­

Ch5. Statistical Learning

-Lab & Exercises #2,5,7,9

­­

**학번 : 182STG12**

**이름 : 오혜윤**

c

Problem Description

In Chapter 5 lab, we learn about resampling methods such as CV, Bootstrap. Using sampling methods, we generate the samples for training set and validation set. Then, we fit various models in order to predict the dataset. The first model is logistic regression and we use the function called glm(). In Auto data, there are many variables such as mpg and horsepower. We predict the mpg value using horsepower.

We generate different samples by using different sampling methods. There are several sampling methods like LOOCV, K-fold validation, and bootstrap. Then, we fit glm models. As a result, we calculate the mean of CV error rate.

In Chapter 5 exercises, we compare the results of many samplings. In the first question, we calculate the probability of bootstrap sampling and compare the result with the original value. In the second and third question, using Default and Weekly dataset, we want to calculate the CV error rate. By splitting the dataset and comparing values, we can calculate the CV error rate. In the last question, we generate the bootstrap samples. Also, we compare the original estimates of the population and the estimates by using bootstrap samples. We can conclude that the estimates of bootstrap is very close to the original estimates of the population.

Results

**CH5. Lab Review**

In this lab, it was mostly about the sampling methods. First, using Auto dataset, we learn about the use of the validation set approach to estimate the test error rates that result from fitting various linear models. We begin by using the sample() function to split the set of observations. By using sample() function, we split the observations into the training set and the validation set. Through this process, we can compare the prediction results and the validation set.

After that, we learn about the LOOCV and K-fold validation. The LOOCV and K-fold validation are validation methods. The LOOCV estimate can be automatically computed for any generalized linear model using the glm() and cv.glm() functions. The computation time of K-fold validation is much shorter than that of LOOCV. Also, cv error of K-fold validation method (about 19.1) is much lower than that of LOOCV (24.23).

Finally, we learn about the bootstrap. The bootstrap approach can be used to assess the variability of the coefficient estimates and predictions from a statistical learning method. The bootstrap is one of the most popular validation methods.

We can apply many validation approaches to the dataset in this lab. Comparing each validation methods, we can choose the best sampling method.

**CH5 Exercises**

**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.**

1. **What is the probability that the first bootstrap observation is not the jth observation from the original sample? Justify your answer.**

Given that there are n observations and bootstrap sampling draws items. Excluding the jth observation, the total number of items are n-1. The probability is (n-1)/n=(.

1. **What is the probability that the second bootstrap observation is not the jth observation from the original sample?**

The probability is the same as above (). Again, the second time you pick an observation, the set of observations you start with is the same, because you are sampling with replacement.

1. **Argue that the probability that the jth observation is not in the bootstrap sample is (1 − 1/n)n.**

The probability that the jth sample is not the first sample in your bootstrap is . The total bootstrap sample size is n. So, we need to pick n different observations and none of them should be the jth one. As bootstraping does sampling with replacement, the probabilities of each observation are independent of one another. If that is the case, then we just have to multiply ) , n times, therefore the answer is .

1. **When n = 5, what is the probability that the jth observation is in the bootstrap sample?**

The probability that the jth observation is in the bootstrap sample is just going to be . With n=5, the answer will be =0.6723.

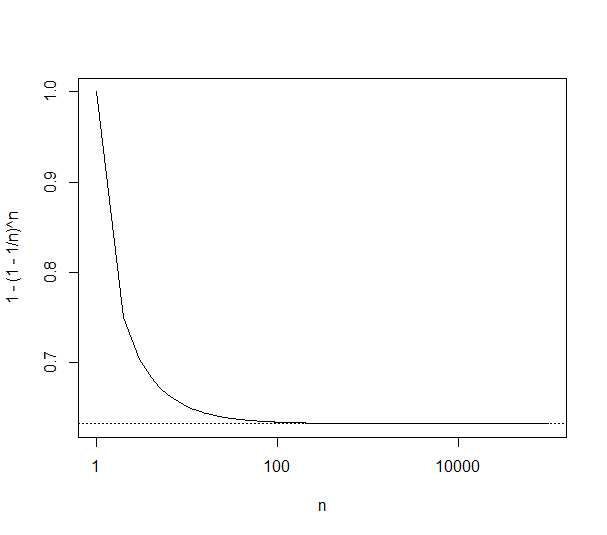
1. **When n = 100, what is the probability that the jth observation is in the bootstrap sample?**

For n = 100, = 0.6340.

1. **When n = 10, 000, what is the probability that the jth observation is in the bootstrap sample?**

For n = 10000, = 0.6321.

1. **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.**



In this plot, abline represents 1-exp(-1). The limit of as n goes to infinity is exp(-1). Therefore, as n gets larger, we can see approaches 1 - exp(-1), which is approximately equal to 0.632.

1. **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.**

The mean(store) call yields the proportion of bootstrap samples that contain the 4th observation. As expected, we see the selection probability is approximately equal to , which is approximate equal to 1 - exp(-1) (0.632).

**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.**

1. **Fit a logistic regression model that uses income and balance to predict default.**

|  |
| --- |
| AIC: 1585 |
| glm.fit = glm(default ~ income + balance,data = Default,family = "binomial")  summary(glm.fit) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Deviance Residual | | | | |
| Min | 1Q | Median | 3Q | Max |
| -2.4725 | -0.1444 | -0.0574 | -0.0211 | 3.7245 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Coefficients: |  |  |  |  |
|  | Estimate | Std. Error | z value | Pr(>|z|) |
| (Intercept) | -1.154e+01 | 4.348e-01 | -26.545 | < 2e-16 \*\*\* |
| income | 2.081e-05 | 4.985e-06 | 4.174 | 2.99e-05 \*\*\* |
| balance | 5.647e-03 | 2.274e-04 | 24.836 | < 2e-16 \*\*\* |

1. **Using the validation set approach, estimate the test error of this model. In order to do this, you must perform the following steps:**

**i. Split the sample set into a training set and a validation set.**

|  |
| --- |
| R codes: |
| >train = sample(dim(Default)[1], dim(Default)[1] / 2) |

**ii. Fit a multiple logistic regression model using only the training observation.**

|  |
| --- |
| AIC: 740.4 |
| glm.fit = glm(default ~ income + balance,data = Default[train,],family = "binomial")  summary(glm.fit) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Deviance Residual | | | | |
| Min | 1Q | Median | 3Q | Max |
| -2.3583 | -0.1268 | -0.0475 | -0.0165 | 3.8116 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Coefficients: |  |  |  |  |
|  | Estimate | Std. Error | z value | Pr(>|z|) |
| (Intercept) | -1.208e+01 | 6.658e-01 | -18.148 | <2e-16 \*\*\* |
| income | 1.858e-05 | 7.573e-06 | 2.454 | 0.0141 \* |
| balance | 6.053e-03 | 3.467e-04 | 17.457 | < 2e-16 \*\*\* |

**iii. Obtain a prediction of default status for each individual in the validation set by computing the posterior probability of default for that individual and classifying the individual to the default category if the posterior probability is greater than 0.5.**

|  |
| --- |
| R codes: |
| >glm.probs = predict(fit.glm, newdata = Default[-train, ], type="response")  >glm.pred=rep("No",5000)  >glm.pred[glm.probs>0.5] = "Yes" |

**iv. Compute the validation set error, which is the fraction of the observations in the validation set that are misclassified.**

|  |
| --- |
| R codes: |
| > mean(glm.pred != Default[-train, ]$default)  **[1] 0.0286** |

1. **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.**

|  |  |  |  |
| --- | --- | --- | --- |
| Simulation Result | | | |
|  | Simul1 | Simul2 | Simul3 |
| validation error | **0.0236** | **0.028** | **0.0268** |

The validation errors of simulations are 0.0236, 0.028, 0.0268 separately. The average of all three errors is about 0.026.

1. **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.**

|  |
| --- |
| R codes: |
| > train <- sample(dim(Default)[1], dim(Default)[1] / 2)  >glm.fit <- glm(default ~ income + balance + **student**, data = Default[train,], family = "binomial")  >pred.glm <- rep("No", length(probs))  >probs <- predict(glm.fit, newdata = Default[-train, ], type = "response")  >pred.glm[probs > 0.5] <- "Yes"  >mean(pred.glm != Default[-train, ]$default)  **[1] 0.0246** |

The validation error of the model including the “student” variable is 0.0246. It is lower than the previous errors. However, the difference is about 0.002. Therefore, it doesn’t seem that adding the “student” dummy variable leads to a high reduction in the validation error.

**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).**

1. **Fit a logistic regression model that predicts Direction using Lag1 and Lag2.**

|  |
| --- |
| AIC: 1494.2 |
| glm.fit = glm(Direction ~ Lag1 + Lag2, data= Weekly, family = binomial)  summary(glm.fit) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Deviance Residual | | | | |
| Min | 1Q | Median | 3Q | Max |
| -1.623 | -1.261 | 1.001 | 1.083 | 1.506 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Coefficients: |  |  |  |  |
|  | Estimate | Std. Error | z value | Pr(>|z|) |
| (Intercept) | 0.22122 | 0.06147 | 3.599 | 0.000319 \*\*\* |
| Lag1 | -0.03872 | 0.02622 | -1.477 | 0.139672 |
| Lag2 | 0.06025 | 0.02655 | 2.270 | 0.023232 \* |

1. **Fit a logistic regression model that predicts Direction using Lag1 and Lag2 using all but the first observation.**

|  |
| --- |
| AIC: 1492.5 |
| glm.fit = glm(Direction ~ Lag1 + Lag2, data= Weekly[-1,], family = binomial)  summary(glm.fit) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Deviance Residual | | | | |
| Min | 1Q | Median | 3Q | Max | |
| -1.6258 | -1.2617 | 0.9999 | 1.0819 | 1.5071 | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Coefficients: |  |  |  |  |
|  | Estimate | Std. Error | z value | Pr(>|z|) |
| (Intercept) | 0.22324 | 0.06150 | 3.630 | 0.000283 \*\*\* |
| Lag1 | -0.03843 | 0.02622 | -1.466 | 0.142683 |
| Lag2 | 0.06085 | 0.02656 | 2.291 | 0.021971 \* |

1. **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?**

The prediction for the first observation is “Up”. This observation was not correctly classified as the true direction is “Down”.

1. **Write a for loop from i = 1 to i = n, where n is the number of observations in the data set, that performs each of the following steps:**

**i. Fit a logistic regression model using all but the ith observation to predict Direction using Lag1 and Lag2.**

**ii. Compute the posterior probability of the market moving up for the ith observation.**

**iii. Use the posterior probability for the ith observation in order to predict whether or not the market moves up.**

**iv. Determine whether or not an error was made in predicting the direction for the ith observation. If an error was made, hen indicate this as a 1, and otherwise indicate it as a 0.**

|  |
| --- |
| R codes: |
| >error <- rep(0, dim(Weekly)[1])  >for (i in 1:dim(Weekly)[1]) {  glm.fit <- glm(Direction ~ Lag1 + Lag2, data = Weekly[-i, ], family = "binomial")  pred.up <- predict.glm(glm.fit, Weekly[i, ], type = "response") > 0.5  true.up <- (Weekly[i, ]$Direction == "Up")  if (pred.up != true.up)  error[i] <- 1  }  >error |

The total number of 1 in error is 490 and the total number of errors is 1089.

1. **Take the average of the n numbers obtained in (d)iv in order to obtain the LOOCV estimate for the test error. Comment on the results.**

The LOOCV estimate for the test error rate is 490/1089. (44.9954%.)

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

1. **Based on this data set, provide an estimate for the population mean of medv. Call this estimate ˆμ.**

The estimate for the population mean of medv is 22.5328.

1. **Provide an estimate of the standard error of ˆμ. Interpret this result.**

The estimate for the standard error of mu\_hat is 0.4089.

1. **Now estimate the standard error of ˆμ using the bootstrap. How does this compare to your answer from (b)?**

|  |  |  |
| --- | --- | --- |
| Bootstrap | | |
| Original | Bias | Std.error |
| 22.5328 | 0.0085 | 0.4119 |

Compared to the answer from (b), standard error of mu\_hat of bootstrap is little bit higher. Its value is 0.4119 and the difference between two values is about 0.003. Therefore, the bootstrap standard error of mu\_hat is very close to the answer from (b).

1. **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).**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | CI (Bootstrap) | | CI(ttest) | |
| 95%CI | 21.7090 | 23.3566 | 21.7295 | 23.3361 |

The bootstrap confidence interval is very close to the one provided by the t.test() function

1. **Based on this data set, provide an estimate, ˆμmed, for the median value of medv in the population.**

The median value of medv is 21.2.

1. **We now would like to estimate the standard error of ˆμmed. 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.**

|  |  |  |
| --- | --- | --- |
| Bootstrap | | |
| Original | Bias | Std.error |
| 21.2 | -0.0025 | 0.3744 |

We get an estimated median value of 21.2 which is equal to the value obtained in (e), with a standard error of 0.3744 which is relatively small compared to median value.

1. **Based on this data set, provide an estimate for the tenth percentile of medv in Boston suburbs. Call this quantity ˆμ0.1. (You can use the quantile() function.)**

The estimate for the tenth percentile of medv is 12.75.

1. **Use the bootstrap to estimate the standard error of ˆμ0.1. Comment on your findings.**

|  |  |  |
| --- | --- | --- |
| Bootstrap | | |
| Original | Bias | Std.error |
| 12.75 | 0.0261 | 0.4912 |

We get an estimated tenth percentile value of 12.75 which is again equal to the value obtained in (g), with a standard error of 0.4912 which is relatively small compared to percentile value.

river?

Discussion

We applied various validation approaches to many models. In the Default dataset, we split the observations into the training set and the validation set. Then, we fitted the binomial glm model into the dataset. Comparing the prediction values and the validation set, we calculated the validation error. In the Boston dataset, by using bootstrap method, we could find that the estimate of bootstrap sampling was very close to the estimated value of the population. Therefore, we could conclude that the bootstrap sampling was very correct and powerful.

By investigating the datasets, we found many interesting things. Especially, by using bootstrap sampling, we could understand datasets better. Also, by splitting training set and test dataset, we could calculate the CV errors and learn about sampling methods.

Appendix (R

**R codes**

**# Chaper 5 Lab: Cross-Validation and the Bootstrap**

**# The Validation Set Approach**

**library(ISLR);set.seed(1);train=sample(392,196)**

**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)**

**# 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)**

**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**

**# 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**

**# 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)**

**# Estimating the Accuracy of a Linear Regression Model**

**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,boot.fn,1000)**

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

**#exercises 2,5,7,9**

**##2-----------------------------------------------------------------------**

**###g)---------------------------------------------------------------------**

**n = 1:100000**

**plot(n, 1 - (1 - 1/n)^n, typ = "l", log = "x");abline(h = 1 - exp(-1), lty = "dotted")**

**###h)---------------------------------------------------------------------**

**n = 10000;store = rep(NA, n)**

**for (i in 1:n) {**

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

**}**

**mean(store)**

**##5----------------------------------------------------------------------**

**###a)---------------------------------------------------------------------**

**library(ISLR)**

**set.seed(1)**

**glm.fit = glm(default ~ income + balance, data = Default, family = "binomial");summary(glm.fit)**

**###b)---------------------------------------------------------------------**

**#i:**

**train = sample(dim(Default)[1], dim(Default)[1] / 2)**

**#ii:**

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

**#iii:**

**glm.probs = predict(glm.fit, newdata = Default[-train, ], type="response")**

**glm.pred=rep("No",5000);glm.pred[glm.probs>0.5] = "Yes"**

**#iv:**

**mean(glm.pred != Default[-train, ]$default)**

**###c)---------------------------------------------------------------------**

**simul = function() {**

**train = sample(dim(Default)[1], dim(Default)[1] / 2)**

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

**summary(glm.fit)**

**glm.probs = predict(glm.fit, newdata = Default[-train, ], type="response")**

**glm.pred=rep("No",5000)**

**glm.pred[glm.probs>0.5] = "Yes"**

**return(mean(glm.pred != Default[-train, ]$default))**

**}**

**simul() ; simul() ; simul();**

**###d)---------------------------------------------------------------------**

**train <- sample(dim(Default)[1], dim(Default)[1] / 2)**

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

**pred.glm <- rep("No", length(probs))**

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

**pred.glm[probs > 0.5] <- "Yes"**

**mean(pred.glm != Default[-train, ]$default)**

**##7----------------------------------------------------------------------**

**###a)---------------------------------------------------------------------**

**glm.fit = glm(Direction ~ Lag1 + Lag2, data= Weekly, family = binomial);summary(glm.fit)**

**###b)---------------------------------------------------------------------**

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

**###c)---------------------------------------------------------------------**

**glm.probs = predict(glm.fit, newdata = Weekly[1,], type = "response")**

**glm.pred=ifelse(glm.probs>0.5,"Up","Down");glm.pred**

**Weekly[1,]$Direction**

**###d)---------------------------------------------------------------------**

**error <- rep(0, dim(Weekly)[1])**

**for (i in 1:dim(Weekly)[1]) {**

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

**pred.up <- predict.glm(glm.fit, Weekly[i, ], type = "response") > 0.5**

**true.up <- (Weekly[i, ]$Direction == "Up")**

**if (pred.up != true.up)**

**error[i] <- 1**

**}**

**error**

**###e)---------------------------------------------------------------------**

**mean(error)**

**##9----------------------------------------------------------------------**

**library(MASS);set.seed(1);attach(Boston)**

**###a)---------------------------------------------------------------------**

**mu\_hat=mean(medv);mu\_hat**

**###b)---------------------------------------------------------------------**

**se\_hat = sd(medv) /sqrt(length(medv));se\_hat**

**###c)---------------------------------------------------------------------**

**boot.ftn=function(data, index) return(mean(data[index]))**

**library(boot)**

**bstrap=boot(medv, boot.ftn, 1000);bstrap**

**###d)---------------------------------------------------------------------**

**CI=c(bstrap$t0 - 2 \* 0.4119, bstrap$t0 + 2 \* 0.4119);CI**

**t.test(medv)**

**###e)---------------------------------------------------------------------**

**medv.med = median(medv);medv.med**

**###f)---------------------------------------------------------------------**

**boot.ftn= function(data, index) return(median(data[index]))**

**boot(medv, boot.ftn, 1000)**

**###g)---------------------------------------------------------------------**

**tenth = quantile(medv, c(0.1));tenth**

**###h)---------------------------------------------------------------------**

**boot.ftn= function(data, index) return(quantile(data[index], c(0.1)))**

**boot(medv,boot.ftn, 1000)**