FYS-STK 4155 Project 3 Additional Exercise: Bias-Variance Tradeoff Analysis in the Context of Regression

Lasse Totland, Domantas Sakalys, Synne Mo Sandnes, Semya A. Tønnessen

(Dated: December 18, 2022)

In this bonus project, we perform an analysis of the bias-variance tradeoff in the context of regression. We analyze the bias-variance and Mean Square Error (MSE) for linear regression using Ordinary Least Squares (OLS) and Ridge, then deep learning using a Multi-Layer Perceptron (MLP) model and finally we look at ensemble methods using decision trees and random forests. All methods are tested using the Franke function as our data set. Overall we find that the Ridge method produces results with both low bias and variance while saving computational time. The overall MSE is consistently quite low for this model. Random forests also produce good results, achieving a lower bias in a shorter amount of time than Ridge. However, this model requires more computational time.

I. INTRODUCTION

In this exercise, the aim is to perform an analysis of the bias-variance tradeoff in the context of regression. We use three of the methods previously discussed in the course: Linear Regression using the Ordinary Least Square (OLS) method and Ridge regression [1], then deep learning using a Feed Forward Neural Network (FFNN) [2], and finally Ensemble methods using decision trees and random forests [3]. For an overview of these algorithms, we refer to our previous projects from this course, cited in the reference section.

For any supervised machine learning algorithm, we can decompose the error into bias and variance. Bias indicates the difference between the predictions of the model and the true values, while the variance deals with the spread of the data. A model with high bias oversimplifies the prediction and will underfit the data. Meanwhile, a model with high variance fit the training data closely, but will be unable to fit new data that it has not previously trained on, and so the data will be overfitted. Finding the right balance between both components as we attempt to minimize them is called the bias-variance tradeoff [4].

When choosing a model for a regression problem, it is important to consider both the bias and variance. Simple models, such as linear and logistic regression models, generally have a high bias and a low variance. More complex models, such as random forests, generally have a lower bias but higher variance.

We will study the bias-variance as a function of the complexity of the model using the Franke function as our data set. In section II we introduce bias-variance in further detail. Our results and a discussion of these are presented in section III, before we go over the conclusion and final outlook in section IV.

II. METHOD

Bias-Variance

The aim of supervised machine learning problems is to find a mathematical representation f which can explain the relationship between input predictors $\boldsymbol{x}$ and an observed outcome $\boldsymbol{y}$

$$\boldsymbol{y} = f(\boldsymbol{x}) + \epsilon, \tag{1}$$

where ϵ is some noise in the data. The algorithm learns from a training data set and iteratively makes predictions on the training data, in a process that can be thought of as learning. The learning stops when the algorithm achieves an acceptable level of performance.

The bias-variance is an estimate of how well an approximation performs, visualized by studying the complexity of the model. In order to find a measure of the expected error in a function, we want to minimize the error in the model when predicting the outcome from the predictors. Such an error is usually a measure of distance between the predicted outcome and the

observed outcome, for all observations. For this aim, we can use the MSE,

$$MSE\,(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2 \tag{2}$$

The error can be decomposed into the sum of three quantities,

$$\mathbb{E}[(\boldsymbol{y} - \tilde{\boldsymbol{y}})^2] = (\text{Bias}[\tilde{y}])^2 + \text{Var}[\tilde{f}] + \sigma^2, \tag{3}$$

The bias of the model $\tilde{\boldsymbol{y}}$ compared to the continuous function $\boldsymbol{f}$ that we wish to obtain is given by

$$(\text{Bias}[\tilde{y}])^2 = (\boldsymbol{y} - \mathbb{E}[\tilde{\boldsymbol{y}}])^2, \tag{4}$$

and the variance by

$$\text{Var}[\tilde{f}] = \frac{1}{n}\sum_{i}(\tilde{y}_i - \mathbb{E}[\tilde{\boldsymbol{y}}])^2. \tag{5}$$

Finally, σ is the variance of the noise.

The bias measures the difference between the model prediction and the outcome, and depends primarily on the model which we choose and how the model assumptions interpret the relationship between the predictors and the outcome. If the training data show a low performance, the model suffers from a high bias, as can be seen in figure 1. This is known as underfitting, and in such cases the model assumptions fail to understand the relationship between the predictors and the real outcome.

The variance measures how the train sets affects the model, by looking at the deviation from the true data. We want the model to be able to generalize for most train sets, or else it is only useful to the specific data set we train the model on. Such cases are known as overfitting, where the model is too well suited for the data set we trained it on. This is usually an issue for more complex models, and can be seen in poor test set performances. This can be seen visualized in the figure 1, where we see how the prediction error in the test sample increases for high complexity [4].

To summarize, a model having a high bias means that it predicts inaccurate results, even if we only see a small variance in these predictions. Having a lower bias, but with higher variance, implies that out predictions are centered around the true value, but vary markedly. As the model complexity increases, the variance tends to increase and the squared bias tends to decrease. The opposite behavior occurs as the model complexity is decreased. Even the best models are approximations of complex relationships between data, and therefore there is an irreducible error that cannot be avoided. When predicting the model we can attempt to ensure the optimal bias-variance trade-off, in order to ensure we achieve trustworthy results.

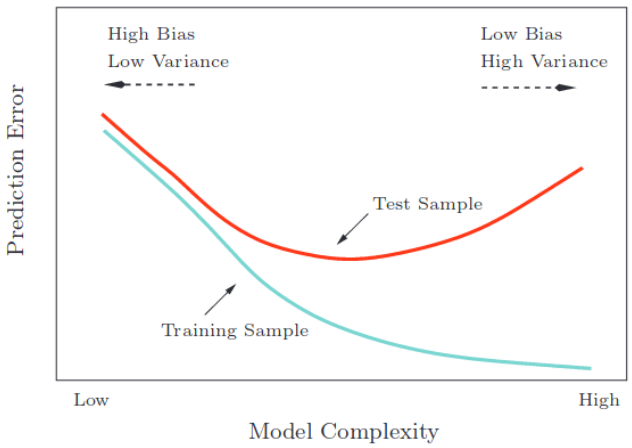

Figure 1. Test and training error as a function of model complexity. The image is figure 2.11 from Hastie et al. [5]

When studying linear regression, we use the Ordinary Least Squares (OLS) method and the Ridge methods, obtained using `LinearRegression` and `Ridge`, from the `linear_model` library in `scikit-learn`. For the deep learning study, we use a Fast Forward Neural Network (FFNN) obtained using `MLPRegressor` from `neural_network`. Finally, we study the ensamble methods using decision trees and random forests, obtained from `BaggingClassifier` and `RandomForestRegressor`, from the `ensemble` package. For resampling, we use the bootstrap method, in order to get the best possible estimates.

Data Set

The data set used to analyze the bias-variance is the two-dimensional Franke function, which is common to use in order to evaluate different suface interpolation techniques. The function is a weighted sum of four exponentials,

$$\begin{aligned} f(x,y) = &\frac{3}{4}\exp\left(-\frac{(9x-2)^2}{4} - \frac{(9y-2)^2}{4}\right) \\ &+\frac{3}{4}\exp\left(-\frac{(9x+1)^2}{49} - \frac{(9y+1)}{10}\right) \\ &+\frac{1}{2}\exp\left(-\frac{(9x-7)^2}{4} - \frac{(9y-3)^2}{4}\right) \\ &-\frac{1}{5}\exp\left(-(9x-4)^2 - (9y-7)^2\right), \end{aligned}$$

defined for $x, y \in [0,1]$. For our purposes, we sample the function at 40 uniformly distributed data points, ϵ,

$$z = f(x,y) + \epsilon,$$

where f is the Franke function and the noise is generated from a normal distribution $\epsilon \sim N(0, \sigma = 0.1)$. The data set is split at random into two groups, a training set and a test set, using 20% of the set for testing and the remaining 80% for training. The train set consists of the portion of data to fit the model, while the test set is the portion of data to evaluate the model performances. Assessing the performances on the test set provides a better estimate of how the model will perform on unseen data, as this set contains data not previously seen by the model.

III. RESULTS AND DISCUSSION

Linear Regression

We begin by studying the methods used for linear regression, where we use the polynomial degree as a measure for the complexity. Figure 2 shows the bias-variance as a function of the complexity of the model when using the OLS method, and figure 3 shows the same using Ridge regression, both for a bootstrap with $n = 100$. Linear regression models such as these generally have a high bias and a low variance, which is in accordance with what we observe for lower complexities from both figures.

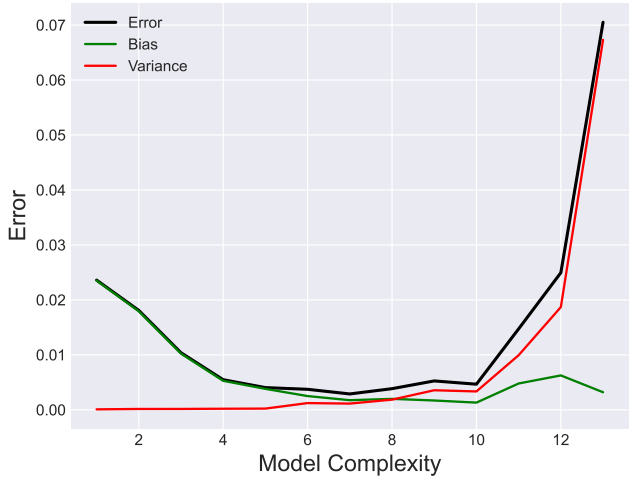

Figure 2. Bias-variance using the OLS method for $n = 100$ bootstraps. The complexity is the polynomial degree, shown up to an order of 15.

The data set we approximate is non-linear, and therefore we use linear regression with polynomial features to approximate it. It is thus as expected that a polynomial with degree 1 is insufficient to fit the training samples, as can be seen from the high bias observed for this degree. When using the OLS method, a polynomial of degree 7 obtains a low bias-variance and should therefore approximate the true function much better. For higher degrees the model will overfit the training data, as can be seen from how the variance increases rapidly after a polynomial of degree 10.

For Ridge, the variance increases only moderately compared to OLS. The ridge model is better at handling the error caused by the variance, due to the introduction of the l2 parameter which deals with overfitting. This ensures that the model does not blow up for higher complexity. In figure 3 we have visualized two instances for two different l2 parameters. In the lower plot, where λ is lower, we can see how the variance in-

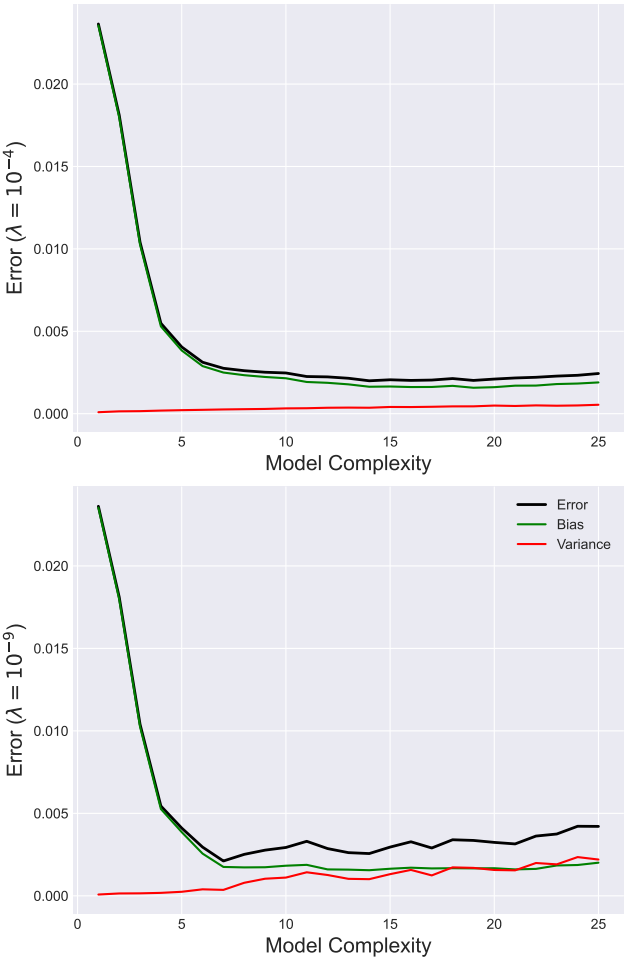

Figure 3. Bias-variance using the Ridge method with $\lambda = 1 \times 10^{-4}$ (top) and $\lambda = 1 \times 10^{-9}$ (bottom) for $n = 100$ bootstraps. The complexity is the polynomial degree, shown up to an order of 25. Note how a low parameter makes the model less robust to overfitting.

creases more, and we get a model closer to the OLS. If we set $\lambda = 0$ we would get OLS. For a higher λ, Ridge achieves a very good model for higher polynomial orders, and overfitting does not seem to become an issue.

Deep Learning

Next, we study the bias-variance for deep learning methods using a Multi-Layer Perceptron (MLP) model. The bias-variance is presented as a function of the number of nodes in one hidden layer in figure 4. As expected, the bias decreases with increasing complexity, before eventually the variance increases. Around 25 neurons, the variance is greater than the bias. It is interesting to see that the lowest error we achieve is greater than what we achieved for both OLS and Ridge.

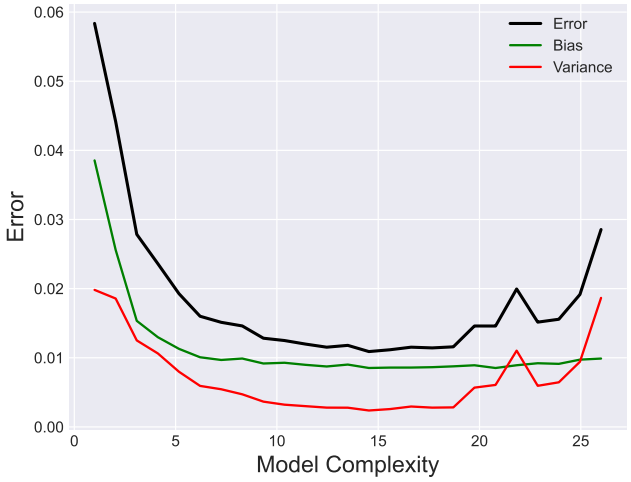

Figure 4. Bias-variance using the FFNN method for a Multilayer Perceptron regressor for 25 neurons and 1 hidden layer. The number of bootstraps is $n = 100$.

If we increase the number of hidden layers from one to two, we get the plot presented in figure 5. Here, the variance only increases moderately, and the lowest error is somewhat lower than the

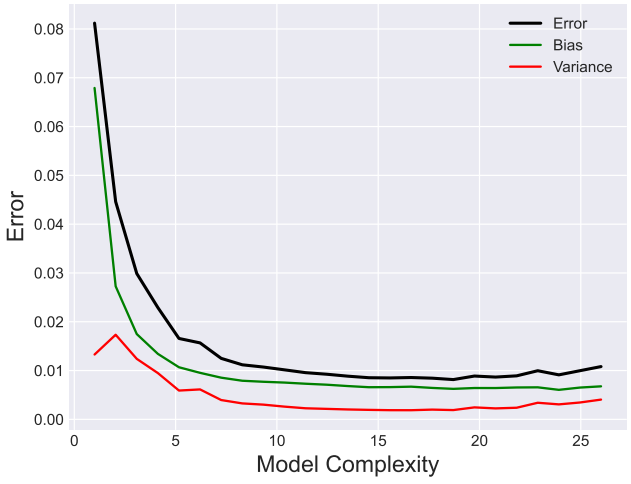

Figure 5. Bias-variance using the FFNN method for a Multilayer Perceptron regressor for 25 neurons and 2 hidden layers. The number of bootstraps is $n = 100$.

model with only one layer. While the bias is high in the beginning, it decreases quickly. From Neural Network models we expect to see that overfitting becomes more prominent as we increase the complexity [5], and the variance does begin to increase somewhat as the complexity increases, however not nearly as much as we saw for OLS.

Ensemble Methods

Finally, we study two ensemble methods: the decision tree algorithm and the random forests algorithm.

Figure 6 shows the bias-variance and MSE for a decision tree algorithm trained on the Franke data with a number of bootstraps $n = 100$. The complexity corresponds to the maximum depth of the tree , which limits the number of nodes in the tree [6]. Deeper trees have more complex decision rules and a more complex fitter of the model. As expected, the bias is high for low complexity whilst the variance increases with higher complexity. More complex models generally have a lower bias but a higher variance, and we can see that the variance begins to increase for quite a low order of complexity.

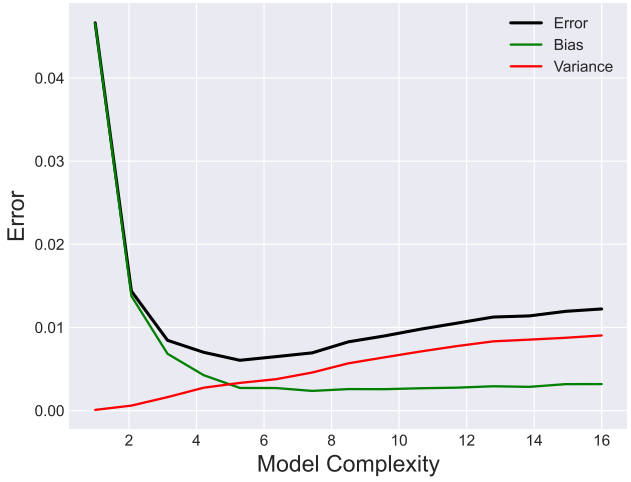

Figure 6. Bias-variance using the decision tree algorithm on the Franke data set for $n = 100$ bootstraps.

The basic decision tree method can be used to build more complex networks, such as random forests. In figure 7 we plot the bias-variance and MSE for such a random forest algorithm trained on the Franke data with a number of bootstraps $n = 100$. The bias does not change much from what we saw in figure 6, but the variance is lower, and there is clear improvement in the MSE. This method uses averaging to improve the predictive accuracy and control overfitting, which is the reason for the lower variance even as the complexity is increasing. This allows us to use a more complex model when this is needed. The random forest model therefore remains stable for higher levels of complexity compared to the basic decision tree method.

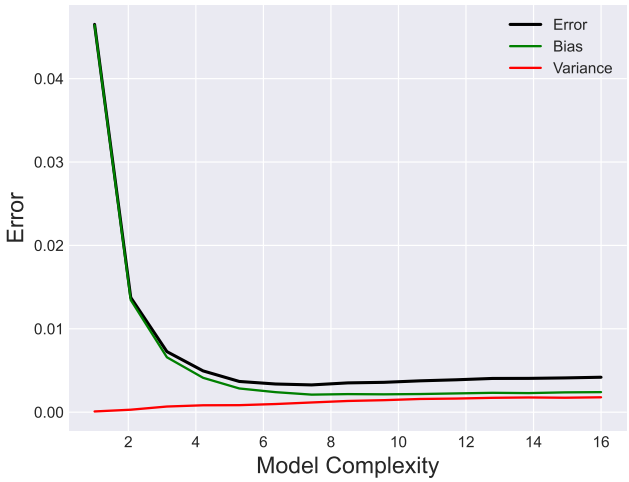

Figure 7. Bias-variance using the random forests algorithm on the Franke data set for $n = 100$ bootstraps.

IV. CONCLUSION

Based on our results we have been able to see how the bias and variance vary for different algorithms. Generally, the bias starts off high, before decreasing with increasing complexity, whilst the variance starts of low and will increase with the complexity. This behaviour leads to a point where the MSE is at its lowest, and where we are most likely to avoid both overfitting and underfitting.

For OLS the variance quickly became large

enough that overfitting was an issue, something that was not observed for the penalized Ridge model. Overfitting generally was not an issue for Ridge, MLP or random trees. Ridge and random trees behave in a similar manner, with the variance increasing slowly for higher complexity, but the bias decreases more quickly for the random trees method. Both achieve a low error, but Ridge unexpectedly achieves the lowest of the two. Ridge is also more computationally efficient, and so it seems as if this model is a better choice, based on computational time and performance.

Generally, when choosing the best model and parameters for a data set, we have to consider the type of data set. The linear regression methods we study are well suited for lower complexity models, as they often require less computational time. However, many models require more complexity and flexibility, making random forests, or other variations of decision trees good choices. In this case, it seems that that Ridge is the optimal models in terms of bias-variance, error, and computational time. While all methods are able to obtain a low bias-variance, Ridge manages to do so consistently, for higher levels of complexity.

[1] S. M. S. Lasse Totland, Domantas Sakalys, "Regression analysis and resampling methods," (2022).

[2] S. M. S. Lasse Totland, Domantas Sakalys, "Developing neural networks for regression and classification problems," (2022).

[3] S. M. S. S. A. T. Lasse Totland, Domantas Sakalys, "Prediction of heart disease diagnosis with neural network and random forest machine learning models," (2022).

[4] M. Hjorth-Jensen, *Applied Data Analysis and Machine Learning* (2021) jupyter Book.

[5] J. F. Trevor Hastie, Robert Tibshirani, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. (Springer, 2009).

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, Journal of Machine Learning Research **12**, 2825 (2011).