Lda-Qda

Chase Enzweiler

10/30/2017

LDA and QDA Functions

```
library(MASS)
```

```
## Warning: package 'MASS' was built under R version 3.3.2
```

```
library(mvtnorm)
```

```
## Warning: package 'mvtnorm' was built under R version 3.3.2
```

LDA

Implement an lda function that computes the necessary estimates for LDA

```
# function for computing LDA estimates
###########################################################################
# input: X predictor matrix nxp, y response matrix a factor length n

# output: list(pi_hat : prior prob vector length K, mu_hat : Kxp matrix where each row contains mean of

my_lda <- function(X, y){

  # number of groups/classes

  K <- length(levels(y))

  # number of observations
  n <- dim(X)[1]

  # number of predictors
  p <- dim(X)[2]

  # column bind response with predictors
  yX_mat <- cbind(y, X)

  # calculate the prior probability vector
  pi_hat <- c()

  for (i in 1:K){

    pi_hat[i] <- dim(yX_mat[yX_mat[,1] == levels(y)[i],])[1]

  }
```

```r
  pi_hat <- (pi_hat / n)

  # calculate the group mean matrix k x p, where its group mean for each predictor

  mu_hat <- matrix(0, ncol = p, nrow = K)

  for (i in 1:K){

    mu_hat[i,] <- colMeans(yX_mat[yX_mat[,1] == levels(y)[i],-1])

  }

  rownames(mu_hat) <- levels(y)

  colnames(mu_hat) <- colnames(X)

  # calculate the pxp covariance matrix of predictors

  sums <- 0

  for (k in 1:K){

    # the x_i
    sub_mat <- as.matrix(yX_mat[yX_mat[,1] == levels(y)[k], -1] )

    first_sum <- 0

    for (i in 1:nrow(sub_mat)){

      first_sum <- first_sum + (sub_mat[i,] - mu_hat[k,]) %*% t( (sub_mat[i,]) - mu_hat[k,])

    }

    sums <- sums + first_sum

  }

  sigma_hat <- sums / (n - K)


  return(list("pi_hat" = pi_hat,"mu_hat" = mu_hat, "sigma_hat" =  sigma_hat, levels_order = levels(y)))

}
```

Implement a function called predict_my_lda() that generates predictions based on the output from my_lda()

```r
# function predict_my_lda
predict_my_lda <- function(fit, newdata){

  # input: fit output from my_lda(), newdata mxp matrix of new observations(assuming no response column
  # output: list(class : length m factor vector each elements indicate the predicted class of observati
```

```r
  # number of observations of newdata
  m <- dim(newdata)[1]

  # number of groups
  K <- length(fit$pi_hat)

  # calculate the posterior probabilities

  posterior <- matrix(0, nrow = m, ncol = K)

  for (i in 1:m){

    denominator <- 0

    for (k in 1:K){

      #bayes denominator

      f_l <- dmvnorm(x = newdata[i,], mean = fit$mu_hat[k,], sigma = fit$sigma_hat)

      denominator <- denominator + (f_l * fit$pi_hat[k])
    }

    # now can calculate the posterior probabilities

    for (k in 1:K){

      posterior[i,k] <- (dmvnorm(x = newdata[i,], mean = fit$mu_hat[k,], sigma = fit$sigma_hat) * fit$pi


    }

  }

  # posterior is matrix of probabilities for each observation that they would be in a given class. Clas
  colnames(posterior) <- fit$levels_order

  # class, make a length m factor vector of the predicted class

  class <- c()

  # find predicted class for each new observation
  for (i in 1:m){

    # find which col aka which class has highest posterior probability
    index <- which.max(posterior[i,])

    class[i] <- fit$levels_order[index]
  }

  return(list("posterior" = posterior, "predicted_class" = factor(class)))

}
```

Train your LDA on the first 140 observations of iris and predict the last 10 observations

```r
# train and predict on iris
my_lda_fit <- my_lda(iris[1:140, -5], iris$Species[1:140])

predict_my_lda(my_lda_fit, iris[141:150,-5])
```

```
## $posterior
##             setosa   versicolor virginica
##  [1,] 1.822023e-43 2.360129e-06 0.9999976
##  [2,] 1.204284e-34 8.851349e-04 0.9991149
##  [3,] 1.002964e-36 1.618792e-03 0.9983812
##  [4,] 2.289667e-44 1.633764e-06 0.9999984
##  [5,] 1.027581e-44 5.095900e-07 0.9999995
##  [6,] 1.184605e-37 1.553062e-04 0.9998447
##  [7,] 1.098815e-34 9.868582e-03 0.9901314
##  [8,] 7.724661e-34 4.664455e-03 0.9953355
##  [9,] 2.353301e-39 2.112746e-05 0.9999789
## [10,] 2.848375e-32 2.112626e-02 0.9788737
##
## $predicted_class
##  [1] virginica virginica virginica virginica virginica virginica virginica
##  [8] virginica virginica virginica
## Levels: virginica
```

```r
# compare it to lda()

lda_fit <- lda(Species ~ ., data = iris[1:140,])

predict(lda_fit, newdata = iris[141:150,])
```

```
## $class
##  [1] virginica virginica virginica virginica virginica virginica virginica
##  [8] virginica virginica virginica
## Levels: setosa versicolor virginica
##
## $posterior
##           setosa   versicolor virginica
## 141 1.822023e-43 2.360129e-06 0.9999976
## 142 1.204284e-34 8.851349e-04 0.9991149
## 143 1.002964e-36 1.618792e-03 0.9983812
## 144 2.289667e-44 1.633764e-06 0.9999984
## 145 1.027581e-44 5.095900e-07 0.9999995
## 146 1.184605e-37 1.553062e-04 0.9998447
## 147 1.098815e-34 9.868582e-03 0.9901314
## 148 7.724661e-34 4.664455e-03 0.9953355
## 149 2.353301e-39 2.112746e-05 0.9999789
## 150 2.848375e-32 2.112626e-02 0.9788737
##
## $x
##           LD1         LD2
## 141 -6.941596  1.91004258
## 142 -5.400922  2.01138715
## 143 -5.815751  0.07968611
## 144 -7.105066  1.67329123
```

```
## 145 -7.141806  2.55679939
## 146 -5.933464  1.70985582
## 147 -5.470291 -0.31886548
## 148 -5.288619  0.95110588
## 149 -6.208771  2.50152274
## 150 -5.025815  0.52177854
```

QDA

Implement a function called my_qda() that computes the necessary estimates for QDA.

```
# function for the qda estimates
# input: X: the predictor matrix, which is an n × p matrix
#- y: the response vector, which is a factor vector of length n

# output: list(- pi_hat: the prior probability vector, which is a vector of length K - mu_hat: a K × p

my_qda <- function(X, y){

  # number of groups/classes

  K <- length(levels(y))

  # number of observations
  n <- dim(X)[1]

  # number of predictors
  p <- dim(X)[2]

  # column bind response with predictors
  yX_mat <- cbind(y, X)

  # calculate the prior probability vector
  pi_hat <- c()

  for (i in 1:K){

    pi_hat[i] <- dim(yX_mat[yX_mat[,1] == levels(y)[i],])[1]

  }

  pi_hat <- (pi_hat / n)

  # calculate the group mean matrix k x p, where its group mean for each predictor

  mu_hat <- matrix(0, ncol = p, nrow = K)

  for (i in 1:K){

    mu_hat[i,] <- colMeans(yX_mat[yX_mat[,1] == levels(y)[i],-1])

  }
```

```r
  rownames(mu_hat) <- levels(y)

  colnames(mu_hat) <- colnames(X)

  # now find the p × p × K array, where sigma_hat[,,k] contains the covariance matrix of the predictors

  # array to store each groups covariance matrix
  sigma_hat <- array(0, dim = c(p,p,K), dimnames = list(colnames(X), colnames(X)))

  # observaations in group k
  n_k <- pi_hat * n

  # iterate through each group
  for (k in 1:K){

    # the x_i
    sub_mat <- as.matrix(yX_mat[yX_mat[,1] == levels(y)[k], -1] )

    # store the addition of covariance matrix for each obs.
    sums <- 0

    # iterate through each obs
    for (i in 1:nrow(sub_mat)){

       sums <- sums + (1/(n_k[k] - 1)) * (sub_mat[i,] - mu_hat[k,]) %*% t( (sub_mat[i,]) - mu_hat[k,])

    }


    sigma_hat[,,k] <- sums

  }


  return(list("pi_hat" = pi_hat, "mu_hat" = mu_hat, "sigma_hat" = sigma_hat, "levels_order" = levels(y))


}
```

Implement a function called predict_my_qda() that generates predictions based on the output from my_qda().

```r
# function to generate predictions for my_qda function

# input
# - fit: the output from my_qda()
# - newdata: a m × p matrix of new observations

# output
# list(- class: a length-m factor vector; each of its elements indicate the predicted class of an obser

predict_my_qda <- function(fit, newdata){
```

```r
  # number of observations of newdata
  m <- dim(newdata)[1]

  # number of groups
  K <- length(fit$pi_hat)

  # calculate the posterior probabilities

  posterior <- matrix(0, nrow = m, ncol = K)

  for (i in 1:m){

    denominator <- 0

    for (k in 1:K){

      #bayes denominator

      f_l <- dmvnorm(x = newdata[i,], mean = fit$mu_hat[k,], sigma = fit$sigma_hat[,,k])

      denominator <- denominator + (f_l * fit$pi_hat[k])
    }

    # now can calculate the posterior probabilities

    for (k in 1:K){

      posterior[i,k] <- (dmvnorm(x = newdata[i,], mean = fit$mu_hat[k,], sigma = fit$sigma_hat[,,k]) * f


    }

  }

  # posterior is matrix of probabilities for each observation that they would be in a given class. Clas
  colnames(posterior) <- fit$levels_order

  # class, make a length m factor vector of the predicted class

  class <- c()

  # find predicted class for each new observation
  for (i in 1:m){

    # find which col aka which class has highest posterior probability
    index <- which.max(posterior[i,])

    class[i] <- fit$levels_order[index]
  }

  return(list("posterior" = posterior, "predicted_class" = factor(class)))

}
```

Train your QDA on the first 140 observations in the dataset iris and predict the last 10 observations.

```r
# train and predict on iris
my_qda_fit <- my_qda(iris[1:140, -5], iris$Species[1:140])

predict_my_qda(my_qda_fit, iris[141:150, -5])
```

```
## $posterior
##               setosa   versicolor virginica
##  [1,] 1.593400e-174 2.124111e-09 1.0000000
##  [2,] 1.657172e-144 4.562809e-08 1.0000000
##  [3,] 7.217888e-126 5.351414e-04 0.9994649
##  [4,] 9.559272e-184 1.278474e-06 0.9999987
##  [5,] 9.198115e-184 3.512176e-10 1.0000000
##  [6,] 5.455780e-150 1.315944e-08 1.0000000
##  [7,] 3.404338e-124 3.143837e-04 0.9996856
##  [8,] 1.323189e-133 1.767812e-03 0.9982322
##  [9,] 2.679955e-155 1.731190e-06 0.9999983
## [10,] 8.559298e-119 7.284787e-02 0.9271521
##
## $predicted_class
##  [1] virginica virginica virginica virginica virginica virginica virginica
##  [8] virginica virginica virginica
## Levels: virginica
```

```r
# confirm answers with qda()

qda_fit <- qda(Species ~ ., data = iris[1:140,])

predict(qda_fit, newdata = iris[141:150,])
```

```
## $class
##  [1] virginica virginica virginica virginica virginica virginica virginica
##  [8] virginica virginica virginica
## Levels: setosa versicolor virginica
##
## $posterior
##             setosa   versicolor virginica
## 141 1.593400e-174 2.124111e-09 1.0000000
## 142 1.657172e-144 4.562809e-08 1.0000000
## 143 7.217888e-126 5.351414e-04 0.9994649
## 144 9.559272e-184 1.278474e-06 0.9999987
## 145 9.198115e-184 3.512176e-10 1.0000000
## 146 5.455780e-150 1.315944e-08 1.0000000
## 147 3.404338e-124 3.143837e-04 0.9996856
## 148 1.323189e-133 1.767812e-03 0.9982322
## 149 2.679955e-155 1.731190e-06 0.9999983
## 150 8.559298e-119 7.284787e-02 0.9271521
```

Confusion Matrix

```r
# training and test set
set.seed(100)
train_idx <- sample(nrow(iris), 90)
train_set <- iris[train_idx, ]
test_set <- iris[-train_idx, ]
```

Train LDA and QDA based on train_set

```r
# train lda and qda
lda_train <- my_lda(train_set[,-5], train_set$Species)

qda_train <- my_qda(train_set[,-5], train_set$Species)
```

Generate predictions on test_set

```r
# predict lda and qda
lda_predict <- predict_my_lda(lda_train, newdata = test_set[,-5])

qda_predict <- predict_my_qda(qda_train, newdata = test_set[,-5])
```

Compute the confusion matrix for each method.

```r
# confusion matrix

# lda confusion matrix
setosa_count <- dim(test_set[test_set[,5] == "setosa",])[1]

versicolor_count <- dim(test_set[test_set[,5] == "versicolor",])[1]

virginica_count <- dim(test_set[test_set[,5] == "virginica",])[1]

lda_set_count <- length(lda_predict$predicted_class[lda_predict$predicted_class == "setosa"])

lda_versi_count <- length(lda_predict$predicted_class[lda_predict$predicted_class == "versicolor"])

lda_virg_count <- length(lda_predict$predicted_class[lda_predict$predicted_class == "virginica"])


lda_confusion <- matrix(diag(c(lda_set_count, lda_versi_count, lda_virg_count, 0)), ncol = 4, nrow = 4,

lda_confusion[,4] <- c(24, 17, 19, 60)
lda_confusion[4,] <- c(24, 18, 18, 60)

# qda confusion matrix

qda_set_count <- length(qda_predict$predicted_class[qda_predict$predicted_class == "setosa"])
```

```r
qda_versi_count <- length(qda_predict$predicted_class[qda_predict$predicted_class == "versicolor"])

qda_virg_count <- length(qda_predict$predicted_class[qda_predict$predicted_class == "virginica"])

qda_confusion <- matrix(diag(c(qda_set_count, qda_versi_count, qda_virg_count, 0)), ncol = 4, nrow = 4,

qda_confusion[,4] <- c(24, 17, 19, 60)
qda_confusion[4,] <- c(24, 18, 18, 60)

# LDA Confusion Matrix
data.frame(lda_confusion)
```

```
##                 predict.setosa predict.versicolor predict.virginica total
## True setosa                 24                  0                 0    24
## True versicolor              0                 18                 0    17
## True virginica               0                  0                18    19
## total                       24                 18                18    60
```

```r
# QDA confusion Matrix
data.frame(qda_confusion)
```

```
##                 predict.setosa predict.versicolor predict.virginica total
## True setosa                 24                  0                 0    24
## True versicolor              0                 18                 0    17
## True virginica               0                  0                18    19
## total                       24                 18                18    60
```