Atividade avaliativa 3 - Resampling

Jonatas Varella

05 de maio de 2019

Exercício 05

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.

```r
options(scipen=999)
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 3.5.3
```

```r
set.seed(1)
bd = Default
reg =  glm(default ~ balance + income, data = bd, family = "binomial")
summary(reg)
```

```
##
## Call:
## glm(formula = default ~ balance + income, family = "binomial",
##     data = bd)
##
## 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 ***
## balance       0.005647103   0.000227373  24.836 < 0.0000000000000002 ***
## income        0.000020809   0.000004985   4.174            0.0000299 ***
## ---
## 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) 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. 5.4 Exercises 199 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.

```
set.seed(1)
train = sample(1:nrow(bd), nrow(bd)*.7)
bd.train = bd[train, ]
bd.test = bd[-train, ]
```

```
reg1 =  glm(default ~ balance + income, data = bd.train, family = "binomial")
summary(reg1)
```

```
##
## Call:
## glm(formula = default ~ balance + income, family = "binomial",
##     data = bd.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5199  -0.1369  -0.0538  -0.0192   3.7568
##
## Coefficients:
##                    Estimate    Std. Error z value             Pr(>|z|)
## (Intercept) -11.772664368   0.531040524 -22.169 < 0.0000000000000002 ***
## balance       0.005787210   0.000277882  20.826 < 0.0000000000000002 ***
## income        0.000021223   0.000006015   3.529             0.000418 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2050.5  on 6999  degrees of freedom
## Residual deviance: 1076.2  on 6997  degrees of freedom
## AIC: 1082.2
##
## Number of Fisher Scoring iterations: 8
```

```
reg.probs = predict(reg1, bd.test, type = "response")
reg.pred = ifelse(reg.probs > .5, "Yes", "No")
```

```
table(reg.pred, bd.test$default)
```

```
##
## reg.pred   No  Yes
##      No  2889   72
##      Yes   12   27
```

```
mean(reg.pred == bd.test$default)
```

```
## [1] 0.972
```

(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.

```
set.seed(2)
train = sample(1:nrow(bd), nrow(bd)*.7)
bd.train = bd[train, ]
bd.test = bd[-train, ]
reg1 =  glm(default ~ balance + income, data = bd.train, family = "binomial")
```

```r
reg.probs = predict(reg1, bd.test, type = "response")
reg.pred = ifelse(reg.probs > .5, "Yes", "No")
table(reg.pred, bd.test$default)
```

```
##
## reg.pred   No  Yes
##      No  2900   65
##      Yes    7   28
```

```r
mean(reg.pred == bd.test$default)
```

```
## [1] 0.976
```

```r
set.seed(3)
train = sample(1:nrow(bd), nrow(bd)*.7)
bd.train = bd[train, ]
bd.test = bd[-train, ]
reg1 =  glm(default ~ balance + income, data = bd.train, family = "binomial")
reg.probs = predict(reg1, bd.test, type = "response")
reg.pred = ifelse(reg.probs > .5, "Yes", "No")
table(reg.pred, bd.test$default)
```

```
##
## reg.pred   No  Yes
##      No  2908   57
##      Yes   11   24
```

```r
mean(reg.pred == bd.test$default)
```

```
## [1] 0.9773333
```

```r
set.seed(4)
train = sample(1:nrow(bd), nrow(bd)*.7)
bd.train = bd[train, ]
bd.test = bd[-train, ]
reg1 =  glm(default ~ balance + income, data = bd.train, family = "binomial")
reg.probs = predict(reg1, bd.test, type = "response")
reg.pred = ifelse(reg.probs > .5, "Yes", "No")
table(reg.pred, bd.test$default)
```

```
##
## reg.pred   No  Yes
##      No  2893   63
##      Yes   14   30
```

```r
mean(reg.pred == bd.test$default)
```

```
## [1] 0.9743333
```

Resposta

Em ambos os casos, observa-se que o percentual de acerto é de 97%, demonstrando que independente da amostra, os resultados são similares.

(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.

```
set.seed(1)
train = sample(1:nrow(bd), nrow(bd)*.7)
bd.train = bd[train, ]
bd.test = bd[-train, ]
reg2 =  glm(default ~ balance + income + student, data = bd.train, family = "binomial")
reg.probs = predict(reg2, bd.test, type = "response")
reg.pred = ifelse(reg.probs > .5, "Yes", "No")
table(reg.pred, bd.test$default)
```

```
##
## reg.pred   No  Yes
##      No  2889   71
##      Yes   12   28
```

```
mean(reg.pred == bd.test$default)
```

```
## [1] 0.9723333
```

Resposta

O percentual de acerto é similar entre o modelo com e sem a variavel "student". Ambos com 97%

Exercício 06

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.

```
reg1 =  glm(default ~ balance + income, data = bd, family = "binomial")
summary(reg1)
```

```
##
## Call:
## glm(formula = default ~ balance + income, family = "binomial",
##     data = bd)
##
## 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 ***
## balance       0.005647103   0.000227373  24.836 < 0.0000000000000002 ***
## income        0.000020809   0.000004985   4.174            0.0000299 ***
## ---
## 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)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.

```
boot.fn = function(data,index) +
  return(coef(glm(default ~ balance + income, data = data, family = "binomial", subset = index)))
```

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

```
library(boot)
boot(bd, boot.fn,1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = bd, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##            original             bias       std. error
## t1* -11.54046843668 -0.00831566486939 0.424036814602
## t2*   0.00564710294  0.00000260022452 0.000226866294
## t3*   0.00002080898  0.00000005412312 0.000004589265
```

d)Comment on the estimated standard errors obtained using the glm() function and using your bootstrap function.

Os erros estimados através da função summary() e através da função boot() são similares.

Exercício 08

We will now perform cross-validation on a simulated data set.

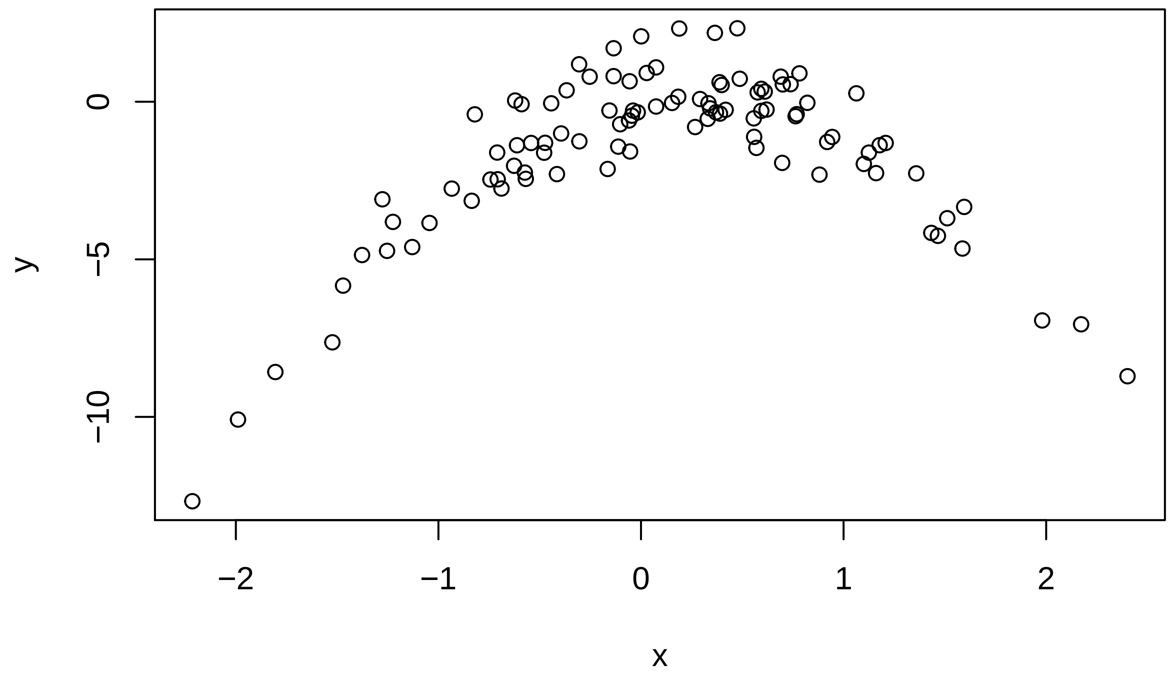
a)Generate a simulated data set as follows:

```
set.seed(1)
x=rnorm(100)
y=x-2*x^2+rnorm (100)
```

Resposta $n = 100$ p (Preditor) $= 2$

b) Create a scatterplot of X against Y . Comment on what you find.

```
plot(x = x, y = y)
```



c) Set a random seed, and then compute the LOOCV errors that result from fitting the following four models using least squares:

$$i).Y = 0 + 1X + \epsilon$$

$$ii).Y = 0 + 1X + {}_2X^2 + \epsilon$$

$$iii).Y = 0 + 1X + {}_2X2_3X^3 + \epsilon$$

$$iv).Y = 0 + 1X + {}_2X2_3X^3 + {}_4X^4 + \epsilon$$

Note you may find it helpful to use the data.frame() function to create a single data set containing both X and Y

```
set.seed(1)
bd2 = data.frame(x,y)
reg1 = glm(y ~ x)
cv.glm(bd2, reg1)$delta
```

```
## [1] 7.288162 7.284744
```

```
reg2 = glm(y ~ poly(x, 2))
cv.glm(bd2, reg2)$delta
```

```
## [1] 0.9374236 0.9371789
```

```
reg3 = glm(y ~ poly(x, 3))
cv.glm(bd2, reg3)$delta
```

```
## [1] 0.9566218 0.9562538
```

```r
reg4 = glm(y ~ poly(x, 4))
cv.glm(bd2, reg4)$delta
```

```
## [1] 0.9539049 0.9534453
```

d) Repeat (c) using another random seed, and report your results. Are your results the same as what you got in (c)? Why?

```r
set.seed(2)
reg1 = glm(y ~ x)
cv.glm(bd2, reg1)$delta
```

```
## [1] 7.288162 7.284744
```

```r
reg2 = glm(y ~ poly(x, 2))
cv.glm(bd2, reg2)$delta
```

```
## [1] 0.9374236 0.9371789
```

```r
reg3 = glm(y ~ poly(x, 3))
cv.glm(bd2, reg3)$delta
```

```
## [1] 0.9566218 0.9562538
```

```r
reg4 = glm(y ~ poly(x, 4))
cv.glm(bd2, reg4)$delta
```

```
## [1] 0.9539049 0.9534453
```

Resposta

Os valores são exatamente os mesmos, pois não há aleatoriedade nas divisões do conjunto de treinamento/teste.

e) Which of the models in (c) had the smallest LOOCV error? Is this what you expected? Explain your answer.

O menor erro é do modelo ii, de: 0.93. O resultado é esperado, uma vez que esse modelo representa a formula verdadeira: $ii).Y = 0 + 1X + _2X^2 + \epsilon$

f) Comment on the statistical significance of the coefficient estimates that results from fitting each of the models in (c) using least squares. Do these results agree with the conclusions drawn based on the cross-validation results?

```r
summary(reg4)
```

```
##
## Call:
## glm(formula = y ~ poly(x, 4))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0550  -0.6212  -0.1567   0.5952   2.2267
##
## Coefficients:
##              Estimate Std. Error t value             Pr(>|t|)
## (Intercept)  -1.55002    0.09591 -16.162 < 0.0000000000000002 ***
## poly(x, 4)1   6.18883    0.95905   6.453         0.00000000459 ***
## poly(x, 4)2 -23.94830    0.95905 -24.971 < 0.0000000000000002 ***
## poly(x, 4)3   0.26411    0.95905   0.275                0.784
## poly(x, 4)4   1.25710    0.95905   1.311                0.193
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.9197797)
##
##     Null deviance: 700.852  on 99  degrees of freedom
## Residual deviance:  87.379  on 95  degrees of freedom
## AIC: 282.3
##
## Number of Fisher Scoring iterations: 2
```

Resposta

Apenas os modelos linear e quadrático apresentaram signficância estatística.

Exercício 09

We will now consider the Boston housing data set, from the MASS library.

a)Based on this data set, provide an estimate for the population mean of medv. Call this estimate $\hat{u}$.

```r
library(MASS)
```

```
## Warning: package 'MASS' was built under R version 3.5.3
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```r
bd = Boston
names(bd)
```

```
##  [1] "crim"    "zn"      "indus"   "chas"    "nox"     "rm"      "age"
##  [8] "dis"     "rad"     "tax"     "ptratio" "black"   "lstat"   "medv"
```

```r
u_hat = mean(bd$medv)
```

Resposta

$\hat{u} = 22.5$

b) Provide an estimate of the standard error of $\hat{u}$. Interpret this result. Hint: We can compute the standard error of the sample mean by dividing the sample standard deviation by the square root of the number of observations.

```r
u_err = sd(bd$medv) / sqrt(length(bd$medv))
```

Resposta

$\sigma^2 = 0.40$

c) Now estimate the standard error of $\hat{u}$ using the bootstrap. How does this compare to your answer from (b)?

```r
library(boot)
boot.fn = function(data, index) +
  return(mean(data[index]))
```

```r
bootstrap = boot(bd$medv, boot.fn, 1000)
bootstrap
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = bd$medv, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##     original      bias    std. error
## t1* 22.53281 -0.0196087   0.4196688
```

Resposta:

O erro calculado na letra b) foi de 0.40, enquanto utilizando o bootstrap foi de 0.41. Portanto, os valores são similares.

d) Based on your bootstrap estimate from (c), provide a 95 % confidence interval for the mean of medv. Compare it to the results obtained using t.test(Boston$medv).Hint: You can approximate a 95 % confidence interval using the formula $[\hat{u}2SE(\hat{u}), \hat{u} + 2SE(\hat{u})]$

Obs.: Para calcular o intervalo de confiança da letra (d), use a formula CI = [Mu - 1.96*SE(Mu) , Mu + 1.96*SE(Mu)], isto é, use 1,96 ao invés de 2.

```r
CI_up = u_hat + 1.96*u_err
CI_low =u_hat - 1.96*u_err
t.test(bd$medv)
```

```
##
##  One Sample t-test
##
## data:  bd$medv
## t = 55.111, df = 505, p-value < 0.00000000000000022
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  21.72953 23.33608
## sample estimates:
## mean of x
##  22.53281
```

Resposta

Com a formula: O intervalo de confiança está entre 21.7 e 23.3 T test: Intervalo de 21.7 e 23.3

Ou seja, os resultado são idênticos.

e) Based on this data set, provide an estimate, $\hat{u}_med$ **, for the median value of medv in the population.**

```r
median(bd$medv)
```

```
## [1] 21.2
```

Resposta:

Mediana = 21.2

f) We now would like to estimate the standard error of $\hat{u}_{med}$. Unfortunately, there is no simple formula for computing the standard error of the median. Instead, estimate the standard error of the median using the bootstrap. Comment on your findings.

```
boot_se = function(data, index) +
  return(median(data[index]))
boot(bd$medv, boot_se, R = 1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = bd$medv, statistic = boot_se, R = 1000)
##
##
## Bootstrap Statistics :
##     original    bias    std. error
## t1*     21.2 -0.04115   0.3796621
```

Resposta:

A mediana é de 21.2 e o erro padrão de 0.37

g) Based on this data set, provide an estimate for the tenth percentile of medv in Boston suburbs. Call this quantity $\hat{u}_{0.1}$. (You can use the quantile() function.)

```
quantile(bd$medv, 0.1 )
```

```
##   10%
## 12.75
```

h) Use the bootstrap to estimate the standard error of $\hat{u}_{0.1}$. Comment on your findings.

```
boot_10 = function(data, index) +
  return(quantile(data[index], c(0.1)))
boot(bd$medv, boot_10, R = 1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = bd$medv, statistic = boot_10, R = 1000)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*    12.75 0.03245   0.4997742
```

Resposta:

$\hat{u}_{0.1} = 12.75$ $\sigma\hat{u}_{0.1} = 0.530$