HW8

Yigao Li

April 9, 2018

8.4 - 1

```
plot(NA, NA, type = "n", xlim = c(0, 100), ylim = c(0, 100), xlab = "X", ylab = "Y")
lines(x = c(80,80), y = c(0,100))
lines(x = c(30,30), y = c(0,100))
lines(x = c(30,80), y = c(80,80))
lines(x = c(80,100), y = c(40,40))
lines(x = c(90,90), y = c(40,100))
text(x = 80, y = 10, labels = c("t1"))
text(x = 30, y = 10, labels = c("t2"))
text(x = 100, y = 40, labels = c("t3"))
text(x = 40, y = 80, labels = c("t4"))
text(x = 90, y = 60, labels = c("t5"))
text(x = 15, y = 50, labels = c("R1"))
text(x = 55, y = 90, labels = c("R2"))
text(x = 55, y = 40, labels = c("R3"))
text(x = 85, y = 70, labels = c("R4"))
text(x = 95, y = 70, labels = c("R5"))
text(x = 90, y = 20, labels = c("R6"))
```

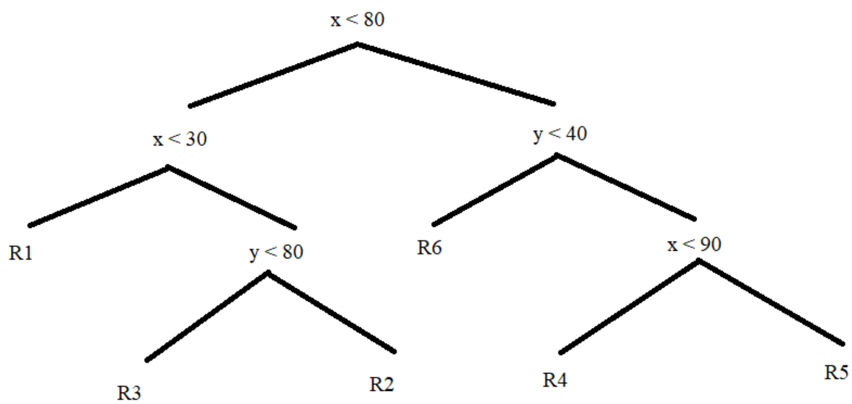
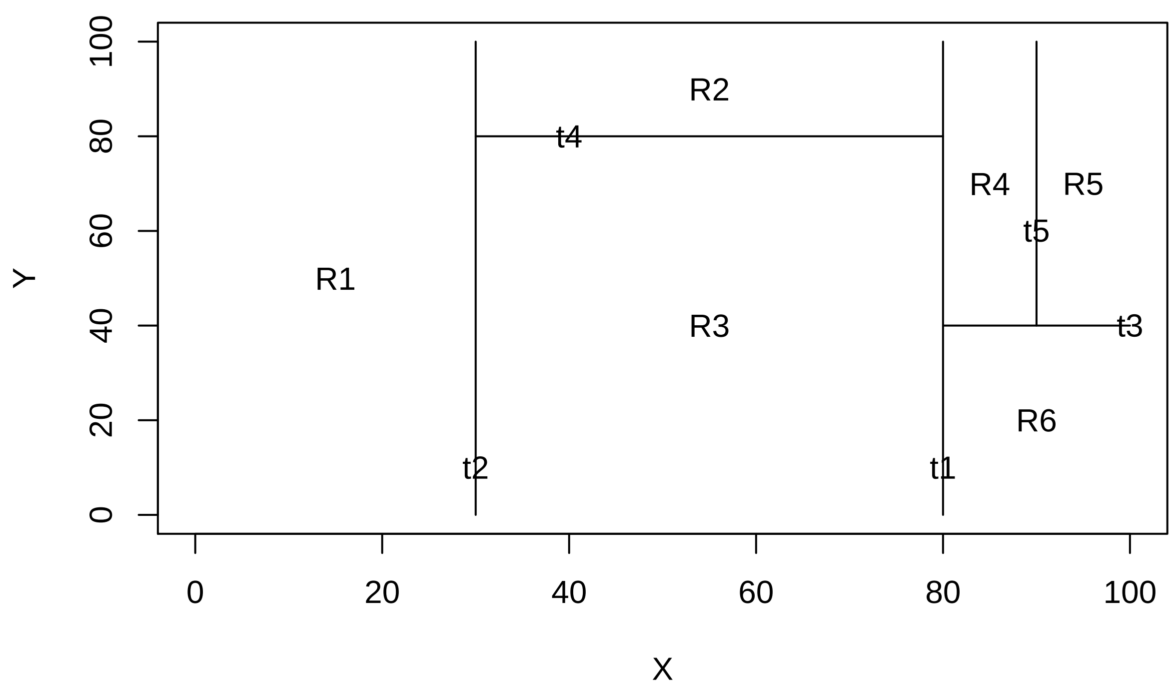


Figure 1: Decision Tree



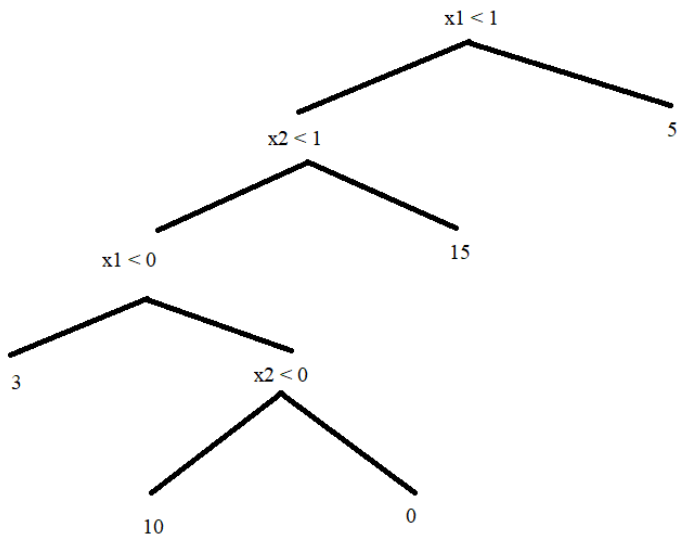


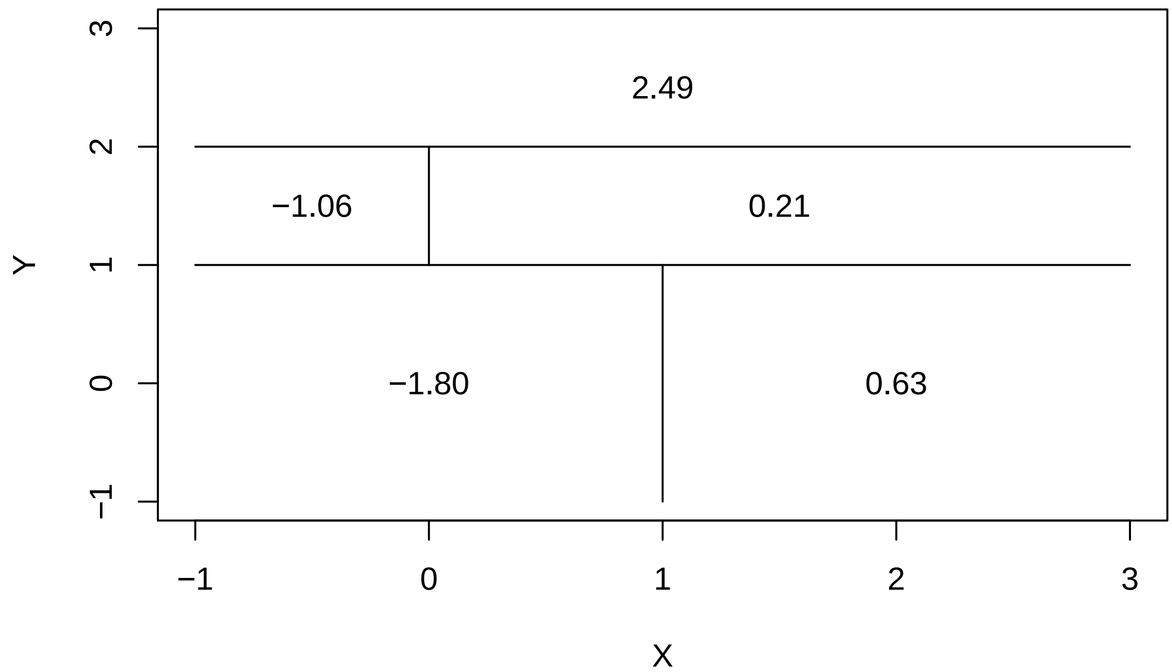
Figure 2: Tree

8.4 - 4

(a)

(b)

```
plot(NA, NA, type = "n", xlim = c(-1,3), ylim = c(-1,3), xlab = "X", ylab = "Y")
lines(x = c(-1,3), y = c(1,1))
lines(x = c(1,1), y = c(-1,1))
lines(x = c(-1,3), y = c(2,2))
lines(x = c(0,0), y = c(1,2))
text(x = 1, y = 2.5, labels = c("2.49"))
text(x = -0.5, y = 1.5, labels = c("-1.06"))
text(x = 1.5, y = 1.5, labels = c("0.21"))
text(x = 0, y = 0, labels = c("-1.80"))
text(x = 2, y = 0, labels = c("0.63"))
```



8.4 - 9

(a)

```
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 3.4.3
```

```
library(tree)
```

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

```
oj <- OJ
```

```
set.seed(1)
n <- dim(oj)[1]
train <- sample(n, 800)
oj.train <- oj[train,]
oj.test <- oj[-train,]
```

(b)

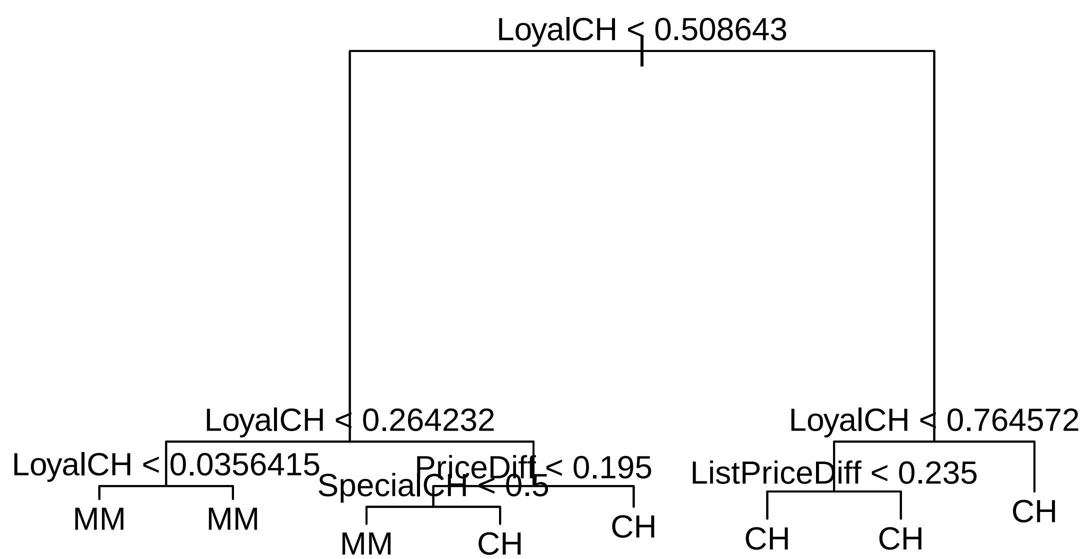
```
oj.tree <- tree(Purchase ~ ., data = oj.train)
summary(oj.tree)
```

```
##
## Classification tree:
## tree(formula = Purchase ~ ., data = oj.train)
## Variables actually used in tree construction:
## [1] "LoyalCH"       "PriceDiff"     "SpecialCH"     "ListPriceDiff"
## Number of terminal nodes:  8
## Residual mean deviance:  0.7305 = 578.6 / 792
## Misclassification error rate: 0.165 = 132 / 800
```

Decision tree uses 4 predictors and 8 terminal nodes. Training error rate is 0.165.

(d)

```r
plot(oj.tree)
text(oj.tree, pretty = 0)
```



Only "LoyalCH" predictor is considered in the first 2 levels. The bottom left nodes split by "LoyalCH" but get the same classification result.

(e)

```r
oj.pred <- predict(oj.tree, oj.test, type = "class")
table(oj.test$Purchase, oj.pred)
```

```
##      oj.pred
##        CH  MM
##    CH 147  12
##    MM  49  62
```

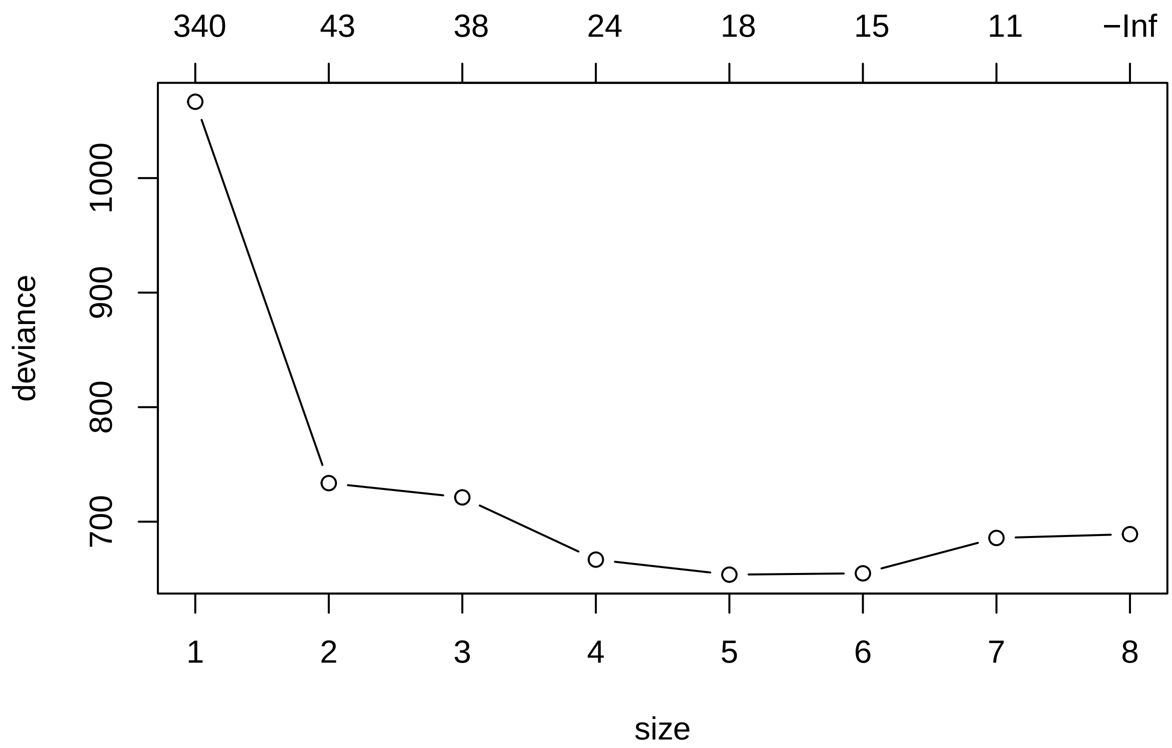
The test error rate is $\frac{49+12}{147+12+49+62} \approx 0.226$.

(f)

```r
oj.cv <- cv.tree(oj.tree)
```

(g)

```r
plot(oj.cv, type = "b")
```

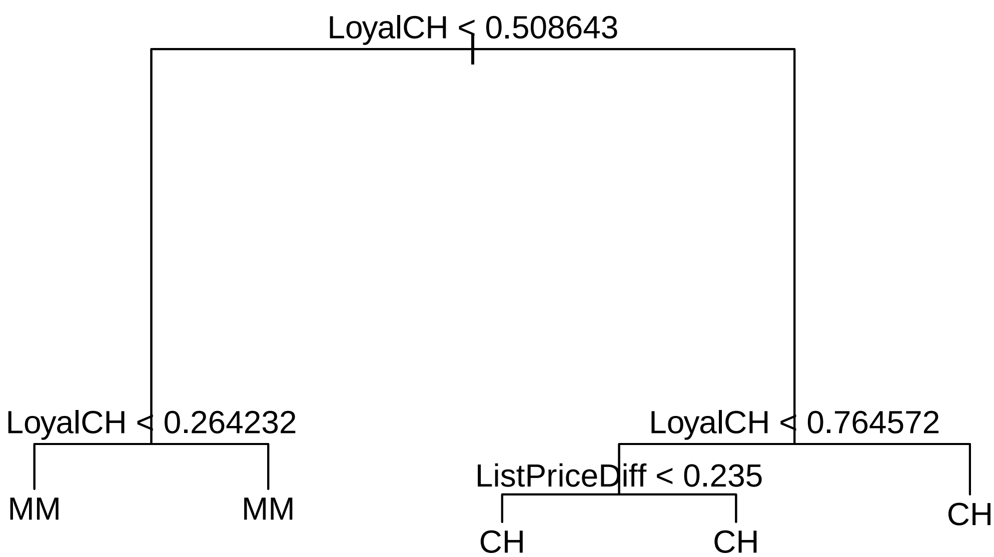


(h)

Tree size = 5

(i)

```r
oj.prune <- prune.tree(oj.tree, best = 5)
plot(oj.prune)
text(oj.prune)
```



(j)

```r
summary(oj.prune)
```

```
##
## Classification tree:
## snip.tree(tree = oj.tree, nodes = 4:5)
## Variables actually used in tree construction:
## [1] "LoyalCH"       "ListPriceDiff"
## Number of terminal nodes:  5
## Residual mean deviance:  0.7829 = 622.4 / 795
## Misclassification error rate: 0.1825 = 146 / 800
```

Training error rate of unpruned tree is 0.165.
Training error rate of pruned tree is 0.1825, which is higher than that of unpruned tree.

(k)

```
oj.prune.pred <- predict(oj.prune, oj.test, type = "class")
table(oj.test$Purchase, oj.prune.pred)
```

```
##     oj.prune.pred
##       CH  MM
##   CH 119  40
##   MM  30  81
```

Test error rate of unpruned tree is 0.226.
Test error rate of pruned tree is $\frac{30+40}{270} \approx 0.26$, which is also higher than that of unpruned tree.

8.4 - 12

```
load("mnist_all.RData")
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.
```

```
index <- (train$y == 3 | train$y == 6)
mnist <- train$x[index,]
mnist.y <- train$y[index]
mnist <- as.data.frame(mnist)
n <- length(mnist)
mnist.var <- c()
for (i in c(1:n)){
  mnist.var[i] <- var(mnist[,i])
}
var.df <- data.frame(c(1:n), mnist.var)
var.df <- var.df[order(mnist.var, decreasing = TRUE),]
head(var.df, 25)
```

```
##     c.1.n. mnist.var
## 544    544  13358.38
## 545    545  13088.04
## 543    543  12951.18
## 515    515  12908.53
## 573    573  12879.42
## 352    352  12824.61
## 351    351  12789.34
## 572    572  12745.48
## 574    574  12742.86
## 629    629  12716.39
## 628    628  12695.26
## 516    516  12624.71
## 186    186  12619.52
## 325    325  12619.00
## 630    630  12609.90
## 631    631  12598.17
```

```
## 546    546  12572.99
## 353    353  12543.94
## 213    213  12523.84
## 487    487  12487.50
## 548    548  12474.68
## 627    627  12442.04
## 180    180  12441.06
## 324    324  12439.74
## 632    632  12437.70
```

```
topvar.index <- c(544, 545, 543, 515, 573,
                  352, 351, 572, 574, 629,
                  628, 516, 186, 325, 630,
                  631, 546, 353, 213, 487,
                  548, 627, 180, 324, 632)
mnist <- mnist[,topvar.index]
mnist$y <- as.factor(mnist.y/3-1)     # 0 for number 3;  1 for number 6
index.test <- (test$y == 3 | test$y == 6)
mnist.test <- test$x[index.test,]
mnist.test.y <- test$y[index.test]
mnist.test <- as.data.frame(mnist.test)
mnist.test <- mnist.test[,topvar.index]
mnist.test$y <- as.factor(mnist.test.y/3-1)
```

Bagging

```
set.seed(2)
bag.mnist <- randomForest(y ~ ., data = mnist, mtry = 25, importance = TRUE)
bag.mnist
```

```
##
## Call:
##  randomForest(formula = y ~ ., data = mnist, mtry = 25, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 25
##
##         OOB estimate of  error rate: 2.57%
## Confusion matrix:
##      0    1 class.error
## 0 5959  172  0.02805415
## 1  138 5780  0.02331869
```

```
bag.pred <- predict(bag.mnist, newdata = mnist.test)
table(bag.pred, mnist.test$y)
```

```
##
## bag.pred   0   1
##        0 988  24
##        1  22 934
```

Bagging test error rate is $\frac{24+22}{988+24+22+934} \approx 0.023374$

Random Forest

```r
set.seed(3)
rf.mnist <- randomForest(y ~ ., data = mnist, mtry = 5, importance = TRUE)
rf.mnist
```

```
##
## Call:
##  randomForest(formula = y ~ ., data = mnist, mtry = 5, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 2.37%
## Confusion matrix:
##      0    1 class.error
## 0 5968  163  0.02658620
## 1  123 5795  0.02078405
```

```r
rf.pred <- predict(rf.mnist, newdata = mnist.test)
table(rf.pred, mnist.test$y)
```

```
##
## rf.pred   0   1
##       0 991  26
##       1  19 932
```

Random forest test error rate is $\frac{19+26}{1968} \approx 0.022866$

Comparing to Logistic Regression

```r
mnist.glm <- glm(y ~ .,data = mnist, family = binomial)
glm.prob <- predict(mnist.glm, newdata = mnist.test, type = "response")
glm.pred <- rep(0, length(glm.prob))
glm.pred[glm.prob > 0.5] = 1
table(glm.pred, mnist.test.y)
```

```
##         mnist.test.y
## glm.pred   3   6
##        0 967  40
##        1  43 918
```

Logistic Regression test error rate is approximately 0.042.

Overall, both bagging and random forest methods perform better than logistic regression. Among all three methods, **random forest** gives the lowest test error rate of 2.2866%.