Linear and Quadratic Classification

Isabella Chittumuri

12/2/2020

9.11 Do a classification analysis on the fish data in Table 6.17 as follows. Assume P1 =P2= P3

y1 = aroma, y2 = flavor, y3 = texture, and y4 = moisture

```
getwd()
```

```
## [1] "/Users/isabellachittumuri/Documents/Hunter College/Fall 2020/Stat 717/HW"
```

```
fish <- read.csv("fish.csv")
head(fish)
```

```
##   method  y1  y2  y3  y4
## 1      1 5.4 6.0 6.3 6.7
## 2      1 5.2 6.2 6.0 5.8
## 3      1 6.1 5.9 6.0 7.0
## 4      1 4.8 5.0 4.9 5.0
## 5      1 5.0 5.7 5.0 6.5
## 6      1 5.7 6.1 6.0 6.6
```

(a) Find the linear classification functions.

```
library(MASS)
# Linear Discriminant Analysis
fish.lda <- lda(method ~ ., data=fish); fish.lda
```

```
## Call:
## lda(method ~ ., data = fish)
##
## Prior probabilities of groups:
##         1         2         3
## 0.3333333 0.3333333 0.3333333
##
## Group means:
##         y1       y2       y3       y4
## 1 5.383333 5.708333 5.441667 5.983333
## 2 5.258333 5.233333 5.308333 5.875000
## 3 4.975000 4.833333 5.908333 6.233333
##
## Coefficients of linear discriminants:
##            LD1        LD2
```

```
## y1 -0.04209828  1.9002068
## y2  3.30466832 -1.8134516
## y3 -1.89592960 -1.4591098
## y4 -0.90613124  0.2139395
##
## Proportion of trace:
##    LD1    LD2
## 0.9601 0.0399
```

First and second discriminant functions are linear combinations of the four variables. These discriminants are scaled :

$$LD_1 = -0.0421 * Aroma + 3.3047 * Flavor - 1.8959 * Texture - 0.9061 * Moisture$$
$$LD_2 = 1.9002 * Aroma - 1.8135 * Flavor - 1.4591 * Texture + 0.2139 * Moisture$$

Percentage separations achieved by the first discriminant function is 96.01% Percentage separations achieved by the second discriminant function is 3.99%

(b) Find the classification table using the linear classification functions in part (a) (assuming $\Sigma 1 = \Sigma 2 = \Sigma 3$).

```
# LDA Predictions
pred <- predict(fish.lda, fish)$class

# Classification table
tt <- table(Predicted = pred, Actual = fish$method); tt
```

```
##           Actual
## Predicted  1  2  3
##         1  9  3  0
##         2  3  7  1
##         3  0  2 11
```

```
# Misclassification error
1-sum(diag(tt))/sum(tt)
```

```
## [1] 0.25
```

We created predictions based on the linear classification functions and produced it's corresponding classification table. After, we found the misclassification error to be 25%.

(c) Find the classification table using quadratic classification functions (assuming population covariance matrices are not equal).

```
# Quadratic Discriminant Analysis
fish.qda <- qda(method ~ ., data=fish); fish.qda
```

```
## Call:
## qda(method ~ ., data = fish)
##
```

```
## Prior probabilities of groups:
##         1         2         3
## 0.3333333 0.3333333 0.3333333
##
## Group means:
##         y1       y2       y3       y4
## 1 5.383333 5.708333 5.441667 5.983333
## 2 5.258333 5.233333 5.308333 5.875000
## 3 4.975000 4.833333 5.908333 6.233333
```

```
# QDA Predictions
pred <- predict(fish.qda, fish)$class

# Classification table
tt <- table(Predicted = pred, Actual = fish$method); tt
```

```
##           Actual
## Predicted  1  2  3
##         1 11  3  0
##         2  1  7  1
##         3  0  2 11
```

```
# Misclassification error
1-sum(diag(tt))/sum(tt)
```

```
## [1] 0.1944444
```

We created predictions based on the quadratic classification functions and produced it's corresponding classification table. After, we found the misclassification error to be 19.4%

In comparison, the quadratic classification functions seemed to better predict the data than the linear classification functions, because it produced a lower misclassification error.

(d) Find the classification table using linear classification functions and the holdout method.

```
set.seed(555)

# Spilt for Holdout method
spilt <- sample(2, nrow(fish),
                replace = T,
                prob = c(0.5, 0.5))

# Train data
training <- fish[spilt == 1,]

# LDA for Train
train.lda <- lda(method ~ ., data=training); train.lda
```

```
## Call:
## lda(method ~ ., data = training)
##
## Prior probabilities of groups:
```

```
##         1         2         3
## 0.4666667 0.3333333 0.2000000
##
## Group means:
##         y1       y2       y3       y4
## 1 5.500000 5.728571 5.471429 6.071429
## 2 4.800000 4.880000 5.100000 5.880000
## 3 5.466667 5.200000 6.400000 6.466667
##
## Coefficients of linear discriminants:
##          LD1       LD2
## y1 -1.102505  0.425533
## y2 -2.756906  1.084834
## y3  2.254219  1.598532
## y4  1.511304 -1.037351
##
## Proportion of trace:
##    LD1    LD2
## 0.7808 0.2192
```

```
# Predictions for Train
train.pred <- predict(train.lda, training)$class

# Classification table
train.table <- table(Predicted = train.pred, Actual = training$method); train.table
```

```
##          Actual
## Predicted 1 2 3
##         1 7 1 0
##         2 0 4 0
##         3 0 0 3
```

```
# Misclassification error on train
1-sum(diag(train.table))/sum(train.table)
```

```
## [1] 0.06666667
```

We spilt the data so that 50% was in train and 50% was in test. We created predictions using the train model against the training data and produced it's corresponding classification table. After, we found the misclassification error on train to be 6.67%.

```
# Test data
testing <- fish[spilt == 2,]

# Predictions for Test
test.pred <- predict(train.lda, testing)$class

# Classification table
test.table <- table(Predicted = test.pred, Actual = testing$method); test.table
```

```
##          Actual
## Predicted 1 2 3
```

```
##         1 5 6 0
##         2 0 1 3
##         3 0 0 6
```

```
# Misclassification error on test
1-sum(diag(test.table))/sum(test.table)
```

```
## [1] 0.4285714
```

We created predictions using the train model against the testing data and produced it's corresponding classification table. After, we found the misclassification error on test to be 42.86%.

(e) Find the classification table using a nearest neighbor method.

```
library('class')
# Even split train/test (50:50)
train <- fish[c(1:6, 13:18, 25:30), ]
test <- fish[c(7:12, 19:24, 31:36), ]
cl <- factor(c(rep("1", 6), rep("2", 6), rep("3", 6)))

# Using knn=6
knn.train <- knn(train=train, test=train, cl=cl, k=6, prob=T); knn.train
```

```
##  [1] 1 1 1 2 1 1 2 2 2 2 2 3 3 3 3 3 3 3
## attr(,"prob")
##  [1] 0.8333333 0.8333333 0.8333333 0.6666667 0.6666667 0.8333333 0.5000000
##  [8] 0.8333333 0.6666667 0.6666667 0.5000000 0.5000000 1.0000000 0.8333333
## [15] 1.0000000 0.8333333 0.6666667 0.6666667
## Levels: 1 2 3
```

```
table(knn.train)
```

```
## knn.train
## 1 2 3
## 5 6 7
```

```
help(knn)

# Error on train data
(tt <- table(knn.train, cl))
```

```
##          cl
## knn.train 1 2 3
##         1 5 0 0
##         2 1 5 0
##         3 0 1 6
```

```
1-sum(diag(tt))/sum(tt)
```

```
## [1] 0.1111111
```

We spilt the data so that 50% was in train and 50% was in test. The square root of 36 (the number of observations) equaled 6. Therefore, we used 6 clusters for the nearest neighbor method. We created predictions using the train model against the training data and produced it's corresponding classification table. After, we found the misclassification error on train to be 11.11%

```
knn.test <- knn(train=train, test=test, cl=cl, k=6, prob=F); knn.test
```

```
##  [1] 1 2 1 1 1 1 2 3 2 1 2 2 3 3 3 3 3 2
## Levels: 1 2 3
```

```
table(knn.test)
```

```
## knn.test
## 1 2 3
## 6 6 6
```

```
## error on test data
(tt <- table(knn.test, cl))
```

```
##         cl
## knn.test 1 2 3
##        1 5 1 0
##        2 1 4 1
##        3 0 1 5
```

```
1-sum(diag(tt))/sum(tt)
```

```
## [1] 0.2222222
```

We created predictions using the train model against the testing data and produced it's corresponding classification table. After, we found the misclassification error on test to be 22.22%.

With equal probability in train and test, the nearest neighbor method seemed to better predict the train data than the holdout method, because it produced a lower misclassification error on test.