ISLR - Ch5 - Resampling Methods

Darshan Patel

1/30/2019

The following set of problems are from the applied exercises section in ISLR Chapter 4: Classification.

```
rm(list = ls())
library(MASS)
library(ISLR)
library(tidyverse)
```

```
## Warning: package 'tibble' was built under R version 3.4.4
## Warning: package 'tidyr' was built under R version 3.4.4
## Warning: package 'purrr' was built under R version 3.4.4
## Warning: package 'dplyr' was built under R version 3.4.4
```

```
library(gridExtra)
library(class)
library(boot)
```

Question 5: In Chapter 4, logistic regression was used to predict the probability of `default` using `income` and `balance` on the `Default` data set. Now estimate the test error of this logistic regression model using the validation set approach. Do not forget to set a random seed.

```
set.seed(24)
df = Default
```

(a) Fit a logistic regression model that uses `income` and `balance` to predict `default`.

```
model = glm(data = df, default~income+balance, family = binomial)
summary(model)
```

```
##
## Call:
## glm(formula = default ~ income + balance, family = binomial,
##     data = df)
##
## 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) -1.154e+01  4.348e-01 -26.545  < 2e-16 ***
## income       2.081e-05  4.985e-06   4.174 2.99e-05 ***
## balance      5.647e-03  2.274e-04  24.836  < 2e-16 ***
## ---
## 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 set error of this model. In order to do this, perform the following steps.

i. Split the sample set into a training set and a validation set.

```r
indices = sample(1:nrow(df), size = 0.7 * nrow(df))
train = df[indices,]
test = df[-indices,]
```

ii. Fit a multiple logistic regression model using only the training observations.

```r
model2 = glm(data = train, default~income+balance, family = binomial)
summary(model2)
```

```
##
## Call:
## glm(formula = default ~ income + balance, family = binomial,
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5238  -0.1463  -0.0577  -0.0212   3.7162
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.160e+01  5.089e-01 -22.801  < 2e-16 ***
## income       2.295e-05  5.769e-06   3.979 6.93e-05 ***
## balance      5.685e-03  2.665e-04  21.327  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2157.1  on 6999  degrees of freedom
## Residual deviance: 1145.7  on 6997  degrees of freedom
## AIC: 1151.7
##
## Number of Fisher Scoring iterations: 8
```

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.

```r
post_pred = predict(model2, test, type = "response")
pred = rep("No", nrow(test))
pred[post_pred > 0.5] = "Yes"
table(pred, test$default)
```

```
##
## pred    No  Yes
```

```
##    No  2904   56
##    Yes   13   27
```

iv. Compute the validation set error, which is the fraction of the observations in the validation set that are misclassified.

```
100 * (56 + 13) / nrow(test)
```

```
## [1] 2.3
```

(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.

```
error_list = c()
for(i in 1:3){
  set.seed(100 * i)
  indices = sample(1:nrow(df), size = 0.7*nrow(df))
  train = df[indices,]
  test = df[-indices,]
  temp = glm(data = train, default~income+balance, family = binomial)
  post_pred = predict(temp, test, type = "response")
  pred = rep("No", nrow(test))
  pred[post_pred > 0.5] = "Yes"
  error = 100 * (nrow(test) - sum(diag(table(pred, test$default)))) / nrow(test)
  error_list = c(error_list, error)
}
error_list
```

```
## [1] 2.533333 2.400000 3.100000
```

The errors found are relatively similar to one another as well as the one above.

(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(24)
indices = sample(1:nrow(df), size = 0.7*nrow(df))
train = df[indices,]
test = df[-indices,]
model3 = glm(data = train, default~income+balance+student, family = binomial)
post_pred = predict(model3, test, type = "response")
pred = rep("No", nrow(test))
pred[post_pred > 0.5] = "Yes"
(nrow(test) - sum(diag(table(pred, test$default)))) / nrow(test)
```

```
## [1] 0.02266667
```

Including the dummy variable for `student` does not lead to a big reduction in the test error rate.

Question 6: Continue considering the use of a logistic regression model to predict the probability of `default` using `income` and `balance` on the `Default` data set. In particular, 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.

```
set.seed(65)
indices = sample(1:nrow(df), size = 0.7*nrow(df))
train = df[indices,]
test = df[-indices,]
```

(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.

```
model = glm(data = train, default~income+balance, family = binomial)
summary(model)
```

```
##
## Call:
## glm(formula = default ~ income + balance, family = binomial,
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4525  -0.1432  -0.0575  -0.0212   3.7186
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.145e+01  5.173e-01 -22.137  < 2e-16 ***
## income       1.873e-05  5.945e-06   3.151  0.00163 **
## balance      5.632e-03  2.699e-04  20.868  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2043.8  on 6999  degrees of freedom
## Residual deviance: 1099.1  on 6997  degrees of freedom
## AIC: 1105.1
##
## Number of Fisher Scoring iterations: 8
```

The estimated standard error for the coefficient estimates associated with `income` and `balance` is $5.956e-06$ and $2.699e-04$ respectively.

(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 logistic regression model.

```
boot.fn = function(df, index){
  train = df[index,]
  temp = glm(data = train, default~income+balance, family = binomial)
  summary(temp)$coefficients
}
```

(c) Use the `boot()` function together with the `boot.fn()` function to estimate the standard errors of the logistic regression coefficients for `income` and `balance`.

```
boot(df, boot.fn, 10)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = df, statistic = boot.fn, R = 10)
##
##
## Bootstrap Statistics :
##            original          bias      std. error
## t1*   -1.154047e+01   7.055815e-02  5.088194e-01
## t2*    2.080898e-05  -1.036082e-06  4.704386e-06
## t3*    5.647103e-03  -3.736877e-05  2.977706e-04
## t4*    4.347564e-01   2.453393e-03  2.461032e-02
## t5*    4.985167e-06   6.604653e-08  1.201595e-07
## t6*    2.273731e-04   1.272331e-06  1.353116e-05
## t7*   -2.654468e+01   2.932284e-01  4.319237e-01
## t8*    4.174178e+00  -2.631244e-01  8.967841e-01
## t9*    2.483628e+01  -2.912184e-01  4.148110e-01
## t10*  2.958355e-155  7.196098e-147 1.649395e-146
## t11*   2.990638e-05   1.727216e-03  3.531948e-03
## t12*  3.638120e-136  3.152776e-125 9.969887e-125
```

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

All estimated standard errors obtained using either the `glm()` function or the bootstrap function were calculated to be within a small ballpark of 0.

Question 7: The `cv.glm()` function can be used 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. Take this different approach in order to compute the LOOCV error for a simple regression model on the `Weekly` data set. Recall that the LOOCV error is given by

$$CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{Err}_i$$

where $\mathbf{Err}_i = I(y_i \neq \hat{y}_i)$.

(a) Fit a logistic regression model that predicts `Direction` using `Lag1` and `Lag2`.

```
df = Weekly
model = glm(data = df, Direction~Lag1+Lag2, family = binomial)
predictions = predict(model, df)
pred = rep("Down", nrow(df))
pred[predictions > 0.5] = "Up"
table(pred, df$Direction)
```

```
##
```

```
## pred   Down  Up
##   Down  470 576
##   Up     14  29
```

(b) Fit a logistic regression model that predicts `Direction` using `Lag1` and `Lag2` *using all but the first observation.*

```r
train = df[-1,]
model2 = glm(data = train, Direction~Lag1+Lag2, family = binomial)
```

(c) Use the model from (b) to predict the direction of the first observation. Do this by predicting that the first observation will go up if $P($`direction="Up"|Lag1,Lag2`$) > 0.5$. Was this observation correctly classified?

```r
test = df[1,]
prediction = ifelse(predict(model2, test, type = "response") > 0.5, "Yes", "No")
prediction == test$Direction
```

```
##     1
## FALSE
```

```r
prediction
```

```
##     1
## "Yes"
```

```r
test$Direction
```

```
## [1] Down
## Levels: Down Up
```

This observation was not 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 step:

(e) 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 going up for the ith observation.

(iii) Use the posterior probability for the ith 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 ith observation. If an error was made, then indicate this as a 1, and otherwise indicate it as a 0.

```r
error_list = c()
for(i in 1:nrow(df)){
  train = df[-i,]
  test = df[i,]
  temp = glm(data = train, Direction~Lag1+Lag2, family = binomial)
  prediction = ifelse(predict(temp, test, type = "response") > 0.5, "Up", "Down")
  error = ifelse(prediction == test$Direction, 0, 1)
  error_list = c(error_list, error)
}
```

(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.

```r
mean(error_list)
```

```
## [1] 0.4499541
```

The average test error made when performing LOOCV to predict `Direction` using `Lag1` and `Lag2` is 44.99%. This is a moderately high test error. Look for improvements using other variables or interactions.

Question 8: Perform cross-validation on a simulated data set.

(a) Generate a simulated data set as follow:

```
set.seed(8)
x = rnorm(100)
eps = rnorm(100)
y = x-2*x^2+eps
```

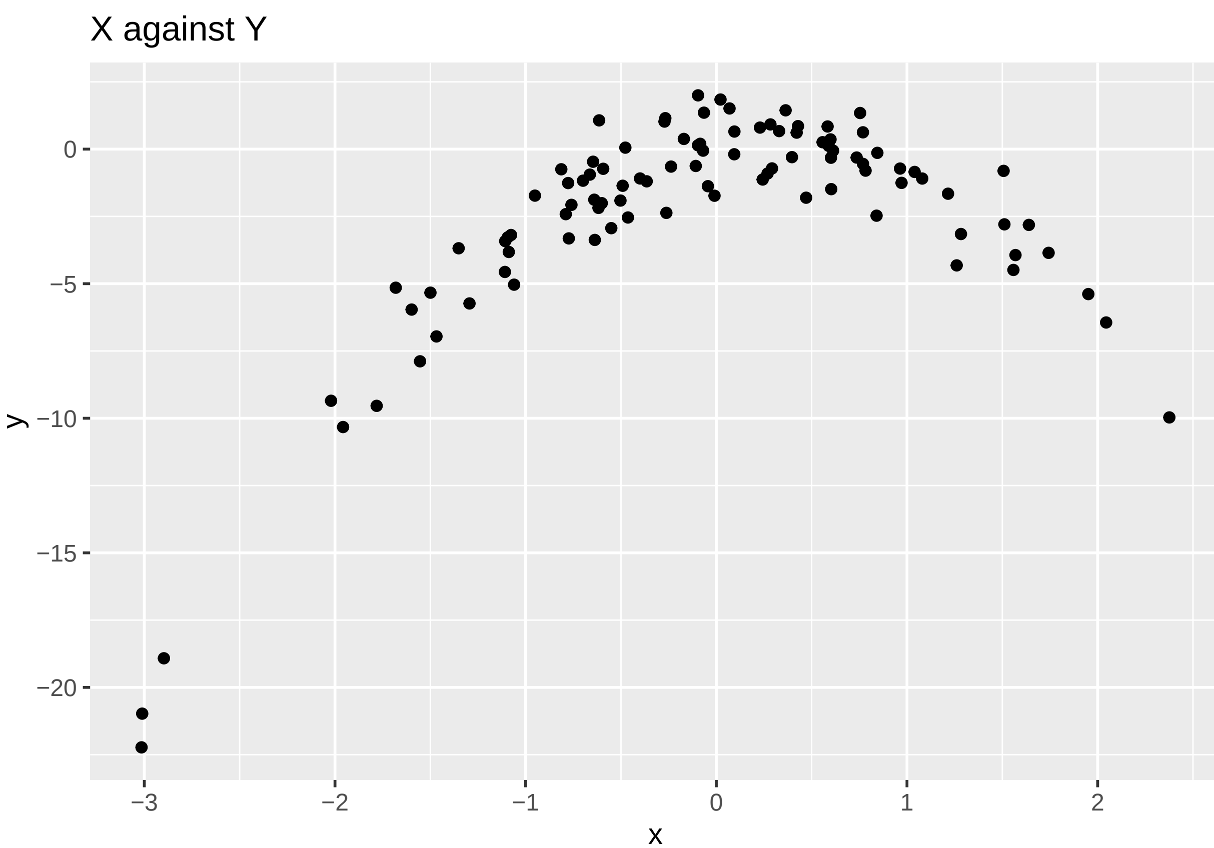
In this data set, what is n and what is p? Write out the model used to generate the data in equation form.

In this data set, $n = 100$ and $p = 2$. The model used to generate the data is

$$Y = X - 2X^2 + \varepsilon$$

(b) Create a scatterplot of X against Y. Comment some findings.

```
df = data.frame(x = x, y = y)
ggplot(df, aes(x = x, y = y)) + geom_point() + ggtitle("X against Y")
```



The data has an inverse parabolic shape. The `x`s and `y`s are closely related.

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

i. $Y = \beta_0 + \beta_1 X + \varepsilon$
ii. $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \varepsilon$

iii. $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \varepsilon$
iv. $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \beta_4 X^4 + \varepsilon$

```r
set.seed(100)
x = rnorm(100)
eps = rnorm(100)
y = x-2*x^2+eps
df = data.frame(y = y, x = x, x2 = x^2, x3 = x^3, x4 = x^4)

model1a = glm(data = df, y~x)
model2a = glm(data = df, y~x+x2)
model3a = glm(data = df, y~x+x2+x3)
model4a = glm(data = df, y~x+x2+x3+x4)

cv.glm(df, model1a)$delta
```

```
## [1] 9.060636 9.056661
```

```r
cv.glm(df, model2a)$delta
```

```
## [1] 0.6511909 0.6509495
```

```r
cv.glm(df, model3a)$delta
```

```
## [1] 0.6665339 0.6661944
```

```r
cv.glm(df, model4a)$delta
```

```
## [1] 0.6671261 0.6667107
```

(d) Repeat (c) using another random seed and report results. Are the results the same as ones in (c)? Why?

```r
set.seed(202)

model1b = glm(data = df, y~x)
model2b = glm(data = df, y~x+x2)
model3b = glm(data = df, y~x+x2+x3)
model4b = glm(data = df, y~x+x2+x3+x4)

cv.glm(df, model1b)$delta
```

```
## [1] 9.060636 9.056661
```

```r
cv.glm(df, model2b)$delta
```

```
## [1] 0.6511909 0.6509495
```

```r
cv.glm(df, model3b)$delta
```

```
## [1] 0.6665339 0.6661944
```

```r
cv.glm(df, model4b)$delta
```

```
## [1] 0.6671261 0.6667107
```

The results are exactly the same because setting the seed does nothing to the process of LOOCV. In LOOCV, a model is created n times where exactly one observation is left out and tested on. There is no sense of randomness here.

(e) Which of the models in (c) had the smallest LOOCV error? Was this expected? Explain.

The model with the quadratic term had the smallest LOOCV error. This was expected because Y was calculated using a quadratic term of X.

(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 based on the cross-validation results?

```
summary(model1a)$coefficients[,4]
```

```
##  (Intercept)            x
## 2.522871e-10 5.992723e-02
```

```
summary(model2a)$coefficients[,4]
```

```
##  (Intercept)            x           x2
## 5.087084e-01 2.693404e-19 6.193006e-57
```

```
summary(model3a)$coefficients[,4]
```

```
##  (Intercept)            x           x2           x3
## 5.378034e-01 5.580099e-07 2.066472e-55 5.596999e-01
```

```
summary(model4a)$coefficients[,4]
```

```
##  (Intercept)            x           x2           x3           x4
## 8.582117e-01 2.315416e-07 8.125543e-23 9.436831e-01 1.606201e-01
```

The coefficient estimates in the linear and quadratic models were statistically significant whereas it loses significance in the tertiary and fourth degree polynomial models.

Question 9: 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 $\hat{\mu}$.

```
df = Boston
mu_hat = mean(df$medv)
mu_hat
```

```
## [1] 22.53281
```

(b) Provide an estimate of the standard error of $\hat{\mu}$. Interpret this result. *Hint: Compute the standard error of the sample mean by dividing the sample standard deviation by the square root of the number of observations.*

```
se_hat = sd(df$medv) / sqrt(nrow(df))
se_hat
```

```
## [1] 0.4088611
```

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

```
set.seed(99)

boot_mean = function(df, index){
  mean(df[index])
}

boot(df$medv, boot_mean, 10)
```

```
##
```

```
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = df$medv, statistic = boot_mean, R = 10)
##
##
## Bootstrap Statistics :
##     original       bias    std. error
## t1* 22.53281 -0.09081028   0.4785729
```

Using the bootstrap, the standard error of $\hat{\mu}$ is slightly higher than by using the direct calculation.

(d) Based on the 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: Approximate a 95% confidence interval using the formula* $[\hat{\mu} - 2SE(\hat{\mu}), \hat{\mu} + 2SE(\hat{\mu})]$.

```
err = 2*se_hat * 0.4785729
paste("(", mu_hat - err, ", ", mu_hat + err, ")", sep = "")
```

```
## [1] "(22.1414665940002, 22.9241460542211)"
```

```
t.test(df$medv)$conf.int
```

```
## [1] 21.72953 23.33608
## attr(,"conf.level")
## [1] 0.95
```

The confidence interval created using the formulas is narrower than the one created using `t.test`.

(e) Based on this data set, provide an estimate, $\hat{\mu}_{\text{med}}$, for the median value of `medv` in the population.

```
mu_hat = median(df$medv)
mu_hat
```

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

(f) Now estimate the standard error of $\hat{\mu}_{\text{med}}$. 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 the findings.

```
set.seed(99)

boot_med = function(df, index){
  median(df[index])
}

boot(df$medv, boot_med, 10)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = df$medv, statistic = boot_med, R = 10)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*     21.2   -0.06   0.4520324
```

The standard error in the median is smaller than for the mean.

(g) Based on this data set, provide an estimate for the tenth percentile of `medv` in Boston suburbs. Call this quantity $\hat{\mu}_{0.1}$. *Hint: Using the* ***`quantile()`*** *function.*

```
mu_hat_tenth = quantile(df$medv, 0.1)
mu_hat_tenth
```

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

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

```
set.seed(99)

boot_tenth_quantile = function(df, index){
  quantile(df[index], 0.1)
}

boot(df$medv, boot_tenth_quantile, 10)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = df$medv, statistic = boot_tenth_quantile, R = 10)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*    12.75   -0.02   0.5774465
```

The standard error associated with the estimate of the 10% percentile is larger than the one for the median, or 50% percentile.

All of the practice applied exercises in this document are taken from "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani.