Yonsei University, Department of Applied Statistics

Chapter 5 실습

YBIGTA Science Team 신보현

April 6, 2019

1. Validation Set Approach

```
library(ISLR)
set.seed(1)
train = sample(1:392,196)
lm.fit = lm(mpg~horsepower, data=Auto, subset=train)
attach(Auto)
test.pred = predict(lm.fit,data = Auto[-train,])
mean((mpg[-train] - test.pred)^2)
```

```
## [1] 101.0386
```

2. LOOCV

glm은 로지스틱 회귀를 적용할 때 썼던 함수로, family=”binomial”만 붙이지 않으면 회귀분석과 동일하게 작동한다. 여기서는 CV를 해주는 cv.glm 함수와의 연동을 위해 glm 함수를 사용한다. 둘은 모두 boot library에 있다.

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

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

```
cv.err$delta
```

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

델타의 의미가 무엇일까?
A vector of length two. The first component is the raw cross-validation estimate of prediction error. The second component is the adjusted cross-validation estimate. The adjustment is designed to compensate for the bias introduced by not using leave-one-out cross-validation.
또는 구글링 결과 The first value of delta is the standard k-fold estimate and the second is bias corrected.
아직 정확한 의미는 모르겠다....
이제 차수를 올려가며 LOOCV를 시행하고 각각의 테스트 MSE에 대한 추정치를 구해보자.

```
cv.result = rep(0,5)
for (i in 1:5){
glm.fit = glm(mpg~poly(horsepower,degree=i))
cv.err = cv.glm(Auto, glm.fit)
cv.result[i] = cv.err$delta[1]
}
```

앞서 살펴보았듯이, 차수를 2차로 올렸을 때 가장 급격한 오차의 감소를 보였고 그 이후에는 차수를 높여도 크게 감소하지 않음을 확인할 수 있다. 이를 통해 차수를 높여도 bias의 감소는 크게 이루어지지 않지만 variance은 크게 증가할 것이므로 굳이 높이지 않는 것이 좋을 것이다.

3. K-Fold Cross-Validation

```
cv.result = rep(0,10)
for (i in 1:10){
glm.fit = glm(mpg~poly(horsepower,degree=i))
cv.err = cv.glm(Auto, glm.fit,K=10)
cv.result[i] = cv.err$delta[1]
}
```

4. Bootstrap

부트스트랩의 가장 큰 장점 중 하나는 거의 모든 상황에 적용이 가능하다는 것이다. 어떠한 복잡하고 수학적인 계산이 필요로하지 않는다. R에서는 부트스트랩을 두 가지 단계에 의해서 할 수 있다. 첫 째로 관심있는 통계량을 계산하는 함수를 만든다. 둘 째로 boot() 함수를 이용하여 복원추출로 관측치를 계속적으로 추출한다.
boot() 함수는 Portfolio 데이터를 이용해서 시행한다. 5.2 장에서 살펴보았듯이, 관심있는 통계량은 $\frac{Var(Y)-Cov(X,Y)}{Var(X)+Var(Y)-2Cov(X}$
이다. 따라서 해당 함수를 먼저 만들자.

```
alpha.fn = function(data,index){
X = data$X[index]
Y = data$Y[index]
return((var(X)-cov(X,Y))/(var(X)+var(Y)-2*cov(X,Y)))
}
alpha.fn(Portfolio,1:100)


## [1] 0.4241679
```

해당 함수는 100개의 데이터를 이용하여 α을 추정한다. 이러한 과정을 랜덤하게 100개의 관측치를 복원추출하여 α을 추정해보자.

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

```
## [1] 0.4656526
```

위 코드를 1000번 실행하여 α에 대한 추정치를 구하는 과정이 부트스트랩인데 R에서는 이를 위한 내장 함수가 있다.

```
library(boot)
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* 0.4241679 -0.000751324  0.08802813
```

위 결과로부터 원래의 데이터 세트를 사용하여, $\hat{\alpha} = 0.424, \widehat{S.E}(\hat{\alpha}) = 0.088$ 임을 알 수 있다.
부트스트랩 접근법은 통계적 학습 방법으로부터 계수 추정치와 예측의 변동성을 평가하고 싶을 때 사용될 수 있다.
Auto 데이터 세트에서 horsepower을 사용하여 mpg을 예측한 선형 회귀의 추정된 계수의 변동성을 살펴본다.
먼저 데이터 세트와 인덱스를 인풋으로하는 함수를 만들자.

```
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
```

boot.fn 함수는 관측치들을 복원추출함으로써 절편과 기울기의 부트스트랩 추정치를 만들기 위해서 사용될 수 있다. 두 개의 예시를 보자.

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

```
## (Intercept)  horsepower
##  38.7387134  -0.1481952
```

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

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

이렇게 boot.fn 함수를 1000번 시행해서 각각의 추정치를 기록한 후 표준오차를 기록할 수 있지만 훨씬 편리한 boot 함수를 사용하자.

```
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
```

이를 회귀 분석을 시행했을 때의 표준오차와 비교해보자.

```
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
```

부트스트랩을 통한 추정치의 표준오차와는 사뭇 다른 결과를 확인할 수 있다. 이것이 부트스트랩이 좋지 않음을 의미할까?
단순선형회귀에서 계수의 분산은 σ^2에 의존하고 이는 unknown이기 때문에 SSE을 통해서 추정한다. 따라서 단순선형회귀에서 표준오차 공식은 선형 모델이 맞는지보다는 σ^2가 맞는지에 의존한다. 91쪽 Figure 3.8에서 데이터간에 비선형 관계를 확인했고 그에 따라서 선형 적합에 따른 잔차는 부풀질것이며 σ^2의 추정치 또한 그럴 것이다. 이에 더해서, 단순선형회귀는 독립변수 x가 고정되어 있다는 다소 비 현실적인 가정을 하는데 부트스트랩 접근법은 이러한 가정을 하지 않기 때문에 절편과 기울기의 표준오차 추정치에 대한 좀 더 정확한 결과를 준다.
그렇다면 비선형 관계로 회귀 모형을 적합시켰을 때는 어떠한 결과가 나올까?

```
boot.fn = function(data,index) return(coef(lm(mpg~horsepower+I(horsepower^2),data=data,subset=index)))
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 -7.017193e-02 2.1145821352
## t2* -0.466189630  1.004660e-03 0.0337000425
## t3*  0.001230536 -3.381866e-06 0.0001219536
```

```
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
```

비선형 관계에 좀 더 가까운 mpg와 horsepower의 관계를 반영하여 비선형 회귀 모형을 적합했다. 부스트랩 추정치와 회귀모형의 추정치의 표준오차가 이전보다는 좀 더 가까워졌음을 확인할 수 있었다.
결국, 회귀분석은 반응변수와 예측변수의 관계가 선형적이라는 강력한 가정이 있을 때 좀 더 정확한 결과가 나옴을 확인할 수 있었다. 반면, 부트스트랩은 따로 가정이 없기 때문에 더 유연하게 사용할 수 있는 것으로 보인다.