Bias-variance trade-off analysis as a function of model complexity

Ada Weinert Ravn
Faculty of Mathematics and Natural Sciences, University of Oslo

(Dated: January 5, 2022)

I. INTRODUCTION

In this bonus task we will take a look at the bias and variance trade-off in terms of model complexity for three common sets of regression models, namely Linear regression, Ensemble models and SVM. All the code used in this project can be found in the GitHub directory [1].

II. BIAS-VARIANCE TRADE-OFF

The information contained here has been taken from Project 1.

The art of machine learning is creating accurate models with high accuracy. A common way to assess model accuracy is through the MSE. The cost function of the MSE is

$$C(\beta, X) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2 = E[(y_i - \hat{y}_i)^2]$$

$$= Bias(\hat{y}_i)^2 + Var(\epsilon) + Var(\hat{y}_i)$$

Bias describes the errors from assumption, e.g. we assume that the true form of a model is linear. The greater the bias the stronger the assumptions of the model. Variance is the average of the squared deviation, or error, from the true values. So in order to minimize the MSE we need to minimize the bias and variance of the model. The variance of the noise is given by the data set and can't be changed.

Intuitively, having high bias will mean our model will have low variance, as we have assumed much of the model, which means our predicted errors will be smaller. Likewise, if we have low bias, we will have higher variance. This property is called the *bias-variance trade-off*. A big part of finding the best model for a given data set is to balance this bias-variance trade-off, finding the right amount of assumptions to make in order for the variance to be acceptable.

[1] https://github.com/AWRavn/Bias-variance-trade-off-analysis-as-a-function-of-model-complexity

III. IMPLEMENTATION

Franke Function was used as the tested algorithm with 20x20 grid of values. Parts of the implementations were appropriated from Project 1. bias_variance_decomp() function from the mlxtend library has been used to calculating the metrics, with 100 bootstraps for each method.

IV. RESULTS AND DISCUSSION

A. Linear Regression

We implement the model complexity in terms of polynomial degrees. In the analysis we're using three algorithms: Ordinary Least Squares (OLS), Ridge with $\lambda = 0.001$ and Lasso with $\lambda = 0.0000001$. The outputs can be seen in Figure 1, 2 and 3 respectively.

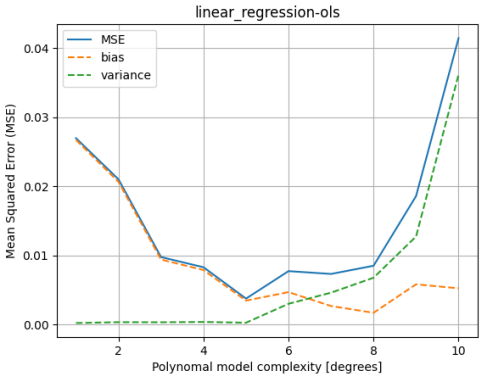

Figure 1. Bias-variance trade-off for Ordinary Least Squares (OLS).

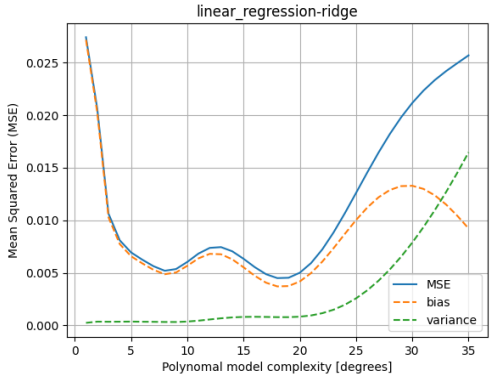

Figure 2. Bias-variance trade-off for Ridge.

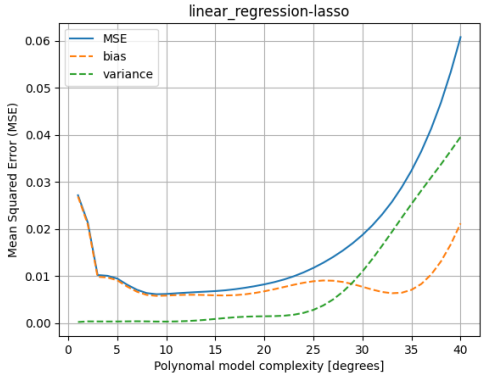

Figure 3. Bias-variance trade-off for Lasso.

For all of those models we have bias as the main contributor towards the MSE, then at a certain point, different for each model, its position is changed with the variance. The location of that crossing is considered to be optimal degree for the model, we have that for OLS that degree is 6 for Ridge 32 and for Lasso 28. As opposed to OLS, Ridge and Lasso models result in a lower error for higher degrees of complexity. Interestingly for the Lasso algorithm the relative contributions of bias and variance appear to remain relatively constant for lower degrees of complexity. For the included dataset OLS achieved best results, followed by Lasso and finally Ridge.

B. Ensemble Methods

For the ensemble methods the complexity of the model can be implemented in terms of the depth of each tree. We will interpret output of the bias-variance trade-off for the following models: Decision Trees, Bagging, Gradient Boosting and Random Forests. The outputs can be seen in Figure 4, 5, 6 and 7.

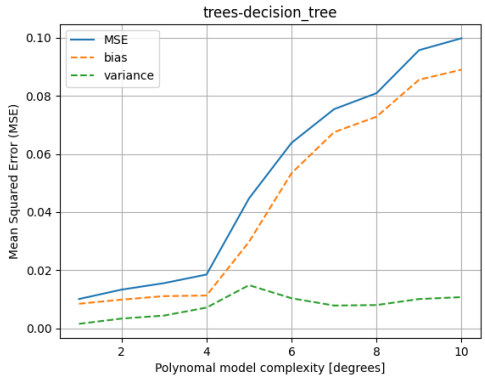

Figure 4. Bias-variance trade-off for Decision Trees.

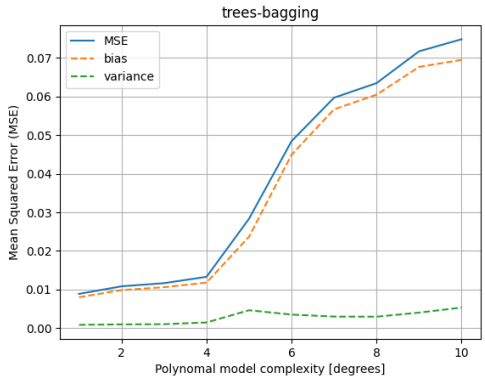

Figure 5. Bias-variance trade-off for Bagging.

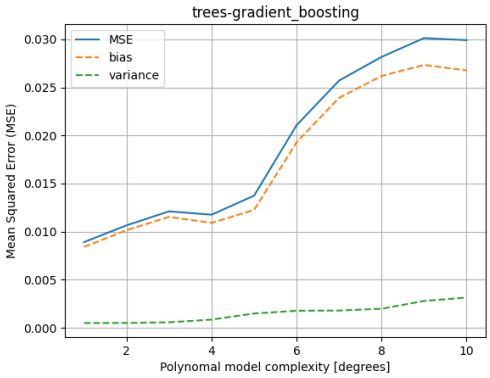

Figure 6. Bias-variance trade-off for Gradient Boosting.

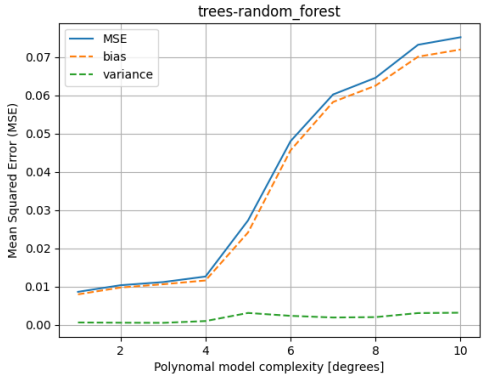

Figure 7. Bias-variance trade-off for Random Forests.

Here the outputs of all four models appear very similar: a steadily increasing MSE, with very strong contribution of bias, and consistently low (but rising) variance. For even higher complexity the error is expected to taper off and remain stable as maximum potential tree depth is reached. Examples of this behavious can be seen in Figure 8 and 9.

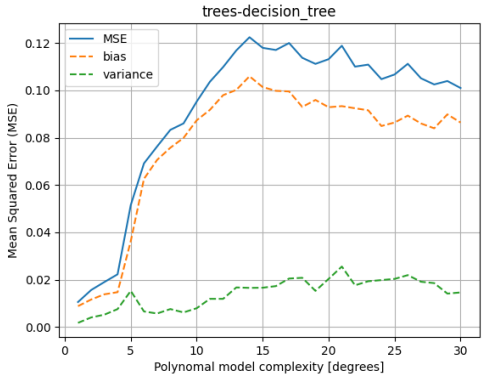

Figure 8. Bias-variance trade-off for Decision Trees.

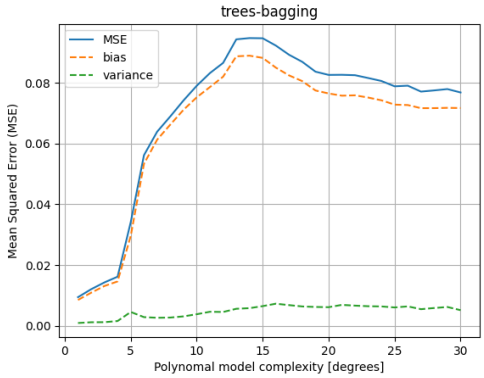

Figure 9. Bias-variance trade-off for Bagging.

C. SVM

Here we will use the Support Vector Regressor with the Polynomial kernel, where the degree of complexity will once again be seen in terms of polynomial degrees. The output can be seen in Figure 10

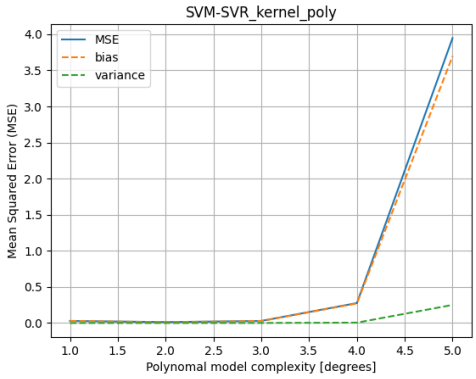

Figure 10. Bias-variance trade-off for Support Vector Machines (SVM), with the Polynomial kernel.

For this graph the error exponentially increases for each degree of complexity, indicating that a linear solution might have worked better. As was the case with the Ensemble methods, the variance is consistently a relatively small contributor to overall MSE. It is expected that this model would work much better for a different set of data, one that doesn't simulate terrain.

V. SUMMARY AND CONCLUSIONS

Linear regression methods struggle with rise in variance for higher complexity levels of the model. Ridge and Lasso models with regulation may regularise the output to decrease the variance at the cost of bias though. The models are computationally efficient but simplistic in assumption in terms of predicted variables. They assume constant variance around the mean and are therefore severely affected by outliers in data.

For all types of Ensemble models the depth of the tree determines the variance. Pruning of the tree may help control it. Notably decision trees are able to handle missing data relatively well, and don't require much data preprocessing to work efficiently. The big downside is that the algorithm is not suitable for predicting continuous values.

SVM based regression is at its best when working with

high dimensional spaces with data with clear degree of separation. It is not suitable for large datasets, and is not robust to noise in data.

We finish with a list of lowest MSE achieved for each algorithm:

LinearRegression: 0.0037826681251264733

Ridge(alpha=0.001): 0.00387

Lasso(alpha=1e-07): 0.00616

DecisionTreeRegressor: 0.01016

BaggingRegressor: 0.00889

GradientBoostingRegressor: 0.00891

RandomForestRegressor: 0.00867

SVR(kernel='poly'): MSE: 0.01289