多元统计 10

罗震林 17306071

1 Question 9.2

1.1 树的构造

```r
library(rpart)
library(rpart.plot)

cleveland <- read.table("cleveland.txt", header = TRUE,stringsAsFactors = T)
cleveland <- cleveland[,-15]
out <- rpart(diag~ .,data = cleveland)
out
rpart.plot(out,type = 1,extra = 1,roundint = FALSE)
```

```
n= 296

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 296 136 buff (0.54054054 0.45945946)
   2) thal=norm 163  36 buff (0.77914110 0.22085890)
     4) ca< 0.5 114  12 buff (0.89473684 0.10526316) *
     5) ca>=0.5 49  24 buff (0.51020408 0.48979592)
      10) cp=abnang,angina,notang 29   7 buff (0.75862069 0.24137931) *
      11) cp=asympt 20   3 sick (0.15000000 0.85000000) *
   3) thal=fix,rev 133  33 sick (0.24812030 0.75187970)
     6) cp=abnang,angina,notang 44  21 buff (0.52272727 0.47727273)
      12) ca< 0.5 27   8 buff (0.70370370 0.29629630) *
      13) ca>=0.5 17   4 sick (0.23529412 0.76470588) *
     7) cp=asympt 89  10 sick (0.11235955 0.88764045)
      14) oldpeak< 0.55 21   8 sick (0.38095238 0.61904762)
        28) thatach< 149 7   2 buff (0.71428571 0.28571429) *
        29) thatach>=149 14   3 sick (0.21428571 0.78571429) *
      15) oldpeak>=0.55 68   2 sick (0.02941176 0.97058824) *
```

图 1: 输出结果

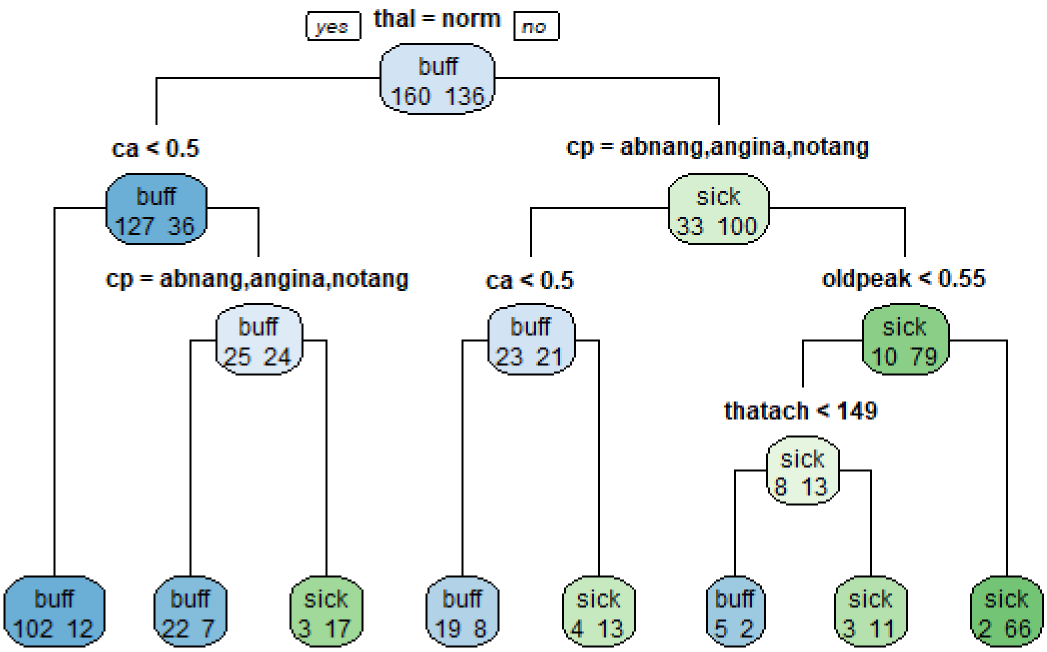

图 2: cleveland 分类树

使用 Gini 指数为原则的构造的树的结果见上，与书上使用熵的结果存在差异。相同点是都是以 tahl 变量作为第一次分裂的条件，且树的深度都为 5；不同点是在变量的选择上，Gini 指数构造的树没有使用 age 和 exang 两个变量，且分裂的位置有所不同

- 误判率

```
prediction <- predict(out,data=cleveland,type = "class")
table(cleveland$diag,prediction)
mean(cleveland$diag!=prediction)
```

表 1: 预测结果

实际 \ 预测	buff	sick
buff	148	12
sick	29	107

回代结果见表1，可以计算得到回代的误判率为 $41/296 = 0.1385135$，其中有 12 个 buff 被误判为 sick，29 个 sick 被误判为 buff。误判率高于熵构造的树

1.2 age 的最佳分割

```
Age <- sort(cleveland$age)
L_Buff <- c(); L_Sick <- c();B_Buff <- c();B_Sick <- c() # initialize
for (i in Age) {
  L_Buff <- c(L_Buff,sum(cleveland$diag[cleveland$age <= i]=='buff'))
  L_Sick <- c(L_Sick,sum(cleveland$diag[cleveland$age <= i]=='sick'))
  B_Buff <- c(B_Buff,sum(cleveland$diag[cleveland$age > i]=='buff'))
  B_Sick <- c(B_Sick,sum(cleveland$diag[cleveland$age > i]=='sick'))
} # storage all situations of split

tao_l <- 1 - (L_Buff/(L_Buff+L_Sick))^2 - (L_Sick/(L_Buff+L_Sick))^2 # impurity of left node
tao_r <- 1 - (B_Buff/(B_Buff+B_Sick))^2 - (B_Sick/(B_Buff+B_Sick))^2 # impurity of right node
tao <- 1-((L_Buff+B_Buff)/nrow(cleveland))^2 -
    ((L_Sick+B_Sick)/nrow(cleveland))^2 # impurity of parent node
delta <- tao - tao_l*((L_Buff+L_Sick)/nrow(cleveland)) -
    tao_r*((B_Buff+B_Sick)/nrow(cleveland)) # goodness-of-split

## plot
par(mfrow=c(1,2))
plot(Age,tao_l, type="l", col="blue",
     xlab="Age at split", ylab = expression(i(tau)))
points(Age,tao_r, type="l", col="red")
legend("bottom",legend = c("Left","Right"),
       lty = 1,col=c("blue","red"),
       )

plot(Age,delta,type="l", col="red",
     xlab="Age at split", ylab = "Goodness of split")

Age[which.max(delta)]
```

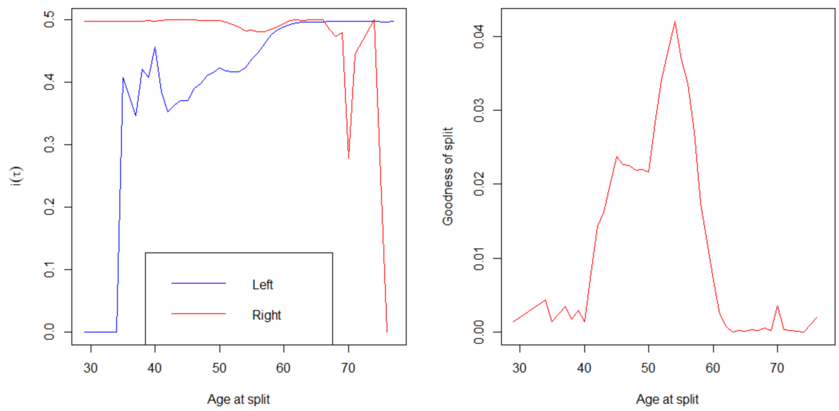

图 3: $i(\tau_L)$（蓝色）和 $i(\tau_R)$（红色）（左）和最佳分割 $\Delta i(s,\tau)$(右)

从图3的左图可以看出，$i(\tau_R)$ 在 age=70 时有一个显著下降；而从右图可以看到 goodness-of-split 最大值在 age=54，这都与书上的结果一致。综上，age 的最佳分割在 age=54 处

2 Question 9.4

记第一种分割的树为 T_1, 第二种树为 T_2，根节点为 τ, 其左右子结点分别为 τ_L 和 τ_R. 两种树的分裂情况见表2

表 2: 两种树的分裂情况

	disease	no disease	total
τ_L	300	100	400
τ_R	100	300	400
total	400	400	800

(a) 树 T_1 的分裂情况

	disease	no disease	total
τ_L	200	400	600
τ_R	200	0	200
total	400	400	800

(b) 树 T_2 的分裂情况

2.1 Resubstitution Error Rate

$$R^{re}(T_1) = \frac{1}{4} \times \frac{1}{2} + \frac{1}{4} \times \frac{1}{2} = \frac{1}{4}$$
$$R^{re}(T_2) = \frac{1}{3} \times \frac{3}{4} + 0 = \frac{1}{4}$$

所以树 T_1 和 T_2 的 Resubstitution Error Rate 相等

2.2 Goodness-of-Split

为了判断哪种分裂更适合未来树的生长，这里我们采用 good-of-split, 即 $\Delta i(s,\tau)$ 进行判断，其中 impurtiy function $i(\cdot)$ 采用 Gini diversity index

- 树 T_1

$$i(\tau) = 1 - (\frac{1}{2})^2 - (\frac{1}{2})^2 = \frac{1}{2}$$
$$i(\tau_L) = i(\tau_R) = 1 - (\frac{3}{4})^2 - (\frac{1}{4})^2 = \frac{3}{8}$$
$$\Delta i(s,\tau) = i(\tau) - \frac{1}{2}i(\tau_L) - \frac{1}{2}i(\tau_R) = \frac{1}{8}$$

- 树 T_2

$$i(\tau) = 1 - (\frac{1}{2})^2 - (\frac{1}{2})^2 = \frac{1}{2}$$
$$i(\tau_L) = 1 - (\frac{1}{3})^2 - (\frac{2}{3})^2 = \frac{4}{9}$$
$$i(\tau_R) = 1 - 1^2 = 0$$
$$\Delta i(s,\tau) = i(\tau) - \frac{3}{4}i(\tau_L) - \frac{1}{4}i(\tau_R) = \frac{1}{6}$$

树 T_1 的 goodness-of-split 小于 T_2，所以第二种分裂更适合未来树的生长

3 Question 9.8

- 读取数据，并将数据集进行随机分割，70% 的数据放入训练集，30% 放入测试集

```
library(MASS)
library(rpart)
library(rpart.plot)

vehicle <- read.table("vehicle3.txt", header = T)
vehicle <- vehicle[,-20]

# partitioning
set.seed(1234)
ind <- sample(2,nrow(vehicle),replace = T,prob = c(0.7,0.3))
traintset <- vehicle[ind==1,]
testset <- vehicle[ind==2,]
```

- 生成树并作图

```
# build the tree
ct_vehicle <- rpart(class~ .,data = traintset)
ct_vehicle
summary(ct_vehicle)

## plot
plot(ct_vehicle,uniform = T,branch = 0.6,margin = 0.1)
text(ct_vehicle,all = T,use.n = T,cex = 0.75,pos=3)
```

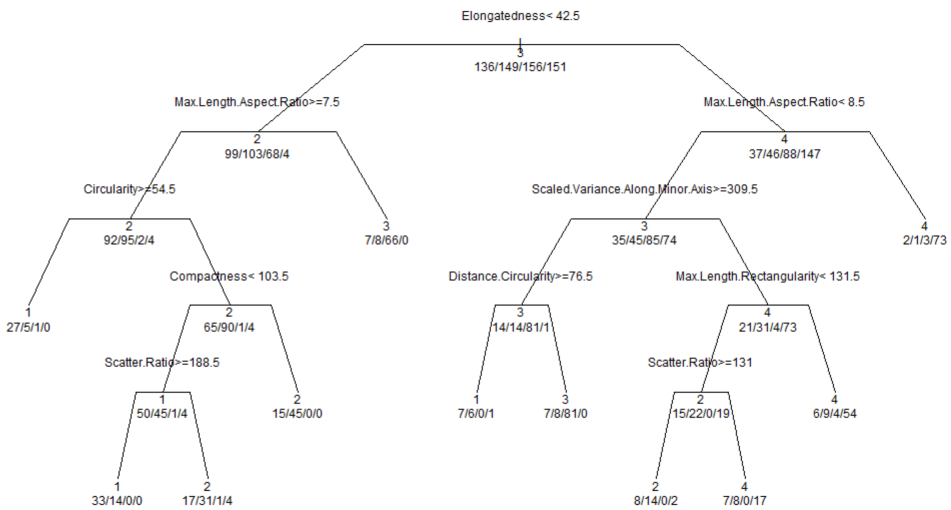

图 4: vehicle 分类树

如图4所示，生成的树一共进行了 10 次分割，深度为 5。由 rpart() 函数的默认设置可知，此时的 complexity parameter：$\alpha = 0.01$。进一步，我们可以查看不同的 CP 值对应的分割情况及误差大小

```r
## complexity parameter
printcp(ct_vehicle)
```

	CP	nsplit	rel error	xerror	xstd
1	0.215596	0	1.00000	1.05275	0.023291
2	0.133028	1	0.78440	0.81193	0.027362
3	0.091743	2	0.65138	0.71560	0.027862
4	0.050459	4	0.46789	0.52064	0.027134
5	0.021789	5	0.41743	0.45872	0.026394
6	0.016055	7	0.37385	0.46330	0.026458
7	0.013761	8	0.35780	0.46101	0.026426
8	0.010000	10	0.33028	0.44495	0.026194

图 5: CP 的不同情况

很明显，随着 CP 的减小，误差也越来越小，但同时，分割次数也越来越多，会使树很复杂

- 剪枝

根据图5，我们选择 CP=0.013761 来进行剪枝

```r
pr_cp <- ct_vehicle$cptable[,1]
prune_tree <- prune(ct_vehicle,cp=pr_cp[7])
prediction <- predict(prune_tree,testset, type = "class")
plot(prune_tree,uniform = T,branch = 0.6,margin = 0.1)
text(prune_tree,all = T, use.n = T, cex = 0.75,pos=3)
```

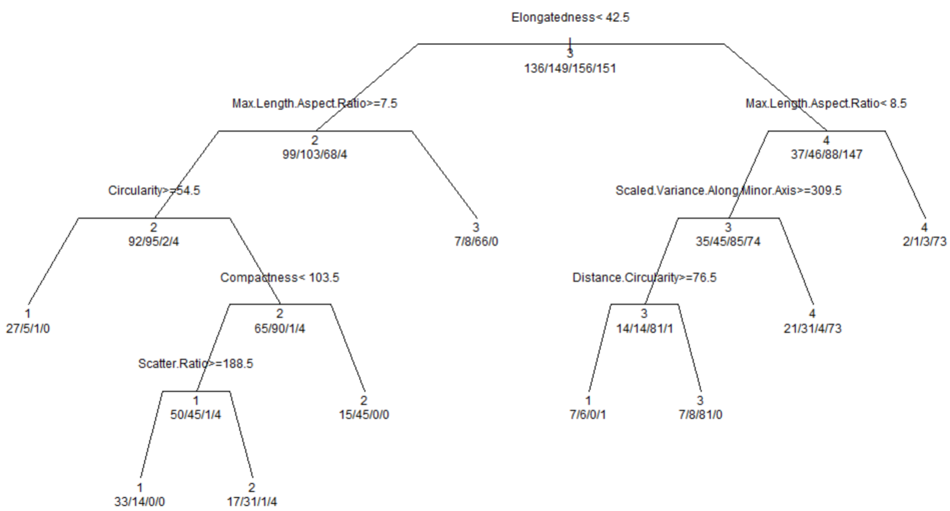

图 6: 剪枝后的树

剪枝后，生成树见图6，树的分割次数为 8，为了比较剪枝前和剪枝后的树的好坏，我们将测试集的数据代入，比较剪枝前和剪枝后的误判率

- 误判率

```
# without pruning
re <- predict(ct_vehicle, testset, type = "class")
mean(testset$class!=re)
# prune
prediction <- predict(prune_tree,testset, type = "class")
mean(testset$class!=prediction)
```

计算得到剪枝前的误判率为 0.3110236，剪枝后的误判率为 0.3188976。剪枝后误判率增加，但是只增加了 0.0007 左右，处于可接受的范围，若需要分类树更为简洁，则可以选择剪枝过后的分类树

注：还可以进一步选择更大的 CP 值进行剪枝，但误判率会增加，当 CP=0.016055 时，误判率为 0.3464567，误判率增长较大，所以不再进一步剪枝