STATS 415 HW 5

Jiahao Cheng

Feburary 2025

Problem 1

(a) For LDA, we need the following parameters:
1. Prior probabilities of groups ($\hat{\pi_k}$)
2. Group means ($\hat{\mu_k}$)
3. Pooled Variance($\hat{\sigma^2}$)

For QDA, we need the following parameters:
1. Prior probabilities of groups ($\hat{\pi_k}$)
2. Group means ($\hat{\mu_k}$)
3. Group Variance($\hat{\sigma_k^2}$)

From Figure 1, we know that
For LDA,
1. $\hat{\pi_{-1}} = 0.444$, $\hat{\pi_1} = 0.556$
2. $\hat{\mu_{-1}} = -1$, $\hat{\mu_1} = 2.6$
3. $\hat{\sigma^2} = 4.457$

For QDA,
1. $\hat{\pi_{-1}} = 0.444$, $\hat{\pi_1} = 0.556$
2. $\hat{\mu_{-1}} = -1$, $\hat{\mu_1} = 2.6$
3. $\hat{\sigma_{-1}^2} = 2.5$, $\hat{\sigma_1^2} = 4.24$

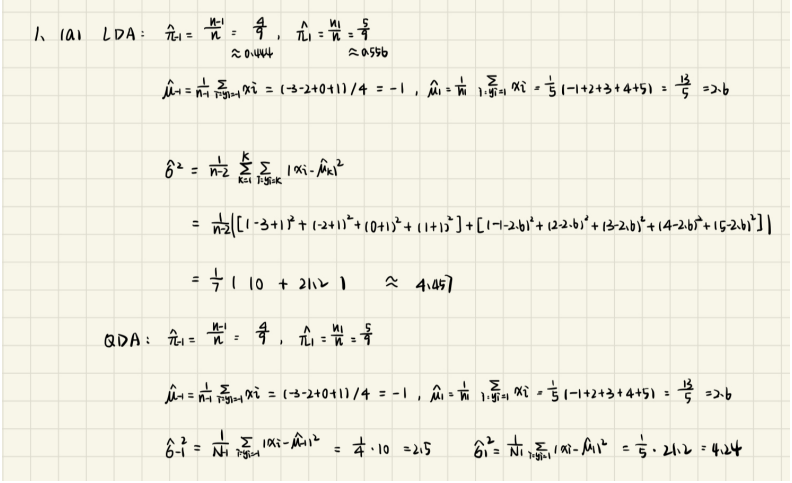

Figure 1: Parameters for LDA and QDA

(b) The discriminant functions for LDA and QDA are shown in Figure 2.

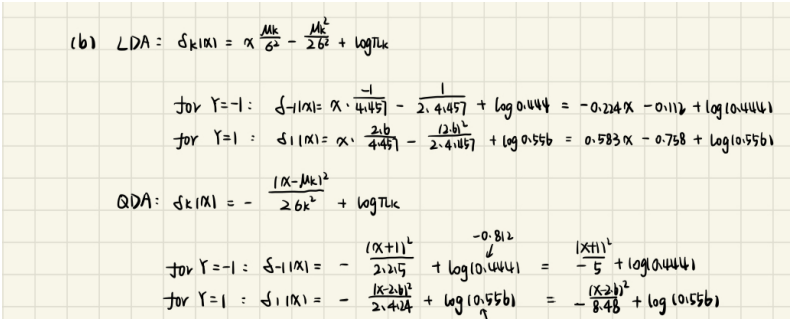

Figure 2: Discriminant functions for LDA and QDA

(c) From Figure 3, we know that the train data prediction errors for LDA and QDA are all 22.22%.

```
> X_train <- c(-3, -2, 0, 1, -1, 2, 3, 4, 5)
> Y_train <- factor(c(-1, -1, -1, -1, 1, 1, 1, 1, 1))
> training_data <- data.frame(X = X_train, Y = Y_train)
> lda_model <- lda(Y ~ X, data = training_data)
> lda_pred <- predict(lda_model)$class
> lda_error <- mean(lda_pred != Y_train)
> lda_error
[1] 0.2222222
> qda_model <- qda(Y ~ X, data = training_data)
> qda_pred <- predict(qda_model)$class
> qda_error <- mean(qda_pred != Y_train)
> qda_error
[1] 0.2222222
```

Figure 3: Train errors for LDA and QDA

(d) From Figure 4, we know that the test data prediction errors for LDA and QDA are all 25%.

```
> X_test <- c(-1.5, -1, 0, 1, 0.5, 1, 2.5, 5)
> Y_test <- factor(c(-1, -1, -1, -1, 1, 1, 1, 1))
> test_data <- data.frame(X = X_test, Y = Y_test)
> lda_pred <- predict(lda_model, test_data)$class
> lda_error <- mean(lda_pred != Y_test)
> lda_error
[1] 0.25
> qda_pred <- predict(qda_model, test_data)$class
> qda_error <- mean(qda_pred != Y_test)
> qda_error
[1] 0.25
```

Figure 4: Test errors for LDA and QDA

(e) Since the train data is quite small, LDA is the more suitable model.
Even if the group variances are different, that violates the variance assumption in LDA, but the data set is small, and train/test errors for LDA and QDA are similar, so we prefer the LDA which is the simpler and more robust model

Problem 2

(a) Figure 5 shows the code.

```
> library(ISLR)
> data(Auto)
> median_mpg <- median(Auto$mpg)
> Auto$mpg01 <- ifelse(Auto$mpg > median_mpg, 1, 0)
```

Figure 5: Create mpg01

(b) From Figures 6, 7, 8, we can see that there is a strong association between weight and mpg01.

```
> boxplot(weight ~ mpg01, data=Auto,
+         main = "Boxplot of weight by mpg_01",
+         xlab = "mpg_01",
+         ylab = "weight",
+         col = c("blue", "green"))
> plot(x=Auto$mpg, y=Auto$weight)
```

Figure 6: Plot code

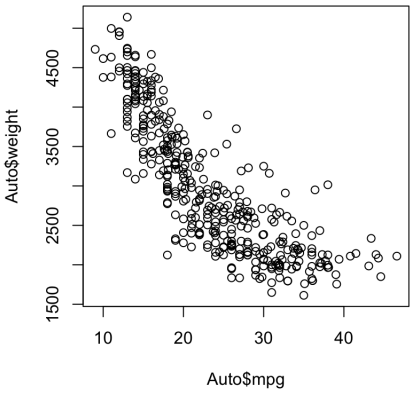

Figure 7: Scatter Plot

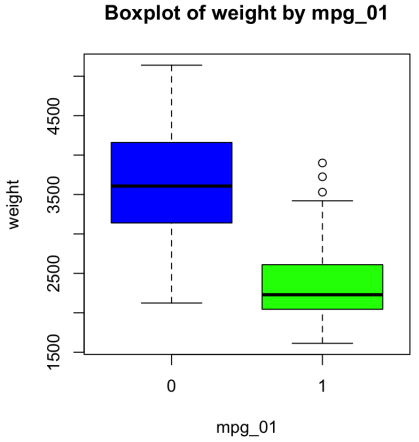

Figure 8: Box Plot

(c) Figure 9 shows the code.

```
> Auto_mpg <- data.frame(mpg01 = Auto$mpg01, weight = Auto$weight)
> library(caTools)
> set.seed(123)
> split <- sample.split(Auto_mpg$mpg01, SplitRatio = 0.7)
> train_set <- subset(Auto_mpg, split == TRUE)
> test_set <- subset(Auto_mpg, split == FALSE)
```

Figure 9: Spliting Train and Test Data

(d) Figure 10 shows the code. The test error is 13.56%

```
> lda_model <- lda(mpg01 ~ weight, data = train_set)
> lda_pred <- predict(lda_model, test_set)$class
> lda_error <- mean(lda_pred != test_set$mpg01)
> lda_error
[1] 0.1355932
```

Figure 10: LDA test error

(e) Figure 11 shows the code. The test error is 13.56%

```
> qda_model <- qda(mpg01 ~ weight, data = train_set)
> qda_pred <- predict(qda_model, test_set)$class
> qda_error <- mean(qda_pred != test_set$mpg01)
> qda_error
[1] 0.1355932
```

Figure 11: QDA test error

(f) Figure 12 shows the code. The test error is 12.71%

```
> log_model <- glm(mpg01 ~ weight, data = train_set, family = binomial)
> summary(log_model)

Call:
glm(formula = mpg01 ~ weight, family = binomial, data = train_set)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) 12.7058593  1.5552160   8.170 3.09e-16 ***
weight      -0.0045054  0.0005599  -8.047 8.50e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 379.84  on 273  degrees of freedom
Residual deviance: 151.07  on 272  degrees of freedom
AIC: 155.07

Number of Fisher Scoring iterations: 6

> log_probs <- predict(log_model, newdata = test_set, type = "response")
> log_pred <- ifelse(log_probs > 0.5, 1, 0)
> log_error <- mean(log_pred != test_set$mpg01)
> log_error
[1] 0.1271186
```

Figure 12: Logistic test error

(g) Figure 13 shows the code and the test error list.
K = 5 performs the best that has 11.85% error rate.

```
> library(class)
> train_X <- scale(train_set$weight)
> test_X <- scale(test_set$weight)
> train_Y <- train_set$mpg01
> test_Y <- test_set$mpg01
> K_values <- 1:20
> test_errors <- numeric(length(K_values))
> for (i in K_values) {
+   pred_Y <- knn(train = train_X, test = test_X, cl = train_Y, k = K_values[i])
+   test_errors[i] <- mean(pred_Y != test_Y)
+ }
> data.frame(Errors = test_errors, K = K_values)
      Errors  K
1  0.1610169  1
2  0.1440678  2
3  0.1271186  3
4  0.1271186  4
5  0.1186441  5
6  0.1355932  6
7  0.1186441  7
8  0.1186441  8
9  0.1186441  9
10 0.1186441 10
11 0.1271186 11
12 0.1271186 12
13 0.1186441 13
14 0.1186441 14
15 0.1271186 15
16 0.1355932 16
17 0.1355932 17
18 0.1271186 18
19 0.1355932 19
20 0.1355932 20
> which.min(test_errors)
[1] 5
> min(test_errors)
[1] 0.1186441
```

Figure 13: KNN test error