Cross-Validation and the Bootstrap

Dr Liwan Liyanage - Western Sydney University

2 July 2018

Cross-Validation and the Bootstrap

- The Validation Set Approach

We explore the use of the validation set approach in order to estimate the test error rates that result from fitting various linear models on the Auto data set.

Before we begin, we use the set.seed() function in order to set a seed for R's random number generator, so one can obtain precisely the same results as those shown below.

It is generally a good idea to set a random seed when performing an analysis such as cross-validation that contains an element of randomness, so that the results obtained can be reproduced precisely at a later time.

We begin by using the sample() function to select a Training Data Set

First split the set of observations into two halves, by selecting a random subset of 196 observations out of the original 392 observations.

We refer to these observations as the training data set.

```
library (ISLR)
attach(Auto)
dim(Auto)
```

```
## [1] 392   9
```

```
Auto=na.omit(Auto)
dim(Auto)
```

```
## [1] 392   9
```

```
set.seed (1)
```

We then use the subset option in lm() to fit a linear regression using only the observations corresponding to the training set.

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

```
##
## Call:
## lm(formula = mpg ~ horsepower, data = Auto, subset = tra
##
## Coefficients:
## (Intercept)   horsepower
##     40.3404      -0.1617
```

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.

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

```
## [1] 26.14142
```

Note that the -train index below selects only the observations that are not in the training set.

Therefore, the estimated test MSE for the linear regression fit is 26.14.

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

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

```
## [1] 19.82259
```

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

```
## [1] 19.78252
```

These error rates are 19.82 and 19.78, respectively.

If we choose a different training set instead, then we will obtain somewhat different errors on the validation set.

```
set.seed(2)
train=sample(392 ,196)
lm.fit=lm(mpg~horsepower,subset=train)
```

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

```
## [1] 23.29559
```

```r
mean((mpg-predict(lm.fit2,Auto))[-train ]^2)
```

```
## [1] 18.8792
```

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

```
## [1] 19.2574
```

Results

Using this split of the observations into a training set and a validation set, we find that the validation set error rates for the models with linear, quadratic, and cubic terms are 23.30, 18.90, and 19.26, respectively. These results are consistent with our previous findings: a model that predicts mpg using a quadratic function of horsepower performs better than a model that involves only a linear function of horsepower, and there is little evidence in favor of a model that uses a cubic function of horsepower.

Leave-One-Out Cross-Validation

The LOOCV estimate can be automatically computed for any generalized linear model using the glm() and cv.glm() functions.

We used the glm() function to perform logistic regression by passing in the family="binomial" argument.

But if we use glm() to fit a model without passing in the family argument, then it performs linear regression, just like the lm() function.

```
glm.fit=glm(mpg~horsepower,data=Auto)
coef(glm.fit)
```

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

Alternatively

```r
lm.fit =lm(mpg~horsepower ,data=Auto)
coef(lm.fit)
```

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

Note that they yield identical linear regression models.

cv.glm() Cross Valitation function

we will perform linear regression using the glm() function rather than the lm() function because the former can be used together with cv.glm().

The cv.glm() function is part of the boot library.

```
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. In this case the numbers are identical (up to two decimal places) and correspond to the LOOCV statistic. Our cross-validation estimate for the test error is approximately 24.23.

We can repeat this procedure for increasingly complex polynomial fits.

To automate the process, we use the for() function to initiate a for loop which iteratively fits polynomial regressions for polynomials of order i = 1 to i = 5, computes the associated cross-validation error, and stores it in the ith element of the vector cv.error. We begin by initializing the vector. This command will likely take a couple of minutes to run.

```
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
```

We see a sharp drop in the estimated test MSE between the linear

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] 24.20520 19.18924 19.30662 19.33799 18.87911 19.02
##  [8] 19.71201 18.95140 19.50196
```

Notice that the computation time is much shorter than that of LOOCV.

We saw that the two numbers associated with delta are essentially the same when LOOCV is performed.

When we instead perform k-fold CV, then the two numbers associated with delta differ slightly.

The first is the standard k-fold CV estimate.

The second is a biascorrected version.

On this data set, the two estimates are very similar to each other.

The Bootstrap

We illustrate the use of the bootstrap on two example estimating the Accuracy of a Statistic of Interest

Performing a bootstrap analysis in R entails only two steps.

-First, we must create a function that computes the statistic of interest.

-Second, we 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.

Example

To illustrate the use of the bootstrap using the Portfolio data set in the ISLR package to illustrate boot() function

Suppose that we wish to invest a fixed sum of money in two financial assets that yield returns of X and Y , respectively, where X and Y are random quantities.

We will invest a fraction α of our money in X, and will invest the remaining 1 - α in Y .

Since there is variability associated with the returns on these two assets, we wish to choose α to minimize the total risk, or variance, of our investment.

In other words, we want to minimize Var(αX +(1 -α)Y).

One can show that the value that minimizes the risk is given by

$\alpha = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}}$

Data Explore

```
library(ISLR)
attach(Portfolio)
View(Portfolio)
dim(Portfolio)
```

```
## [1] 100   2
```

```
head(Portfolio)
```

```
##            X          Y
## 1 -0.8952509 -0.2349235
## 2 -1.5624543 -0.8851760
## 3 -0.4170899  0.2718880
## 4  1.0443557 -0.7341975
## 5 -0.3155684  0.8419834
## 6 -1.7371238 -2.0371910
```

Estimating alpha.fn()

We must first create a function, alpha.fn(), which takes as input the (X, Y) data as well as a vector indicating which observations should be used to estimate α. The function then outputs the estimate for α based on the selected observations.

```
alpha.fn=function(data,index){
X=data$X[index]
Y=data$Y[index]
return ((var(Y)-cov (X,Y))/(var(X)+var(Y) -2*cov(X,Y)))}
```

The following command tells R to estimate α using all 100 observations.

```
alpha.fn(Portfolio,1:100)
```

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

Sampling with replacement to generate a new bootstrap data set

The next command uses the sample() function to randomly select 100 observations from the range 1 to 100, with replacement. This is equivalent to constructing a new bootstrap data set and recomputing $\hat{\alpha}$ based on the new data set.

```
set.seed (1)
alpha.fn(Portfolio,sample(100,100,replace =T))
```

```
## [1] 0.5963833
```

We can implement a bootstrap analysis by performing this command many times, recording all of the corresponding estimates for α, and computing the resulting standard deviation. However, the boot() function automates this approach.

Below we produce R = 1,000 bootstrap estimates for α.

```r
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
```

The final output shows that using the original data, $\hat{\alpha} = 0.5758$, and that the bootstrap estimate for SE($\hat{\alpha}$) is 0.0886.

Estimating the Accuracy of a Linear Regression Model

The bootstrap approach can be used to assess the variability of the coefficient estimates and predictions from a statistical learning method. Here we use the bootstrap approach in order to assess the variability of the estimates for β_0 and β_1, the intercept and slope terms for the linear regression model that uses horsepower to predict mpg in the Auto data set. We will compare the estimates obtained using the bootstrap to those obtained using the formulas for $SE(\beta_0)$ and $SE(\beta_1)$ described earlier.

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 β_0 and β_1 on the entire data set using the usual linear regression coefficient estimate formulas

```
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
##  38.7387134  -0.1481952
```

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

```
## (Intercept)  horsepower
##  40.0383086  -0.1596104
```

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

```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  0.02972191 0.860007896
## t2* -0.1578447 -0.00030823 0.007404467
```

As discussed in Linear Regression These can be obtained using the summary() function.

Standard formulas can be used to compute the standard errors for the regression coefficients in a 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-1
## horsepower  -0.1578447 0.006445501 -24.48914  7.031989e-
```

The standard error estimates for $\hat{\beta}_0$ and $\hat{\beta}_1$ obtained using the formulas in linear regression are 0.717 for the intercept and 0.0064 for the slope.

Discussion:

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. In linear regression we rely on certain assumptions. For example, they depend on the unknown parameter σ^2, the noise variance.
- We then estimate σ^2 using the RSS. Now although the formula for the standard errors do not rely on the linear model being correct,the estimate for σ^2 does.
- We noticed earlier that there is a non-linear relationship in the data, and so the residuals from a linear fit will be inflated, and so will $\hat{\sigma}^2$.
- Secondly,the standard formulas assume (somewhat unrealistically) that the x_is are fixed, and all the variability comes from the variation in the errors ϵ_i.
- The bootstrap approach does not rely on any of these assumptions, and so it is likely giving a more accurate estimate of the standard errors of $\hat{\beta}_0$ and $\hat{\beta}_1$ than is the summary()

Comparison

Below we compute the bootstrap standard error estimates and the standard 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
set.seed (1)
boot(Auto ,boot.fn ,1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Auto, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
```

OUTPUT Bootstrap Estimates

```
##
## 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
```

OUTPUT Linear Regression Estimates

```
##                      Estimate   Std. Error   t value
## (Intercept)     56.900099702 1.8004268063  31.60367 1.74
## horsepower      -0.466189630 0.0311246171 -14.97816  2.2
## I(horsepower^2)  0.001230536 0.0001220759  10.08009  2.1
```

NOTE:Since this model provides a good/better fit to the data,there is now a better correspondence between the bootstrap estimates and the standard estimates of $SE(\hat{\beta}_0)$, $SE(\hat{\beta}_1)$ and $SE(\hat{\beta}_2)$.