Resampling Methods

Contents

1 Getting Set Up **1**

1.1 Setting chunk options 1

1.2 Installing Packages 1

2 The Validation Set Approach **1**

3 Leave Out One Cross Validation **2**

3.1 K-Fold Cross Validation 3

4 The Bootstrap **3**

4.1 Estimating the Accuracy of a Linear Regression Model 3

1 Getting Set Up

1.1 Setting chunk options

```
knitr::opts_chunk$set(warning=FALSE, message=FALSE)
knitr::purl("resampling-methods.Rmd")
```

```
##
##
## processing file: resampling-methods.Rmd

## output file: resampling-methods.R
```

1.2 Installing Packages

```
install.packages('ISLR')
install.packages('boot')
```

2 The Validation Set Approach

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

lm.fit=lm(mpg~horsepower, data=Auto, subset=train)
```

We now use the predict() function to estimate the response for all 392 observations, and we use the mean() function to calculate the MSE of the 196 observations in the validation set. Note that the -train index below selects only the observations that are not in the training set.

```
mean((mpg-predict(lm.fit ,Auto))[-train ]^2)
```

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

We can use the poly() function to estimate the test error for the quadratic and cubic regressions.

```
lm.fit2 = lm(mpg~poly(horsepower, 2), data=Auto, subset=train)
mean((mpg-predict(lm.fit2, Auto))[- train]^2)
```

```
## [1] 18.71646
```

```
lm.fit3 = lm(mpg~poly(horsepower, 3), data=Auto, subset=train)
mean((mpg-predict(lm.fit3, Auto))[- train]^2)
```

```
## [1] 18.79401
```

3 Leave Out One Cross Validation

```
library(boot)
glm.fit=glm(mpg~horsepower, data=Auto)
cv.err=cv.glm(Auto, glm.fit)
cv.err$delta
```

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

The cv.glm() function produces a list with several components. The two numbers in the delta vector contain the cross-validation results. We can repeat this procedure for increasingly complex polynomial fits.

```
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]
}
cv.error
```

```
## [1] 24.23151 19.24821 19.33498 19.42443 19.03321
```

3.1 K-Fold Cross Validation

The cv.glm() function can also be used to implement k-fold CV. Below we use k = 10, a common choice for k, on the Auto data set. We once again set a random seed and initialize a vector in which we will store the CV errors corresponding to the polynomial fits of orders one to ten.

```
set.seed(17)
cv.error.10 = rep(0,10)
for (i in 1:10){
glm.fit = glm(mpg~poly(horsepower, i), data=Auto)
cv.error.10[i] = cv.glm(Auto, glm.fit, K=10)$delta[1]
}
cv.error.10
```

```
##  [1] 24.27207 19.26909 19.34805 19.29496 19.03198 18.89781 19.12061 19.14666
##  [9] 18.87013 20.95520
```

4 The Bootstrap

4.1 Estimating the Accuracy of a Linear Regression Model

Here we use the bootstrap approach in order to assess the variability of the estimates for 0 and 1, the intercept and slope terms for the linear regression model that uses horsepower to predict mpg in the Auto data set.

We first create a simple function, boot.fn(), which takes in the Auto data set as well as a set of indices for the observations, and returns the intercept and slope estimates for the linear regression model. We then apply this function to the full set of 392 observations in order to compute the estimates of 0 and 1 on the entire data set.

```
boot.fn = function(data, index)
return(coef(lm(mpg~horsepower, data=data, subset=index)))
boot.fn(Auto, 1:392)
```

```
## (Intercept)  horsepower
##  39.9358610  -0.1578447
```

The boot.fn() function can also be used in order to create bootstrap estimates for the intercept and slope terms by randomly sampling from among the observations with replacement. Here we give two examples.

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

```
## (Intercept)  horsepower
##  40.3404517  -0.1634868
```

```
boot.fn(Auto, sample(392,392,replace=T))
```

```
## (Intercept)  horsepower
##  40.1186906  -0.1577063
```

Next, we use the boot() function to compute the standard errors of 1,000 bootstrap estimates for the intercept and slope terms.

```
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  0.0544513229 0.841289790
## t2* -0.1578447 -0.0006170901 0.007343073
```

Lets compare the standard erros with those obtained from the summary function

```
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. Does this indicate a problem with the bootstrap? In fact, it suggests the opposite. Bootstrap makes fewer assumptions about the distribution of variance among different components in the model. Below we compute the bootstrap standard error estimates and the stan dard linear regression estimates that result from fitting the quadratic model to the data.

```
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  3.511640e-02 2.0300222526
## t2* -0.466189630 -7.080834e-04 0.0324241984
## t3*  0.001230536  2.840324e-06 0.0001172164
```

```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
```