On Linear Regression Methods for a Terrain Dataset

Torstein Solheim Ølberg
Institute for Theoretical Astrophysics, University of Oslo
P.O. Box 1029 Blindern 0315 Oslo, Norway

October 10, 2020

Contents

1 Abstract

Performing analysis in machine learning is a powerful tool. Here we look at three methods for approximating a 3D data set, Ordinary Least Squares, Ridge and Lasso regression. Use MSE, Bootstrap resampling and Cross Validation resampling to estimate which models fit a made up dataset and a terrain dataset the best and find that the 20th degree OLS fit best.

2 Introduction

The field of data analysis has been revolutionised by the introduction of machine learning in the middle of the 20th century. This process revolves around the approximation of models to some set of data by the brute force use of computers. In this report we will try study three simple linear regression models, the Ordinary Least Squares (OLS), Ridge and Lasso regression methods, on a made up data set and then use it on a real dataset and try to find the best possible method.

3 Data Description

The first dataset we make ourselves, produced by the so called Franke function. This is a function consisting of the sum of four exponentials and produces a plane dataset since it takes two separate variables x and y.

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

We choose to make the dataset with x and y values from $[0,1]$, and choose linearly 10^4 pairs of points. We should also make the dataset more realistic by adding some noise to the data. For this we choose to draw a set of random numbers, equal to the number of data points, drawn from a normal distribution with mean value 0 and standard deviance equal to 10^{-4}. Then we add these points to our Franke function values.

The second, and "real" dataset is taken from the United States Geological Survey [1]. Here we find an elevation dataset from the SRTM 1 Arc-second global group from the region around the Oslo Fjord. It is simply a measurement of the height in meters above sea level. We extract the small part in the top of the left corner showing Finnemarka. You can see the dataset in figure (1)

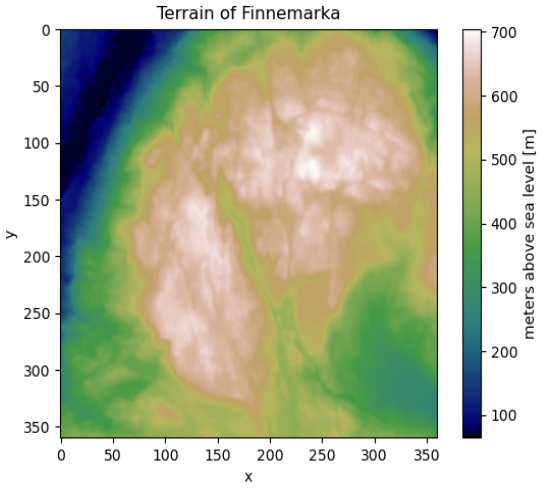

Figure 1: The Finnemark elevation dataset

4 Methods

4.1 Scaling

It is useful to scale our data before we perform any approximation of our dataset, as this will help us to account for problems with outliers and particularly large values when we fit. We will use the simple standard scaling from *Sklearn.preprocessing* called scale. This subtracts the mean value from the data points.

4.2 Building the Model

When performing a linear regression, we want a model on the form

$$z = \sum_{i=0}^{p}\sum_{j=0}^{i}\beta_{\sum_{k=0}^{i}+j}x^{i-j}y^{j} \tag{2}$$

This might look odd, but is just a compact way of writing a standard 2 dimensional p degree polyno-

mial. for $p = 2$ this is simply

$$z = \beta_0 + \beta_1 x + \beta_2 y + \beta_3 x^2 + \beta_4 xy + \beta_5 y^2$$

To perform this regression we need to set up a matrix X on the form

$$\vec{X} = \begin{bmatrix} \sum_{i=0}^{p} \sum_{j=0}^{i} x_0^{i-j} y_0^j \\ \sum_{i=0}^{p} \sum_{j=0}^{i} x_1^{i-j} y_1^j \\ \cdots \\ \sum_{i=0}^{p} \sum_{j=0}^{i} x_n^{i-j} y_n^j \end{bmatrix} \tag{3}$$

where n are the number of x, y pairs in the dataset. Then we can split our dataset into two groups, a training and a test group. We use 20% of our dataset as a test and the rest as training group.

4.2.1 Ordinary Least Squares

Now we are prepared to model our data with the OLS method. This is a simple task where we approximate the β values by finding the minimum value of the cost function

$$C(\vec{\beta}) = \frac{1}{n} \left\{ \left(\vec{y} - \vec{X}\vec{\beta} \right)^T \left(\vec{y} - \vec{X}\vec{\beta} \right) \right\}$$

It is simple to show, by taking the derivative of C that the minimum point can be found from

$$\vec{\beta} = \left(\vec{X}\vec{X}^T \right)^{-1} \vec{X}^T \vec{y} \tag{4}$$

as long as $\vec{X}\vec{X}^T$ is transposable. If they are not, we can approximate them by the Moore-Penrose pseudo-inverse instead, and use this result.

4.2.2 Ridge Regression

Another way to perform the Linear regression is the Ridge method. This is almost the same as the OLS, except the introduction of a constant λ times the norm of β into the cost function.

$$C(\vec{\beta}) = \frac{1}{n} \left\{ \left(\vec{y} - \vec{X}\vec{\beta} \right)^T \left(\vec{y} - \vec{X}\vec{\beta} \right) \right\} + \lambda ||\beta||_2^2$$

This gives us a way to find beta in the same way as for the OLS except with a λ times an identity matrix

$$\vec{\beta} = \left(\vec{X}\vec{X}^T + \lambda \vec{I} \right)^{-1} \vec{X}^T \vec{y} \tag{5}$$

With the same assumption as for the OLS.

4.2.3 Lasso Regression

The last regression method is the Lasso regression, where we define the cost function to be equal to

$$C(\vec{\beta}) = \frac{1}{n} \left\{ \left(\vec{y} - \vec{X}\vec{\beta} \right)^T \left(\vec{y} - \vec{X}\vec{\beta} \right) \right\} + \lambda ||\beta||_1$$

where we have used $||\beta||_1 = \sum_i |\beta|$. This doesn't have a numerical solution to the derivative of C like the other two methods. However, the python package *sklearn* has a useful method *linear_model.Lasso* which we can use to perform this minimalisation task for us.

4.3 Analysis methods

We need to find a way to measure which method is the best at approximating our dataset, and to do this a simple measure might be to look at the Mean Square Error and use this as our prediction error.

4.3.1 Mean Square Error

The MSE is simply the mean value of the square of the errors between the approximations and the measured values.

$$MSE = \frac{1}{n} \sum_i (z_i - \hat{z}_i)^2 \tag{6}$$

where z are the measured values and $\hat{z}$ are the true values.

4.3.2 R Squared

We can also calculate the R^2 value of the model, which is defined as

$$R^2 = 1 - \frac{\sum_i (z_i - \hat{z}_i)^2}{\sum_i \left(z_i - \bar{\hat{z}} \right)^2} \tag{7}$$

$\bar{\hat{z}}$ is the mean value of the approximations.

4.3.3 Bootstrap Resampling

A more complicated, but still quite simple way to find the best fit model is to perform the bootstrap resampling method. This method is based on the assumption that we can perform the prediction of our data a number of times a, where we each time choose a random subset of the test and training group with replacing. Thus we get a new train and test groups, with the same size as the original once, and we can find the MSE for each and averaging over all the MSEs. From this we can also split the MSE into three other parts, the Bias, the Variance and the variance of the error. To get this, we can look at the MSE

$$\begin{aligned} MSE =& \frac{1}{n}\sum_i (z_i - \hat{z}_i)^2 \\ =& \mathrm{E}\left[\left(\vec{z} - \hat{\vec{z}}\right)^2\right] \end{aligned}$$

where E denotes the expected value. If we add $E[\hat{\vec{z}}] - E[\hat{\vec{z}}]$ inside the square and perform the squaring, we can rewrite this as

$$\begin{aligned} MSE =& \mathrm{E}\left[\left(\hat{\vec{z}} - E[\hat{\vec{z}}]\right)^2 \right. \\ &+ \left(\vec{z} - E[\hat{\vec{z}}]\right)^2 \\ &\left. +2\left(E[\hat{\vec{z}}] - \vec{z}\right)\left(\hat{\vec{z}} - E[\hat{\vec{z}}]\right)\right] \end{aligned}$$

The estimated value of an estimated value is simply the estimated value, and the estimated value of a measured value is simply the measured value. This lets us rewrite the above as

$$\begin{aligned} MSE =& \mathrm{E}\left[\left(\hat{\vec{z}} - E[\hat{\vec{z}}]\right)^2\right] \\ &+ E\left[\left(\vec{z} - E[\hat{\vec{z}}]\right)^2\right] \\ &+ 2\left(E[\hat{\vec{z}}] - \vec{z}\right)\left(E[\hat{\vec{z}}] - E[\hat{\vec{z}}]\right) \\ =& \mathrm{E}\left[\left(\hat{\vec{z}} - E[\hat{\vec{z}}]\right)^2\right] \\ &+ E\left[\left(\vec{z} - E[\hat{\vec{z}}]\right)^2\right] \\ =& \mathrm{Var}(\hat{\vec{z}}) + \sigma^2 + \mathrm{Bias}(\vec{z}, \hat{\vec{z}})^2 \end{aligned}$$

The Last step her is just splitting the left term into the variance of the model and the variance of the assumed noise. This result gives us the bias and the variance

$$\mathrm{Bias} = \frac{1}{n}\sum_i \left(z_i - E[\hat{\vec{z}}]\right)^2 \tag{8}$$

$$\mathrm{Var} = \frac{1}{n}\sum_i \left(\hat{\vec{z}} - E[\hat{\vec{z}}]\right)^2 \tag{9}$$

As we can see, the bias is simply the average of the difference between the measured value and the estimated approximation value. The Variance is the same, but the measured value is exchanged for the approximated value.

4.3.4 Cross Validation

We can then calculate the MSE, bias and variance and plot them as a function of our polynomial degree, to find at what value the bias and variance has the optimal values.
Last of all, we can perform the so called Cross Validation resampling method (CV). Here we shuffle and perform a random split of data into test and training groups a number of times N and calculate the MSE every time. Then we find the mean of this value and use this as our MSE estimate.

4.4 Performing the analysis

First We will perform the analysis methods for OLS, finding the best fit polynomial degree. Then we will find the optimal lambda for Ridge and Lasso, using a degree of 5. This we do for the Franke function to test our method, afterwords we can perform the same thing for the terrain data. Last, we use this lambdas for the terrain data to find the optimal degree and compare the best fit models for OLS, Ridge and Lasso.

5 Results

Performing the OLS estimate of the Franke dataset with a fifth degree polynomial, we get an R^2 value of 0.9927 This also gives us the following visualisation of the approximation

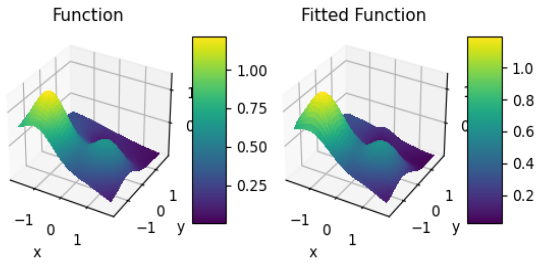

Figure 2: The Franke and the fitted approximation with OLS

To the left in figure (2) we see the actual Franke function and to the right we see the 5th order approximation with OLS.
For the OLS we get a MSE as a function for the polynomial degree for both the training and test data.

Figure 3: MSE as a function of polynomial degree for both the test and train data set groups

We see that both dataset groups MSEs get reduced with higher polynomial degree until we reach around 20 and then they both rise up, but the test rises faster.
The Bootstrap for the OLS gives us a plot of the MSE, Bias and Variance.

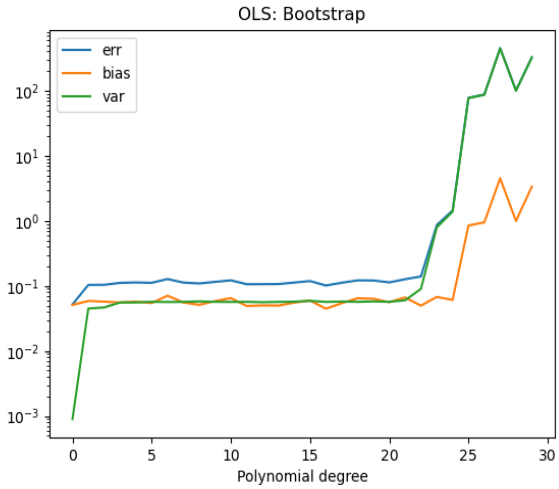

Figure 4: Bootstrap MSE, bias and variance for OLS. Rises in value after 20th degree.

They all stay quite low until after 20th degree where they rise up.
The CV gives us a MSE which we can compare with the MSE of Bootstrap

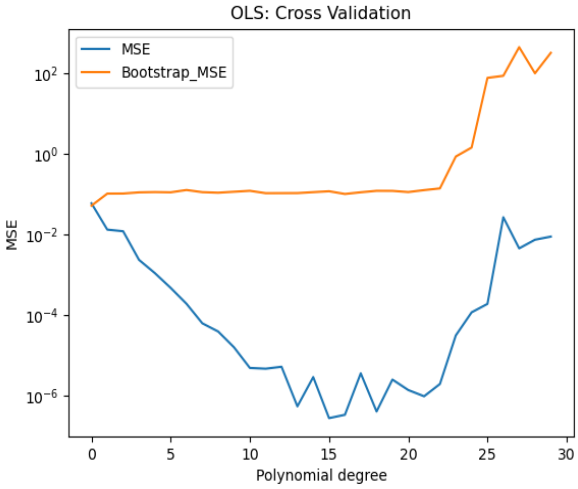

Figure 5: CV MSE estimate for OLS compared with the Bootstrap MSE. The CV has a minimum which the Bootstrap doesn't have

We see that the CV gives us a minimum point that the Bootstrap didn't.
For the Ridge we get a MSE as a function for the lambda for both the training and test data

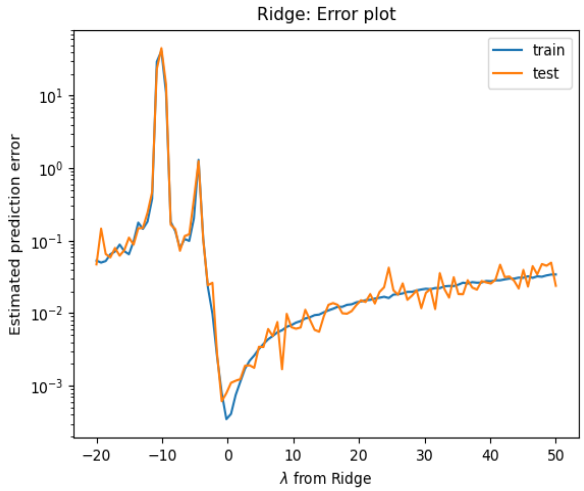

Figure 6: MSE as a function of lambda for both the test and train data set groups

We see that both dataset groups MSEs look mostly the same. They fluctuate a little until they reach a minimum around zero and rise afterwards slowly up.
The Bootstrap for the Ridge gives us a plot of the MSE, Bias and Variance.

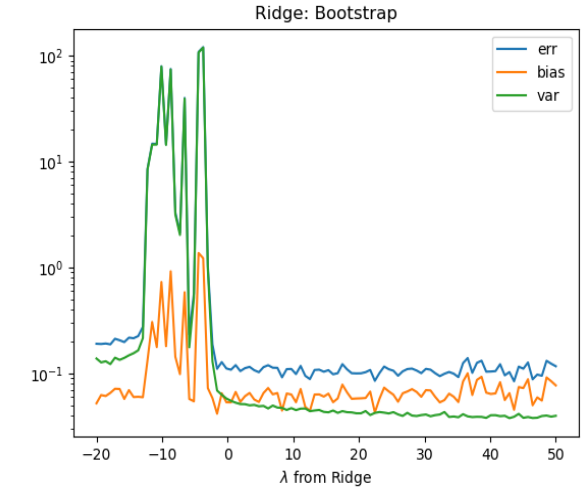

Figure 7: Bootstrap MSE, bias and variance for Ridge. After lambda equal to zero they all stabilise at a low value.

We see all three quantities fluctuate for negative values of lambda, but stabilise after zero for quite a low value.
The CV gives us a MSE which we can compare with the MSE of Bootstrap

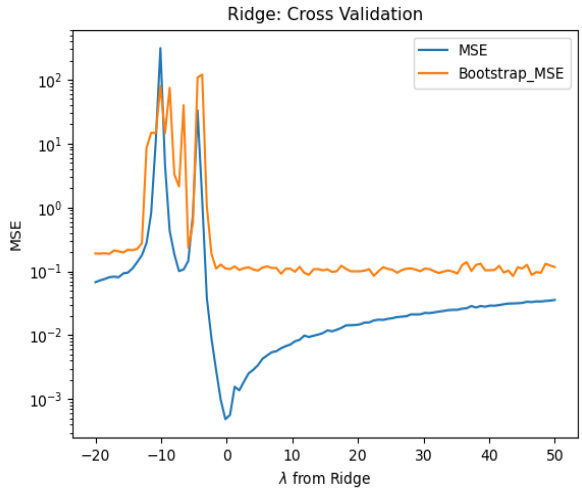

Figure 8: CV MSE estimate for Ridge compared with the Bootstrap MSE. The CV has a minimum which the Bootstrap doesn't have

We see that the CV gives us a minimum point that the Bootstrap didn't at lambda around zero. but it doesn't stabilise after zero but rather looks like it converges to the same value as Bootstrap for some very high lambda.
For the Lasso we get a MSE as a function for the polynomial degree for both the training and test data.

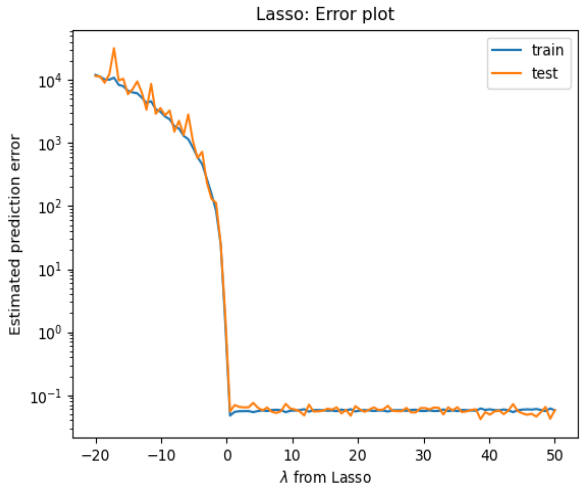

Figure 9: MSE as a function of lambda for both the test and train data set groups

We see that both dataset groups have the same evolution, falling down at zero and staying the same after this

The Bootstrap for the Lasso gives us a plot of the MSE, Bias and Variance.

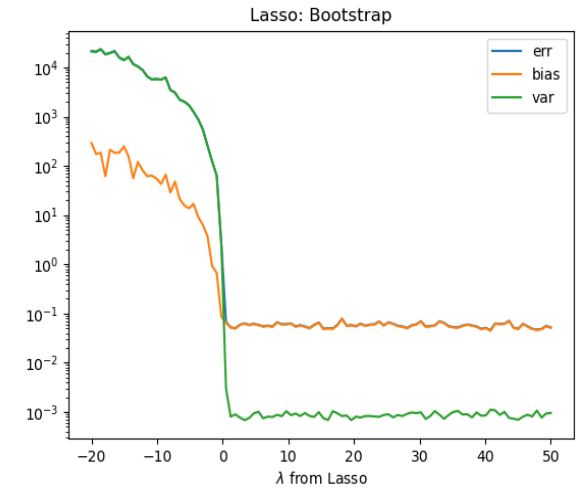

Figure 10: Bootstrap MSE, bias and variance for Lasso. Has the same general form as the error plot, but we can't see the error because it follows the other values so much.

Varians starts high but drop off a lot at zero. Until then, the error follows this. Then, at lambda equal to zero, the bias passes the variance and the error starts following this.
The CV gives us a MSE which we can compare with the MSE of Bootstrap

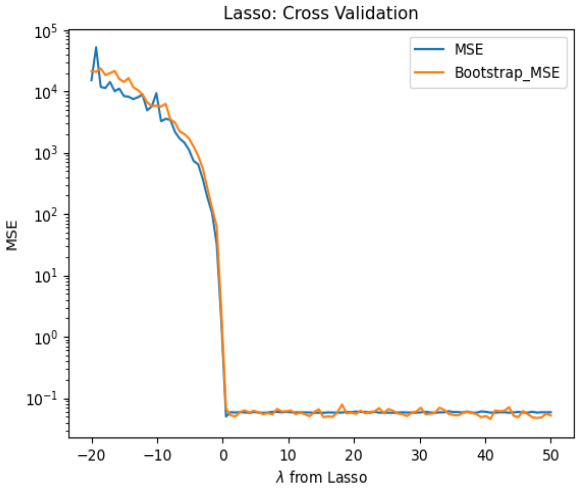

Figure 11: CV MSE estimate for Lasso compared with the Bootstrap MSE. The CV and Bootstrap look the same.

We see that the CV and Bootstrap gives us the same form.
For the OLS we get the following analysis plots

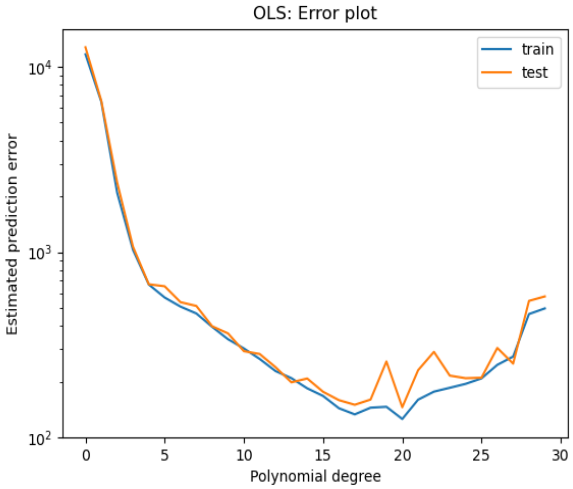

Figure 12: MSE for OLS for both training and test groups with the terrain data set

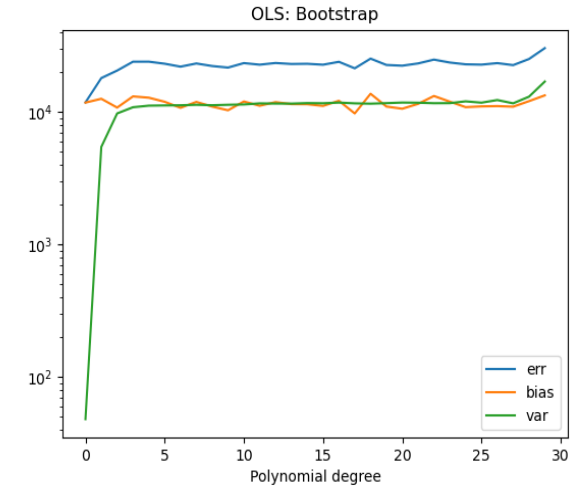

Figure 13: MSE, Bias and Variance for OLS with the terrain data set

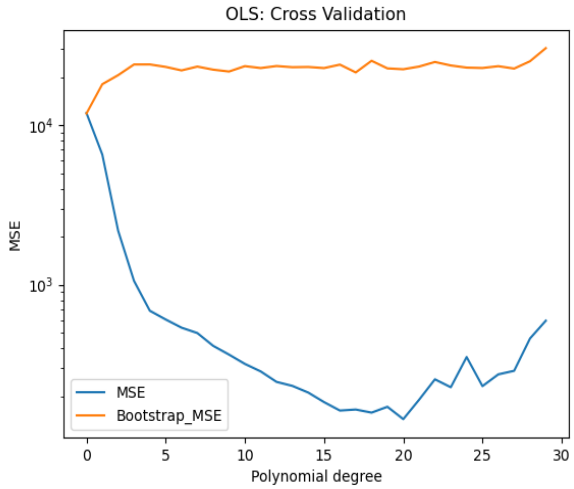

Figure 14: MSE for OLS for both CV and Bootstrap with the terrain data set

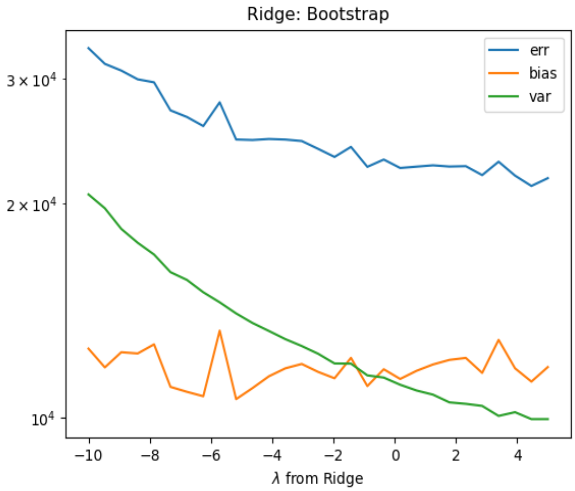

Figure 16: MSE, Bias and Variance for Ridge with the terrain data set

We see that all three plots gives us a minimum at around the same polynomial degree, though the Bootstrap result is almost constant for all polynomial degrees.
For the Ridge we get the following analysis plots

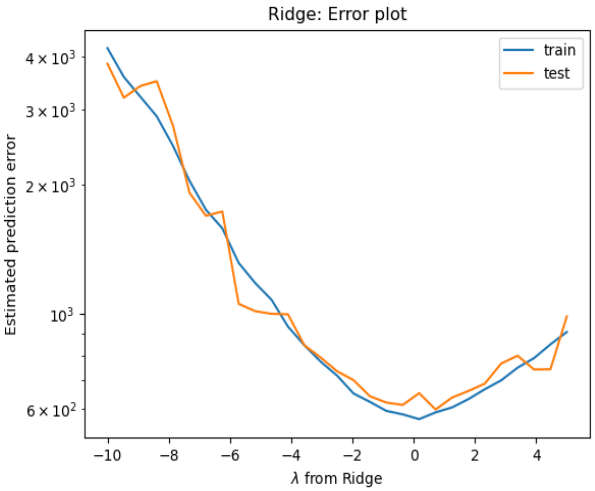

Figure 15: MSE for Ridge for both training and test groups with the terrain data set

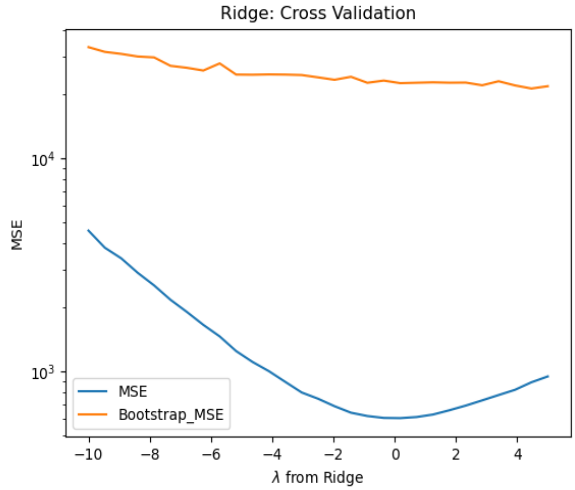

Figure 17: MSE for Ridge for both CV and Bootstrap with the terrain data set

We see that the error and CV gives us a minimum lambda value, but the Bootstraps just drops of for rising values.
For the Lasso we get the following analysis plots

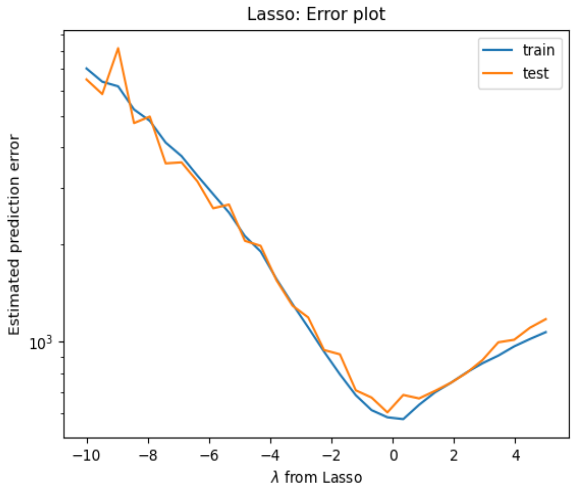

Figure 18: MSE for Lasso for both training and test groups with the terrain data set

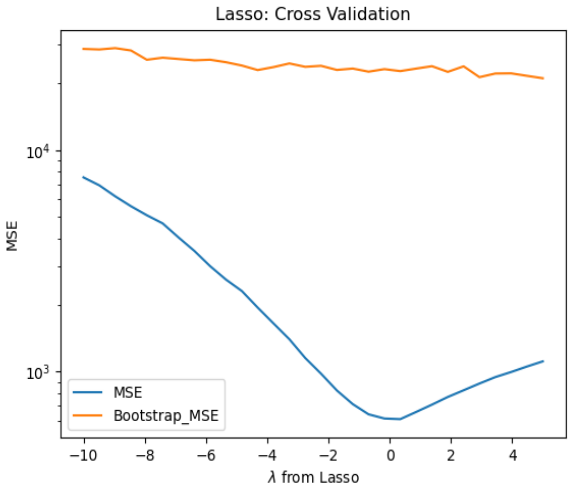

Figure 20: MSE for Lasso for both CV and Bootstrap with the terrain data set

We see that the error and CV gives us a minimum lambda value, but the Bootstraps just drops of for rising values just like we saw for Ridge.
Using these optimal values, we get the following analysis plots for Ridge as functions of polynomial degree.

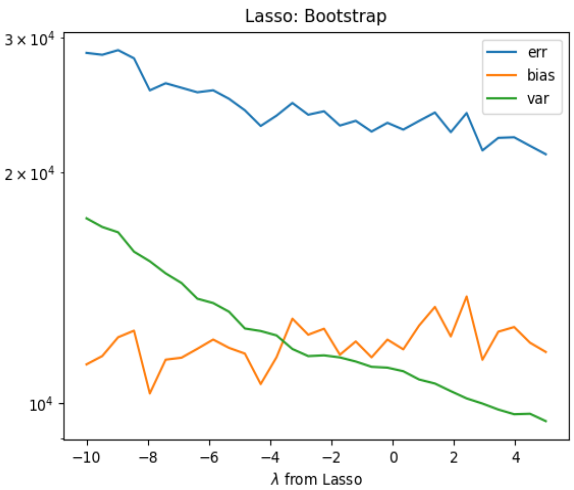

Figure 19: MSE, Bias and Variance for Lasso with the terrain data set

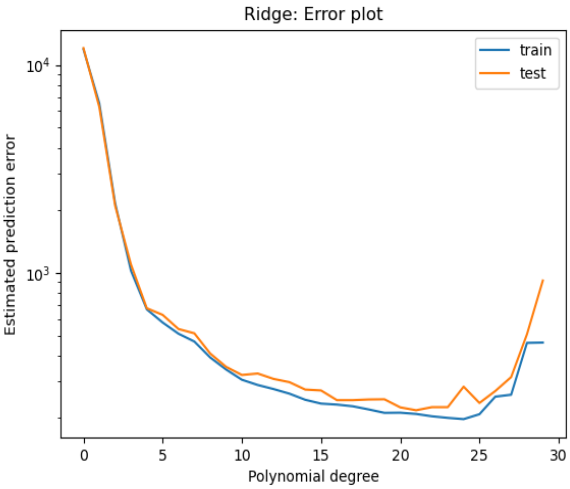

Figure 21: MSE for Ridge for both training and test groups with the terrain data set as a function of polynomial degree

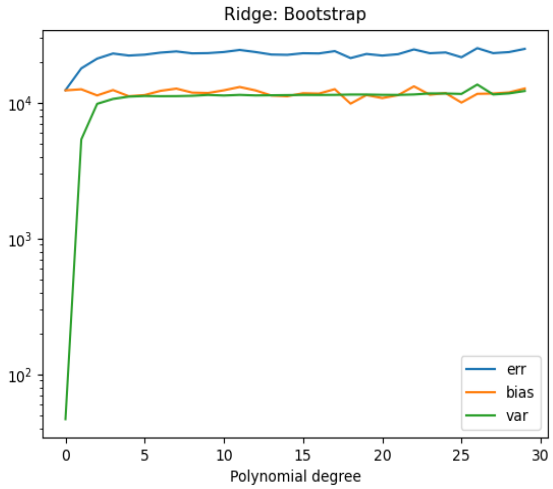

Figure 22: MSE, Bias and Variance for Ridge with the terrain data set as a function of polynomial degree

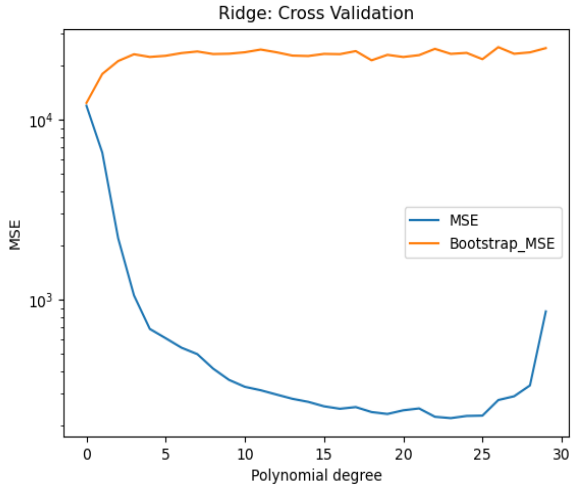

Figure 23: MSE for Ridge for both CV and Bootstrap with the terrain data set as a function of polynomial degree

We see that the error and CV gives us a minimum lambda value, but the Bootstraps stay constant for most of the plot.
For the optimal models we get the following estimated errors

	OLS	Ridge
MSE	188.6160408	227.8683254
Bootstrap MSE	28119.19033	24358.41955
CV MSE	157.1029735	225.2837576

6 Discussion

From the analysis of the Franke function we can see that all the different methods have a best fit value and in the error plots. However, the bootstrap results are odd for every regression method. Though the mean square error looks like it is the sum of the variance and the bias like we would expect, the variance follows the same evolution of the bias, instead of being the opposite like we would expect. As for the Cross Validation we see about the same evolution as the Error. As for the terrain data, we see almost the same things as for the Franke function, but we can still use the MSE and the CV to find an optimal lambda and polynomial degree. We find $p = 20$ as the optimal for OLS, $\lambda = 0.1$ and $p = 23$ for the Ridge and $\lambda = 0.2$ for the Lasso. However, for the Lasso regression, with the best fit lambda we couldn't get the Lasso regression to converge into a best fit, making this method unavailable. From the calculations of the errors of these best fits, we have that the OLS gives the best fit for our dataset.
A problem we have overlooked in this analysis is that we have not calculated confidence interval. This would have been a simple task, but I didn't have the time to do so.
Another problem is that we have chosen to perform the estimation of lambda first, and then the estimation of polynomial degree for the Ridge and Lasso. This was done to save time, as the calculations where very time consuming.

7 Conclusion

We saw that all the methods estimate a best fit for the Franke function, and for the terrain data, except that the Lasso doesn't converge for the polynomial degree. However we get a best fit for the other two, and from our best fit models we end up with the model best fitting our terrain data is the Ordinary Least Squares model of a 20th degree polynomial.

References

[1] United States Geological Survey;
Earth explorer
https://earthexplorer.usgs.gov/
Published 23/09-2014
Downloaded 10/10-2020