Homework 1: Bias, Variance, & Resampling

MACS 30100: Perspectives on Computational Modeling
University of Chicago

Ari Weil

1/26/2021

Overview

For each of the following prompts, produce responses *with* code in-line. While you are encouraged to stage and draft your problem set solutions using any files, code, and data you'd like within the private repo for the assignment, *only the final, rendered PDF with responses and code in-line will be graded.*

Bias & Variance

1. (10 points) Consider the following eight plots based on model fits, assuming the data generating process,

$$y = x^3 - 2x^2 + 1.5x + \epsilon.$$

Specifically, the figure shows two different fits of a model (one for each row) on each of the four samples (one for each column). So, e.g., column 1 shows two versions of a model fit to the same sample of 100 observations, which were drawn at random from the data generating process in the equation above. *Describe the difference between the two models in terms of complexity, bias, and variance. Responses should be at least a few sentences.*

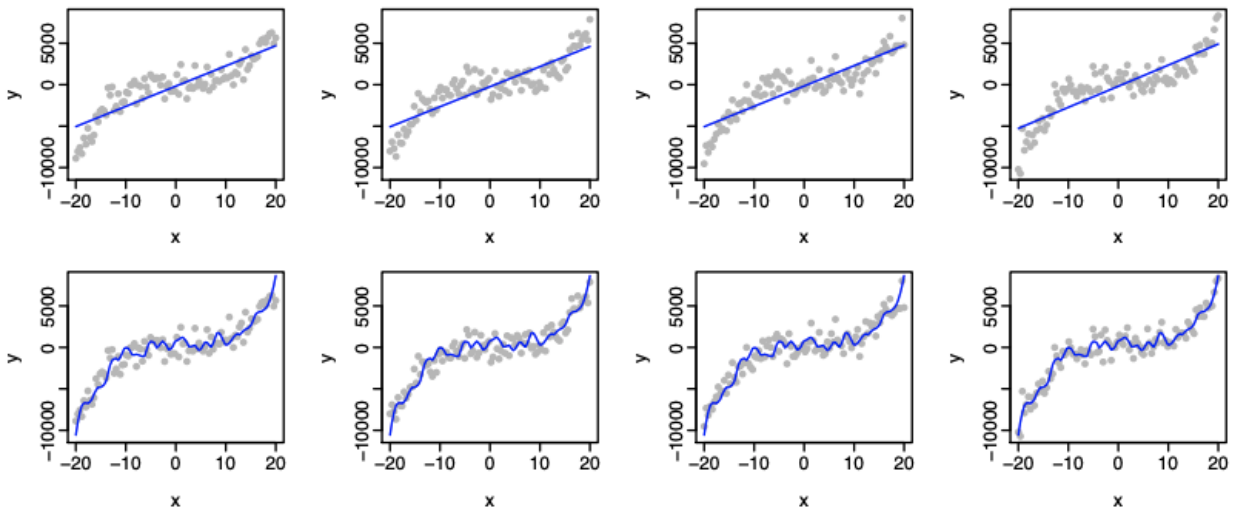


Figure 1: Two models; Four Random Samples

- Here we compare a linear model (row 1) to what looks like a LOESS model (row 2). The linear model is less flexible, less complex, and more biased than the other model. The data generating process is not linear, but rather a polynomial function, and so there is bias introduced in trying to represent that relationship with a strict linear relationship. The second row model is more flexible–it curves to match a lot of the observations in the data. It is also less biased–it more clearly represents the complex data without reducing to a linear relationship. However, with this flexibility comes increased variance. Variance refers to how much the predicted values ($\hat{f}$) change when using new training data. Here, the linear model has low variance and its predictions don't change much over the four sets. In contrast, the more flexible lower model is designed to closely match the observations, and so its predicted values can change much more between training sets.

2. (10 points) Building on the previous question and considering the following figure from ch. 2 of the ISL book, think about training and test error moving from a less flexible model towards more a more flexible model. Specifically, the figure is becoming more "flexible" as the number of neighbors, k, decreases, thereby picking up more local behavior. We haven't yet covered kNN or other supervised classifiers, but the logic of the figure should be apparent, where the level of flexibility of a model will directly influence training and testing error. *Explain why these two curves have the shapes they do. Responses should be at least a few sentences.*

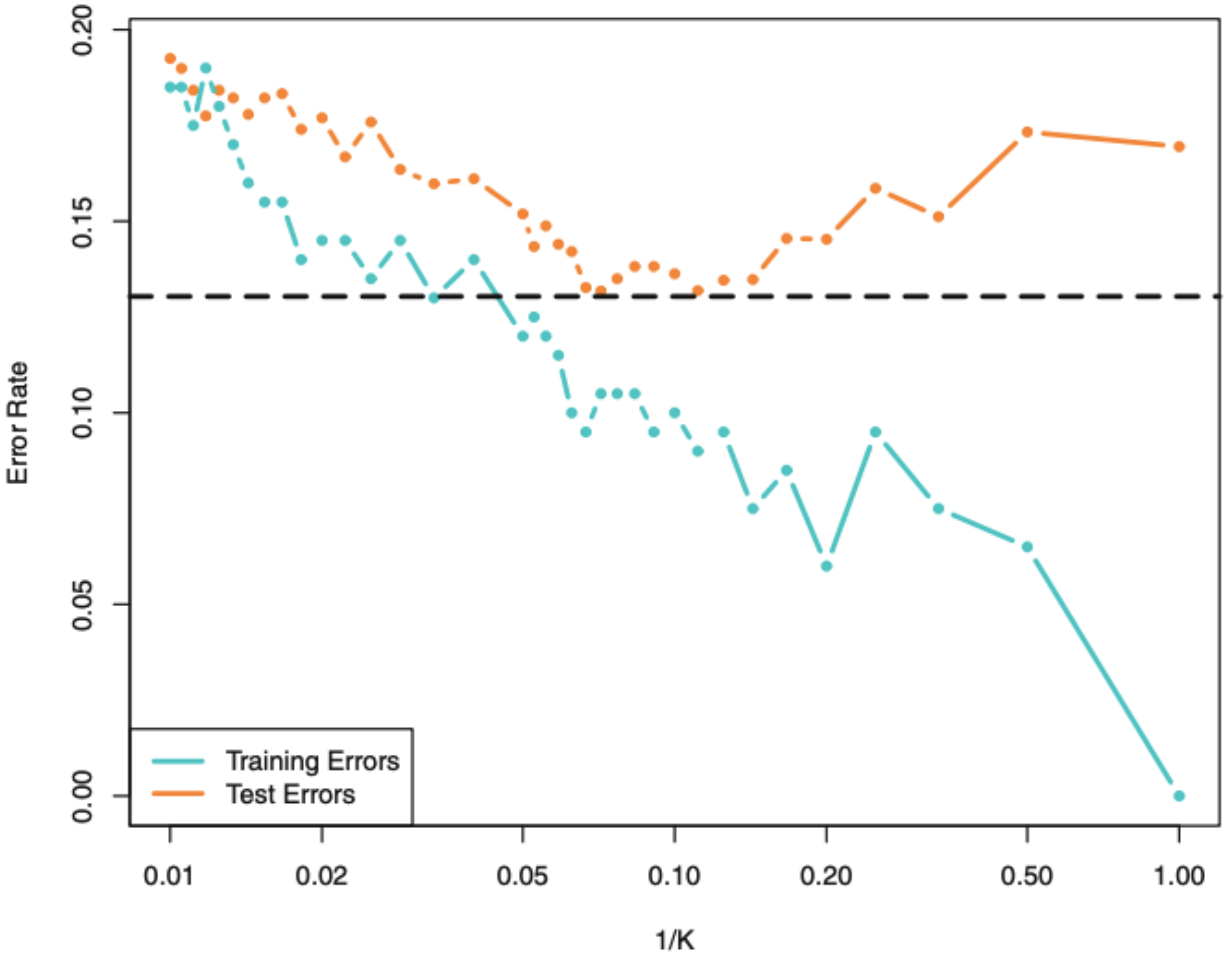


Figure 2: Figure 2.17 from ISL

- The main takeaway from this graph is that the curves diverge as they approach K = 1 and the model becomes more flexible. The divergence is that training error decreases while test error increases. With a more flexible model, it will begin to fit the training data perfectly. But this overfitting means that the model performs well on the training set but classifies the test set poorly.
- The training error overall has a straightforward trajectory: the closer to K=1, the lower the training error. The model is fitting closer and closer to the points in the training set. The test error, on the other hand, has a more complex trajectory: a bit of a U-shape. As the model gets more flexible, there are improvements to both training and test error. But after 1/K = 0.1, the test error rate begins to increase. This is because the model is fitting too tightly to the training set, and thus has more variance and fits new data (test set) worse.
- This divergence suggests that, in this case, the researcher could try different number of neighbors to find the ideal balance of train and test error rate. Here, around 1/K = 0.10 might be a good tradeoff in minimizing both error rates, because that is the lowest the test error gets before it begins to increase again.

3. (10 points) When the sample size, n, is very large, and the number of predictors, p, is small, would we expect the performance of a flexible model to be better or worse than an inflexible method? Justify your answer.

- A more flexible model would perform better here. While flexible methods risk overfitting the training data, here there is a very large sample, so there is a lot of data to train on, and methods such as cross-validation can reduce the error rate. Thus, we would chose a more flexible model.

4. (10 points) When the number of predictors, p is very large, and the sample size, n, is small, would we expect the performance of a flexible model to be better or worse than an inflexible method? Justify your answer.

    - A more inflexible model would perform better here. With a small sample size, there isn't much data to train the data on. So using a very flexible model would risk overfitting to the small training data and producing a high test error rate.

5. (10 points) When the relationship between the predictors, $\mathbf{X}$, and response, y, is highly non-linear, would we expect the performance of a flexible model to be better or worse than an inflexible method? Justify your answer.

    - A more flexible model will be better for a highly non-linear relationship. An inflexible model, such as a linear model, might try to fit a relationship that differs from the underlying relationship. A flexible model will fit the complex non-linear relationship better.

6. (10 points) Why can minimizing the training mean squared error (MSE) lead to overfitting? *Responses should be at least a few sentences.*

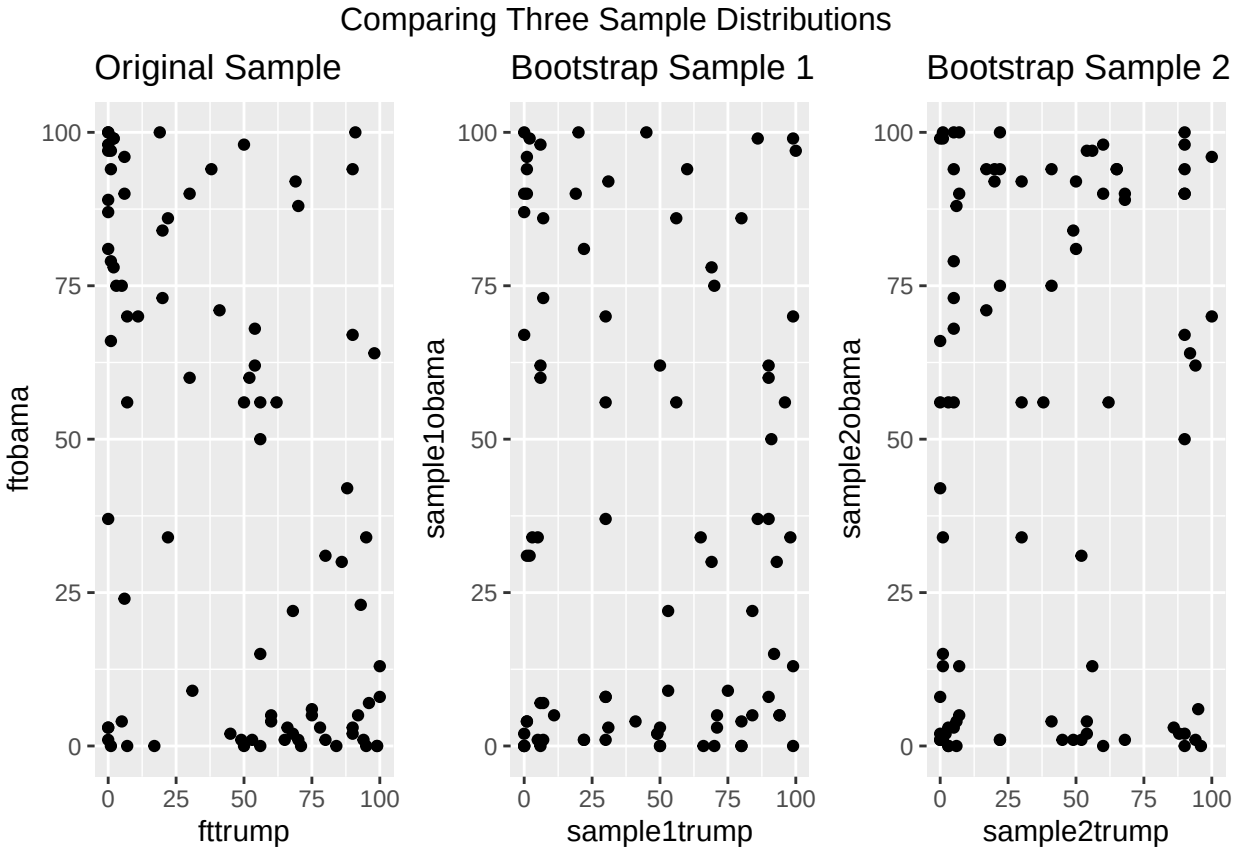
    - Mean squared error is a measure of the difference between predicted and actual values. Minimizing training MSE can lead to overfitting because this fits the model very well to the patterns of the initial training set. But this makes the model less able to properly fit new data. So you can end up with a model fitting very close to the training set with a low training MSE, but that then has a high test MSE, because it is not finding the same patterns in the test set. This means there is too much variance–the measure of how well the model can be applied to new data. In this case, minimizing training MSE means that when applied to the test set the difference between the series of y_i and $\hat{y}_i$ is large (the model begins mispredicting more often on outside data if it is fitted too close to the training set).

7. (10 points) Recall bootstrapping involves a process of drawing random samples of size n through sampling *with* replacement. Create and plot two bootstrapped samples manually (i.e., *not using a function like* `boot()`). For reference, also plot the original data set to compare distributions. *Note:* When loading the data, you may simply drop the `NA`s, rather than impute, for ease.

8. (10 points) Recall bootstrapping involves a process of drawing random samples of size n through sampling with replacement. Using the 2016 ANES data we've used a few times already, create and plot two bootstrapped samples manually (i.e., not using a function like boot()). For reference, also plot the original data set to compare distributions of feelings toward Trump (fttrump) versus feelings toward Obama (ftobama). Note: When loading the ANES data, you may simply drop the NAs, rather than impute, for ease (though in practice, this strategy isn't recommended).

```
library(tidyverse)
library(here)
library(ggplot2)
library(gridExtra)

# Read in data
anes <- read_csv(here("data", "anes_pilot_2016.csv"))

# Set seed for reproducibility
set.seed(1234)

# Clean by dropping NAs
anes_boot <- anes %>%
  drop_na() %>%
  select(fttrump, ftobama)

# Plot trump vs obama
plot1 <- ggplot(anes_boot, aes(x = fttrump, y = ftobama)) +
  geom_point() +
  ggtitle("Original Sample")

# Now create two boostrapped samples by sampling with replacement
## First sample
sample1obama <- sample(anes_boot$ftobama, size=nrow(anes_boot), replace=TRUE)
sample1trump <- sample(anes_boot$fttrump, size=nrow(anes_boot), replace=TRUE)
sample1 <- data.frame(sample1obama, sample1trump)

## Second sample
sample2obama <- sample(anes_boot$ftobama, size=nrow(anes_boot), replace=TRUE)
sample2trump <- sample(anes_boot$fttrump, size=nrow(anes_boot), replace=TRUE)
sample2 <- data.frame(sample1obama, sample1trump)

# Create plots and compare
plot2 <- ggplot(sample1, aes(x = sample1trump, y = sample1obama)) +
  geom_point() +
  ggtitle('Bootstrap Sample 1')

plot3 <- ggplot(sample2, aes(x = sample2trump, y = sample2obama)) +
  geom_point() +
  ggtitle("Bootstrap Sample 2")

grid.arrange(plot1, plot2, plot3, nrow = 1,
```

```
        top = "Comparing Three Sample Distributions")
```



8. (5 points) Discuss the distributions from the previous question. Do they look mostly similar as is expected by sampling with replacement? Why or why not, do you think? *Respond with a few sentences.*

    - The samples look fairly similar. There is a cluster at the bottom of each panel, in the bottom left where respondents think poorly of both and the bottom right where respondents think poorly of Obama and highly of Trump. In the original, there is a cluster in the top left of people who really like Obama and dislike Trump. This is not as prominent in bootstrap sample 1, but is there in the second boostrap. But overall, they are similar plots. This what is expected with sampling with replacement, where the mean of the samples should equal the mean of the population,and the standard deviations should also match because the draws are independent.

9. (15 points) The median for feelings toward Obama (`ftobama`) is 39.5. Using a package or any function/method you'd like, bootstrap the standard error of our statistic of interest (which is the median in this case) based on 1000 draws from the data. You might consider writing a simple helper function to speed along the bootstrapping process, but this is up to you of course. Then, construct the 95% confidence interval around your bootstrapped estimate. Report your results and offer a few points of discussion. *Respond with a few sentences.*

    - The median of feelings towards Obama is 39.5. When I boostrap the standard error over 1000 draws, the standard error of this estimate is 13.522. The 95% confidence interval is (10.65, 63.66). This confidence interval is roughly $median \pm 2 \cdot standarderror$. We interpret this interval as saying that 95% of confidence intervals we draw from the samples will contain the true value. I was surprised by how large this standard error is, but I believe it is because this is such a small

sample of data (88 observations of ftobama) and it is skewed. The histogram below shows a large cluster of people with low approval of Obama and then a group from 70-100% approval. This bimodal split creates a large dispersion of the data, which is reflected in the large standard error and quite wide confidence interval.

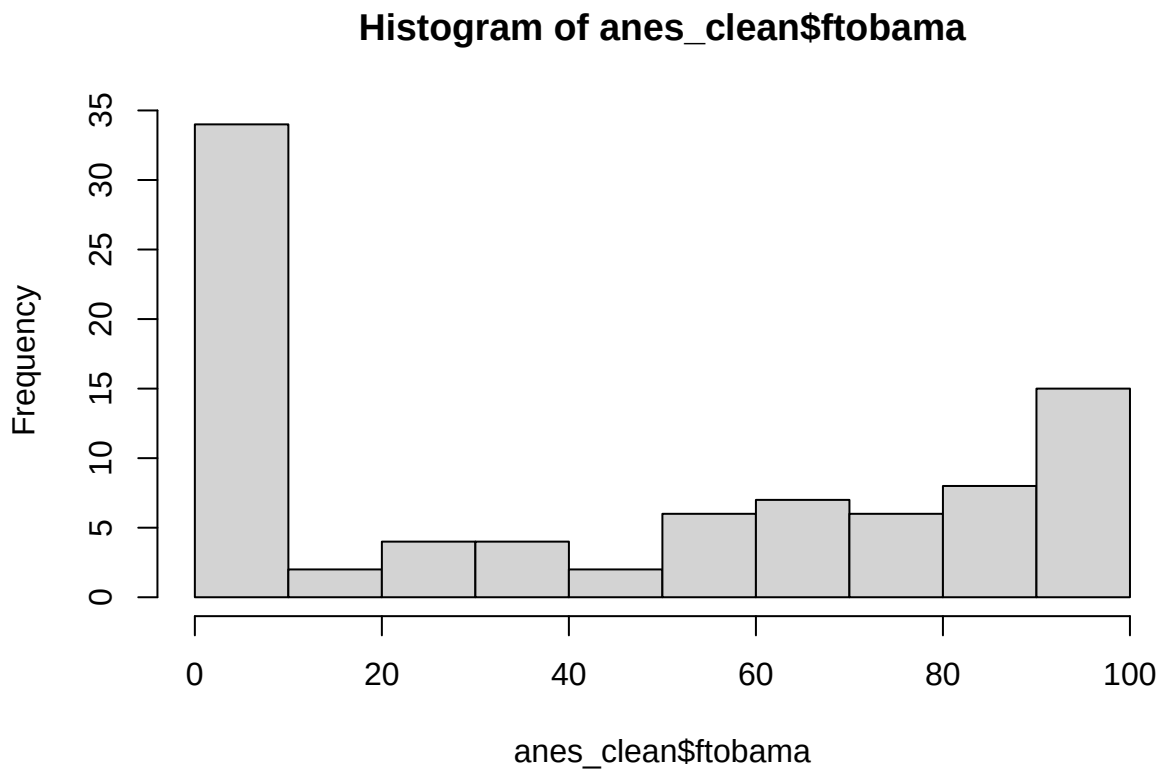
```
library(boot)

# Read in the data
anes <- read_csv(here("data", "anes_pilot_2016.csv"))

# Clean data
# Drop all missing observations
anes_clean <- anes %>% drop_na()

# Check Obama column
summary(anes_clean$ftobama)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    3.00   39.50   43.76   81.75  100.00
```

```
length(anes_clean$ftobama)
```

```
## [1] 88
```

```
# Visualize distribution of the Obama sample
hist(anes_clean$ftobama)
```



```r
# Helper function that returns the median of ftobama
med <- function(data, i) {
  median(data[i], na.rm = T)
}

# Set seed for reproducibility
set.seed(1234)

# Use boot to take draws and call helper function to return the median
boot_med <- boot(data = anes_clean$ftobama, statistic = med, R = 1000)

# Examine the bootstrap result
boot_med
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = anes_clean$ftobama, statistic = med, R = 1000)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*     39.5   2.347    13.52237
```

```r
# Calculate the 95% confidence interval for the median
boot.ci(boot_med)
```

```
## Warning in boot.ci(boot_med): bootstrap variances needed for studentized
## intervals
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot_med)
##
## Intervals :
## Level      Normal              Basic
## 95%   (10.65, 63.66 )   (17.00, 64.00 )
##
## Level     Percentile            BCa
## 95%   (15, 62 )   (13, 61 )
## Calculations and Intervals on Original Scale
```

10. (10 points) How are bootstrapping and cross-validation approaches to resampling different? How are they similar? Why does any of this matter from both social science and computational modeling perspectives?

    - Bootstrapping and cross-validation are both resampling approaches. One key difference is in sampling method: bootstrapping does sampling with replacement while cross-validation does not.

This creates a few differences: 1) bootstrapped samples are the same size as the original while CV are not, 2) bootstrapped samples can have the same observation multiple times in a sample while CV cannot. Cross-validation also has a range of bias levels within its different types, with LOOCV (where K = N) being the most unbiased, but 5- or 10-fold having less variance in exchange with a little more bias.

- Both measures matter a lot for social science. While the machine learning and industry world have made validation a regular feature, some parts of social science are still not using these methods. In my field of political science, a lot of major papers still just fit a model once without training and testing separately. Fitting once is effectively only fitting on training data, so they're not able to say how the model would perform on an outside set of data.
- In terms of application, each has different best uses. For example, bootstrapping is particularly useful for getting a sense of uncertainty around estimates, so it used for estimating standard errors around parameters of interest. CV is often used to estimate the test MSE of the model.