ISL - Chapter 5 Lab Tutorials
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

Main Contents:

1. Cross-Validation
2. The Bootstrap

5.3. Lab: Cross-Validation and the Bootstrap

5.3.1. The Validation Set Approach

```r
library(ISLR)
attach(Auto)
set.seed(1)
train <- sample(392, 196)
```

Regression Models

```r
# Linear Regression model
lm.fit <- lm(mpg ~ horsepower, data = Auto, subset = train)
pol.1.mse <- round(mean((mpg - predict(lm.fit, Auto))[-train]^2),2)
# Polynomial regression: power = 2
lm.fit2 <- lm(mpg ~ poly(horsepower, 2), data = Auto, subset = train)
pol.2.mse <- round(mean((mpg - predict(lm.fit2, Auto))[-train]^2),2)
# Polynomial regression: power = 3
lm.fit3 <- lm(mpg ~ poly(horsepower, 3), data = Auto, subset = train)
pol.3.mse <- round(mean((mpg - predict(lm.fit3, Auto))[-train]^2),2)
models <- c('Linear Regression', 'Polynomial Regression: power 2', 'Polynomial Regression: power 3')
mses <- c(pol.1.mse, pol.2.mse, pol.3.mse)
data.frame(Models = models, MSE = mses)
```

```
##                           Models   MSE
## 1              Linear Regression 26.14
## 2 Polynomial Regression: power 2 19.82
## 3 Polynomial Regression: power 3 19.78
```

5.3.2. Leave-One-Out Cross-Validation

To use `cv.glm()` from package `boot`.

```
library(boot)
# glm() for Linear Regression model instead of lm()
glm.fit <- glm(mpg ~ horsepower, data = Auto)
round(coef(glm.fit),2)
```

```
## (Intercept)  horsepower
##       39.94       -0.16
```

```
# Cross-validation
cv.err <- cv.glm(Auto, glm.fit)
print(paste0('Cross-Validation error: ', round(cv.err$delta[1],4), ', ', round(cv.err$delta[2],4)))
```

```
## [1] "Cross-Validation error: 24.2315, 24.2311"
```

`cv.err` for Polynomial fit of degree 1,2,3,4,5

```
cv.error <- rep(0,5)
for (i in 1:5) {
  glm.fit <- glm(mpg ~ poly(horsepower, i), data = Auto)
  cv.error[i] <- cv.glm(Auto, glm.fit)$delta[1]
}
degrees <- 1:5
data.frame(Polynomial_Regression_Degree = degrees, MSE = round(cv.error,2))
```

```
##   Polynomial_Regression_Degree   MSE
## 1                            1 24.23
## 2                            2 19.25
## 3                            3 19.33
## 4                            4 19.42
## 5                            5 19.03
```

Comment: sharp drop in MSE from Linear to Quadratic but not much there after.

5.3.3. k-Fold Cross-Validation

```r
set.seed(17)
cv.error.10 <- rep(0, 10)
for (i in 1:10) {
  glm.fit <- glm(mpg ~ poly(horsepower, i), data = Auto)
  # k = 10
  cv.error.10[i] <- round(cv.glm(Auto, glm.fit, K = 10)$delta[1],2)
}
t(data.frame(Polynomial_Degree = 1:10, MSE = cv.error.10))
```

```
##                    [,1]  [,2]  [,3]  [,4]  [,5]  [,6] [,7]  [,8]  [,9]
## Polynomial_Degree  1.00  2.00  3.00  4.00  5.00  6.00  7.0  8.00  9.00
## MSE               24.21 19.19 19.31 19.34 18.88 19.02 18.9 19.71 18.95
##                   [,10]
## Polynomial_Degree  10.0
## MSE                19.5
```

5.3.4. The Bootstrap

To estimate the accuracy of a test-statistic, for example:

$$t = \frac{var(Y) - var(X)}{var(X) + var(Y) - 2\ cov(X, Y)}$$

```
# fn to compute test statistic
alpha.fn <- function(data, index) {
  X <- data$X[index]
  Y <- data$Y[index]
  test.stat <- (var(Y) - cov(X,Y)) / (var(X) + var(Y) - 2*cov(X,Y))
  return(test.stat)
}
p <- round(alpha.fn(Portfolio, 1:100),4)
```

p-value after *Bootstrapping* is 0.5758.

```
# Bootrapping
set.seed(1)
p <- round(alpha.fn( Portfolio, sample(100, 100, replace = T)), 4)
```

Alternatively, instead of `1:100`, we can use function `sample()`, giving a new bootstrapped p-value of 0.5964.

Bootrapping: `boot()` from package `boot`

```
# boot() for bootstrapping, from 'boot' library
boot(Portfolio, alpha.fn, R = 1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Portfolio, statistic = alpha.fn, R = 1000)
##
##
## Bootstrap Statistics :
##      original        bias    std. error
## t1* 0.5758321 -7.315422e-05  0.08861826
```

Estimating the Accuracy of a Linear Regression Model

```
boot.fn <- function(data, index) {
  return(coef(lm(mpg ~ horsepower, data = data, subset = index)))
}
round(boot.fn(Auto, 1:392),2)
```

```
## (Intercept)  horsepower
##       39.94       -0.16
```

Test-statistic $= \mu$

```r
set.seed(1)
round(boot.fn(Auto, sample(392, 392, replace = T)),2)
```

```
## (Intercept)  horsepower
##       38.74       -0.15
```

Test-statistic $= SE$

```r
boot(Auto, boot.fn, 1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Auto, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##       original        bias    std. error
## t1* 39.9358610  0.0296667441 0.860440524
## t2* -0.1578447 -0.0003113047 0.007411218
```

```r
# Compare against Linear model
summary(lm(mpg ~ horsepower, data = Auto))$coef
```

```
##               Estimate  Std. Error   t value      Pr(>|t|)
## (Intercept) 39.9358610 0.717498656  55.65984 1.220362e-187
## horsepower  -0.1578447 0.006445501 -24.48914  7.031989e-81
```

```r
boot.fn <- function(data, index) {
  coefficients(lm(mpg ~ horsepower + I(horsepower^2), data = data, subset = index))
}
set.seed(1)
boot(Auto, boot.fn, 1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Auto, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##         original        bias     std. error
## t1* 56.900099702  6.098115e-03 2.0944855842
## t2* -0.466189630 -1.777108e-04 0.0334123802
## t3*  0.001230536  1.324315e-06 0.0001208339
```

```r
summary(lm(mpg ~ horsepower + I(horsepower^2), data = Auto))$coef
```

```
##                      Estimate   Std. Error   t value      Pr(>|t|)
## (Intercept)     56.900099702 1.8004268063  31.60367 1.740911e-109
## horsepower      -0.466189630 0.0311246171 -14.97816  2.289429e-40
## I(horsepower^2)  0.001230536 0.0001220759  10.08009  2.196340e-21
```