CSE 546 SPRING 2021: HW1

Name: Honda Li

Bias-Variance tradeoff

B.1.a

When m is relative small (Relative to the observed samples), the variance will be huge, and when the value of m is huge, then the bias will be huge.

When $m = 1$, the function $\hat{f}(x)$ got decay into a k-nn method that checks of the closest labor. The model is literally looking for the $\hat{y}$ by looking for x_i that is closest to the test sample. This will produce a great amount of variance, and the more sample there is, the more variance we have for the estimated model.

When $m = n$, we have $\hat{f}(x)$ returning the average of the samples, regardless of what the input is. This means that as we have more and more sample, the average will converge. Meaning all estimated models converge to one model. But the bias is not reduced because some ground truths might not be a horizontal line.

B.1.b

We defined $\bar{f}^{(j)} = \frac{1}{m}\sum_{i=(j-1)m+1}^{jm} f(x_i)$, which is just the average of the ground truth function $f(x)$ over the j th partition.

Objective: figure out the Bias-squared as: $\frac{1}{n}\sum_{i=1}^{n}\left(\mathbb{E}\left[\hat{f}_m(x_i)\right] - f(x_i)\right)^2$.

Let's start by considering the quantity: $\mathbb{E}\left[\hat{f}_m(x_i)\right]$.

$$\begin{aligned}\mathbb{E}\left[\hat{f}_m(x_i)\right] &\underset{(1)}{=} \mathbb{E}\left[c_{\lceil \frac{i}{m} \rceil}\right] \\ &= \frac{1}{m}\sum_{i=(j-1)m+1}^{jm} f(x_i) \quad \text{where } j = \left\lceil \frac{i}{m} \right\rceil \\ &= \hat{f}^{(\lceil \frac{i}{m} \rceil)}\end{aligned} \tag{B.1.b.1}$$

(1): Because, for any x_i, it will only fall onto one of the partition, and the partition is $\lceil \frac{i}{m} \rceil$. This true because $x_i = \frac{i}{n}$ and $i \in \mathbb{N}$. So the index of the sample tells us which interval it's falling onto and what the value of j is going to be for that sample x_i. Hence, it set all other c_j to zeroes, except when $j = \lceil \frac{i}{m} \rceil$

Now, we consider the Bias-squared in this way:

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{n}\left(\mathbb{E}\left[\hat{f}_m(x_i)\right] - f(x_i)\right)^2 &= \frac{1}{n}\sum_{i=1}^{n}\left(\bar{f}^{\lceil \frac{i}{m} \rceil} - f(x_i)\right)^2 \\ &\underset{(2)}{=} \frac{1}{n}\sum_{j=1}^{n/m}\sum_{i=(j-1)m-1}^{jm}\left(\bar{f}^{(j)} - f(x_i)\right)^2\end{aligned} \tag{B.1.b.2}$$

(2): This is true because, there is a group of indices i that is going to fall under the j th groups, and those indices are in the range of $i \in \mathbb{N} \cap [(j-1)m - 1, jm]$ and for all such index i, the value of $\bar{f}^{\lceil \frac{i}{m} \rceil}$ is going to be the same. Because we rounded the fraction $\frac{i}{m}$.

B.1.c

Let's dive into the math starting with the given definition of average variance in the problem statement, which is:

$$\begin{aligned}
&\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\left(\hat{f}_m(x_i)-\mathbb{E}\left[\hat{f}_m(x_i)\right]\right)^2\right] \\
=&\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\left[\left(\hat{f}_m(x_i)-\mathbb{E}\left[\hat{f}_m(x_i)\right]\right)^2\right] \\
=&\frac{1}{n}\sum_{j=1}^{n/m}\sum_{i=(j-1)m+1}^{mj}\mathbb{E}\left[\left(\hat{f}_m(x_i)-\mathbb{E}\left[\hat{f}_m(x_i)\right]\right)^2\right]
\end{aligned} \tag{B.1.c.1}$$

Let's pause for a moment and think about the fact that, the outter sum is summing over each of the partition. And we know that for all $i \in \mathbb{N} \cap [(j-1)m+1, jm]$, which is just all the indices for the sample that are presented in the j th partition, the value of $\hat{f}_m(x_i)$ and the value of $\mathbb{E}\left[\hat{f}_m(x_i)\right]$ is going to be the same and they are: c_j and $\bar{f}^{(j)}$.

$$\begin{aligned}
\frac{1}{n}\sum_{j=1}^{n/m}\sum_{i=(j-1)m+1}^{mj}\mathbb{E}\left[\left(\hat{f}_m(x_i)-\mathbb{E}\left[\hat{f}_m(x_i)\right]\right)^2\right] =&\frac{1}{n}\sum_{j=1}^{n/m}\sum_{i=(j-1)m+1}^{mj}\mathbb{E}\left[\left(c_j-\bar{f}^{(j)}\right)^2\right] \\
=&\frac{1}{n}\sum_{j=1}^{n/m}m\mathbb{E}\left[(c_j-\bar{f}^{(j)})^2\right]
\end{aligned} \tag{B.1.c.2}$$

And hence, we have proven the first equality stated in the problem. Next, let's observe the fact that c_j is the average of all the samples over the j th partition which is given as $\frac{1}{m}\sum_{i=(j-1)m+1}^{jm} y_i$ and in this case $y_i = f(x_i) + \epsilon$, but at the same time $\bar{f}^j = \frac{1}{m}\sum_{i=(j-1)m+1}^{mj} f(x_i)$ and hence we know that: $c_j - \bar{f}^{(j)} = \frac{1}{m}\sum_{i=(j-1)m+1}^{jm}\epsilon_i$, which is conveniently giving us the result that:

$$\begin{aligned}
\frac{1}{n}\sum_{j=1}^{n/m}m\mathbb{E}\left[(c_j-\bar{f}^{(j)})^2\right] &= \frac{1}{n}\frac{n}{m}m\mathbb{E}\left[\left(\frac{1}{m}\sum_{i=(j-1)m+1}^{jm}\epsilon_i\right)^2\right] \\
&= \mathbb{E}\left[\left(\frac{1}{m}\sum_{i=1}^{m}\epsilon_i\right)^2\right]
\end{aligned} \tag{B.1.c.3}$$

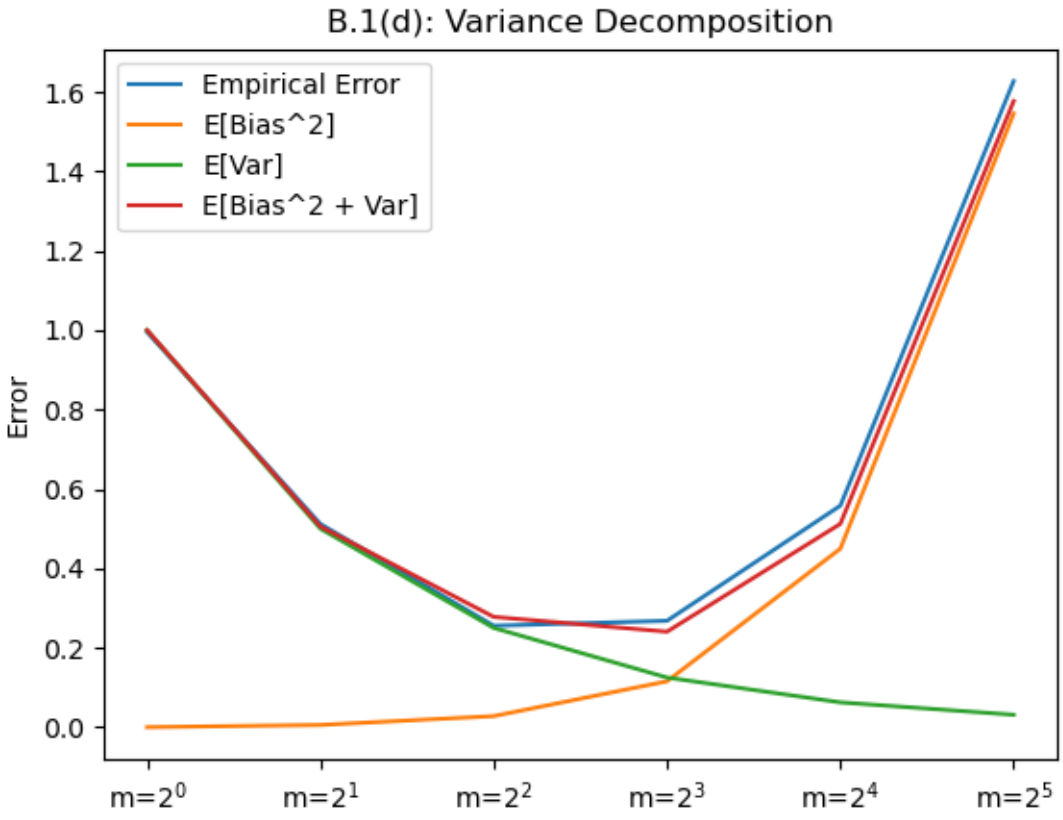
Let's pause for a moment and remember that $\epsilon \sim \mathcal{N}(0, \sigma^2)$, and in this case, we take the average for m of them, hence, $\frac{1}{m}\sum_{i=`'}^{m}\epsilon_i \sim \mathcal{N}(0, \sigma^2/m)$.

However, we also need to note that for any Gaussian Random variable say: $X \sim \mathcal{N}(0, \sigma^2)$, the variance $\sigma^2 = \mathbb{E}\left[X^2\right] - \mathbb{E}\left[X\right]^2$(The mean is zero!) which implies that $\mathbb{E}\left[X^2\right] = \sigma^2$. Hence, combining this fact and the previous statement (Basically $X = \frac{1}{m}\sum_{i=1}^{m}\epsilon_i$), we know that:

$$\mathbb{E}\left[\left(\frac{1}{m}\sum_{i=1}^{m}\epsilon_i\right)^2\right] = \frac{\sigma^2}{m} \tag{B.1.c.4}$$

B.1.d

Programming Part. Here is the Plot I got:



And this is the code I used to generate the graph:

```
# Code is for B1(d), HW1, CSE 546.
import numpy as np
sin = np.sin
cos = np.cos
array = np.array
PI = np.pi
rnd = np.random.normal
arange = np.arange

import math
ceil = math.ceil
zeros = np.zeros
mean = np.mean

import matplotlib.pyplot as plt
scatter = plt.scatter
plot = plt.plot
show = plt.show
xticks = plt.xticks
title = plt.title
legend = plt.legend
save = plt.savefig
ylabel = plt.ylabel

def f(x):
    return 4*sin(x*PI)*cos(6*PI*x)


def Epsilon(length):
```

```
    return rnd(0, 1, length)


def GenerateData(n=256):
    Xgrid = array([(II + 1)/n for II in range(n)])
    return Xgrid, f(Xgrid) + Epsilon(n)


def yHat(data, m, n=256):
    """
        Given the data, fit it with average data points on each of the partition.
    Args:
        data:
        m:
        n:
    Returns:
        Predicted value
    """
    y = zeros(n)
    for JJ in range(int(n/m)):
        UpperBound = min((JJ + 1)*m, n)
        LowerBound = JJ*m
        y[LowerBound: UpperBound] = mean(data[LowerBound: UpperBound])
    return y


def fBar(xgrid, m, n=256):
    return yHat(f(xgrid), m, n)


def AvgBiasSqaured(m, n=256):
    """
        The expected vaue of the biases error squared. Because it's expected value,
        this will use the underlying generative model instead of using data to get
        the error from the biases squared.

    Args:
        data:

    Returns:

    """
    Xgrid = array([(II + 1)/n for II in range(n)])
    FBar = fBar(Xgrid, m, n)
    F = f(Xgrid)
    return mean((FBar - F)**2)


def AvgVariance(m):
    return 1/m


def AvgEmpiricalErr(yhat, n=256):
    Xgrid = array([(II + 1)/n for II in range(n)])
    return mean((f(Xgrid) - yhat)**2)


def main():
    def FitDemo():
        Xs, Ys = GenerateData()
        scatter(Xs, Ys)
        Yhat = yHat(Ys, 8)
        YBar = fBar(Xs, 8)
        plot(Xs, Yhat, c="red")
        plot(Xs, YBar, c="green")
        show()
    FitDemo()
    def PlotErrors():
```

```
        Error1 = [] # Empirical error from 256 random samples.
        Error2 = [] # Expected Bias Squared
        Error3 = [] # Expcted Variance Square
        Error4 = [] # Expected Errors
        _, SampledData = GenerateData()
        for m in [2**II for II in range(6)]:
            Error1.append(AvgEmpiricalErr(yHat(SampledData, m)))
            Error2.append(AvgBiasSqaured(m))
            Error3.append(AvgVariance(m))
            Error4.append(Error2[-1] + Error3[-1])
        plot(Error1)
        plot(Error2)
        plot(Error3)
        plot(Error4)
        xticks(range(6), [f"m=$2^{II}$" for II in range(6)])
        ylabel("Error")
        legend(["Empirical␣Error", "E[Bias^2]", "E[Var]", "E[Bias^2␣+␣Var]"])
        title("B.1(d):␣Variance␣Decomposition")
        save("B.1(d)plot.png")

    PlotErrors()


if __name__ == "__main__":
    import os
    print(f"wd:␣{os.getcwd()}")
    print(f"script␣running␣on{os.curdir}")
    main()
```

B.1.e