discriminant_analysis.R

jovyan

2022-11-30

```r
Comp_priors <- function(train_labels) {
  #' Compute the priors of each class label
  #'
  #' @param train_labels a vector of labels with length equal to n
  #' @return a probability vector of length K = 10


  K <- 10
  pi_vec <- rep(0, K)

  ###########################################################################
  #  TODO                                                                   #
  ###########################################################################

  # for both LDA and QDA, the prior is estimated by the mean
  for (k in 0:K-1){
    pi_vec[k+1] <- sum(train_labels == k)/length(train_labels)
  }

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

  return(pi_vec)
}



Comp_cond_means <- function(train_data, train_labels) {
  #' Compute the conditional means of each class
  #'
  #' @param train_data a n by p matrix containing p features of n training points
  #' @param train_labels a vector of labels with length equal to n
  #'
  #' @return a p by 10 matrix, each column represents the conditional mean given
  #'   each class.

  K <- 10
  p <- ncol(train_data)
  mean_mat <- matrix(0, p, K)
  ###########################################################################
  #  TODO                                                                   #
```

```r
  ##########################################################################

  train_data_df <- as.data.frame(train_data)
  train_labels_df <- as.data.frame(train_labels)

  train_data_df$label <- train_labels_df[,1]
  train_data_df <- train_data_df[order(train_data_df$label),]

  for (k in 0:K-1) {
  mean_mat[,k+1] <- as.matrix(colMeans(train_data_df[train_data_df$label == k,]))[1:p,]
  }

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

  return(mean_mat)
}



Comp_cond_covs <- function(train_data, train_labels, cov_equal = FALSE) {
  #' Compute the conditional covariance matrix of each class
  #'
  #' @param train_data a n by p matrix containing p features of n training points
  #' @param train_labels a vector of labels with length equal to n
  #' @param cov_equal TRUE if all conditional covariance matrices are equal,
  #'   otherwise, FALSE
  #'
  #' @return
  #'  if \code{cov_equal} is FALSE, return an array with dimension (p, p, K),
  #'    containing p by p covariance matrices of each class;
  #'  else, return a p by p covariance matrix.

  K <- 10
  p <- ncol(train_data)


  ##########################################################################
  #  TODO                                                                  #
  ##########################################################################

  cov_arr <- NA
  n <- length(y_train)
  train_data_df <- as.data.frame(train_data)
  train_labels_df <- as.data.frame(train_labels)

  train_data_df$label <- train_labels_df[,1]
  train_data_df <- train_data_df[order(train_data_df$label),]

  mu <- Comp_cond_means(train_data, train_labels)

  if (cov_equal == FALSE) {
```

```
    cov_arr <- array(0,c(p,p,K))
  # QDA
    for (k in 0:K-1) {

    cov_arr[,,k+1] <- 1/(sum(train_labels == k)-1)*((t(train_data_df[train_data_df$label == k,1:p])
                - mu[,k+1])%*%t(t(train_data_df[train_data_df$label == k,1:p])
                               - mu[,k+1]))
    }
  }

  if (cov_equal == TRUE){
    cov_arr_1 <- array(0,c(p,p,K))
    cov_arr <- matrix(0,p,p)
  # LDA
    for (k in 0:K-1) {

      cov_arr_1[,,k+1] <- 1/(n-K)*
        ((t(train_data_df[train_data_df$label == k,1:p])
          - mu[,k+1])%*%t(t(train_data_df[train_data_df$label == k,1:p])
                          - mu[,k+1]))
    }

      cov_arr <- cov_arr_1[,,1]+cov_arr_1[,,2]+cov_arr_1[,,3]+cov_arr_1[,,4]
      +cov_arr_1[,,5]+cov_arr_1[,,6]+cov_arr_1[,,7]
      +cov_arr_1[,,8]+cov_arr_1[,,9]+cov_arr_1[,,10]

  }
  return(cov_arr)
  ##########################################################################
  #                         END OF YOUR CODE                               #
  ##########################################################################
}




Predict_posterior <- function(test_data, priors, means, covs, cov_equal) {

  #' Predict the posterior probabilities of each class
  #'
  #' @param test_data a n_test by p feature matrix
  #' @param priors a vector of prior probabilities with length equal to K
  #' @param means a p by K matrix containing conditional means given each class
  #' @param covs covariance matrices of each class, depending on \code{cov_equal}
  #' @param cov_equal TRUE if all conditional covariance matrices are equal;
  #'   otherwise FALSE.
  #'
  #' @return a n_test by K matrix: each row contains the posterior probabilities
  #'   of each class.

  n_test <- nrow(test_data)
  K <- length(priors)
  posteriors <- matrix(0, n_test, K)
```

```
  ###########################################################################
  #  TODO                                                                   #
  ###########################################################################

  #LDA
  if (cov_equal == TRUE) {
  # test data is 4000*64
  sigmas <- as.matrix(diag(covs)) #64*1
  denominator <- rep(0,n_test,K)
  # parameters wiz subscripts
  for (l in 1:K) {
    diff <- test_data - means[,l] #4000*64
    prod <- (diff^2) %*% ((-1/2)/sigmas) #4000*1
    denominator <- denominator + priors[l]*exp(prod)
  }

  for (k in 1:K) {
      diff <- test_data - means[,k] #4000*64
      prod <- (diff^2) %*% ((-1/2)/sigmas) #4000*1
      numerator <- priors[k]*exp(prod)
      posteriors[,k] <- numerator/denominator
      }
  }

  #QDA
  if (cov_equal == FALSE) {
    # test data is 4000*64
    denominator <- rep(0,n_test,K)
    for (l in 1:K){
      sigmas <- as.matrix(diag(covs[,,l])) #64*1
      diff <- test_data - means[,l] #4000*64
      prod <- (diff^2) %*% ((-1/2)/sigmas)
      denominator <- denominator + priors[l]*exp(prod)
    }

    for (k in 1:K) {
      sigmas <- as.matrix(diag(covs[,,k])) #64*1
      diff <- test_data - means[,k] #4000*64
      prod <- (diff^2) %*% ((-1/2)/sigmas) #4000*1
      numerator <- priors[k]*exp(prod)
      posteriors[,k] <- numerator/denominator
    }

  }


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

  return(posteriors)
}
```

```r
Predict_labels <- function(posteriors) {

  #' Predict labels based on the posterior probabilities over K classes
  #'
  #' @param posteriors A n by K posterior probabilities
  #'
  #' @return A vector of predicted labels with length equal to n

  n_test <- nrow(posteriors)
  pred_labels <- rep(NA, n_test)

  ###########################################################################
  #  TODO                                                                   #
  ###########################################################################

  for (i in 1:n_test) {
  pred_labels[i] <- which.max(posteriors[i,])
  }

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

  return(pred_labels)
}
```