Лабораторна 4: Перехресна перевірка, Бутстрап.

The validation set approach

Ми досліджуємо використання підходу валідаційного набору для оцінки тестової помилки, для різних лінійних моделей на наборі даних Auto. З допомогою функції sample () розділемо вибірку на дві половини, вибравши випадкову підмножину зі 196 спостережень.

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

Для більш детальної інформації використайте ?sample. З допомогою опції subset використовуємо лише навчальну вибірку для оцінки моделі.

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

Використовуючи функцію predict() для оцінки залежної змінної для всіх 392 спостережень ми обчислюємо MSE для 196 спостережень валідаційної множини. Індекс -train дозволяє вибрати ті спостереження, що не належать тренувальній вибірці.

```
> attach (Auto)
> mean((mpg -predict (lm.fit ,Auto))[-train ]^2)
[1] 26.14
```

Отже, оцінка MSE становить 26.14. З допомогою функції poly() оцінюємо тестову помилку для квадратичної та кубічної регресій.

```
> lm.fit2=lm(mpg~poly(horsepower ,2) ,data=Auto ,subset =train )
> mean((mpg -predict (lm.fit2 ,Auto))[-train ]^2)
[1] 19.82
> lm.fit3=lm(mpg~poly(horsepower ,3) ,data=Auto ,subset =train )
> mean((mpg -predict (lm.fit3 ,Auto))[-train ]^2)
[1] 19.78
```

Отримані оцінки 19.82 та 19.78 відповідно. Змінивши розбиття на множини, отримаємо дещо інші результати.

```
> set.seed (2)
> train=sample (392 ,196)
> lm.fit =lm(mpg~horsepower ,subset =train)
> mean((mpg -predict (lm.fit ,Auto))[-train ]^2)
[1] 23.30
> lm.fit2=lm(mpg~poly(horsepower ,2) ,data=Auto ,subset =train )
> mean((mpg -predict (lm.fit2 ,Auto))[-train ]^2)
[1] 18.90
> lm.fit3=lm(mpg~poly(horsepower ,3) ,data=Auto ,subset =train )
> mean((mpg -predict (lm.fit3 ,Auto))[-train ]^2)
[1] 19.26
```

Тепер отримали оцінки 23.30, 18.90 та 19.26. По-суті, ми отримали те ж, що і раніше: модель для передбачення mpg на основі квадратичної функції від horsepower є краща ніж лінійна модель та немає достатньо доказів, що кубічна модель дасть кращі результати.

Leave-One-Out Cross-Validation

Оцінку для LOOCV можна обчислити дл довільної узагальненої лінійної моделі з використанням функцій glm() і cv.glm(). Якщо ми використаємо glm() для оцінки моделі не вказуючи параметр family, тоді буде побудована лінійна регресія як і у випадку використання функції lm(). Отже,

```
> glm.fit=glm(mpg~horsepower ,data=Auto)
> coef(glm.fit)
(Intercept)   horsepower
39.936        - 0.158
```

і

```
> lm.fit =lm(mpg~horsepower ,data=Auto)
> coef(lm.fit)
(Intercept )  horsepower
39.936        -0.158
```

Ми використаємо функцію glm(), оскільки вона дає можливість використовувати функцію cv.glm(), яка є частиною бібліотеки boot.

```
> library (boot)
> glm.fit=glm(mpg~horsepower ,data=Auto)
> cv.err =cv.glm(Auto ,glm.fit)
> cv.err$delta
1            1
24.23        24.23
```

Дана функція повертає список з декількох компонент. Два числа у векторі дельта містять результати перехресної перевірки. Ми отримали однакові результати для LOOCV. Перший відповідає звичайній оцінці, а другий скорегованій.

Отже, ми отримали оцінку для тестової помилки на рівні 24.23. Повторимо алгоритм оцінки тестової помилки для поліноміальних функцій. Використаємо функцію for() для побудови циклу для оцінки моделей зі степенями від i = 1 до i = 5 та оцінки тестової помилки, яка записується у вектор cv.error.

```
> 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] 24.23 19.25 19.33 19.42 19.03
```

Бачимо зниження помилки при переході до квадратичної моделі та відсутність істотного покращення для вищих степенів.

k-Fold Cross-Validation

Функція cv.glm() може бути використана також для реалізації k-групової перехресної перевірки. Ми використаємо k = 10, ми також задамо вектор для збереження оцінок тестових помилок для степеневих функції з показниками від 1 до 10.

```
> 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] 24.21 19.19 19.31 19.34 18.88 19.02 18.90 19.71 18.95 19.50
```

Ми не бачимо достатньо доказів, що поліноми третього і вищого степенів покращують результати. Також, варто звернути увагу, що в цьому випадку числа вектора дельта є різними.

Бутстрап

Продемонструємо використання бутстрапу для двох випадків:

1. Припустимо, що ми хочемо вкласти фіксовану суму грошей у два фінансових активи, що мають дохідність X та Y відповідно, де X та Y випадкові величини. Ми вкладемо частку α наших грошей у X, а решту 1 - α в Y. Ми хочемо вибрати α, щоб мінімізувати загальну суму ризик наших інвестицій.

2. Дані Auto для передбачення mpg.

Приклад 1. Метод бутстрапу можна використовувати до будь-яких моделей. Він не вимагає важких обчислень. Нам потрібно всього задати два кроки:

Перший, задати правило обчислення статистики, що нас цікавить.
Другий, використовуючи функцію boot(), яка є частиною бібліотеки boot, провести сам бутстрап шляхом багаторазового відбору спостережень з повторенням. Необхідні дані Portfolio задані в пакеті ISLR. Задамо спочатку функцію alpha.fn(), яка на основі вибірок (X, Y) з вказаним діапазоном для них обчислює та повертає оцінку для параметра α.

```
> 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)))
+ }
```

Оцінити α на основі всіх 100 спостережень

```
> alpha.fn(Portfolio ,1:100)
[1] 0.576
```

Використовуючи функцію sample() випадково вибираємо 100 спостережень з діапазону від 1 до 100 з повторенням та знову оцінюємо α.

```
> set.seed (1)
> alpha.fn(Portfolio ,sample (100 ,100 , replace =T))
[1] 0.596
```

Бутстрап аналі проводимо шляхом багаторазового повторення попередньої процедури, зберігаючи отримані оцінки α. Для цього можемо побудувати цикл, або використати вбудовані можливості функції 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*   0.5758      -7.315e    -05 0.0886
```

Отже, ми отримали оцінку для α 0.5758, а її стандартне відхилення 0.0886.

Приклад 2.

Ми використаємо бутстрап для того, щоб оцінити середньоквадратичне відхилення параметрів лінійної регресійної моделі β_0 та β_1, моделі, яка використовує horsepower для прогнозування mpg на даних Auto. Ми порівняємо отримані оцінки, з оцінками отриманими з класичних формул. Створимо спочатку функцію boot.fn (), яка приймає дані Auto, а також набір потрібних для побудови моделі індексів і повертає оцінки параметрів моделі лінійної регресії.

Використаємо цю функції для повного набору з 392 спостережень для обчислення оцінок параметрів моделі звичайної лінійної регресії.

```
> boot.fn=function (data ,index )
+ return (coef(lm(mpg~horsepower ,data=data ,subset =index)))
> boot.fn(Auto ,1:392)
(Intercept)  horsepower
39.936       -0.158
```

Цю функцію boot.fn() ми можемо використати для побудови оцінок на основі бутстрапу. Наприклад,

```
> set.seed (1)
> boot.fn(Auto ,sample(392 ,392 , replace =T))
(Intercept)  horsepower
38.739       -0.148
> boot.fn(Auto, sample(392 ,392 , replace =T))
(Intercept )  horsepower
40.038       -0.160
```

Використовуючи функцію boot() обчислимо оцінку середньоквадратичного відхилення для параметрів моделі на основі 1,000 повторень бутстрапу.

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

ORDINARY NONPARAMETRIC BOOTSTRAP

Call:
boot(data = Auto, statistic = boot.fn, R = 1000)

Bootstrap Statistics :
    original     bias     std. error
t1* 39.936       0.0297   0.8600
t2* -0.158      -0.0003   0.0074
```

Ми отримали середньоквадратичне відхилення для β_0 0.86, а для β_1 – 0.0074. Використання класичних формул для обчислення цих величин:

```
> summary(lm(mpg~horsepower ,data=Auto))$coef
             Estimate Std. Error t value  Pr(>|t|)
(Intercept)    39.936    0.71750    55.7 1.22e-187
horsepower     -0.158    0.00645   -24.5  7.03e-81
```

Ми отримали середньоквадратичне відхилення для β_0 0.717, а для β_1 – 0.0064.

Про що говорить різниця у отриманих оцінках? Насправді немає жодної підстави вважати, що оцінки отримані на основі бутстрапу є поганими. Стандартні формулиотримані на основі певних припущень. Вони також залежать від невідомого параметра σ^2, дисперсії залишків. Крім того, ми пам'ятаємо, що, в розглянутих даних існує нелінійна залежність, а отже, і оцінки дисперсії залишків будуть завищені. Також, стандартні формули припускають, що незалежні змінні не є випадковими, тобто вся випадковість "ховається" в залишках. Бутстрап не покладається на жодне з цих припущень, і тому він, в даному випадку, є, ймовірно, кращим для оцінки відхилень оцінок параметрів. Обчислимо середньоквадратичні відхилення оцінок параметрів лінійної регресії на основі квадратичної функції. Оскільки в цьому випадку модель краще описує дані, то і отримані оцінки відхилення є ближчими.

```
> 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*   56.900     6.098e-03 2.0945
t2*  -0.466     -1.777e-04 0.0334
t3*   0.001      1.324e-06 0.0001

> summary(lm(mpg~horsepower+I(horsepower^2),data=Auto))$coef
                 Estimate  Std. Error  t value  Pr(>|t|)
(Intercept)       56.9001     1.80043       32  1.7e-109
horsepower        -0.4662     0.03112      -15   2.3e-40
I(horsepower^2)    0.0012     0.00012       10   2.2e-21
```