决策树模型与实现

王君銘 吴艳辉

厦门大学

1 决策树的基本思想

决策树的结构就像一棵树，它利用树的结构将数据记录进行分类, 树的一个叶结点就代表某个条件下的一个记录集, 根据记录字段的不同取值建立树的分支；在每个分支子集中重复建立下层结点和分支, 便可生成一棵决策树。利用决策树进行分析, 可以容易地找到一些具有商业价值的潜在的规则信息。

2 决策树的技术难点

建决策树, 就是根据记录字段的不同取值建立树的分支, 以及在每个分支子集中重复建立下层结点和分支。建决策树的关键在于建立分支时对记录字段不同取值的选择。选择不同的字段值, 会使划分出来的记录子集不同, 影响决策树生长的快慢以及决策树结构的好坏, 从而导致找到的规则信息的优劣。决策树算法的技术难点也就是选择一个好的分支取值。利用一个好的取值来产生分支, 不但可以加快决策树的生长, 而且最重要的是, 产生的决策树结构好, 可以找到较好的规则信息。相反, 如果根据一个差的取值来产生分支, 不但减慢决策树的生长速度, 而且会使产生的决策树分支过细, 结构性差, 从而难以发现一些本来可以找到的有用的规则信息。

3 决策树的优化算法

决策树的优化是决策树学习算法中十分重要的分支，本文介绍以 ID3 为基础，改进的决策树优化算法。每当选择一个新的属性时，算法不是仅仅

考虑该属性带来的信息增益，而是考虑到选择该属性后继续选择的属性带来的信息增益，即同时考虑树的两层结点。本文介绍的改进算法的时间复杂性与 ID3 相同，对于逻辑表达式的归纳，改进算法则明显优于 ID3。ID3 是基于信息熵的决策树分类算法, 根据属性集的取值选择实例的类别,ID3 的算法核心是在决策树中各级结点上选择属性, 用信息增益率作为属性选择标准, 使得在每一非叶结点进行测试时, 能获得关于被测试例子最大的类别信息, 使用该属性将例子集分成子集后, 系统的熵值最小, 期望该非叶结点到达各后代叶结点的平均路径最短, 使生成的决策树平均深度较小, 提高分类速度和准确率.

ID3 的基本原理如下: 设 $E = F_1 \times F_2 \times \ldots \times F_n$ 是 n 维有穷向量空间, 其中 F_j 是有穷离散符号集,E 中的元素 $e = \langle v_1, v_2, \ldots, v_n \rangle$ 叫作例子, 其中 $v_j \in F_j, j = 1, 2, ..., n$. 设 PE 和 NE 是 E 的两个例子集, 分别叫作正例集和反例集.

假设向量空间 E 中的正例集 PE 和反例集 NE 的大小分别为 P 和 n,ID3 基于下列两个假设:(1) 在向量空间 E 上的一棵正确决策树对任意例子的分类概率同 E 中正反例的概率一致;(2) 一棵决策树能对一例子作出正确类别判断所需的信息量为

$$I(p, n) = -\frac{p}{p+n} \log_2 \frac{p}{p+n} - \log_2 \frac{n}{p+n} \tag{1}$$

如果以属性 A 作为决策树的根,A 具有 v 个值 $\langle v_1, v_2, \ldots, v_n \rangle$, 它将 E 分为 $v\langle E_1, E_2, \ldots, E_v \rangle$ 假设 E_i 中含有 p_i 个正例和 n_i 个反例, 子集 E_i 的信息熵为 $I(p_i, n_i)$, 以属性 A 为根分类后的信息熵为 $E(A) = \sum_{i=1}^{v} \frac{p_i+n_i}{p+n} I(p_i + n_i)$, 因此, 以 A 为根的信息增益是 $gain(A) = I(p, n) - E(A)$。ID3 选择使 gain（A）最大（即 E（A）最小）的属性 A^* 作为根结点. 对 A^* 的不同取值对应的 E 的 v 个子集 E_i 递归调用上述过程, 生成 A^* 的子结点 $B_1, B_2, \ldots, B_v$. ID3 的基本原理是基于两类分类问题, 但很容易将其扩展到多类, 设样本集 S 共有 C 类样本, 每类样本数为 $P_i, i = 1, 2, ...C$. 如果以属性 A 作为决策树的根,A 具有 v 个值 $v_1, v_2, \ldots, v_n$, 它将 E 分为 v 个子集 $\{E_1, E_2, \ldots, E_v\}$, 假设 E_i 中含有第 j 类样本的个数为 $P_i j, j = 1, 2, ..., C$, 那么, 子集 E_i 的信息量是 $I(E_i)$,

$$I(E_i) = \sum_{j=1}^{C} -\frac{P_{ij}}{|E_i|} * \log \frac{P_{ij}}{|E_i|} \tag{2}$$

以 A 为根分类后的信息熵为

$$E(A) = \sum_{i=1}^{v} \frac{|E_i|}{|E|} * I(E_i) \tag{3}$$

选择属性 A^* 使 E(A) 最小, 信息增益也将最大.

优化算法的基本思路为: 首先用 ID3 选择属性 F_1, 建立树 T_1, 左右子树的属性分别为 F_2, F_3;F_2, F_3, 重建树 T_2, T_3; 比较 T_1, T_2, F_3 的结点个数, 选择结点最少的树, 对于选定树的子结点采用同样的方法, 递归建树.

ID3 选择属性 A 作为新的属性的原则是,A 使得 E(A) 最大, 这种启发式方法存在一个弊端, 即算法往往偏向于选择属性取值较多的属性, 而属性值较多的属性却不总是最优的属性. ID3 学习简单的逻辑表达式能力较差。本文介绍针对这一问题提出的如下的改进方案, 设 A 为候选的属性,A 有 V 个属性值, 对应的概率分别为 $P_1, P_2, ...P_v$, 按照最小信息熵原理对属性 A 扩展,$B_1, B_2, ..., B_v$ 为 v 个子结点选择的属性, 分别对应的信息熵为 $E(B_1), E(B_2), ...E(B_v)$, 则

$$E'(A) = \sum_{i}^{v} P_i * E(B_i) \tag{4}$$

算法选择属性 A^* 的标准是 A^* 使得 $E'(A)$ 最小。算法的详细步骤如下:

(1) 对任意未选择的属性 A, 假设 A 有 v 个属性值, 对应的概率分别是 $P_1, P_2, ...P_v$, 以属性 A 扩展, 生成 v 个子结点 $B_1, B_2, ..., B_v$, B_i 是属性 A 取第 i 个值时, 按照最小信息熵原理选择的 A 的后继属性, 分别对应的信息熵为 $E(B_1), E(B_2), ...E(B_v)$。

(2) 根据公式, 计算 $E'(A)$。

(3) 选择 A^* 使得 $E'(A)$ 最小, 将 A^* 作为新选的属性。

(4) 利用步骤 (1) 的计算结果, 建立结点 A^* 的后继结点 $B_1, B_2, ..., B_v$。

(5) 对所有的 B_i, 若为叶结点, 则停止扩展此结点, 否则递归执行 (1)-(5) 的过程。

该算法 (1) — (4) 是选定 A 作为新的属性, 与 ID3 相比, 不是计算选择 A 后带来的信息增益, 而是继续选择 A 的后继属性, 计算系统的熵值, 也就是说, 该算法改进了选择新属性的启发式函数, 以达到更好的分类效果. MID3 的时间复杂性与 ID3 相同, 两者的差别只在于属性选择的计算上, 即 ID3 采用公式 (3) 选择属性, 而 MID3 利用公式 (4), 假设类别个数为 m, 属性个数为 n, 属性值的平均个数为 v,$v \ll n$, 公式 (2) 的时间复杂性为 $O(m)$,

则公式 (3) 计算每个属性 A 的时间复杂性为 $O(m*v)$, 所以计算所有的属性的时间为 $O(n*m*v)$, 这就是 ID3 选择一个结点的属性的时间开销。比较公式 (3) 和 (4), 很容易看出 MID3 选择一个结点的属性的时间复杂性为 $O(v*n*m*v)$, 由于 $v \ll n, O(v*n*m*v) = O(n*m*v) = O(m*n)$, 即 ID3 与 MID3 在属性选择上时间复杂性相同。

4 决策树模型的 R 代码实现

```
set.seed (10)
 n = 1500
 x = matrix(rnorm(n*2),ncol=2)
 z = 2*x[,1] + 2*x[,2]
 p = exp(z)/(1+exp(z)) #p(y=1)
 u = runif(n)
 y = (p>u)

# create training and test set
 data = data.frame(x1=x[,1],x2=x[,2],y=as.factor(y))
 train = sample(n,n*0.4)
 data_train = data[train,]
 data_test = data[-train,]
 ytrue = data_test[,"y"]
 plot(data_train$x1,data_train$x2,col=data_train$y)
```

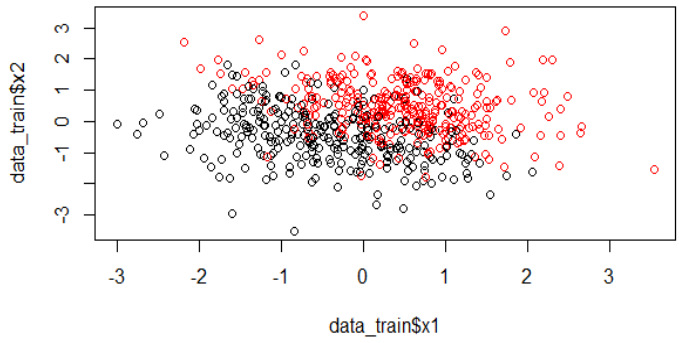

图 1: 数据分布图

```
# Classification Tree #
install.packages("rpart.plot")
set.seed(200)
fit0 = rpart(y~.,data_train,control=rpart.control(cp=0))
fit = prune(fit0,cp=fit0$cptable[which.min(fit0$cptable[,"xerror"]),"CP"])
rpart.plot(fit,box.palette=c("Reds","Blues"))

# test err
yhat = predict(fit,data_test,type="class")
err = 1-mean(ytrue==yhat)
err
[1] 0.2277778
```

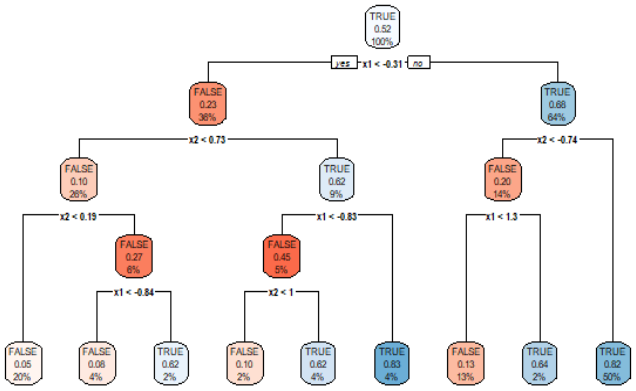

图 2: 决策树结果图

参考文献

[1] Moore AW, Papagiannaki K. Toward the accurate identification of network applications. In: Dovrolis C, ed. Proc. of the PAM 2005.LNCS 3431, Heidelberg: Springer-Verlag, 2005：41-54.

[2] 徐鹏，林森. 基于 C4.5 决策树的流量分类方法 [J] 软件学报，20（10）：2693-2700.