LDA classification Vignette

Oscar Trevizo

January 2023

Contents

Purpose

To demonstrate the LDA approach to solving classification problems. There are Various classification techniques.

- Linear Discriminant Analysis (LDA)
- Quadratic Discriminant Analysis (QDA)
- Logistic Regression / Generalized Linear Models (GLM)

This vignette focuses on LDA.

Functions

Adapted from R functions shared by faculty in Harvard data science class (2021). See references at the bottom of this notebook.

```
###
#
# prediction.metrics function -- to return a list with all the metrics values
#
# Based on R functions shared by faculty in Harvard data science class (2021). See references.
#
# Input: truth and predicted lists.
#
# Returns a list with:
# [1] OBS = Observations or truth cases
# [2] Accuracy. ACC = sum(truth == predicted) * 100/length(truth)
# [3] Sensitivity. TPR True Positive Rate = TP/(TP + FN) = TP/P
# [4] Specificity. TNR True Negative Rate = TN/(FP + TN) = TN/N
# [5] Precision. Positive Predictive Value. PPV = TP/(TP + FP)
# [6] Negative Predictive Value. NPV = TN/(TN + FN)
# [7] False Discovery Rate. FDR = FP/(TP + FP)
# [8] False Positive Rate. FPR = FP/(FP + TN) = FP/N
# [9] True Positives. TP = sum(truth == 1 & predicted == 1)
# [10] True Negatives. TN = sum(truth == 0 & predicted == 0)
# [11] False Positives. FP = sum(truth == 0 & predicted == 1)
# [12] False Negatives. FN = sum(truth == 1 & predicted == 0)
# [13] Positives. P = TP + FN  # total number positives in the truth data
# [14] Negatives. N = FP + TN  # total number of negatives
#
prediction.metrics = function(truth, predicted) {
    # same length:
    if (length(truth) != length(predicted)) {
        stop("truth and predicted must be same length!")
    }
    # check for missing values (we are going to compute metrics on non-missing
    # values only)
    bKeep = !is.na(truth) & !is.na(predicted)
    predicted = predicted[bKeep]
    truth = truth[bKeep]
    # only 0 and 1:
    if (sum(truth %in% c(0, 1)) + sum(predicted %in% c(0, 1)) != 2 * length(truth)) {
        stop("only zeroes and ones are allowed!")
    }
    # how predictions align against known training/testing outcomes: TP/FP=
    # true/false positives, TN/FN=true/false negatives
    TP = sum(truth == 1 & predicted == 1)
    TN = sum(truth == 0 & predicted == 0)
    FP = sum(truth == 0 & predicted == 1)
    FN = sum(truth == 1 & predicted == 0)
    P = TP + FN  # total number of positives in the truth data
    N = FP + TN  # total number of negatives
    # Add the following output to return (OAT 11/9/2021)
    OBS = length(truth)
    ACC = sum(truth == predicted)/length(truth)
```

```
    TPR = TP/P
    TNR = TN/N
    PPV = TP/(TP + FP)
    NPV = TN/(TN + FN)
    FDR = FP/(TP + FP)
    FPR = FP/N

    # Returned a named list
    output <- list(OBS=OBS, ACC=ACC, TPR=TPR, TNR=TNR, PPV=PPV,
                   NPV=NPV, FDR=FDR, FPR=FPR, TP=TP,
                   TN=TN, FP=FP, FN=FN, P=P, N=N)
    return(output)
}

print.the.metrics = function(metrics){
  cat(' OBS = ', metrics$OBS, '..................number of observations')
  cat('\n ACC = ', metrics$ACC, '..................Accuracy')
  cat('\n TPR = ', metrics$TPR, '..................True Positive Rate')
  cat('\n TNR = ', metrics$TNR, '..................True Negative Rate')
  cat('\n PPV = ', metrics$PPV, '..................Positive Predictive Value (Precision)')
  cat('\n NPV = ', metrics$NPV, '..................Negative Predictive Value')
  cat('\n FDR = ', metrics$FDR, '..................False Discover Rate')
  cat('\n FPR = ', metrics$FPR, '..................False Positive Rate')
  cat('\n TP  = ', metrics$FP, '..................True Positives')
  cat('\n TN  = ', metrics$TN, '..................True Negatives')
  cat('\n FP  = ', metrics$TN, '..................False Positives')
  cat('\n FN  = ', metrics$FN, '..................False Negatives')
  cat('\n P   = ', metrics$P, '..................Positives')
  cat('\n N   = ', metrics$N, '..................Negatives')


}
# Logistic regression
lgr.pred.ftn = function(formula, df.train, df.test){
  glm.fit <- glm(formula, data = df.train, family = binomial)
  glm.probs <- predict(glm.fit, newdata = df.test, type = "response")
  glm.pred <- rep(0, dim(df.test)[1])
  glm.pred[glm.probs>0.5]=1
  return(glm.pred)
}

# Linear Discriminant Analysis (LDA)
lda.pred.ftn = function(formula, df.train, df.test){
  lda.fit <- lda(formula, data = df.train)
  lda.pred <- predict(lda.fit, df.test)
  lda.class <- lda.pred$class
  return(lda.class)
}

# Quadratic Discriminant Analysis (QDA)
qda.pred.ftn = function(formula, df.train, df.test){
  qda.fit <- qda(formula, data = df.train)
  qda.pred <- predict(qda.fit, df.test)
```

```
  qda.class <- qda.pred$class
  return(qda.class)
}
```

Simulate the data

Play with two sets of Normally distributed sets of data with different means. We can change the number of samples and we can move the means around.

```
# From Harvard data science class (see references at the end of this notebook)

set.seed(11)

N = 1000
mu = 4

# Our measuring variable is continuous, numeric...
# ...it has two Normal distribution waves
x <- c(rnorm(N), rnorm(N, mean=mu))

# Our outcome is categorical, A and B xxxx times each
# ...the idea is to match A and B to a number x
y <- rep(c("A", "B"), each=N)

# Make a data.frame with 1 and 0 values for Y
df <- data.frame(Y=ifelse(y=="A",0, 1), X=x)
```

Build train and test sets

```
set.seed(12321)

# Get 2:1 random sample ratio for Train:Test sets
sampleTrain <- sample(c(TRUE,FALSE,TRUE), nrow(df), rep=TRUE)
df.train <- df[sampleTrain,]
df.test <- df[!sampleTrain,]
```

Boxplot and histogram

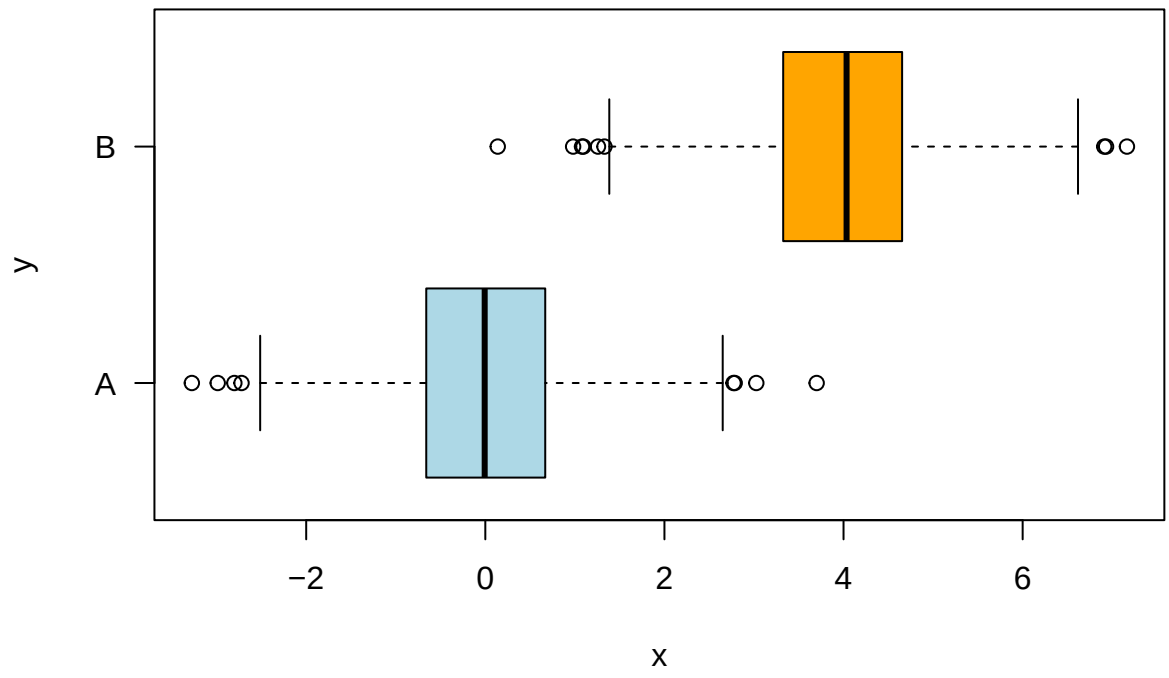
In a boxplot, we want to have the categorical variable in the horizontal axis.

That is why we see a formula x~y below.

The chart gives us more information if we plot it horizontally in this case.

The histogram adds information to visualize the behavior relationship between the outcome, categorical valuable, and predictor, numeric variable.

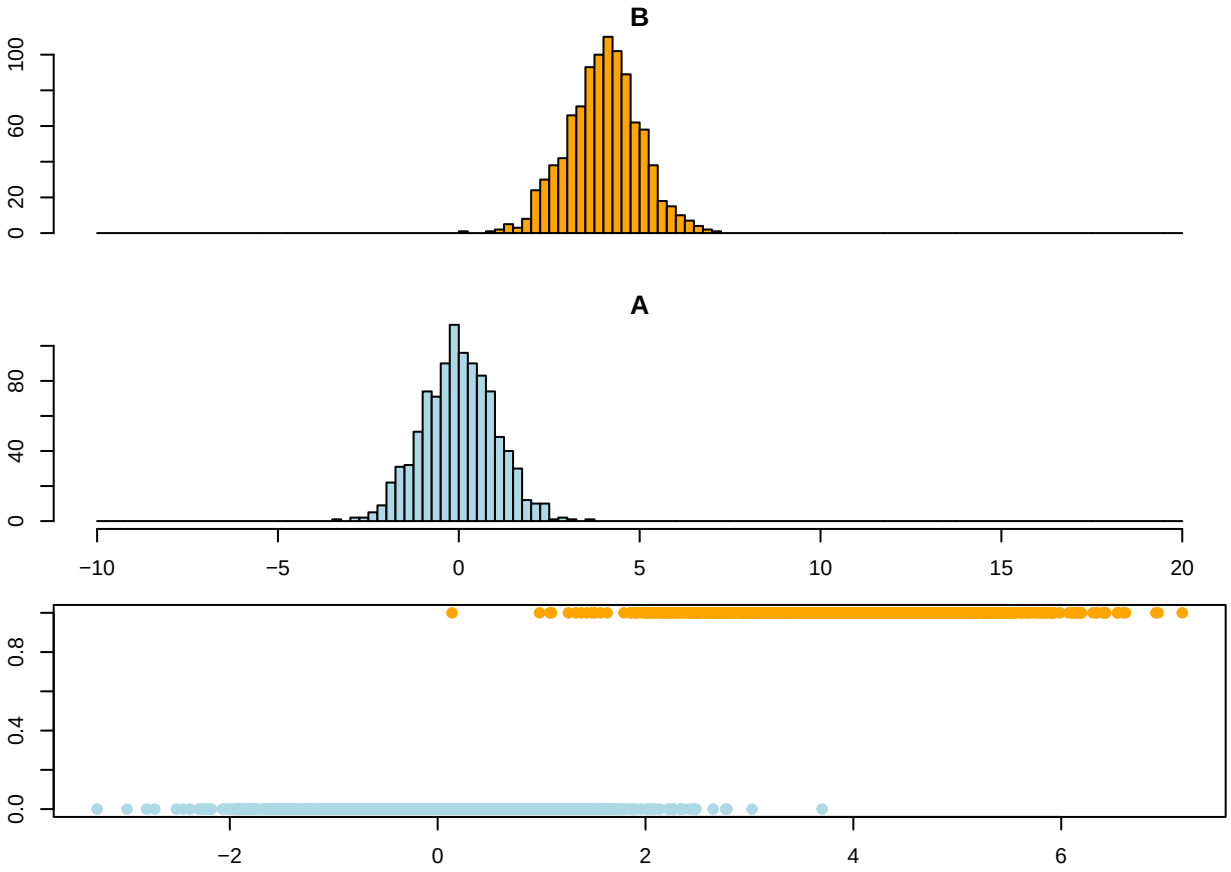
```
# From Harvard data science class (see references)
boxplot(x~y, col=c("lightblue","orange"), horizontal=T, las=1)
```




```
# Now place a histogram on top of another histogram

oldpar <- par(mfrow=c(3, 1), mar=c(2,2,1,1))

breaks <- seq(-10, 20, by=0.25)

# Histogram for 'B'
hist(x[y=="B"], breaks=breaks, col='orange', main="B", xaxt='n')

# Histogram for 'A'
hist(x[y=="A"], breaks=breaks, col='lightblue', main="A")

plot(x, ifelse(y=="A", 0,1), breaks=breaks, col=ifelse(y=="A", "lightblue","orange"), pch=19)
```

```
## Warning in plot.window(...): "breaks" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "breaks" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "breaks" is not a
## graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "breaks" is not a
## graphical parameter

## Warning in box(...): "breaks" is not a graphical parameter
```

```
## Warning in title(...): "breaks" is not a graphical parameter
```




```
par(oldpar)
```

Linear Discriminant Analysis (LDA)

Needs library{MASS}

Fit the model

```
##
#
# LDA from library{MASS}
#
##
lda.fit <- lda(Y~X, data = df.train)
summary(lda.fit)
```

```
##         Length Class  Mode
## prior   2      -none- numeric
## counts  2      -none- numeric
```

```
## means   2      -none- numeric
## scaling 1      -none- numeric
## lev     2      -none- character
## svd     1      -none- numeric
## N       1      -none- numeric
## call    3      -none- call
## terms   3      terms  call
## xlevels 0      -none- list
```

Predict

```
lda.pred <- predict(lda.fit, df.test)

names(lda.pred)
```

```
## [1] "class"     "posterior" "x"
```

Confusion matrix

```
##
#
# Continued based on ISLR 4.6.3 p.161-162
#
#

lda.class <- lda.pred$class

table(lda.class, df.test$Y)
```

```
##
## lda.class   0   1
##         0 314  12
##         1   8 340
```

```
mean(lda.class == df.test$Y)
```

```
## [1] 0.9703264
```

- The confusion matrix is based on the test set.
- The confusion matrix indicates the number of observations correctly predicted not to be in Y.
- And it indicated the number of observations correctly predicted to be in Y.
- The `mean()` function calculates the diagonals over the total.
- These results parallel those from linear regression in Problem 1.

LDA prediction metrics

- Now I will use the function from from above

```
##
#
# Based on functions from above
#

lda.pred <- lda.pred.ftn(Y~X, df.train, df.test)

lda.metrics <- prediction.metrics(df.test$Y, lda.pred)

print.the.metrics(lda.metrics)
```

```
##  OBS =  674 ..................number of observations
##  ACC =  0.9703264 ..................Accuracy
##  TPR =  0.9659091 ..................True Positive Rate
##  TNR =  0.9751553 ..................True Negative Rate
##  PPV =  0.9770115 ..................Positive Predictive Value (Precision)
##  NPV =  0.9631902 ..................Negative Predictive Value
##  FDR =  0.02298851 ..................False Discover Rate
##  FPR =  0.02484472 ..................False Positive Rate
##  TP  =  8 ..................True Positives
##  TN  =  314 ..................True Negatives
##  FP  =  314 ..................False Positives
##  FN  =  12 ..................False Negatives
##  P   =  352 ..................Positives
##  N   =  322 ..................Negatives
```

References

- Harvard "Elements of Statistical Learning" (2021) taught by professors Dr. Sivachenko, Dr. Farutin
- Book "An Introduction to Statistical Learning with Applications in R" (ISLR) by Gareth James et al
- otrevizo GitHub R/toolbox/prediction_metrics_classification.Rmd