Bias variance analysis

Sakarias Frette, William Hirst, Mikkel Metzsch Jensen
(Dated: December 16, 2021)

I. INTRODUCTION

In this additional paper we are going to perform an analysis of the bias-variance trade-off on a regression type problem using decision trees, boosting (gradient-boosted-trees) and feed forward neural network respectively. We are going to use the so-called Ames housing data which concerns housing sale price in connection to relevant real estate features, such square area, number of rooms etc.

In the theory section we are going to introduce the bias-variance error decomposition and the bootstrap resampling method which constitutes the foundation of the bias variance analysis. In the implementation section we are going to introduce the data set in more detail, and explain hos this is handled to fit our scope. In addition we are going to briefly explain the regression methods in use and how these is implemented in our code. Finally in the result and discussion section we present the bias-variance analysis itself.

II. THEORY

A. Bias-Variance decomposition[1]

We consider a general dataset $\mathcal{L}$ which consists of the datapoints $\boldsymbol{X}_{\mathcal{L}} = \{(y_j, \boldsymbol{X}_j), j = 0, \ldots, n-1\}$. We assume that the true target data $\mathbf{y}$ originates from a model $f(x)$ with added normal distributed noise ϵ (with variance σ^2 and zero mean) given as

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

The Mean Square Error (MSE) between the target data $\mathbf{y}$ and a given prediction $\tilde{\mathbf{y}}$ can be written out as

$$C(\mathbf{X}, \boldsymbol{\beta}) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2 = \mathbb{E}[(\mathbf{y} - \tilde{\mathbf{y}})^2]. \tag{2}$$

We are going to show that this can be decomposed into bias, variance and the noise variance σ^2. This is known as the Bias-Variance tradeoff. We have

$$\begin{aligned}
\mathbb{E}[(\mathbf{y} - \tilde{\mathbf{y}})^2] &= \mathbb{E}[(f + \epsilon - \tilde{y})^2] \\
&= \mathbb{E}[(f + \epsilon - \tilde{y} + \mathbb{E}[\tilde{y}] - \mathbb{E}[\tilde{y}])^2] \\
&= \mathbb{E}[f^2 + f\epsilon - \tilde{y}f + f\,\mathbb{E}[\tilde{y}] - f\,\mathbb{E}[\tilde{y}] \\
&\quad + f\epsilon + \epsilon^2 - \tilde{y}\epsilon + \epsilon\,\mathbb{E}[\tilde{y}] - \epsilon\,\mathbb{E}[\tilde{y}] \\
&\quad - \tilde{y}f - \tilde{y}\epsilon - \tilde{y}\,\mathbb{E}[\tilde{y}] + \tilde{y}\,\mathbb{E}[\tilde{y}] + \tilde{y}^2 \\
&\quad + f\,\mathbb{E}[\tilde{y}] + \epsilon\,\mathbb{E}[\tilde{y}] - \tilde{y}\,\mathbb{E}[\tilde{y}] - \mathbb{E}[\tilde{y}]^2 + \mathbb{E}[\tilde{y}]^2 \\
&\quad - f\,\mathbb{E}[\tilde{y}] - \epsilon\,\mathbb{E}[\tilde{y}] + \tilde{y}\,\mathbb{E}[\tilde{y}] - \mathbb{E}[\tilde{y}]^2 + \mathbb{E}[\tilde{y}]^2].
\end{aligned}$$

By taking the expectation value of the components we get

$$\begin{aligned}
\mathbb{E}[(\mathbf{y} - \tilde{\mathbf{y}})^2] &= \mathbb{E}[(f - \mathbb{E}[\tilde{y}])^2] + \mathbb{E}[\epsilon^2] + \mathbb{E}[(\mathbb{E}[\tilde{y}] - \tilde{y})^2] \\
&\quad + 2\,\mathbb{E}[(f - \mathbb{E}[\tilde{y}])\epsilon] + 2\,\mathbb{E}[\epsilon(\mathbb{E}[\tilde{y}] - \tilde{y})] \\
&\quad + 2\,\mathbb{E}[\epsilon(\mathbb{E}[\tilde{y}] - \tilde{y})(f - \mathbb{E}[\tilde{y}])] \\
&= \mathbb{E}[(f - \mathbb{E}[\tilde{y}])^2] + \mathbb{E}[\epsilon^2] + \mathbb{E}[(\mathbb{E}[\tilde{y}] - \tilde{y})^2] \\
&\quad + 2(f - \mathbb{E}[\tilde{y}])\,\mathbb{E}[\epsilon] + 2\,\mathbb{E}[\epsilon]\,\mathbb{E}[(\mathbb{E}[\tilde{y}] - \tilde{y})] \\
&\quad + 2(f - \mathbb{E}[\tilde{y}])\,\mathbb{E}[(\mathbb{E}[\tilde{y}] - \tilde{y})].
\end{aligned}$$

Next we use that that the the noise ϵ is normally distributed with a mean equal to 0, and variance σ^2. Thus we have $\mathbb{E}[\epsilon] = 0$ and $\mathbb{E}[\epsilon^2] = \sigma^2$ which yields

$$\begin{aligned}
\mathbb{E}[(\mathbf{y} - \tilde{\mathbf{y}})^2] &= \mathbb{E}[(f - \mathbb{E}[\tilde{y}])^2] + \sigma^2 + \mathbb{E}[(\mathbb{E}[\tilde{y}] - \tilde{y})^2] \\
&\quad + 2(f - \mathbb{E}[\tilde{y}])\,\mathbb{E}[(\mathbb{E}[\tilde{y}] - \tilde{y})].
\end{aligned}$$

Finally we have that the expectation value of an expectation value is the expectation value itself, $\mathbb{E}[\mathbb{E}(x)] = \mathbb{E}(x)$, which makes the last term in the above cancel out. Thus we get

$$\begin{aligned}
\mathbb{E}[(\mathbf{y} - \tilde{\mathbf{y}})^2] &= \mathbb{E}[(f - \mathbb{E}[\tilde{y}])^2] + \mathbb{E}[(\tilde{y} - \mathbb{E}[\tilde{y}])^2] + \sigma^2 \\
&= \frac{1}{n}\sum_i (f_i - \mathbb{E}[\tilde{y}])^2 + \frac{1}{n}\sum_i (\tilde{y}_i - \mathbb{E}[\tilde{y}])^2 + \sigma^2 \\
&= \text{Bias}^2 + \text{Variance} + \text{Noise}.
\end{aligned} \tag{3}$$

Hence we arrived at the bias-variance relation. The decomposition shows that the MSE error can be explained in terms of

- Bias: A systematic error of the prediction.
- Variance: How much the predictions vary/fluctuates.
- Noise: The contribution from the noise.

[1] Parts of the theory discussed in the following section is heavily influenced or taken from project 1

However, in practice we don't know f_i (the un-noised data points), but in practice this is the only available quantity for which we can approximate the bias. By using our noised data points y_i as the best approximation we basically absorb the noise into the bias term. We see this by substituting f_i with y_i such that we get

$$\begin{aligned}\mathbb{E}[(y-\mathbb{E}[\tilde{y}])^2] &= \mathbb{E}[(f+\epsilon-\mathbb{E}[\tilde{y}])^2] \\ &= \mathbb{E}\left[\left(\epsilon+(f-\mathbb{E}[\tilde{y}]\right)^2\right] \\ &= \mathbb{E}[\epsilon^2]+\mathbb{E}[(f-\mathbb{E}[\tilde{y}])^2]+\mathbb{E}[-2\epsilon(f-\mathbb{E}[\tilde{y}])] \\ &= \sigma^2+\mathbb{E}[(f-\mathbb{E}[\tilde{y}])^2]+\mathbb{E}[-2\epsilon]\,\mathbb{E}[f-\mathbb{E}[\tilde{y}] \\ &\quad +\mathrm{Cov}(-2\epsilon, f-\mathbb{E}[\tilde{y}]) \\ &= \sigma^2+\mathbb{E}[(f-\mathbb{E}[\tilde{y}])^2] = \sigma^2+\mathrm{bias}^2.\end{aligned}$$

B. Bootstrap

In order to calculate the bias and variance from a single dataset we use bootstrapping. Bootstraping is a resampling method for which we resample the data by drawing new data points from with replacement from our available data set. Lets say we have a $N = 5$ dataset given as $\{1,2,3,4,5\}$. A possible bootstrap sample could then be $\{1,2,2,4,5\}$. For a sufficiently large data set there is practical no chance of getting the same sample twice, and thus this will generate slightly different data set for each resample. By drawing B bootstrap samples we can perform B predictions for which we can analyze the bias and variance in the prediction results according to equation 3.

III. IMPLEMENTATION

The numerical implementations can be found on our github.

A. Data

We are going to address the so-called Ames Housing dataset, which is an alternative to the well known Boston housing data. The data set contains 80 features specifying different aspect of real estates and the corresponding sale price (target value). It has a total of 1460 entries which we divide into train and test data with a 80-20% split. We are going to focus on regression methods and thus we trim away all features with a classification structure, such as categories of neighbourhoods and house styles. Notice that while binary classes could be quantized rather easily, the multi-class features would require a subjective ranking when transforming from categories to numbers. Thus we decide to exclude all category based features and we are left with 35 features. Among those we calculate the correlation value (Pearson's Correlation Coefficient) between each feature and the target value. In order to reduce the data set even more we pick only the top ten features, which is listed in the following, ordered from highest to lowest correlation number.

1. YearRemodAdd: Remodel date (same as construction date if no remodeling or additions)
2. YearBuilt: Original construction date.
3. TotRmsAbvGrd: Total rooms above grade (does not include bathrooms).
4. FullBath: Full bathrooms above grade.
5. 1stFlrSF: First Floor square feet.
6. TotalBsmtSF: Total square feet of basement area.
7. GarageArea: Size of garage in square feet.
8. GarageCars: Size of garage in car capacity.
9. GrLivArea: Above grade (ground) living area square feet.
10. OverallQual: Rates the overall material and finish of the house:
    - 10 Very Excellent
    - 9 Excellent
    - 8 Very Good
    - 7 Good
    - 6 Above Average
    - 5 Average
    - 4 Below Average
    - 3 Fair
    - 2 Poor
    - 1 Very Poor

We see that the first nine features essentially concerns age, area and number of rooms. Thus one could argue that the most effective data set reduction could be done by combining some of these into single categories like total area or total number of rooms. Another effective way to reduce the number of features properly is done by principal component analysis. However, for this paper we are not particular interested in maximizing efficiency or prediction quality but rather to compare performance and bias-variance trade-off across multiple methods. Thus we are satisfied with the top ten chosen features.

Since some of the features takes relatively big values numbers like year stamps and area, while other deals with small quantities like number of rooms, we perform a min-max scaling which scales all columns of the design matrix to the interval $[0,1]$. This is done by shifting the mean and scaling by a factor such that the minimum takes value 0 and maximum takes value 1.

B. Regressions method

We are going to utilize three different methods for the regression problem based on the housing data. We choose to use decision trees, boosting (gradient-boosted-trees), and feed forward neural network.

1. Decision Tree

In the regression case a decision tree divides the predictor space (the set of possible values $x_1, x_2, \ldots, x_p$) into J distinct and non-overlapping regions $R_1, R_2, \ldots, R_J$. For every new observation that falls into the region R_j we simply predict the mean value $\bar{y}_{Rj}$ of the training values in R_j. Since it becomes computational challenging to consider all possible partitions we are going to use a top-down approach. That is, we will consider the best split at each step rather than looking ahead and picking the splits that will lead to a better tree in the future. For a given feature j we consider a cut-point s which splits the predictor space into two regions

$$R_1 : \{X | x_j < s\},$$
$$R_2 : \{X | x_j \geq s\},$$

where X denotes all the features in the form of the design matrix. We want to choose the feature j and cut point s such that we minimize the MSE given as

$$\text{MSE}(j, s) = \sum_{i:x_i \in R_1} (y_i - \bar{y}_{R1})^2 + \sum_{i:x_i \in R_2} (y_i - \bar{y}_{R2})^2$$

For N data points there is a maximum of $N - 1$ cut points which will make an unique contribution to the error. Thus for p features the total combination of splits is proportional to $p \times N$. Thus we can actually effort to go through all of them and pick the combination of j and s which minimize the MSE. After the first cut the tree branches into two regions for which we can repeat the splitting process again. We continue to split the branches until the max depth is reached, or when a certain regions contains less point than some predefinied limit. In all cases we have to stop splitting a certain branch when there is only one training point left in the region. We denote the end of a branch as a leaf.

We use the DecisionTreeRegressor class from sklearn.tree to perform the training. Our implementation can be found on our github in bias_variance_analysis.py which is executed through decision_tree.py.

2. Boosting

The basic idea of boosting or gradient boosting is to combine weak classifiers in order to create a combined strong classifier. We are going to use a shallow decision tree as our weak classifier and thus we might denote this method as gradient-boosted-trees. We decide on a number of weak estimators M for which we want to create a model on the form $\tilde{y} = F(x)$. We build up the model in stages $m \in 1, 2, \ldots, M$ by adding a new estimator h_m such that

$$F_{m+1} = F_m(x) + h_m(x) = y \quad \Longleftrightarrow \quad h_m(x) = y - F_m(x)$$

For each step we want to choose $h_m(x)$ such that we minimize the MSE, by fitting h to the residual $y - F_m(x)$. Thus each added estimator tries to correct the errors from the previous model.

For the implementation of this method we use AdaBoostRegressor from sklearn.ensemble. Our implementaiton can be found on our github in bias_variance_analysis.py which is executed through boosting.py.

3. Feed Forward Neural Network

For details about the Feed Forward Neural Network we refer to a previous paper that we made about the subject. For this implementation we used Sequential from tensorflow.keras which can be found on our github in bias_variance_analysis.py which is executed through FFNN.py.

IV. RESULTS AND DISCUSSION

A. Decision Tree

We begin by analyzing the bias-variance trade-off for the decision tree. Here we identify the tree depth to be the complexity variable. For this we could effort to use 5000 bootstrap samples. The result is shown in figure 1.

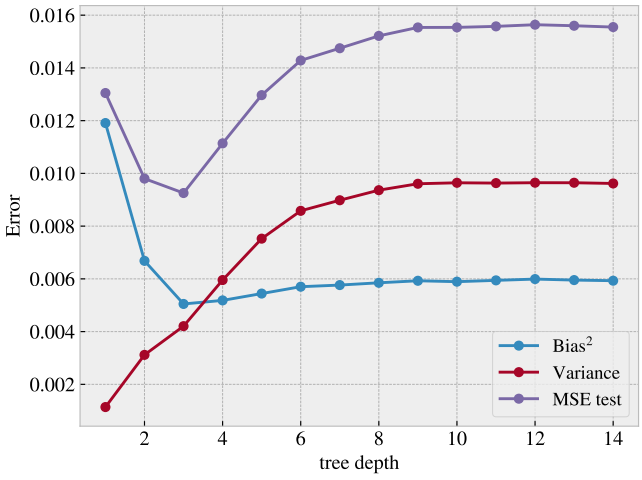
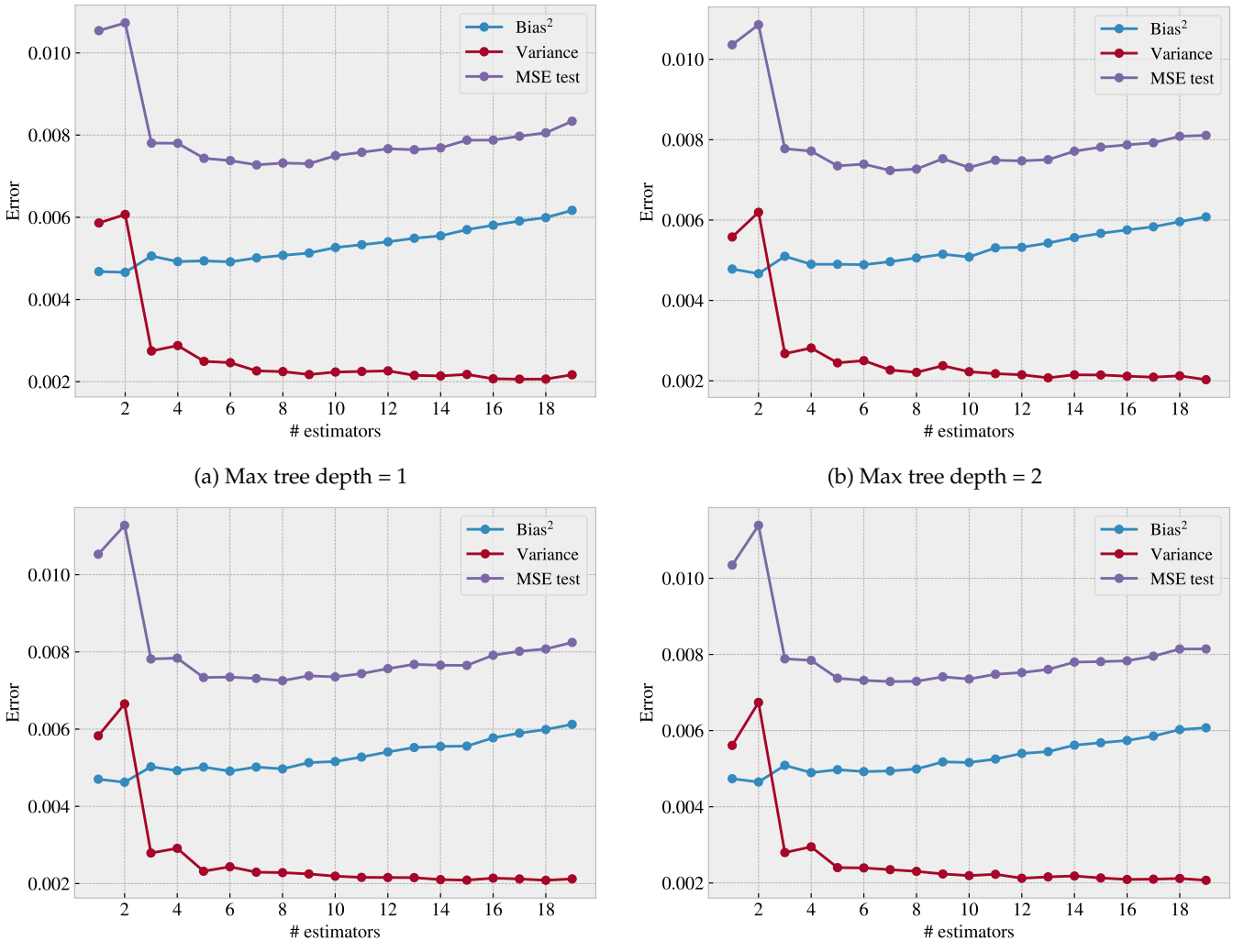


Figure 1: Bias-variance trade-off for a decision tree with a varying max tree depth. For this analysis we used 5000 bootstrap samples.

From figure 1 we see a quite typical example of the bias-variance trade-off where the initial increase in complexity results in a reduce in bias and a simultaneously increase in the variance. The sweet spot in our case is found to be at max tree depth of 3 for which the total test MSE is at its lowest. While the variance increase somewhat monotonically towards a plateau, the bias hits a minimum at tree depth = 3 and then slowly increases to another plateau. Since both the bias and variance plateaus it does not show any promise of better predictions for a further increased complexity.

B. Boosting

Next we expand on the single decision tree model by introducing an ensemble of trees. From the previous result (see figure 1) we found the best classifier with tree depth = 3 thus we try to train the model with (weak) classifiers of depth 1, 2, 3 and 4. We identify the number of estimators in the ensemble as the complexity in this case. This computation was more expensive and we had to settle on 100 bootstrap samples. The results is shown in figure 2, for which we see little to no difference regarding the choice of the classifier max depth. For all cases we see that the variance decrease with complexity while the bias increases. This is an opposite relation compared to our observations regarding a single decision tree for increasing depth. Thus when expecting the values further we see that bias is approximately on the same size as observed for the single decision tree, while the variance drops to about 20% of the variance plateau for the single decision tree case. Overall the boosting seems to decrease the variance which results in the lowest MSE given at approximately 0.075 compared to about 0.095 for the single decision tree.



(a) Max tree depth = 1

(b) Max tree depth = 2

(c) Max tree depth = 3

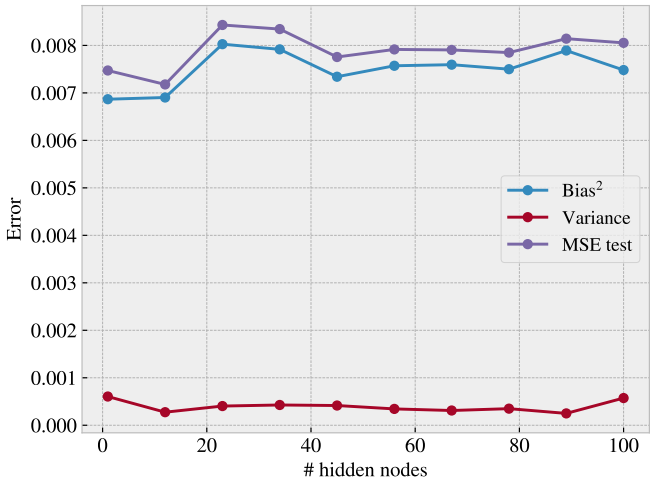
(d) Max tree depth = 4

Figure 2: Bias-variance trade-off for boosting (gradient-boosted-trees) with a varying number of estimators for a (weak) classifier of max depth 1,2,3 and 4 respectively.. For this analysis we used 100 bootstrap samples.

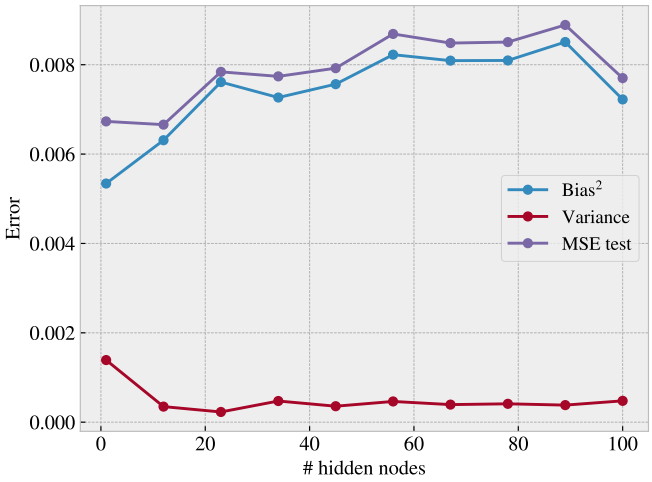
C. Feed Forward Neural Network

Finally we perform a similar bias-variance analysis for the feed forward neural network. Here we have two main candidates for the complexity: Number of hidden nodes, and number of hidden layers. Thus we perform a multiple of analysis, first with fixed number of layers and varying number of hidden nodes per layer (same for every layer) and then vice versa. We use again 100 bootstrap samples. The results are shown in figure 3 and 4. Watching the case of fixed number of hidden layers and increasing number of nodes (figure 3) we see that the bias carries most of the error. There seems to be a common trend of the bias dropping slightly while the bias increases slightly over the first few increments of the number of nodes. Apart from that the individual fluctuations does not seem to follow any straight forward trend. Going on to the case of fixed number of nodes and variable number of layers (figure 4) we still observe that most of the error lies in the bias. It is difficult to state anything about the trend as the complexity increases. Considering both result we found a few combinations where the total test MSE just dips below 0.006. While the bias seems to be about the same level as seen for the two previous method the difference appears to be the ability to reduce the variance.

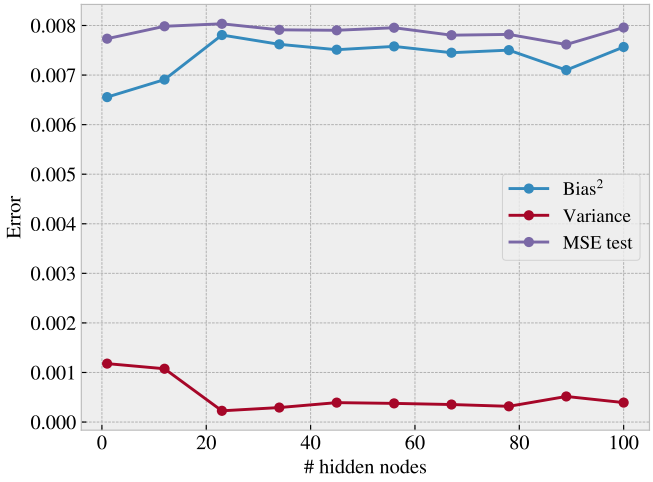
We would expect these results to be heavily influenced by the specifics of the data set, and thus it could be interesting to perform a similar investigation for a completely different data set.



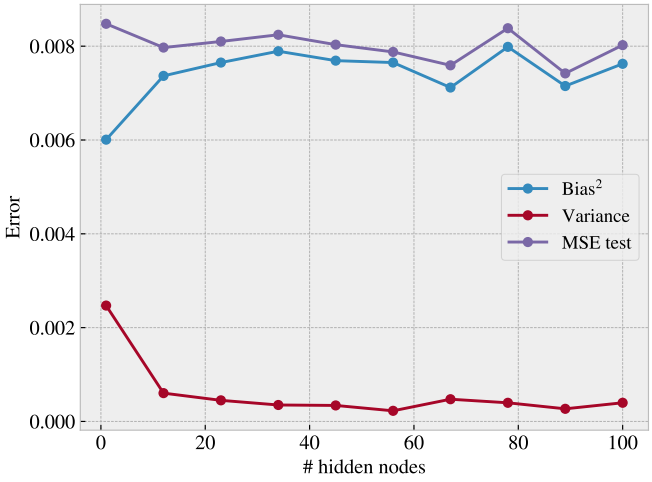
(a) Number of hidden layers = 1



(b) Number of hidden layers = 3

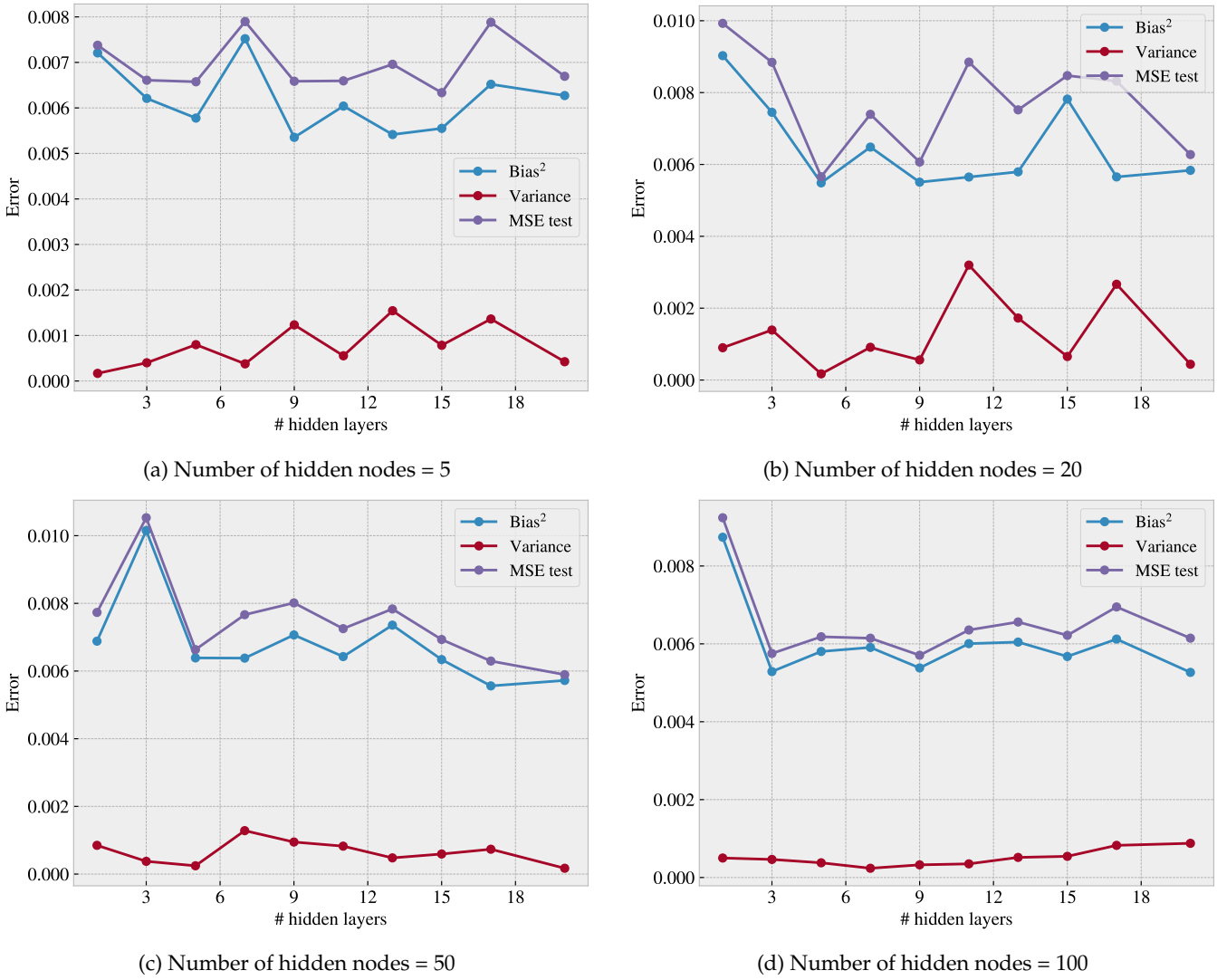


(c) Number of hidden layers = 5



(d) Number of hidden layers = 10

Figure 3: Bias-variance trade-off for a feed forward neural network with a varying number of hidden nodes for a network of 1, 3, 5 and 10 hidden layers respectively. For this analysis we used 100 bootstrap samples.



(a) Number of hidden nodes = 5

(b) Number of hidden nodes = 20

(c) Number of hidden nodes = 50

(d) Number of hidden nodes = 100

Figure 4: Bias-variance trade-off for a feed forward neural network with a varying number of hidden layers for a network with 5, 20, 50 and 100 hidden nodes on each hidden layer respectively. For this analysis we used 100 bootstrap samples.

V. CONCLUSION

When comparing the various method we found that the single decision tree followed a somewhat typical bias-variance trade-off where increasing the complexity would decrease the bias with a cost of the variance going up. For the boosting (boosted-trees) we more or less found an opposite relation where an increase in complexity would decrease the variance with the cost of the bias going up. The bias of these two methods was about the same size, however the variance was clearly reduced by boosting. When comparing the lowest MSE among the tested values the boosting method gave 0.075 while a single decision tree gave 0.095. When investigating the feed forward neural network we found an even greater reduction of the variance such that most of the error was carried by the bias. While we did not see any clear connections between the complexity, and the bias and variance respectively, we see about the same bias level as the other two methods which resulted in a slightly better lowest MSE being just below 0.006.

We suspect that our results is quite heavily influenced by the specifics of the data set, and thus it would be interesting to perform a similar investigation on a different data set.

In a future study of the problem it would be interesting to repeat the analysis for a more complex problem. Given that we found an ideal tree-depth of 3, one might be interested in studying if our findings are specific to this relatively "simple" problem or if they are in did general.