Comparing discriminant rules. ROC curve and other methods

Gregoire Gasparini, Aurora Hofman, Sarah Musiol, Beatriu Tort

10 de marzo de 2020

From the description file:

The “spam” concept is diverse: advertisements for products/web sites, make money fast schemes, chain letters, pornography… Our collection of spam e- mails came from our postmaster and individuals who had filed spam. Our collection of non-spam e-mails came from filed work and personal e-mails, and hence the word ’george’ and the area code ’650’ are indicators of non-spam. These are useful when constructing a personalized spam filter. One would either have to blind such non-spam indicators or get a very wide collection of non-spam to generate a general purpose spam filter.

Attribute Information:

-The last column of ’spambase.data’ denotes whether the e-mail was considered spam (1) or not (0), i.e. unsolicited commercial e-mail. -Most of the attributes indicate whether a particular word or character was frequently occurring in the e-mail. -The run-length attributes (55-57) measure the length of sequences of consecutive capital letters.

1. Use the script spam.R to read the data from the SPAM e-mail database.

```r
source("Discriminant_Rules/spam_email_database/spam.R")
# why is this not working????
```

```r
spam <- read.table("spam_email_database/spambase.data", sep = ",")

spam.names <-
  c(
    read.table(
      "spam_email_database/spambase.names",
      sep = ":",
      skip = 33,
      nrows = 53,
      as.is = TRUE
    )[, 1],
    "char_freq_#",
    read.table(
      "spam_email_database/spambase.names",
      sep = ":",
      skip = 87,
      nrows = 3,
      as.is = TRUE
    )[, 1],
    "spam.01"
  )

names(spam) <- spam.names
```

2. Dividing data in training set and test set

Task description:

Divide the data into two parts: 2/3 for the training sample, 1/3 for the test sample. You should do it in a way that SPAM e-mail are 2/3 in the training sample and 1/3 in the test sample, and that the same happens for NO SPAM e-mails.

```
ind.train <- sample(1:nrow(spam), 2 / 3 * nrow(spam))

spam_train <- spam[ind.train, ]
spam_test <- spam[-ind.train, ]
```

3. Classification using the Training sample

Task description:

Consider the following three classification rules:

-Logistic regression fitted by maximum likelihood (IRWLS, glm). -Logistic regression fitted by Lasso (glment). -k-nn binary regression (you can use your own implementation or functions knn and knn.cv from the R package class).

Use the training sample to fix the tunning parameters (when needed) and to estimate the model parameters (when needed).

```
response <- "spam.01"
explanatory <-
  colnames(spam_train)[colnames(spam_train) != response]
# Logistic regression by maximum likelihood
glm_spam <-
  glm(spam_train$spam.01 ~ ., family = "binomial", data = spam_train)
glm_spam




# by IRWLS from https://www.stt.msu.edu/users/pszhong/R-code-for-lecture-12.txt

########################################################
## Applying the IRWLS method to logistic regression model
## with Bernoulli response
########################################################

## Step 1: Set the initial estimates

beta <- c(0.8, 1.5)
eta <- beta[1] + beta[2] * X
mu <- exp(eta) / (1 + exp(eta))
```

```
## Step 2: Compute the adjusted dependent response.

nz <- eta + (Y - mu) / (mu * (1 - mu))

## Step 3: Compute weights for the weighted least square.

w <- mu * (1 - mu)

## Step 4: Obtain the initial estimates of beta0 and beta1 through the following weighted least square

lmod <- lm(nz ~ X, weights = w)

## Step 5: Repeat the above steps until the convergence of beta0 and beta1.

for (i in 1:5)
{
  eta <- lmod$fit
  mu <- exp(eta) / (1 + exp(eta))  ## inverse of logit function
  nz <- eta + (Y - mu) / (mu * (1 - mu))
  w <- mu * (1 - mu)
  lmod <- lm(nz ~ X, weights = w)
  cat(i, coef(lmod), "\n")
}
```

```
# Logistic regression by Lasso
glmnet_spam <-
  glmnet(spam_train[, explanatory], spam_train$spam.01, family = "binomial")
```

```
# KNN binary regression
library(class)

knn_spam <- knn()
```

4. Use the test sample to compute and plot the ROC curve for each rule.

5. Compute also the misclassification rate for each rule when using the cut point c = 1/2.

```
c = 1/2
```

6. Compute l_{val} for each rule.