hw4_starter.R (Problem 4 and Problem 5)

Erica Zhou

Nov 30, 2022

Problem 4

```
rm(list = ls())  # remove the existing environment

## You should set the working directory to the folder of hw3_starter by
## uncommenting the following and replacing YourDirectory by what you have
## in your local computer / labtop

setwd("~/STA314/sta314-hw4")

## Load utils.R and discriminant_analysis.R

source("utils.R")
source("discriminant_analysis.R")


## Load the training and test data
train <- Load_data("digits_train.txt")
```

```
## Rows: 7000 Columns: 65
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (65): X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, ...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
test <- Load_data("digits_test.txt")
```

```
## Rows: 4000 Columns: 65
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (65): X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, ...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
x_train <- train$x
y_train <- train$y

x_test <- test$x
y_test <- test$y
```

```
#####################################################################
#                          Part a.                                  #
# TODO:  estimate the priors, conditional means and conditional     #
#        covariance matrices under LDA,                             #
#        predict the labels of test data by using the fitted LDA    #
#        compute its misclassification error rate                   #
#####################################################################

priors <- Comp_priors(y_train)
means <- Comp_cond_means(x_train,y_train)
covs <- Comp_cond_covs(x_train,y_train, TRUE)
post <- Predict_posterior(x_test,priors,means,covs,TRUE)
post_label <- Predict_labels(post)

# the error rate is
mean(y_test != post_label)
```

```
## [1] 0.90825
```

```
#####################################################################
#                       END OF YOUR CODE                            #
#####################################################################



#####################################################################
#                          Part b.                                  #
# TODO:  estimate the priors, conditional means and conditional     #
#        covariance matrices under QDA,                             #
#        predict the labels of test data by using the fitted LDA    #
#        compute its misclassification error rate                   #
#####################################################################

priors <- Comp_priors(y_train)
means <- Comp_cond_means(x_train,y_train)
covs <- Comp_cond_covs(x_train,y_train, FALSE)
post <- Predict_posterior(x_test,priors,means,covs,FALSE)
post_label <- Predict_labels(post)

# the error rate is
mean(y_test != post_label)
```

```
## [1] 0.9025
```

```
#####################################################################
#                       END OF YOUR CODE                            #
#####################################################################



#####################################################################
#                          Part c.                                  #
# TODO:  fit LDA and QDA by using the R package                     #
#        report their test errors                                   #
```

```
#########################################################################

library(MASS)

lda.fit <- lda(y ~ x, data = train)
lda.pred <- predict(lda.fit, test)$class
mean(lda.pred != test$y)
```

```
## [1] 0.10225
```

```
qda.fit <- qda(y ~ x, data = train)
qda.pred <- predict(qda.fit, test)$class
mean(qda.pred != test$y)
```

```
## [1] 0.04075
```

```
#########################################################################
#                        END OF YOUR CODE                               #
#########################################################################
```

Problem 5

```
#head(Boston)
```

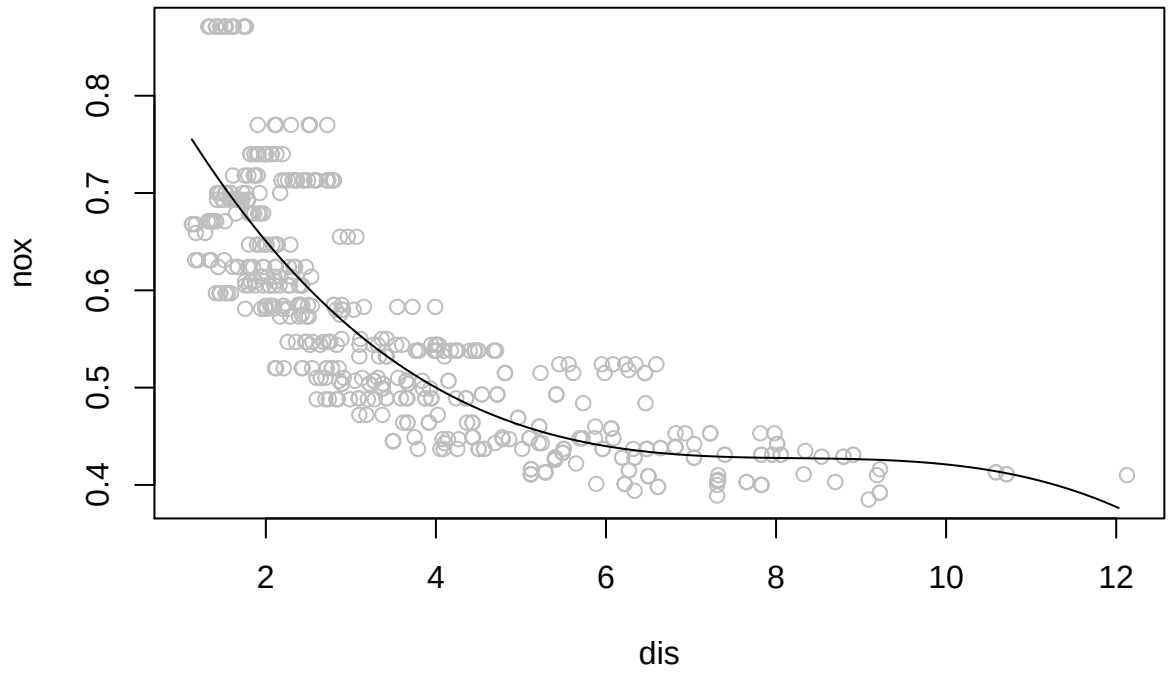
1.

```
set.seed(1)
poly.fit <- glm(nox ~ poly(dis, 3), data = Boston)
summary(poly.fit)
```

```
##
## Call:
## glm(formula = nox ~ poly(dis, 3), data = Boston)
##
## Deviance Residuals:
##       Min         1Q     Median         3Q        Max
## -0.121130  -0.040619  -0.009738   0.023385   0.194904
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.554695   0.002759 201.021  < 2e-16 ***
## poly(dis, 3)1 -2.003096   0.062071 -32.271  < 2e-16 ***
## poly(dis, 3)2  0.856330   0.062071  13.796  < 2e-16 ***
## poly(dis, 3)3 -0.318049   0.062071  -5.124 4.27e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.003852802)
##
##     Null deviance: 6.7810  on 505  degrees of freedom
## Residual deviance: 1.9341  on 502  degrees of freedom
## AIC: -1370.9
##
## Number of Fisher Scoring iterations: 2
```

```
dislims <- range(Boston$dis)
dis.grid <- seq(from = dislims[1], to = dislims[2], by = .1)
```

```
pred.value <- predict(poly.fit, list(dis = dis.grid))
plot(nox ~ dis, data = Boston, col = "grey")
lines(dis.grid, pred.value)
```



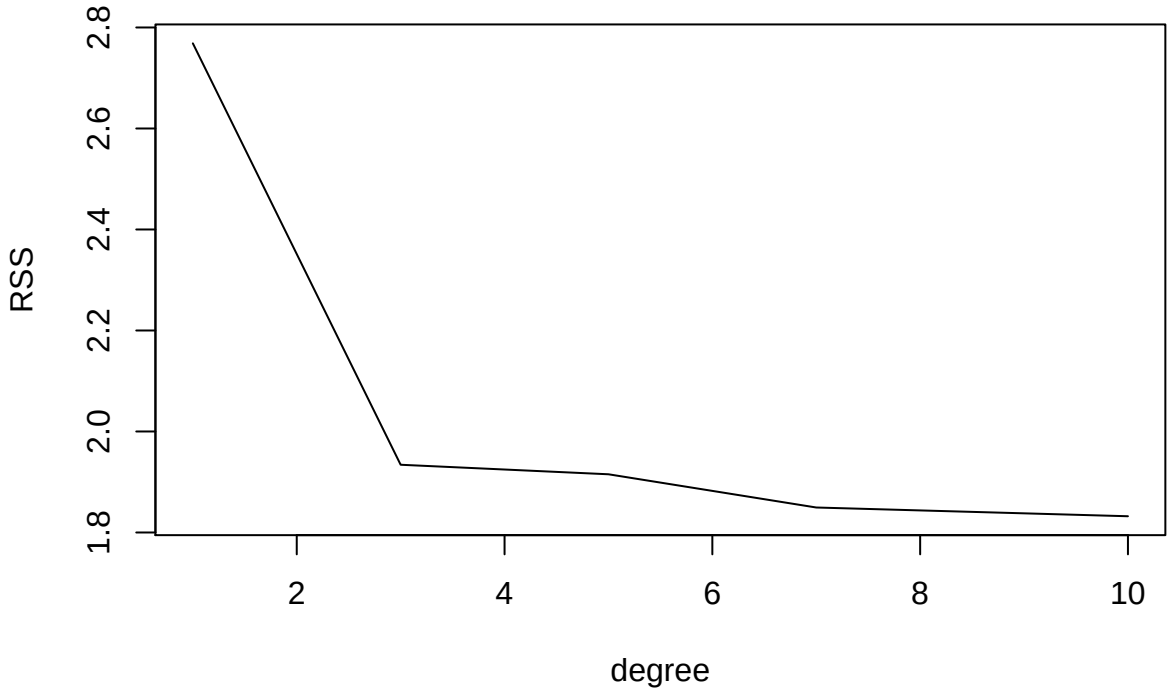
2.

```
rss <- rep(0, 10)
for (i in 1:10) {
  poly.fit <- lm(nox ~ poly(dis, i), data = Boston)
  rss[i] <- sum(poly.fit$residuals^2)
}

rss[c(1,3,5,7,10)]
```

```
## [1] 2.768563 1.934107 1.915290 1.849484 1.832171
```

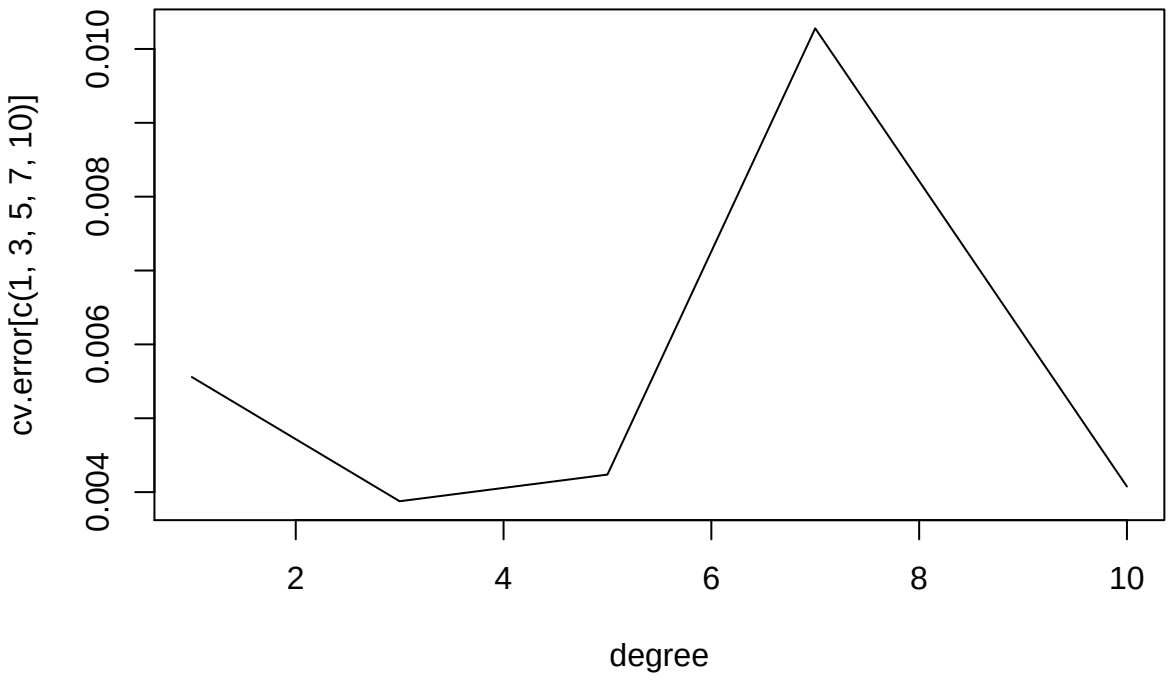
```
plot(c(1,3,5,7,10),rss[c(1,3,5,7,10)], xlab = "degree", ylab = "RSS", type = "l")
```



3.

```
set.seed(1)
#install.packages("ISLR")
library(ISLR)
library(boot)
cv.error <- rep(0,10)
for (i in 1:10) {
  poly.fit <- glm(nox ~ poly(dis, i), data = Boston)
  cv.error[i] <- cv.glm(Boston, poly.fit, K=10)$delta[1]
}

plot(c(1,3,5,7,10),cv.error[c(1,3,5,7,10)],xlab = "degree",type = "l")
```




```
#index not value
which.min(cv.error[c(1,3,5,7,10)])
```

```
## [1] 2
```

Because the cv error is the lowest when the polynomial degree is 3, we should choose degree 3 (for this seed).

4.

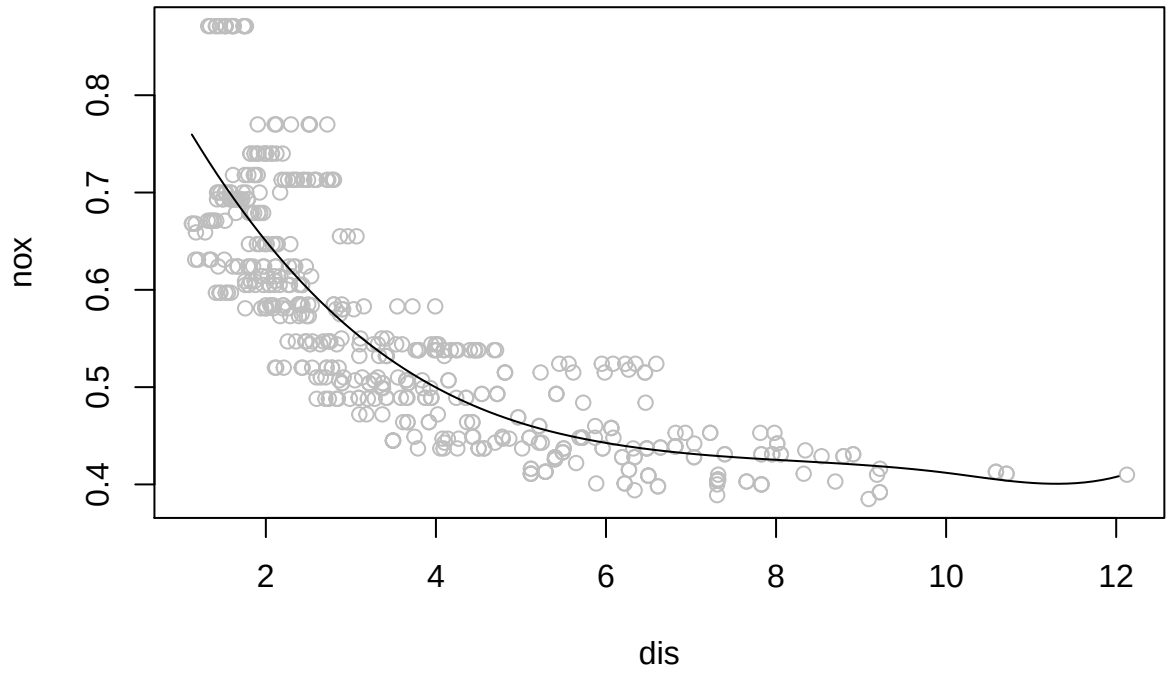
```
library(splines)
bs.fit <- glm(nox ~ bs(dis, knots = c(6, 10)), data = Boston)
summary(bs.fit)
```

```
##
## Call:
## glm(formula = nox ~ bs(dis, knots = c(6, 10)), data = Boston)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.12352  -0.04028  -0.01029   0.02302   0.19445
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  0.75963    0.01040  73.069  < 2e-16 ***
## bs(dis, knots = c(6, 10))1 -0.23407    0.02434  -9.617  < 2e-16 ***
## bs(dis, knots = c(6, 10))2 -0.34115    0.02107 -16.189  < 2e-16 ***
## bs(dis, knots = c(6, 10))3 -0.33132    0.03560  -9.307  < 2e-16 ***
## bs(dis, knots = c(6, 10))4 -0.36815    0.05393  -6.826 2.53e-11 ***
## bs(dis, knots = c(6, 10))5 -0.34878    0.06303  -5.533 5.08e-08 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.003861197)
##
##     Null deviance: 6.7810  on 505  degrees of freedom
## Residual deviance: 1.9306  on 500  degrees of freedom
## AIC: -1367.8
##
## Number of Fisher Scoring iterations: 2
```

From the plot in (1), we see that are probably some change in trend at dis = 6 and dis = 10, thus, I choose these two numbers as knots. The resulting fit is:

```
pred <- predict(bs.fit, list(dis = dis.grid))
plot(nox ~ dis, data = Boston, col = "grey")
lines(dis.grid, pred)
```

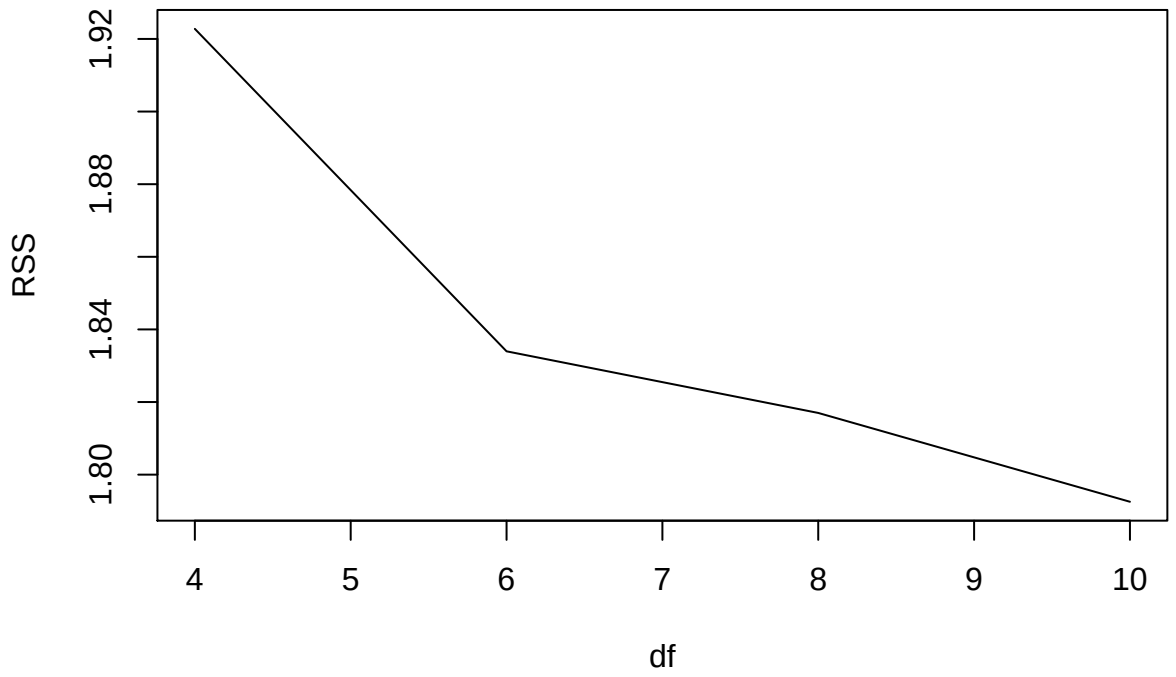


5.

```
rss <- rep(0, 10)
for (i in 4:10) {
  bs.fit <- glm(nox ~ bs(dis, df = i), data = Boston)
  rss[i] <- sum(bs.fit$residuals^2)
}
rss[c(4,6,8,10)]
```

```
## [1] 1.922775 1.833966 1.816995 1.792535
```

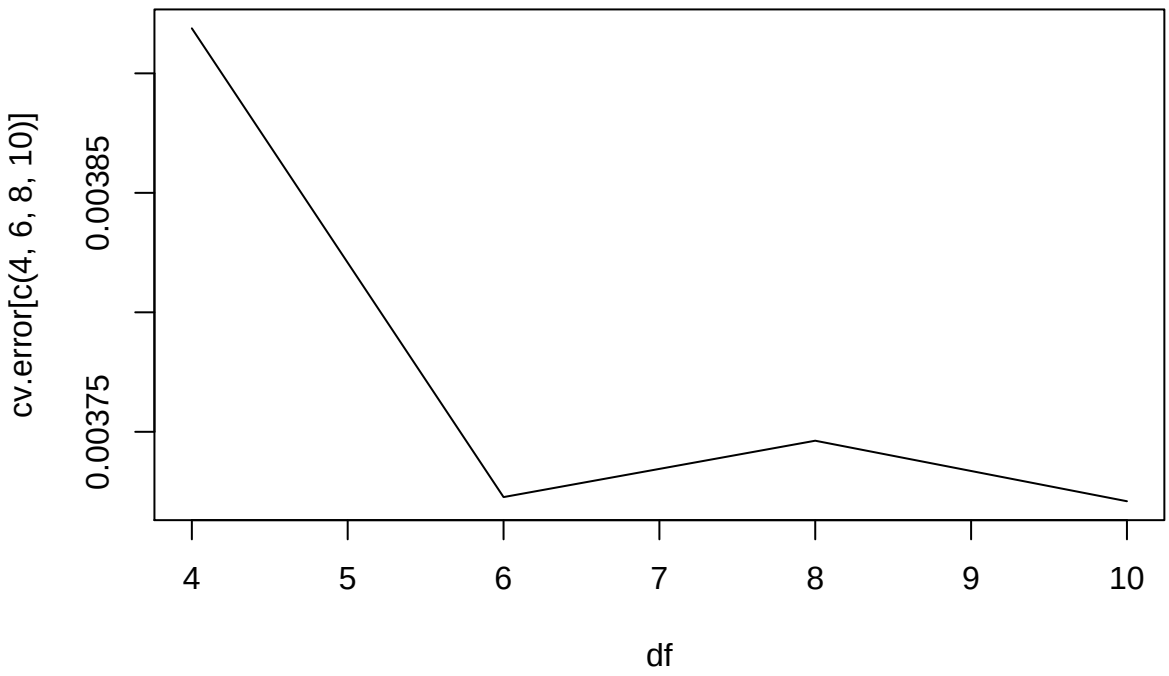
```
plot(c(4,6,8,10),rss[c(4,6,8,10)],xlab = "df", ylab = "RSS", type = "l" )
```



We see that the RSS keeps dropping as the degree of freedom increases when we focus on df = 4,6,8,10. 6.

```
options(warn=-1)
set.seed(1)
cv.error <- c()
for (i in 4:10) {
  bs.fit <- glm(nox ~ bs(dis, df = i), data = Boston)
  cv.error[i] <- cv.glm(Boston, bs.fit, K = 10)$delta[1]
}

plot(c(4,6,8,10),cv.error[c(4,6,8,10)],xlab = "df", type = "l")
```




```
#index not value
which.min(cv.error[c(4,6,8,10)])
```

```
## [1] 4
```

I start from df = 4 because it will be too small if it's less than 3. For this seed, the cv error achieve its minimum at df = 10 (index = 4), and thus, we may choose df = 10 for the basic spline regression.