Project 1 - OLS, Ridge and Lasso Regressions

FYS-STK 4155

by
Ottesen, Dan Fredrik Clemp

Oslo, October 10, 2020
Gitbut repository url: `https://github.com/danfredrik/Project_1`

ABSTRACT

This report is structured as a comparative study between the accuracy of three different machine learning algorithms on two different data sets, with various pre-defined assumptions, and a secondary focus on computational efficiency. I find that Ridge Regression with bootstrapping outperforms both Ordinary Least Squares and Lasso Regression with bootstrapping and cross-validation on a small data set with smooth slope values for small tuning parameters, without being too costly in terms of computational time. For a large and noisier data set, I find that the simple OLS is preferable, because it produces the best results while keeping the run time considerably lower than the alternatives.

Contents

1 Introduction and assumptions

Linear regression methods are some of the most popular tools in statistics, and is one of the core methodologies in academics and business as it is both easy to apply and interpret. In this report, I will be focusing on the Ordinary Least Squares (OLS) method, comparing that to two other methods, the Ridge Regression and the Lasso Regression. I will run a simple OLS Regression, while running a Ridge Regression with bootstrapping and a series of different penalty terms and Lasso Regression with both bootstrapping and cross validation, also with a series of penalty terms. Due to the difference in methodology this is not meant as a side by side comparison of the performance of the three methods, but rather as an illustration of the trade off between performance and computational efficiency for linear methods in general. I will consistently be using a 80/20 train/test split throughout all analysis.
I will first talk about the methods in general terms, before describing the bootstrapping and cross validation procedures in detail. The first data test set of interest is run on two evenly spaced vectors between 0 and 1, with 20 data points, where the dependent variable is generated using Franke's Function, given by

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

The second data set is map data retrieved from EarthExplorer (USGS (2020)), where a subset of the minimum size data set is chosen for computational reasons. The are of choice has been randomly selected from rural Arizona. The two data sets vary in their complexity. While Franke's function gives a smooth shaped three dimensional data plot with one large and one smaller peak, the map data has considerably more noise, with less visible accentuation.

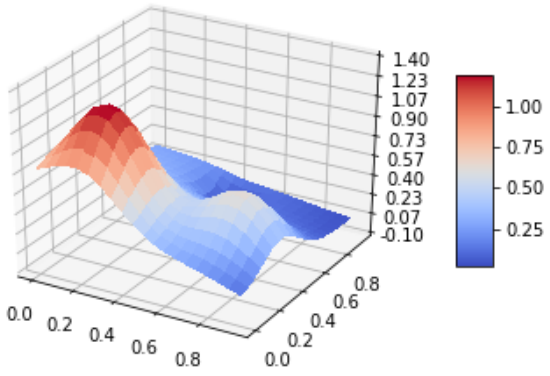

Figure 1: Plot of data points generated with Franke's function

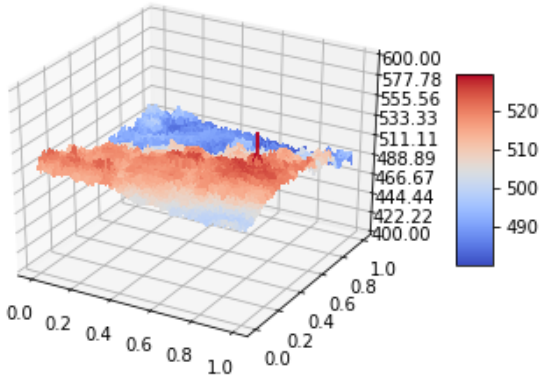

Figure 2: Plot of map data

1.1 Method for choosing most suited model

There are various existing determinants to measure model fit quality, but since my chosen regression methods are not directly comparable, I will be using Mean Squared Error (MSE) for brevity. The MSE is suited for comparison both since it is widely applied in practice and because it punishes large outliers, compared to methods which do not square the errors, such as the Mean Average Error.

In addition to the MSE I will be reporting computational times, given in seconds, for all model specifications. This is both to get a rough idea on how computationally heavy the methods are, but also to get a feel for how well suited they are for scaling to larger data sets, since the map data consists of considerably more data points than the data generated for Franke's Function.

The design matrix is consistently fitted with at most fifth degree polynomials. This is again a choice of brevity, since several regression specifications are biased as a result of underfitting.

1.1.1 Bias/Variance Trade-Off

For all of the regression specifications I will also provide a plot of the bias and variance for some choice of variable, such as degree of polynomials in the design matrix, or the size of the tuning parameters. We decompose the bias and variance after optimizing the MSE function, given by

$$
\begin{aligned}
&\mathbb{E}\left[(y-\hat{y})^2\right] \\
&\quad = \mathbb{E}([(y-\hat{f}(x))^2] \\
&\quad = \mathbb{E}[f(x)+\epsilon-\hat{f}(x))^2] \\
&\quad = \mathbb{E}[(f(x)-\hat{f}(x))^2] + \mathbb{E}[\epsilon^2] + 2\mathbb{E}[(f(x)-\hat{f}(x))\epsilon] \\
&\quad = \mathbb{E}[(f(x)-\hat{f}(x))^2] + \mathbb{E}[\epsilon^2] + 2\mathbb{E}[(f(x)-\hat{f}(x))]\mathbb{E}[\epsilon] \\
&\quad = \mathbb{E}[(f(x)-\hat{f}(x)^2] + \sigma^2 \\
&\quad = \frac{1}{n}\sum_i (f_i - \mathbb{E}[\hat{y}])^2 + \frac{1}{n}\sum_i (\hat{y}_i - \mathbb{E}[\hat{y}])^2 + \sigma^2
\end{aligned}
\tag{2}
$$

where $\frac{1}{n}\sum_i (f_i - \mathbb{E}[\hat{y}])^2$ refers to the bias, $\mathbb{E}[\hat{y}])^2 + \frac{1}{n}\sum_i (\hat{y}_i - \mathbb{E}[\hat{y}])^2$ is the variance, and finally σ^2 is a noise term. In practical terms, one would expect the bias to drop when adding new parameters due to increasing the complexity of the parameters. However, when complexity increases too much the variance is likely to 'explode' as a result of overfitting.

2 Regression methods

2.1 Ordinary Least Squares

2.2 Ordinary Least Squares[1]

The OLS method aims to causally measure the effect of matrix $\boldsymbol{X}$, of size $n \times p$ on a vector $\boldsymbol{y}$ of size $n \times 1$. p corresponds to the number of predictor variables on the response variable y. With the caveat of several assumptions being in place, such as linear, homoskedastic, independent and normally distributed errors $\boldsymbol{\epsilon}$, one can calculate a model

$$\boldsymbol{y} = \boldsymbol{\beta X} + \boldsymbol{\epsilon}$$

where the objective is to minimize the parameter ϵ through finding the optimal β, minimizing the cost function

$$C(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n-1} (y_i - \hat{y}_i)^2 \tag{3}$$

where one through substitution can show that equates to

$$C(\boldsymbol{\beta}) = \frac{1}{n}\{(\boldsymbol{y} - \boldsymbol{X\beta})^T(\boldsymbol{y} - \boldsymbol{X\beta})\} \tag{4}$$

where if you take the derivative of $C(\beta)$ with respect to β you get that the optimal solution is given by

$$\boldsymbol{\beta} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{5}$$

[1]The source of all theory regarding all regression methods is Morten Hjort-Jensen (2020),slide set produced for FYS-STK4155

We then estimate this to obtain the model

$$\hat{y} = \beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i}, \quad i = 1, ..., n \tag{6}$$

2.2.1 Confidence intervals

Confidence intervals for parameters β are common in practice because they convey useful information about the confidence of the estimates and the formula is given by

$$\overline{X} = \pm t_{\alpha/2} \cdot \frac{\sigma}{\sqrt{n}} \tag{7}$$

A 95% confidence interval is chosen for this report, with 1.96 as the value for $t_{\alpha/2}$. The latter is a simplification, since one should choose this value using a t-table and the number of sample. It is however, close enough to not induce too much bias in the confidence intervals. In practical terms a 95% confidence intervals tells us that if we re-run the test with a different sample, the estimator is estimated to fall within the interval 19 out of 20 times, or in other terms that one is likely to estimate a parameter falling outside the interval 5 out of a 100 times.

2.3 Ridge Regression

The ridge regression function similarly to OLS, but adds a penalty term λ to the parameters, which is now given by

$$\boldsymbol{\beta} = (\boldsymbol{X}^T \boldsymbol{X} + \boldsymbol{\lambda} \boldsymbol{I})^{-1} \boldsymbol{X}^T \boldsymbol{y} \tag{8}$$

where $\boldsymbol{I}$ is a pp identity matrix. The cost function is now defined as

$$C(\boldsymbol{\beta}) = \frac{1}{n} \sum_{i=1}^{n-1} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p-1} \beta_j^2 \tag{9}$$

In practical terms the difference between OLS and Ridge regression is that a λ value of < 0 reduces the influence of small eigenvalue parameters, while a large value of λ

increase the bias. There is no correct value of λ and the user must find this through trial and error.

2.4 Lasso Regression

Least Absolute Shrinkage and Selection Operator (Lasso) deviates slightly from the Ridge regression in that the penalty term is defined in a slightly different way. One should still choose a penalty term λ, but the cost function is defined differently

$$C(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n-1}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p-1}|\beta_j| \tag{10}$$

A favorable feature with adding the absolute value of parameters is that it shrinks the influence of the least important parameters toward zero, and this is therefore a favored tool for feature selection.

2.5 Bootstrapping and K-fold Cross Validation

Bootstrapping and K-fold Cross Validation is commonly referred to as 'resampling methods'. In simple terms, bootstrapping refers to using the sample distribution to drawing replica sets with the same distribution, to increase the number of data points for analysis. The K-fold cross validation, which in this report is implemented without the validation step, refers to dividing the training set in k parts, where $k-1$ sets is used as training sets, fitted on the remaining set, so that all sets k are used as testing sets. Bootstrapping is particularly useful when you have small data sets and you can rightfully assume that the sample distribution is similar to the true distribution and cross-validation increases the robustness of analysis, since the user is able to conduct several tests on the same data set. In this report I will consistently choose 5 cross-validation sets and I will use varying numbers of bootstraps, depending on computational efficiency. Bootstrapping is computationally very expensive, meaning that one run the risk of increasing computation time exponentially when adding many bootstrap resampling. While this is advisable when running research projects, I've decided against this due to the goals this report has aimed to achieve.

3 Results

This section presents the results of analysis, with a special focus on MSE and run-time. Other results will be reported when prompted by the assignment.

3.1 Franke's Function

3.1.1 Ordinary Least Squares

	Train	Test
MSE	0.0020	0.0036
R^2	0.9760	0.9564
Run time	0.0016s	

The fit of the test set is poor compared to the training set, likely due to the lack of resampling. The test error is nearly double that of the train error, but the run time is almost zero, making it an efficient tool for investigative analysis. We can see how the fit looks compared to the original set by plotting it.

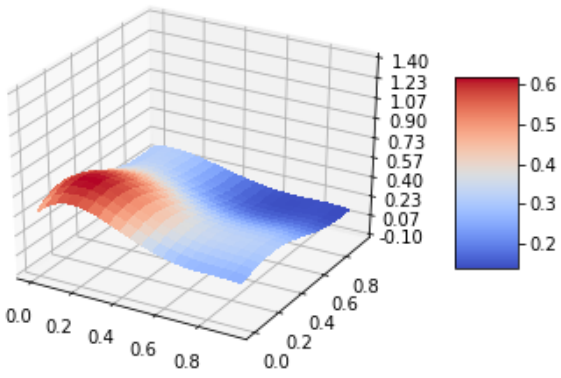

Figure 3: Plot of the fitted Franke's function

As can be seen, the fit fails to capture the height of the peaks, as the plot appears to be smoothed compare to the original set. This could be the result of underfitting or simply a consequence of chance, which also is likely given the lack of resampling. I will next report the β estimates along with their 95% confidence interval and the MSE plot for different degrees of polynomial fits.

Table 2: 95% Confidence Intervals for β Estimators

	2.5%	μ	97.5%
β_0	0.4318	0.4372	0.4425
β_1	1.4696	1.7102	1.9508
β_2	0.6927	0.9408	1.1888
β_3	-7.9506	-6.6918	-5.4330
β_4	-3.1887	-2.4791	-1.7695
β_5	-3.1661	-1.8589	-0.5516
β_6	3.0680	5.8707	8.6734
β_7	5.0500	6.3768	7.7036
β_8	1.9466	3.4125	4.8784
β_9	-6.5594	-3.6487	-0.7380
β_{10}	-1.7924	1.0450	3.8824
β_{11}	-8.3472	-7.0091	-5.6708
β_{12}	-1.4497	-0.2632	0.9233
β_{13}	-6.7086	-5.2259	-3.7432
β_{14}	5.9621	8.9118	11.8615
β_{15}	-3.2084	-2.1371	-1.0657
β_{16}	1.6924	2.2589	2.8254
β_{17}	0.4963	0.9808	1.4652
β_{18}	-1.3511	-0.8331	-0.3150
β_{19}	2.1800	2.8004	3.4208
β_{20}	-5.6001	-4.4844	-3.3688

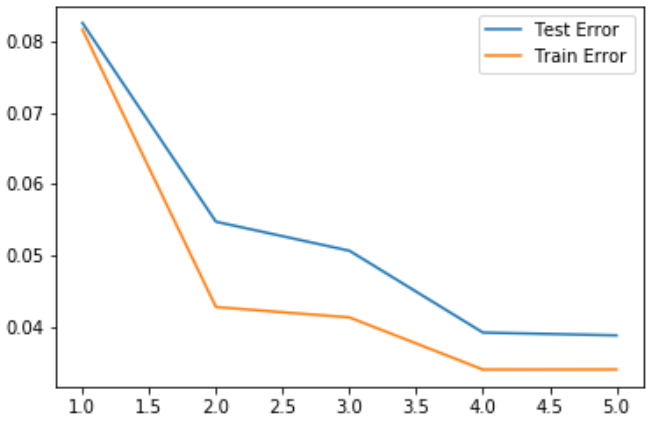

Figure 4: MSE as a function of polynomial degree

As can be seen from the MSE plot, the MSE is decreasing with the degree of polynomials, indicating that it is not overfitted, and can possibly be underfitted, and the test error is consistently underperforming the train error.

3.1.2 Adding random noise to the model

Sometimes adding noise to a model can improve the fit due to its regulazation effect. I have here added a random number between 0.0001 and 0.01 to the original data set to see if it may help the model improve its fit, illustrating it with a bias/variance plot. Since the variance is likely to be low, the results should not differ much from the MSE plot.

	Train	Test
MSE	0.0022	0.0022
R^2	0.9732	0.9693

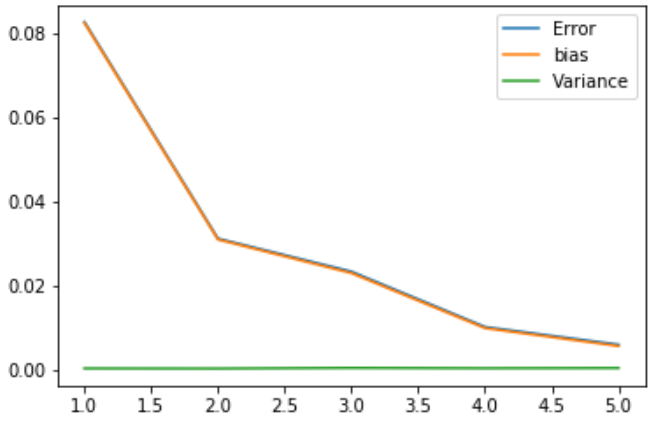

Figure 5: Bias/variance as a function of polynomial degree with noise

The table and the plot makes it clear that the model with noise is performing better than the model without noise.

3.1.3 Introducing bootstrapping

I re-run the regression with bootstrapping, using 500 bootstrap samples, and reverting to the data set without noise, showing that the bias/variance show consistent

results with the case of no bootstrapping, and that the model might be underfitted with only five degrees of polynomials.

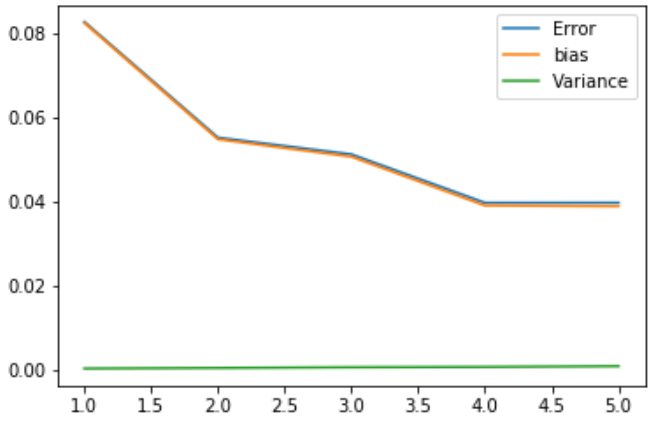

Figure 6: OLS bias/variance with 500 bootstraps as a function of polynomial degrees

3.1.4 Ridge Regression with bootstrapping

I move on to Ridge Regression, where I will look at how the ridge regression performs in terms of MSE across 100 different λs, both with and without bootstrapping, and where the model with bootstrapping is the main regression of interest for run time.

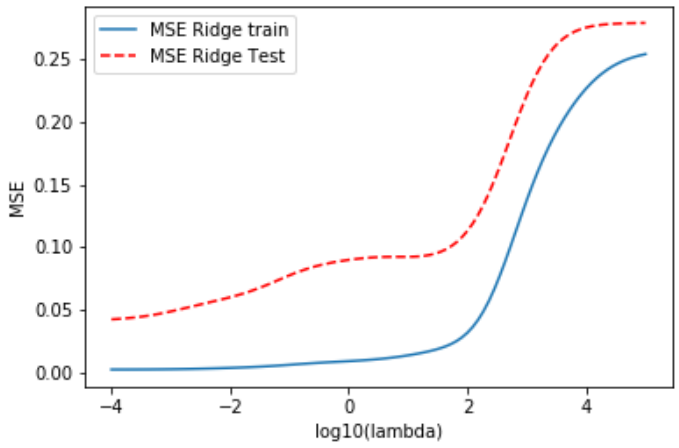

Figure 7: MSE for ridge without bootstraps, n $\lambda = 100$

Encouragingly, the Ridge regression responds very well to bootstrapping for all values of λ, and the MSE when running 100 bootstraps for small penalty terms is close to zero. As predicted by the theory, the plots show that the reason for the

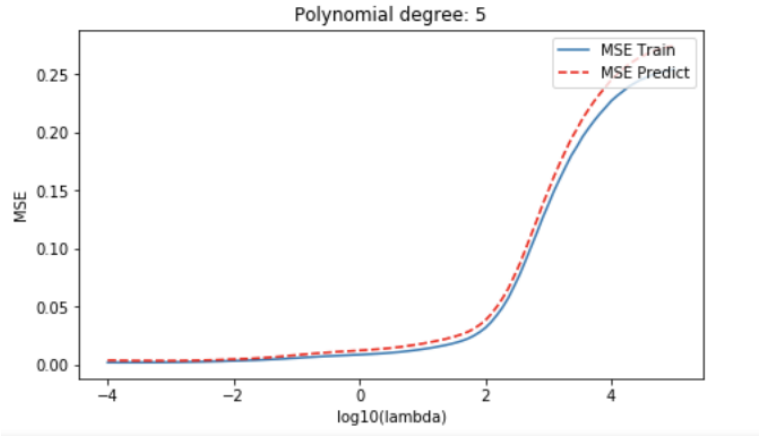

Figure 8: MSE for ridge with 100 bootstraps, n $\lambda = 100$

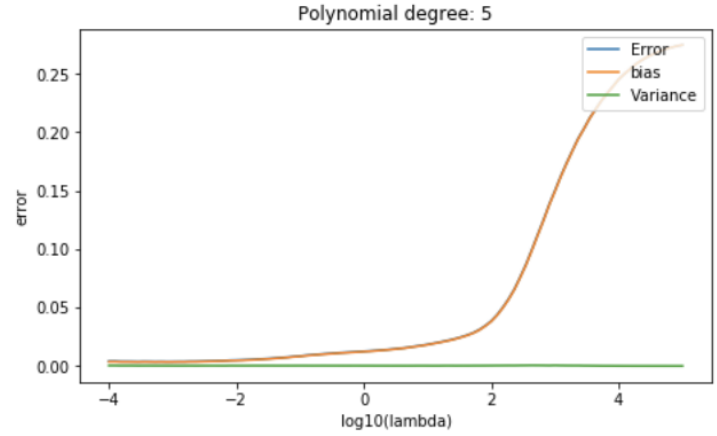

Figure 9: Bias/variance for ridge with 100 bootstraps, n $\lambda = 100$

relative poor performance when increasing the value of λ is due to underfitting, in other words, the penalty terms reduce the influence of the parameters on the fit to the point where their influence almost disappear. The run time, however, is 8.8211 seconds, 5.500 times longer than the simple OLS regression. Regardless of that however, the fit is significantly better and the improvement is worth it, especially when data sets are moderately large and smaller.

3.1.5 Lasso Regression with Cross Validation

Lastly, we'll take a look at the MSE and bias/variance from the lasso regression, also using 5 cross-validations for each combination of 100 values of λ and 100 bootstraps.

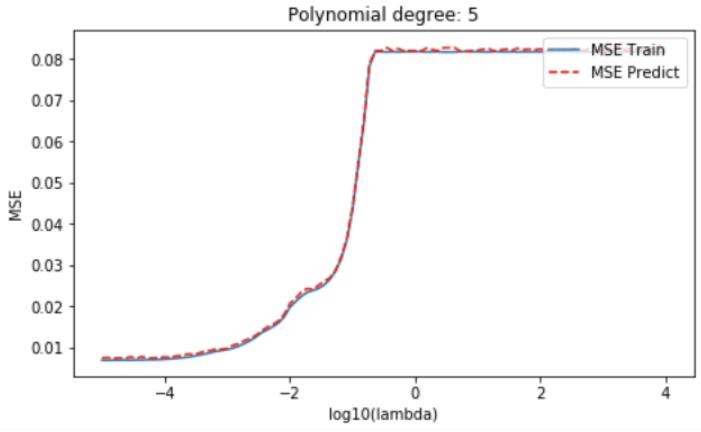

Figure 10: MSE for lasso with 100 bootstraps, n $\lambda = 100$, k = 5

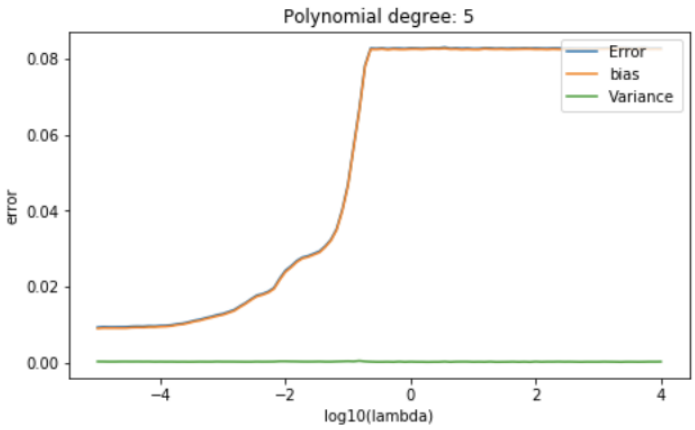

Figure 11: Bias/variance for lasso with 100 bootstraps, n $\lambda = 100$, k = 5

The results show that the fit is similarly strong to ridge for very small values of λ, when using lasso regression, but that it is more sensitive to larger values of *lambda*.

From moderately small values of λ, the influence of the parameters are reduced to zero, and the MSE and the bias/variance both stabilizes and flattens. One large issue here is run time (46.6782 seconds, more than 5 times larger than the ridge regression, consistent with k = 5). This shows that fine tuning the parameter is very important when running lasso regressions, both to get a usable result but also as an efficiency measure.

3.2 Map data

The map data set has, as previously mentioned, been downloaded from Earth Explorer (2020).

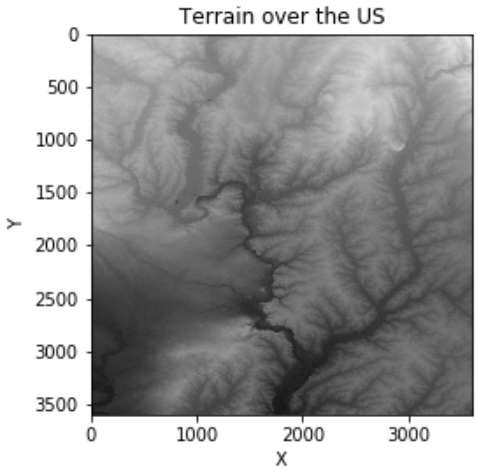

Figure 12: Terrain plot original data set

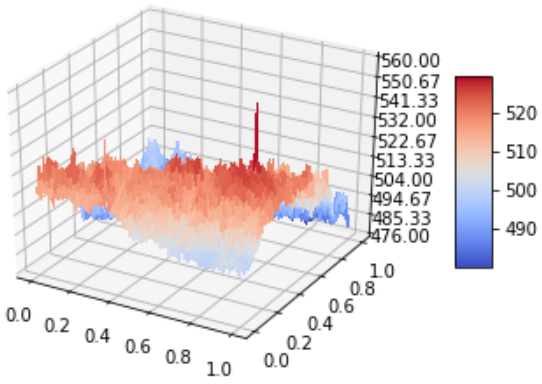

Figure 13: Topology plot 400 × 400 area Arizona

The original data set is a 3601 $\times$ 3601 area plot but as a simplification I've only chosen to regress the 400 400 area in the top left corner. As can be seen from the 3d-plot, the outline is considerably more noisy than the output from Franke's function, and is therefore unlikely to be fitted as well for any linear regression specification. The large spike, which may represent a tower or a simple data error is something linear regression functions are particularly unlikely to predict, and it is therefore going to represent an outlier, which will be punished by the MSE measurement. To not increase computation time too much, the number of bootstraps and λs have been decreased to 20, while keeping the number of cross-validations at $k = 5$.

3.2.1 Ordinary Least Squares

Similarly as in the case of Franke's Function, I regress a simple OLS and present similar results, where the measurement of interest is the MSE.

	Train	Test
MSE	16.1360	15.9129
R^2	0.9080	0.9098
Run time	0.1140s	

The train and test data fits the plane reasonably well, given the very noisy surface, and the R^2 value is predictably worse than in the Franke's function, as a result of not catching all the noise. The test and train data perform similarly well for both the train and test set. The run time is increased by $\times \sim 70$, which is dramatic given that n only has increased fourfold.

The map prediction shows that OLS capture the general trend in the data quite well, but that the leftmost peak is way too large, which indicates that using a polynomial degree of five is underfitting the data.

3.2.2 Ridge Regression with bootstrapping

The zero variance for the Ridge regression with bootstrap again suggests that the plot is underfitted. Interestingly, the Ridge Regression does not seem to outperform

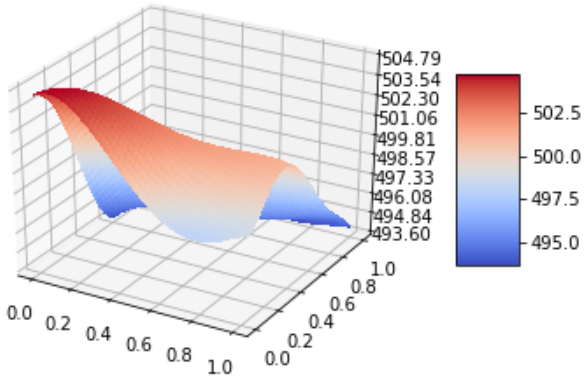

Figure 14: Predicted map plot, OLS

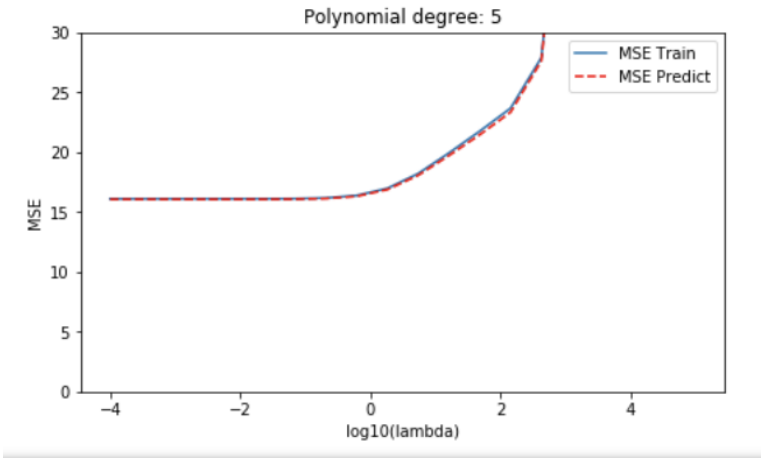

Figure 15: MSE plot, Ridge Regression, map data, n boostraps = 20, n λs = 20

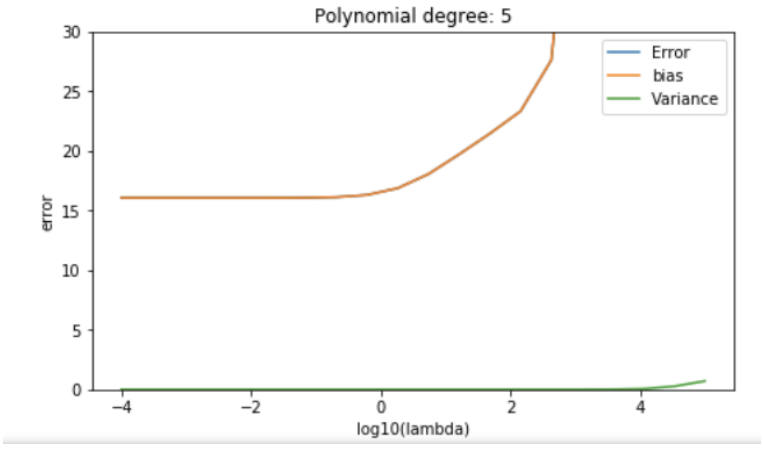

Figure 16: Bias/variance plot, Ridge Regression, map data, n bootstraps = 20, n λs = 20

OLS for any value of λ, which is of interest because the run time has increased more than 100 times to 131.9789 seconds. The bootstrapping may increase robustness of the test, but the large increase in run time may be impractical for some applications. For large values of λ, the MSE and bias explodes, and the plot y-axes has therefore been adjusted so that the plots are readable.

3.2.3 Lasso Regression with Cross Validation

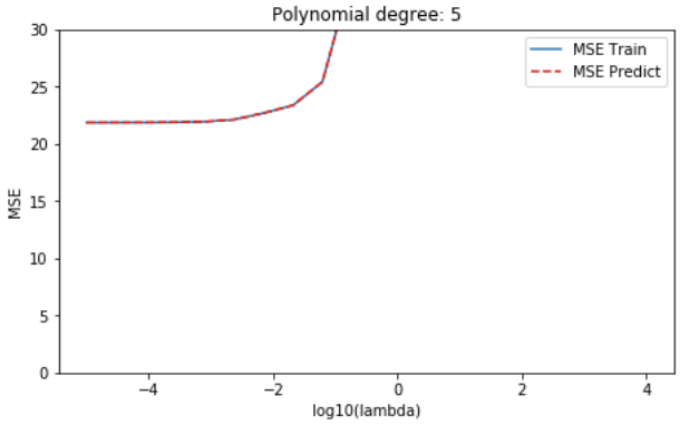

Figure 17: MSE plot, Lasso Regression, map data, n bootstraps = 20, n λs = 20, k = 5

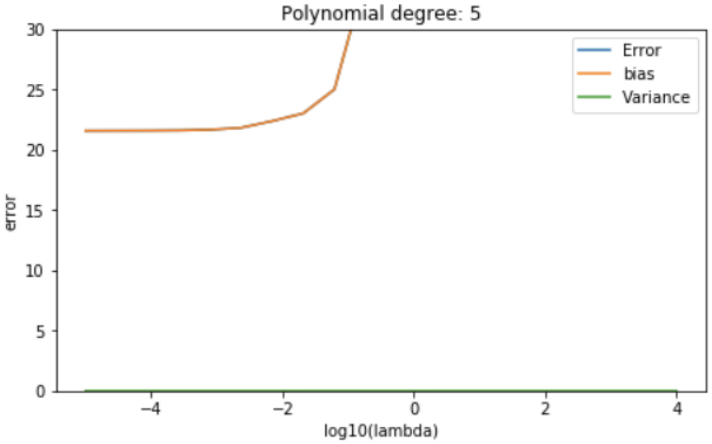

Figure 18: Bias/variance plot, Lasso Regression, map data, n bootstraps = 20, n λs = 20, k = 5

The results from the lasso regression show that Lasso perform considerably worse than both OLS and Ridge regression for all vaues of λ. This may in part be the result of introducing cross validation, which increases the robustness of the results.

Similarly to in the case of Franke's function, the Lasso Regression is more sensitive to the selection of λ than the Ridge Regression and a smaller value must be chosen to not shrink the parameter influence too much. The run time has also increased relative to the Ridge Regression, taking 72 minutes to run, which is ~ 32 times more than the Ridge Regression.

4 Conclusion

The results of this analysis is meant to be an indicator of relevant performance between three regression methods with three different sets of resampling methods, and how it affects both prediction accuracy and run time. Introducing resampling and several tuning parameters for Ridge and Lasso regressiions unsurprisingly increases algorithm run time considerably, especially for large data sets. The results, however are not that much better than for a simple OLS. The Ridge Regression performs slightly better on a small, smoothed data set, but the OLS performs best on a large and noisy data set. This means that for simple analysis, and in the absence of running more complex methods, like SVD or principal component analysis, OLS may be a sufficiently good method, especially on large data sets, especially when run time efficiency and computing power is an issue. If Ridge or Lasso is the methodology of choice, the analysis show the importance of choosing the correct tuning parameter. A too large tuning parameter λ drastically worsens the results of the analysis and is also counter productive for efficiency.

References

Hjort-Jensen, M. (Retrieved: 09.10.2020). Data analysis and machine learning: Linear regression and more advanced regression analysis. *University of Oslo, FYS-STK4155*.

USGS (Accessed: 01.10.2020). Earth explorer. *https://earthexplorer.usgs.gov*.