ISYE 7406 Homework 4

October 10, 2021

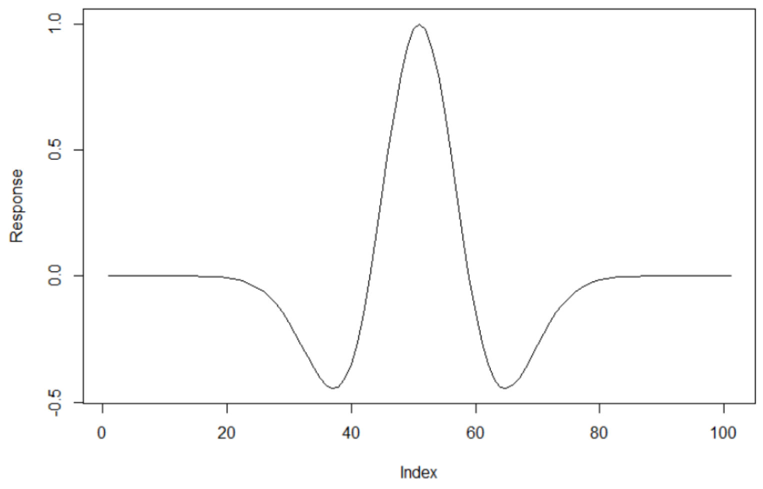
Introduction

This report will review the LOESS, Nadaraya-Watson (NW) kernel smoothing, and Spline smoothing statistical properties and the challenges they face. The purpose of analyzing the statistical properties is to generate models (LOESS, Nadaraya-Watson (NW) kernel smoothing, and Spline smoothing) to identify the estimation challenges through comparison plotting. Then we compare the Bias, Variance, and Mean Squared Error (MSE) to determine which method performed best. We also provide an analysis on whether it's a fair comparison between the three models. Lastly, we repeat the same procedure shown above but with another design that has non-equidistant points. The purpose of this report is to also identify other findings and visually explore the dataset for important/unusual patterns.

Exploratory Data Analysis

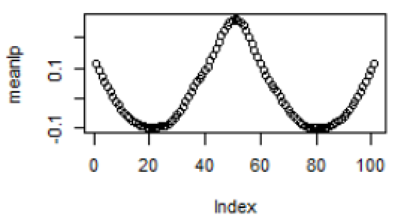
The exploratory data analysis will check for important/unusual patterns in the data through plots and summary statistics. According to the summary statistics on the x formula ***2*pi*seq(-1, 1, length=n)*** which is the equidistant points in [-2π, 2π] shown below, it was interesting to see how the Median and Mean equaled 0.000 (technically 3.55271E-14) and the 1st & 3rd quantile equaled (+/- 3.142) suggesting that most of the points fall in the middle of the distribution. When running the true Mexican hat function on these 101 x values, it was interesting to see how the function did look like a Mexican hat (known as a sombrero).

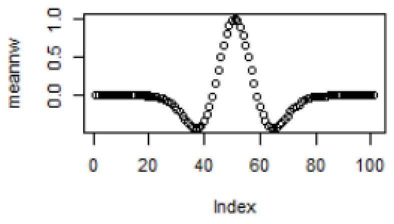
```
> summary(x)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 -6.283  -3.142   0.000   0.000   3.142   6.283 
```

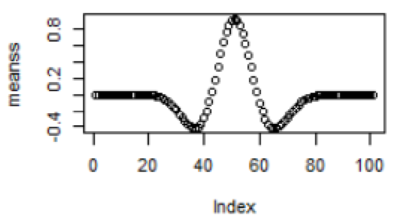


Methodology – Modeling for Training and Test errors

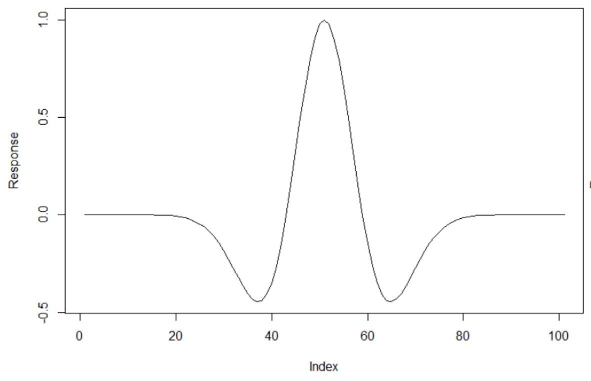
I performed 1000 Monte Carlo runs and generated a data set in the form of (xi,Yi) with xi being the Mexican hat function of x. For each run, I computed the LOESS (with span = 0.75), Nadaraya-Watson kernel smoothing with "normal" kernel and bandwidth =0.2 and Spline smoothing with the default tuning parameter. In the office hours (TA was Yuyang Shi), it was confirmed that we can use the "normal" kernel instead of Gaussian because the ksmooth function doesn't have a Gaussian option. I then applied the mean of each method to retrieve the plots below for LOESS, Nadaraya-Watson, and Spline smoothing respectively. Before comparing the plots with the x values, I wanted to see how they looked in comparison with the true Mexican hat plot. You can see that both the Nadaraya-Watson and Spline smoothing look very similar to the true Mexican hat function while the simpler model (LOESS) did not perform very well.

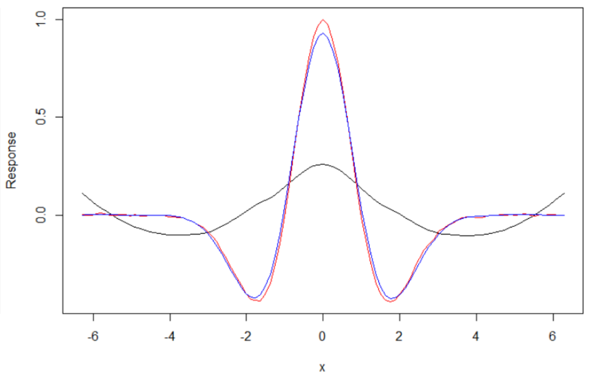






Now when I plotted all three models together against the x values (right plot), you can see again how the Nadaraya-Watson (red line) and Spline smoothing (blue line) perform a lot better compared to LOESS (black line). I added the true Mexican hat plot (left plot) for easy visual comparison. It's very distinct but you can see how the Nadaraya-Watson performed a little more accurate on the top of the hill compared to the Spline smoothing.





We dissect this further since using a birds-eye view is not appropriate for this analysis by plotting the Bias, Variance, and MSE of the models. But before we get into the plots, I think it's important to clarify the characterizations of the terms above.

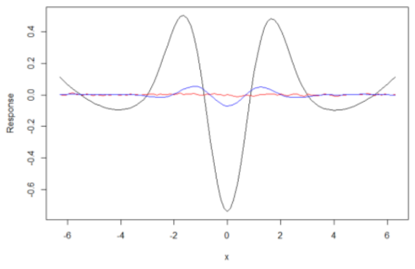
The **Bias** is the quantity that the true measure varies from the model's estimate based on the training data. A high bias will cause the model to underfit which according to Mastersindatascience.org "occurs when the algorithm is unable to capture relevant relations between features and target outputs and typically includes more assumptions about the target function or end result. The simpler the algorithm, the more bias it has likely introduced."

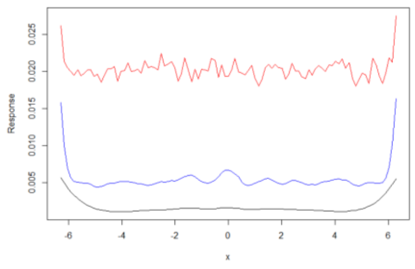
The **Variance** specifies the true value estimate and how much it will differ when a different training set is used however it should not be confused with being the measure of overall accuracy (that's where MSE comes in). According to Mastersindatascience.org, "A model with high-level variance may reflect random noise in the training data set instead of the target function and will result in significant changes to the projections of the target function.. A model with low variance means sampled data is close to where the model predicted it would be." Basically, a high variance can cause overfitting when the training data fluctuations are exaggerated.

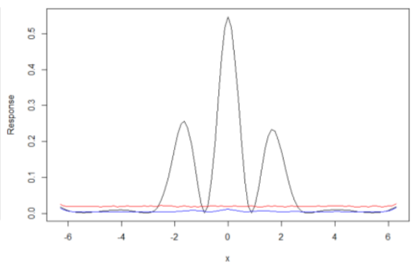
The Mean Squared Error (**MSE**) as the professor stated in his office hours represents the testing error of the model and this is usually the one that we want to look at for the model accuracy. However, we should keep in mind that the different procedures we use whether it's changing from equidistant to non-equidistant points or changing the parameters of the models will result in different MSE's. We must realize that while we aim for a low bias and variance, the relationship between them is a trade-off meaning when bias goes up, variance goes down and vice versa. This means it is up to us to decide where the trade-off should be which is usually based on the type of model and of course the lowest MSE while trying to balance the model's bias and variance to not under/over fit the data.

Results & Findings

Now that we understand the goal, we can provide analysis on the plots shown below on the Bias, Variance, and MSE respectively. In the Bias (left) plot, we can see that the LOESS model had the largest Bias with a mean of 0.0153675 compared 0.0008792373 (Nadaraya-Watson) which performed the best and 0.0008816932 (Spline Smoothing). In the Variance (middle) plot, we can see that the LOESS model performed the best with a mean of 0.00172738 compared to 0.02029143 (Nadaraya-Watson) which was the worst and 0.005499024 (Spline smoothing). The LOESS model having the highest bias and low variance makes sense since LOESS is a very simple model and as we stated before, “The simpler the algorithm, the more bias it has likely introduced.” The oversimplification of the model always results in a large error on training and test data. On the other extreme, Nadaraya-Watson has the lowest bias and the highest variance. Towarddatascience.com suggests that, “A model with high variance pays a lot of attention to training data and does not generalize on the data which it hasn’t seen before.” This means that although this model performed the best prediction, it’ll probably perform a lot worse when using a different dataset. This leads us to the MSE and as you can see, the Spline smoothing performed the best which makes sense since it had the best balance between the Bias and Variance.



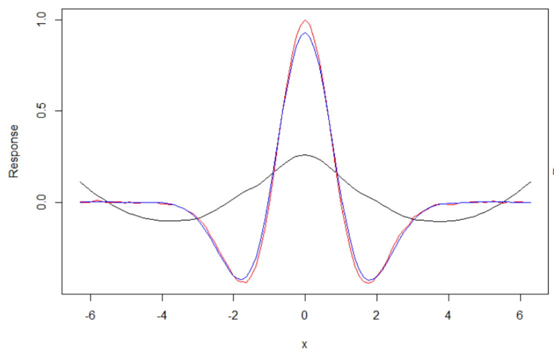


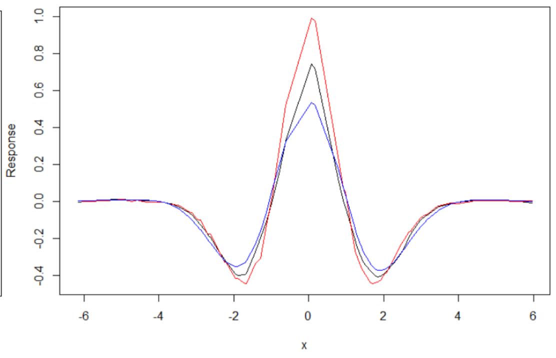


I believe that the comparison of these three methods is a fair comparison for the following reasons. For the LOESS model, the professor tells us to use a span of 0.75. I researched and found out from statsdirect.com that, “the default span is α = 0.75 because if you choose a span that is too small then there will be insufficient data near x for an accurate fit, resulting in a large variance. If the span is too large than the regression will be over-smoothed, resulting in a loss of information, hence a large bias.” The span basically controls the amount of smoothing. For the Nadaraya-Watson model, “a very tiny bandwidth corresponds to a "connect-the-dots" type of drawing. With a very large bandwidth, it will basically estimate every y-value as the mean of all the y-values”, stated by stat.berkeley.edu. Using a bandwidth of 0.2 means is considered tiny according to the article where they create a cross validation based on the range of 0.1 to 1.0 to see which bandwidth performed the best. Lastly, for the Spline smoothing model, the spar (smoothing parameter) is set to the default. This value usually ranges from 0 to 1 according to this particular Berkeley documentation. However, I found out that there is a cross validation parameter that you can include in the smooth.spline function to find the optimal spar value which resulted in 0.6093388. I believe that the parameters shown here are pretty reasonable especially when part 2 of the homework has the span, bandwidth, and spar at 0.3365, 0.2 (same as before), and 0.7163 respectively.

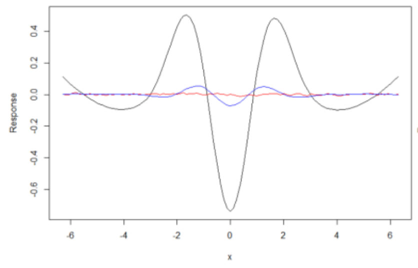
PART 2

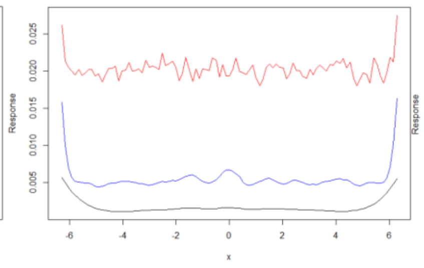
Now for Part 2, I did the same process in Part 1 (left plot) but changed the parameters based on the instructions and used non-equidistant points which resulted in the mean estimation (right plot) below. Already I can see that this model performed a lot rougher on all methods. The LOESS model performed better than before but was still not as good as the other models. Decreasing the span from 0.75 to 0.3365 as we stated before will cause a rougher shape for LOESS but it helped its overall shape since that also significantly reduced the Bias. Surprisingly, the Nadaraya-Watson looked a lot worse than part 1 even though the bandwidth was unchanged. This means that using non-equidistant points caused a significant change to the model which makes sense since we stated earlier that having a high variation will cause the model to overfit the training data so now we are seeing that consequence. Lastly, the Spline smoothing model performed worse than before but I believe this is because we are using a spar of 0.7163 assigned by the professor that I believe is not the optimal value (0.6093388) as I stated before.

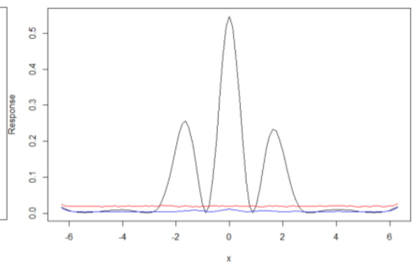


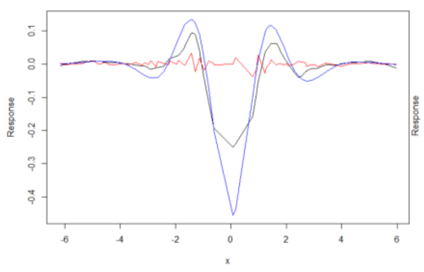


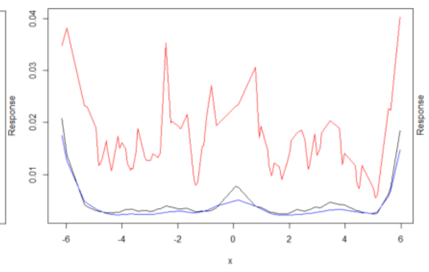
Since we have reviewed the mean estimations of Part 2, we can now talk about the comparisons of the Bias, Variance, and MSE shown respectively on the 2nd row especially compared to Part 1 on the 1st row.

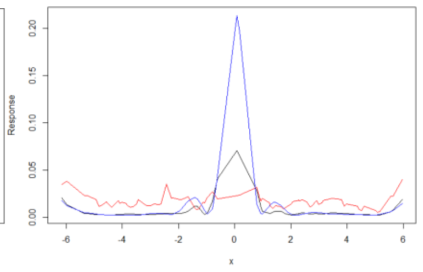








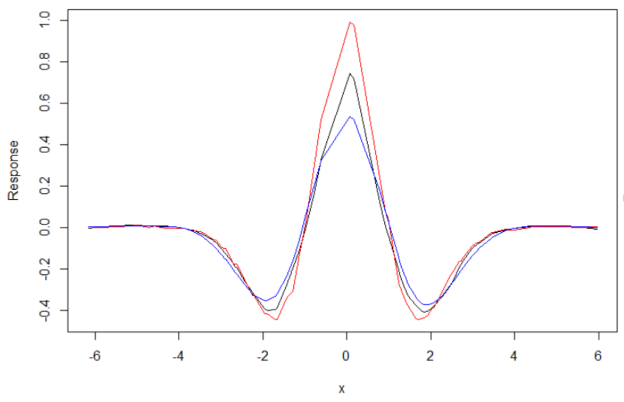


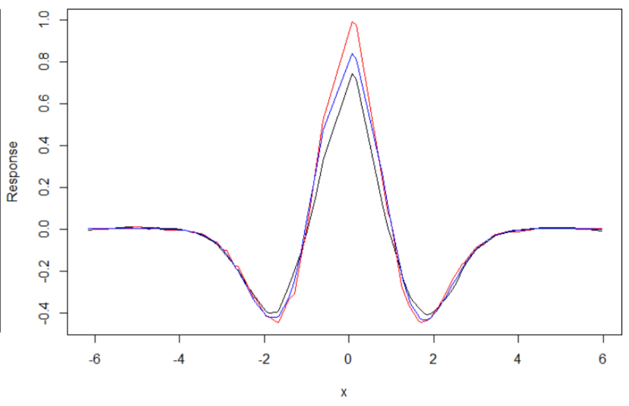


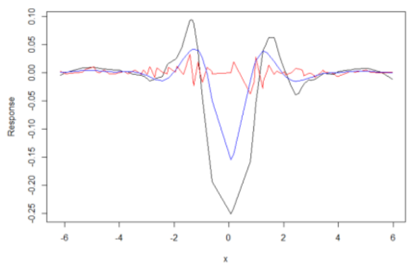
Again, keeping in mind that Part 2 x values are non-equidistant, the Bias plot showed that the LOESS model improved compared to the 1st model and the Nadaraya-Watson model performed a little worse than the 1st model but still performed the best amongst the other models. However, the Spline smoothing model had a huge jump in bias and I believe this is because of the nature of non-equidistant data on the model as well as not using the optimal spar value. For the variance plot, we can see that the variance for Nadaraya-Watson is still the highest but performs a lot rougher than the Part 1 and the Spline smoothing actually performed very similarly to the LOESS model. The MSE plot was the most interesting and as we can see, it seems like the non-equidistant points causes the Spline smoothing to perform poorly with a mean of 0.1083572 compared to 0.01534167 (Nadaraya-Watson) and 0.007064206 (LOESS). What's interesting is that it looks like Spline smoothing performs the worst visually but it actually performs very well at most x points on the tails but then performs very poorly between (+/- 1) and when we average these points, it technically outperforms the Nadaraya-Watson even though Nadaraya Watson performance is more consistent. However, averaging the points would probably not be the correct procedure since the misclassification is so extreme for the points between (+/- 1) and thus the Nadaraya-Watson would be the best model for Part 2.

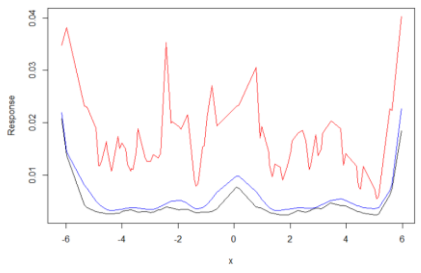
PART 3

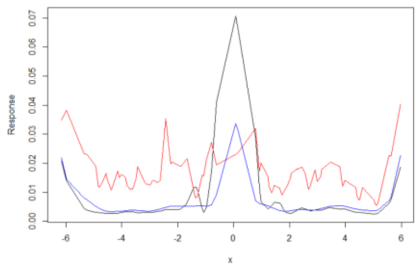
However, before I conclude, I decided to run the model (consider it Part 3) where I changed the spar value from 0.7163 to 0.6093388 (right plot) and I placed Part 2's plot on the left for comparison. You can see that the Spline smoothing now performs better than the LOESS model. I then plotted the Bias, Variance and MSE plots respectively shown below.









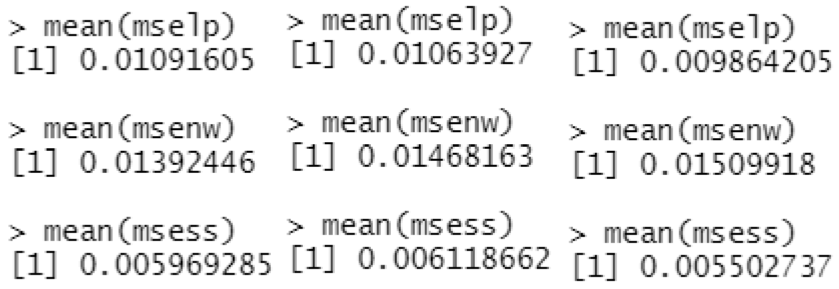


The Spline smoothing model performed better on all accounts and most importantly, you can see that the Spline smoothing MSE is similar to Nadaraya-Watson's MSE for points between (+/- 1) so based on this analysis, I would change my previous answer and say that the Spline smoothing model performed the best for the Mexican hat distribution. It's also worth noting that the average MSE was 0.005574356 which was smaller than the LOESS model in Part 2.

Conclusion

When analyzing all the cross-validated models, it is clear that the Spline smoothing model performed the best in terms of average test error performance and variance but Nadaraya-Watson performed better on average for the Bias. We can see that the LOESS model did not perform as well as the other models regardless of Part 1 and Part 2. I also noticed that the results were extremely close amongst most of the models so I decided to run different seeds to get a better understanding of the average performance. However, I doubled checked with 3 different seeds (shown below) and even without using the optimal spar value for each run, the Spline smoothing model outperformed all models in each instance and most likely more if additional seeds were run. This leads me to believe that the Spline smoothing model is the clear winner and would provide consistent accurate results for other datasets similar to this one. It’s worth noting that changing the bandwidth (Nadaraya-Watson) or span (LOESS) could help but I still don’t see these beating the Spline smoothing (with optimal spar value) because the LOESS is too simplistic and the Nadaraya-Watson was already using a very small bandwidth. However out of curiosity for the Nadaraya-Watson, I changed the bandwidth from 0.2 to 0.5 and saw that the MSE changed from 0.01534167 to 0.008069045. This was a significant improvement but I noticed that the variance plot was very high and it still didn’t perform as well as the optimal Spline smoothing at 0.005574356. Therefore, I tried iterating different bandwidth values and saw that ~0.7 performed the best with an average MSE of 0.07198675. I then tried getting the best span value for LOESS and saw that 0.25 worked well with an average MSE of 0.006381143. In the future I plan on re-attempting this type of problem set for my own work and hopefully will be able to better understand each model’s strengths in a personal work-based application.

Set.seed(99) Set.seed(88) Set.seed(77)



```
> mean(mselp)        > mean(mselp)        > mean(mselp)
[1] 0.01091605       [1] 0.01063927       [1] 0.009864205

> mean(msenw)        > mean(msenw)        > mean(msenw)
[1] 0.01392446       [1] 0.01468163       [1] 0.01509918

> mean(msess)        > mean(msess)        > mean(msess)
[1] 0.005969285      [1] 0.006118662      [1] 0.005502737
```