Problem Set 5: Discriminant Analysis

Donggyun Kim
27008257

4/12/2018

1) LDA

1.1) Function lda_fit()

```r
lda_fit <- function(X, y) {
  X <- as.matrix(X)
  y <- factor(y)
  clss <- levels(y)
  K <- length(clss)
  N <- dim(X)[1]
  P <- dim(X)[2]

  n <- numeric(K)
  for (k in 1:K) {
    n[k] <- sum(y == clss[k])
  }

  pi_hat <- numeric(K)
  for (k in 1:K) {
    pi_hat[k] <- n[k] / N
  }

  mu_hat <- matrix(0, nrow = K, ncol = P)
  for (k in 1:K) {
    index <- y == clss[k]
    for (p in 1:P) {
      mu_hat[k, p] <- mean(X[index, p])
    }
  }

  arr <- array(0, c(P, P, K))
  for (k in 1:K) {
    index <- y == clss[k]
    for (i in 1:P) {
      for (j in 1:P) {
        arr[i, j, k] <- t(X[index, i] - mu_hat[k, i]) %*% (X[index, j] - mu_hat[k, j])
      }
    }
  }
  sigma_hat <- matrix(0, nrow = P, ncol = P)
  for (k in 1:K) {
    sigma_hat <- sigma_hat + arr[, , k]
  }
  sigma_hat <- sigma_hat / (N - K)
  list(pi_hat = pi_hat,
       mu_hat = mu_hat,
       sigma_hat = sigma_hat)
}
```

1.2) Function lda_predict()

```
lda_predict <- function(fit, newdata) {
  pi_hat <- fit$pi_hat
  mu_hat <- fit$mu_hat
  sigma_hat <- fit$sigma_hat
  K <- length(pi_hat)

  X <- as.matrix(newdata)
  N <- dim(X)[1]
  P <- dim(X)[2]
  class <- numeric(N)

  delta_hat <- numeric(K)
  for (n in 1:N) {
    for (k in 1:K) {
      delta_hat[k] <- log(pi_hat[k]) - t(mu_hat[k, ]) %*% solve(sigma_hat) %*%
        mu_hat[k, ] / 2 + t(mu_hat[k, ]) %*% solve(sigma_hat) %*% X[n, ]
    }
    class[n] <- which.max(delta_hat)
  }

  fx <- matrix(0, nrow = N, ncol = K)
  for (n in 1:N) {
    for (k in 1:K) {
      fx[n, k] <- 1 / ((2 * pi)^(P/2) * det(sigma_hat)^(1/2)) *
        exp(-t(X[n, ] - mu_hat[k, ]) %*% solve(sigma_hat) %*% (X[n, ] - mu_hat[k, ]))
    }
  }

  denominator <- numeric(N)
  for (n in 1:N) {
    for (k in 1:K) {
      denominator[n] <- denominator[n] + pi_hat[k] * fx[n, k]
    }
  }

  posterior <- matrix(0, nrow = N, ncol = K)
  for (n in 1:N) {
    for (k in 1:K) {
      posterior[n, k] <- pi_hat[k] * fx[n, k] / denominator[n]
    }
  }

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

1.3) Classification with LDA

```
training <- c(1:47, 51:97, 101:146)
testing <- c(48:50, 98:100, 147:150)
fit <- lda_fit(iris[training, 1:4], iris$Species[training])
fit
```

```
## $pi_hat
## [1] 0.3357143 0.3357143 0.3285714
##
## $mu_hat
##          [,1]     [,2]     [,3]      [,4]
## [1,] 5.008511 3.429787 1.463830 0.2489362
## [2,] 5.953191 2.772340 4.289362 1.3319149
## [3,] 6.619565 2.973913 5.584783 2.0282609
##
## $sigma_hat
##            [,1]       [,2]       [,3]       [,4]
## [1,] 0.27084678 0.09672053 0.16682306 0.03908908
## [2,] 0.09672053 0.11913165 0.05524487 0.03340797
## [3,] 0.16682306 0.05524487 0.18140540 0.04254695
## [4,] 0.03908908 0.03340797 0.04254695 0.04345135
```

```
lda_predict(fit, iris[testing, 1:4])
```

```
## $class
##  [1] 1 1 1 2 2 2 3 3 3 3
##
## $posterior
##               [,1]         [,2]         [,3]
##  [1,] 1.000000e+00 3.621157e-36 4.361375e-74
##  [2,] 1.000000e+00 7.344352e-46 1.961308e-86
##  [3,] 1.000000e+00 2.740535e-40 1.325343e-79
##  [4,] 4.406034e-36 1.000000e+00 1.971779e-09
##  [5,] 6.557505e-20 1.000000e+00 1.658028e-16
##  [6,] 5.443644e-37 1.000000e+00 4.020728e-09
##  [7,] 4.615677e-70 9.587777e-05 9.999041e-01
##  [8,] 2.626177e-68 2.315291e-05 9.999768e-01
##  [9,] 2.942458e-79 4.991078e-10 1.000000e+00
## [10,] 2.850688e-65 5.482183e-04 9.994518e-01
```

2) QDA

2.1) Function qda_fit()

```
qda_fit <- function(X, y) {
  X <- as.matrix(X)
  N <- dim(X)[1]
  P <- dim(X)[2]
```

```r
  y <- factor(y)
  clss <- levels(y)
  K <- length(clss)

  n <- numeric(K)
  for (k in 1:K) {
    n[k] <- sum(y == clss[k])
  }

  pi_hat <- numeric(K)
  for (k in 1:K) {
    pi_hat[k] <- n[k] / N
  }

  mu_hat <- matrix(0, nrow = K, ncol = P)
  for (k in 1:K) {
    for (p in 1:P) {
      mu_hat[k, p] <- mean(X[y == clss[k], p])
    }
  }

  sigma_hat <- array(0, c(P, P, K))
  for (k in 1:K) {
    for (i in 1:P) {
      for (j in 1:P) {
        sigma_hat[i, j, k] <- t(X[y == clss[k], i] - mu_hat[k, i]) %*%
          (X[y == clss[k], j] - mu_hat[k, j]) / (n[k] - 1)
      }
    }
  }

  list(pi_hat = pi_hat,
       mu_hat = mu_hat,
       sigma_hat = sigma_hat)
}
```

2.2) Function qda_predict()

```r
qda_predict <- function(fit, newdata) {
  pi_hat <- fit$pi_hat
  mu_hat <- fit$mu_hat
  sigma_hat <- fit$sigma_hat
  X <- as.matrix(newdata)
  N <- dim(X)[1]
  P <- dim(X)[2]
  K <- length(pi_hat)

  class <- numeric(N)
  delta_hat <- numeric(K)
  for (n in 1:N) {
    for (k in 1:K) {
      delta_hat[k] <- -t(X[n, ]) %*% solve(sigma_hat[, , k]) %*% X[n, ] / 2 +
        t(X[n, ]) %*% solve(sigma_hat[, , k]) %*% mu_hat[k, ] -
        t(mu_hat[k, ]) %*% solve(sigma_hat[, , k]) %*% mu_hat[k, ] / 2 -
        log(det(sigma_hat[, , k])) / 2 + log(pi_hat[k])
    }
    class[n] <- which.max(delta_hat)
  }

  fx <- matrix(0, nrow = N, ncol = K)
  for (n in 1:N) {
    for (k in 1:K) {
      fx[n, k] <- 1 / ((2 * pi)^(P / 2) * det(sigma_hat[, , k])^(1/2)) *
        exp(-t(X[n, ] - mu_hat[k, ]) %*% solve(sigma_hat[, , k]) %*%
              (X[n, ] - mu_hat[k, ]))
    }
  }

  denominator <- numeric(N)
  for (n in 1:N) {
    for (k in 1:K) {
      denominator[n] <- denominator[n] + pi_hat[k] * fx[n, k]
    }
  }

  posterior <- matrix(0, nrow = N, ncol = K)
  for (n in 1:N) {
    for (k in 1:K) {
      posterior[n, k] <- pi_hat[k] * fx[n, k] / denominator[n]
    }
  }

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

2.3) Classification with QDA

```
training <- c(1:47, 51:97, 101:146)
testing <- c(48:50, 98:100, 147:150)
fit <- qda_fit(iris[training, 1:4], iris$Species[training])
fit
```

```
## $pi_hat
## [1] 0.3357143 0.3357143 0.3285714
##
## $mu_hat
##          [,1]     [,2]     [,3]      [,4]
## [1,] 5.008511 3.429787 1.463830 0.2489362
## [2,] 5.953191 2.772340 4.289362 1.3319149
## [3,] 6.619565 2.973913 5.584783 2.0282609
##
## $sigma_hat
## , , 1
```

```
##
##             [,1]        [,2]        [,3]        [,4]
## [1,] 0.12688252 0.101914894 0.016618871 0.010878816
## [2,] 0.10191489 0.149962997 0.011753006 0.009814986
## [3,] 0.01661887 0.011753006 0.031924144 0.006373728
## [4,] 0.01087882 0.009814986 0.006373728 0.011683626
##
## , , 2
##
##             [,1]       [,2]       [,3]       [,4]
## [1,] 0.26558742 0.08519889 0.17036078 0.05522202
## [2,] 0.08519889 0.10291397 0.08056892 0.04264107
## [3,] 0.17036078 0.08056892 0.19923219 0.07143386
## [4,] 0.05522202 0.04264107 0.07143386 0.04048104
##
## , , 3
##
##             [,1]       [,2]       [,3]       [,4]
## [1,] 0.42338647 0.10318841 0.31674879 0.05143478
## [2,] 0.10318841 0.10419324 0.07381643 0.04808696
## [3,] 0.31674879 0.07381643 0.31598551 0.04999517
## [4,] 0.05143478 0.04808696 0.04999517 0.07896135
```

```
qda_predict(fit, iris[testing, 1:4])
```

```
## $class
##  [1] 1 1 1 2 2 2 3 3 3 3
##
## $posterior
##                [,1]         [,2]         [,3]
##  [1,]  1.000000e+00 3.624020e-41 1.236227e-63
##  [2,]  1.000000e+00 9.257777e-57 1.811408e-81
##  [3,]  1.000000e+00 5.205005e-46 8.512853e-72
##  [4,] 1.654272e-133 1.000000e+00 1.907411e-08
##  [5,]  2.929219e-50 1.000000e+00 2.813120e-11
##  [6,] 1.413067e-118 9.999998e-01 2.153270e-07
##  [7,] 8.416072e-235 6.005936e-08 9.999999e-01
##  [8,] 1.046532e-252 1.407734e-06 9.999986e-01
##  [9,] 1.326582e-293 5.684392e-12 1.000000e+00
## [10,] 3.627650e-224 3.112374e-03 9.968876e-01
```

3) k-Nearest Neighbors

3.1) Function knn_predict()

```
knn_predict <- function(X_train, X_test, y_train, k) {
  train <- as.matrix(X_train)
  rownames(train) <- 1:nrow(train)
  test <- as.matrix(X_test)
  y_train <- factor(y_train)
  I <- nrow(test)
  y_test <- numeric(I)
  for (i in 1:I) {
    X <- sweep(train, 2, test[i, ], "-")
    dst <- apply(X, 1, function(x) sum(x^2))
    index <- as.numeric(names(sort(dst)[1:k]))
    factors <- y_train[index]
    y_test[i] <- names(which.max(table(factors)))
  }
  factor(y_test)
}
```

3.2) Classification with k-NN

```
training <- c(1:47, 51:97, 101:146)
testing <- c(48:50, 98:100, 147:150)
train_set <- iris[training, ]
test_set <- iris[testing, ]
pred_knn <- knn_predict(train_set[, -5], test_set[, -5], train_set$Species, k=1)
pred_knn
```

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

3.3) k-NN CV

```
find_kcv <- function(X_train, Y_train, k = 1:10, nfold = 4) {
  N <- nrow(X_train)
  index <- N / nfold
  ter <- matrix(0, nrow = length(k), ncol = nfold)
  for (i in 1:nfold) {
    index_i <- index * (i - 1) + 1
    index_j <- index * i
    index_itoj <- index_i:index_j
    train_x <- X_train[-index_itoj, ]
    test_x <- X_train[index_itoj, ]
    train_y <- Y_train[-index_itoj]
    test_y <- Y_train[index_itoj]
    for (j in seq_along(k)) {
      pred_knn <- knn_predict(train_x, test_x, train_y, k[j])
      tbl <- table(test_y, pred_knn)
      ter[j, i] <- 1 - sum(diag(tbl)) / sum(tbl)
    }
  }
  k[which.min(apply(ter, 1, mean))]
}

find_kcv(train_set[ , -5], train_set[ , 5])
```

```
## [1] 6
```

4) Confusion matrix

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

fit <- lda_fit(train_set[, 1:4], train_set[, 5])
class_lda <- lda_predict(fit, test_set[, 1:4])$class
prd_lda <- class_lda
prd_lda[class_lda == 1] <- "setosa"
prd_lda[class_lda == 2] <- "versicolor"
prd_lda[class_lda == 3] <- "virginica"
prd_lda <- factor(prd_lda)

fit <- qda_fit(train_set[, 1:4], train_set[, 5])
class_qda <- qda_predict(fit, test_set[, 1:4])$class
prd_qda <- class_qda
prd_qda[class_qda == 1] <- "setosa"
prd_qda[class_qda == 2] <- "versicolor"
prd_qda[class_qda == 3] <- "virginica"
prd_qda <- factor(prd_qda)

find_kcv(train_set[, 1:4], train_set[, 5])
```

```
## [1] 9
```

```
prd_knn <- knn_predict(train_set[, 1:4], test_set[, 1:4], train_set[, 5], k = 9)

# confusion matrix
table(test_set[, 5], prd_lda)
```

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

```
table(test_set[, 5], prd_qda)
```

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

```
table(test_set[, 5], prd_knn)
```

```
##             prd_knn
##              setosa versicolor virginica
##   setosa         24          0         0
##   versicolor      0         17         0
##   virginica       0          2        17
```

```
# test error rate of lda
1 - sum(diag(table(test_set[, 5], prd_lda))) / sum(table(test_set[, 5], prd_lda))
```

```
## [1] 0.01666667
```

```
# test error rate of qda
1 - sum(diag(table(test_set[, 5], prd_qda))) / sum(table(test_set[, 5], prd_qda))
```

```
## [1] 0.01666667
```

```
# test error rate of knn
1 - sum(diag(table(test_set[, 5], prd_knn))) / sum(table(test_set[, 5], prd_knn))
```

```
## [1] 0.03333333
```

Test error rates of lda and qda are same. Test error rate of knn is greater than lda and qda.