Stat 228 Midterm 2

Marisa Papagelis

04/29/2022

```
mydata <- read.csv(file=paste(filepath, "midtermII.csv", sep=""), header=TRUE)
```

Split Data The first step in our data analysis is to split the data into two equal-sized subsets of 21000 for training and testing.

```
set.seed(1)
ind.train = sample(1:nrow(mydata),21000,replace=FALSE)
train.set = mydata[ind.train,]
test.set = mydata[-ind.train,]
```

Principal Component Analysis In performing principal component analysis on the pixels of the training set, we first remove the label column from the data. We don't need to standardize these variables, since they are all on the same scale. We find that 53 PC's are the minimum number of PC's necessary to capture at least 90% of variation of the images in the training set. We subset these 53 PC's to use for the rest of the data analysis.

```
PCA <- prcomp(x=train.set[,-1], center=FALSE, scale=FALSE)
M <- which.max(cumsum(PCA$sdev^2)/sum(PCA$sdev^2) >= 0.9) # M = 53
# subset M = 53 PC's for rest of analysis
PC.train <- PCA$x[,1:M]
PC.test <- predict(PCA, newdata=test.set[,-1])[,1:M]
```

Linear Discriminant Analysis Once we fit LDA on the 53 principal components for all the training images, we predict the labels for the testing images using a threshold of 0.5 by default. The confusion matrix shows the true labels (grouping.test) against the predictions [(1/0) signifying (Yes/No)]. The misclassification error rate on the test set can be calculated by dividing the total number of incorrect predictions (sum of first column in matrix) by the total number of predictions (sum of all values in matrix). The misclassification error rate on the test set was 0.136381.

In discussing the applicability of using LDA in this circumstance, we look at the LDA assumptions. LDA depends on three assumptions: normality, equal variance, and multicollinearity. The normality assumption is met using the normality probability plots for PC1-PC4. It is important to note that slight deviation from multivariate normality won't strongly impact the result of LDA. Next, multicollinearity does not exist. To check the homogeneity assumption, we perform the Box M's test under the null hypothesis that all groups have the same variance-covariance matrix. As shown by the p-value of nearly 0, we reject the null hypothesis and conclude the homogeneity assumption is not met. As a result, observations would be over-classified into the groups(s) with larger variance-covariance matrix. LDA leads to a linear discriminant boundary between classes, so for some data sets such as this one, where the boundary may not be linear, LDA may perform poorly compared to other methods. In this circumstance, the LDA method may not perform best, and remedies may be possible through transformations of the x-variables, or another method such as QDA.

```
library(MASS)
grouping.train <- as.factor(train.set$label)
```

```r
fit.lda <- lda(PC.train, grouping.train)
# predict digit labels for testing images
lda.prediction <- predict(fit.lda, newdata=PC.test)$class
# confusion matrix to compare predictions with true values
grouping.test <- as.factor(test.set$label)
Y.hat <- ifelse(lda.prediction == grouping.test,1,0)
table(grouping.test, Y.hat)
```

```
##              Y.hat
## grouping.test    0    1
##             0  150 1925
##             1  103 2224
##             2  447 1619
##             3  381 1870
##             4  295 1766
##             5  337 1513
##             6  167 1903
##             7  333 1840
##             8  406 1657
##             9  245 1819
```

```r
# misclassification error rate on test set
misclass = sum(lda.prediction!=grouping.test)/length(lda.prediction)
misclass # 0.136381
```

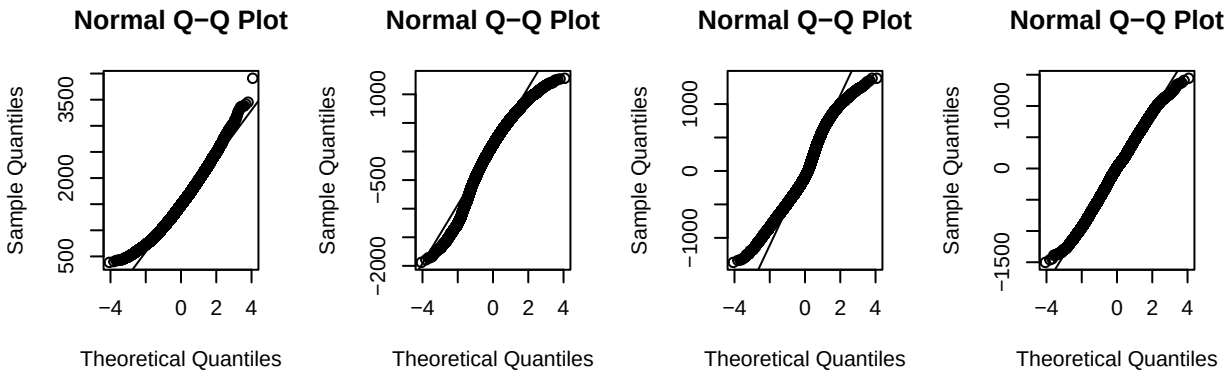
```
## [1] 0.136381
```

```r
# normality assumption
par(mfrow=c(2,4))
qqnorm(PC.train[,1])
qqline(PC.train[,1])
qqnorm(PC.train[,2])
qqline(PC.train[,2])
qqnorm(PC.train[,3])
qqline(PC.train[,3])
qqnorm(PC.train[,4])
qqline(PC.train[,4])
# homogeneity assumption
library(biotools)
```

```
## ---
## biotools version 4.2
```

```r
boxM(PC.train, grouping.train)
```

```
##
##  Box's M-test for Homogeneity of Covariance Matrices
##
## data:  PC.train
## Chi-Sq (approx.) = 854073, df = 12879, p-value < 2.2e-16
```



Quadratic Discriminant Analysis Compared to LDA, QDA relaxes the homogeneity assumption and allows a different covariance-matrix for each class. Under the QDA assumptions, the decision boundary between classes is quadratic, not linear, so it may work better in this scenario. To test this theory, we fit a QDA on the same 53 PC's for the testing images and predict the labels for the testing images using the same threshold of 0.5 by default. The confusion matrix is interpreted the same way as previously, and the misclassification error rate on the test set for QDA was 0.042, which is much better than the misclassification error rate from LDA (0.136381), showing that QDA does work better in this scenario.

```
fit.qda <- qda(PC.train, grouping.train)
# predict digit labels for testing images
qda.prediction <- predict(fit.qda, newdata=PC.test)$class
# confusion matrix to compare predictions with true values
Y.hat <- ifelse(qda.prediction == grouping.test,1,0)
table(grouping.test, Y.hat)
```

```
##              Y.hat
## grouping.test    0    1
##             0   40 2035
##             1  115 2212
##             2   66 2000
##             3  125 2126
##             4   64 1997
##             5   90 1760
##             6   73 1997
##             7  121 2052
##             8   77 1986
##             9  111 1953
```

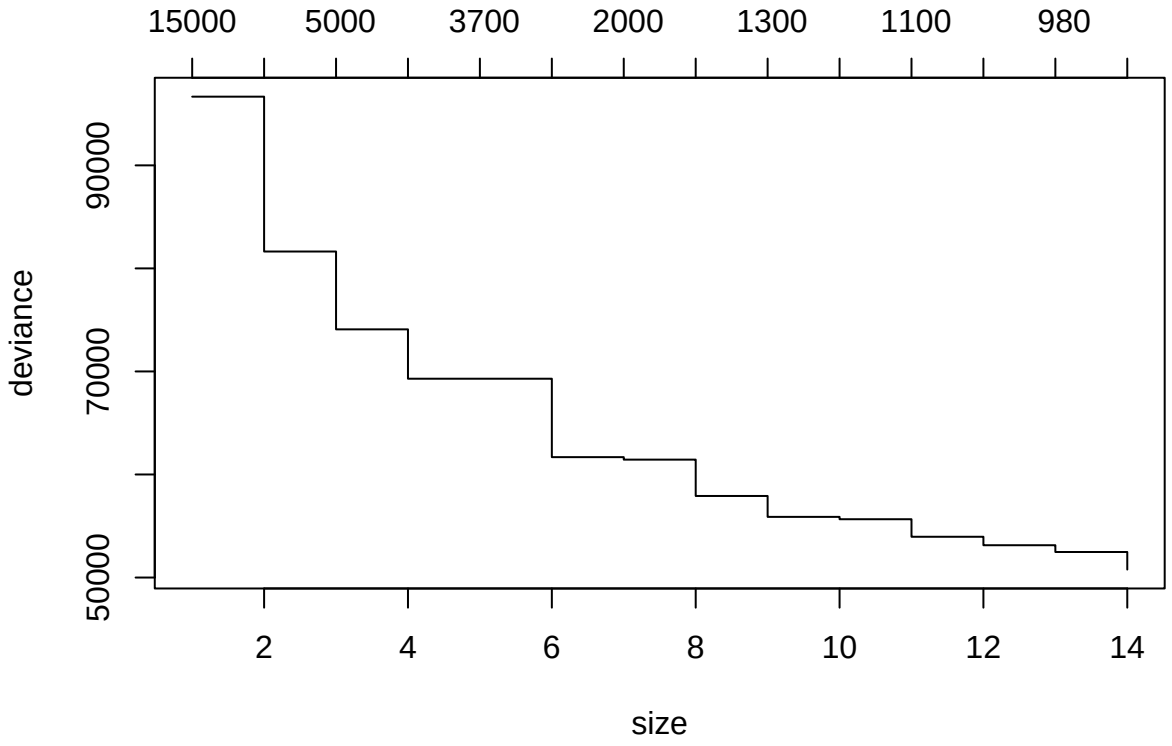
```
# misclassification error rate on test set
misclass = sum(qda.prediction!=grouping.test)/length(qda.prediction)
misclass # 0.042
```

```
## [1] 0.042
```

Classification Tree Using 10-fold cross validation, we determine that the optimal number of terminal nodes is 14 nodes, the same number as in the original tree. The plot of the final pruned tree has 14 nodes, where the value at the end of each node represents a specified label value. PC1-PC9 are the only variables actually used in the tree construction. PC3 appears to be the most significant predictor for the label, so it is split first. PC2 and PC7 follow as important predictors. The training misclassification error rate is 0.3929 and the residual mean deviance is 2.359. In predicting the classes of observations in the test set, we create a table comparing the predictions with the true digit labels in the test set. The diagonal in the table shows the correctly classified observations for each label, and the misclassification error rate is calculated as the sum of

all the other misclassified observations (not in the diagonal) divided by the total number of observations. This time the misclassification error rate on the test set is 0.9401238.

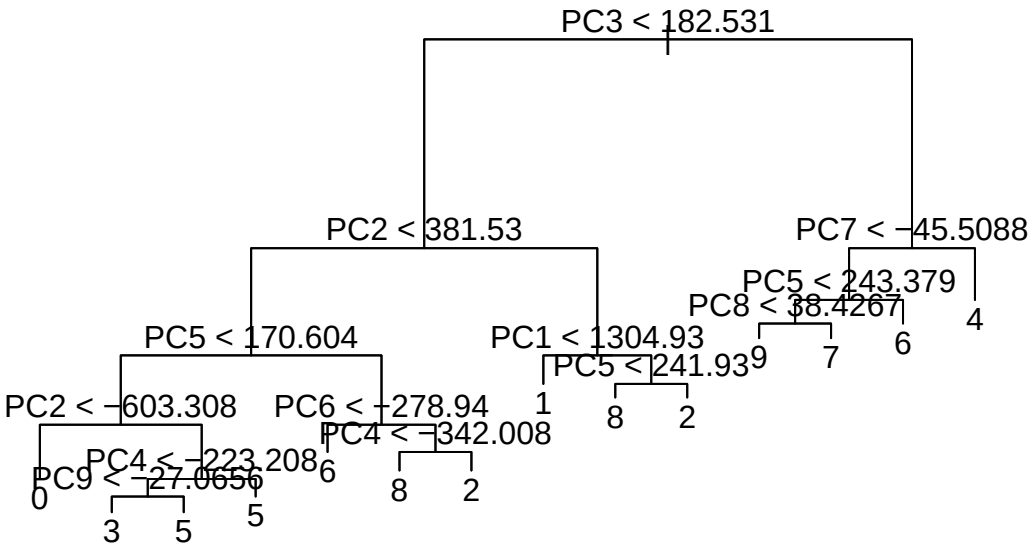
```
library(tree)
# combine grouping with PC's for CART
PC.train.grouping <- data.frame(grouping.train, PC.train)
PC.test.grouping <- data.frame(grouping.test, PC.test)
# fit original tree for pruning
original.tree <- tree(grouping.train~.,data=PC.train.grouping)
# identify best size tree based on 10 fold CV
result <- cv.tree(original.tree, FUN=prune.tree, K=10)
plot(result) # size 14 is optimal
```




```
# fit a new tree with optimal number of nodes
final.tree <- prune.tree(original.tree, best = 14)
summary(final.tree)
```

```
##
## Classification tree:
## tree(formula = grouping.train ~ ., data = PC.train.grouping)
## Variables actually used in tree construction:
## [1] "PC3" "PC2" "PC5" "PC4" "PC9" "PC6" "PC1" "PC7" "PC8"
## Number of terminal nodes:  14
## Residual mean deviance:  2.359 = 49520 / 20990
## Misclassification error rate: 0.3929 = 8250 / 21000
```

```
plot(final.tree)
text(final.tree, pretty=0)
```




```
# predict classes of observations in the test set
tree.prediction <- predict(final.tree, newdata=PC.test.grouping)
# table to compare predictions with true values
test.Y = apply(tree.prediction,1,which.max)
# relabel columns
test.Y[test.Y==1]=0
test.Y[test.Y==2]=1
test.Y[test.Y==3]=2
test.Y[test.Y==4]=3
test.Y[test.Y==5]=4
test.Y[test.Y==6]=5
test.Y[test.Y==7]=6
test.Y[test.Y==8]=7
test.Y[test.Y==9]=8
test.Y[test.Y==10]=9
table(grouping.test, test.Y)
```

```
##              test.Y
## grouping.test    0    1    2    3    4    5    6    7    8    9
##             0 1520    2   61   13   27  314   31    9   64   34
##             1    0 1923   21    1    2   96    6    0  276    2
##             2   31   70 1408   10   51  240  110   17  120    9
##             3   21   68   33 1106   56  511   75    9  359   13
##             4    1   69   43    3 1695   17   93   11   40   89
##             5  130   79   28   84  188 1050   61    2  159   69
##             6   19   66  361    0   92  228 1254    5   43    2
##             7    3  104   18    4  124   30   51 1499   68  272
##             8   20   78  110   70  176  615   67    6  886   35
##             9    5   77   10    9 1403   20  132  110   65  233
```

```
# misclassification error rate on test set
misclass = (210000-(1520+1923+1408+1106+1695+1050+1254+1499+886+233))/210000
misclass # 0.9401238
```

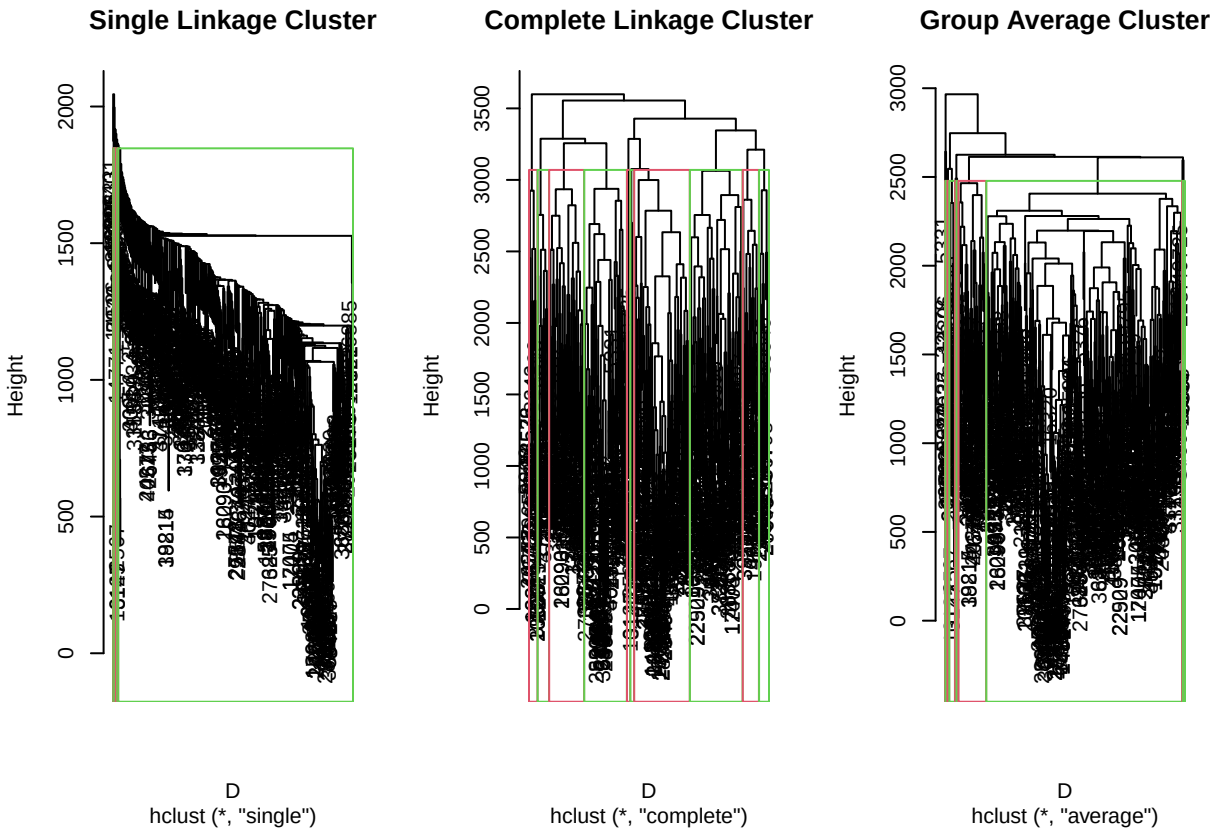
```
## [1] 0.9401238
```

Comparison of Approaches In comparing the LDA, QDA, and CART approaches for the purpose of multi-class classification, we first compare the misclassification error rates on the test set. The misclassification error rates were 0.136381, 0.042, and 0.9401238 for LDA, QDA, and CART respectively. When looking at these values, QDA appears to be most effective based on misclassification error rate alone. LDA is somewhat comparable, and CART is not great at all. When comparing LDA and QDA directly, we already know from previous analysis that they are very similar methods, and QDA would work better with this data set due to it's leniency with the homogeneity assumption, which the dataset fails using the Box M test. Within LDA, 2 and 8 were misclassified most frequently and within QDA, 3 and 7 were. Under the QDA assumption, the decision boundary between classes is quadratic, rather than linear, which is a better fit for a data set where the pixels between some numbers can be very similar.

The unsatisfactory prediction performance of the CART is likely due to over-fitting on the model, since the optimal model did not prune any nodes through CV. CART's in general have a much worse predictive performance than other methods. In the comparison table, we can further see that a large majority of misclassified values (1403 observations) came from a misclassification of 9 as 4. This makes sense because 9 and 4 look very similar to one another in terms of pixels. The fitted classification tree does not work very well for multi-class classification in this context, as most PC's do not heavily represent a majority of one label. Instead they are more evenly spaced, preventing set decision partitions from being effective. To obtain a better predictive model, one could try an advanced tree-based method, such as bagging, random forests or boosting.

Hierarchical Clustering In realizing the 10 cluster hierarchical clustering solution on a random sample of 500 training images, the first thing to notice is that all three linkage methods, single, complete and group average, yield different results. The single linkage cluster yields a result with nine smaller clusters and one distinct large cluster containing most of the observations in the data set. We notice that the solution based on single linkage shows clear evidence of the chaining effect, and thus this solution may not give a useful description of the data. The complete linkage criterion shows two very small clusters, four medium clusters, and four larger clusters. Each set of clusters is pretty evenly spaced within the dendrogram. While the complete linkage criterion shows a better spread of the clusters, we need to be weary of the crowding effect, where a point by be closer in Euclidean distance to points from another cluster than its own. While it is difficult to see this in the dendrogram itself, we can look at the observation indexes to see if this is the case. The group average criterion shows seven smaller clusters on either side of the dendogram, with two larger clusters in the middle. Out of the two larger clusters, the larger one is roughly (at least) six times larger than the smaller of the two. The group average criterion yields a compromise between the other two methods, alleviating some of the effects of both the chaining and the crowding effects. In this case, the group average clusters look a bit more similar to the single linkage clusters.

```
# random sample of 500 training images
set.seed(228)
hier.train = sample(1:nrow(PC.train),500,replace=FALSE)
hier.train.set = PC.train[hier.train,]
# fit linkages
D = dist(hier.train.set)
fit1 = hclust(D, method="single")
fit2 = hclust(D, method="complete")
fit3 = hclust(D, method="average")
# plot linkages
par(mfrow=c(1,3))
plot(fit1,main="Single Linkage Cluster")
rect.hclust(fit1,k=10,border=2:3)
plot(fit2,main="Complete Linkage Cluster")
rect.hclust(fit2,k=10,border=2:3)
plot(fit3,main="Group Average Cluster")
rect.hclust(fit3,k=10,border=2:3)
```




```
# observation indexes
result.fit1 = cutree(fit1,k=10)
result.fit2 = cutree(fit2,k=10)
result.fit3 = cutree(fit3,k=10)
```