

中山大学数据科学与计算机学院

移动信息工程专业-人工智能

本科生实验报告

(2017-2018 学年秋季学期)

课程名称: Artificial Intelligence

教学班级	周五 5-6 节	专业 (方向)	软件工程 (移动信息工程) M2
学号	15352204	姓名	林丹

一、实验题目

实验四——决策树算法

二、实验内容

1. 算法原理

● 什么是决策树?

决策树是有监督的分类模型。

决策树是一种树形结构, 能够对于样本的特征向量预测其标签。

树中包含叶子节点与内部节点。内部节点是某一种特征, 叶子节点是预测的标签结果。

● 算法关键:

构造决策树的关键步骤是选择分裂的特征

启发式方法:

目标是选择一个有较大决定作用的属性进行分类

● 判断一个特征 (属性) 对数据集的重要性目前常用的三种指标:

• 使用信息增益: ID3 算法

假设数据集 D 的标签有 $\{d_0, d_1, \dots, d_m\}$, 个数分别为 n_0, n_1, \dots, n_m

数据集的信息熵

$$H(D) = - \sum_D P(d_i) \log P(d_i)$$
$$P(d_i) = \frac{n_i}{\sum_k^m n_k}$$

特征 A 的特征取值有 $\{a_0, a_1, \dots, a_q\}$,特征 A 对数据集 D 的条件熵:

$$H(D|A = a_i) = -\sum_L P(d_i|a_i) \log P(d_i|a_i)$$

$$H(D|A) = \sum_i^q p(a_i) \times H(D|A = a_i)$$

信息增益=互信息

$$I(D; A) = H(D) - H(D|A)$$

信息增益越大，或者说特征 A 对数据集 D 的条件熵越小，表明。。。

• 使用信息增益率：C4.5

使用信息增益率 C4.5 的原因是因为 ID3 算法的局限性，这里举一个极端的例子。

证明 ID3 算法偏向于选择属性取值个数多的特征：

假设在一个关于是否买电脑的数据集里面，增加一列属性为身份证号码(每个人唯一的 ID)，那么每个 ID 有对应的一个标签为“买”或者“不买”。

根据公式

$$H(D|A = a_i) = -\sum_L P(d_i|a_i) \log P(d_i|a_i)$$

$$H(D|A = \text{“身份证号码”}) = \sum_i^q p(a_i) \times H(D|A = a_i)$$

$$= -\sum_L P(d_i|a_i) \log P(d_i|a_i) = 14 \times \frac{1}{14} \times (-1 \times \log(1) - 0 \times \log(0)) = 0$$

$$G(D, A) = H(D) - H(D|A)$$

所以， $G(D, A = \text{“身份证号码”}) = H(D)$

所以属性身份证号码的信息增益最大，根据 ID3 算法选择身份证号码作为分裂属性。

但是，显然身份证号码对于判断一个人是否买电脑是没有实际意义的，原因出在身份证号码这个属性取值个数太多，导致条件熵太小，所以信息增益太大。

所以 ID3 算法偏向于选择属性取值个数多的特征。

为了平衡信息增益和属性特征值个数之间的矛盾，引入信息增益率，分子为信息增益，分母特征值 A 的熵，特征值 A 的熵反映了这个特征本身的不确定性，如果这个属性本身不确定性就很大，那就越不倾向于选取它。选择信息增益率最大的作为分裂属性。

假设特征 A 的特征取值有 $\{a_0, a_1, \dots, a_q\}$,

为了简单起见，假设每一个特征 a_i 出现的次数一样，即 $p(a_i) = \frac{1}{d}$

特征 A 的信息熵：

$$\text{GainRatio}_A(D) = \text{Gain}_A(D) / \text{SplitInfo}_A(D)$$

$$\text{SplitInfo}_A(D) = - \sum_{j=1}^v \frac{|D_j|}{|D|} \times \log_2 \left(\frac{|D_j|}{|D|} \right)$$

$$\text{splitInfo}(D, A) = -P(d_i) \log P(d_i) = -d \times \frac{1}{d} \times \log \left(\frac{1}{d} \right)$$

可以看到随着 d 的增加, $\text{splitInfo}(D, A)$ 增加, 意味着不确定性增加, 信息增益率减小, 符合我们的期望。

• 使用 GINI 指数: CART

前的 ID3 和 C4.5 都是利用了信息熵的模型, 来反映模型的“不纯度”。

CART 决策树采用 GINI 指数来反映数据集的“纯度”

$$\text{gini}(D_j | A = A_j) = \sum_{i=1}^n p_i(1 - p_i) = 1 - \sum_{i=1}^n p_i^2$$

$$\text{gini}(D, A) = \sum_{j=1}^v p(A_j) \times \text{gini}(D_j | A = A_j)$$

跟信息熵对比, 相当于红色框的部分由条件熵换成了 GINI 指数

我们看到 GINI 指数是没有负号的, 所以基尼系数越小, 则不纯度越低, 特征越好。

2. 伪代码

决策树分类算法流程图:

初始化

- 创建根结点, 它拥有全部数据集和全部特征

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。

划分数据

- 根据这个特征的取值, 将当前数据集划分为若干个子数据集

创建结点

- 为每个子数据集创建一个子结点, 并删去刚刚选中的特征。

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。

完成建树

- 叶子结点采用多数投票的方式判定自身的类别。



2.1 原始的决策树算法（离散值，没有剪枝）：

Function tree = make_tree(D, L, mode)

Input: m-samples-dataset with n attributes $D \in \mathbb{R}^{m \times n}$,

corresponding label set $L \in \mathbb{R}^{m \times 1}$

a selected attribute A have d values $\{a_1, a_2, \dots, a_d\}$

mode: 1 for ID3; 2 for C4.5; 3 for GINI

U_label, unique set for set L

Initialize:

Attribute = 1..n

tree.dim = 0

if n equals 1 or length(U_label) equals 1, then

tree.child = max frequency label ; return;

endif

for each attribute:

if mode==1, then compute gain(D,A)

else if mode==2, then compute gainRatio(D,A)

else if mode==3, then compute GINI(D,A)

select the most important attribute A;

tree.dim = A as divided attribute

tree.Nf = $\{a_1, a_2, \dots, a_d\}$ as divided attribute values

if d==1, then

tree.child = max frequency label; return;

endif

Attribute' = Attribute delete A

tree.child(d+1) = max frequency label in A;

Loop:

for 1..d

matcheds = find(D.A == a_d);

datao = data(matcheds,dims);

tree.child(i) = make_tree(D(matcheds), label(matcheds), mode);

endLoop

Output: tree

End Function make_tree



2.2 利用决策树预测算法:

Function predicts = use_tree_predict(D, matcheds, tree, U_label)

Input: m-samples-dataset with n attributes $D \in \mathbb{R}^{m \times n}$,

matcheds, vector of samples belong to this node

U_label, unique set for labels

tree, a structure (dim, Nf[], child[])

dim, divided attribute

Nf[], divided attribute value set

child[], the child nodes of divided node

if tree.dim == 0, then

predicts(matcheds) = tree.child; return;

Attribute = 1:n, then delete tree.dim

Loop:

for x = 1:unique(A) { a_1, a_2, \dots, a_d }

matcheds' = matcheds(find(D(matcheds) == a_x));

if a_x exists in tree.Nf ,

then

tree' = tree.child(find(a_x == tree.Nf));

predicts = predicts + use_tree_predict(D(matcheds), matcheds', tree', U_label);

else

tree' = tree.child(d+1);

predicts = predicts + use_tree_predict(D(matcheds), matcheds', tree', U_label);

endif

EndLoop

End Function make_tree



3. 关键代码截图（带注释） 基于 MATLAB 的实现

3.1 构建一棵决策树：

```
1 function tree = make_tree(data, labels, inc_node, mode)
2 % data -- 输入的样本特征数据矩阵，行为样本，列为特征， N_sample*N_attri
3 % labels -- 输入的样本标签向量
4 % inc_node -- 递归边界，数据子集的样本个数
5 % mode -- 模式，1: ID3; 2:C4.5; 3:GINI
6
7 %% 函数部分：
8 [N_sample, N_attri]=size(data);
9 U_label=unique(labels); % U_label = [-1,+1]
10 N_Ulabel = length(U_label); % N_label = 2
11 tree.dim = 0;
12 %tree.child = [];
13
14 %% 如果剩余训练样本太小(小于inc_node)，或只剩一个，或只剩一类标签，退出
15 if ((inc_node > N_attri) || (N_attri == 1) || (N_Ulabel == 1))
16 %如果剩余训练样本太小(小于inc_node)，或只剩一个，或只剩一类标签，退出
17 H = hist(labels, N_Ulabel); %统计样本的标签，分别属于每个标签的数目 H(1×标签数目)
18 [~,largest] = max(H);
19 %得到包含样本数最多的那个标签的索引，记为largest 包含多少个样本，记为m
20 tree.Nf = [];
21 tree.child = U_label(largest);%姑且直接返回其中包含样本数最多一类作为其标签
22 return
23 end
24
25 %% 计算数据集的信息熵 H_Data
26 %P_label = zeros(1,N_Ulabel);
27 for i=1:N_Ulabel
28 P_label(i) = length( find( labels == U_label(i))) / N_sample;
29 end
30 H_Data = sum( -P_label .* H_log( P_label ) );
```

选择一个分裂属性：

```
36 for x = 1:N_attri %遍历每一个特征
37 S_Avalue = data( :, x); % 每一个特征的所有样本聚合
38 U_Avalue = unique( S_Avalue ); % 该特征的不重复特征值集合
39 N_UAvalue = length( U_Avalue ); % 该特征的不重复特征值个数
40 P_Matrix = zeros( N_Ulabel, N_UAvalue); % 特征矩阵（概率矩阵）的构造
```



```
41 - for i=1:N_Ulabel % 两行 -1, +1
42 -     for j=1:N_UAvalue
43 -         matcheds = find( (labels==U_label(i))&(S_Avalue==U_Avalue(j)) );
44 -         P_Matrix(i,j) = length(matcheds);
45 -     end
46 - end

47 - Pk = sum(P_Matrix);
48 - %取P的每一列的和，也就是得到当前所有样本中，这个特征的特征值==这个特征值的样本数目
49 - %Pk(1×特征值数目)表示这个特征的特征值等于每个特征值的样本数目
50 - P_sum = repmat(Pk, N_Ulabel, 1); %把Pk复制成判别标签个数 的行，列还是一组
51 - %这主要在保证P1作被除数时不等于0
52 - P_sum = P_sum + eps*(P_sum==0);
53 - P_Matrix = P_Matrix ./ P_sum;
54 - % 得到当前所有样本中，这个特征的值等于每个特征值且标签等于每个标签的样本，
55 - % 占当前这个特征值中的样本的比例
56 - Pk = Pk / sum(Pk);
57 - %得到当前所有样本中，这个特征的值等于每个特征值的样本，
58 - % 占当前样本总数的比例
59 - % 例如 A=age: Pk = [5/14, 4/14, 5/14];

61 - %% 计算 条件概率
62 - % Sum_subH = sum(-P_Matrix .* log2(P_Matrix));
63 - Sum_subH = sum(-P_Matrix .* H_log(P_Matrix));
64 - %对特征集中每个特征分别计算信息熵 info(1×特征值数目)
65 - H_DA = sum( Pk .* Sum_subH );

66 - %% 计算gain和gainRatio
67 - gain(x) = H_Data - H_DA; % 信息增益
68 - splitInfo = (sum(-Pk .* H_log(Pk))); % 特征本身的不确定性
69 - gainRatio(x) = gain(x) / splitInfo; % 信息增益率

70 - %% 计算GINI指数
71 - sub_gini = sum( P_Matrix .* ( 1-P_Matrix) );
72 - GINI(x) = sum( Pk .* sub_gini );
73 - end

75 - %% 选出分裂的特征
76 - if mode == 1
77 -     [~,dim] = max(gain);
78 - elseif mode == 2
79 -     [~,dim] = max(gainRatio);
80 - elseif mode == 3
81 -     [~,dim] = min(GINI);
82 - end
```



```

83 - tree.dim = dim ;%记为树的分裂特征
84 - Nf = unique(data(:,dim));
85 - %得到选择的这个作为分裂特征的特征的那一行 也就是得到当前所有样本的这个特征的特征值
86 - Nbins = length(Nf); %得到这个特征的无重复的特征值的数目
87 - tree.Nf = Nf; %记为树的分类特征向量 当前所有样本的这个特征的特征值

89 - %% 判断是否继续从数据子集分裂
90 - H = hist(labels,N_Ulabel );
91 - %统计当前所有样本的标签，分别属于每个标签的数目 H(1×标签数目)
92 - [~,largest] = max(H);
93 - %得到包含样本数最多的那个标签的索引，记为largest 包含多少个样本，记为m
94 - if Nbins == 1 %无重复的特征值的数目==1，即这个特征只有这一个特征值，就不能进行分裂
95 -     tree.Nf = []; %因为不以这个特征进行分裂，所以Nf和split_loc都为空
96 -     tree.child= U_label(largest); %姑且将这个特征的标签就记为包含样本数最多的那个标签
97 -     return
98 - end

100 - %% 增加一个多余的叶子节点，处理测试集中没有出现过的特征值
101 - tree.child(Nbins+1).dim = 0;
102 - tree.child(Nbins+1).Nf = [];
103 - tree.child(Nbins+1).child = U_label(largest);

105 - %% 创建子节点
106 - dims = 1:N_attri ;
107 - dims(dims==dim)=[]; % 删除这个特征
108 - %如果当前选择的这个作为分裂特征的特征是个离散特征
109 - for i = 1:Nbins %遍历这个特征下无重复的特征值的数目
110 -     matched = find(data(:,dim) == Nf(i));
111 -     %找到当前所有样本的这个特征的特征值为Nf(i)的索引们
112 -     data0 = data(matched,dims);
113 -     label0 = labels(matched);
114 -     tree.child(i) = make_tree( data0, label0,inc_node, mode);%递归
115 -     %因为这是个离散特征，所以分叉成Nbins个，分别针对每个特征值里的样本，进行再分叉
116 - end

```

3.2 使用决策树进行预测：

```

1 - function predicts = use_tree_predict(data, matched, tree, U_label)
2 - %Classify recursively using a tree
3 - % data -- 输入的样本特征数据矩阵
4 - % matched -- 一个向量
5 - % tree -- 建立的决策树
6 - % U_label -- 不重复出现的标签， [-1,+1]

```




```
7 - [N_sample, N_attri]=size(data);
8 - predicts = zeros(N_sample,1); %设置每个样本的初始预测标签都是0
9 - % 迭代的边界条件
10 - if (tree.dim == 0) %这说明达到了树的叶子节点
11 -     predicts(matcheds) = tree.child; %得到样本对应的标签是tree.child
12 -     return
13 - end

15 - %% 不是最终节点，接续往下找
16 - dim = tree.dim; %得到分裂特征
17 - dims= 1:N_attri; %得到特征索引
18 - dims(dims==dim)=[]; % 删除这个特征

20 - %% 根据得到的决策树对测试样本进行分类
21 - U_Avalue = unique(data(:,dim)); %得到这个样本集中这个特征的无重复特征值的集合
22 - for i = 1:length(U_Avalue) %遍历每个特征值，根据特征值来建一颗树
23 -     % 如果U_Avalue(i)不在tree.Nf中出现过，返回0
24 -     if any( tree.Nf==U_Avalue(i) )
25 -         %tree.Nf为树的分类特征向量 当前所有样本的这个特征的特征值
26 -         data0 = data(:, dims);
27 -         matcheds0 = matcheds(find(data(matcheds,dim) == U_Avalue(i)));
28 -         %找到当前测试样本中这个特征的特征值==分裂值的样本索引
29 -         tree0 = tree.child(find(U_Avalue(i)==tree.Nf));
30 -         predicts = predicts + use_tree_predict(data0, matcheds0, tree0, U_label);
31 -         %对这部分样本再分叉

32 -     else % 特征值没有出现过
33 -         data0 = data(:, dims);
34 -         matcheds0 = matcheds(find(data(matcheds,dim) == U_Avalue(i)));
35 -         tree0 = tree.child( length(tree.Nf) ); % 例外情况的结点
36 -         predicts = predicts + use_tree_predict(data0, matcheds0, tree0, U_label);
37 -         %对这部分样本再分叉
38 -     end
39 - end
```

4. 创新点&优化（如果有）

优化思路：预剪枝

原本的算法最后的数据集结点数为1，为防止过拟合，设置结点样本数小于一定阈值结束递归。

在前面的代码截图中已经体现，主要修改在一个地方：

```
%% 如果剩余训练样本太小(小于inc_node)，或只剩一个，或只剩一类标签，退出
if ((inc_node > N_attri) || (N_attri == 1) || (N_Ulabel == 1))
    %如果剩余训练样本太小(小于inc_node)，或只剩一个，或只剩一类标签，退出
    H = hist(labels, N_Ulabel); %统计样本的标签，分别属于每个标签的数目 H(1×标签数目)
    [~,largest] = max(H);
    %得到包含样本数最多的那个标签的索引，记为largest 包含多少个样本，记为m
    tree.Nf = [];
    tree.child = U_label(largest);%姑且直接返回其中包含样本数最多一类作为其标签
    return
end
```

三、 实验结果及分析

1. 实验结果展示示例（可图可表可文字，尽量可视化）【小数据集】

利用 PPT 中的例子：

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no

Table 3.1

将字符串转化为数字，例如 age 特征中，特征值“age≤30”转化为 0，“31..40” 转化为 1，“age>30” 转化为 2。

age	income	student	credit_rating	label
0	2	0	0	-1
0	2	0	1	-1
1	2	0	0	1
2	1	0	0	1
2	0	1	0	1
2	0	1	1	-1
1	0	1	1	1
0	1	0	0	-1
0	0	1	0	1
2	1	1	0	1
0	1	1	1	1
1	1	0	1	1
1	2	1	0	1
2	1	0	1	-1

Table 3.2

- $H(D)=0.940$ 14个训练样本中, 9个买了电脑

$$H(D) = -\frac{9}{14} \log_2 \frac{9}{14} - (1 - \frac{9}{14}) \log_2 (1 - \frac{9}{14})$$

H_Data 0.9403

1. 选择第一层分类属性:

① 使用 ID3 算法

- $H(D)=0.940$
- $H(D|A="age")=0.694$
 $g(D, A) = I(D; A) = H(D) - H(D|A)$
- $g(D, A="age")=0.246$
- $g(D, A="income")=0.029$
- $g(D, A="student")=0.151$
- $g(D, A="credit_rating")=0.048$

- $H(D|A="age")=0.694$

$$H(D|A="age") = \frac{5}{14} \times \left(-\frac{2}{5} \log_2 \frac{2}{5} - \frac{3}{5} \log_2 \frac{3}{5} \right) + \frac{4}{14} \times \left(-\frac{4}{4} \log_2 \frac{4}{4} - \frac{0}{4} \log_2 \frac{0}{4} \right) + \frac{5}{14} \times \left(-\frac{3}{5} \log_2 \frac{3}{5} - \frac{2}{5} \log_2 \frac{2}{5} \right)$$

gain [0.2467,0.0292,0.1518,0.0481]

选择 Gain 最大的属性 1 作为分裂特征。

② 使用 C4.5 算法

$$SplitInfo(D, A = age) = -\frac{5}{14} \times \log \frac{5}{14} - \frac{4}{14} \times \log \frac{4}{14} - \frac{5}{14} \times \log \frac{5}{14} = 1.5774$$

$$gainRatio(D, A = age) = \frac{gain(D, A=age)}{SplitInfo(D, A=age)} = \frac{0.2467}{1.5774} = 0.15639$$

$$gainRatio(D, A = income) = 0.0188$$

$$gainRatio(D, A = student) = 0.1518$$

$$gainRatio(D, A = credit_rating) = 0.0488$$

gainRatio [0.1564,0.0188,0.1518,0.0488]

选择信息增益率指数最大的属性, 即第一个属性 age。

③ 使用 GINI 指数作为属性:

$$Gini(D, A = age) = \frac{5}{14} \times \left(1 - \frac{2^2}{5} - \frac{3^2}{5} \right) + \frac{4}{14} \times \left(1 - \frac{4^2}{4} - \frac{0^2}{4} \right) + \frac{5}{14} \times \left(1 - \frac{3^2}{5} - \frac{2^2}{5} \right) = 0.34285$$

$$Gini(D, A = income) = 0.4405$$

$$Gini(D, A = student) = 0.3673$$

$$Gini(D, A = credit_rating) = 0.4286$$

GINI =
0.3429 0.4405 0.3673 0.4286

选择 GINI 指数最小的属性, 第一个属性为 age。

根据 age 属性的取值, 分成 3 个数据集, 如上文 Table3.3 所示:

根据这个属性 1 的特征取值，将当前数据集划分为 3 个子数据集

age	income	student	credit_rating	label
0	2	0	0	-1
0	2	0	1	-1
0	1	0	0	-1
0	0	1	0	1
0	1	1	1	1
1	2	0	0	1
1	0	1	1	1
1	1	0	1	1
1	2	1	0	1
2	1	0	0	1
2	0	1	0	1
2	0	1	1	-1
2	1	1	0	1
2	1	0	1	-1

Table 3.3

删除分裂的特征，剩余三个特征：


1.1 age 第一个特征值 0:

age	income	student	credit_rating	label
0	2	0	0	-1
0	2	0	1	-1
0	1	0	0	-1
0	0	1	0	1
0	1	1	1	1

Table 3.4

① 使用 ID3 算法


$$H(D) = -\frac{3}{5}\log\left(\frac{3}{5}\right) - \frac{2}{5}\log\left(\frac{2}{5}\right) = 0.97095$$

 H_Data 0.9710

$$H(D|A = \text{income}) = \frac{2}{5}\left(-\frac{2}{2}\log\left(\frac{2}{2}\right) - \frac{0}{2}\log\left(\frac{0}{2}\right)\right) + \frac{2}{5}\left(-\frac{1}{2}\log\left(\frac{1}{2}\right) - \frac{1}{2}\log\left(\frac{1}{2}\right)\right) + \frac{1}{5}\left(-\frac{1}{1}\log\left(\frac{1}{1}\right)\right) = \frac{2}{5}$$

$$\text{gain}(D, A = \text{income}) = 0.571$$

$$\text{gain}(D, A = \text{student}) = 0.9710$$

 gain [0.5710, 0.9710, 0.0200]

$$\text{gain}(D, A = \text{credit_rating}) = 0.02$$

C4.5 算法和 GINI 指数算法过程类似，得到的结果和 ID3 恰好一样，这里省略。

选择当前的第 2 个特征作为分裂特征，即 student 属性。

根据这个 student 属性的特征取值，将当前数据集划分为 2 个子数据集

income	student	credit_rating	label
2	0	0	-1
2	0	1	-1
1	0	0	-1
0	1	0	1
1	1	1	1

Table 3.5

删除分裂的特征 **student**，剩余 2 个特征：

1.1.1 student 特征的第一个特征值 0:

income	student	credit_rating	label
2	0	0	-1
2	0	1	-1
1	0	0	-1

Table 3.6

满足递归边界的第一个条件：

样本属于同一标签“-1”，则将当前结点标记为“-1”叶结点。

1.1.2 student 特征的第一个特征值 1:

income	student	credit_rating	label
0	1	0	1
1	1	1	1

Table 3.7

满足递归边界的第一个条件：

样本属于同一标签“1”，则将当前结点标记为“1”叶结点。

1.2 age 的第二个特征值

age	income	student	credit_rating	label
1	2	0	0	1
1	0	1	1	1
1	1	0	1	1
1	2	1	0	1

Table 3.8

满足递归边界的第一个条件：

样本属于同一标签“1”，则将当前结点标记为“1”叶结点。

1.3 age 的第三个特征值“2”（即 $age > 40$ ）

age	income	student	credit_rating	label
2	1	0	0	1

2	0	1	0	1
2	0	1	1	-1
2	1	1	0	1
2	1	0	1	-1

Table 3.9

$$H(D) = -\frac{3}{5}\log\left(\frac{3}{5}\right) - \frac{2}{5}\log\left(\frac{2}{5}\right) = 0.97095$$

 H_Data 0.9710

$$H(D|A = \text{income}) = \frac{3}{5}\left(-\frac{2}{3}\log\left(\frac{2}{3}\right) - \frac{1}{3}\log\left(\frac{1}{3}\right)\right) + \frac{2}{5}\left(-\frac{1}{2}\log\left(\frac{1}{2}\right) - \frac{1}{2}\log\left(\frac{1}{2}\right)\right) = 0.95097$$

$$\text{gain}(D, A = \text{income}) = 0.02$$

 gain [0.0200,0.0200,0.9710]

$$\text{gain}(D, A = \text{student}) = 0.02$$

$$\text{gain}(D, A = \text{credit_rating}) = 0.971$$

选择当前的第 3 个特征作为分裂特征，即 **credit_rating** 属性。

根据这个 **credit_rating** 属性的特征取值，将当前数据集划分为 2 个子数据集

income	student	credit_rating	label
1	0	0	1
0	1	0	1
1	1	0	1
0	1	1	-1
1	0	1	-1

Table 3.10

删除分裂的特征 **credit_rating**，剩余 2 个特征：

1.3.1 **credit_rating** 的第一个特征值 “0”

income	student	credit_rating	label
1	0	0	1
0	1	0	1
1	1	0	1

Table 3.11

满足递归边界的第一个条件：

样本属于同一标签 “1”，则将当前结点标记为 “1” 叶结点。

1.3.2 **credit_rating** 的第一个特征值 “1”

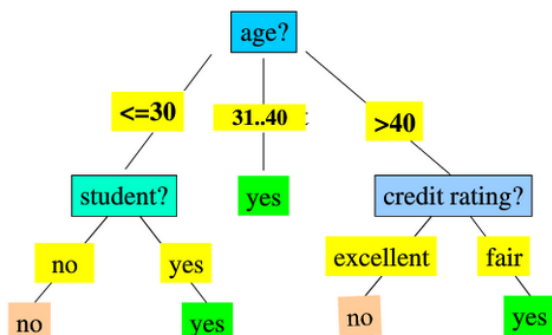
income	student	credit_rating	label
0	1	1	-1
1	0	1	-1

Table 3.12

满足递归边界的第一个条件：

样本属于同一标签“-1”，则将当前结点标记为“-1”叶结点。

根据 MATLAB 工作区得到 tree 结构体，画出决策树：



2. 评测指标展示即分析（如果实验题目有特殊要求，否则使用准确率）

2.1 以整个测试集作为测试样本，计算第一次分裂选择的属性：

dim	4
gain	[0.0823,0.0137,0.0153,0.1041,0.0017,0.0011,0.0052,0.0060,0.0092]
gainRatio	[0.0167,0.0070,0.0096,0.0335,0.0029,0.0013,0.0032,0.0033,0.0189]
GINI	[0.4371,0.4750,0.4751,0.4260,0.4828,0.4832,0.4805,0.4801,0.4781]

可以看到无论是哪一种模型，选择的都是第四个属性（MATLAB 数组下标从 1 开始）

ID3:

字段	值	最小值	最大值
dim	4	4	4
Nf	13x1 double	0	12
child	1x14 struct		

C4.5:

字段	值	最小值	最大值
dim	4	4	4
Nf	13x1 double	0	12
child	1x14 struct		

GINI:

字段	值	最小值	最大值
dim	4	4	4
Nf	13x1 double	0	12
child	1x14 struct		

2.2 划分测试集和验证集:

● 交叉验证:

将数据集分 5 等份, 如果数据集样本个数不是 5 的倍数, 那么舍弃余数部分。

每一份轮流做为验证器, 其他的数据做测试集。

● 如何分 5 等份?

已知 train 中有 787 个样本, 最后保留 785 个样本, 每份有 157 个样本;

策略① 按照样本出现顺序, 前 157 个为第一份, 接着 157 个为第二份, 如此类推, 最后 157 个为第五份;

这个划分的策略有缺陷: 如果样本所属标签不是均匀分配的, 比如这个 train 中前面的标签都是+1, 后面的标签都是-1。如果训练数据都是一类标签会导致生成的决策树的叶子节点都是一样的标签, 没有意义。

策略② 等间隔取样本

第一份样本 sample1 为: 1, 6, 11, 16,...

第二份样本 sample2 为: 2, 7, 12, 17,...

第三份样本 sample3 为: 3, 8, 13, 18,...

第四份样本 sample4 为: 4, 9, 14, 19,...

第五份样本 sample5 为: 5, 10, 15, 20,...

● 交叉验证, 得到的 5 次训练和验证的准确率之后取平均值。

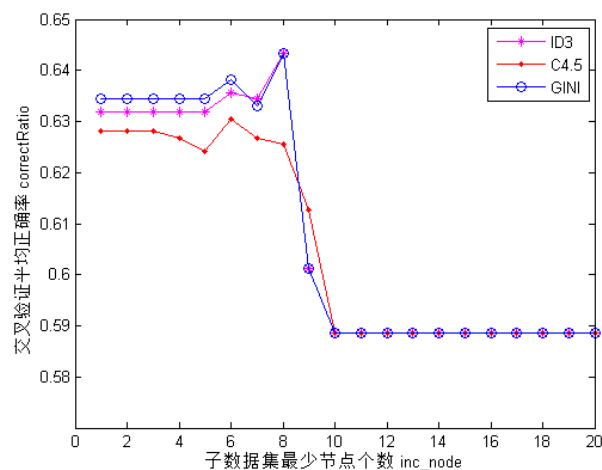
没有优化的时候: 交叉验证得到的平均准确率:

ID3: 0.631847133757962

C4.5: 0.628025477707006

GINI: 0.634394904458599

● 考虑优化, 设置结点样本数小于一定阈值结束递归:



分析: **GINI 指数**总体表现最好, **阈值设为 8** 的时候交叉验证的平均准确率最高 0.6433。

四、 思考题

1. 决策树有哪些避免过拟合的方法?

1) 前剪枝

● 定义一个高度, 当决策树达到该高度时就停止分裂;

- 定义一个阈值，使得当达到某个节点的实例个数小于阈值时，就可以停止分裂；
假设选择了某个特征进行划分。如果划分后，决策树在验证集上的准确率不提高，则无需分裂；

2) 后剪枝：

- 先构建一个完整的决策树（不考虑过拟合的问题）；
 - 然后对一些结点的子树用叶子结点来代替，使其成为叶子结点，该叶子的标签为该结点的子树中标签的众数；
 - 计算剪枝后的决策树的性能比原来的树更好时，才真正删除该结点。
- 判断性能可以利用误分类率，或者是融入模型复杂度。

2、C4.5 相比于 ID3 的优点是什么？

1) ID3 算法偏向于选择属性取值个数多的特征。前面在算法原理中证明了为什么。

ID3 算法偏向于选择属性取值个数多的特征。

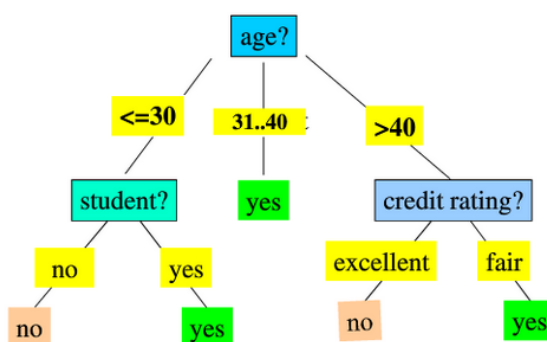
2) 根据上面那一条，可以推出 ID3 不能直接处理连续型特征。

举一个极端的例子，如果数据集所有的特征都是连续型特征，并且每一个特征的特征值都值出现了一次，那么所有特征的条件熵都等于零，他们的信息增益就为 $H(D)$ ，无法选出最“重要的”属性。

而 C4.5 算法引入了特征自己本身的熵，可以处理连续型特征。

3、如何用决策树来判断特征的重要性？

以前面出现的 一个决策树为例：



可以根据特征在决策树的层次来判断，因为越先进行分类的特征越重要。

结论：处于决策树层次越小的特征，重要性越大。

根结点 age 的层次为 1，该特征最重要。