C4.5算法

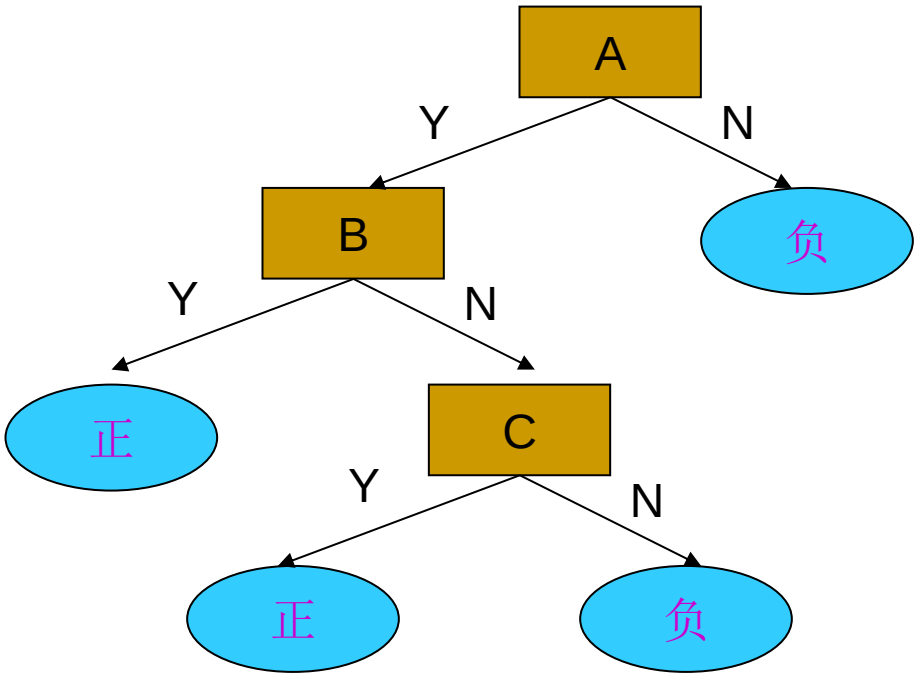
- 算法步骤
  - 过程二：剪枝
    - 错误率降低剪枝(Reduced-Error Pruning ,REP)
      - 数据量足够时，使用与训练样例截然不同验证样本，来评估通过后修剪方法从树上修剪节点的效用
    - 悲观剪枝(Pessimistic Error Pruning ,PEP)
      - 数据量有限时，使用所有可用的数据进行训练，但进行统计测试来估计修剪一个特定的节点是否有可能改善在训练集合外的实例上的性能

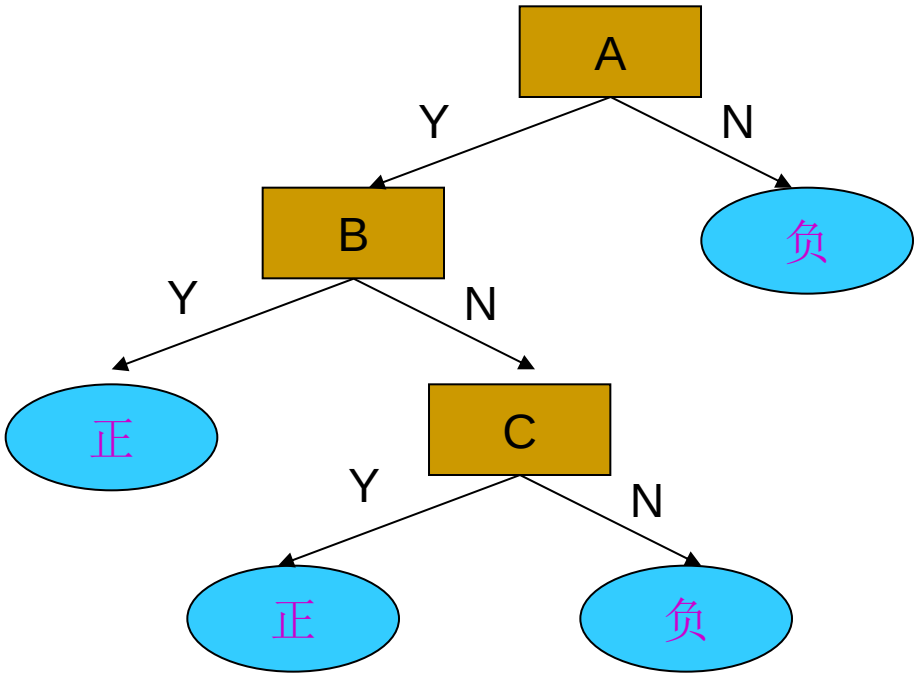
错误率降低剪枝(REP)

- 从训练集合推导出决策树，增长决策树直到尽可能好地拟合训练数据，允许过度拟合发生
- 将决策树转化为等价的规则集合，方法是为从根节点到叶节点的每一条路径创建一条规则
- 通过删除任何能导致估计精度提高的前件来修剪每一条规则
- 按照修剪过的规则的估计精度对它们进行排序，并按这样的顺序应用这些规则来分类后来的实例

例子

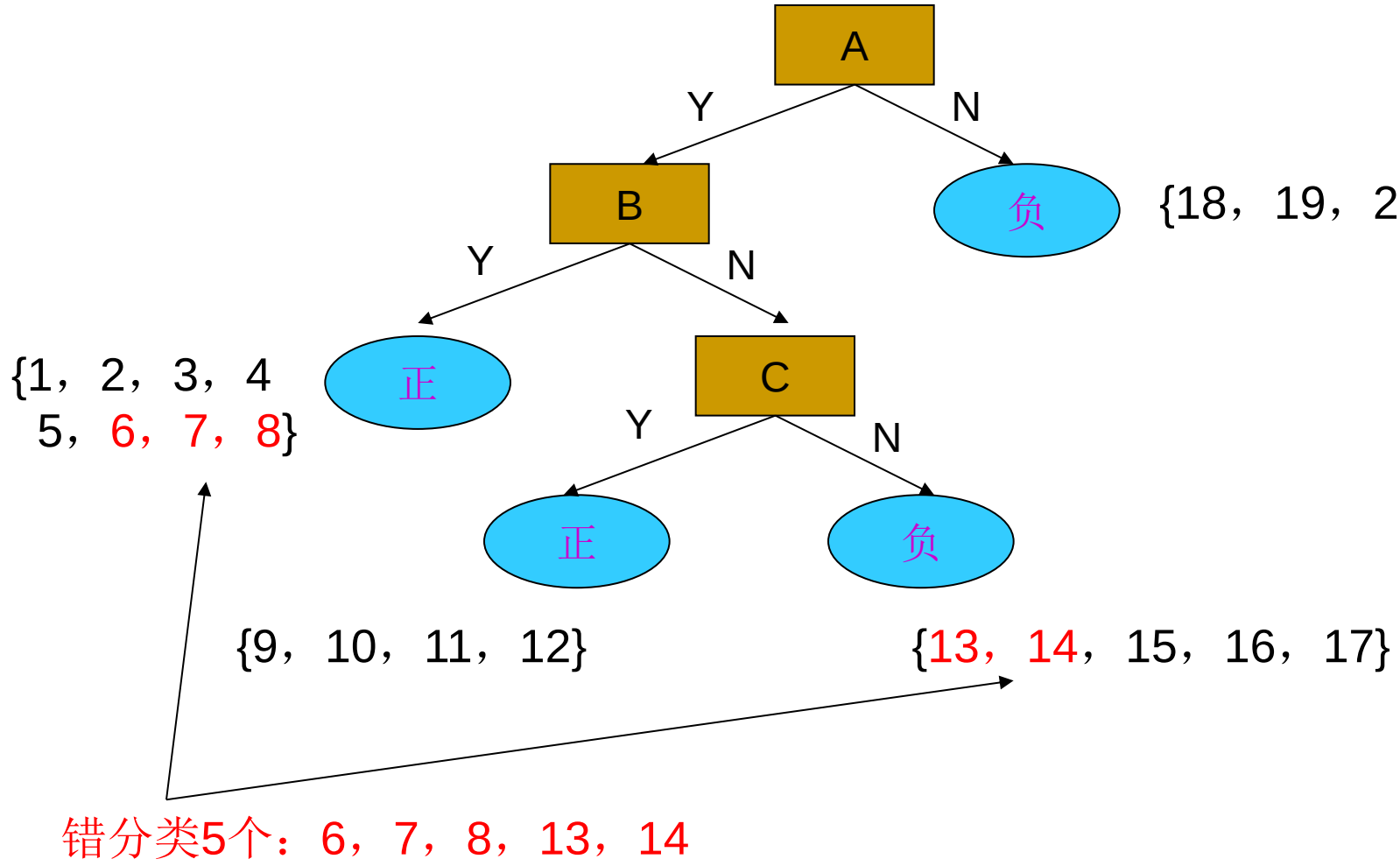
一棵通过
训练集合
学好的决策树



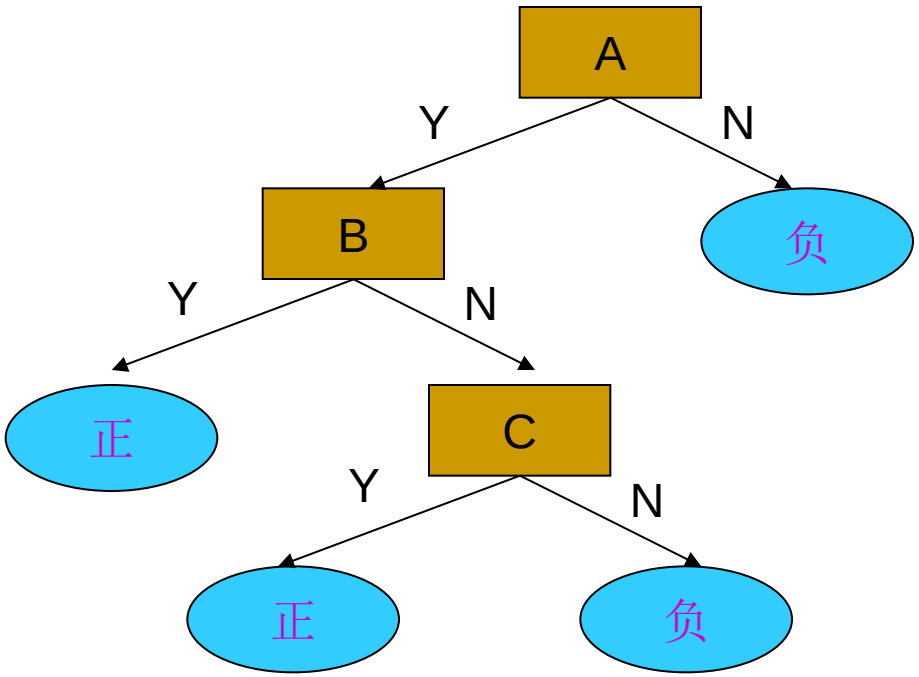


对以上的决策树通过右侧的验证集合进行测试，发现其中有5个错分类。

实例	A	B	C	类别	错分类
1	Y	Y	Y	+	
2	Y	Y	Y	+	
3	Y	Y	Y	+	
4	Y	Y	Y	+	
5	Y	Y	Y	+	
6	Y	Y	N	-	*
7	Y	Y	N	-	*
8	Y	Y	N	-	*
9	Y	N	Y	+	
10	Y	N	Y	+	
11	Y	N	Y	+	
12	Y	N	Y	+	
13	Y	N	N	+	*
14	Y	N	N	+	*
15	Y	N	N	-	
16	Y	N	N	-	
17	Y	N	N	-	
18	N	N	N	-	
19	N	Y	N	-	
20	N	Y	Y	-	


A
Y
N
B
负
{18，19，20}
Y
N
{1，2，3，4
5，6，7，8}
正
C
Y
N
正
负
{9，10，11，12}
{13，14，15，16，17}
错分类5个：6，7，8，13，14

第1步 将决策树规则化



规则1 IF A=Y AND B=Y
THEN +

规则2 IF A=Y AND B=N AND C=Y
THEN +

规则3 IF A=Y AND B=N AND C=N
THEN –

规则4 IF A=N
THEN -

第2步 规则精度的计算

规则1 IF A=Y AND B=Y
THEN +

规则2 IF A=Y AND B=N AND C=Y
THEN +

规则3 IF A=Y AND B=N AND C=N
THEN –

规则4 IF A=N
THEN -

规则	分类正确的数目	分类错误的数目	精度
1	5	3	5/8
2	4	0	4/4
3	3	2	3/5
4	3	0	3/3

第3步 对规则进行修剪

规则1 IF A=Y AND B=Y
THEN +

规则3 IF A=Y AND B=N AND C=N
THEN –

规则	分类正确的数目	分类错误的数目	精度
1	5	3	5/8
2	4	0	4/4
3	3	2	3/5
4	3	0	3/3

规则2与规则4精度为100%，保留

规则	去掉A	去掉B	去掉C	去掉AB	去掉BC	去掉AC	选择
1	5/10	11/17					去掉B
3	4/6	6/8	3/9	8/10	6/17	4/10	去掉AB

第4步 最终规则集合为

规则1 IF A=Y AND B=Y
THEN +
规则2 IF A=Y AND B=N AND C=Y
THEN +
规则3 IF A=Y AND B=N AND C=N
THEN –
规则4 IF A=N
THEN -

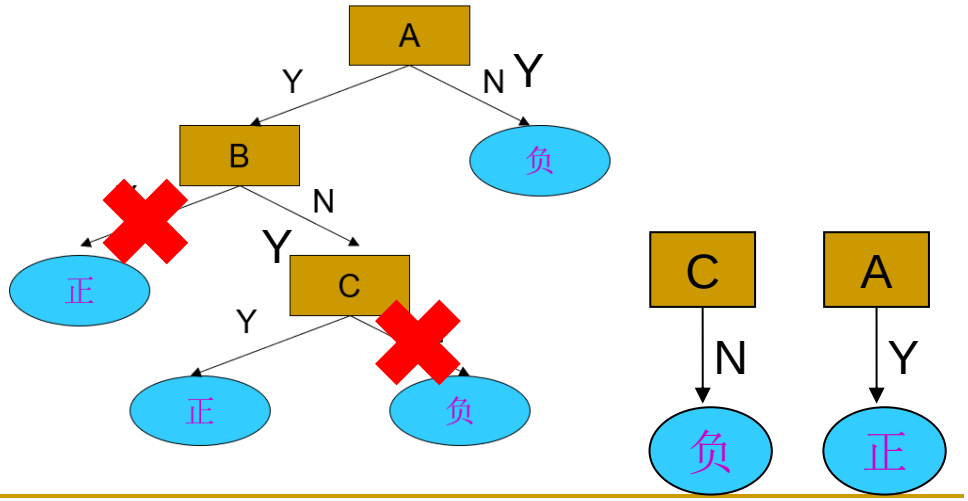
原始规则集合

规则1 IF A=Y
THEN + （11/17）
规则2 IF A=Y AND B=N AND C=Y
THEN + (4/4)
规则3 IF C=N
THEN – (8/10)
规则4 IF A=N
THEN - (3/3)

剪枝后的规则集合

规则1 IF A=Y AND B=N AND C=Y
THEN + (4/4)
规则2 IF A=N
THEN - (3/3)
规则3 IF C=N
THEN – (8/10)
规则4 IF A=Y
THEN + （11/17）

重新排序后的规则集合



第5步 根据精度和泛化能力对对规则进行排序

IF A=Y AND B=N AND C=Y THEN +： {9,10,11,12}

IF A=N THEN - {18，19，20}

IF C=N THEN – ：{6,7,8,13,14,15,16,17}

IF A=Y THEN +：{1,2,3,4,5}

尽管{13，14}仍然被错分，但整个模型的精度提高了

悲观剪枝(PEP)

- 完全使用训练数据来生成决策树，又用这些训练数据来完成剪枝
  - 过程
    - 与REP类似，差别在于规则精度的计算原理和方法上

悲观剪枝(PEP)

- 原理
  - 把一颗子树（具有多个叶子节点）的分类用一个叶子节点来替代的话，在训练集上的误判率肯定是上升的，但是在新数据上不一定。
  - 对于一颗叶子节点，它覆盖了N个样本，其中有E个错误，那么该叶子节点的错误率为

$$(E+0.5)\ /N$$

这个0.5就是惩罚因子

悲观剪枝(PEP)

- 原理
  - 一颗子树，它有L个叶子节点，那么该子树的误判率估计为

$$\frac{\sum E_i + 0.5 * L}{\sum N_i}$$

  - 剪枝后内部节点变成了叶子节点，其误判个数J也需要加上一个惩罚因子，即

J+0.5

悲观剪枝(PEP)

- 原理
  - 一棵树错误分类一个样本值为1，正确分类一个样本值为0，该树错误分类的概率（误判率）为e，则e的计算公式为：

$$e = \frac{\sum E_i + 0.5 * L}{\sum N_i}$$

  - 那么树的误判次数就是伯努利分布，可以估计出该树的误判次数均值和标准差：

$$E(subtree\_err\_count) = N * e$$

$$var(subtree\_err\_count) = \sqrt{N * e * (1 - e)}$$

悲观剪枝(PEP)

- 原理
  - 把子树替换成叶子节点后，该叶子的误判次数也是一个伯努利分布，其概率误判率e计算公式如下：

(E+0.5)/N

  - 叶子节点的误判次数均值为：

$$E(leaf\_err\_count) = N * e$$

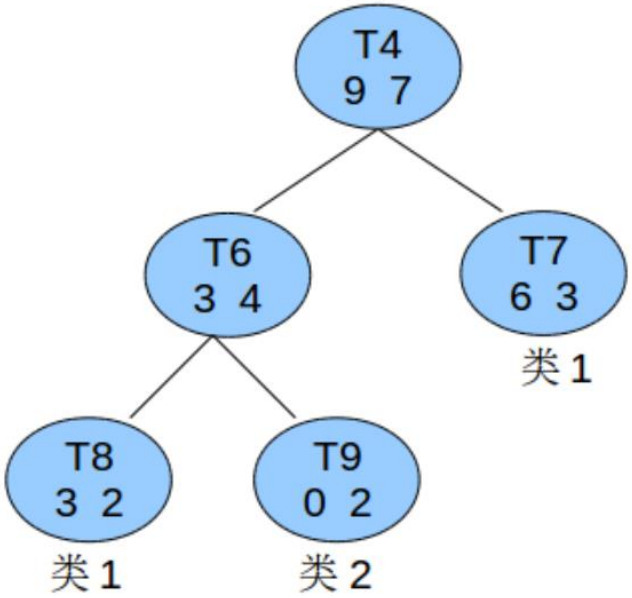
悲观剪枝(PEP)

- 原理
  - 剪枝的条件

$$E(subtree\_err\_count) + var(subtree\_err\_count) > E(leaf\_err\_count)$$

悲观剪枝(PEP)

- 例子



比如T4这棵子树的误差率：$\frac{\sum E_i+0.5*L}{\sum N_i}$

$$\frac{(3+2)+0.5*3}{16}=\frac{6.5}{16}=0.40625$$

子树误差率的标准误差：$E(subtree\_err\_count)=N*e$

$var(subtree\_err\_count)=\sqrt{N*e*(1-e)}$

$$\sqrt{16*0.40625*(1-0.40625)}=1.96$$

子树替换为一个叶节点后，其误差率为：(E+0.5)/N

$$\frac{7+0.5}{16}=0.46875$$

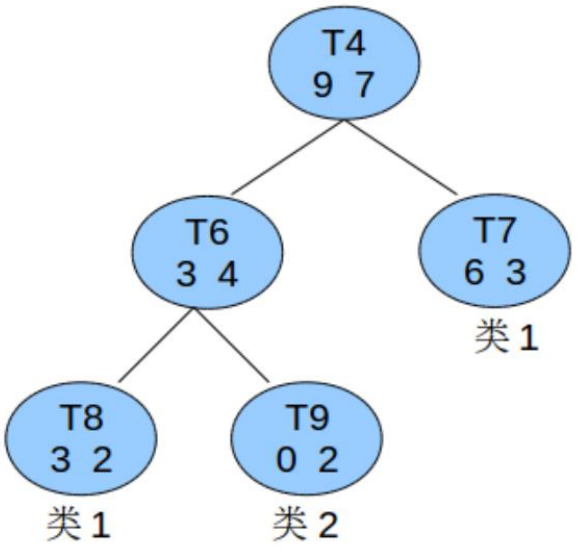
悲观剪枝(PEP)

- 例子

$$E(subtree\_err\_count) = N * e$$

$$var(subtree\_err\_count) = \sqrt{N * e * (1 - e)}$$

$$E(leaf\_err\_count) = N * e$$



$$E(subtree\_err\_count) + var(subtree\_err\_count) > E(leaf\_err\_count)$$

因为

$$16 * 0.40625 + 1.96 = 6.5 + 1.96 = 8.46 > 16 * 0.46875 = 7.5$$

所以

应该把T4的所有子节点全部剪掉，T4变成一个叶子节点。