STATS415hw9

Yunguo Cai

3/28/2018

```
library(tree)
```

```
## Warning: package 'tree' was built under R version 3.4.4
```

```
library(MASS)
attach(crabs)
```

(a)

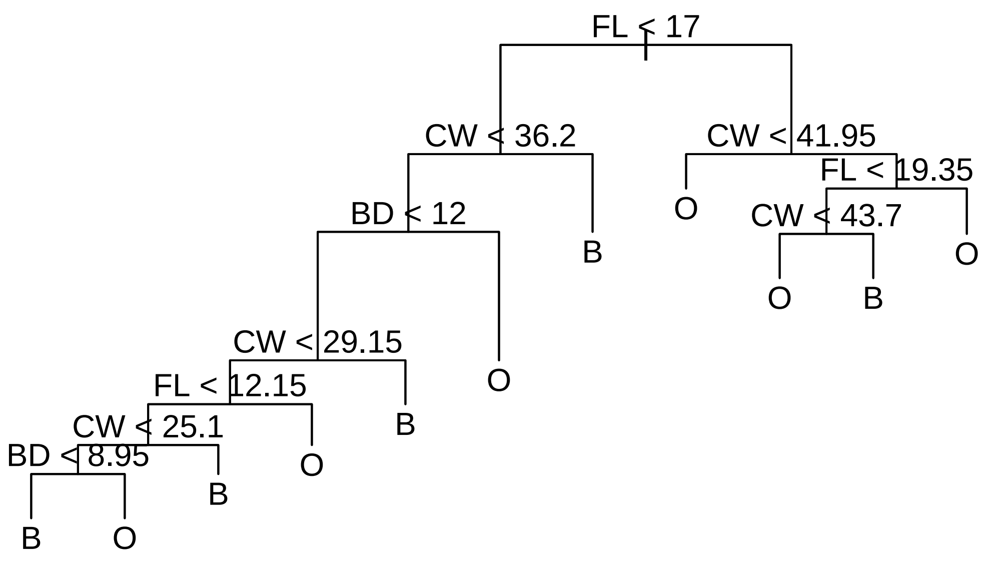
```
set.seed(45678)
crab_bm = which(crabs$sp == "B" & crabs$sex == "M")
crab_bf = which(crabs$sp == "B" & crabs$sex == "F")
crab_om = which(crabs$sp == "O" & crabs$sex == "M")
crab_of = which(crabs$sp == "O" & crabs$sex == "F")
train_id = c(sample(crab_bf, size = trunc(0.80 * length(crab_bf))),
sample(crab_bm, size = trunc(0.80 * length(crab_bm))),
sample(crab_of, size = trunc(0.80 * length(crab_of))),
sample(crab_om, size = trunc(0.80*length(crab_om))))
crab_train = crabs[train_id, ]
crab_test = crabs[-train_id, ]
```

(b)

```
tree.species=tree(sp~.-index-sp,crab_train)
set.seed(45678)
cv.species=cv.tree(tree.species,FUN = prune.misclass)
cv.species
```

```
## $size
## [1] 14 11  7  4  2  1
##
## $dev
## [1] 25 25 30 30 58 96
##
## $k
## [1]      -Inf  0.000000  2.500000  2.666667 11.000000 33.000000
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
prune.species=prune.misclass(tree.species,best=8)
plot(prune.species)
text(prune.species,pretty=0)
```



```r
summary(prune.species)
```

```
##
## Classification tree:
## snip.tree(tree = tree.species, nodes = c(5L, 9L, 15L))
## Variables actually used in tree construction:
## [1] "FL" "CW" "BD"
## Number of terminal nodes:  11
## Residual mean deviance:  0.3231 = 48.15 / 149
## Misclassification error rate: 0.04375 = 7 / 160
```

```r
tree.pred_train=predict(prune.species,crab_train,type="class")
table(tree.pred_train,crab_train$sp)
```

```
##
## tree.pred_train  B  O
##               B 75  2
##               O  5 78
```

```r
training_err=sum(tree.pred_train != crab_train$sp)/160
training_err
```

```
## [1] 0.04375
```

```r
tree.pred_test=predict(prune.species,crab_test,type="class")
table(tree.pred_test,crab_test$sp)
```

```
##
## tree.pred_test  B  O
##              B 14  0
##              O  6 20
```

```r
test_err=sum(tree.pred_test != crab_test$sp)/40
test_err
```

```
## [1] 0.15
```

From the cross validation, the best tree with no more than 8 splits is the one with 7 splits (size = 8). Only 3 variables are used in this tree, which are FL, CW, and BD. Since the first split is based on FL, it seems to be the most important predictor. The traning error is 0.04375, and the test error is 0.15.

(c)

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.4.4
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
set.seed(45678)
rf.crabs=randomForest(sp~.-index-sp,data=crabs,subset=train_id,ntree=1000,mtry=3,importance=TRUE)
rf.pred_train = predict(rf.crabs,crab_train,type="class")
rf.pred_test = predict(rf.crabs,crab_test,type="class")
table(rf.pred_train,crab_train$sp)
```

```
##
## rf.pred_train  B  O
##             B 80  0
##             O  0 80
```

```
train_err.rf=sum(rf.pred_train != crab_train$sp)/160
train_err.rf
```

```
## [1] 0
```

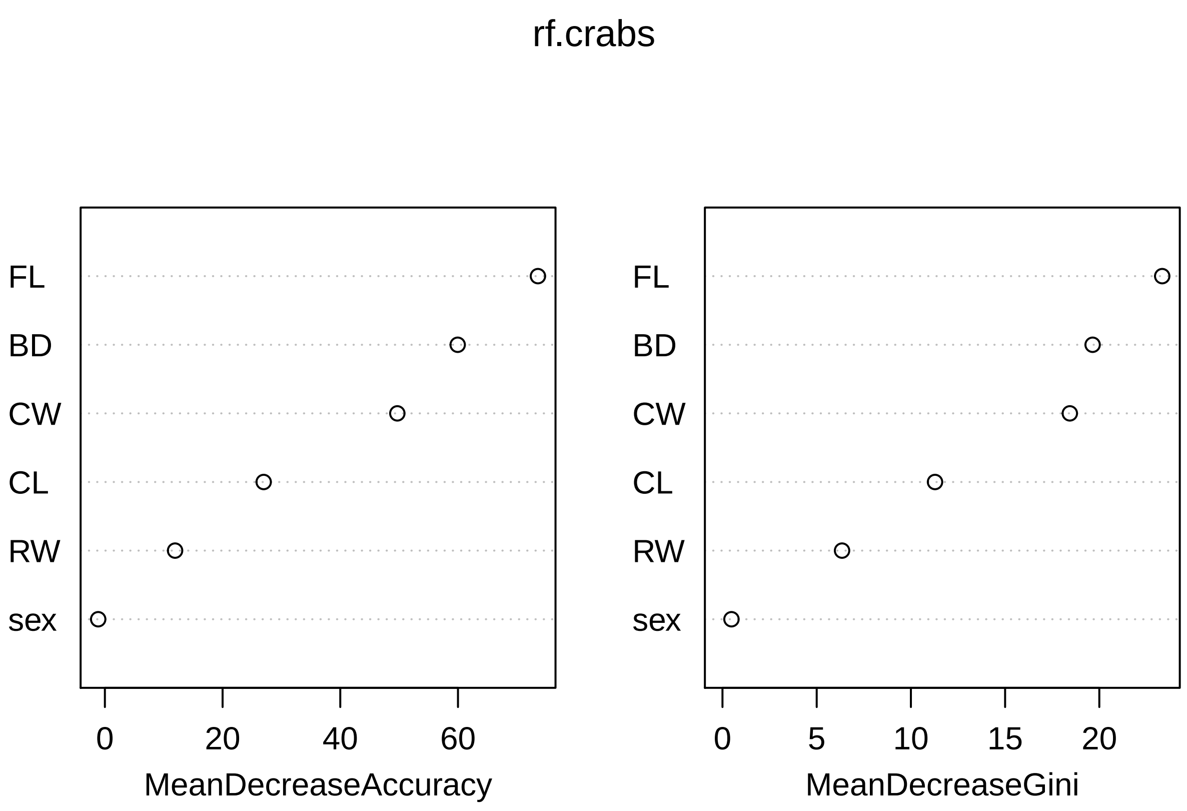
```
table(rf.pred_test,crab_test$sp)
```

```
##
## rf.pred_test  B  O
##            B 18  0
##            O  2 20
```

```
test_err.rf=sum(rf.pred_test != crab_test$sp)/40
test_err.rf
```

```
## [1] 0.05
```

```
varImpPlot(rf.crabs)
```



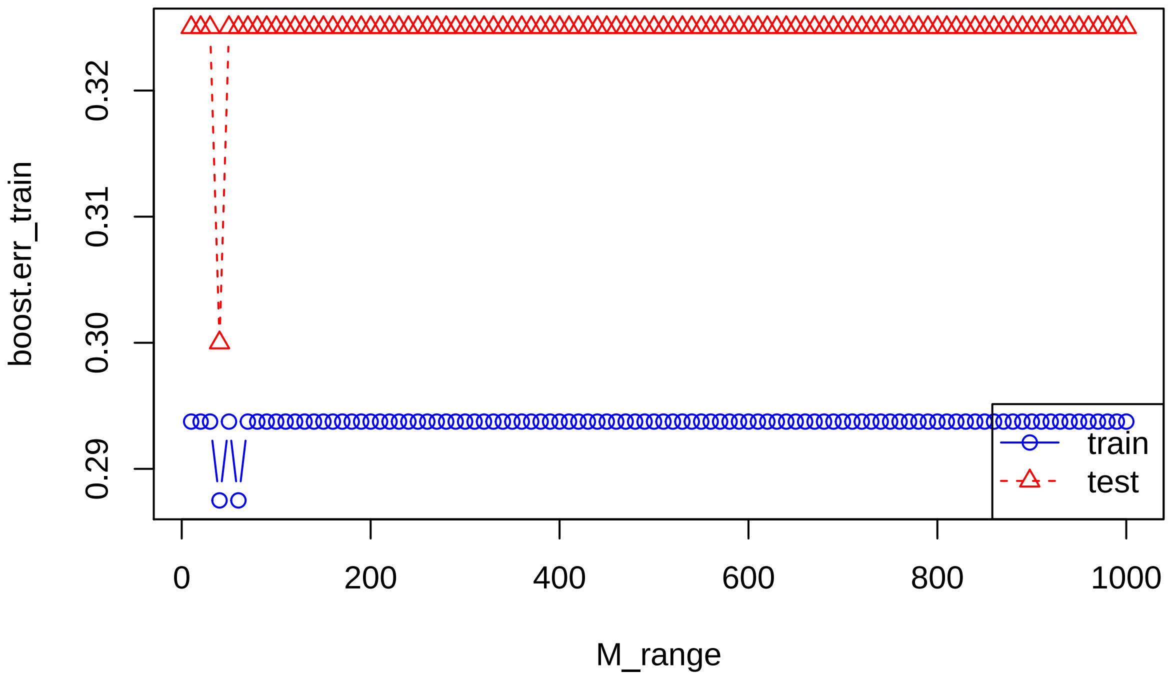
The variable importance plot shows that FL,BD and CW are the most important predictors, which is the same as the previous single tree chosen by cross validation. FL ranks first in the importance plot, while the variable used in the first split in the single tree is also FL. So FL seems to be the most important predictor. The traning error is 0, and the test error is 0.05.

(d)

```
library(gbm)
```

```
## Loading required package: survival

## Loading required package: lattice

## Loading required package: splines

## Loading required package: parallel

## Loaded gbm 2.1.3
```

```
set.seed(45678)
crabs$sp = as.numeric(crabs$sp)-1
# 0 -> blue, 1 -> orange
crab_train = crabs[train_id,]
crab_test = crabs[-train_id,]
n = 100
boost.err_train = rep(0, n)
boost.err_test = rep(0, n)
M_range = seq(from = 10, to = 1000, length.out = n)
for (i in seq(1,n)){
boost.crabs = gbm(sp ~ ., data = crab_train, distribution = "adaboost", n.trees = M_range[i])
boost.pred_train = predict(boost.crabs, newdata = crab_train, n.trees = M_range[i])
boost.pred_train = sign(boost.pred_train)
boost.pred_train[boost.pred_train==-1] = 0
boost.err_train[i] = sum(boost.pred_train != crab_train$sp)/160
boost.pred_test = predict(boost.crabs, newdata = crab_test, n.trees = M_range[i])
boost.pred_test = sign(boost.pred_test)
boost.pred_test[boost.pred_test==-1] = 0
boost.err_test[i] = sum(boost.pred_test != crab_test$sp)/40
}
plot(M_range, boost.err_train, col = "blue", lty = 1, pch = 1, type = "b",
ylim = c(min(min(boost.err_train), min(boost.err_test)), max(max(boost.err_train), max(boost.err_test))
lines(M_range, boost.err_test, col = "red", lty = 2, pch = 2, type = "b")
legend("bottomright", c("train", "test"), col = c("blue", "red"), lty = c(1, 2), pch = c(1,2))
```

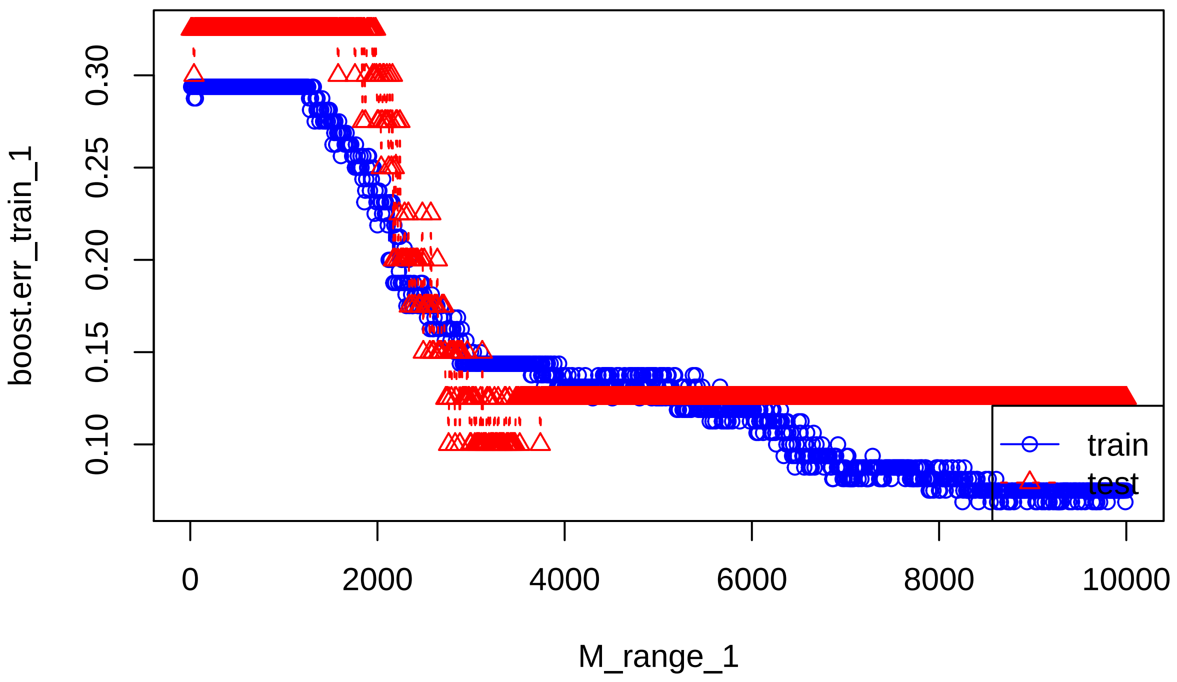


```
boost.err_train[n]
```

```
## [1] 0.29375
```

```
boost.err_test[n]
```

```
## [1] 0.325
```



```
## [1] 0.075
```

```
## [1] 0.125
```

(e) From previous results, the random forest performs best with training error 0 and test error 0.05, and it's relatively stable whatever the sample order of different color and gender combinations are. In terms of the importance of the variables, the results are consistent between the single pruned tree and the random forest tree. There's no obvious trend in the training and test error as the number of trees changes, and the training and test error of adaboost are worse than both the single pruned tree and random forest, regardless of the value of M. When I run the M range up to 10000, it becomes much better. It shows obvious decreasing trend and much lower errors. So I think the bad performance of boost is caused by the low upper bound of M, which is 1000. We can conclude that we need a large M to get more correct adaboost model.