Final

Liyuan Tang

2020/6/6

Problem 1

(a). We can take the majority, so the minimum prediction rule is to predict all value as 1. The corresponding risk = $\frac{1}{8}$

(b) For this tree, we still use the optimal rule that predicts $y = \text{argmax}_i(p_i)$. So for $x \in [0,1] \times [0,3]$, the predicted y value is 0. And for $x \in [1,3] \times [0,3]$, the predicted y value is 0 also. For the risk, we can get misclassification only when $x \in [0,1] \times [0,3]$ and the sample y value is 1.

$$\begin{aligned} p(y = 1 \text{ and } x \in [0,1] \times [0,3]) &= p(y = 1) \cdot p(x \in [0,1] \times [0,3] | y = 1) \\ &= \frac{1}{8} \cdot 1 \\ &= \frac{1}{8} \end{aligned}$$

So the split did not reduce the risk.

(c). The risk for the different split should be the same. It is because the predicted y value will always be 0 as previous.

Consider the following two cases:

1). Shift the split line to the right. Then on the right side, the risk will be 0. For the left side, the conditional density function $p(x \in [0,1] \times [0,3] | y = 1) = 1$ always. So the risk will not change.

2). Shift the line to the left, at a_0. The prediction rule will not change, since in whatever part of the split, the majority of data will always be 0. So the predicted value wil be 0 always.

Thus, there is no different split of R_0 into two axis parallel boxes that would the reduce the risk.

(d).

$$\begin{aligned} risk &= p(y = 0 \text{ and classified as } 1) + p(y = 1 \text{ and classified as } 0) \\ &= \frac{7}{8} \cdot \frac{1}{8} + \frac{1}{8} \cdot \frac{7}{8} \\ &= \frac{7}{32} \end{aligned}$$

(e). The risk of the probabilistic rule for this tree occurs when the predicted y value is different with the

original y value. We first compute the conditional distribution of y given x.

$$
\begin{aligned}
p(X \in [0,1] \times [0,3]) &= \frac{1}{3}\frac{7}{8} + \frac{1}{8} = \frac{5}{12} \\
p(Y = 1 | X \in [1,3] \times [0,3]) &= 0 \\
p(Y = 0 | X \in [1,3] \times [0,3]) &= 1 \\
p(Y = 0 | X \in [0,1] \times [0,3]) &= \frac{p(X \in [0,1] \times [0,3] | Y = 0) \cdot p(Y = 0)}{p(X \in [0,1] \times [0,3])} = \frac{\frac{1}{3}\frac{7}{8}}{\frac{5}{12}} = \frac{7}{10} \\
p(Y = 1 | X \in [0,1] \times [0,3]) &= \frac{3}{10}
\end{aligned}
$$

Then for the risk, it can only occur when $x \in [0,1] \times [0,3]$

$$
\begin{aligned}
& p(\hat{y} = 0 \text{ and } x \in [0,1] \times [0,3] \text{ and } y = 1) + p(\hat{y} = 1 \text{ and } x \in [0,1] \times [0,3] \text{ and } y = 0) \\
&= \frac{7}{10} \cdot \frac{3}{10} \cdot \frac{5}{12} + \frac{3}{10} \cdot \frac{7}{10} \cdot \frac{5}{12} \\
&= \frac{7}{40}
\end{aligned}
$$

(f) No. We can consider the following two cases:

1). If we move the split line a little bit to the left. For example, consider $N_l = [0, 0.75] \times [0, 3]$ and $N_r = (0.75, 3] \times [0, 3]$. For the risk in N_l, it will not change since the conditional density function for y will not change. Then, consider N_r. Previously the risk in N_r is 0, but now it will increase since the purity of Y=1 is not 0. Thus, the risk will increase if we shift the line to the left.

2). Move the line to the right. For example, consider $N_l = [0, a_0] \times [0, 3]$ and $N_r = (a_0, 3] \times [0, 3]$. $(3 > a_0 \geq 1)$ The risk in N_r is 0 as in part (e). Now consider N_l. We can first compute the conditional density function of y given x as a function of a_0.

$$
\begin{aligned}
p(X \in [0, a_0] \times [0, 3]) &= \frac{a_0}{3}\frac{7}{8} + \frac{1}{8} \\
p(Y = 0 | X \in [0, a_0] \times [0, 3]) &= \frac{p(X \in [0, a_0] \times [0, 3] | Y = 0) p(Y = 0)}{p(X \in [0, a_0] \times [0, 3])} = \frac{\frac{a_0}{3}\frac{7}{8}}{\frac{a_0}{3}\frac{7}{8} + \frac{1}{8}} = \frac{7a_0}{7a_0 + 3} \\
p(Y = 1 | X \in [0, a_0] \times [0, 3]) &= \frac{3}{7a_0 + 3}
\end{aligned}
$$

Then, we can compute the risk.

$$
\begin{aligned}
risk &= (\frac{a_0}{3}\frac{7}{8} + \frac{1}{8}) \cdot \frac{7a_0}{7a_0 + 3} \cdot \frac{3}{7a_0 + 3} \cdot 2 \\
&= \frac{1}{4}\frac{7a_0}{7a_0 + 3}
\end{aligned}
$$

If we increase a_0 from 1 to 3, the risk will increase. So there is not a different split that has a smaller risk.

Problem 2

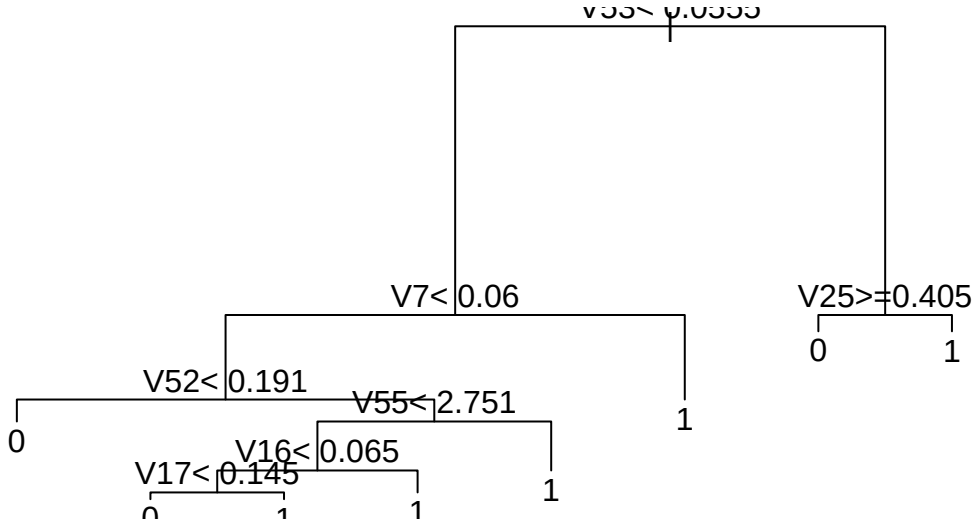
(a)

```
library(rpart)
library(tree)
spam.data = read.table("spam.data")
```

```
spam.traintest = read.table("spam.traintest")

spam.train = spam.data[spam.traintest==0,]
spam.test = spam.data[spam.traintest==1,]

spam.tree = rpart(formula=V58~., data=spam.train, method='class')
plot(spam.tree)
text(spam.tree,pretty=0)
```




```
# predict
#summary(spam.tree)
yhat.train=predict(spam.tree,newdata=spam.train,type="class")
yhat.test=predict(spam.tree,newdata=spam.test,type="class")

tb = table(yhat.train, spam.train[,58])
er.train = (tb[1,2] + tb[2,1]) / nrow(spam.train)
tb = table(yhat.test, spam.test[,58])
er.test = (tb[1,2] + tb[2,1]) / nrow(spam.test)
```

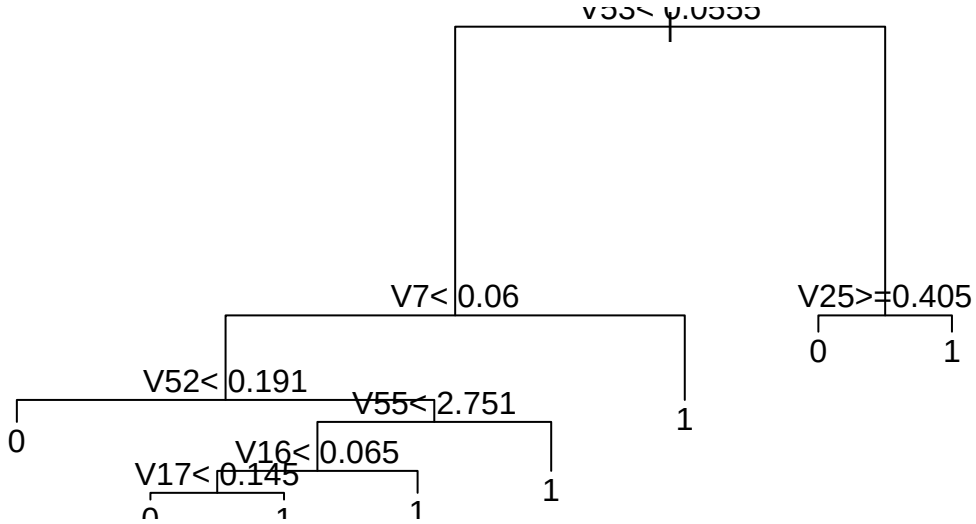
The error rate on the training is 0.0880914and on test sets is 0.1022135.

(b)

```
spam.tree = rpart(formula=V58~., data=spam.train, method='class', cp = 0.001)
spam.table = spam.tree$cptable
ind.min = which.min(spam.table[, "xerror"])
err.spam =  spam.table[ind.min, "xerror"]
se.spam = spam.table[ind.min, "xstd"]
onese = err.spam + se.spam

ind.opt = min((1:nrow(spam.table))[spam.table[, "xerror"] < onese])
cp.opt = spam.table[ind.opt, "CP"]

prune.spam = prune(tree = spam.tree, cp = cp.opt)
plot(prune.spam)
text(prune.spam,pretty=0)
```




```
yhat.train.prune=predict(prune.spam,newdata=spam.train,type="class")
yhat.test.prune=predict(prune.spam,newdata=spam.test,type="class")

tb = table(yhat.train.prune, spam.train[,58])
er.train.prune = (tb[1,2] + tb[2,1]) / nrow(spam.train)
tb = table(yhat.test.prune, spam.test[,58])
er.test.prune = (tb[1,2] + tb[2,1]) / nrow(spam.test)
```

The error rate on the training is 0.0880914and on test sets is 0.1022135.

Problem 4

(a)

```r
knn.classifier <- function (X.train, y.train, X.test, k.try = 1, pi = rep(1/K, K), CV = F) {
  n.test = nrow(X.test)
  knn.matrix = matrix(NA, nrow = n.test, ncol = length(k.try))
  for (i in 1:n.test) {
    # find distance
    curr.x = X.test[i,]
    dist.x = sweep(X.train, 2, as.numeric(curr.x)) #X.train - curr.x
    dist.x = sqrt(rowSums(dist.x^2)) # square

    # sort
    y.train.m = y.train[order(dist.x)]
    dist.x = dist.x[order(dist.x)]

    # CV
    if (CV & dist.x[1] == 0) {
      y.train.m = y.train.m[-1]
    } else {
      y.train.m = y.train.m
    }

    # knn
    for (j in 1:length(k.try)) {
      k = k.try[j]   # find k
      pred.y = y.train.m[1:k]    # select the first kth terms
      tb.y = table(pred.y)[table(pred.y) == max(table(pred.y))]    # find the most common term(s)
      if (length(tb.y) == 1) {
        # find the most common y
        y.name = names(which.max(table(pred.y)))
      } else {
        # 1. find prior
        y.level = levels(factor(y.train))
        prior.y = numeric(length(tb.y))
        for (k in 1:length(tb.y)) {
          curr.y = names(tb.y)[k]
          prior.y[k] = pi[y.level == curr.y]
        }
        # 2. break the tie by prior
        max.num = sum(prior.y == max(prior.y))
        if (max.num == 1) {
          # Select the highest prior
          y.name = names(tb.y)[which.max(prior.y)]
        } else {
          tb.max = tb.y[prior.y == max(prior.y)]
          y.name = names(tb.max)[sample(1:max.num, 1)]
        }
      }
      knn.matrix[i, j] = y.name
    }
  }
  return(knn.matrix)
```

```
}
```

(b)

```
iris.f = knn.classifier(iris[,-5], iris[,5], iris[,-5], 5, rep(1/3, 3), FALSE)
iris.t = knn.classifier(iris[,-5], iris[,5], iris[,-5], 5, rep(1/3, 3), TRUE)

iris.f.mis = sum(iris.f != iris[,5])
iris.t.mis = sum(iris.t != iris[,5])
```

The number of misclassifications CV = TRUE is 5. The number of misclassifications CV = FALSE is 5

(c)

```
zip.train = read.table("zip-train.dat")
zip.test = read.table("zip-test.dat")
k.try = c(1, 3, 7, 11, 15, 21, 27, 35, 43)

p = numeric(10)
for (i in 0:9) {
  p[i+1] = sum(zip.train[,1] == i) / nrow(zip.train)
}
train.mat = knn.classifier(zip.train[,2:257], zip.train[,1], zip.train[,2:257], k.try, p, TRUE)

err.train = numeric(9)
for (i in 1:9) {
  err.train[i] = sum(train.mat[,i] != zip.train[,1])
}
k.opt = k.try[which.min(err.train)]
```

The optimal choice of k is 1. The corresponding error rate is 0.035

(d)

```
test.mat = knn.classifier(zip.train[,2:257], zip.train[,1], zip.test[,2:257], k.opt, p, TRUE)
err.test = sum(test.mat != zip.test[,1]) / nrow(zip.test)
```

The test set error rate is 0.0792227.