Lab Week 4: Bias, Variance, Cross-Validation

Instructor: Blake Miller

09 February 2021

ISLR Exercise 5.5

```
# In Chapter 4 we used logistic regression ... We will now estimate the test error
# of this logistic regression model using the validation set approach.

library(ISLR)
# # Set Seed so that same sample can be reproduced in future.
set.seed(1)

# (a) Fit a logistic regression model that uses "income"
#     and "balance" to predict "default".
default_fit_m0 <- glm(default ~ income + balance, data = Default,
                      family = binomial)

# (b) Using validation set approach,
#     estimate the test error of this model. (Function based approach)

validation_error_estimate <- function() {

  # i. Split the data into a training and validation set.
  N <- floor(.8*nrow(Default))
  train_idx <- sample(1:nrow(Default), N)
  default_train <- Default[train_idx,]
  default_test <- Default[-train_idx,]

  # ii. Fit a multiple logit regression model on training observations
  default_fit <- glm(default ~ income + balance, data = default_train,
                     family = "binomial")

  # iii. Obtain prediction for each observation in validation
  #      set by computing the posterior probability
  #      and classifying if it is greater than 0.5.

  # "response" gives predicted probabilities rather than log odds
  prob_obs <- predict(default_fit, newdata = default_test, type = "response")

  # Vector of "No"s (dim returns a vector)
  pred_obs <- rep("No", dim(default_test)[1])
  pred_obs[prob_obs > 0.5] <- "Yes"
```

```r
  # iv. Compute the validation set error (fraction
  #     of obs in the validations set that are misclassified)
  return(mean(pred_obs != default_test$default))

}

print("The validation error is:")
```

```
## [1] "The validation error is:"
```

```r
print(validation_error_estimate())
```

```
## [1] 0.026
```

```r
# (c) Repeat the process three times
print("Three addtional estimates of the validation set error give:")
```

```
## [1] "Three addtional estimates of the validation set error give:"
```

```r
print(validation_error_estimate())
```

```
## [1] 0.024
```

```r
print(validation_error_estimate())
```

```
## [1] 0.0265
```

```r
print(validation_error_estimate())
```

```
## [1] 0.0315
```

```r
# (d) Consider a logit model that also includes a dummy variable for student.

validation_error_estimate <- function(){

  # i. Predictors are income, balance, AND student
  N <- floor(.8*nrow(Default))
  train_idx <- sample(1:nrow(Default), N)
  default_train <- Default[train_idx,]
  default_test <- Default[-train_idx,]
  # ii.
  default_fit <- glm(default ~ income +  + student, data = default_train,
                     family = "binomial")
  # iii.
  prob_obs <- predict(default_fit, newdata = default_test, type = "response")
  pred_obs <- rep("No", dim(default_test)[1])
  pred_obs[prob_obs > 0.5] <- "Yes"
  # iv.
  return(mean(pred_obs != default_test$default))
}

print("Adding student as a dummay variable results:")
```

```
## [1] "Adding student as a dummay variable results:"
```

```r
# Doesn't appear the addition of student leads to a reduction of test error.
print(validation_error_estimate())
```

```
## [1] 0.0295
```

ISLR Exercise 5.6

```r
# Compute estimates for the standard error of "income" and "balance"
# logit coefficients (1) using bootstrap (2) using the standard formula
# for computing standard errors in the glm() function

# (a) Determine the estimated SE
set.seed(1)
default_fit = glm(default ~ income + balance, data = Default, family = binomial)
summary(default_fit)
```

```
##
## Call:
## glm(formula = default ~ income + balance, family = binomial,
##     data = Default)
##
## Deviance Residuals:
##     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 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1579.0  on 9997  degrees of freedom
## AIC: 1585
##
## Number of Fisher Scoring iterations: 8
```

```r
# (b) Write a function that outputs the coefficents of the glm model

boot.fn <- function(data, index) return(coef(glm(default ~ income + balance,
    data = data, family = "binomial", subset = index)))

# (c) Use boot() to estimate SE
library(boot)
# 50 bootstrap replications of boot.fn
```

```r
# statistic parameter [2] of boot requires data, vector of indexes to bootstrap.
boot(Default, boot.fn, 50)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Default, statistic = boot.fn, R = 50)
##
##
## Bootstrap Statistics :
##          original        bias     std. error
## t1* -1.154047e+01 -5.661486e-02 4.847786e-01
## t2*  2.080898e-05 -7.436578e-08 4.456965e-06
## t3*  5.647103e-03  1.854126e-05 2.639029e-04
```

ISLR Exercise 5.7

```r
# Compute the LOOCV (Leave-One-Out Cross-Validation) test error estimate.
# Weekly Data:
#   Predicting if the market goes up or down based on the
#   percentage return for previous weeks.

# (a) Fit a logit model to predict "direction"
direction.fit = glm(Direction ~ Lag1 + Lag2, data = Weekly, family = "binomial")
#summary(direction.fit)

# (b) Fit a logit model to predict "direction", using all but the first obs
direction.fit = glm(Direction ~ Lag1 + Lag2, data = Weekly[-1, ], family = binomial)
#summary(direction.fit)

# (c) Use the model from b to predict the direction of the first obs
predict.glm(direction.fit, Weekly[1, ], type = "response") > 0.5
```

```
##    1
## TRUE
```

```r
# WRONG: Prediction was UP, true Direction was DOWN.

# (d) write a loop from 1 to n...

# Vector of 0s same length of data
count = rep(0, dim(Weekly)[1])

# For i in 0 to length of data set
for (i in 1:(dim(Weekly)[1])) {

    # i. Fit a logit model
    glm.fit = glm(Direction ~ Lag1 + Lag2, data = Weekly[-i, ], family = binomial)
```

```r
    # ii. Compute posterior probability for the i-th obs if market goes up.
    is_up = predict.glm(glm.fit, Weekly[i, ], type = "response") > 0.5

    # iii. Determine whether a error was made in predicting the direction
    #      for the i-th obs. If a error was made 1, otherwise 0 (from count)
    is_true_up = Weekly[i, ]$Direction == "Up"
    if (is_up != is_true_up)
        count[i] = 1
}
print("Number of errors:")
```

```
## [1] "Number of errors:"
```

```r
sum(count)
```

```
## [1] 490
```

```r
print("LOOCV estimates a test error rate of:")
```

```
## [1] "LOOCV estimates a test error rate of:"
```

```r
mean(count)
```

```
## [1] 0.4499541
```