15 - LDA & QDA, Training/Testing Data. Cross-Validation

Junvie Pailden

SIUE, F2017, Stat 589

October 23, 2017

Case 1: Credit Scoring on German Bank

The German credit data set was obtained from the UCI Machine Learning Repository. The data set, which contains attributes and outcomes on 1000 loan applications.

This dataset classifies people described by a set of attributes/predictors as good or bad credit risks.

- Number of rows = 1000
- Number of attributes = 20

```
credit <- read.csv("germancredit.csv")
names(credit)
```

```
#  [1] "Default"         "checkingstatus1" "duration"
#  [4] "history"         "purpose"         "amount"
#  [7] "savings"         "employ"          "installment"
# [10] "status"          "others"          "residence"
# [13] "property"        "age"             "otherplans"
# [16] "housing"         "cards"           "job"
# [19] "liable"          "tele"            "foreign"
```

Select Numeric Attributes/Predictors Only

```r
credit1 <- credit %>%
  select(Default, duration, amount, installment, age)
head(credit1, 2)
```

```
#   Default duration amount installment age
# 1      No        6   1169           4  67
# 2     Yes       48   5951           2  22
```

Summary Measures by Class Default - Mean

```
credit1 %>%
  group_by(Default) %>%
  summarise_all(mean)
```

```
# # A tibble: 2 x 5
#   Default duration amount installment   age
#    <fctr>    <dbl>  <dbl>       <dbl> <dbl>
# 1      No       19   2985         2.9    36
# 2     Yes       25   3938         3.1    34
```

Summary Measures by Class Default - Std Deviation

```
credit1 %>%
  group_by(Default) %>%
  summarise_all(sd)
```

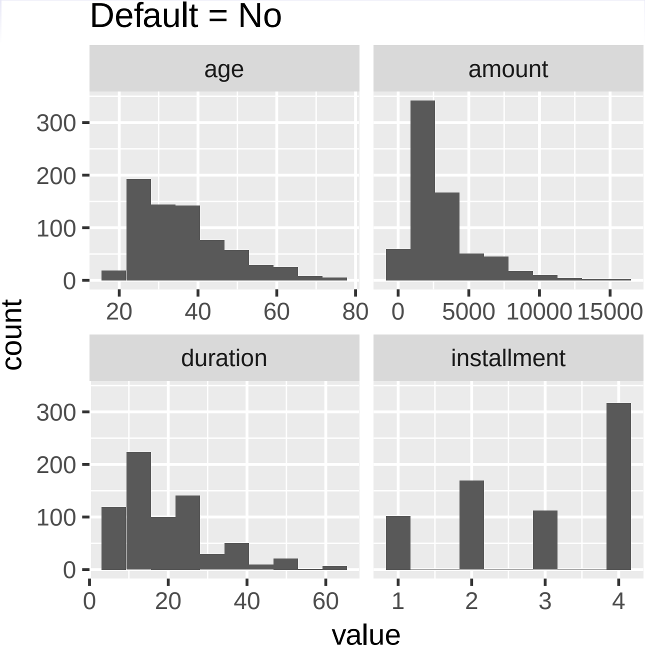
```
# # A tibble: 2 x 5
#   Default duration amount installment   age
#    <fctr>    <dbl>  <dbl>       <dbl> <dbl>
# 1      No       11   2401         1.1    11
# 2     Yes       13   3536         1.1    11
```

Histogram of Attributes

```r
library(tidyr)
library(ggplot2)
credit1.long <- credit1 %>%
  gather(variable, value, -Default)
head(credit1.long, 4)
```

```
#   Default variable value
# 1      No duration     6
# 2     Yes duration    48
# 3      No duration    12
# 4      No duration    42
```

```r
credit1.long %>%
  filter(Default == "No") %>%
  ggplot(aes(x = value)) +
  geom_histogram(bins = 10) +
  facet_wrap(~variable, scales = 'free_x') +
  ggtitle("Default = No")
```

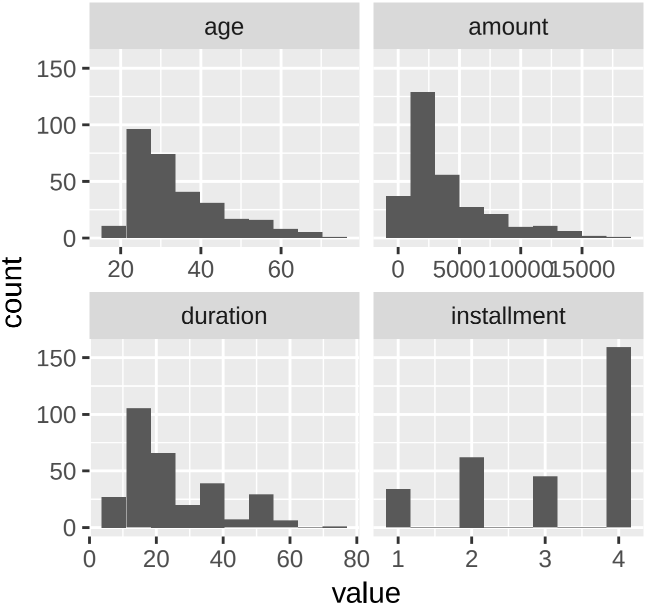


Default = No

age
amount
duration
installment
count
value
0
100
200
300
20
40
60
80
0
5000
10000
15000
0
20
40
60
1
2
3
4

```r
credit1.long %>%
  filter(Default == "Yes") %>%
  ggplot(aes(x = value)) +
  geom_histogram(bins = 10) +
  facet_wrap(~variable, scales = 'free_x') +
  ggtitle("Default = Yes")
```

Default = Yes


age
amount
duration
installment
150
100
50
0
20
40
60
0
5000
10000
15000
0
20
40
60
80
1
2
3
4
count
value

Split Data into Training and Testing Sets

Randomly split data into 70% training and 30% testing.

Build/train the classification (LDA/QDA) model on the training set and check the model performance on the testing set.

```
set.seed(1) # fix seed to get same training set
train <- sample(nrow(credit1), size = 0.7*nrow(credit1))
cred.train <- credit1[train, ]
dim(cred.train)
```

```
# [1] 700   5
```

```
cred.test <- credit1[-train,]
dim(cred.test)
```

```
# [1] 300   5
```

Build LDA classifier/model on Training Data

```
library(MASS)
# Default ~. includes all attributes
# Build model on training data
cred.lda <- lda(Default ~. , data = cred.train)
# Confusion Matrix on Training Data
(cm.lda.train <- table(Predicted = predict(cred.lda)$class
      Default = cred.train$Default))
```

```
#           Default
# Predicted  No Yes
#       No  475 178
#       Yes  20  27
```

```
sum(diag(prop.table(cm.lda.train))) # accuracy
```

```
# [1] 0.72
```

LDA prediction for new observation

New individual apply for a loan.

```
predict(cred.lda,
  newdata = data.frame(duration = 6, amount = 1100,
                       installment = 4, age = 67))
```

```
# $class
# [1] No
# Levels: No Yes
#
# $posterior
#     No  Yes
# 1 0.88 0.12
#
# $x
#    LD1
# 1 -1.9
```

LDA prediction on the test data

```
cred.test.lda <- predict(cred.lda, newdata = cred.test)
(cm.lda.test <- table(Predicted = cred.test.lda$class,
      Default = cred.test$Default))
```

```
#          Default
# Predicted  No Yes
#       No  199  84
#       Yes   6  11
```

```
sum(diag(prop.table(cm.lda.test))) # accuracy
```

```
# [1] 0.7
```

Build QDA classifier/model on Training Data

```
# Default ~. includes all attributes
# Build model on training data
cred.qda <- qda(Default ~. , data = cred.train)
# Confusion Matrix on Training Data
(cm.qda.train <- table(Predicted = predict(cred.qda)$class,
      Default = cred.train$Default))
```

```
#           Default
# Predicted  No Yes
#       No  442 153
#       Yes  53  52
```

```
sum(diag(prop.table(cm.qda.train))) # accuracy
```

```
# [1] 0.71
```

QDA prediction on the test data

```
cred.test.qda <- predict(cred.qda, newdata = cred.test)
(cm.qda.test <- table(Predicted = cred.test.qda$class,
      Default = cred.test$Default))
```

```
#          Default
# Predicted  No Yes
#       No  188  68
#       Yes  17  27
```

```
sum(diag(prop.table(cm.qda.test))) # accuracy
```

```
# [1] 0.72
```

With equal prior and equal cost of misclassification, on the test data QDA (72%) performed slightly better than LDA (70%).

Assign Unequal Priors I

```
table(credit1$Default)/nrow(credit1)
```

```
#
#  No Yes
# 0.7 0.3
```

```
cred.qda2 <- qda(Default ~. , data = cred.train,
                 prior = c(0.7, 0.3))
cred.test.qda2 <- predict(cred.qda2, newdata = cred.test)
(cm.qda.test <- table(Predicted = cred.test.qda2$class,
      Default = cred.test$Default))
```

Assign Unequal Priors II

```
#          Default
# Predicted  No Yes
#       No  186  68
#       Yes  19  27
```

```r
sum(diag(prop.table(cm.qda.test))) # accuracy
```

```
# [1] 0.71
```

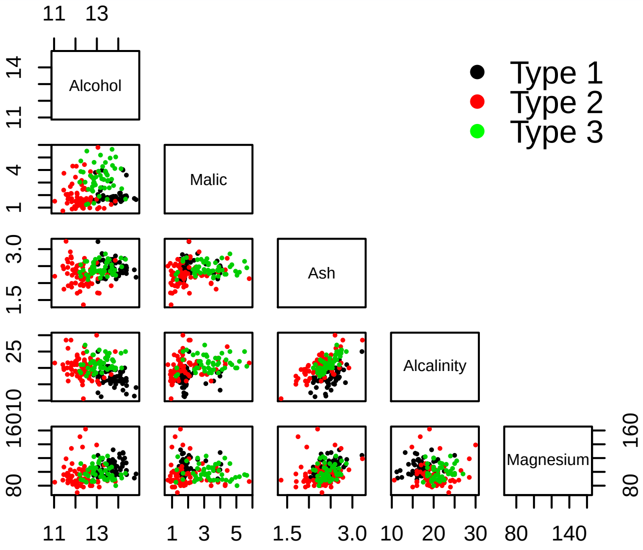
Case 2: Wine Data

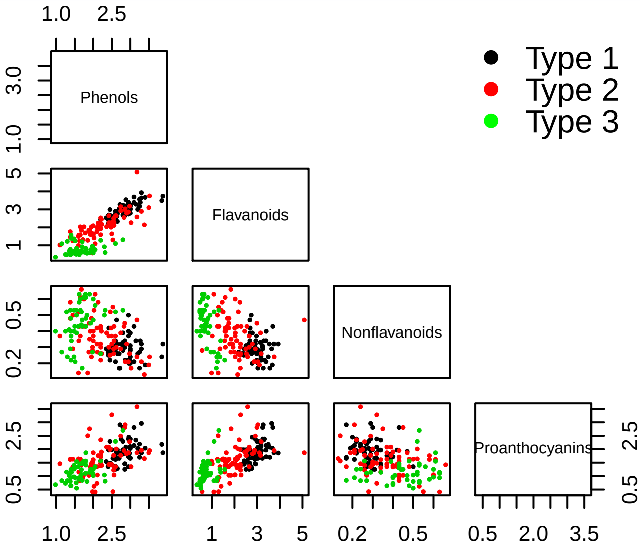
We consider the wine data set first used in Lecture 12: PCA additional example.

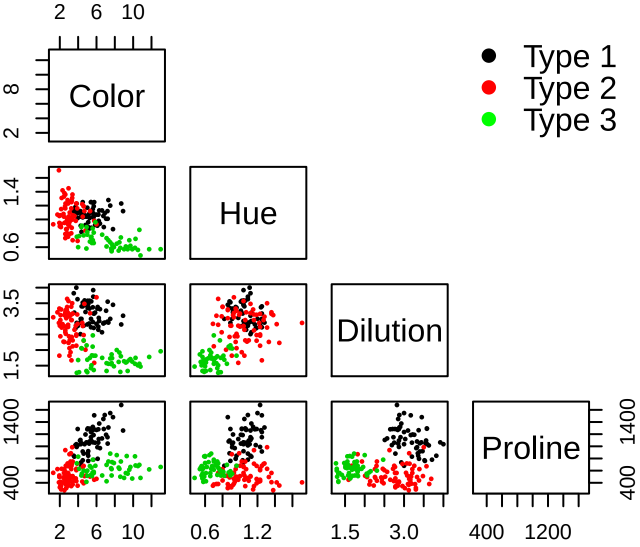
The wine data is the result of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.

```r
data(wine, package = 'rattle.data')
names(wine)
```

```
#  [1] "Type"            "Alcohol"         "Malic"
#  [4] "Ash"             "Alcalinity"      "Magnesium"
#  [7] "Phenols"         "Flavanoids"      "Nonflavanoids"
# [10] "Proanthocyanins" "Color"           "Hue"
# [13] "Dilution"        "Proline"
```


Alcohol
Malic
Ash
Alcalinity
Magnesium
11
13
14
11
4
1
3.0
1.5
25
10
160
80
160
80
11
13
1
3
5
1.5
3.0
10
20
30
80
140
● Type 1
● Type 2
● Type 3


Phenols
Flavanoids
Nonflavanoids
Proanthocyanins
Type 1
Type 2
Type 3
1.0
2.5
3.0
1.0
5
3
1
0.5
0.2
2.5
0.5
1.0
2.5
1
3
5
0.2
0.5
0.5
2.0
3.5
2.5
0.5


Color
Hue
Dilution
Proline
● Type 1
● Type 2
● Type 3
2
6
10
2
8
0.6
1.4
1.5
3.5
400
1400
0.6
1.2
1.5
3.0
400
1200
400
1400

LDA on Wine Data

The purpose of linear discriminant analysis (LDA) in this example is to find the linear combinations of the original variables (the 13 chemical concentrations here) that gives the best possible separation between the groups (wine `Type`) in our data set.

If we want to separate the wines by `Type`, the wines come from three different `Type`, so the number of groups $G = 3$, and the number of variables is 13 $p = 13$. The maximum number of useful discriminant functions that can separate the wines by `Type` is the minimum of $G - 1$ and p which is 2. Thus, we can find at most 2 useful discriminant functions to separate the wines by cultivar, using the 13 chemical concentration variables.

LDA on Wine Data I

```r
wine.lda <- MASS::lda(Type ~ ., data=wine)
wine.pred.lda <- predict(wine.lda, wine)
(cm.lda.wine <- table(Predicted = wine.pred.lda$class,
      Default = wine$Type))
```

```
#          Default
# Predicted  1  2  3
#         1 59  0  0
#         2  0 71  0
#         3  0  0 48
```

```r
sum(diag(prop.table(cm.lda.wine))) # accuracy
```

```
# [1] 1
```

LDA on Wine Data II

Don’t get too excited about getting an accuracy of 100% when predicting values based on models that were built using those values.

Many data scientists think that the failure to develop a stronger predictive model is due to the problem of “over-fitting.“”

Over-fitting occurs when a model/procedure fails to accurately predict future events because it is too closely tailored to past events.

k-fold Cross Validation

The k-fold cross validation is implemented by randomly dividing the set of observations into k groups, or folds, of approximately equal size. The first fold is treated as a testing set, and the method is fit on the remaining k-1 folds (training set).

The procedure (classification) is then applied on the observations in the first fold (testing set) and the error (misclassication) rate is computed.

This procedure is repeated k times; each time, a different group of observations is treated as a validation set. This process results in k estimates of error rate. The k-fold CV estimate of the error rate is computed by averaging these k values.

Leave one out cross validation (LOOCV), LDA

The LOOCV cross-validation approach is a special case of k-fold cross-validation in which k = n.

```
wine.lda.cv <- MASS::lda(Type ~ ., data=wine, CV = TRUE)
(cm.lda.wine.cv <- table(Predicted = wine.lda.cv$class,
      Default = wine$Type))
```

```
#           Default
# Predicted  1  2  3
#         1 59  1  0
#         2  0 69  0
#         3  0  1 48
```

```
sum(diag(prop.table(cm.lda.wine.cv))) # accuracy
```

```
# [1] 0.99
```

QDA on Wine Data, LOOCV

```
wine.qda.cv <- MASS::qda(Type ~ ., data=wine, CV = TRUE)
(cm.qda.wine.cv <- table(Predicted = wine.qda.cv$class,
      Default = wine$Type))
```

```
#          Default
# Predicted  1  2  3
#         1 59  1  0
#         2  0 70  0
#         3  0  0 48
```

```
sum(diag(prop.table(cm.qda.wine.cv))) # accuracy
```

```
# [1] 0.99
```