Report Project 3 - Additional Exercise

Erlend Øydvin
(Dated: December 17, 2021)

Abstract

In this exercise I have investigated the bias-variance trade-off for three different regression methods for a weather radar image. The methods used was a polynomial fit using Linear regression (OLS and Ridge), a Neural Network and a Decision tree. The results show that the polynomial model has both high bias and variance and is thus not a good model for this problem. The neural network and decision tree generates solutions with lower error and seems to not adapt so much to noise for complex models.

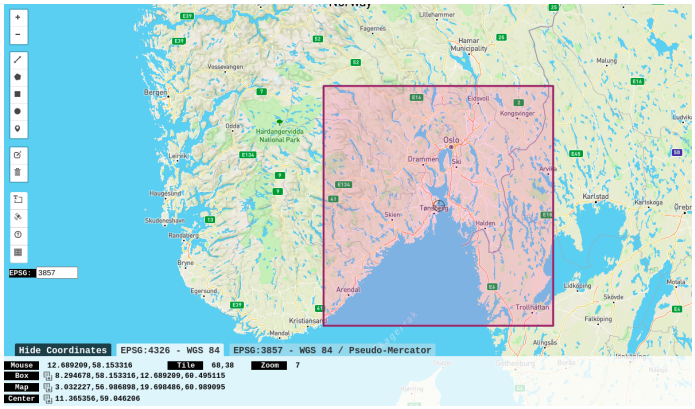

FIG. 1. Study area, a 253x215 km radar grid over Eastern Norway (red rectangle).

I. INTRODUCTION

In this report we have studied a regression problem where we try to fit a model to a surface using three different regression techniques, OLS and Ridge, a neural network and a decision tree. The main objective is to study the different methods bias variance trade-off. Studying bias variance trade-off is useful for modeling and studying physical problems as it can help distinguish between what is the bigger trend in the data and what is noise.

Weather radar data was downloaded from Norwegian Meteorological Institute's OPENDAP service Thredds. The python library netCDF4 was used to download a 253x215 km grid over Eastern Norway. The study area with radar grid can be seen in Figure 1. The precipitation event used in this project was on 8th of September 2018 at 5am. The pyhton code can be found at the following git-repository: `https://github.com/erlnwind/FYS-STK4155/tree/main/Project_3`

II. METHODS AND THEORY

Measuring precipitation

A weather radar works by sending out an electromagnetic pulse. Reflectively is the amount of transmitted power returned to the radar receiver after hitting precipitation. The more precipitation, the more power is reflected back. Reflected power can thus be related to rain rate.

Error

The main goal for the regression problem is to minimize the mean squared error

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

where y contains reference data and $\tilde{y}$ contains an approximation to y.

OLS and Ridge

In project 1 it was shown that an analytical solution to the OLS and Ridge regression problem could be found by solving the following equation

$$\vec{\beta} = (A^T A - \lambda I)^{-1} A^T \vec{y} \tag{2}$$

where the new parameter λ is a parameter that determines the shrinking effect on β. Setting λ equal zero amounts to finding the OLS solution. I is the identity matrix.

Neural network

The first step in a Feedforeward Neural Network is to compute the weighted sum of the input features for each neuron in the input layer as follows

$$z_j^l = \sum_{i=1}^{F} w_{ij}^l x_i + b_j^l. \tag{3}$$

Then each input z_j is passed through a activation function

$$a_j^l = f(z_j^l). \tag{4}$$

Then this process is repeated for every layer l in the network up to the output layer denoted by L. In the output layer z_j^L is calculated for each neuron and then passed to the output function. For the regression problem the output function is given by the linear function

$$a_j^L = f(z_j^L) = z_j^L, \tag{5}$$

After doing doing the feed forward we compute the predicted error and update the weights accordingly. This is

done with the back-propagation algorithm. The first step in back-propagation is to calculate

$$\delta_j^L = f'(z_j^L)\frac{\partial \mathcal{C}}{\partial (a_j^L)} \tag{6}$$

for the output layer where the cost function C is given by Equation 1 and $f'(z_j^L)$ is the derivative of the activation function. Then the next steps is to compute

$$\delta_j^l = \sum_k \delta_k^{l+1} w_{kj}^{l+1} f'(z_j^l) \tag{7}$$

for all layers back to the input layer. Back-propagation constructs new weights and we can update the weigths and biases as follows

$$\begin{aligned} w_{jk}^l &\leftarrow= w_{jk}^l - \eta \delta_j^l a_k^{l-1}, \\ b_j^l &\leftarrow b_j^l - \eta \frac{\partial \mathcal{C}}{\partial b_j^l} = b_j^l - \eta \delta_j^l, \end{aligned} \tag{8}$$

Decision Tree

A decision tree is divided into a root node, the interior nodes and leafs. Each node specifies a test that decides which branch (classification) a given instance belongs to. The leafs provides the final classification of an instance. A tree is built by doing a binary splitting in the root node of the data such that we obtain the lowest MSE given by

$$\sum_{i:x_i \in R_j} (y_i - \overline{y}_{R_1})^2 + \sum_{i:x_i \in R_2} (y_i - \overline{y}_{R_2})^2 \tag{9}$$

The same process is repeated further up the tree, for every node, we do another binary splitting of the data. The process is complete when we reach a given tree depth or all data has been given an individual category.

Parameters and hyper parameters

Each of the three methods have their own parameters and hyper-parameters that can be tuned and experimented with in order to obtain different predictions. In regression we can experiment with the learning rate, the regularization term and the polynomial degree. In the neural network we can in addition experiment with the choice of activation function, number of epochs and the architecture of the neural network. For the decision tree we can change the complexity of the model by the depth of the tree, this also becomes a hyper-parameter.

K-fold cross-validation

In machine learning it is common to split the dataset into a train group and a test group. The algorithm is first trained against the train group and the tested against the test group. This routine can expose over-fitting of data. In this report we have used k-fold cross validation. k-fold cross validation works by shuffling the data-set and dividing it into into k groups. Then one group is taken out as test group and the remaining groups are used to train the algorithm. This is repeated for all groups. The resulting R2 is obtained by taking the mean R2 of all groups.

III. RESULTS

In order to make OLS regression not produce errors due to singular matrices, the data have been centered and scaled by dividing by the variance. The centering is also necessary for Ridge regression to not constrain the first regression parameter. Centering and scaling was then done with all methods so that the results are more comparable.

Data

The precipitation event used in this project are from the 8th of September 2018 at 5am in the area presented in Figure 1. A high-resolution 253x215 km grid over Eastern Norway was downloaded from Norwegian Meteorological Institute's OPENDAP service. High resolution images seems to make the polynomial regression technique not produce any over-fitting. Since the goal of this report was to study the bias-variance trade-off of different regression methods the image was scaled down to a scale where the polynomial fit more easily produces over-fitting. In this case I scaled down the picture to 16x14 cells, simply by only using every 16th cell. The two radar images are shown in Figure 2. Reflectively is recorded in dBZ (Decibel relative to Z), which is a logarithmic and dimensionless unit. Thus the values in Figure 2 can be negative.

Regression using a polynomial

Ridge regression and OLS method from scikit-learn was used to fit a polynomial of different degrees to the low resolution radar image presented in Figure 2. The mean squared error was computed for different polynomial degrees using k-fold cross validation. The results are shown in Figure 3. Here we can observe that OLS suffers from overfitting already from polynomial degree 3. Increasing the regularization parameter seems to prevent this overfitting. Increasing the model complexity first reduces the bias thus lowering the MSE. As the model starts to adapt to noise the polynomial model starts to overfit, increasing the variance.

Regression using Neural Network

The MLP regressor (Neural Network) from scikit-learn was used to predict the low resolution radar image. The ReLU activation function with the adam solver was used. In order to study the bias-variance trade-off the Neural Network complexity was restricted to having only two hidden layers. Then complexity was added by gradually increasing the number of neurons in both layers. The mean squared error was computed for different architectures using k-fold cross validation. The results are shown in Figure 4, where

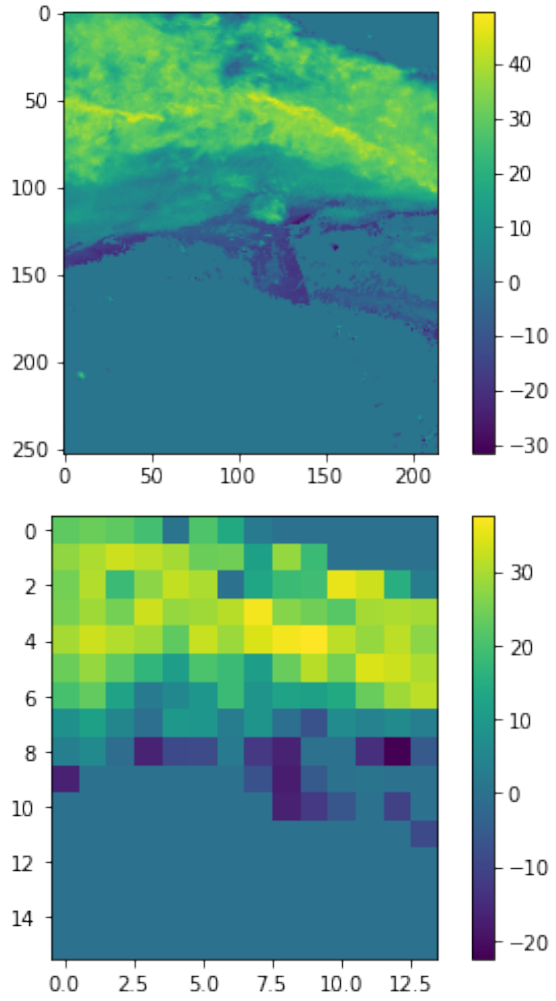

FIG. 2. Upper: High-resolution radar reflectivity for precipitation event. Lower: Low res-resolution radar reflectivity for precipitation event.

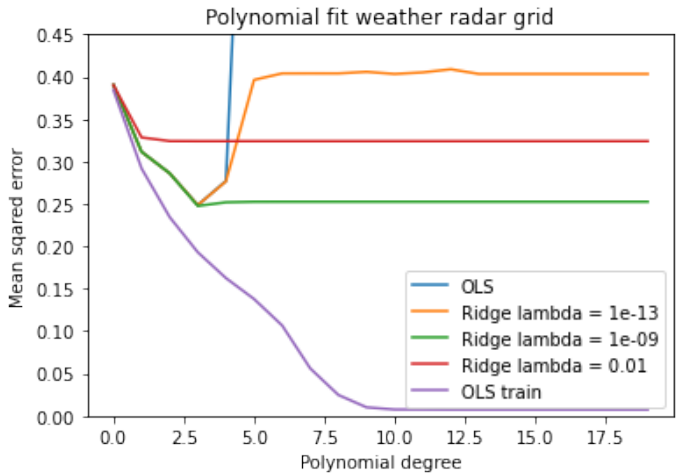

FIG. 3. k-fold cross-validation error for test group for different polynomial degrees.

the labels along the x-axis indicate the number of neurons in each layer. In Figure 4 we can observe that increasing the number of neurons reduces the mean squared error, decreasing the bias. Increasing the number of neurons further does not seem to split the train and test groups from each other, indicating that the network is not adapting to noise.

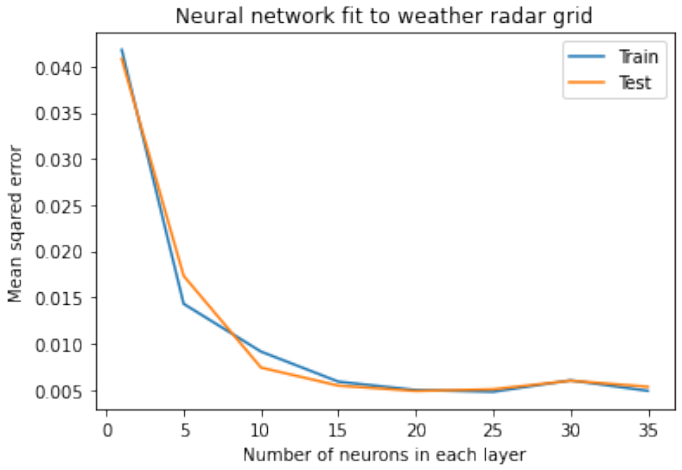

FIG. 4. Mean squared error for a Neural network fit to a weather radar grid.

Regression using a Decision Tree

The decision tree regressor from scikit-learn was used to predict the low resolution radar image. The complexity of the model was represented by the depth of the tree, the deeper tree the more complex model. The mean squared error was computed for different tree depths using k-fold cross-validation. The results are shown in Figure 5. Here we can observe that the error is reduced for increased model complexity, decreasing the bias. After tree depth of 10 the decision tree seems to perfectly fit the train data, reducing the MSE to zero. The test dataset have a larger MSE, indicating that the model is adapting to noise.

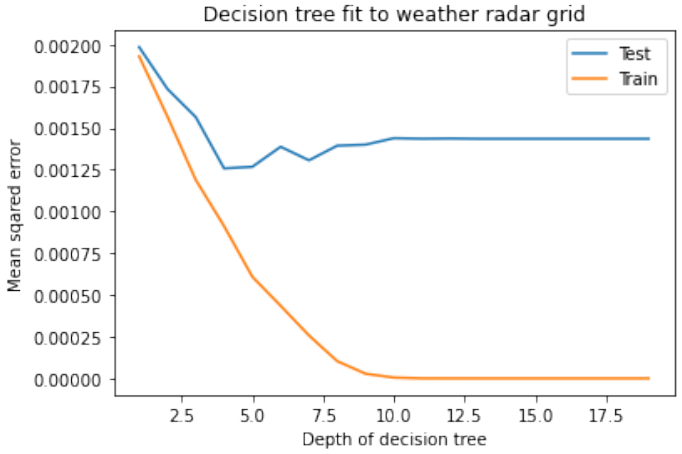

FIG. 5. Mean squared error for a Decision Tree fit to a weather radar grid.

IV. DISCUSSION

OLS regression is exposed to over-fitting and does not handle very well complex polynomials. This could be due to the polynomial model being not very suitable to model this surface and that OLS regression is exposed to overfit the data. Including the Ridge regularization prevents over-fitting of the data, but also restricts the spread in regression coefficients thus increasing the bias and the error. The polynomial model can be good for finding physical relations, but does not seem like the best model for this surface as it has a high bias. The neural network method seems to better model the surface when the number of neurons is increased. After a about 15 neurons the error does not seem to decrease more. An interesting aspect here is that the train and test dataset generates approximately the same MSE, indicating that the neural network is not adapting to noise and thus not overfitting the data. The neural network thus seems to have a low bias and a low variance. The decision tree method also reduces bias for increased model complexity. The MSE for the test group seems to go down before going up again around polynomial degree 5. This could indicate that the model is exposed to model some noise and thus has a slightly higher variance for deeper trees.

V. CONCLUSION

The weather radar grid was modeled using three different regression techniques. The linear regression model with a polynomial fit seems to not be very suitable to model this surface. The neural network and the decision tree seems able to provide both a low variance and a low bias.