} [S(Y) - S(Y|F_j)] \end{aligned} \quad (9)$$

从以上分析可以看出，特征选择的公式可以统一化为一个公式：

$$best(F) = \arg \max_{F_j \in F} \frac{S(Y) - S(Y|F_j)}{punishment} \quad (10)$$

对于信息增益率方法， $punishment = H(F_j)$ ，其余方法为 1。

2.2 伪代码

1、 决策树建树过程基本框架

输入: 训练集 X , 训练集 y , 父节点 $parent$, 分支的取值 $branch$

```
1: function build_tree( $X, y, parent, brach$ )
2:    $F \leftarrow feature(X)$ 
3:    $root \leftarrow Node(branch, parent)$ 
4:   if  $X$  is empty then
5:      $root.label \leftarrow parent.label$ 
6:   else if  $F$  is empty then
7:      $root.label \leftarrow vote(X)$ 
8:   else if  $is\_same(y)$  then
9:      $root.label \leftarrow y[0]$ 
10:  else
11:     $root.label \leftarrow vote(y)$ 
12:     $S_y \leftarrow M(y)$ 
13:    for each  $F_j \in F$  do
14:       $S_j \leftarrow 0$ 
15:      for each  $(F_{ji}, X_i, y_i) \in divide(F_j, X, y)$  do
16:         $S_j \leftarrow S_j + \frac{|y_i|}{|y|} \times M(y_i)$ 
17:      end for
18:       $S_j \leftarrow \frac{S_y - S_j}{punishment}$ 
19:       $S_{best}, F_{best} \leftarrow \max(S_{best}, F_{best}, S_j, F_j)$ 
20:    end for
21:     $root.feature \leftarrow F_{best}$ 
22:    // 根据所选属性划分数据集
23:    for each  $(branch, X_i, y_i) \in divide(F_{best}, X, y)$  do
24:       $sub\_tree = build\_tree(X_i, y_i, root, branch)$  // 递归建树
25:       $root.add\_child(sub\_tree)$ 
26:    end for
27:  end if
28:  return  $root$ 
29: end function
```

2.3 关键代码

- 1、特征选择：遍历所有属性（编号为数字），根据该属性划分数据集为若干子数据集，计算子数据集在某种指标上的得分，最后求这些子数据集得分的均值。如果该指标带有惩罚参数，得分均值需要乘上惩罚参数。

```
score_best, feature_best = None, None
score_y = self._M(y)
# col为特征编号
for col in cols:
    score_j = 0
    # 根据特征划分数据集并计算得分（频率*指标得分的和）
    for sub_rows in self._divide(col, rows).values():
        score_j += len(sub_rows) / len(rows) \
            * self._M(self._y.take(sub_rows, 0))
    score_j = score_y - score_j
    # 处理有惩罚的指标，如信息增益率
    if self._punish:
        feature = self._X[:, col].take(rows, 0)
        score_j /= self._M(feature)
    # 找出得分最高的属性
    if score_best is None or score_j > score_best:
        score_best = score_j
        feature_best = col
```

图 1: 特征选择

- 2、数据集划分与递归：选择出最优属性后，需要将其从属性集合中删除，并创建一个内部结点，代表一种划分方法。该结点的类标签由数据集根据多数投票原则得出，当在测试集上无相应分支时返回这个内部结点的类标签。根据最优属性划分数据集递归建树，值得注意的地方是 cols 需要传副本而不是本身，否则在回溯时 cols 的值将发生改变。

```
# 删除最优属性
cols.remove(feature_best)
# 根据所选属性创建新内部节点，其label根据多数投票原则得到
root = self.create_node(tag=(branch,
                             feature_best,
                             self._vote(y)),
                        parent=parent)
for branch, sub_rows in self._divide(feature_best, rows).items():
    self._build_tree(sub_rows, cols[:], root, branch)
```

图 2: 数据集划分与递归

- 3、 预测：预测的过程就是顺着决策树不断匹配的过程。取测试数据在对应属性上的值，如果有匹配的分支则进入分支继续重复该过程，否则返回结点的类标签。

```
def _predict(self, test_data, node_id):
    while True:
        node = self.get_node(node_id)
        # tag[1]存储划分属性
        pro = node.tag[1]
        # 到了叶节点
        if pro is None:
            break
        # 取测试数据某一属性的值
        condition = test_data[pro]
        # 匹配当前节点的每个子节点
        for child in self.children(node_id):
            if child.tag[0] == condition:
                node_id = child.identifier
                break
        else:
            break
    return node.tag[2], node.is_leaf()
```

图 3: 预测

3 实验结果及分析

3.1 实验结果展示示例

1、 数据集样本

挑选实验数据前 20 行作为训练集，21-25 行作为测试集。

1	high	high	5more	more	small	low	0
2	high	low	2	4	med	med	0
3	vhigh	med	4	4	small	med	0
4	low	vhigh	4	4	small	med	0
5	med	vhigh	3	4	small	low	0
6	med	high	2	more	big	high	1
7	high	vhigh	4	2	big	low	0
8	vhigh	high	3	4	med	high	0
9	high	vhigh	4	4	small	low	0
10	med	vhigh	4	2	big	low	0
11	vhigh	vhigh	2	2	big	med	0
12	low	vhigh	5more	2	small	high	0
13	med	vhigh	3	2	big	low	0
14	low	med	5more	more	big	low	0
15	low	high	4	2	big	med	0
16	med	med	2	2	med	high	0
17	med	med	5more	4	med	low	0
18	med	vhigh	5more	2	small	med	0
19	low	high	2	more	big	med	1
20	low	med	5more	more	med	med	1

图 4: 训练集

21	low	vhigh	2	more	small	high	0
22	vhigh	low	3	2	med	high	0
23	med	med	4	more	small	high	1
24	vhigh	med	5more	2	small	med	0
25	low	vhigh	5more	4	med	med	1

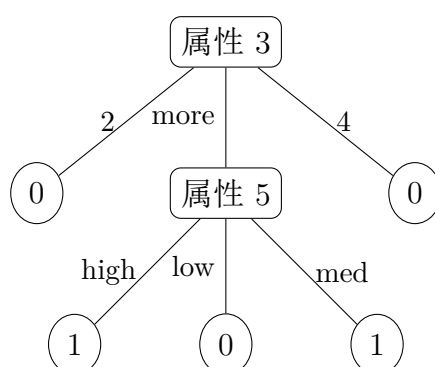
图 5: 测试集

2、建树过程

以 CART 树为例，第一次递归根节点有全部数据集，各个属性计算得到的基尼系数为 0 : 0.219, 1 : 0.2, 2 : 0.203, 3 : 0.12, 4 : 0.23, 5 : 0.225，其中最小的是属性 3，因此选择属性 3 作为划分标准。属性 3 有三种取值：取值为 2 的有第 6, 9, 10, 11, 12, 14, 15, 17 行，y 值全部为 0；取值为 4 的有第 1, 2, 3, 4, 7, 8, 16 行，y 值全部为 0；取值为 more 的有第 0, 5, 13, 18, 19 行，计算得到属性 0、1、2、4、5 的基尼系数分别为 0.267, 0.467, 0.267, 0.267, 0，最小的是属性 5。继续以属性 5 划分数数据集，以此类推，过程正确。

层数	分支	数据集/行	基尼系数	选择	类
1	-	[0, 1, 2, ... , 17, 18, 19]	0.219, 0.2, 0.203, 0.12, 0.230, 0.225	3	-
2	4	[1, 2, 3, 4, 7, 8, 16]	-	-	0
2	2	[6, 9, 10, 11, 12, 14, 15, 17]	-	-	0
2	more	[0, 5, 13, 18, 19]	0:0.267, 1:0.467, 2:0.267, 4:0.267, 5:0	5	-
3	low	[0, 13]	-	-	0
3	high	[5]	-	-	1
3	med	[18, 19]	-	-	1

3、决策树建树结果



4、决策结果

对于测试集第 0 行，其属性 3 的值为 more，进入 more 子树；其属性 5 的值为 high，预测为 1，符合；

对于测试集第 1 行，其属性 3 的值为 2，预测为 0，符合；

以此类推，预测结果符合预期；

测试数据/行	预测标签
21	1
22	0
23	1
24	0
25	0

3.2 评测指标展示及分析

1、准确率

采用 5 折交叉验证，数据集 80% 作为训练集，20% 作为验证集，分别评测 ID3、C4.5 和 CART 在 5 次数据集划分的准确率，最后求平均准确率，如下表。

三种模型的准确率都达到了 95%，其中 ID3 和 CART 准确率相近（表中一样是已经四舍五入之后的结果），C4.5 效果最好，平均准确率在 95.3%。

模型	5 折交叉验证准确率					平均准确率
ID3	92.6%	96.9%	95.7%	95.0%	95.4%	95.1%
C4.5	95.1%	96.6%	95.7%	94.7%	94.4%	95.3%
CART	92.6%	96.9%	95.7%	95.0%	95.4%	95.1%

4 思考题

- 决策树有哪些避免过拟合的方法？

导致过拟合的原因主要有两种：建模时没有对决策树的生长进行限制，导致其完美拟合训练数据，但是在测试数据上表现较差；样本数据含有噪声或样本缺乏代表性，模型错误地理解了输入输出之间的关系。对应的方法有两种：

- (1) 剪枝：对于当前节点，在选择了某个属性后，如果它提高在验证集上的准确率就继续划分，否则将当前节点作为叶节点返回（预剪枝）；决策树建好后，自底向上考虑非叶节点，如果将其变成叶子节点能提高在验证集的准确率就把它变成叶子节点。
- (2) 用多棵决策树的综合结果作为预测结果。在 GBDT 模型中，采用很多棵树来学习训练集，而且每棵树的学习结果只作为全局学习结果的一小部分。

- C4.5 相比于 ID3 的优点是什么，C4.5 又可能有什么缺点？

优点：

- a. 用信息增益率来选择属性，克服 ID3 选择属性时偏向选择取值多的属性的不足；
- b. 既可以处理离散型属性，也可以处理连续性属性。

缺点：

- a. 在建树过程中需要计算属性的熵，算法效率降低；
 - b. 只适合于处理能全部驻留在内存中的训练集，当训练集很大无法存下时，程序无法运行。
- 如何用决策树来进行特征选择（判断的重要性）？

对于一个数据集，可以先用决策树模型取训练，找出准确率最高时对应的决策树。决策树的每个内部节点都包含一个特征，给每个内部节点的特征赋予一个权重，根节点最大，越往下权重越小，不在决策树中的属性权重最小。如此就能选出数据集中的重要特征。