154Lab9

Jiyoon Clover Jeong

10/30/2017

LDA

```
my_lda <- function(X , y){


  K <- nlevels(y)
  n <- length(y)
  p <- dim(X)[2]

  splited <- split(X,y)
  pi_hat <- sapply(splited, nrow) / n
  #pi_hat

  mu_hat <- t(sapply(splited, colMeans))
  #class(mu_hat)
  #mu_hat


  sigma_hat <- matrix(0,p,p)

  for(i in 1:K){

    J <- dim(splited[[i]])[1]
    for(j in 1:J){
      xi <-as.matrix(splited[[i]][j, , drop = F])
      xi
      sigma_hat <- sigma_hat + t(xi - mu_hat[i,]) %*% (xi - mu_hat[i,])
      sigma_hat
    }

  }
  sigma_hat <- (1 / (n - K)) * sigma_hat
  sigma_hat

  return(list(pi_hat = pi_hat, mu_hat = mu_hat, sigma_hat = sigma_hat))


}


mylda <- my_lda(iris[1:140,1:4], iris[1:140,5])
lda_default <- lda(Species ~ ., data = iris[1:140,])

mylda$pi_hat
```

```
##     setosa versicolor  virginica
```

```
## 0.3571429 0.3571429 0.2857143
```

```r
lda_default$prior
```

```
##     setosa versicolor  virginica
## 0.3571429  0.3571429  0.2857143
```

```r
mylda$mu_hat
```

```
##            Sepal.Length Sepal.Width Petal.Length Petal.Width
## setosa           5.0060       3.428       1.4620       0.246
## versicolor       5.9360       2.770       4.2600       1.326
## virginica        6.6225       2.960       5.6075       1.990
```

```r
lda_default$means
```

```
##            Sepal.Length Sepal.Width Petal.Length Petal.Width
## setosa           5.0060       3.428       1.4620       0.246
## versicolor       5.9360       2.770       4.2600       1.326
## virginica        6.6225       2.960       5.6075       1.990
```

```r
mylda$sigma_hat
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length   0.27294270  0.09738394   0.17311423  0.03823650
## Sepal.Width    0.09738394  0.11884526   0.05682628  0.03123066
## Petal.Length   0.17311423  0.05682628   0.18806971  0.04520000
## Petal.Width    0.03823650  0.03123066   0.04520000  0.03909781
```

```r
# https://www.quora.com/Mathematical-Modeling-How-are-posterior-probabilities-calculated-in-linear-disc


predict_my_lda <- function(fit, newdata){

  dmvnormm <- data.frame()
  m <- dim(newdata)[1]
  K <- dim(fit$mu_hat)[1]

  posterior <- matrix(0, m, K)


  for(i in 1:K){
    dmvnormm <- rbind(dmvnormm,dmvnorm(newdata, fit$mu_hat[i,], fit$sigma_hat))
  }

  dmvnormm


  for(i in 1:m){
    numerator <- sum(dmvnormm[,i] * fit$pi_hat)
    for(j in 1:K){
      posterior[i, j] <- dmvnormm[j, i] * fit$pi_hat[j] / numerator

    }
  }

  colnames(posterior) <- names(fit$pi_hat)
```

```r
  posterior

  class <- apply(posterior, 1, function(x)  names(which.max(x)))

  return(list(class = class, posterior = posterior))
}


predictlda <- predict_my_lda(mylda, iris[141:150, -5])
predictlda_default <- predict(lda_default, iris[141:150,])




predictlda_default$class
```

```
##  [1] virginica virginica virginica virginica virginica virginica virginica
##  [8] virginica virginica virginica
## Levels: setosa versicolor virginica
```

```r
predictlda$class
```

```
##  [1] "virginica" "virginica" "virginica" "virginica" "virginica"
##  [6] "virginica" "virginica" "virginica" "virginica" "virginica"
```

```r
# Q : ???
predictlda_default$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
```

```r
predictlda$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
```

QDA

```r
my_qda <- function(X , y){

  n <- length(y)
  p <- dim(X)[2]
  K <- nlevels(y)
  splited <- split(X,y)
  pi_hat <- sapply(splited, nrow) / n
  #pi_hat

  mu_hat <- t(sapply(splited, colMeans))
  #class(mu_hat)
  #mu_hat

  sigma_hat <- array(0, dim = c(p, p, K))

  for(i in 1:K){
    sigma_hat[,,i] <- cov(splited[[i]])
  }

  sigma_hat

  return(list(pi_hat = pi_hat, mu_hat = mu_hat, sigma_hat = sigma_hat))


}

myqda <- my_qda(iris[1:140,1:4], iris[1:140,5])
qda_default <- qda(Species ~ ., data = iris[1:140,])

myqda$pi_hat
```

```
##     setosa versicolor  virginica
##  0.3571429  0.3571429  0.2857143
```

```r
qda_default$prior
```

```
##     setosa versicolor  virginica
##  0.3571429  0.3571429  0.2857143
```

```r
myqda$mu_hat
```

```
##            Sepal.Length Sepal.Width Petal.Length Petal.Width
## setosa           5.0060       3.428       1.4620       0.246
## versicolor       5.9360       2.770       4.2600       1.326
## virginica        6.6225       2.960       5.6075       1.990
```

```r
qda_default$means
```

```
##            Sepal.Length Sepal.Width Petal.Length Petal.Width
## setosa           5.0060       3.428       1.4620       0.246
## versicolor       5.9360       2.770       4.2600       1.326
## virginica        6.6225       2.960       5.6075       1.990
```

```r
myqda$sigma_hat
```

```
## , , 1
##
##            [,1]        [,2]        [,3]        [,4]
## [1,] 0.12424898 0.099216327 0.016355102 0.010330612
## [2,] 0.09921633 0.143689796 0.011697959 0.009297959
## [3,] 0.01635510 0.011697959 0.030159184 0.006069388
## [4,] 0.01033061 0.009297959 0.006069388 0.011106122
##
## , , 2
##
##            [,1]       [,2]       [,3]       [,4]
## [1,] 0.26643265 0.08518367 0.18289796 0.05577959
## [2,] 0.08518367 0.09846939 0.08265306 0.04120408
## [3,] 0.18289796 0.08265306 0.22081633 0.07310204
## [4,] 0.05577959 0.04120408 0.07310204 0.03910612
##
## , , 3
##
##            [,1]       [,2]       [,3]       [,4]
## [1,] 0.46794231 0.11041026 0.35777564 0.05125641
## [2,] 0.11041026 0.11323077 0.08107692 0.04625641
## [3,] 0.35777564 0.08107692 0.34532692 0.05930769
## [4,] 0.05125641 0.04625641 0.05930769 0.07425641
```

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

  dmvnormm <- data.frame()
  m <- dim(newdata)[1]
  K <- dim(fit$mu_hat)[1]

  posterior <- matrix(0, m, K)



  for(i in 1:K){
    dmvnormm <- rbind(dmvnormm,dmvnorm(newdata, fit$mu_hat[i,], fit$sigma_hat[,,i]))
  }

  dmvnormm


  for(i in 1:m){
    numerator <- sum(dmvnormm[,i] * fit$pi_hat)
    for(j in 1:K){
      posterior[i, j] <- dmvnormm[j, i] * fit$pi_hat[j] / numerator

    }
  }

  colnames(posterior) <- names(fit$pi_hat)
  posterior
```

```
  class <- apply(posterior, 1, function(x)  names(which.max(x)))

  return(list(class = class, posterior = posterior))
}


predictqda <- predict_my_qda(myqda, iris[141:150, -5])
predictqda_default <- predict(qda_default, iris[141:150,])




predictqda_default$class
```

```
##  [1] virginica virginica virginica virginica virginica virginica virginica
##  [8] virginica virginica virginica
## Levels: setosa versicolor virginica
```

```
predictqda$class
```

```
##  [1] "virginica" "virginica" "virginica" "virginica" "virginica"
##  [6] "virginica" "virginica" "virginica" "virginica" "virginica"
```

```
# Q : ???
predictqda_default$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
```

```
predictqda$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
```

Confusion matrix (K * K)

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

lda <- lda(Species ~., data = train_set)
qda <- qda(Species ~., data = train_set)

predlda <- predict(lda, test_set)
predqda <- predict(qda, test_set)


table(predlda$class, iris[-train_idx, 5])
```

```
##
##                setosa versicolor virginica
##   setosa           24          0         0
##   versicolor        0         17         1
##   virginica         0          0        18
```

```r
table(predqda$class, iris[-train_idx, 5])
```

```
##
##                setosa versicolor virginica
##   setosa           24          0         0
##   versicolor        0         17         1
##   virginica         0          0        18
```

```r
confusionMatrix(predlda$class, iris[-train_idx, 5])
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         24          0         0
##   versicolor      0         17         1
##   virginica       0          0        18
##
## Overall Statistics
##
##                Accuracy : 0.9833
##                  95% CI : (0.9106, 0.9996)
##     No Information Rate : 0.4
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9747
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                    1.0            1.0000           0.9474
## Specificity                    1.0            0.9767           1.0000
```

```
## Pos Pred Value                    1.0            0.9444           1.0000
## Neg Pred Value                    1.0            1.0000           0.9762
## Prevalence                        0.4            0.2833           0.3167
## Detection Rate                    0.4            0.2833           0.3000
## Detection Prevalence              0.4            0.3000           0.3000
## Balanced Accuracy                 1.0            0.9884           0.9737
```

```
confusionMatrix(predqda$class, iris[-train_idx, 5])
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         24          0         0
##   versicolor      0         17         1
##   virginica       0          0        18
##
## Overall Statistics
##
##                Accuracy : 0.9833
##                  95% CI : (0.9106, 0.9996)
##     No Information Rate : 0.4
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9747
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                    1.0            1.0000           0.9474
## Specificity                    1.0            0.9767           1.0000
## Pos Pred Value                 1.0            0.9444           1.0000
## Neg Pred Value                 1.0            1.0000           0.9762
## Prevalence                     0.4            0.2833           0.3167
## Detection Rate                 0.4            0.2833           0.3000
## Detection Prevalence           0.4            0.3000           0.3000
## Balanced Accuracy              1.0            0.9884           0.9737
```

Multinomial Logistic Regression

```
find_multinom_coef <- function(X , y){
  Y <- dummy(y)
  Y <- Y[,-1]
  n <- length(y)
  p <- dim(X)[2]
  K <- nlevels(y)
  X <- as.matrix(cbind(1,X))

  #B <- matrix(0, p+1, K-1)

  loglike <- function(B){
    c <- 0
```

```r
    B <- matrix(B, ncol = K-1)

    for(i in 1:n){
      a <- 0
      b <- 0

      for(k in 1: (K-1)){
        a <-  a + Y[i,k] *  as.numeric(X[i, ]  %*%  B[, k])

        b <- b + exp(as.numeric(X[i, ]  %*%  B[ ,k]))

      }

      c <- c + a - log(1 + b)

    }

    return(-c)
  }

  optimed <- optim(matrix(0, p+1, K-1), fn = loglike, method="BFGS")
  # optim function flattens the matrix arguments into vectors (columnwise)

  param = optimed$par
  colnames(param) <-levels(y)[-1]

  return(param) #   (p+1) * (K-1)

}




# Check
# loglike(matrix(0, p+1, K-1))
# n * log(K)



find_multinom_coef(X=iris[1:140, 1:4], y=iris$Species[1:140])
```

```
##       versicolor  virginica
## [1,] 17.7254637 -24.631223
## [2,] -6.7005422  -9.107771
## [3,] -6.2433338 -12.869906
## [4,] 13.7900526  23.118285
## [5,] -0.5066336  17.596108
```

```r
iris_multi <- multinom(Species ~ ., data=iris[1:140, ])
```

```
## # weights:  18 (10 variable)
## initial  value 153.805720
## iter  10 value 24.082349
## iter  20 value 6.036653
```

```
## iter  30 value 5.937954
## iter  40 value 5.930515
## iter  50 value 5.926939
## iter  60 value 5.925467
## final  value 5.923988
## converged
```

```r
# ignore the output here.

t(coef(iris_multi))
```

```
##              versicolor  virginica
## (Intercept)  17.7252583 -24.630925
## Sepal.Length -6.7006986  -9.107935
## Sepal.Width  -6.2434619 -12.870044
## Petal.Length 13.7902839  23.118434
## Petal.Width  -0.5060067  17.596721
```

```r
# betafun <- function(beta){     # input beta is vector
#
#     c <- 0
#
#     for(i in 1:n){
#       jj <- 0
#       a <- 0
#       b <- c()
#
#       for(k in 1:(K-1)){
#
#         for(j in 1:(p+1)){
#           jj <- jj+1
#           a <- a + X[i,j] * beta[jj]
#           a
#
#         }
#         a <- a + a * Y[i,k]
#
#       }
#       a
#
#       jj <- 0
#       b <- rep(0, (K-1))
#
#       for(k in 1: (K-1)){
#         for(j in 1:(p+1)){
#           jj <- jj + 1
#
#           b[k] <- b[k] + X[i,j] * beta[jj]
#         }
#         b[2]
#
#       }
#
#       b <- sum(exp(b))
#       b <- b+1
```

```
#
#       b <- log(b)
#
#
#       c <- c + a - b
#
#     }
#
#     return(c)
#
#   }
#
#
# # Check
# betafun(rep(0, (p+1) * (K-1)))
# n * log(K)
#
# optimed <- optim(rep(0, (p+1) * (K-1)), fn = betafun, method="BFGS",  control = list(fnscale = -1))
#
# optimed$par
```