ISL - Chapter 5 Exercises
Resampling Methods

An introduction to Statistical Learning, with Applications in R
- G. James, D. Witten, T. Hastie, R. Tibshirani

Thu Nguyen

26 June, 2019

Contents

Exercise 5

In Chapter 4, we used logistic regression to predict the probability of `default` using `income` and `balance` on the `Default` data set. We will now estimate the test error of this logistic regression model using the validation set approach. Do not forget to set a random seed before beginning your analysis.

(a) Fit a logistic regression model that uses `income` and `balance` to predict `default`.
(b) Using the validation set approach, estimate the test error of this model. In order to do this, you must perform the following steps:

   i. Split the sample set into a training set and a validation set.
   ii. Fit a multiple logistic regression model using only the training observations.
   iii. Obtain a prediction of default status for each individual in the validation set by computing the posterior probability of default for that individual, and classifying the individual to the `default` category if the posterior probability is greater than 0.5.
   iv. Compute the validation set error, which is the fraction of the observations in the validation set that are misclassified.

(c) Repeat the process in (b) three times, using three different splits of the observations into a training set and a validation set. Comment on the results obtained.
(d) Now consider a logistic regression model that predicts the probability of `default` using `income`, `balance`, and a dummy variable for `student.` Estimate the test error for this model using the validation set approach. Comment on whether or not including a dummy variable for `student` leads to a reduction in the test error rate.

(a) Logistic Regression: `default` ~ `income` + `balance`

```r
library(ISLR)
attach(Default)
glm.fits <- glm(default ~ income + balance, data = Default, family = 'binomial')
summary(glm.fits)
```

```
##
## Call:
## glm(formula = default ~ income + balance, family = "binomial",
##     data = Default)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4725  -0.1444  -0.0574  -0.0211   3.7245
##
## Coefficients:
##                   Estimate    Std. Error z value             Pr(>|z|)
## (Intercept) -11.540468437   0.434756357 -26.545 < 0.0000000000000002 ***
## income        0.000020809   0.000004985   4.174            0.0000299 ***
## balance       0.005647103   0.000227373  24.836 < 0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1579.0  on 9997  degrees of freedom
## AIC: 1585
##
## Number of Fisher Scoring iterations: 8
```

(b) Validation set approach

```r
n <- nrow(Default)
set.seed(1)
# (i): Split into train and test, 70% training, 30% test
train <- sample(n, round(n*.7,0))
# (ii): GLM model on train set
glm.fits <- glm(default ~ income + balance, data = Default, family = 'binomial', subset = train)
# (iii): Prediction on test set
glm.probs <- predict(glm.fits, newdata = Default, type = 'response')[-train]
k <- nrow(Default) - length(train)
glm.pred <- rep('No', k)
glm.pred[glm.probs > .5] <- 'Yes'
# (iv): Test set error
default.test <- Default[-train,]$default
print(paste('Validation set MSE:', mean(glm.pred != default.test)))
```

```
## [1] "Validation set MSE: 0.028"
```

(c) Repeating Validation set approaches

```r
set.seed(1)
valid.set.3 <- c()
for (i in 1:3) {
  set.seed(i*7)
  # (i): Split into train and test
  train <- sample(n, round(n*.7,0))
  # (ii): GLM model on train set
  glm.fits <- glm(default ~ income + balance, data = Default, family = 'binomial', subset = train)
  # (iii): Prediction on test set
  glm.probs <- predict(glm.fits, newdata = Default, type = 'response')[-train]
  k <- nrow(Default) - length(train)
  glm.pred <- rep('No', k)
  glm.pred[glm.probs > .5] <- 'Yes'
  # (iv): Test set error
  default.test <- Default[-train,]$default
  valid.set.3[i] <- round(mean(glm.pred != default.test),4)
}
valid.set.3
```

```
## [1] 0.0250 0.0277 0.0260
```

(d) Logistic Regression: `default` ~ `income` + `balance` + `studnent`

```r
valid.set.all.3 <- c()
for (i in 1:3) {
  set.seed(i*7)
  # (i): Split into train and test
  train <- sample(n, round(n*.7,0))
  # (ii): GLM model on train set
  glm.fits <- glm(default ~ income + balance + student,
                  data = Default, family = 'binomial', subset = train)
  # (iii): Prediction on test set
  glm.probs <- predict(glm.fits, newdata = Default, type = 'response')[-train]
  k <- nrow(Default) - length(train)
  glm.pred <- rep('No', k)
  glm.pred[glm.probs > .5] <- 'Yes'
  # (iv): Test set error
  default.test <- Default[-train,]$default
  valid.set.all.3[i] <- round(mean(glm.pred != default.test),4)
}
valid.set.all.3
```

```
## [1] 0.0257 0.0280 0.0270
```

```r
data.frame('Income_Balance' = valid.set.3,
           'Income_Balance_Student' = valid.set.all.3)
```

```
##   Income_Balance Income_Balance_Student
## 1         0.0250                 0.0257
## 2         0.0277                 0.0280
## 3         0.0260                 0.0270
```

Comment: including `student` does not appear to significantly reduce MSE.

■

Exercise 6

We continue to consider the use of a logistic regression model to predict the probability of `default` using `income` and `balance` on the `Default` data set. In particular, we will now compute estimates for the standard errors of the `income` and `balance` logistic regression coefficients in two different ways: (1) using the bootstrap, and (2) using the standard formula for computing the standard errors in the `glm()` function. Do not forget to set a random seed before beginning your analysis.

(a) Using the `summary()` and `glm()` functions, determine the estimated standard errors for the coefficients associated with `income` and `balance` in a multiple logistic regression model that uses both predictors.
(b) Write a function, `boot.fn()`, that takes as input the `Default` data set as well as an index of the observations, and that outputs the coefficient estimates for `income` and `balance` in the multiple logistic regression model.
(c) Use the `boot()` function together with your `boot.fn()` function to estimate the standard errors of the logistic regression coefficients for `income` and `balance`.
(d) Comment on the estimated standard errors obtained using the `glm()` function and using your bootstrap function.

(a) Estimates of the standard errors of the coefficients by a logistic regression

```r
glm.fits <- glm(default ~ income + balance, data = Default, family = 'binomial')
summary(glm.fits)
```

```
##
## Call:
## glm(formula = default ~ income + balance, family = "binomial",
##     data = Default)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4725  -0.1444  -0.0574  -0.0211   3.7245
##
## Coefficients:
##                  Estimate    Std. Error z value             Pr(>|z|)
## (Intercept) -11.540468437   0.434756357 -26.545 < 0.0000000000000002 ***
## income        0.000020809   0.000004985   4.174            0.0000299 ***
## balance       0.005647103   0.000227373  24.836 < 0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1579.0  on 9997  degrees of freedom
## AIC: 1585
##
## Number of Fisher Scoring iterations: 8
```

From the summary above:

- `income`: SE for coefficient: 0.000004985
- `balance`: SE for coefficient: 0.000227373

(b) Bootstrapping the coefficient estimates

```
set.seed(1)
boot.fn <- function(data, index) {
  return(coef(glm.fits <- glm(default ~ income + balance, data = data, subset = index, family = 'binomial'
}
boot.fn(Default, sample(nrow(Default), nrow(Default), replace = T))
```

```
##     (Intercept)          income         balance
## -11.15929684807   0.00002134762   0.00541922542
```

(c) Bootstrapping the standard errors of the coefficients

```
library(boot)
set.seed(1)
boot(Default, boot.fn, 1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Default, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##            original             bias       std. error
## t1* -11.54046843668 -0.00800837897342 0.423927293556
## t2*   0.00002080898  0.00000005870933 0.000004582525
## t3*   0.00564710294  0.00000229997044 0.000226795468
```

From the summary above, after *Bootstrapping*:

- `income`: SE for coefficient: 0.000004582525
- `balance`: SE for coefficient: 0.000226795468

(d) Comparison of estimated standard errors

```
var <- c('income', 'balance')
glm.se <- c(0.000004985, 0.000227373)
boot.se <- c(0.000004582525, 0.000226795468)
data.frame(Variables = var, GLM = glm.se, Bootstrap = boot.se)
```

```
##   Variables         GLM      Bootstrap
## 1    income 0.000004985 0.000004582525
## 2   balance 0.000227373 0.000226795468
```

■

Exercise 7

In Sections 5.3.2 and 5.3.3, we saw that the `cv.glm()` function can be used in order to compute the LOOCV test error estimate. Alternatively, one could compute those quantities using just the `glm()` and `predict.glm()` functions, and a for loop. You will now take this approach in order to compute the LOOCV error for a simple logistic regression model on the `Weekly` data set. Recall that in the context of classification problems, the LOOCV error is given in (5.4).

(a) Fit a logistic regression model that predicts `Direction` using `Lag1` and `Lag2`.
(b) Fit a logistic regression model that predicts `Direction` using `Lag1` and `Lag2` using all but the first observation.
(c) Use the model from (b) to predict the direction of the first observation. You can do this by predicting that the first observation will go up if P(`Direction="Up"|Lag1, Lag2`) > 0.5. Was this observation correctly classified?
(d) Write a for loop from $i = 1$ to $i = n$, where n is the number of observations in the data set, that performs each of the following steps:

    i. Fit a logistic regression model using all but the ith observation to predict `Direction` using `Lag1` and `Lag2`.
    ii. Compute the posterior probability of the market moving up for the i^{th} observation.
    iii. Use the posterior probability for the i^{th} observation in order to predict whether or not the market moves up.
    iv. Determine whether or not an error was made in predicting the direction for the i^{th} observation. If an error was made, then indicate this as a 1, and otherwise indicate it as a 0.

(e) Take the average of the n numbers obtained in (d)iv in order to obtain the LOOCV estimate for the test error. Comment on the results.

(a) Logistic Regression on all observations: `Direction` ~ `Lag1` + `Lag2`

```
attach(Weekly)
glm.fits.all <- glm(Direction ~ Lag1 + Lag2, data = Weekly, family = binomial)
glm.probs.all <- predict(glm.fits, data = Weekly, type = 'response')
summary(glm.fits.all)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2, family = binomial, data = Weekly)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.623  -1.261   1.001   1.083   1.506
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.22122    0.06147   3.599 0.000319 ***
## Lag1        -0.03872    0.02622  -1.477 0.139672
## Lag2         0.06025    0.02655   2.270 0.023232 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1496.2  on 1088  degrees of freedom
## Residual deviance: 1488.2  on 1086  degrees of freedom
```

```
## AIC: 1494.2
##
## Number of Fisher Scoring iterations: 4
```

(b) Logistic Regression on all observations but the 1^{th} observation: `Direction ~ Lag1 + Lag2`

```
glm.fits.b <- glm(Direction ~ Lag1 + Lag2, data = Weekly[-1,], family = binomial)
glm.probs.b <- predict(glm.fits.b, data = Weekly[-1,], type = 'response')
summary(glm.fits.b)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2, family = binomial, data = Weekly[-1,
##     ])
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6258  -1.2617   0.9999   1.0819   1.5071
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.22324    0.06150   3.630 0.000283 ***
## Lag1        -0.03843    0.02622  -1.466 0.142683
## Lag2         0.06085    0.02656   2.291 0.021971 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1494.6  on 1087  degrees of freedom
## Residual deviance: 1486.5  on 1085  degrees of freedom
## AIC: 1492.5
##
## Number of Fisher Scoring iterations: 4
```

(c) Model prediction at threshold $.5$

```
prob <- predict(glm.fits.b, newdata = Weekly[1,], type = 'response')
pred <- ifelse(prob > .5, 'Up', 'Down')
print(paste0('Prediction: ', pred, '; Correct prediction? ', pred == Weekly$Direction[1]))
```

```
## [1] "Prediction: Up; Correct prediction? FALSE"
```

(d) ***LOOCV*** for each $i = 1, 2, \dots, n$

```r
n <- nrow(Weekly)
loocv <- c()
for (i in 1:n) {
  glm.fits.b <- glm(Direction ~ Lag1 + Lag2, data = Weekly[-i,], family = binomial)
  prob <- predict(glm.fits.b, newdata = Weekly[i,], type = 'response')
  loocv[i] <- ifelse(prob > .5, 'Up', 'Down') == Weekly$Direction[i]
}
```

(e) Accuracy of *LOOCV*

```r
loocv <- mean(loocv)
baseline <- table(Weekly$Direction)[2]/n
data.frame(Model = c('LOOCV', 'Baseline'), Accuracy = c(loocv, baseline))
```

```
##       Model  Accuracy
##       LOOCV 0.5500459
## Up Baseline 0.5555556
```

Comment: *LOOCV* does not appear to have effects on the accuracy: the prediction accuracy is the same as the baseline: by predicting the most common `Direction: Up`.

■