2020

DataMining

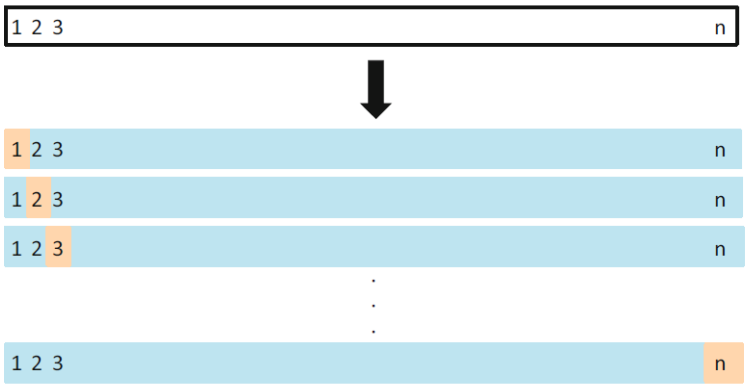
HW4

192STG11 우나영

1. Description

Resampling 은 training set 으로부터 반복적으로 sampling 하여 다양한 sample 들을 모델에 적합하는 기법이다. 이를 통해 모델의 성능을 평가하는 척도를 계산(model assessment)하거나 모델을 선택하는 기준(model selection)을 선정하는 등 모델에 대한 다양한 정보를 얻을 수 있다. 이번 챕터에서는 resampling 기법으로 대표적인 cross validation 과 bootstrap 을 배웠다. Cross validation 을 이해하기 위해서는 validation approach 에 대한 개념을 짚고 넘어가야한다. Validation approach 는 간단하게 test error 를 구하기 위해 전체 데이터 셋을 랜덤하게 training set 과 validation set 으로 나눈 후, training set 으로 적합한 모형에 validation set 을 대입하여 test error rate 를 측정하는 것이다. 이 개념은 매우 간단하다는 장점이 있지만 training set 과 validation set 에 따라 test error rate 가 가변적이다. 또한 일반적으로 관측치의 수가 적을수록 적합한 모형의 test error rate 가 높기 때문에 만약 관측치 수가 적은 training set 으로 적합한 모형을 validation set 으로 test error rate 계산한다면 과대측정(overestimated)될 수 있다. 이를 보정해준 방법이 cross validation 기법이다.

Leave-one-out-cross-validation(LOOCV)은 데이터 전체 관측치 수가 n 일때, n-1 개의 관측치(training set)으로 모델을 적합한 후, 나머지 하나의 관측치(validation set)로 모델의 test error 를 계산하는 방법이다. 이러한 과정을 모든 n 개의 관측치에 반복해 구한 test error 의 평균이 LOOCV 값이 된다. 예를 들면, test MSE 에 대한 LOOCV 는 $CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n} MSE_i$ 이다. 아래는 LOOCV 를 도식화한 것으로 파란색은 training set 을 나타내고 베이지색은 validation set 을 나타낸다.



K-fold cross-validation 은 전체 데이터 셋을 k 개의 그룹으로 균등하게 나눈 후, k-1 개의 그룹(training set)으로 적합한 모델에 나머지 하나의 그룹(validation set)을 대입하여 test error 를 계산하는 방법이다. 이러한 과정을 k 번 반복하여 구한 test error 의 평균이 k-fold cross-validation 의 값이 된다. 즉, test MSE 에 대한 k-fold cross validation 은 $CV_{(k)} =$

$\frac{1}{k}\sum_{i=1}^{k} MSE_i$이다. LOOCV 는 k 가 데이터 관측치 수인 n 에 해당하는 특수한 케이스임을 알 수 있다. 하지만 LOOCV 는 계산 비용이 크기 때문에 k=5 또는 k=10 을 실전에서 사용한다.

Resampling 의 또 다른 기법으로 bootstrap 이 있다. 이는 추정치와 통계적 모델의 정확도의 측도(measure of accuracy)를 측정할 때 매우 유용하며 빈번히 사용된다. Bootstrap 은 original sample 로부터 반복적으로 복원 추출하여 sampling 하는 기법이다. 복원 추출된 sample 로부터 정확도의 측도로 사용되는 bias, variance 그리고 confidence interval 등의 통계량을 계산할 수 있다.

이번 과제에서는 chapter 5 에서 배운 resampling 개념을 숙지한 후 Lab 5.3 을 통해 cross validation 과 bootstrap 을 구현하는 R 코드를 학습하고 이를 바탕으로 예제 2, 5, 7, 9 번을 풀어볼 것이다.

2. Implementation

Lab 5.3 에서는 cross validation 및 bootstrapping 을 구현하는 R code 를 배울 수 있었다. Resampling 의 경우 random 하게 실행하기 때문에 재현성을 위해 seed 넘버를 항상 설정해야 한다. Cross validation 의 경우 boot 패키지의 cv.glm()으로 계산 가능하다. 한가지 흥미로운 사실은 glm() 함수를 이용하여 lm fitting 이 가능하다는 것이다. 이는 glm 의 family = "binomial"이라는 argument 를 생략하면 구현 가능하다. Bootstrapping 을 위해서는 boot()함수를 사용한다. Boot 함수에는 관심 통계량의 계산 함수, original dataset 그리고 시행횟수를 argument 로 넣어준다. 이러한 코드를 사용하여 예제 2, 5, 7, 9 번을 풀어보자.

2. We will now derive the probability that a given observation is part of a bootstrap sample. Suppose that we obtain a bootstrap sample from a set of n observations

(a) What is the probability that the first bootstrap observation is not the jth observation from the original sample? Justify your answer?

j번째 관측치를 뽑을 확률은 $\frac{1}{n}$이므로 j가 아닌 관측치를 뽑을 확률은 $1-\frac{1}{n}$이다.

(b) What is the probability that the second bootstrap observation is not the jth observation from the original sample?

복원 추출이므로 두 번째 bootstrap sample 이 original sample 의 j번째 관측치가 아닐 확률 역시 (a)번과 같이 $1-\frac{1}{n}$이다.

(c) Argue that the probability that the jth observation is not the bootstrap sample is $(1-1/n)^n$.

(a)와 (b)에서 봤듯이 j 번째 관측치가 추출되지 않을 확률은 $1-\frac{1}{n}$ 이다. Bootstrapping 은 복원 추출이기 때문에 이미 추출된 관측치의 영향을 받지 않는다. 따라서 j번째 관측치가 추출되지 않을 확률은 $(1-\frac{1}{n})^n$이다.

(d) When n=5, what is the probability that the jth observation is in the bootstrap sample?

$1-(1-1/5)^5 = 0.67232$

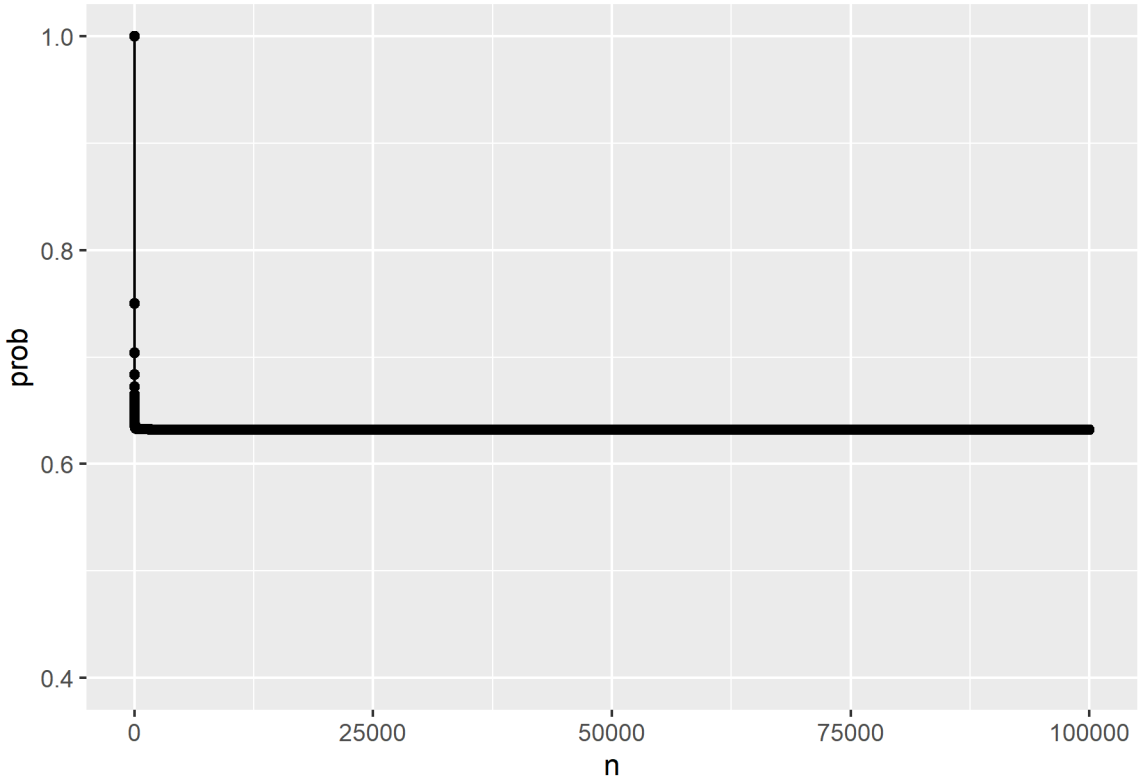
(e) When n=100, what is the probability that the jth observation is in the bootstrap sample?

$1-(1-1/100)^{100} = 0.6339677$

(f) When n=10,000, what is the probability that the jth observation is in the bootstrap sample?

$1-(1-1/10000)^{10000} = 0.632139$

(g) Create a plot that displays, for each integer value of n from 1 to 100,000, the probability that the jth observation is in the bootstrap sample. Comment on what you observe.



특정 관측치가 bootstrap sample 에서 관측될 확률은 n 이 커짐에 따라 0.63 부근으로 수렴한다. 즉, bootstrap sample size 가 일정 범위를 넘어가게 되면 size 에 관계 없이 어떤 관측치가 bootstrap sample 에서 관측될 확률은 0.63 에 가깝다는 것을 알 수 있다.

(h) We will now investigate numerically the probability that a bootstrap sample of size n = 100 contains the jth observation. Here j=4. We repeatedly create bootstrap samples, and each time we record whether or not the fourth observation is contained in the bootstrap sample. Comment on the results obtained.

size 가 100 인 bootstrap sample 을 10000 번 반복적으로 생성시켜 original set 의 4 번째 관측치가 bootstrap sample 에서 관측될 확률을 계산해 본 결과 0.6308 으로 나와 실제로도 (g)에서 구한 이론적 확률과 비슷한 값을 보여준다는 것을 알 수 있다.

5. In Chapter 4, we used logistic regression to predict the probability of default using income and balance on the Default data set. We will now estimate the test error of this logistic regression model using the validation set approach. Do not forget to set a random seed before beginning your analysis.

(a) Fit a logistic regression model that uses income and balance to predict default.

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-11.5405	0.434756	-26.5447	2.96E-155
income	2.08E-05	4.99E-06	4.174178	2.99E-05
balance	0.005647	0.000227	24.83628	3.64E-136

(b) Using the validation set approach, estimate the best error of this model.

Validation set approach 로 train set 과 validation set 을 절반으로 나눈 후 모델의 test error 의 추정치를 계산한 결과 0.0254 가 나왔다. 이때 seed number 를 1 로 설정했다.

(c) Repeat the process in (b) three times, using three different splits of the observations into a training set and a validation set. Comment on the results obtained.

Number of iterations	1	2	3
Estimate	0.0238	0.0264	0.0256

(d) Now consider a logistic regression model that predicts the probability of default using income, balance, and a dummy variable for student. Estimate the test error for this model using the validation set approach. Comment on whether or not including a dummy variable for student leads to a reduction in the test error rate.

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-11.339	0.693703	-16.3456	4.68E-60
income	1.69E-05	1.12E-05	1.502134	0.133063
balance	0.005767	0.000321	17.94718	5.05E-72
studentYes	-0.59922	0.332427	-1.80257	0.071455

7. In Sections 5.3.2 and 5.3.3, we saw that the cv.glm() function can be used in order to compute the LOOCV test error estimate. Alternatively, one could compute those quantities using just the glm() and predict.glm() functions, and a for loop. You will now take this approach in order to compute the LOOCV error for a simple logistic regression model on the Weekly data set. Recall that in the context of classification problems, the LOOCV error is given in (5.4).

(a) Fit a logistic regression model that predicts Direction using Lag1 and Lag2.

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	0.221224	0.061466	3.599145	0.000319
Lag1	-0.03872	0.026217	-1.47701	0.139672
Lag2	0.060248	0.026546	2.26959	0.023232

(b) Fit a logistic regression model that predicts Direction using Lag1 and Lag2 using all but the first observation.

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	0.223243	0.061499	3.630031	0.000283
Lag1	-0.03843	0.026219	-1.46587	0.142683
Lag2	0.060848	0.026561	2.290874	0.021971

(c) Use the model from (b) to predict the direction of the first observation. You can do this by predicting that the first observation will go up if P(Direction = "Up"|Lag1, Lag2) > 0.5. Was this observation correctly classified?

첫번째 관측치는 "Down"이지만 (b)의 모델을 사용한 것은 "Up"으로 잘못 예측하였다.

(d)(e) estimate LOOCV.

test error 에 대한 LOOCV 값이 0.4499541 로 예측력이 55%정도 된다는 것을 알 수 있다.

9. We will now consider the Boston housing data set, from the MASS library.

(a) Based on this data set, provide an estimate for the population mean of medv. all this $\hat{\mu}$.

$\hat{\mu} = 22.53281$

(b) Provide an estimate of the standard error of $\hat{\mu}$. Interpret this result.

$\mathrm{SE}(\hat{\mu}) = 0.4088611$

Hint : We can compute the standard error of the sample mean by dividing the sample standard deviation by the square root of the number of observations.

medv 의 모평균 추정치인 sample mean 의 표준오차는 0.4088611이다.

(c) Now estimate the standard error of $\hat{\mu}$ using the bootstrap. How does this compare to your answer from (b)?

1000 번의 bootstrapping 으로 구한 $\hat{\mu}$의 standard error 값은 0.4106622 로 (b)보다 크다.

(d) Based on your bootstrap estimate from (c), provide a 95% confidence interval for the mean of medv. Compare it to the results obtained using t.test(Boston$medv).

Hint: You can approximate a 95% confidence interval using the formula $[\hat{\mu} - 2\mathrm{SE}(\hat{\mu}), \hat{\mu} + 2SE(\hat{\mu})]$.

Method	95% Confidence Interval
bootstrap	(21.71149, 23.35413)
t.test	(21.72953, 23.33608)

t.test 를 이용해 구한 신뢰구간보다 bootstrap 을 이용한 신뢰구간이 더 넓다는 것을 알 수 있다.

(e) Based on this data set, provide an estimate, $\hat{\mu}_{med}$, for the median value of medv in the population.

$\hat{\mu}_{med} = 21.2$

(f) We now would like to estimate the standard error of $\hat{\mu}_{med}$. Unfortunately, there is no simple formula for computing the standard error of the median. Instead, estimate the standard error of the median using the bootstrap. Comment on your findings.

$\mathrm{SE}(\hat{\mu}_{med}) = 0.3778075$

Bootstrap 을 이용해 구한 $\hat{\mu}_{med}$의 standard error 는 0.3778075이다.

(g) Based on this data set, provide an estimate for the tenth percentile of medv in Boston suburbs. Call this quantity $\hat{\mu}_{0.1}$.(You can use the quantile() function).

$\hat{\mu}_{0.1} = 12.75$

(h) Use the bootstrap to estimate the standard error of $\hat{\mu}_{0.1}$. Comment on your findings.

$SE(\hat{\mu}_{0.1}) = 0.4768056$

Bootstrap 을 이용해 구한 $\hat{\mu}_{0.1}$의 standard error 는 0.4768056이다.

3. Discussion

이번 과제를 통해 Resampling 의 대표적인 기법인 cross validation 과 bootstrapping 을 배울 수 있었다. Resampling 기법은 original set 에서 반복적으로 새로운 sample 을 뽑아 관심 통계량을 구하는 방법이다. Cross validation 은 validation approach 의 단점을 개선시킨 방법으로 leave-one-out-cross-validation(LOOCV)와 k-fold cross validation 이 있다. Bootstrap 은 original set 으로부터 복원추출을 반복적으로 하는 것이다. 예제 2 번을 통해 Bootstrap sample 의 n 이 커짐에 따라 original set 의 각각의 관측치가 Bootstrap sample 에 포함될 확률이 0.63 에 수렴한다는 것을 알 수 있었다. Lab 5.3 에서는 이러한 resampling 을 R 에서 구현하는 코드를 배울 수 있었다. Cross validation 을 구현하기 위한 cv.glm()와 Bootstrap 을 구현하기 위한 boot()가 대표적이다. 마지막으로는 실제 예제 데이터에 resampling 기법을 활용하여 관심 통계량과 그 통계량의 standard error 를 구해볼 수 있었다.

4. Appendix

```
pacman::p_load(ggplot2, tidyverse, dplyr, plotly, processx)

rm(list=ls(all=TRUE))

setwd('C:/Users/dnskd/Desktop/20Spring/datamining/week5')


## 5.3 Lab : Cross-Validation and the Bootstrap

## 5.3.1 The Validation Set Approach

library(ISLR)

set.seed(1)

train = sample(392, 196) # index 만 뽑았다는 점, sample 의 short cut 을 사용했다는 점

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

attach(Auto)

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

lm.fit2 <- lm(mpg ~ poly(horsepower, 2), data = Auto, subset = train)

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

lm.fit3 <- lm(mpg ~ poly(horsepower, 3), data = Auto, subset = train)

mean((mpg - predict(lm.fit3, Auto))[-train]^2)

set.seed(2)

train = sample(392, 196)

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

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

lm.fit2 = lm(mpg ~ poly(horsepower, 2), data = Auto, subset = train)

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

lm.fit3 <- lm(mpg ~ poly(horsepower, 3), data = Auto, subset = train)

mean((mpg - predict(lm.fit3, Auto))[-train]^2)


## 5.3.2 Leave-One-Out Cross-Validation

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

coef(glm.fit)

lm.fit <- lm(mpg ~ horsepower, data = Auto)

coef(lm.fit)

if(!require(boot)) install.packages('boot'); library(boot)

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

cv.err = cv.glm(Auto, glm.fit)

cv.err$delta

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


## 5.3.3 k-Fold Cross-Validation

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


## 5.3.4 The Bootstrap

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)))

}

alpha.fn(Portfolio, 1:100)

set.seed(1)

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

boot(Portfolio, alpha.fn, R = 1000)

boot.fn = function(data, index){

  return(coef(lm(mpg ~ horsepower, data = data, subset = index)))

}

boot.fn(Auto, 1:392)

set.seed(1)

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

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

boot(Auto, boot.fn, 1000)

summary(lm(mpg ~ horsepower, data = Auto))$coef


boot.fn = function(data, index){

  coefficients((lm(mpg ~ horsepower + I(horsepower^2), data = data, subset = index)))

}

set.seed(1)
```

```
boot(Auto, statistic = boot.fn, R = 1000)

summary(lm(mpg ~ horsepower +
I(horsepower^2), data = Auto))$coef



## 5.2

# g.

boot_is <- function(n){

  1-(1-(1/n))^n

}

n <- 1:100000

df <- data.frame(n = n, prob = sapply(n,
boot_is))

df %>% ggplot(aes(n, prob)) + geom_path() +
geom_point() + ylim(c(0.4, 1))

ggsave('ex2.png')


# h.

store = rep(NA, 10000)

for (i in 1:10000){

  store[i] = sum(sample(1:100, rep =
TRUE)==4)>0

}

mean(store)


## 5.5

# a.

library(ISLR)

set.seed(1)

glm.fit <- glm(default ~ income + balance,
data = Default, family = "binomial")

sum.coef <- summary(glm.fit)$coef

write.csv(sum.coef, "fitting.csv")


# b.

train <- sample(nrow(Default),
nrow(Default)*0.5, replace = FALSE)

glm.fit <- glm(default ~ income + balance,
data = Default[train,], family = "binomial")

post <- predict(glm.fit, Default[-train, ],
type = "response")

pred <- ifelse(post > 0.5, "Yes", "No")

mean(pred != Default$default[-train])


# c.

sampling <- function(i){

  set.seed(i)

  train <- sample(nrow(Default),
nrow(Default)*0.5, replace = FALSE)

  glm.fit <- glm(default ~ income + balance,
data = Default[train,], family = "binomial")

  post <- predict(glm.fit, Default[-train, ],
type = "response")

  pred <- ifelse(post > 0.5, "Yes", "No")

  return(mean(pred != Default$default[-
train]))

}

sampling(2); sampling(3); sampling(4)


# d.

set.seed(1)

train <- sample(nrow(Default),
nrow(Default)*0.5, replace = FALSE)

glm.fit <- glm(default ~ income + balance +
student, data = Default[train,], family =
"binomial")

post <- predict(glm.fit, Default[-train, ],
type = "response")

pred <- ifelse(post > 0.5, "Yes", "No")

mean(pred != Default$default[-train])

sum.coef2 <- summary(glm.fit)$coef

write.csv(sum.coef2, 'sum.csv')


## 7.

# a.

glm.fit <- glm(Direction ~ Lag1 + Lag2, data
= Weekly, family = "binomial")

summary_7 <- summary(glm.fit)$coef

write.csv(summary_7, "summary_7.csv")


# b.

glm.fit <- glm(Direction ~ Lag1 + Lag2 ,
data = Weekly[-1, ], family = "binomial")

summary_7_b <- summary(glm.fit)$coef

write.csv(summary_7_b, "summary_7_b.csv")


# c.

ifelse(predict(glm.fit, Weekly[1, ], type =
"response")>0.5, "Up", "Down") == Weekly[1,
"Direction"]


# d.

result <- c()

for(i in 1:nrow(Weekly)){

  glm.fit <- glm(Direction ~ Lag1 + Lag2,
data = Weekly[-i, ], family = "binomial")

  result[i] <- sum(Weekly[i, "Direction"] !=
ifelse(predict(glm.fit, Weekly[i, ], type =
"response")>0.5, "Up", "Down"))

}
```

```
sum(result)/nrow(Weekly)



## 9.
# a.
library(MASS)
names(Boston)
mean(Boston$medv)

# b.
sd(Boston$medv)/sqrt(nrow(Boston))

# c.
library(boot)
set.seed(1)
boot.fn <- function(data, index){
  return(mean(data[index, "medv"]))
}
result <- boot(Boston, boot.fn, 1000)
result

# d.
c(22.53281 - 2*0.4106622, 22.53281 +
2*0.4106622)
t.test(Boston$medv)

# e.
median(Boston$medv)

# f.
set.seed(1)
boot.fn <- function(data, index){
  return(median(data[index, "medv"]))
}
boot(Boston, boot.fn, 1000)

# g.
quantile(Boston$medv, 0.1)

# h.
set.seed(1)
boot.fn <- function(data, index){
  return(quantile(data[index, "medv"], 0.1))
}
boot(Boston, boot.fn, 1000)
```