IMT 573: Module 9 Optional Material

Resampling Methods

Jenny Skytta

Collaborators: List collaborators here.

Objectives

In this optional "lab", we explore resampling methods using the `Auto` dataset. This material is taken from James et al. (2013).

```
# Load standard libraries
library(tidyverse)
```

The Validation Set Approach

Let's start by using the `sample()` function to split (i.e. define) the set of observations into two sets, by selecting a random subset of 196 observations out of the original 392 observations. Note the training set here is 1/2 of the original dataset.

```
# Load the ISLR library
library(ISLR) #Intro to Statistical Learning

# Set a random seed so that results can be reproduced
set.seed(1)

# Define the training set
train <- sample(392,196)
```

After defining the training data we can use it to fit a linear regression model. The `subset` option in `lm()` is helpful here.

We use the `predict()` function to estimate the response for all observations. The `mean()` function is used to calculate the MSE of the validation set. Note that the `-train` index below selects only the observations that are not in the training set.

```
# Fit a linear regression model using only the training set
lm.fit <- lm(mpg ~ horsepower, # y ~ x1
             data = Auto, # dataset
             subset = train) # pulling the train data
# Calculate the MSE of the validation set
mse <- mean((Auto$mpg-predict(lm.fit, Auto))[-train]^2) # formula
mse # mean squared error
```

```
## [1] 23.26601
```

Notice if we choose a different training set the results will be different.

```r
# New random seed
set.seed(2)

# Define new training set
train.new <- sample(392, 196)

# Fit a linear regression model using only the training set
lm.fit.new <- lm(mpg ~ horsepower, data = Auto,
                 subset = train.new)

# Calculate the MSE of the validation set
mse.new <- mean((Auto$mpg-predict(lm.fit.new, Auto))[-train.new]^2)
mse.new # calling the variable
```

```
## [1] 25.72651
```

```r
mse==mse.new # boolean comparison
```

```
## [1] FALSE
```

LOOCV

```r
# Fit the model using the glm function
glm.fit <- glm(mpg ~ horsepower, data = Auto) # y ~ x1 with glm
summary(glm.fit) #summary of the outputs
```

```
##
## Call:
## glm(formula = mpg ~ horsepower, data = Auto)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -13.5710   -3.2592   -0.3435    2.7630   16.9240
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 39.935861   0.717499   55.66   <2e-16 ***
## horsepower  -0.157845   0.006446  -24.49   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 24.06645)
##
##     Null deviance: 23819.0  on 391  degrees of freedom
## Residual deviance:  9385.9  on 390  degrees of freedom
## AIC: 2363.3
##
## Number of Fisher Scoring iterations: 2
```

```r
# default behavior is the same as the liner model
lm.fit.all <- lm(mpg ~ horsepower, data = Auto) # y ~ x1 with lm
summary(lm.fit.all) # summary of the outputs
```

```
##
## Call:
## lm(formula = mpg ~ horsepower, data = Auto)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.5710  -3.2592  -0.3435   2.7630  16.9240
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 39.935861   0.717499   55.66   <2e-16 ***
## horsepower  -0.157845   0.006446  -24.49   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.906 on 390 degrees of freedom
## Multiple R-squared:  0.6059, Adjusted R-squared:  0.6049
## F-statistic: 599.7 on 1 and 390 DF,  p-value: < 2.2e-16
```

```
# LOOCV can be done with the cv.glm function in the boot package
# Leave one out cross validation
library(boot)
# ?cv.glm cross validation for generalized linear model

# Compute LOOCV estimate of the test MSE
cv.err <- cv.glm(Auto, glm.fit)

# Resulting object contains many different things
names(cv.err)
```

```
## [1] "call"  "K"     "delta" "seed"
```

```
# Two num. in the delta vector contain cv results
cv.err$delta
```

```
## [1] 24.23151 24.23114
```

K-Fold CV

```
# K-Fold CV can be done as well

# Compute k-fold CV estimate of the test MSE
cv.err.k10 <- cv.glm(Auto, glm.fit, K=10)

# Two num. in the delta vector contain cv results
cv.err.k10$delta
```

```
## [1] 24.18370 24.17104
```

The Bootstrap

The bootstrap approach can be applied in almost all situations!

You need the following steps:

- Create a function that computes the statistic of interest.
- Use the `boot()` function, which is part of the `boot` library, to perform the bootstrap by repeatedly sampling observations from the data set **with replacement**.

We use the `Portfolio` dataset, described in the reading.

```
# A function that takes as input (X,Y)
# as well as a vector indicating which
# observations should be used to estimate
# alpha

# writing a function that inputs data and index
alpha.fn <- function(data, index) {
  X <- data$X[index] # convert to x
  Y <- data$Y[index] # convert to y
  res <- (var(Y) - cov(X,Y))/(var(X) + var(Y) -2*cov(X,Y))
  return (res) # run equation and output variable as res
}

# Test the function
alpha.fn(Portfolio, 1:100)
```

```
## [1] 0.5758321
```

Now we use the `boot()` function to produce $R = 1,000$ bootstrap estimates for α.

```
# Boostrap estimate of alpha
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 0.001198111  0.09293144
```

The bootstrap approach can be used to assess the variability of the coefficient estimates and predictions from a statistical learning method.

Below we assess the variability of the estimates for the intercept and slope terms for the linear regression model that uses `horsepower` to predict `mpg` in the `Auto` data set.

```
# First create the function for the statistic of interest
boot.fn  <-  function(data, index){
  res <- coef(lm(mpg ~ horsepower,
                 data = data,
                 subset = index))
  return(res)
}
```

```r
# Boostrap estimated of the standard error
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  0.0900211337 0.823538535
## t2* -0.1578447 -0.0007898716 0.007098094
```

```r
# Recall
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
```

Interestingly, these are somewhat different from the estimates obtained using the bootstrap.