Vowel Recognition via Regression, LDA and QDA

Lucy L.

2024-01-03

This write-up is derived from the textbook "The Elements of Statistical Learning" from Chapter 4, exercise 4.9: "Write a computer program to perform a quadratic discriminant analysis by fitting a separate Gaussian model per class. Try it out on the vowel data, and compute the misclassification error for the test data."

In this analysis of vowel recognition data, the focus is on independently recognizing eleven steady-state vowels of British English. This recognition is achieved irrespective of the speaker, utilizing a dedicated training set derived from LPC with log area ratios.

I will proceed by performing Linear Regression, Linear Discriminant Analysis (LDA), and Logistic Regression in addition to Quadratic Discriminant Analysis (QDA).

```
load_vowel_data <- function(doScaling=FALSE, doRandomization=FALSE){
  # R code to load in the vowel data set from the book ESLII
  # Output:
  # res: list of data frames XT

  # Get the training data:
  XTrain = read.csv("train_xl.csv")

  # Extract the true classification for each datum
  labelsTrain = XTrain[,1]

  # Delete the column of classification labels:
  XTrain$y = NULL

  # We try to scale ALL features so that they have mean zero and a
  # standard deviation of one.
  if(doScaling){
    XTrain = scale(XTrain, TRUE, TRUE)
    means  = attr(XTrain,"scaled:center")
    stds   = attr(XTrain,"scaled:scale")
    XTrain = data.frame(XTrain)
  }

  # Sometime data is processed and stored on disk in a certain order.
  # When doing cross validation on such data sets we don't want to bias our
  # results if we grab the first or the last samples. Thus we randomize the order
  # of the rows in the Training data frame to make sure that each
  # cross validation training/testing set is as random as possible.

  if(doRandomization){
    nSamples    = dim(XTrain)[1]
    inds        = sample( 1:nSamples, nSamples)
```

```
    XTrain      = XTrain[inds,]
    labelsTrain = labelsTrain[inds]
  }

  # Get the testing data:
  XTest  = read.csv("test_xl.csv")

  # Extract the true classification for each datum
  labelsTest = XTest[,1]

  # Delete the column of classification labels:
  XTest$y = NULL

  # Scale the testing data using the same transformation as was applied to
  # the training data:
  if( doScaling ){
    XTest = t(apply(XTest, 1, '-', means))
    XTest = t(apply(XTest, 1, '/', stds))
  }

  return(list(XTrain, labelsTrain, XTest, labelsTest))
}

linear_regression_indicator_matrix = function(XTrain, yTrain){
  # R code to compute class indicator matrix for linear regression
  #
  # Inputs:
  #   XTrain = training matrix (without the common column of ones needed to
  #     represent the constant offset)
  #   yTrain = training labels matrix of true classifications with indices 1 - K
  #     (where K is the number of classes)

  K = max(yTrain) # Number of classes
  N = dim( XTrain )[1] # Number of samples

  # Form the indicator response matrix Y
  Y = mat.or.vec(N, K)
  for(ii in 1:N){
    jj = yTrain[ii]
    Y[ii,jj] = 1.
  }

  Y = as.matrix(Y)

  # For the feature vector matrix XTrain (append a leading column of ones) and
  # compute Yhat:
  ones = as.matrix( mat.or.vec( N, 1 ) + 1.0 )
  Xm   = as.matrix( cbind( ones, XTrain ) )
  # Used for predictions on out of sample data
  Bhat = solve( t(Xm) %*% Xm, t(Xm) %*% Y )
  Yhat = Xm %*% Bhat # Discriminant predictions on the training data
  gHat = apply( Yhat, 1, 'which.max' ) # classify this data
```

```
  return( list(Bhat,Yhat,gHat) )
}
```

```
out = load_vowel_data( FALSE )
XTrain = out[[1]]
labelsTrain = out[[2]]
XTest = out[[3]]
labelsTest = out[[4]]
```

```
errors = data.frame(Method = c("Linear Regression",
                               "Logistic Regression",
                               "Linear Discriminant Analysis",
                               "Quadratic Discriminant Analysis"))
errors <- cbind(errors, `Training Error Rate (%)` = c(lr_eRateTrain*100,
                                          log_eRateTrain*100,
                                          lda_eRateTrain*100,
                                          qda_eRateTrain*100))
errors <- cbind(errors, `Test Error Rate (%)` = c(lr_eRateTest*100,
                                          log_eRateTest*100,
                                          lda_eRateTest*100,
                                          qda_eRateTest*100))
print(errors)
```

```
##                            Method Training Error Rate (%) Test Error Rate (%)
## 1               Linear Regression               47.727273            66.66667
## 2             Logistic Regression               22.159091            51.29870
## 3    Linear Discriminant Analysis               31.628788            55.62771
## 4 Quadratic Discriminant Analysis                1.136364            52.81385
```

The misclassification error for both the training and test sets by method are shown in the above table. The error rates for Linear Regression suffer significantly from the effects of masking (K is greater than 3) within the data, which LDA is able to avoid.