Homework 5

Advanced Methods for Data Analysis (36-402)

Due Friday March 12, 2021 *at 3:00 pm ET*

You should **always show all your work** and submit both a writeup and R code.

- Assignments must be submitted through Gradescope as a PDF. Follow the instructions here: `https://www.cmu.edu/teaching/gradescope/`
- Gradescope will ask you to mark which parts of your submission correspond to each homework problem. This is mandatory; if you do not, grading will be slowed down, and your assignment will be penalized.
- Make sure your work is legible in Gradescope. You may not receive credit for work the TAs cannot read. **Note:** If you submit a PDF with pages much larger than 8.5×11", they will be blurry and unreadable in Gradescope.
- For questions involving R code, we strongly recommend using R Markdown. The relevant code should be included with each question, rather than in an appendix. A template Rmd file is provided on Canvas.

1. **Housing Data Revisited Redux.** In Homework 4, you worked with the housing data you used in Homework 2, building additional models and comparing their performance. Now we'll return to those models and compare them using cross-validation. In the problems below, the models we refer to are the ones defined in Homeworks 2 and 4.

   (a) First we will perform 5-fold cross-validation to choose between Models 3–6 as predictors. Combine the original training and test data sets into one data set of size $n = 10605$ by using `housedata <- rbind(housetrain, housetest)`, where `housetrain` and `housetest` are respectively the names of your training and test data sets. For each fold of cross-validation, create the test set by randomly selecting $n_k = 2121$ for each $k = 1, \ldots, 5$. The test folds and corresponding training portions can be created in a manner similar to Demo 3.1:

   ```
   samp <- sample(rep(1:5, 2121), replace = FALSE)
   for (k in 1:5) {
     testd <- housedata[samp == k, ]
     traind <- housedata[!(samp == k), ]
   ```

   The `npreg` function will compute predictions while fitting the model if you run it with a command like

   ```
   model5 <- npreg(Median_house_value ~ Median_household_income +
                   Mean_household_income,
   ```

```
data = traind, newdata = testd,
bws = apply(traind[, c(5,6)], 2, sd) / n^(0.2))
```

This time n is the size of the `traind` data set (8484), which is different from the n in part (d). The predictions will then be in `model5$mean`. For each fold k and each model $m = 3, 4, 5, 6$, report the resulting values of the average squared prediction error within fold k for Model m. You can write a loop to do all the calculations and store the results in a 5×4 matrix.

(b) Compute the average of the five cross-validation error values for each model along with the corresponding standard error as defined in lecture. Comment on which model or models are clearly better than the others. How much better is the best model than the second best compared to the standard errors? Comment on what this says about how good the models are compared to each other.

2. **Optimism and Effective Degrees of Freedom.** In lecture (Lectures 1 and 3), we presented the following result about the relationship between in-sample prediction risk and expected training error:

Let the data consist of $(x_1, Y_1), \ldots, (x_n, Y_n)$, a set of n (predictor, response) pairs with

(i) $Y_i = r(x_i) + \varepsilon_i$ for each i,

(ii) $\varepsilon_1, \ldots, \varepsilon_n$ uncorrelated random variables with mean 0 and common variance σ^2,

(iii) $x_1, \ldots, x_n$ all non-random known values, and

(iv) $r(\cdot)$ a non-random unknown function.

Let $\hat{r}(\cdot)$ be an estimator of $r(\cdot)$ that is determined from the data. Let Y_i' have the same distribution as Y_i for each i with $Y_1', \ldots, Y_n'$ independent of $Y_1, \ldots, Y_n$. Then

$$\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\{Y_i' - \hat{r}(x_i)\}^2\right] - \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\{Y_i - \hat{r}(x_i)\}^2\right] = \frac{2}{n}\sum_{i=1}^{n}\mathrm{Cov}(Y_i, \hat{r}(x_i)). \qquad (1)$$

Prove that (1) is true. You may want to review some of the lectures where prediction risk was discussed, especially the associated hand-written notes.

3. **The Parametric Bootstrap in Regression.** The file `parametric-bootstrap.csv` contains 100 observations of two variables, X and Y. These variables are linearly related, but one of the simple linear regression assumptions is not met. (See Shalizi section 2.3 for more on the simple linear regression assumptions.) Specifically, while the residuals are normally distributed, their variance is not constant in x:

$$\begin{aligned} Y = r(X) + \varepsilon &= \beta_0 + \beta_1 X + \varepsilon \\ \mathbb{E}[\varepsilon | X = x] &= 0 \text{ for all } x \\ \mathrm{Var}(\varepsilon | X = x) &= x^2 + 1. \end{aligned}$$

In this problem, we'll explore the parametric bootstrap and how it helps us estimate uncertainty in our fitted model $\hat{r}(x)$ when standard assumptions are not met.

(a) First, use R to fit a linear regression model to the data. Plot the residuals and other diagnostics, and indicate which plots would have revealed the assumption violation if you didn't know it was present.

(b) A scientist wants your best estimate of the mean of Y when $X = 20$. That is, you are asked to estimate $r(20) = \mathbb{E}[Y|X = 20]$. Using the `predict` function and its `se.fit` argument, make an estimate and give its standard error. What assumptions must be true for that standard error to be accurate, and are they all met?

(c) As you may recall from courses like 36-226, the standard error is an estimate of the width of the *sampling distribution* of your estimate. That is, if we were to repeatedly obtain new samples from the population, fit a linear model to them, and use it to estimate $r(20) = \mathbb{E}[Y|X = 20]$, then calculate the standard deviation of thousands of those estimates, that is the standard error. Using mathematical assumptions, we can calculate a standard error without needing repeated samples from the population.

But when those mathematical assumptions are not met, the parametric bootstrap can help us estimate the standard error instead. As with all bootstrap methods, it involves *resampling*, i.e. creating new datasets and refitting the model to see how the estimates vary. All bootstrapping methods attempt to approximate what would happen if we could get many independent samples from the population; the parametric bootstrap does so by assuming a specific model for the population distribution.

In our case, we are interested in the uncertainty in our estimate for $r(20)$. To obtain each bootstrap sample, we condition on (fix) X to the values in our original sample, and we draw new Y values from the population distribution—or, rather, what we *estimate* to be the population distribution.

Create a function in R that can simulate new Y values from the estimated population distribution. That is, the function should use the slope and intercept you obtained in part (a) to draw

$$Y = \hat{\beta}_0 + \hat{\beta}_1 X + \varepsilon,$$

where ε is drawn from a normal distribution with the variance we gave at the start of the problem. The output of the function should be a data frame of the same size as the one contained in `parametric-bootstrap.csv`, containing an X column (the original X values) and a Y column (the bootstrapped Y values).

(d) Using the function you wrote in part (c), conduct a parametric bootstrap. That write a loop that runs $B = 1000$, and on each loop iteration:

- Call your function from part (c) to get a new bootstrapped dataset
- Fit a linear model to the bootstrapped data
- Use the model to estimate $r(20)$
- Store the estimate in a vector

At the end of the loop, report the standard deviation of your estimates of $r(20)$ as your new estimated standard error. How does the estimate differ from the one you got from `predict`?