RSM8512 Assignment - Classification

Yanbing Chen 1009958752 2023/10/19

Question 3 [20 marks] (Hint: It might be easier to solve Question #2 from ISL first)

This problem relates to the QDA model, in which the observations within each class are drawn from a normal distribution with a class- specific mean vector and a class specific covariance matrix. We consider the simple case where p = 1; i.e. there is only one feature.

Suppose that we have K classes, and that if an observation belongs to the kth class then X comes from a one-dimensional normal distribution, $X - N(\mu_k, \sigma_k^2)$. Recall that the density function for the one-dimensional normal distribution is given in (4.11). Prove that in this case, the Bayes' classifier is not linear. Argue that it is in fact quadratic.

Hint: For this problem, you should follow the arguments laid out in Section 4.4.2, but without making the assumption that $\sigma_1^2 = ... = \sigma_K^2$.

Answer: We have the Bayes Classifier is $P_r(Y = k|X = x) = \frac{\pi_k f_k(x)}{\Sigma_{l=1}^k \pi_l f_l(x)}$ as equation (1) and normal density function $f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} exp(-\frac{1}{2\sigma_k^2}(x - \pi_k))^2$ as equation(2). By plugging (2) into (1), we has equation (3):

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma_k} exp(-\frac{1}{2\sigma_k^2}(x - \mu_k)^2)}{\Sigma_{l=1}^k \pi_l \frac{1}{\sqrt{2\pi}\sigma_l} exp(-\frac{1}{2\sigma_l^2}(x - \mu_l)^2)}$$

.

For QDA model, it assumes that each class has its own covariance matrix, so $\sigma_1 \neq \sigma_2 \neq ... \neq \sigma_k$. After taking log of (3) and rearranging the terms and rearranging terms, we have

$$logp_k(x) = \frac{log(\pi_k) + log(\frac{1}{\sqrt{2\pi}\sigma_k}) + (-\frac{1}{2\sigma_k^2}(x - \mu_k)^2)}{log(\Sigma_{l=1}^k \pi_l \frac{1}{\sqrt{2\pi}\sigma_l} exp(-\frac{1}{2\sigma_l^2}(x - \mu_l)^2))}$$

Since $log(\Sigma_{l=1}^k \pi_l \frac{1}{\sqrt{x\pi}\sigma_l} exp(-\frac{1}{2\sigma_l^2}(x - \mu_l)^2))$ is a constant, and it will not make effect on the function, therefore we have:

$$\delta_k(x) = log(\pi_k) + log(\frac{1}{\sqrt{2\pi}\sigma_k}) + (-\frac{1}{2\sigma_k^2}(x - \mu_k)^2)$$

As you can see from the equation above, $\delta_k(x)$ is a quandratic function of x, so the Bayes' classifier is not linear.

Question 4 [20marks] a-d from ISL. (5 BONUS marks: 4e)

When the number of features p is large, there tends to be a deterioration in the performance of KNN and other local approaches that perform prediction using only observations that are near the test observation for which a prediction must be made. This phenomenon is known as the curse of dimensionality, and it ties into the fact that non-parametric approaches often perform poorly when p is large. We will now investigate this curse.

(a) Suppose that we have a set of observations, each with measurements on p = 1 feature, X. We assume that X is uniformly (evenly) distributed on [0,1]. Associated with each observation is a response value. Suppose that we wish to predict a test observation's response using only observations that are within 10% of the range of X closet to that test observation. For instance, in order to predict the response for a test observation with X = 0.6, we will use observations in the range [0.55,0.65]. On average, what fraction of the available observations will we use to make the prediction?

Answer: When we do not include the situation of $X > 0.95$ and $X < 0.05$, the answer is average 10%.

(b) Now suppose that we have a set of observations, each with measurements on p = 2 features, X_1 and X_2. We assume that (X_1,X_2) are uniformly distributed on [0,1]×[0,1]). We wish to predict a test observation's response using only observations that are within 10 % of the range of X_1 and within 10 % of the range of X_2 closest to that test observation. For instance, in order to predict the response for a test observation with X_1 = 0.6 and X_2 = 0.35, we will use observations in the range [0.55, 0.65] for X_1 and in the range [0.3, 0.4] for X2. On average, what fraction of the available observations will we use to make the prediction?

Answer: The fraction of the available observation is $(10\% * 10\%/(1 * 1) = 10\%^2$

(c) Now suppose that we have a set of observations on p = 100 features. Again the observations are uniformly distributed on each feature, and again each feature ranges in value from 0 to 1. We wish to predict a test observation's response using observations within the 10 % of each feature's range that is closest to that test observation. What fraction of the available observations will we use to make the prediction?

Answer: $(10\%^{100})/(1^{100}) = 10^{-100}$

(d) Using your answers to parts (a)–(c), argue that a drawback of KNN when p is large is that there are very few training observations "near" any given test observation.

Answer: When p is increased linear, the fraction is decreased exponentially.

(e) Now suppose that we wish to make a prediction for a test observation by creating a p-dimensional hypercube centered around the test observation that contains, on average, 10 % of the training observations. For p = 1,2, and 100, what is the length of each side of the hypercube? Comment on your answer.

Answer: average fraction $= 10\%^p$

Question 6 [20 marks]

Suppose we collect data for a group of students in a statistics class with variables X1 = hours studied, X2 = undergrad GPA, and Y = receive an A. We fit a logistic regression and produce estimated coefficient, $\hat{\beta}_0 = -6, \hat{\beta}_1 = 0.05, \hat{\beta}_2 = 1$

The probability function is

$$\hat{p}(X) = \frac{e^{\hat{\beta}+\hat{\beta}_1 X_1 + \hat{\beta}_2 X_2}}{1 + e^{\hat{\beta}+\hat{\beta}_1 X_1 + \hat{\beta}_2 X_2}}$$

(a) Estimate the probability that a student who studies for 40 h and has an undergrad GPA of 3.5 gets an A in the class.

```
beta_0 = -6
beta_1 = 0.05
beta_2 = 1
x1 = 40
x2 = 3.5
exp(beta_0 + beta_1*x1+ beta_2*x2)/(1 + exp(beta_0 + beta_1*x1 + beta_2*x2))
```

```
## [1] 0.3775407
```

(b) How many hours would the student in part (a) need to study to have a 50 % chance of getting an A in the class?

Answer: After transformation, we can get $e^{\hat{\beta}+\hat{\beta}_1 X_1 + \hat{\beta}_2 X_2} = 1$, Therefore,from $\hat{\beta} + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 = 0$ we can have $-6 + 0.05x1 + 3.5 * 1 = 0$

```
(0 - beta_0 - beta_2 * x2) / beta_1
```

```
## [1] 50
```

The student need to study 50 hours to have 50% chance of getting an A.

Question 14 [40 marks]

In this problem, you will develop a model to predict whether a given car gets high or low gas mileage based on the `Auto` data set.

(a) Create a binary variable, `mpg01`, that contains a 1 if `mpg` contains a value above its median, and a 0 if `mpg` contains a value below its median. You can compute the median using the median() function. Note you may find it helpful to use the data.frame() function to create a single data set containing both `mpg01` and the other `Auto` variables.

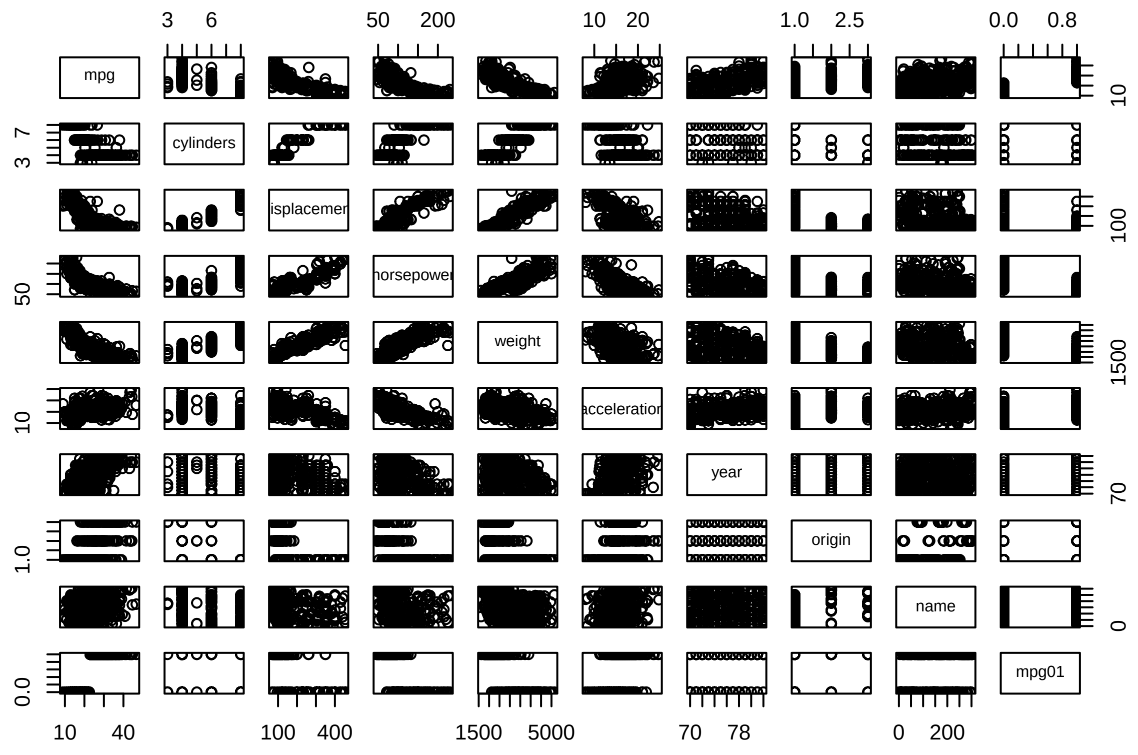
```
data(Auto)
mpg01 = ifelse(Auto$mpg > median(Auto$mpg), 1, 0)
Auto = data.frame(Auto, mpg01)
#head(Auto)
```

(b) Explore the data graphically in order to investigate the association between `mpg01` and the other features. Which of the other features seem most likely to be useful in predicting `mpg01`? Scatterplots and boxplots may be useful tools to answer this question. Describe your findings.

```
cor(Auto[,-9])
```

```
##                     mpg  cylinders displacement horsepower     weight
## mpg           1.0000000 -0.7776175   -0.8051269 -0.7784268 -0.8322442
## cylinders    -0.7776175  1.0000000    0.9508233  0.8429834  0.8975273
## displacement -0.8051269  0.9508233    1.0000000  0.8972570  0.9329944
## horsepower   -0.7784268  0.8429834    0.8972570  1.0000000  0.8645377
## weight       -0.8322442  0.8975273    0.9329944  0.8645377  1.0000000
## acceleration  0.4233285 -0.5046834   -0.5438005 -0.6891955 -0.4168392
## year          0.5805410 -0.3456474   -0.3698552 -0.4163615 -0.3091199
## origin        0.5652088 -0.5689316   -0.6145351 -0.4551715 -0.5850054
## mpg01         0.8369392 -0.7591939   -0.7534766 -0.6670526 -0.7577566
##              acceleration       year     origin      mpg01
## mpg             0.4233285  0.5805410  0.5652088  0.8369392
## cylinders      -0.5046834 -0.3456474 -0.5689316 -0.7591939
## displacement   -0.5438005 -0.3698552 -0.6145351 -0.7534766
## horsepower     -0.6891955 -0.4163615 -0.4551715 -0.6670526
## weight         -0.4168392 -0.3091199 -0.5850054 -0.7577566
## acceleration    1.0000000  0.2903161  0.2127458  0.3468215
## year            0.2903161  1.0000000  0.1815277  0.4299042
## origin          0.2127458  0.1815277  1.0000000  0.5136984
## mpg01           0.3468215  0.4299042  0.5136984  1.0000000
```

```
pairs(Auto)
```



Answer: `displacement`,`horsepower`, `weight` and `cylinders` are good to fit the `mpg01`

(c) Split the data into a training set and a test set.

```
set.seed(1234)

split = sample.split(Auto$mpg01,SplitRatio = 0.7)
train_set = subset(Auto, split == TRUE)
test_set = subset(Auto, split == FALSE)
```

(d) Perform LDA on the training data in order to predict `mpg01` using the variables that seemed most associated with `mpg01` in (b). What is the test error of the model obtained?

```
lda.fit = lda(mpg01 ~ displacement +  horsepower + weight + cylinders, data = train_set)

lda.pred = predict(lda.fit, test_set)
#lda.pred$class
lda_ter = 1 - mean(lda.pred$class == test_set$mpg01)
lda_ter
```

```
## [1] 0.09322034
```

(e) Perform QDA on the training data in order to predict `mpg01` using the variables that seemed most associated with `mpg01` in(b). What is the test error of the model obtained?

```
qda.fit = qda(mpg01 ~ displacement +  horsepower + weight + cylinders, data = train_set)

qda.pred = predict(qda.fit, test_set)
qda_ter = 1 - mean(qda.pred$class == test_set$mpg01)
qda_ter
```

```
## [1] 0.1271186
```

(f) Perform logistic regression on the training data in order to predict `mpg01` using the variables that seemed most associated with `mpg01` in (b). What is the test error of the model obtained?

```
glm.fit = glm(mpg01 ~ displacement +  horsepower + weight + cylinders, data = train_set, family
#summary(glm.fit)

glm.probs = predict(glm.fit, data = test_set,type = 'response')
glm.pred = rep(0, length(glm.probs))
glm.pred[glm.probs > 0.5] = 1
glm_ter = 1 - mean(glm.pred == test_set$mpg01)
```

```
## Warning in glm.pred == test_set$mpg01: longer object length is not a multiple of
## shorter object length
```

```
glm_ter # logistic regression model's test_error
```

```
## [1] 0.4963504
```

(g) Perform naive Bayes on the training data in order to predict `mpg01` using the variables that seemed most associated with `mpg01` in (b). What is the test error of the model obtained?

```
library(e1071)
nb.fit = naiveBayes(mpg01 ~ displacement +  horsepower + weight + cylinders, data = train_set)
nb.pred = predict(nb.fit, test_set)
nb_ter = 1 - mean(nb.pred ==  test_set$mpg01)
nb_ter # naive Bayes's test_error
```

```
## [1] 0.1101695
```

(h) Perform KNN on the training data, with several values of K, in order to predict `mpg01`. Use only the variables that seemed most associated with `mpg01` in (b). What test errors do you obtain? Which value of K seems to perform the best on this data set?

```
train.X = train_set[, c('cylinders', 'weight', 'displacement', 'horsepower')]
test.X =  train_set[, c('cylinders', 'weight', 'displacement', 'horsepower')]
train_mpg01 = train_set[,'mpg01']
test_mpg01 = test_set[, 'mpg01']

set.seed(1234)
```

```
# when k = 1
knn.pred = knn(train.X, test.X, train_mpg01, test_mpg01, k = 1)
knn_ter1 = 1 - mean(knn.pred == test_mpg01 )
```

```
## Warning in `==.default`(knn.pred, test_mpg01): longer object length is not a
## multiple of shorter object length
```

```
## Warning in is.na(e1) | is.na(e2): longer object length is not a multiple of
## shorter object length
```

```
knn_ter1
```

```
## [1] 0.4817518
```

```
# when k = 10
knn.pred = knn(train.X, test.X, train_mpg01, test_mpg01, k = 10)
knn_ter10 = 1 - mean(knn.pred == test_mpg01 )
```

```
## Warning in `==.default`(knn.pred, test_mpg01): longer object length is not a
## multiple of shorter object length

## Warning in is.na(e1) | is.na(e2): longer object length is not a multiple of
## shorter object length
```

```
knn_ter10
```

```
## [1] 0.5072993
```

```
# when k = 100
knn.pred = knn(train.X, test.X, train_mpg01, test_mpg01, k = 100)
knn_ter100 = 1 - mean(knn.pred == test_mpg01 )
```

```
## Warning in `==.default`(knn.pred, test_mpg01): longer object length is not a
## multiple of shorter object length

## Warning in is.na(e1) | is.na(e2): longer object length is not a multiple of
## shorter object length
```

```
knn_ter100
```

```
## [1] 0.4963504
```