PH 245 HW 4 Solutions

GSIs: Xiangyu Hu and Toki Sherbakov

December 9, 2014

Rubric (Total: 20 points, Extra Credit: 3 points):

Problem 1 (2 **extra credit** points)
- part (a): 0 points
- part (b): 1 **extra credit** point for correct plot of CART
- part (c): 1 **extra credit** point for correct values for MCR

Problem 2 (20 points, 1 **extra credit** point)
- part(a) 2 points total: 2 points for correct partitioning of dataset
- part(b) 10 points total: 2 points each (10 points total) for correct MCR for each method (5 methods)
- part(c) & (d) 8 points total: 2 points for correct method, 1 point for correct MCR, 1 point for description
- part(e): 1 **extra credit** point for correct implementation of 5-fold CV

Problem 1

Part a):

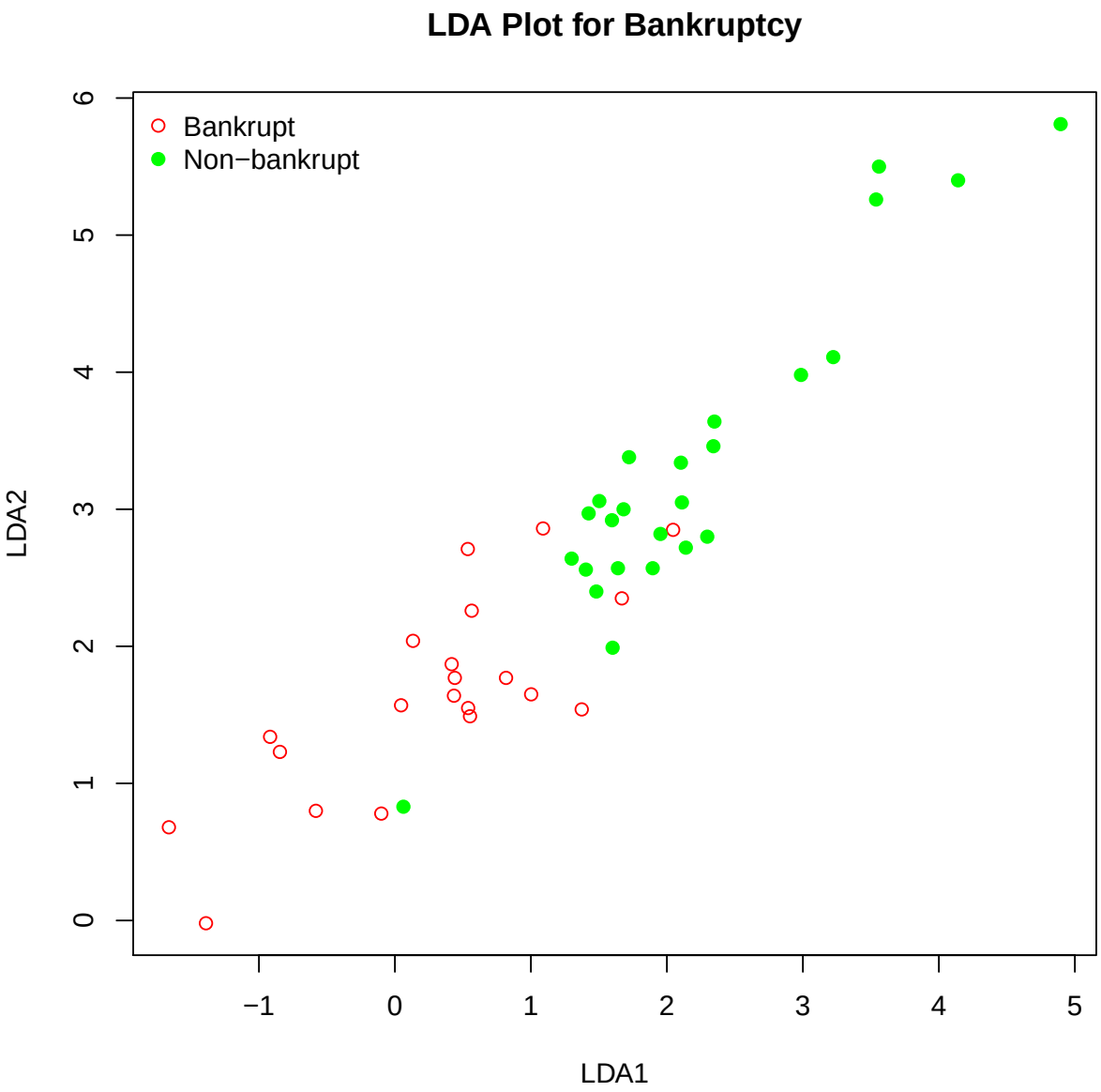
```
# Import finance dataset
finance = read.table("Data-HW4-finance.dat")

# Subset the dataset so that the variables are in X and the outcome in Y
X = finance[,-5]
Y = finance[,5]

# Obtain LDA fit of data
library(MASS)
out.lda = lda(X, Y)

# Obtain values when evaluating the LDA fit
xb1 = as.matrix(X) %*% out.lda$scaling
xb2 = as.matrix(X) %*% rep(1, ncol(X))

# Plot the two quantities
plot(xb1, xb2, type="n", xlab = "LDA1", ylab = "LDA2",
     main = "LDA Plot for Bankruptcy")
points(xb1[Y==0], xb2[Y==0], col="red")
points(xb1[Y==1], xb2[Y==1], col="green", pch=19)
legend("topleft", col = c("red", "green"), pch = c(1, 19),
       legend = c("Bankrupt", "Non-bankrupt"), bty = "n")
```



From the plot, you can see a distinction between bankrupt and not bankrupt. LDA seems to do a good job of being able to classify the two. There are a few points muddled together in the middle which might cause a few misclassifications.

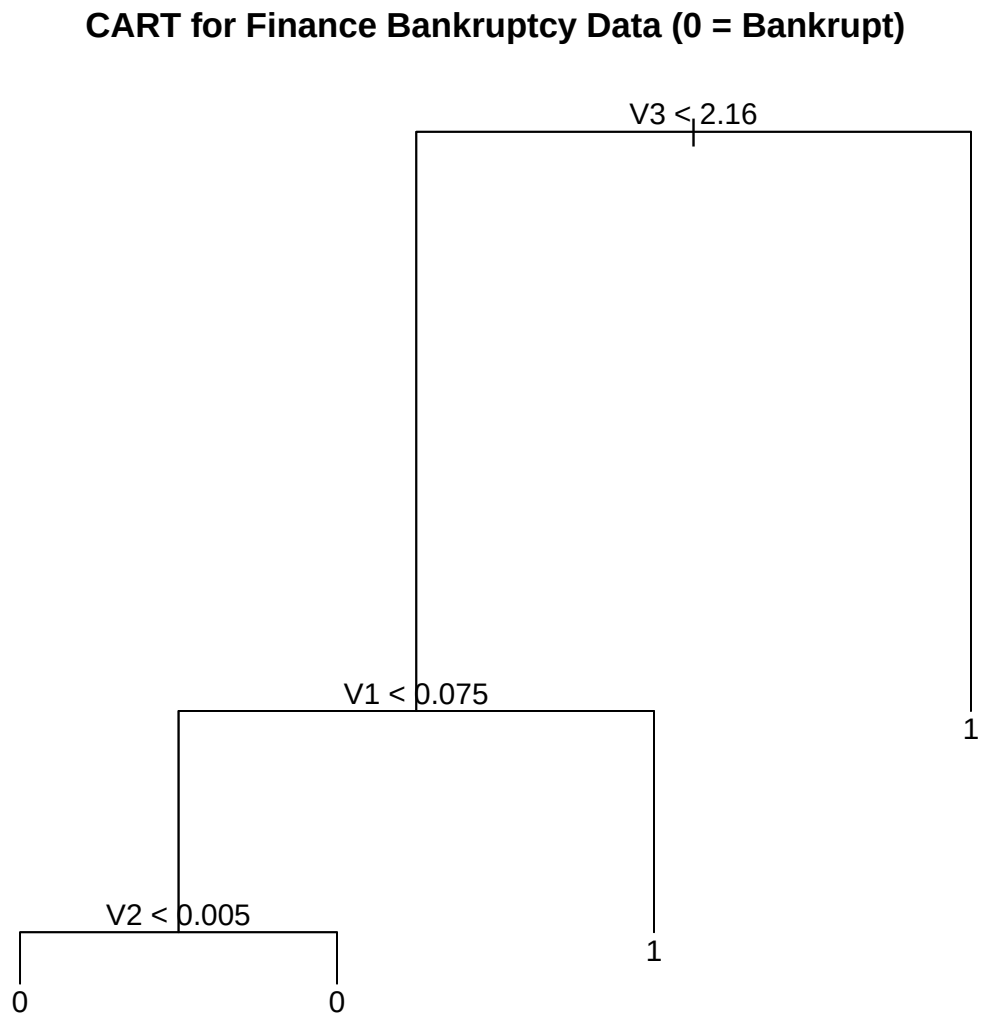
Part b):

```
# Load in CART library
library(tree)

# Apply CART to the full data
out.cart = tree(as.factor(V5)~., data = finance)

# Plot the tree
plot(out.cart)
text(out.cart)
```

```
title("CART for Finance Bankruptcy Data (0 = Bankrupt)")
```



Part c):

```
# Load in necessary libraries to obtain fits for this problem
library(mda)

## Loading required package:  class
## Loaded mda 0.4-4

library(class)

# Obtain QDA, MDA, and KNN fits
out.qda = qda(X, Y)
out.mda = mda(as.factor(Y) ~ as.matrix(X), subclasses = c(5, 5))
```

```r
out.knn = knn(X, X, Y, k = 5)

# Obtain MCR for LDA
Y.lda = predict(out.lda, X)$class
mcr.lda = sum(Y.lda != Y) / length(Y)

# Obtain MCR for QDA
Y.qda = predict(out.qda, X)$class
mcr.qda = sum(Y.qda != Y) / length(Y)

# Obtain the MCR for MDA
Y.mda = predict(out.mda, X, type="class")
mcr.mda = sum(Y.mda != Y) / length(Y)

# Obtain the MCR for KNN
mcr.knn = sum(out.knn != Y) / length(Y)

# Obtain the MCR for CART
Y.cart = predict(out.cart, X, type="class")
mcr.cart = sum(Y.cart != Y) / length(Y)

# Make vector of methods and MCR values
class.methods = c("LDA", "QDA", "MDA", "KNN", "CART")
mcr.vals = c(mcr.lda, mcr.qda, mcr.mda, mcr.knn, mcr.cart)

# Create dataframe with all MCRs for the 5 methods
mcrs = data.frame(Methods = class.methods, MCR = mcr.vals)
mcrs
```

```
##   Methods     MCR
## 1     LDA 0.08696
## 2     QDA 0.06522
## 3     MDA 0.06522
## 4     KNN 0.10870
## 5    CART 0.08696
```

Some of the methods have the same MCR. This can be due to the small sample size. When predicting the outcomes, different methods can predict the same outcomes, causing the same MCR. It looks MDA and QDA have the lowest MCR based on this dataset.

Problem 2
Part a):

```r
# Read in the Wisconsin breast cancer data
cancer = read.table("Data-HW4-breastcancer.dat", header = TRUE)

# Split data into predictors and response
X = as.matrix(cancer[,2:31])
Y = cancer[,1]

# Set seed so that everyone gets the same random sample
set.seed(100)

# Randomly select 400 samples as training set.
```

```r
# This is the index of these 400 selected observations
id.tr = sort(sample(seq(1, nrow(X)), size=400))

# Subset of the data to obtain the training set (400 observations)
X.tr = X[id.tr,]
Y.tr = Y[id.tr]

# Subset of the data to obtain the testing set (169 observations)
X.te = X[-id.tr,]
Y.te = Y[-id.tr]

# See how many cases of cancer there are in the training and testing sets
table(Y.tr)

## Y.tr
##   0   1
## 253 147

table(Y.te)

## Y.te
##   0   1
## 104  65
```

Part b):

```r
# Fit LDA, QDA, MDA, KNN, and CART on training set
out.tr.lda = lda(X.tr, Y.tr)
out.tr.qda = qda(X.tr, Y.tr)
out.tr.mda = mda(Y.tr ~ X.tr, subclasses = c(5,5))
out.tr.knn = knn(X.tr, X.tr, Y.tr, k = 5)
out.tr.cart = tree(as.factor(y)~., cancer, subset=id.tr)

# Obtain MCR for LDA
Y.te.lda = predict(out.tr.lda, as.data.frame(X.te))$class
mcr.te.lda = sum(Y.te.lda != Y.te) / length(Y.te)

# Obtain MCR for QDA
Y.te.qda = predict(out.tr.qda, as.data.frame(X.te))$class
mcr.te.qda = sum(Y.te.qda != Y.te) / length(Y.te)

# Obtain the MCR for MDA
Y.te.mda = predict(out.tr.mda, X.te, type="class")
mcr.te.mda = sum(Y.te.mda != Y.te) / length(Y.te)

# Obtain the MCR for KNN
# Use training set to train the model and obtain predictions on the testing set
Y.te.knn = knn(X.tr, X.te, Y.tr, k = 5)
mcr.te.knn = sum(Y.te.knn != Y.te) / length(Y.te)

# Obtain the MCR for CART
Y.te.cart = predict(out.tr.cart, data.frame(X[-id.tr,]), type="class")
mcr.te.cart = sum(Y.te.cart != Y.te) / length(Y.te)

# Make vector of methods and MCR values
```

```
class.methods = c("LDA", "QDA", "MDA", "KNN", "CART")
mcr.vals.te = c(mcr.te.lda, mcr.te.qda, mcr.te.mda, mcr.te.knn, mcr.te.cart)

# Create dataframe with all MCRs for the 5 methods
mcrs.te = data.frame(Methods = class.methods, MCR = mcr.vals.te)
mcrs.te

##   Methods     MCR
## 1     LDA 0.04142
## 2     QDA 0.03550
## 3     MDA 0.02959
## 4     KNN 0.06509
## 5    CART 0.08876
```

It looks like MDA performs the best (lowest MCR).

Part c):

```
# Fit MDA on training set with subclasses (1,1), (5,5), and (10,10)
out.tr.mda11 = mda(Y.tr ~ X.tr, subclasses = c(1,1))
out.tr.mda55 = mda(Y.tr ~ X.tr, subclasses = c(5,5))
out.tr.mda1010 = mda(Y.tr ~ X.tr, subclasses = c(10,10))

# Obtain MCR for both training and testing sets for subclass (1,1)
Y.tr.mda11 = predict(out.tr.mda11, X.tr, type="class")
mcr.tr.mda11 = sum(Y.tr.mda11 != Y.tr) / length(Y.tr)
Y.te.mda11 = predict(out.tr.mda11, X.te, type="class")
mcr.te.mda11 = sum(Y.te.mda11 != Y.te) / length(Y.te)

# Obtain MCR for both training and testing sets for subclass (5,5)
Y.tr.mda55 = predict(out.tr.mda55, X.tr, type="class")
mcr.tr.mda55 = sum(Y.tr.mda55 != Y.tr) / length(Y.tr)
Y.te.mda55 = predict(out.tr.mda55, X.te, type="class")
mcr.te.mda55 = sum(Y.te.mda55 != Y.te) / length(Y.te)

# Obtain MCR for both training and testing sets for subclass (10,10)
Y.tr.mda1010 = predict(out.tr.mda1010, X.tr, type="class")
mcr.tr.mda1010 = sum(Y.tr.mda1010 != Y.tr) / length(Y.tr)
Y.te.mda1010 = predict(out.tr.mda1010, X.te, type="class")
mcr.te.mda1010 = sum(Y.te.mda1010 != Y.te) / length(Y.te)

# Make vector that alternates "training" and "testing"
mcr.type = rep(c("training", "testing"), 3)

# Make subclasses vector
subclass = c(1,1,5,5,10,10)

# Make MCR values vector for MDA
mcr.vals.mda = c(mcr.tr.mda11, mcr.te.mda11, mcr.tr.mda55, mcr.te.mda55,
                 mcr.tr.mda1010, mcr.te.mda1010)

# Make dataframe with MCR values and names for MDA
mcr.mda = data.frame(Subclass = subclass, Set = mcr.type, MCR = mcr.vals.mda)
mcr.mda
```

```
##   Subclass      Set     MCR
## 1        1 training 0.03000
## 2        1  testing 0.04142
## 3        5 training 0.01500
## 4        5  testing 0.02959
## 5       10 training 0.02000
## 6       10  testing 0.04734
```

It is expected that training MCR is lower than the testing MCR since the model was fit on the training sets. Such pattern was observed. The training MCR is expected to decrease as the model complexity increases. The restuls doesn't meet this expectation quite well. For testing MCR, we would expect a U-shape as the model complexity increases, and our results of testing MCR agree with the expectation. The lowest MCR is with subclass (5,5) which makes sense because if we only have (1,1) subclasses, it's equivalent to LDA (remember that the MCR for LDA in the previous problem was higher than that of MDA) and if we have (10,10) subclasses, that seems to be too many because the MCR goes up again.

Part d):

```
# Fit nearest neighbor on training set for k = 1, 5, and 10
out.tr.knn1 = knn(X.tr, X.tr, Y.tr, k = 1)
out.tr.knn5 = knn(X.tr, X.tr, Y.tr, k = 5)
out.tr.knn10 = knn(X.tr, X.tr, Y.tr, k = 10)

# Obtain MCR for both training and testing sets for k = 1
mcr.tr.knn1 = sum(out.tr.knn1 != Y.tr) / length(Y.tr)
out.te.knn1 = knn(X.tr, X.te, Y.tr, k = 1)
mcr.te.knn1 = sum(out.te.knn1 != Y.te) / length(Y.te)

# Obtain MCR for both training and testing sets for k = 5
mcr.tr.knn5 = sum(out.tr.knn5 != Y.tr) / length(Y.tr)
out.te.knn5 = knn(X.tr, X.te, Y.tr, k = 5)
mcr.te.knn5 = sum(out.te.knn5 != Y.te) / length(Y.te)

# Obtain MCR for both training and testing sets for k = 10
mcr.tr.knn10 = sum(out.tr.knn10 != Y.tr) / length(Y.tr)
out.te.knn10 = knn(X.tr, X.te, Y.tr, k = 10)
mcr.te.knn10 = sum(out.te.knn10 != Y.te) / length(Y.te)

# Make MCR values vector for KNN
mcr.vals.knn = c(mcr.tr.knn1, mcr.te.knn1, mcr.tr.knn5, mcr.te.knn5,
                 mcr.tr.knn10, mcr.te.knn10)

# Make dataframe with MCR values and names for KNN
mcr.knn = data.frame(K = subclass, Set = mcr.type, MCR = mcr.vals.knn)
mcr.knn
```

```
##    K      Set     MCR
## 1  1 training 0.00000
## 2  1  testing 0.07692
## 3  5 training 0.06000
## 4  5  testing 0.06509
## 5 10 training 0.06000
## 6 10  testing 0.05917
```

It is expected that training MCR is lower than the testing MCR since the model was fit on the training sets. Such pattern was observed except for $K = 10$ (could be different if we run it multiple times). The training MCR is expected to decrease as the model complexity increases. The restuls don't meet this expectation quite well. ($K = 5$ and $k = 10$ have same training error, again results could be different if we run it multiple times). For testing MCR, we would expect a U-shape as the model complexity increases. The testing error keeps decreasing as we increase K from 1 to 10, if we increase K further, the testing error is expected to reach its minimum and then increase again. For now with the three possible values we tried with K, $K = 10$ provides the smallest testing error and thus is preferred.

Part e):

```
# Make function to perform V-fold CV on MDA to find the MCR for a given subclass
cv.mda = function(k, X = X.tr, Y = Y.tr, V = 5){

  # Set n to be nrow of given X
  n = nrow(X)

  # See how many observations are going to be in each fold
  obs.fold = n/V

  # Set starting value
  start.val = 1

  # Set empty vector for predicted values of MDA
  pred.mda = vector()

  # Loop through the ids and go fold by fold to perform 5-fold CV
  for(i in 1:V){

    # Set the indices to be used to determine training and testing sets
    ids = start.val:(start.val + obs.fold - 1)

    # Subset of the data to obtain the training set
    X.tr = X[-ids,]
    Y.tr = Y[-ids]

    # Subset of the data to obtain the testing set
    X.te = X[ids,]
    Y.te = Y[ids]

    # Fit MDA to training set
    out.tr.mda = mda(as.factor(Y.tr) ~ X.tr, subclasses = c(k,k))

    # Obtain the MCR for MDA
    pred.mda[ids] = as.numeric(predict(out.tr.mda, X.te, type="class")) - 1

    # Reset start value
    start.val = start.val + obs.fold
  }

  # Obtain MCR for all predicted values from MDA
  mcr.mda = sum(pred.mda != Y)/length(Y)

  # Return the MCR for the given subclass
```

```
  return(mcr.mda)
}

# Go through 1:30 subclasses to see which one performs best
cv.mcr = sapply(1:30, cv.mda)

# Make dataframe to see side by side how the different subclasses perform
cv.df = data.frame(Subclass = 1:30, CV_MCR = cv.mcr)
cv.df
```

```
##    Subclass CV_MCR
## 1         1 0.0450
## 2         2 0.0450
## 3         3 0.0400
## 4         4 0.0550
## 5         5 0.0325
## 6         6 0.0300
## 7         7 0.0475
## 8         8 0.0550
## 9         9 0.0500
## 10       10 0.0475
## 11       11 0.0575
## 12       12 0.0625
## 13       13 0.0500
## 14       14 0.0575
## 15       15 0.0550
## 16       16 0.0650
## 17       17 0.0625
## 18       18 0.0400
## 19       19 0.0625
## 20       20 0.0600
## 21       21 0.0600
## 22       22 0.0550
## 23       23 0.0575
## 24       24 0.0500
## 25       25 0.0550
## 26       26 0.0525
## 27       27 0.0475
## 28       28 0.0475
## 29       29 0.0550
## 30       30 0.0475
```

```
cv.df[order(cv.df$CV_MCR),]
```

```
##    Subclass CV_MCR
## 6         6 0.0300
## 5         5 0.0325
## 3         3 0.0400
## 18       18 0.0400
## 1         1 0.0450
## 2         2 0.0450
## 7         7 0.0475
## 10       10 0.0475
## 27       27 0.0475
```

```
## 28       28 0.0475
## 30       30 0.0475
## 9         9 0.0500
## 13       13 0.0500
## 24       24 0.0500
## 26       26 0.0525
## 4         4 0.0550
## 8         8 0.0550
## 15       15 0.0550
## 22       22 0.0550
## 25       25 0.0550
## 29       29 0.0550
## 11       11 0.0575
## 14       14 0.0575
## 23       23 0.0575
## 20       20 0.0600
## 21       21 0.0600
## 12       12 0.0625
## 17       17 0.0625
## 19       19 0.0625
## 16       16 0.0650
```

```
# Based on 5-fold CV, the optimal number of subclasses is (6,6) for MDA.

# With the optimal number of subclasses, fit a MDA on testing set and find MCR
out.mda.te =  mda(Y.te~X.te, subclass=c(6,6))
pred.mda.te = predict(out.mda.te, X.te, type="class")
mcr.te = sum(pred.mda.te != Y.te)/length(Y.te)
mcr.te
```

```
## [1] 0.01775
```

Based on 5-fold CV, the optimal number of subclasses is (6,6) for MDA. The MCR for the MDA method with the optimal number of subclasses (6,6) is 0.01775.