Stats 101C HW 3

Anna Piskun

10/29/2020

The problems of homework 3 are from the textbook section 4.7 and 5.4:

Problem 1: Exercise 4.7.5

5. We now examine the differences between LDA and QDA.

(a) If the Bayes decision boundary is linear, do we expect LDA or QDA to perform better on the training set? On the test set?

If the Bayes decision boundary is linear, we can assume that the classes have a common covariance matrix. Thus, we expect LDA to perform better on the testing data set. However, LDA is a simpler, less flexible model of which if the assumptions do not hold can lead to high bias. As a result, due to the more flexible nature of a QDA model, it will perform better on the training set; however, this can lead to an overfitted model that won't preform well on the test set.

(b) If the Bayes decision boundary is non-linear, do we expect LDA or QDA to perform better on the training set? On the test set?

If the Bayes decision boundary is non-linear, the classes do not share a common covariance matrix and as such QDA will perform better on both the training and test sets. One downfall of the LDA model, is that if the assumptions do not hold then it can lead to poor estimates and a high bias. Since a shared covariance matrix is one of the key assumptions, and LDA only performs well on linear assumptions it will perform worse than QDA on both of the data sets.

(c) In general, as the sample size n increases, do we expect the test prediction accuracy of QDA relative to LDA to improve, decline, or be unchanged? Why?

As the sample size n increases we expect the test prediction accuracy of QDA relative to LDA to improve. LDA is a very simple model while QDA is more complex in that it needs to estimate more parameters, since there is one covariance matrix for each class. Since QDA is a much more flexible model, it will fit the data much better than LDA. However, the large sample size will help account for the increased variability that accompanies increased flexibility. Thus, it is recommended to use QDA for test sets with a large sample size.

(d) True or False: Even if the Bayes decision boundary for a given problem is linear, we will probably achieve a superior test error rate using QDA rather than LDA because QDA is flexible enough to model a linear decision boundary. Justify your answer.

False. One key issue with QDA is that its flexibility can lead to severe overfitting of the data. If the test set is small, looking at the Bias-Variance trade-off, we would prefer to have lower variability. If this is the case, then it will be better to proceed with LDA which will fit a linear problem well.

Problem 2: Exercise 4.7.13 (the models should be compared using 5-fold CV and not a validation set approach)

13. Using the Boston data set, fit classification models in order to predict whether a given suburb has a crime rate above or below the median. Explore logistic regression, LDA, and KNN models using various subsets of the predictors. Describe your findings.

```r
library(MASS)
attach(Boston)
summary(Boston)
```

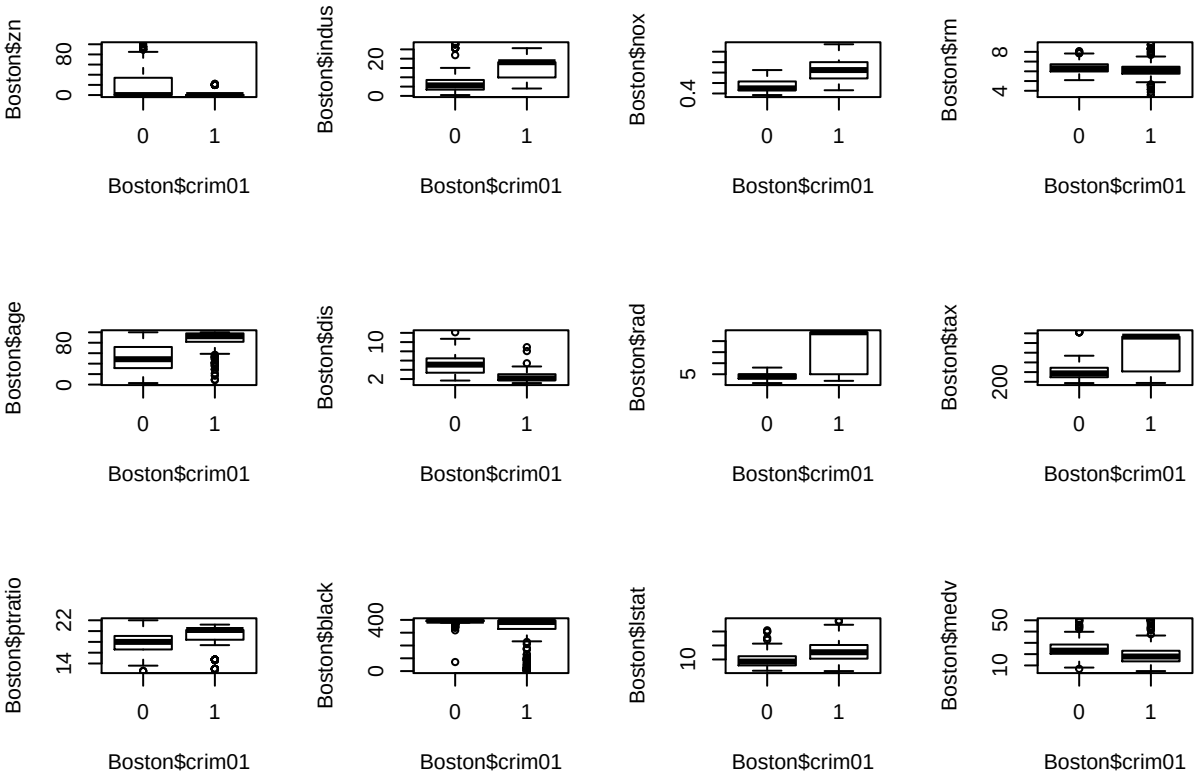
```
##       crim                zn              indus            chas        
##  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000  
##  1st Qu.: 0.08204   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000  
##  Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.00000  
##  Mean   : 3.61352   Mean   : 11.36   Mean   :11.14   Mean   :0.06917  
##  3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000  
##  Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :1.00000  
##       nox               rm              age              dis        
##  Min.   :0.3850   Min.   :3.561   Min.   :  2.90   Min.   : 1.130  
##  1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100  
##  Median :0.5380   Median :6.208   Median : 77.50   Median : 3.207  
##  Mean   :0.5547   Mean   :6.285   Mean   : 68.57   Mean   : 3.795  
##  3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188  
##  Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :12.127  
##       rad              tax           ptratio          black       
##  Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   :  0.32  
##  1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38  
##  Median : 5.000   Median :330.0   Median :19.05   Median :391.44  
##  Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :356.67  
##  3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23  
##  Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :396.90  
##      lstat            medv      
##  Min.   : 1.73   Min.   : 5.00  
##  1st Qu.: 6.95   1st Qu.:17.02  
##  Median :11.36   Median :21.20  
##  Mean   :12.65   Mean   :22.53  
##  3rd Qu.:16.95   3rd Qu.:25.00  
##  Max.   :37.97   Max.   :50.00
```

```r
median <- median(Boston$crim)

Boston$crim01 <- factor(ifelse(Boston$crim > median, 1, 0))


# 1 = crime rate above median, 0 = crime rate below median

par(mfrow = c(3,4))
boxplot(Boston$zn~Boston$crim01)
boxplot(Boston$indus~Boston$crim01)
boxplot(Boston$nox~Boston$crim01)
boxplot(Boston$rm~Boston$crim01)
boxplot(Boston$age~Boston$crim01)
boxplot(Boston$dis~Boston$crim01)
boxplot(Boston$rad~Boston$crim01)
boxplot(Boston$tax~Boston$crim01)
boxplot(Boston$ptratio~Boston$crim01)
boxplot(Boston$black~Boston$crim01)
boxplot(Boston$lstat~Boston$crim01)
boxplot(Boston$medv~Boston$crim01)
```



In order to determine which predictors are the most important we analyze the boxplots with the largest graphical differences between crim01 = 0, and crim01 = 1. Using this strategy, we find that zn, indus, nox, age, dis, rad, and tax are potentially significant predictors that should be included in our model. They have the clearest differences in their individual boxplot between the two levels of our crim01 variable. From these predictors we can randomly choose three different subsets to fit our models to. My first subset will just include *tax*, the second subset will include *tax*, *rad*, and *indus*, and the third subset will include *tax*, *rad*, *indus*, *age*, *dis*, *zn*, and *nox*.

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
#Subset 1
levels(Boston$crim01) <- sub("^0$", "below", levels(Boston$crim01))
levels(Boston$crim01) <- sub("^1$", "above", levels(Boston$crim01))


#specify how to evaluate our models

train_control <- trainControl(method="cv", number = 5,
                              classProbs = TRUE,
                              savePredictions = TRUE)

#KNN

KNNfit <- train(crim01~tax,
                data = Boston, method = 'knn',
                preProc = c("center", "scale"),
                trControl = train_control)
```

```r
#logistic regression

LRfit <- train(crim01 ~ tax,
                data = Boston, method = "glm",
                family = "binomial",
                preProc = c("center", "scale"),
                trControl = train_control)


#LDA

LDAfit <- train(crim01 ~ tax,
               data = Boston, method = "lda",
               preProc = c("center", "scale"),
               trControl = train_control)

KNNfit
```

```
## k-Nearest Neighbors
##
## 506 samples
##   1 predictor
##   2 classes: 'below', 'above'
##
## Pre-processing: centered (1), scaled (1)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 405, 404, 405, 404, 406
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa
##   5  0.9209241  0.8418439
##   7  0.9012180  0.8024252
##   9  0.8953166  0.7906468
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 5.
```

```r
LRfit
```

```
## Generalized Linear Model
##
## 506 samples
##   1 predictor
##   2 classes: 'below', 'above'
##
## Pre-processing: centered (1), scaled (1)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 405, 405, 404, 405, 405
## Resampling results:
##
##   Accuracy   Kappa
##   0.7707824  0.5416641
```

```
LDAfit
```

```
## Linear Discriminant Analysis
##
## 506 samples
##   1 predictor
##   2 classes: 'below', 'above'
##
## Pre-processing: centered (1), scaled (1)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 404, 404, 405, 406, 405
## Resampling results:
##
##   Accuracy   Kappa
##   0.7452064  0.4905066
```

```
#Subset 2: tax + rad + indus

#KNN

KNNfit.1 <- train(crim01~tax + rad + indus,
                  data = Boston, method = 'knn',
                  preProc = c("center", "scale"),
                  trControl = train_control)

#logistic regression

LRfit.1 <- train(crim01 ~ tax + rad + indus,
                  data = Boston, method = "glm",
                  family = "binomial",
                  preProc = c("center", "scale"),
                  trControl = train_control)

#LDA

LDAfit.1 <- train(crim01 ~ tax + rad + indus,
                 data = Boston, method = "lda",
                 preProc = c("center", "scale"),
                 trControl = train_control)

KNNfit.1
```

```
## k-Nearest Neighbors
##
## 506 samples
##   3 predictor
##   2 classes: 'below', 'above'
##
## Pre-processing: centered (3), scaled (3)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 405, 404, 405, 406, 404
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa
##   5  0.9171015  0.8341821
```

```
##   7  0.9072390  0.8144449
##   9  0.9072782  0.8145451
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 5.
```

```
LRfit.1
```

```
## Generalized Linear Model
##
## 506 samples
##   3 predictor
##   2 classes: 'below', 'above'
##
## Pre-processing: centered (3), scaled (3)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 404, 405, 406, 405, 404
## Resampling results:
##
##   Accuracy  Kappa
##   0.822144  0.6444475
```

```
LDAfit.1
```

```
## Linear Discriminant Analysis
##
## 506 samples
##   3 predictor
##   2 classes: 'below', 'above'
##
## Pre-processing: centered (3), scaled (3)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 404, 404, 405, 405, 406
## Resampling results:
##
##   Accuracy   Kappa
##   0.8140846  0.628082
```

```
#Subset 3: all important predictors including tax, rad, indus, age, dis, zn, and nox

#KNN

KNNfit.2 <- train(crim01~tax + rad + indus + age + dis + zn + nox,
                  data = Boston, method = 'knn',
                  preProc = c("center", "scale"),
                  trControl = train_control)

#logistic regression

LRfit.2 <- train(crim01 ~ tax + rad + indus + age + dis + zn + nox,
                  data = Boston, method = "glm",
                  family = "binomial",
                  preProc = c("center", "scale"),
                  trControl = train_control)
```

```r
#LDA

LDAfit.2 <- train(crim01 ~ tax + rad + indus + age + dis + zn + nox,
                  data = Boston, method = "lda",
                  preProc = c("center", "scale"),
                  trControl = train_control)

KNNfit.2
```

```
## k-Nearest Neighbors
##
## 506 samples
##   7 predictor
##   2 classes: 'below', 'above'
##
## Pre-processing: centered (7), scaled (7)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 404, 405, 406, 404, 405
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa
##   5  0.9306302  0.8612984
##   7  0.9247482  0.8495492
##   9  0.9149051  0.8298629
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 5.
```

```r
LRfit.2
```

```
## Generalized Linear Model
##
## 506 samples
##   7 predictor
##   2 classes: 'below', 'above'
##
## Pre-processing: centered (7), scaled (7)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 406, 406, 404, 404, 404
## Resampling results:
##
##   Accuracy   Kappa
##   0.8756078  0.7512157
```

```r
LDAfit.2
```

```
## Linear Discriminant Analysis
##
## 506 samples
##   7 predictor
##   2 classes: 'below', 'above'
##
## Pre-processing: centered (7), scaled (7)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 405, 406, 405, 404, 404
## Resampling results:
```

```
##
##   Accuracy   Kappa
##   0.8439864  0.6880371
```

From the first subset of data (only predictor variable being *tax*) we find using 5-fold CV that a KNN model using k=5 is the best model due to it having the highest model accuracy of 0.915. For the second subset of data (*tax*, *rad*, *indus*) we find that a KNN model using k=5 is once again the best model with an improved accuracy of 0.929. For subset 3 which included all important predictors (*tax*, *rad*, *indus*, *age*, *dis*, *zn*, *nox*) the KNN model with k = 5 was the best model with the highest accuracy of 0.934. Thus, the overall best model is the KNN model with 7 predictors: (*tax*, *rad*, *indus*, *age*, *dis*, *zn*, *nox*, and with k = 5 which had an accuracy percentage of approximately 94%.

Problem 3: Exercise 5.4.8

We will now perform cross-validation on a simulated data set.

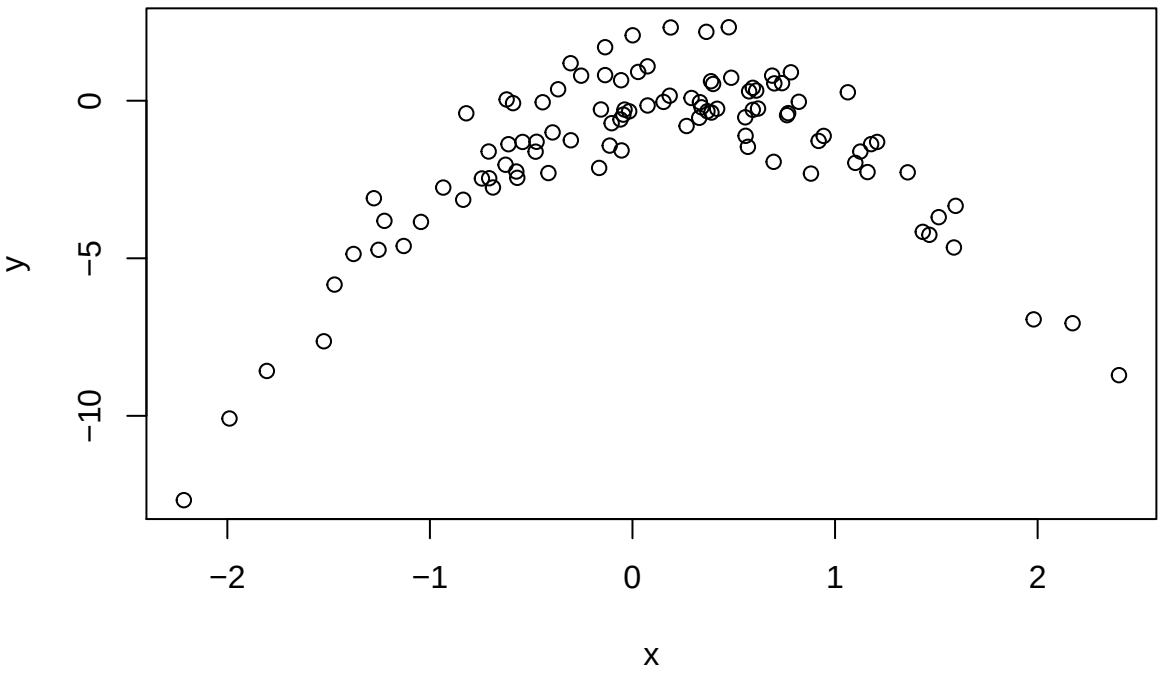
(a) Generate a simulated data set as follows: In this data set, what is n and what is p? Write out the model used to generate the data in equation form.

```
set.seed(1)
x <- rnorm(100)
y <- x-2*x^2+rnorm(100)
```

In this data set, n = 100 and p = 2. The model used to generate the data looks like the following in equation form: Y = X - 2(X^2) + error(e).

(b) Create a scatterplot of X against Y . Comment on what you find.

```
plot(x,y)
```



Looking at the scatterplot of X against Y we see a clear parabolic trend in the data. This suggests a non-linear relationship between x and y. Considering we simulated the data and model, we know that there is a quadratic relationship between x and y.

(c) Set a random seed, and then compute the LOOCV errors that result from fitting the following four models using least squares:

Note you may find it helpful to use the data.frame() function to create a single data set containing both X and Y.

```
library(boot)
```

```
##
## Attaching package: 'boot'
```

```
## The following object is masked from 'package:lattice':
##
##     melanoma
```

```
set.seed(123)
data <- data.frame(x,y)

glm.fit <- glm(y ~ x, data = data,
               family = "gaussian")
cv.glm(data, glm.fit)$delta[1]
```

```
## [1] 7.288162
```

```
glm.fit.1 <- glm(y ~ poly(x, 2), data = data,
                 family = "gaussian")
cv.glm(data, glm.fit.1)$delta[1]
```

```
## [1] 0.9374236
```

```
glm.fit.2 <- glm(y ~ poly(x, 3), data = data,
                 family = "gaussian")
cv.glm(data, glm.fit.2)$delta[1]
```

```
## [1] 0.9566218
```

```
glm.fit.3 <- glm(y ~ poly(x, 4), data = data,
                 family = "gaussian")
cv.glm(data, glm.fit.3)$delta[1]
```

```
## [1] 0.9539049
```

(d) Repeat (c) using another random seed, and report your results. Are your results the same as what you got in (c)? Why?

```
library(boot)

set.seed(147)
data <- data.frame(x,y)

glm.fit <- glm(y ~ x, data = data,
               family = "gaussian")
cv.glm(data, glm.fit)$delta[1]
```

```
## [1] 7.288162
```

```
glm.fit.1 <- glm(y ~ poly(x, 2), data = data,
                 family = "gaussian")
cv.glm(data, glm.fit.1)$delta[1]
```

```
## [1] 0.9374236
```

```
glm.fit.2 <- glm(y ~ poly(x, 3), data = data,
                 family = "gaussian")
cv.glm(data, glm.fit.2)$delta[1]
```

```
## [1] 0.9566218
```

```
glm.fit.3 <- glm(y ~ poly(x, 4), data = data,
                 family = "gaussian")
cv.glm(data, glm.fit.3)$delta[1]
```

```
## [1] 0.9539049
```

The results for D are the same as the ones we got in C. This makes sense since LOOCV does not incorporate randomness and uses the same base data with every additional degree added to the polynomial model.

(e) Which of the models in (c) had the smallest LOOCV error? Is this what you expected? Explain your answer.

The model with degree 2 had the smallest LOOCV error, which is what was expected since we fit a quadratic equation to this model and saw in our scatterplot that the relationship between X and Y is quadratic.

(f) Comment on the statistical significance of the coefficient estimates that results from fitting each of the models in (c) using least squares. Do these results agree with the conclusions drawn based on the cross-validation results?

```
summary(glm.fit.3)
```

```
##
## Call:
## glm(formula = y ~ poly(x, 4), family = "gaussian", data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0550  -0.6212  -0.1567   0.5952   2.2267
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.55002    0.09591 -16.162  < 2e-16 ***
## poly(x, 4)1   6.18883    0.95905   6.453 4.59e-09 ***
## poly(x, 4)2 -23.94830    0.95905 -24.971  < 2e-16 ***
## poly(x, 4)3   0.26411    0.95905   0.275    0.784
## poly(x, 4)4   1.25710    0.95905   1.311    0.193
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.9197797)
##
##     Null deviance: 700.852  on 99  degrees of freedom
## Residual deviance:  87.379  on 95  degrees of freedom
## AIC: 282.3
##
## Number of Fisher Scoring iterations: 2
```

Looking at our model for a polynomial of degree 4, we see that the only statistically significant coefficient estimates result from fitting our quadratic model. These results directly support the conclusions drawn based on the cross-validation results as we saw above. The model with the lowest LOOCV was in fact the quadratic model and once again the summary output above shows that the third and fourth degree polynomial models were not statistically significant.