Statistics. Simplified.
STATOLOGY

MENU

OCTOBER 30, 2020 BY ZACH

Linear Discriminant Analysis in R (Step-by-Step)

Linear discriminant analysis is a method you can use when you have a set of predictor variables and you'd like to classify a response variable into two or more classes.

This tutorial provides a step-by-step example of how to perform linear discriminant analysis in R.

Step 1: Load Necessary Libraries

First, we'll load the necessary libraries for this example:

```
library(MASS)
library(ggplot2)
```

Step 2: Load the Data

For this example, we'll use the built-in **iris** dataset in R. The following code shows how to load and view this dataset:

```
#attach iris dataset to make it easy to work with
attach(iris)

#view structure of dataset
str(iris)

'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0
 $ Species     : Factor w/ 3 levels "setosa","versi
```

We can see that the dataset contains 5 variables and 150 total observations.

For this example we’ll build a linear discriminant analysis model to classify which species a given flower belongs to.

We’ll use the following predictor variables in the model:

- Sepal.length
- Sepal.Width
- Petal.Length
- Petal.Width

And we’ll use them to predict the response variable *Species*, which takes on the following three potential classes:

- setosa
- versicolor
- virginica

Step 3: Scale the Data

One of the key assumptions of linear discriminant analysis is that each of the predictor variables have the same variance. An easy way to assure that this assumption is met is to scale each variable such that it has a mean of 0 and a standard deviation of 1.

We can quickly do so in R by using the **scale()** function:

```
#scale each predictor variable (i.e. first 4 columns
iris[1:4] <- scale(iris[1:4])
```

We can use the apply() function to verify that each predictor variable now has a mean of 0 and a standard deviation of 1:

```
#find mean of each predictor variable
apply(iris[1:4], 2, mean)

 Sepal.Length   Sepal.Width  Petal.Length   Petal.W
-4.484318e-16  2.034094e-16 -2.895326e-17 -3.663049

#find standard deviation of each predictor variable
apply(iris[1:4], 2, sd)

Sepal.Length  Sepal.Width Petal.Length  Petal.Width
           1            1            1            1
```

Step 4: Create Training and Test Samples

Next, we'll split the dataset into a training set to train the model on and a testing set to test the model on:

```
#make this example reproducible
set.seed(1)

#Use 70% of dataset as training set and remaining 3
sample <- sample(c(TRUE, FALSE), nrow(iris), replac
train <- iris[sample, ]
test <- iris[!sample, ]
```

4:52

Step 5: Fit the LDA Model

Next, we'll use the lda() function from the **MASS** package to fit the LDA model to our data:

```
#fit LDA model
model <- lda(Species~., data=train)

#view model output
model

Call:
lda(Species ~ ., data = train)

Prior probabilities of groups:
    setosa versicolor  virginica 
 0.3207547  0.3207547  0.3584906 

Group means:
           Sepal.Length Sepal.Width Petal.Length Pe
setosa       -1.0397484   0.8131654   -1.2891006  -1
versicolor    0.1820921  -0.6038909    0.3403524   0
virginica     0.9582674  -0.1919146    1.0389776   1

Coefficients of linear discriminants:
                    LD1        LD2
Sepal.Length  0.7922820  0.5294210
Sepal.Width   0.5710586  0.7130743
Petal.Length -4.0762061 -2.7305131
Petal.Width  -2.0602181  2.6326229
```

```
Proportion of trace:
   LD1    LD2 
0.9921 0.0079 
```

4:52

Here is how to interpret the output of the model:

Prior probabilities of group: These represent the proportions of each Species in the training set. For example, 35.8% of all observations in the training set were of species *virginica*.

Group means: These display the mean values for each predictor variable for each species.

Coefficients of linear discriminants: These display the linear combination of predictor variables that are used to form the decision rule of the LDA model. For example:

- **LD1:** .792*Sepal.Length + .571*Sepal.Width – 4.076*Petal.Length – 2.06*Petal.Width
- **LD2:** .529*Sepal.Length + .713*Sepal.Width – 2.731*Petal.Length + 2.63*Petal.Width

Proportion of trace: These display the percentage separation achieved by each linear discriminant function.

Step 6: Use the Model to Make Predictions

Once we've fit the model using our training data, we can use it to make predictions on our test data:

```
#use LDA model to make predictions on test data
predicted <- predict(model, test)

names(predicted)

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

This returns a list with three variables:

- **class:** The predicted class
- **posterior:** The posterior probability that an observation belongs to each class
- **x:** The linear discriminants

We can quickly view each of these results for the first six observations in our test dataset:

```
#view predicted class for first six observations in
head(predicted$class)

[1] setosa setosa setosa setosa setosa setosa
Levels: setosa versicolor virginica

#view posterior probabilities for first six observa
head(predicted$posterior)

   setosa   versicolor    virginica
4       1 2.425563e-17 1.341984e-35
6       1 1.400976e-21 4.482684e-40
7       1 3.345770e-19 1.511748e-37
15      1 6.389105e-31 7.361660e-53
17      1 1.193282e-25 2.238696e-45
18      1 6.445594e-22 4.894053e-41

#view linear discriminants for first six observatio
head(predicted$x)

         LD1        LD2
4   7.150360 -0.7177382
6   7.961538  1.4839408
```

```
7   7.504033  0.2731178
15 10.170378  1.9859027
17  8.885168  2.1026494
18  8.113443  0.7563902
```

We can use the following code to see what percentage of observations the LDA model correctly predicted the Species for:

```
#find accuracy of model
mean(predicted$class==test$Species)

[1] 1
```

It turns out that the model correctly predicted the Species for **100%** of the observations in our test dataset.

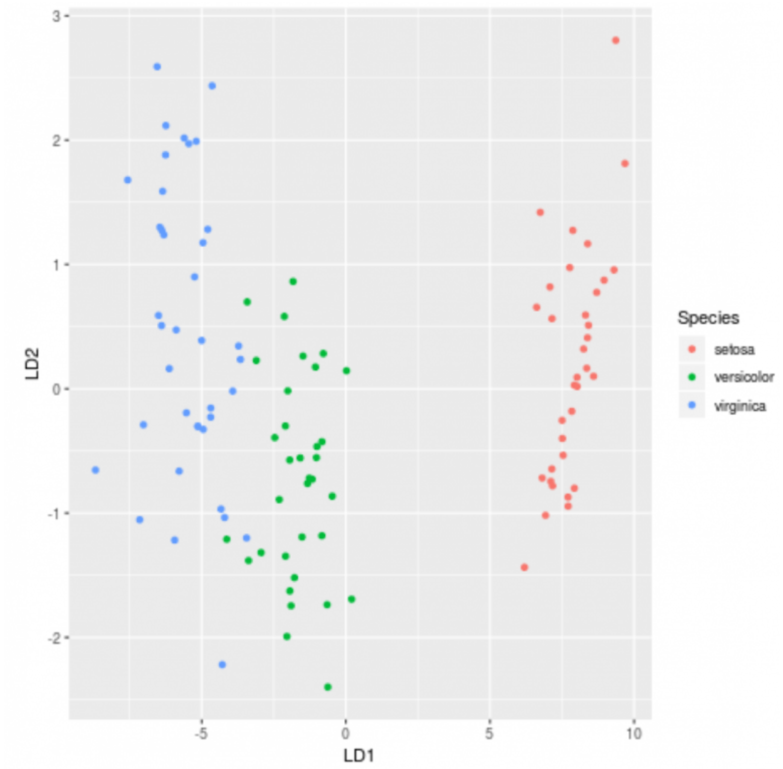
In the real-world an LDA model will rarely predict every class outcome correctly, but this iris dataset is simply built in a way that machine learning algorithms tend to perform very well on it.

Step 7: Visualize the Results

Lastly, we can create an LDA plot to view the linear discriminants of the model and visualize how well it separated the three different species in our dataset:

```
#define data to plot
lda_plot <- cbind(train, predict(model)$x)

#create plot
ggplot(lda_plot, aes(LD1, LD2)) +
  geom_point(aes(color = Species))
```



You can find the complete R code used in this tutorial here.

Published by Zach

View all posts by Zach

PREV

Introduction to Linear Discriminant Analysis

NEXT

How to Calculate Rolling Correlation in R

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment *

4:52

Name *

Email *

Website

POST COMMENT

SEARCH

Search …

ABOUT

Statology is a site that makes learning statistics easy by explaining topics in simple and straightforward ways. **Learn more about us**.

STATOLOGY STUDY

Statology Study is the ultimate online statistics study guide that helps you understand all of the core concepts taught in any elementary statistics course and makes your life so much easier as a student.

STATOLOGY STUDY

CALCULATORS

Try out our free **online statistics calculators** if you're looking for some help finding probabilities, p-values, critical values, sample sizes, expected values, summary statistics, or correlation coefficients.

RECENT POSTS

How to Interpret R Message: The following objects are masked

How to Rename an Object in R (With Examples)

How to Fix in R: invalid type (list) for variable